Содержание

Вступление от переводчика
STM32F10xxx Cortex-M3 programming manual (Руководство программиста)
STM32F105_107_Reference manual (Справочное руководство)
Errata_stm32f107 (Список ошибок семейства **CL**)

Вступление от переводчика

Как появился и кому предназначен этот перевод
Что стоит в первую очередь почитать об **ARM/STM32**
Особенности оформления этого перевода

Как появился и кому предназначен этот перевод

Сам я программист микроконтроллеров, работал, в основном, на **AVR**. Был один приличный проект на **TMS320VC5509a**. Теперь появилась необходимость поработать с **ARM**, а точнее с камнем **STM32F107**.

К сожалению, я недостаточно хорошо знаю английский, чтобы свободно читать документацию. Всякие обзоры еще могу "полистать" и уловить общую суть. А вот когда изучаешь большой документ, типа **Refery Manual** или **Programming Manual**, то ловишь себя на том, что одновременно точно переводить все детали и запоминать все это, как-то не получается. Где-то на пятой странице забываешь о том, что было на первой. Поэтому выработал для себя такой способ. Те разделы документации, к которым предполагается неоднократное обращение, сначала перевожу, а потом уж, досконально изучаю на "ридной мове". Конечно, это очень медленно и мое руководство не в восторге, но по другому у меня не получается.

Поэтому, это документ я оформлял, в первую очередь, для себя любимого. Но, если кто-то найдет этот перевод полезным для себя, и сэкономит на нем кучу времени, я буду только рад. При этом честно предупреждаю, что не могу гарантировать отсутствие ошибок, связанных либо с недостаточным знанием языка, либо с недостаточным пониманием технической сути. Вряд ли такой перевод заинтересует программера, хорошо владеющим английским. Однако, я думаю, что найдется немало людей, которые тоже только начали осваивать **ARM** и владеют языком примерно на моем уровне, или даже еще ниже.

Весь парадокс ситуации с переводами подобной технической документации заключается в том, что вряд ли в обозримом будущем мы увидим профессиональный перевод. Профессиональные переводчика с английского, как правило, не владеют предметом перевода, и не беруться за подобную работу (а если и беруться, то ничего путного из этого не выходит, плавали, знаем). Профессиональным программистам, хорошо владеющим и языком и предметом, это, во первых, неинтересно, а во вторых, они искренне считают, что это никому не нужно, мол, проще и быстрее изучить язык, чем заниматься этой фигней. Поэтому, спасение утопающих (в море информации) - дело рук самих утопающих.

Что стоит в первую очередь почитать об ARM/STM32

Я довольно много времени потерял на то, чтобы определиться, что именно стоит читать в первую очередь. Позволю себе высказать то, что у меня получилось. Опять же, может это позволит кому-то сэкономить драгоценное время. Но данные советы касаются только тех, кто собирается изучать документацию по **STM32F1xx**. Для камней других производителей эти советы могут оказаться некорректными.

1. Начинать стоит с какого-то обзорного материала по архитектуре, типа файла STM32_MCU_family.pdf (10 стр.). На русском языке есть неплохой документ "***Ознакомительное руководство по ARM-микроконтроллерам Cortex-M3***", который в разных вариациях присутствует на многих сайтах.

2. По идее, нодо бы почитать документы от фирмы **ARM**, которые можно найти на ее родном сайте. Однако, во первых, там нативный английский, который заметно сложнее производных языков, типа "Индийского английского" или "Норвежского английского". Во вторых, по крайней мере это касается документации от **ST**, все что надо повторяется в документах на конкретное изделие. Так что "продираться" сквозь эти документы стоит только в исключительных случаях, когда явно не хватает информации во вторичных документах.

3. Затем надо изучить файл ***"STM32F10xxx Cortex-M3 programming manual.pdf"*** (137 стр.). В нем кратко описаны основы архитектуры ядра **ARM**, причем только те аспекты, что важны для изучения **STM32F1xx**.
Здесь есть мой перевод большей части этого документа (кроме описания инструкций).

4. Теперь можно просмотреть файл **datasheet** на свой контроллер. В моем случае это файл "***STM32F105_107xx.pdf***" (93 стр.). Нет особой необходимости в переводе этого файла. Он используется как краткий справочник по возможностям камня, и здесь достаточно самого минимального знания английского.

5. Затем надо изучить файл ***"STM32F105_107_Reference manual.pdf"*** (995 стр.). Здесь есть мой перевод заметной части этого документа.

6. И наконец, надо обязательно почитать список ошибок на свой камень, в моем случае это файл "***Errata_stm32f107.pdf***" (26 стр.). В нем кратко описаны те ошибки ядра **ARM**, что важны для **STM32F1xx**, а также собственные ошибки **ST**. Ничего "страшного" я в нем не нашел, но знать его содержимое, для спокойствия души, надо. Здесь есть мой перевод этого документа.

Это, на мой взгляд, тот кандидатский миниму, что необходим для более или менее уверенной ориентации в теме при программировании **STM32F10xxx**.

Но это только вершина айсберга. Существует куча другой документации, и, в первую очередь, это "Рекомендации по применению" (**AppNote**). Но это узко-специализированная документация, и ее не стоит читать всю подряд для общего развития, ее слишком много. Здесь нужно "симтоматическое лечение": есть проблема, которую не решить с помощью документов общего направления, тогда ищем нужный документ, читаем и решаем проблему. Искать подобные документы нужно, в первую очередь, на сайте **ST**, есть также неплохая подборка ссылок здесь.

Особенности оформления этого перевода

Здесь, в одном документе, находятся переводы трех документов с сайта фирмы **ST**:

- Частичный перевод файла **"STM32F10xxx Cortex-M3 programming manual.PDF"** ревизия 2 от 18.01.2010.
- Частичный перевод файла **"STM32F105_107_Reference manual.pdf"** ревизия 9 от 14.09.2009.
- Полный перевод файла **"Errata_stm32f107.pdf"** ревизия 3 от 14.01.2010.

Я сознательно свалил в кучу все эти документы. Конечно, в таком виде браузер заметно подтормаживает. Однако, возможность быстро найти все упоминания по какому-то узлу, регистру, именованному биту, на мой взгляд, заметно перевешивают этот недостаток (конечно, если смотрим в **OffLine**).

Практически весь оригинальный текст сохранен, что бы при малейшем сомнении в правильности перевода можно было быстро справиться с оригиналом.!!!

Таким фоном выделен текст перевода.
Таким фоном выделены примечания, недоумения и прочая отсебятина переводчика.
Таким фоном выделены те места, которые переводчик "неасилил".

Так как, вследствие изменения формата документа с **PDF** на **HTML**, была утеряна панель навигации, то, в целях компенсации, была использована система иерархических содержаний, по которым можно спускаться вглубь документа и возвращаться назад.

1 STM32F10xxx Cortex-M3 programming manual
Руководство программиста

1 About this document (Об этом документе)
2 The Cortex-M3 processor (Процессор **Cortex-M3**)
3 The Cortex-M3 instruction set (Набор инструкций ядра **Cortex-M3** Пока нет перевода)
4 Core peripherals (Периферия ядра)
5 Revision history (История изменений)

This programming manual provides information for application and system-level software developers. It gives a full description of the STM32F10xxx Cortex™-M3 processor programming model, instruction set and core peripherals.
Это руководство по программированию предоставляет информацию для разработчиков приложений и программного обеспечения системного уровня. Оно дает полное описание модели программирования, набора инструкций и периферии ядра процессора **STM32F10xxx Cortex-M3**.

The STM32F10xxx Cortex™-M3 processor is a high performance 32-bit processor designed for the microcontroller market. It offers significant benefits to developers, including:
Устройства **STM32F10xxx Cortex-M3** являются высокоэффективными 32-х разрядными процессорами, спроектированными для рынка микроконтроллеров. Они предоставляют существенные преимущества разработчикам, включая:

- Outstanding processing performance combined with fast interrupt handling
Выдающуюся производительность в комбинации с быстрой обработкой прерываний
- Enhanced system debug with extensive breakpoint and trace capabilities
Усовершенствованную систему отладки с обширными возможностями установки контрольных точек и трассировки
- Efficient processor core, system and memories
Эффективные ядро процессора, систему и память
- Ultra-low power consumption with integrated sleep modes
Ультрамалое потребление мощности с встроенными режимами сна
- Platform security (Безопасную платформу)

Здесь, в оригинальном тексте, находится полное содержание документа, но, как уже упоминалось в предисловии, оно было переделано в систему иерархических содержаний. Также здесь, в оригинальном тексте, находится перечень таблиц и перечень рисунков. Я ни разу не пользовался подобным сервисом, поэтому счел возможным опустить их.

1 About this document (Об этом документе)

1.1 Typographical conventions (Типографские соглашения)
1.2 List of abbreviations for registers (Список сокращений для регистров)
1.3 About the STM32 Cortex-M3 processor and core peripherals
(О процессоре **STM32 Cortex-M3** и периферии ядра)

This document provides the information required for application and system-level software development. It does not provide information on debug components, features, or operation. This material is for microcontroller software and hardware engineers, including those who have no experience of ARM products.
Этот документ предоставляет информацию, необходимую для разработки приложений и программного обеспечения системного уровня. Он не предоставляет информацию о компонентах, особенностях или операциях отладки. Это материал для программистов и разработчиков оборудования на базе микроконтроллеров, включая тех из них, у кого нет никакого опыта работы с **ARM** продуктами.

1.1 Typographical conventions (Типографские соглашения)

The typographical conventions used in this document are:
В этом документе используются следующие типографские соглашения:

italic *Курсив*	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations. Выделяет важные примечания, вводит специальную терминологию, обозначает внутренние перекрестные ссылки и цитаты.
< and >	Enclose replaceable terms for assembler syntax where they appear in code or code fragments. For example: LDRSB<cond> <Rt>, [<Rn>, #<offset>] Элементы, заключеные в угловые скобки, должны замененяться реальными значениями, согласно синтаксису ассемблера там, где они появляются в коде или его фрагментах. Например: `LDRSB<cond> <Rt>, [<Rn>, #<offset>]`

1.2 List of abbreviations for registers (Список сокращений для регистров)

The following abbreviations are used in register descriptions:
Следующие сокращения используются при описании регистров:

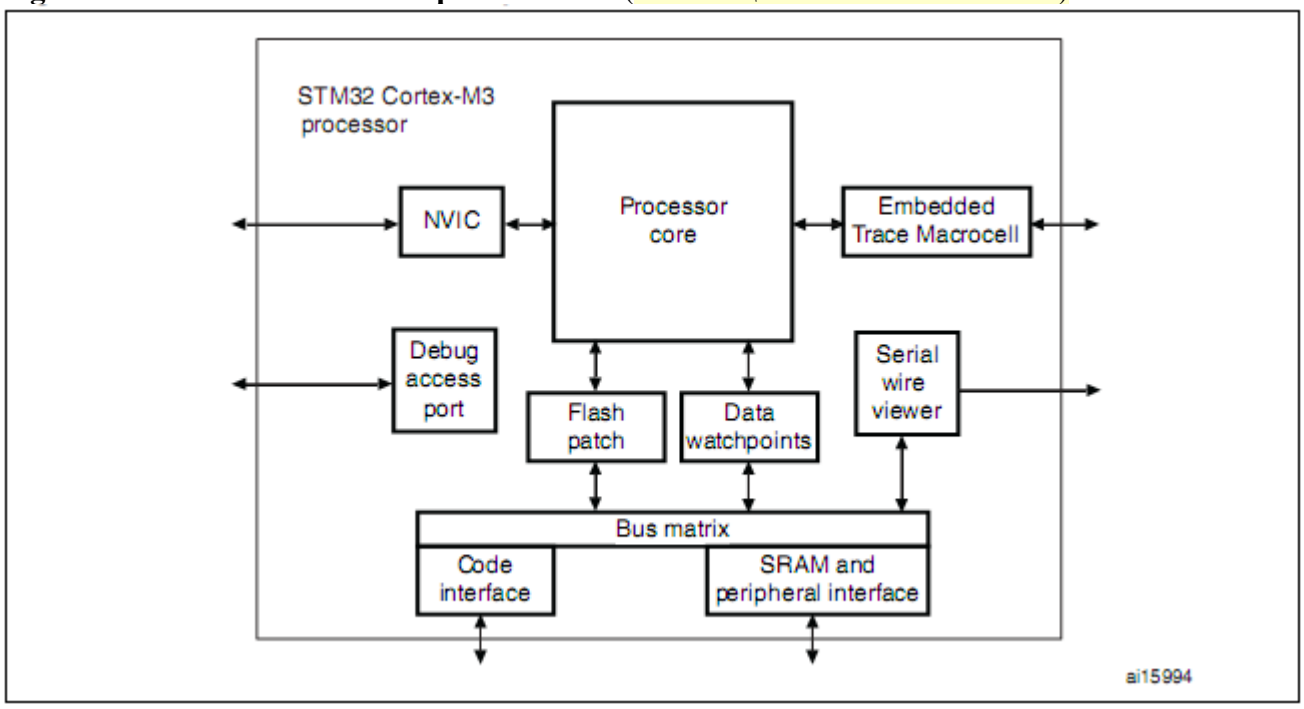
read/write (rw)	Software can read and write to these bits. Программа может читать и записывать эти биты.
read-only (r)	Software can only read these bits. Программа может только читать эти биты.
write-only (w)	Software can only write to this bit. Reading the bit returns the reset value. Программа может только записывать эти биты. При чтении бита возвращается его значение после сброса.
read/clear (rc_w1)	Software can read as well as clear this bit by writing 1. Writing ‘0’ has no effect on the bit value. Программа может читать этот бит, а так же очищать его записью '1'. Запись '0' не имеет эффекта.
read/clear (rc_w0)	Software can read as well as clear this bit by writing 0. Writing ‘1’ has no effect on the bit value. Программа может читать этот бит, а так же очищать его записью '0'. Запись '1' не имеет эффекта.
toggle (t)	Software can only toggle this bit by writing ‘1’. Writing ‘0’ has no effect. Программа может только переключать этот бит записью в него '1'. Запись '0' не имеет эффекта.
Reserved (Res.)	Reserved bit, must be kept at reset value. Зарезервированный бит, должен удерживать то значение, что было получено после сброса.

1.3 About the STM32 Cortex-M3 processor and core peripherals
О процессоре STM32 Cortex-M3 и периферии ядра

1.3.1 System level interface (Системный уровень интерфейса)
1.3.2 Integrated configurable debug (Интегрированный конфигурируемый отладчик)
1.3.3 Cortex-M3 processor features and benefits summary
Особенности процессора **Cortex-M3** и резюме преимуществ
1.3.4 Cortex-M3 core peripherals (Периферия ядра **Cortex-M3**)

The Cortex-M3 processor is built on a high-performance processor core, with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. The processor delivers exceptional power efficiency through an efficient instruction set and extensively optimized design, providing high-end processing hardware including single- cycle 32x32 multiplication and dedicated hardware division.
Процессор **Cortex-M3** построен на высокоэффективном процессорном ядре, с 3-х этапным конвейером Гарвардской архитектуры, что делает его идеальным для встраиваемых приложений. Процессор предоставляет исключительную эффективность потребляемой мощности за счет эффективного набора инструкций и экстенсивно оптимизированного дизайна, обеспечивая требования высокопроизводительных устройств, включая модуль однотактного умножителя 32x32 и встроенный аппаратный делитель.

Figure 1. STM32 Cortex-M3 implementation (Реализация **STM32 Cortex-M3**)



To facilitate the design of cost-sensitive devices, the Cortex-M3 processor implements tightly-coupled system components that reduce processor area while significantly improving interrupt handling and system debug capabilities. The Cortex-M3 processor implements a version of the Thumb® instruction set, ensuring high code density and reduced program memory requirements. The Cortex-M3 instruction set provides the exceptional performance expected of a modern 32-bit architecture, with the high code density of 8-bit and 16-bit microcontrollers.
Чтобы облегчить разработку устройств, чувствительных к стоимости, процессор **Cortex-M3** включает в себя компоненты, тесно связанные с системой, что уменьшает общую площадь микроконтроллера, и в то же время, значительно улучшает обработку прерываний и возможности отладки системы. Процессор **Cortex-M3** реализует версию набора команд **Thumb**, гарантируя

высокую плотность кода и уменьшая потребность программ в памяти. Набор команд **Cortex-M3** обеспечивает исключительную производительность, ожидаемую для современной 32-х разрядной архитектуры, с высокой плотностью кода, характерных для 8-ми и 16-ти разрядных микроконтроллеров.

The Cortex-M3 processor closely integrates a configurable nested interrupt controller (NVIC), to deliver industry-leading interrupt performance. The NVIC includes a non-maskable interrupt (NMI), and provides up to 256 interrupt priority levels. The tight integration of the processor core and NVIC provides fast execution of interrupt service routines (ISRs), dramatically reducing the interrupt latency. This is achieved through the hardware stacking of registers, and the ability to suspend load-multiple and tore-multiple operations. Interrupt handlers do not require any assembler stubs, removing any code overhead from the ISRs. Tail-chaining optimization also significantly reduces the overhead when switching from one ISR to another.
Процессор **Cortex-M3** тесно интегрирован с контроллером вложенных прерываний (**NVIC**), что дает лидирующую в отрасли производительность обработки прерываний. **NVIC** включает немаскируемое прерывание (**NMI**), и обеспечивает до 256 уровней приоритета прерываний. Тесная интеграция ядра процессора и **NVIC** обеспечивает быстрое обслуживание обработчиков прерываний (**ISRs**), заметно уменьшая время ожидания для прерывания. Это достигается за счет аппаратного стека регистров, и способности приостанавливать операции множественной загрузки/сохранения. Обработчики прерываний не требуют никаких ассемблерных прологов и эпилогов, что удаляет любой дополнительный код из **ISR**. Возможность процедуры "посадки на хвост" предыдущему прерыванию также значительно уменьшает необходимость в дополнительных инструкциях, при переключении с одного обработчика прерывания на другое.

To optimize low-power designs, the NVIC integrates with the sleep modes, that include a deep sleep function that enables the STM32 to enter STOP or STDBY mode.
Чтобы оптимизировать устройства с низким энергопотреблением, **NVIC** интегрируется с режимами сна, что включает режим глубокого сна, который дает возможность **STM32** входить в режим **STOP** или **STDBY**.

1.3.1 System level interface (Системный уровень интерфейса)

The Cortex-M3 processor provides multiple interfaces using AMBA® technology to provide high speed, low latency memory accesses. It supports unaligned data accesses and implements atomic bit manipulation that enables faster peripheral controls, system spinlocks and thread-safe Boolean data handling.
Процессор **Cortex-M3** обеспечивает множество интерфейсов, используя технологию **AMBA technology**, чтобы обеспечить высокую скорость, и низкую задержку доступа к памяти. Он поддерживает невыровненный доступ к данным и атомарные побитовые манипуляции, которые позволяют ускорить управление периферией, system spinlocks и безопасную обработку Булевых данных, при работе в мульти-тредовом режиме.

1.3.2 Integrated configurable debug
Интегрированный конфигурируемый отладчик

The Cortex-M3 processor implements a complete hardware debug solution. This provides high system visibility of the processor and memory through either a traditional JTAG port or a 2-pin Serial Wire Debug (SWD) port that is ideal for small package devices.
Процессор **Cortex-M3** предоставляет законченное аппаратное решение для отладки. Это обеспечивает высокую видимость процессора и памяти с помощью традиционного порта **JTAG** или через 2-х проводный последовательный порт отладки **SWD** (*Serial Wire Debug*), который идеален для устройств в мало-выводных корпусах.

For system trace the processor integrates an Instrumentation Trace Macrocell (ITM) alongside data watchpoints and a profiling unit. To enable simple and cost-effective profiling of the system events these generate, a Serial Wire Viewer (SWV) can export a stream of software-generated messages, data trace, and profiling information through a single pin.
Для трассировки системы в процессор интегрирован модуль **ITM** (*Instrumentation Trace Macrocell*) наряду с модулями **watchpoints** и профилирования. Чтобы обеспечить простое и дешевое профилирование системных событий, которые они генерируют, модуль **SWV** (*Serial Wire Viewer*) может экспортировать поток программно сгенерированных сообщений, трассировки данных, и информацию для профайлера через единственный вывод.

The optional Embedded Trace Macrocell™ (ETM) delivers unrivalled instruction trace capture in an area far smaller than traditional trace units, enabling many low cost MCUs to implement full instruction trace for the first time.
Впервые, опционный модуль **ETM** обеспечивает непревзойденный захват трассировки инструкций в области, намного меньшей, чем дают традиционные модули трассировки, давая возможность многим дешевым микропроцессорам осуществить полную трассировку инструкций.
Чувствую, здесь сильно напортачил.

1.3.3 Cortex-M3 processor features and benefits summary
Особенности процессора Cortex-M3 и резюме преимуществ

- Tight integration of system peripherals reduces area and development costs

Тесная интеграция системной периферии уменьшает стоимость разработки и площадь кристалла

- Thumb instruction set combines high code density with 32-bit performance

Набор команд **Thumb** дает хорошую комбинацию высокой плотности кода и производительности 32-х разрядных систем

- Code-patch ability for ROM system updates

Способность исправления и обновления кода для **ROM based** систем *Flash вместо ROM*

- Power control optimization of system components

Оптимизированное управление потреблением системных компонентов

- Integrated sleep modes for low power consumption

Интегрированные режимы сна для устройств с низким потреблением

- Fast code execution permits slower processor clock or increases sleep mode time

Быстрое выполнение кода позволяет уменьшить время работы процессора, а значит увеличить время нахождения в режиме сна

- Hardware division and fast multiplier

Аппаратный делитель и быстрый модуль умножения

- Deterministic, high-performance interrupt handling for time-critical applications

Детерминированная, высокоэффективная обработка прерываний для приложений, критичных к времени выполнения

- Extensive debug and trace capabilities:

Обширные возможности по отладке и трассировке:

- Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging and tracing. Порт **SWD** (*Serial Wire Debug*) и модуль **SWT** (*Serial Wire Trace*) сокращают число выводов, требуемых для отладки и трассировки.

1.3.4 Cortex-M3 core peripherals (Периферия ядра Cortex-M3)

These are: (Периферией ядра являются:)

Nested vectored interrupt controller (Контроллер вложенных векторов прерываний)

The nested vectored interrupt controller (NVIC) is an embedded interrupt controller that supports low latency interrupt processing.
Контроллер вложенных векторов прерываний (**NVIC**) является встроенным контроллером прерываний, который поддерживает обработку прерываний с низким времени ожидания.

System control block (Блок управления системой)

The system control block (SCB) is the programmers model interface to the processor. It provides system implementation information and system control, including configuration, control, and reporting of system exceptions.
Блок управления системой (**SCB**) является интерфейсом для программирования процессора. Он предоставляет информацию о реализации системы и возможность управления системой, включая ее конфигурацию, управление, и получение сообщений о системных исключениях.

System timer (Системный таймер)

The system timer, SysTick, is a 24-bit count-down timer. Use this as a Real Time Operating System (RTOS) tick timer or as a simple counter.
Системный таймер **SysTick**, является 24-х битовым таймером обратного отсчета. Используйте его в качестве **tick**-таймера операционной системы реального времени (**RTOS**), или как простой счетчик.

2 The Cortex-M3 processor (Процессор Cortex-M3)

2.1 Programmers model (Модель программирования)
2.2 Memory model (Модель памяти)
2.3 Exception model (Модель исключений)
2.4 Fault handling (Обработка ошибок)
2.5 Power management (Управление потреблением)

2.1 Programmers model (Модель программирования)

2.1.1 Processor mode and privilege levels for software execution
Режимы процессора и уровни привилегий для программного кода
2.1.2 Stacks (Стеки)
2.1.3 Core registers (Регистры ядра)
2.1.4 Exceptions and interrupts (Исключения и прерывания)
2.1.5 Data types (Типы данных)
2.1.6 The Cortex microcontroller software interface standard (CMSIS)
Стандартный программный интерфейс к микроконтроллеру **Cortex** (**CMSIS**)

This section describes the Cortex-M3 programmers model. In addition to the individual core register descriptions, it contains information about the processor modes and privilege levels for software execution and stacks.
Этот раздел описывает модель программирования **Cortex-M3**. В дополнение к описаниям индивидуальных регистров ядра, он содержит информацию о режимах процессора и уровнях привилегий для программного кода и стеков.

2.1.1 Processor mode and privilege levels for software execution
Режимы процессора и уровни привилегий для программного кода

The processor modes are: (У процессора есть следующие режимы:)

Thread mode	Used to execute application software. The processor enters Thread mode when it comes out of reset. Используется для выполнения кода приложения. Процессор входит в режим **Thread**, когда выходит из сброса.
Handler mode	Used to handle exceptions. The processor returns to Thread mode when it has finished exception processing. Используется для обработки исключений. Процессор возвращается в режим **Thread**, когда заканчивает обработку исключения.

The privilege levels for software execution are: Уровни привилегий для программного кода:

Unprivileged The software: (Код программы:)

- Has limited access to the MSR and MRS instructions, and cannot use the CPS instruction
  Имеет ограниченный доступ к инструкциям **MSR** и **MRS**, и не может использовать инструкцию **CPS**

- Cannot access the system timer, NVIC, or system control block
  Не может обратиться к регистрам системного таймера, **NVIC**, или **SCB**
- Might have restricted access to memory or peripherals.
  Возможно, имеет ограниченный доступ к памяти или периферии.

	Unprivileged software executes at the unprivileged level. Непривилегированный программный код выполняется на непривилегированном уровне.
Privileged	The software can use all the instructions and has access to all resources. Код программы может использовать все инструкции и имеет доступ ко всем ресурсам. Privileged software executes at the privileged level. Привилегированный программный код выполняется на привилегированном уровне.

In Thread mode, the CONTROL register controls whether software execution is privileged or unprivileged, see *CONTROL register on page 20*. In Handler mode, software execution is always privileged.
В режиме **Thread** регистр **CONTROL** управляет, будет ли код привилегированным или непривилегированным, см. параграф "Регистр **CONTROL**" в разделе 2.1.3. В режиме **Handler** всегда дают привилегию программному коду.

Only privileged software can write to the CONTROL register to change the privilege level for software execution in Thread mode. Unprivileged software can use the SVC instruction to make a supervisor call to transfer control to privileged software.
Только привилегированный код может писать в регистр **CONTROL**, чтобы изменить уровень привилегий для программного кода режима **Thread**. Непривилегированный программный код может использовать инструкцию **SVC**, чтобы вазвать супервизор, который передаст управление привилегированному коду.

2.1.2 Stacks (Стеки)

The processor uses a full descending stack. This means the stack pointer indicates the last stacked item on the stack memory. When the processor pushes a new item onto the stack, it decrements the stack pointer and then writes the item to the new memory location. The processor implements two stacks, the main stack and the process stack, with independent copies of the stack pointer, see *Stack pointer on page 14*. Процессор использует полный нисходящий стек. Это означает, что указатель вершины стека указывает на последний сохраненный в стеке элемент в памяти стека. Когда процессор помещает новый элемент на стек, он декрементирует указатель вершины стека и затем пишет этот элемент на новое место в памяти. Процессор поддерживает два стека, **main stack** и **process stack**, с независимыми копиями указателя вершины стека, см. параграф "Указатель стека" в разделе 2.3.1.

In Thread mode, the CONTROL register controls whether the processor uses the main stack or the process stack, see *CONTROL register on page 20*. In Handler mode, the processor always uses the main stack. The options for processor operations are:
В режиме **Thread** регистр **CONTROL** управляет, будет ли процессор использовать **main stack** или **process stack**, см. параграф "Регистр **CONTROL**" в разделе 2.1.3. В режиме **Handler** процессор всегда использует **main stack**. Для операций процессора возможны следующие варианты:

Table 1. Summary of processor mode, execution privilege level, and stack use options
Режимы процессора, уровни привилегий выполнения, и стек: варианты использования

Processor mode	Used to execute	Privilege level for software execution	Stack used
Thread	Applications	Privileged or unprivileged(1)	Main stack or process stack(1)
Handler	Exception handlers	Always privileged	Main stack

1. See *CONTROL register on page 20*. (См. параграф "Регистр **CONTROL**" в разделе 2.1.3)

2.1.3 Core registers (**Регистры ядра**)

General-purpose registers (Регистры общего назначения)
Stack pointer (Указатель стека)
Link register (Регистр связи)
Program counter (Счетчик программы)
Program status register (Регистр статуса программы)
Application program status register (Регистр статуса приложения)
Interrupt program status register (Регистр статуса прерываний)
Execution program status register (Регистр статуса исключений)
Interruptible-continuable instructions
(Прерываемые инструкции работы с блоком данных)
If-Then block (Блок инструкций для **If-Then**)
Exception mask registers (Регистры масок исключений)
Priority mask register (Регистры масок приоритетов)
Fault mask register (Регистр масок ошибок)
Base priority mask register (Регистр масок базы приоритетов)
CONTROL register (Регистр **CONTROL**)

Figure 2. Processor core registers (Регистры ядра процессора)

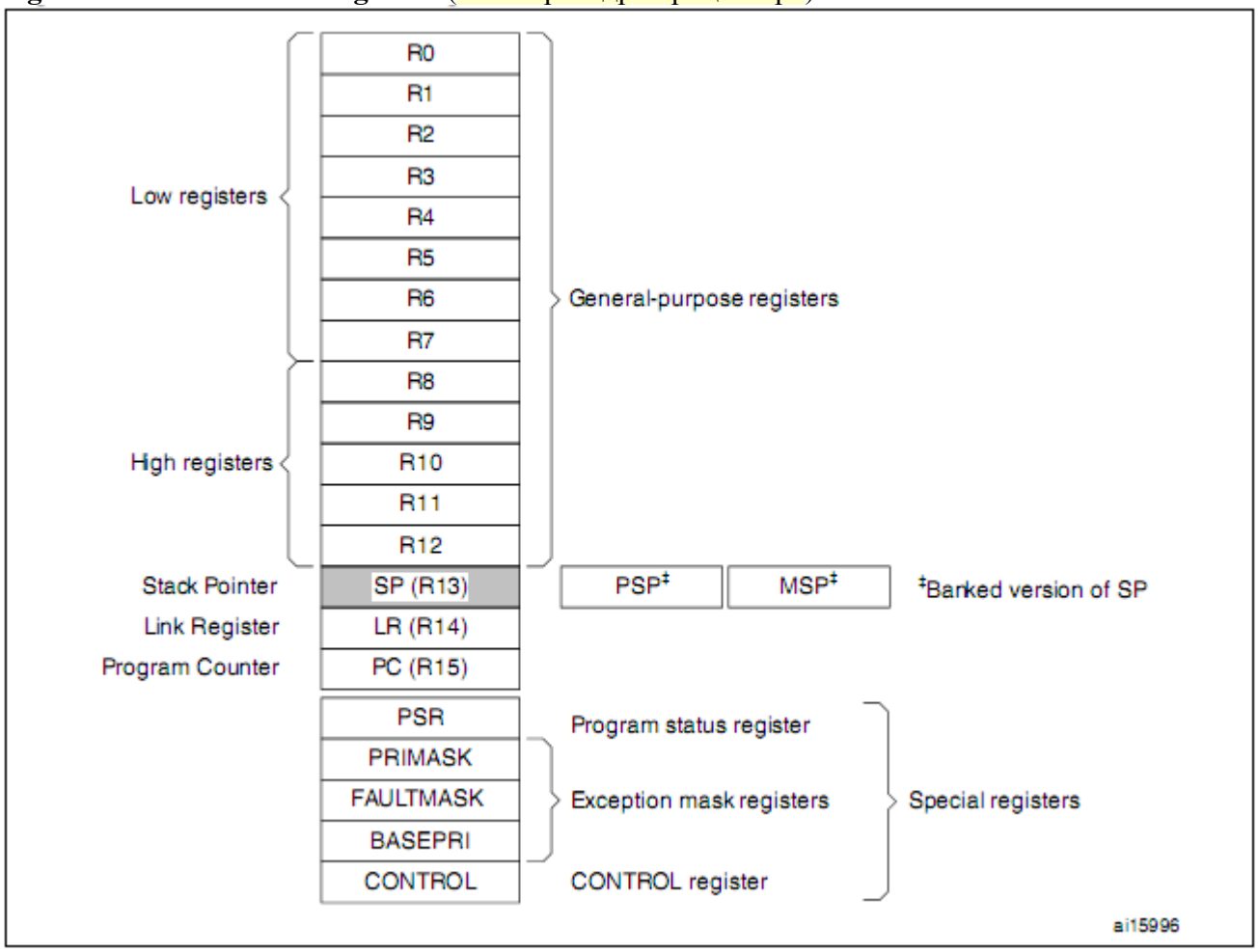


Table 2. Core register set summary (Перечень регистров ядра)

Name	Type[1]	Required privilege[2]	Reset value	Description
R0-R12	read-write	Either	Unknown	*General-purpose registers on page 14* См. "Регистры общего назначения"
MSP	read-write	Privileged	See description	*Stack pointer on page 14* (См. "Указатель стека")
PSP	read-write	Either	Unknown	*Stack pointer on page 14* (См. "Указатель стека")
LR	read-write	Either	0xFFFFFFFF	*Link register on page 14* (См. "Регистр связи")
PC	read-write	Either	See description	*Program counter on page 14* См. "Счетчик программы"
PSR	read-write	Privileged	0x01000000	*Program status register on page 15* См. "Регистр статуса программы"
ASPR	read-write	Either	0x00000000	*Application program status register on page 16 (*См. "Регистр статуса программы приложения")
IPSR	read-only	Privileged	0x00000000	*Interrupt program status register on page 17* См. "Регистр статуса прерываний"
EPSR	read-only	Privileged	0x01000000	*Execution program status register on page 18* См. "Регистр статуса исключений"
PRIMASK	read-write	Privileged	0x00000000	*Priority mask register on page 19* См. "Регистры масок приоритетов"
FAULTMASK	read-write	Privileged	0x00000000	*Fault mask register on page 19* См. "Регистр масок ошибок"
BASEPRI	read-write	Privileged	0x00000000	*Base priority mask register on page 20* См. "Регистр масок базы приоритетов"
CONTROL	read-write	Privileged	0x00000000	*CONTROL register on page 20* См. "Регистр **CONTROL**"

1. Describes access type during program execution in thread mode and Handler mode. Debug access can differ. Описывает тип доступа во время выполнения программы в режимах **Thread** и **Handler**. Доступ отладчика может отличаться.

2. An entry of Either means privileged and unprivileged software can access the register. Термин **Either** означает, что и привилегированный и непривилегированный код может обратиться к регистру.

General-purpose registers (Регистры общего назначения)

R0-R12 are 32-bit general-purpose registers for data operations.
R0-R12 - это 32-х битные регистры общего назначения для манипуляций с данными.

Stack pointer (Указатель стека)

The Stack Pointer (SP) is register R13. In Thread mode, bit[1] of the CONTROL register indicates the stack pointer to use:
Регистр **R13** или **SP** - это Указатель вершины стека. В режиме **Thread bit[1]** регистра **CONTROL** указывает, какой из указателей стека используется:

- 0 = Main Stack Pointer (MSP). This is the reset value.
  Указатель на **Main Stack** (**MSP**). Это указатель по умолчанию после сброса.
- 1 = Process Stack Pointer (PSP). (Указатель на **Process Stack** (**PSP**))

On reset, the processor loads the MSP with the value from address 0x00000000.
После сброса процессор загружает в **MSP** значение, которое находится по адресу **0x00000000**.

Link register (Регистр связи)

The Link Register (LR) is register R14. It stores the return information for subroutines, function calls, and exceptions. On reset, the processor loads the LR value 0xFFFFFFFF.
Регистр **R14** или **LR** - это Регистр связи. Он хранит информацию о точке возврата из подпрограмм, вызванных функций и исключений. После сброса процессор загружает в **LR** значение **0xFFFFFFFF**.

Program counter (Счетчик программы)

The Program Counter (PC) is register R15. It contains the current program address. Bit[0] is always 0 because instruction fetches must be halfword aligned. On reset, the processor loads the PC with the value of the reset vector, which is at address 0x00000004.
Регистр **R15** или **PC** - это Счетчик Программы. Он содержит адрес на текущую инструкцию программы. Бит [0] всегда 0, потому что операция выборки инструкции должны быть выровненна до полуслова. После сброса процессор загружает в **PC** значение вектора обработчика сброса, который находится по адресу **0x00000004**.

Program status register (Регистр статуса программы)

The Program Status Register (PSR) combines:
Регистр статуса программы (**PSR**) объединяет в себе:

- Application Program Status Register (APSR) (Регистр статуса приложения)
- Interrupt Program Status Register (IPSR) (Регистр статуса прерываний)
- Execution Program Status Register (EPSR) (Регистр статуса исключений)

These registers are mutually exclusive bitfields in the 32-bit PSR. The bit assignments are as shown in Figure 3 and Figure 4.
Эти регистры имеют непересекающиеся битовые поля, отраженные на 32-х разрядный регистр **PSR**. Назначение бит показано на рисунках 3 и 4.

Figure 3. APSR, IPSR and EPSR bit assignments (Назначение бит **APSR**, **IPSR** и **EPSR**)

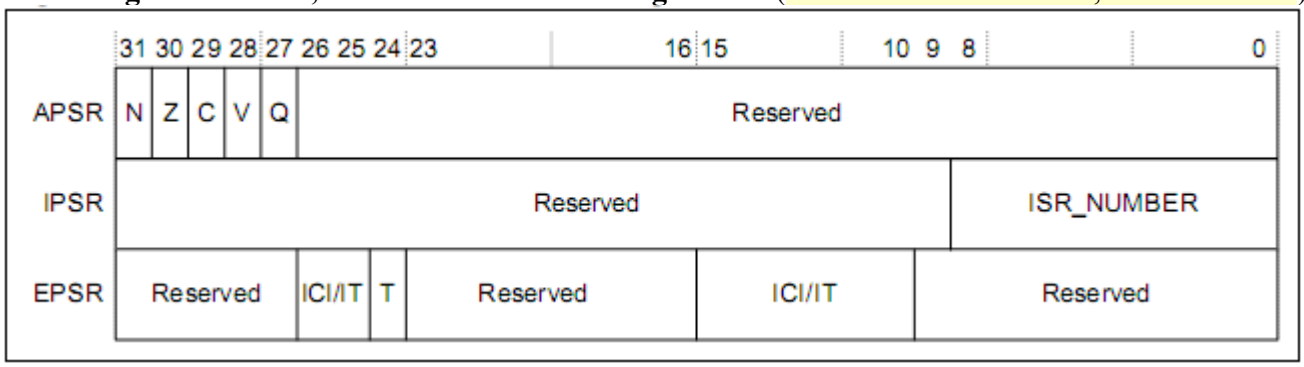
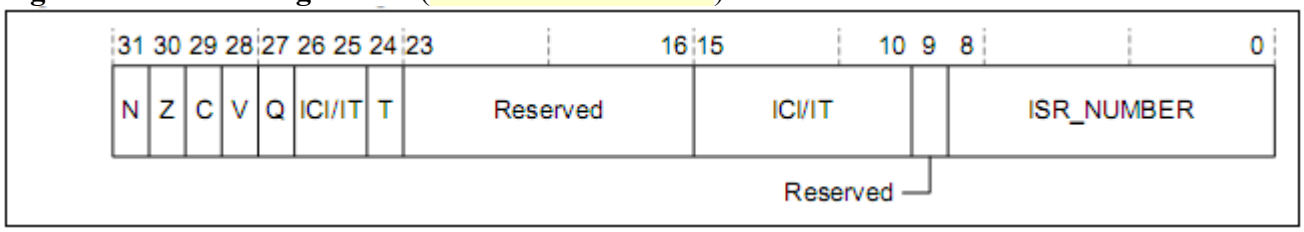


Figure 4. PSR bit assignments (Назначение бит **PSR**)



Access these registers individually or as a combination of any two or all three registers, using the register name as an argument to the MSR or MRS instructions. For example:
Можно обращаться к этим регистрам индивидуально, или как к комбинации любых двух, или всех трех регистров, используя название регистра как аргумент для инструкции **MSR** или **MRS**. Например:

- Read all of the registers using PSR with the MRS instruction
Читаем все регистры используя инструкцию **MRS** с аргументом **PSR**
- Write to the APSR using APSR with the MSR instruction.
Записываем в **APSR** используя инструкцию **MRS** с аргументом **APSR**

The PSR combinations and attributes are:
Возможные комбинации и атрибуты регистра **PSR**

Table 3. PSR register combinations (Комбинации регистра **PSR**)

Register	Type	Combination
PSR	read-write(1), (2)	APSR, EPSR, and IPSR
IEPSR	read-only	EPSR and IPSR
IAPSR	read-write(1)	APSR and IPSR
EAPSR	read-write(2)	APSR and EPSR

1. The processor ignores writes to the IPSR bits. (Процессор игнорирует запись в биты **IPSR**)
2. Reads of the EPSR bits return zero, and the processor ignores writes to the these bits
Чтение битов **EPSR** возвращает нуль, и процессор игнорируют запись в биты

See the instruction descriptions MRS on page 99 and MSR on page 100 for more information about how to access the program status registers.
См. описание инструкций **MRS** и **MSR** для получения дополнительной информации о том, как обратиться к регистрам состояния программы.

Application program status register (Регистр статуса приложения)

The APSR contains the current state of the condition flags from previous instruction executions. See the register summary in *Table 2 on page 13* for its attributes. The bit assignments are:
APSR содержит текущее состояние флагов условия от выполнения предыдущей инструкции. См. атрибуты регистра в Таблице 2 **"Перечень регистров ядра"**. Биты имеют следующее назначение:

Table 4. APSR bit definitions (Назначение битов регистра **APSR**)

Bits	Description
Bit 31	**N**: Negative or less than flag: (Флаг "Отрицательный" или "меньше чем") **0:** Operation result was positive, zero, greater than, or equal Результат операции был: "Положительный", "Ноль", "Больше или равно" **1:** Operation result was negative or less than. Результат операции был: "Отрицательный" или "меньше чем"
Bit 30	**Z**: Zero flag: (Флаг нулевого результата) **0:** Operation result was not zero (Результат операции не был равен нулю) **1:** Operation result was zero. (Результат операции был равен нулю)
Bit 29	**C**: Carry or borrow flag: (Флаг переноса или заема) **0:** Add operation did not result in a carry bit or subtract operation resulted in a borrow bit В операциии сложения не был выставлен бит переноса, или в операции вычитания использовался бит заема **1:** Add operation resulted in a carry bit or subtract operation did not result in a borrow bit. В операциии сложения был выставлен бит переноса, или в операции вычитания не использовался бит заема.
Bit 28	**V**: Overflow flag: (Флаг переполнения) **0:** Operation did not result in an overflow (В результате операции не было переполнения) **1:** Operation resulted in an overflow. (В результате операции было переполнение)
Bit 27	**Q**: Sticky saturation flag: (Флаг насыщения) **0:** Indicates that saturation has not occurred since reset or since the bit was last cleared to zero Не было события насыщения после последнего сброса, или после последней очистки этого бита **1:** Indicates when an SSAT or USAT instruction results in saturation. Показывает, что было событие насыщения после использования инструкции **SSAT** или **USAT**. This bit is cleared to zero by software using an MRS instruction. Этот бит можно очистить программно, с помощью инструкции **MRS**.
Bit 26:0	Reserved.

Interrupt program status register (Регистр статуса прерываний)

The IPSR contains the exception type number of the current Interrupt Service Routine (ISR). See the register summary in *Table 2 on page 13* for its attributes. The bit assignments are:
IPSR содержит номер исключения текущей процедуры обработки прерывания (**ISR**). См. атрибуты регистра в Таблице 2 **"Перечень регистров ядра"**. Биты имеют следующее назначение:

Table 5. IPSR bit definitions (Назначение битов регистра **IPSR**)

Bits	Description
Bit 31:9	Reserved.
Bit 8:0	**ISR_NUMBER**: This is the number of the current exception: (Это номер текущего исключения:) 0: Thread mode 1: Reserved 2: NMI 3: Hard fault 4: Memory management fault 5: Bus fault 6: Usage fault 7: Reserved .... 10: Reserved 11: SVCall 12: Reserved for Debug 13: Reserved 14: PendSV 15: SysTick 16: IRQ0(1) .... 83: IRQ67(1) see *Exception types on page 32* for more information. См. раздел 2.3.2 "Типы исключений" для дополнительной информации.

1. See STM32 product reference manual/datasheet for more information on interrupt mapping
См. справочники и руководства по продуктам **STM32** для дополнительной информации по отражению прерываний.

Execution program status register (Регистр статуса исключений)

The EPSR contains the Thumb state bit, and the execution state bits for either the:
EPSR содержит бит использования набора команд **Thumb**, а также следующие биты состояния выполнения инструкций:

- If-Then (IT) instruction (Инструкция **If-Then** (**IT**))
- Interruptible-Continuable Instruction (ICI) field for an interrupted load multiple or store multiple instruction.
Поле **ICI** состояния прерываемых инструкций загрузки/сохранения блока данных.

See the register summary in *Table 2 on page 13* for the EPSR attributes. The bit assignments are:
См. атрибуты регистра **EPSR** в Таблице 2 **"Перечень регистров ядра"**. Биты имеют следующее назначение:

Table 6. EPSR bit definitions (Назначение битов регистра **EPSR**)

Bits	Description
Bit 31:27	Reserved.
Bits 26:25, 15:10	**ICI**: Interruptible-continuable instruction bits Биты прерываемой инструкции работы с блоком данных See *Interruptible-continuable instructions on page 18*. См. параграф **"Прерываемые инструкции работы с блоком данных"**
Bits 26:25, 15:10	**IT**: Indicates the execution state bits of the IT instruction, see IT on page 93. Биты состояние выполнения инструкции **IT**.
Bit 24	Always set to 1. (Всегда 1)
Bits 23:16	Reserved.
Bit 9:0	Reserved.

Attempts to read the EPSR directly through application software using the MSR instruction always return zero. Attempts to write the EPSR using the MSR instruction in application software are ignored. Fault handlers can examine EPSR value in the stacked PSR to indicate the operation that is at fault. See *Section 2.3.7: Exception entry and return on page 36*
Попытка читать **EPSR** непосредственно из прикладной программы, с помощью инструкции **MSR** всегда возвращают нуль. Попытка записи в **EPSR** с помощью инструкции **MSR** в прикладной программе игнорируется. Обработчики ошибок могут проверять значение **EPSR** в сохраненном в стеке **PSR**, чтобы определить операцию, которая привела к ошибке.
См. раздел 2.3.7: "Вход в исключение и выход из него".

Interruptible-continuable instructions (Прерываемые инструкции работы с блоком данных)

When an interrupt occurs during the execution of an LDM or STM instruction, the processor:
Если случается прерывание во время работы инструкций **LDM** или **STM**, то процессор:

- Stops the load multiple or store multiple instruction operation temporarily
Временно приостанавливает инструкции работы с блоком данных
- Stores the next register operand in the multiple operation to EPSR bits[15:12].
Сохраняет следующий регистровый операнд такой инструкции в битах **[15:12]** регистра **EPSR**.

After servicing the interrupt, the processor: (После обслуживания прерывания, процессор:)

- Returns to the register pointed to by bits[15:12]
Возвращается к регистру, на который указывают биты **[15:12]**
- Resumes execution of the multiple load or store instruction.
Продолжает выполнение загрузки/сохранения блока данных.

When the EPSR holds ICI execution state, bits[26:25,11:10] are zero.
Когда **EPSR** хранит состояние выполнения **ICI**, в битах **[26:25,11:10]** находятся нули.

If-Then block (Блок инструкций для If-Then)

The If-Then block contains up to four instructions following a 16-bit IT instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See IT on page 93 for more information.
Блок инструкций для **If-Then** содержит до четырех инструкций, следующих за 16-ти битной инструкцией **IT**. Каждая инструкция в этом блоке - условное выражение. Условия для инструкций либо все одинаковые, либо некоторые могут быть инверсией других. См. описание инструкций **IT**.

Exception mask registers (Регистры масок исключений)

The exception mask registers disable the handling of exceptions by the processor. Disable exceptions where they might impact on timing critical tasks.
Регистры масок исключений отключают обрабатку исключений процессором. Отключите те исключения, которые могли бы воздействовать на время выполнения критических задач.

To access the exception mask registers use the MSR and MRS instructions, or the CPS instruction to change the value of PRIMASK or FAULTMASK. See MRS on page 99, MSR on page 100, and CPS on page 97 for more information.
Чтобы обратиться к регистрам масок исключений используют инструкции **MSR** и **MRS** или **CPS**, чтобы изменить значение **PRIMASK** или **FAULTMASK**. См. описание инструкций **MRS**, **MSR**, **CPS** для получения дополнительной информации.

Priority mask register (Регистры масок приоритетов)

The PRIMASK register prevents activation of all exceptions with configurable priority. See the register summary in *Table 2 on page 13* for its attributes. Figure 5 shows the bit assignments.
Регистр **PRIMASK** предотвращает активацию всех исключений с конфигурируемым приоритетом. См. атрибуты регистра в Таблице 2 **"Перечень регистров ядра"**. Рисунок 5 показывает назначение битов.

Figure 5. PRIMASK bit assignments (Назначение битов регистра **PRIMASK**)

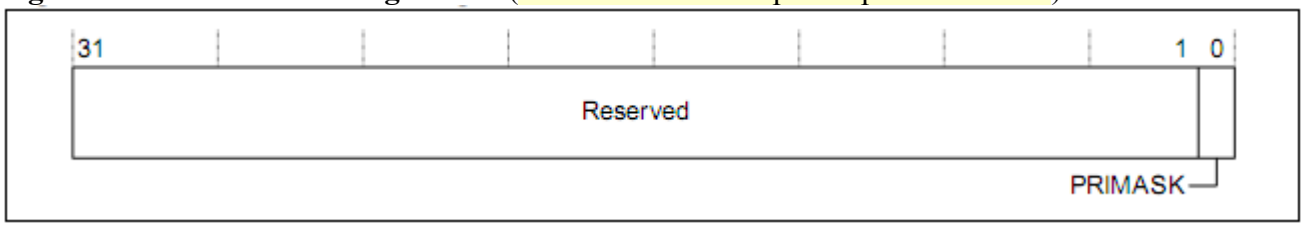


Table 7. PRIMASK register bit definitions (Описание битов регистра **PRIMASK**)

Bits	Description
Bit 31:1	Reserved.
Bits 0	**PRIMASK**: **0:** No effect (Нет эффекта) **0:** Prevents the activation of all exceptions with configurable priority. Предотвращает активацию всех исключений с конфигурируемым приоритетом

Fault mask register (Регистр масок ошибок)

The FAULTMASK register prevents activation of all exceptions except for Non-Maskable Interrupt (NMI). See the register summary in *Table 2 on page 13* for its attributes. Figure 6 shows the bit assignments.
Регистр **FAULTMASK** предотвращает активацию всех исключений, за исключением немаскируемого прерывания (**NMI**). См. атрибуты регистра в Таблице 2 **"Перечень регистров ядра"**. Рисунок 5 показывает назначение битов.

Figure 6. FAULTMASK bit assignments (Назначение битов регистра **FAULTMASK**)

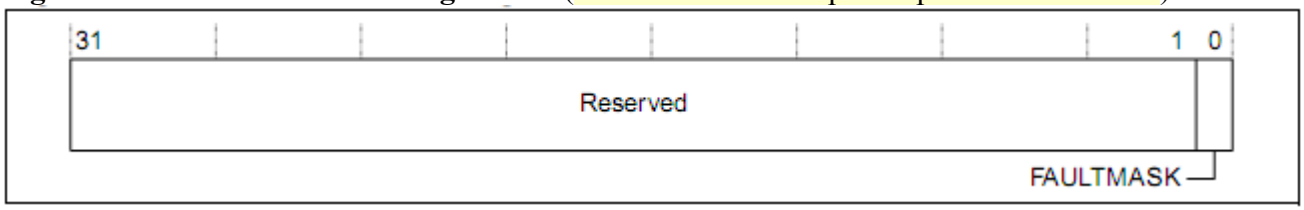


Table 8. FAULTMASK register bit definitions (Описание битов регистра **FAULTMASK**)

Bits	Description
Bit 31:1	Reserved.
Bits 0	**FAULTMASK**: **0:** No effect (Нет эффекта) **1:** Prevents the activation of all exceptions except for NMI. предотвращает активацию всех исключений, кроме **NMI**.

The processor clears the FAULTMASK bit to 0 on exit from any exception handler except the NMI handler. Процессор очищает бит **FAULTMASK** по выходу из любого обработчика исключений, кроме **NMI**.

Base priority mask register (Регистр масок базы приоритетов)

The BASEPRI register defines the minimum priority for exception processing. When BASEPRI is set to a nonzero value, it prevents the activation of all exceptions with same or lower priority level as the BASEPRI value. See the register summary in *Table 2 on page 13* for its attributes. Figure 7 shows the bit assignments.
Регистр **BASEPRI** определяет минимальный приоритет для обработки исключения. Когда **BASEPRI** установлен в значение отличное от нуля, он предотвращает активацию всех исключений с тем же самым или более низким приоритетом, чем значение **BASEPRI**. См. атрибуты регистра в Таблице 2 **"Перечень регистров ядра"**. Рисунок 7 показывает назначение битов.

Figure 7. BASEPRI bit assignments (Назначение битов регистра **BASEPRI**)

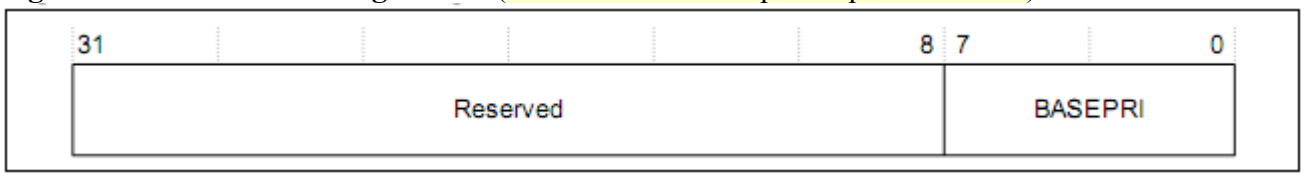


Table 9. BASEPRI register bit assignments (Описание битов регистра **BASEPRI**)

Bits	Description
Bit 31:8	Reserved.
Bits 7:4	**BASEPRI[7:4]** Priority mask bits[1] **0x00:** No effect (Нет эффекта) **Nonzero:** defines the base priority for exception processing. The processor does not process any exception with a priority value greater than or equal to BASEPRI. (Определяет базовый приоритет для обработки исключения. ЦПУ не обрабатывает исключения с приоритетным значением, больше чем, или равный **BASEPRI**.)
Bit 3:0	Reserved.

1. This field is similar to the priority fields in the interrupt priority registers. See *Interrupt priority registers (NVIC_IPRx) on page 111* for more information. Remember that higher priority field values correspond to lower exception priorities.
Это поле похоже на поле приорита в регистрах приоритета прерывания. См. раздел 4.2.7 "Регистр приоритетов прерываний (NVIC_IPRx)" для получения дополнительной информации. Помните, что более высокие значения приоритета соответствуют более низким приоритетам исключения.

CONTROL register (Регистр CONTROL)

The CONTROL register controls the stack used and the privilege level for software execution when the processor is in Thread mode. See the register summary in *Table 2 on page 13* for its attributes. Figure 8 shows the bit assignments.
Регистр **CONTROL** управляет используемым стеком и уровнем привилегий для программного кода, когда процессор находится в режиме **Thread**. См. атрибуты регистра в Таблице 2 **"Перечень регистров ядра"**. Рисунок 8 показывает назначение битов.

Figure 8. CONTROL bit assignments (Назначение битов регистра **CONTROL**)

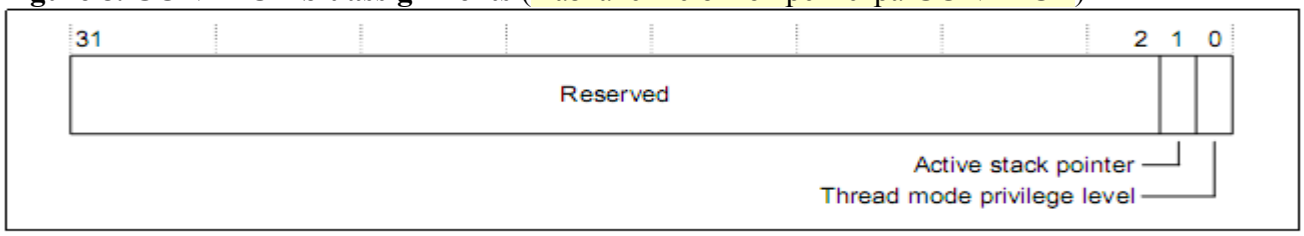


Table 10. CONTROL register bit definitions (Описание битов регистра **CONTROL**)

Bits	Description
Bit 31:2	Reserved.
Bits 1	**ASPSEL**: Active stack pointer selection (Выбор активного стека) Selects the current stack: (Выбирает текущий стек:) **0:** MSP is the current stack pointer (Указатель на текущий стек - **MSP**) **1:** PSP is the current stack pointer. (Указатель на текущий стек — **PSP**) In Handler mode this bit reads as zero and ignores writes. В режиме **Handler** этот бит читается как ноль, а запись игнорируется.
Bit 0	**TPL**: Thread mode privilege level (Уровень привелегий в режиме **Thread**) Defines the Thread mode privilege level. Определяет уровень привелегий в режиме **Thread** **0:** Privileged **1:** Unprivileged.

Handler mode always uses the MSP, so the processor ignores explicit writes to the active stack pointer bit of the CONTROL register when in Handler mode. The exception entry and return mechanisms update the CONTROL register.
Режим **Handler** всегда использует **MSP**, поэтому процессор игнорирует явную запись в этот бит регистра **CONTROL** в этом режиме. Процедуры входа и возвращения из исключений обновляют регистр **CONTROL**.

In an OS environment, it is recommended that threads running in Thread mode use the process stack and the kernel and exception handlers use the main stack.
В среде **OS** рекомендуется, чтобы треды запускались в режиме **Thread** и использовали стек **process**, а ядро и обработчики исключений использовали стек **main**.

By default, Thread mode uses the MSP. To switch the stack pointer used in Thread mode to the PSP, use the MSR instruction to set the Active stack pointer bit to 1, see MSR on page 100.
По умолчанию, в режиме **Thread** используется **MSP**. Чтобы переключить указатель стека, используемый в режим **Thread**, на **PSP**, используйте инструкцию **MSR**, чтобы установить бит активного указателя стека в '1', см. описание **MSR**.

When changing the stack pointer, software must use an ISB instruction immediately after the MSR instruction. This ensures that instructions after the ISB execute using the new stack pointer. See ISB on page 99.
При смене указателя стека, программа должна немедленно использовать инструкцию **ISB** после инструкции **MSR**. Это гарантирует, что инструкции, следующие после **ISB**, будут выполняться с использованием нового указателя стека. См. описание **ISB**.

2.1.4 Exceptions and interrupts (Исключения и прерывания)

The Cortex-M3 processor supports interrupts and system exceptions. The processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions. An exception changes the normal flow of software control. The processor uses handler mode to handle all exceptions except for reset. See *Exception entry on page 37* and *Exception return on page 38* for more information.
Процессор **Cortex-M3** поддерживает прерывания и системные исключения. Процессор и контроллер вложенных векторов прерываний (**NVIC**) обрабатывают все исключения в порядке их приоритетов. Исключение изменяет нормальное течение управления программой. Процессор использует режим **Handler** при обслуживании всех исключений, кроме сброса. См. параграфы "Вход в исключение" и "Возврат из исключения" для получения дополнительной информации.

The NVIC registers control interrupt handling. See *Nested vectored interrupt controller (NVIC) on page 104* for more information.
NVIC регистрирует обрабатывающее прерывание управления. См. раздел 4.2 "Контроллер вложенных векторов прерываний (NVIC)" для получения дополнительной информации.

2.1.5 Data types (Типы данных)

The processor: (Процессор:)

- Supports the following data types: (Поддерживает следующие типы данных:)

  -- 32-bit words
  -- 16-bit halfwords
  -- 8-bit bytes

- supports 64-bit data transfer instructions.

поддерживает инструкции обмена 64-х битными данными

- manages all memory accesses (data memory, instruction memory and Private Peripheral Bus (PPB)) as little-endian. See *Memory regions, types and attributes on page 24* for more information.

управляет всем доступом к памяти (к памяти данных, к памяти команд и к Частной Периферийной Шине (**PPB**)) в режиме **little-endian**. См. параграф "Регионы памяти, тип и атрибуты" для получения дополнительной информации.

2.1.6 The Cortex microcontroller software interface standard (CMSIS)
Стандартный программный интерфейс к микроконтроллеру Cortex (CMSIS)

For a Cortex-M3 microcontroller system, the Cortex Microcontroller Software Interface Standard (CMSIS) defines:

Для микроконтроллеров системы **Cortex-M3** стандартный программный интерфейс к микроконтроллеру **Cortex** (Библиотека **CMSIS**) определяет:

- A common way to: (Общепринятый способ для:)
  - Access peripheral registers (Доступа к регистрам периферии)
  - Define exception vectors (Назначения векторов исключений)
- The names of: (Общепринятый способ именования для:)
  - The registers of the core peripherals (Регистров периферии ядра)
  - The core exception vectors (Векторов исключений ядра)
- A device-independent interface for RTOS kernels, including a debug channel

Интерфейс к **RTOS** ядра и каналам отладчика, независимый от реализации устройства

The CMSIS includes address definitions and data structures for the core peripherals in the Cortex-M3 processor. It also includes optional interfaces for middleware components comprising a TCP/IP stack and a Flash file system. **CMSIS** включает определение адресов и структур данных для периферии ядра и процессора **Cortex-M3**. Она включает также опционные интерфейсы для компонентов связующего ПО, таких как стек протоколов **TCP/IP** и файловую систему для **Flash**.

CMSIS simplifies software development by enabling the reuse of template code and the combination of CMSIS-compliant software components from various middleware vendors. Software vendors can expand the CMSIS to include their peripheral definitions and access functions for those peripherals. **CMSIS** упрощает разработку программного обеспечения, допуская многократное использование шаблонов кода и комбинирование компонентов **CMSIS**-совместимого ПО от различных поставщиков связующего ПО. Поставщики ПО могут расширить **CMSIS**, чтобы включать в нее свои определения для регистров периферии и функции доступа к ним для своих внешних устройств.

This document includes the register names defined by the CMSIS, and gives short descriptions of the CMSIS functions that address the processor core and the core peripherals.

Этот документ использует названия регистров, определенные в **CMSIS**, и дает короткие описания функций **CMSIS**, которые обращаются к ядру процессора и основным внешним устройствам.

This document uses the register short names defined by the CMSIS. In a few cases these differ from the architectural short names that might be used in other documents. The following sections give more information about the CMSIS:

Этот документ использует аббревиатуры регистров, определенные в **CMSIS**. В некоторых случаях

они отличаются от тех, что даны при описании архитектуры ядра, и которые могут использоваться в других документах. Последующие разделы дают подробную информацию о **CMSIS**:

- *Section 2.5.4: Power management programming hints on page 43*

Раздел 2.5.4: "Рекомендации по программированию управлением потреблением"

- *Intrinsic functions on page 49* (Раздел 3.2 "**Intrinsic functions**")

- *The CMSIS mapping of the Cortex-M3 NVIC registers on page 105*

Раздел 4.2.1 "Как **CMSIS** отображает регистры **Cortex-M3 NVIC**"

- *NVIC programming hints on page 113*

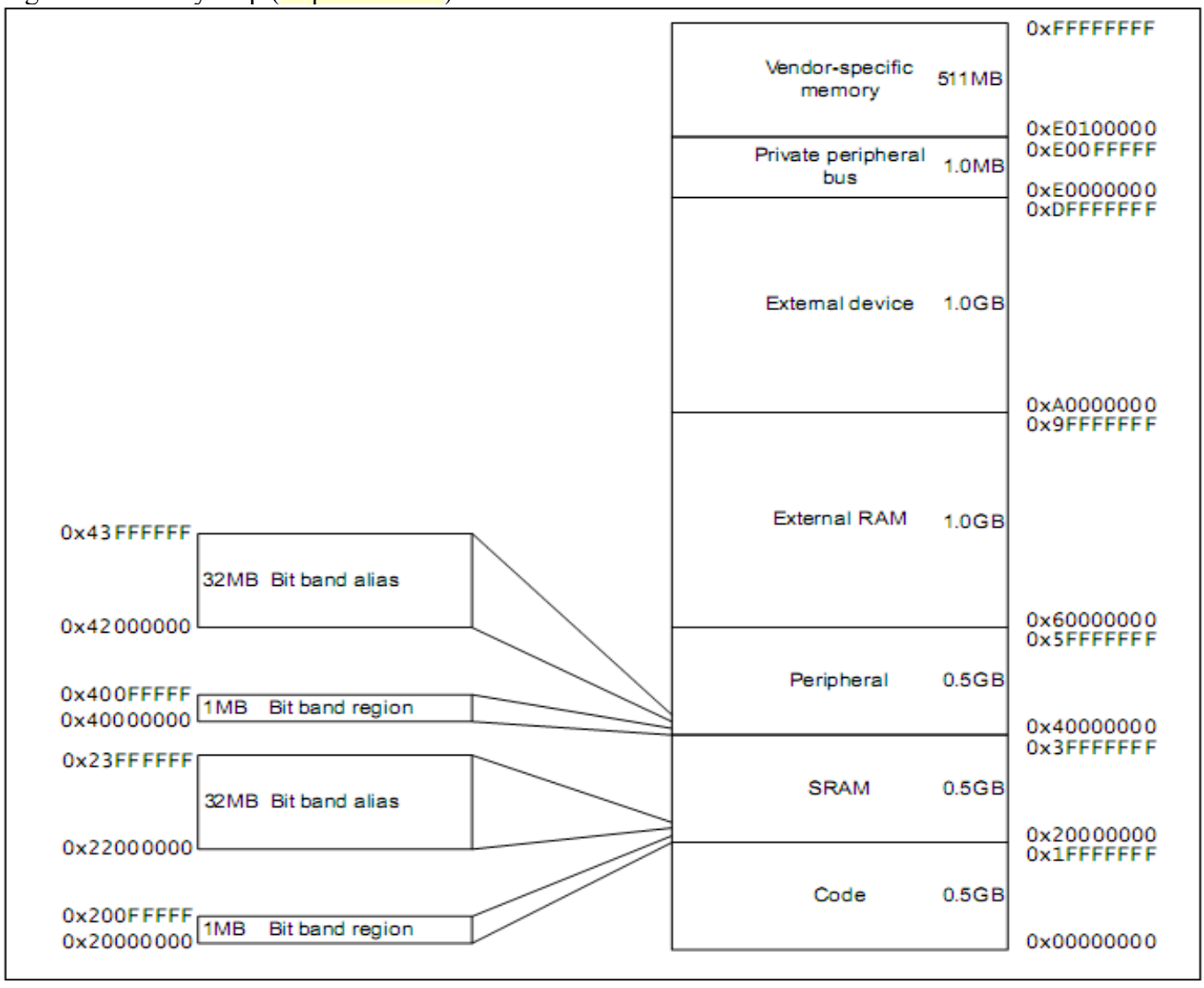
Параграф "Советы по программированию **NVIC**" в разделе 4.2.10.

2.2 Memory model (Модель памяти)

2.2.1 Memory regions, types and attributes (Регионы памяти, тип и атрибуты)
2.2.2 Memory system ordering of memory accesses
Как контроллер памяти регулирует порядок доступа к ней
2.2.3 Behavior of memory accesses (Поведение при доступе к памяти)
2.2.4 Software ordering of memory accesses
(Программное упорядочивание доступа к памяти)
2.2.5 Bit-banding (Побитовые операции в памяти)
2.2.6 Memory endianness (Порядок байтов **endianness**)
2.2.7 Synchronization primitives (Примитивы синхронизации)
2.2.8 Programming hints for the synchronization primitives
Рекомендации по программированию примитивов синхронизации

This section describes the processor memory map, the behavior of memory accesses, and the bit-banding features. The processor has a fixed memory map that provides up to 4 GB of addressable memory.
Этот раздел описывает карту памяти процессора, его поведение при доступе к памяти, и особенность **bit-banding** (побитовой работы). Процессор имеет фиксированную карту памяти, которая обеспечивает до 4 ГБайт адресуемой памяти.

Figure 9. Memory map (Карта памяти)



The regions for SRAM and peripherals include bit-band regions. Bit-banding provides atomic operations to bit data, see *Section 2.2.5: Bit-banding on page 27*.
В области **SRAM** и памяти периферии есть область, допускающая побитовую работу. Это функция **Bit-banding**, которая обеспечивает атомарные операции побитовой работы с данными, см. раздел 2.2.5: "Побитовые операции в памяти".

The processor reserves regions of the Private peripheral bus (PPB) address range for core peripheral registers, see *Section 4.1: About the STM32 core peripherals on page 104*.
Процессор резервирует область памяти **PPB** (*Private Peripheral Bus*) для адресации к регистрам периферии ядра, см. раздел 4.1 "О периферии ядра для **STM32**".

2.2.1 Memory regions, types and attributes
Регионы памяти, тип и атрибуты

The memory map splits the memory map into regions. Each region has a defined memory type, and some regions have additional memory attributes. The memory type and attributes determine the behavior of accesses to the region.
Карта памяти разбивает память на регионы. У каждого региона есть определенный тип памяти, а у некоторых регионов есть дополнительные атрибуты памяти. Тип памяти и атрибуты памяти определяют поведение при доступе к региону памяти.

The memory types are: (Есть следующие типы памяти:)

Normal	The processor can re-order transactions for efficiency, or perform speculative reads. Процессор может переупорядочить транзакции для повышения эффективности доступа, или выполнять мнимое чтение.
Device	The processor preserves transaction order relative to other transactions to Device or Strongly-ordered memory. При обмене с устройством, процессор не меняет порядок транзакций, относительно других транзакций устройства, или относительно транзакций доступа к памяти типа **Strongly-ordered**.
Strongly-ordered	The processor preserves transaction order relative to all other transactions. При обмене с памятью типа **Strongly-ordered**, процессор не меняет порядок транзакций относительно всех других транзакций.

The different ordering requirements for Device and Strongly-ordered memory mean that the memory system can buffer a write to Device memory, but must not buffer a write to Strongly-ordered memory.
Различные требования к смене порядка транзакций для Устройств и памяти типа **Strongly-ordered** означает, что контроллер памяти может буферизовать запись в память Устройств, но не делает это при записи в память **Strongly-ordered**.

The additional memory attributes include:
Имеются следующие дополнительные атрибуты памяти:

Execute Never (XN) Никогда не исполнять	Means the processor prevents instruction accesses. Any attempt to fetch an instruction from an XN region causes a memory management fault exception. Это означает, что процессор блокирует доступ к инструкциям. Любая попытка выборки инструкции в области **XN** вызывает исключение ошибки управления памятью.

2.2.2 Memory system ordering of memory accesses
Как контроллер памяти регулирует порядок доступа к ней

For most memory accesses caused by explicit memory access instructions, the memory system does not guarantee that the order in which the accesses complete matches the program order of the instructions, providing this does not affect the behavior of the instruction sequence. Normally, if correct program execution depends on two memory accesses completing in program order, software must insert a memory barrier instruction between the memory access instructions, see *Section 2.2.4: Software ordering of memory accesses on page 26*.
Для большинства операций доступа к памяти, вызванных явными инструкциями доступа к памяти, контроллер памяти не гарантирует, что порядок, в котором идет реальный доступ, соответствует порядку инструкций в коде программы, но это никак не затрагивает поведение самих инструкций. Обычно, если корректное выполнение программы зависит от времени завершения двух доступов к памяти, согласно их порядку в коде программы, то программа должна вставить инструкцию барьера памяти между такими инструкциями доступа к памяти, см. раздел 2.2.4: "Процедура доступа к памяти".

However, the memory system does guarantee some ordering of accesses to Device and Strongly-ordered memory. For two memory access instructions A1 and A2, if A1 occurs before A2 in program order, the ordering of the memory accesses caused by two instructions is:
Однако, система памяти гарантирует некоторое упорядочение доступа к памяти Устройств и памяти типа **Strongly-ordered**. Для двух инструкци доступа к памяти **A1** и **A2**, если **A1** стоит перед **A2** в коде программы, то будет следующий порядок доступа к памяти:

Table 11. Ordering of memory accesses(1) (Порядок доступа к памяти)

A1	A2			
	Normal access	Device access		Strongly ordered access
		Non-shareable	Shareable	
Normal access	-	-	-	-
Device access, non-shareable	-	<	-	<
Device access, shareable	-	-	<	<
Strongly ordered access	-	<	<	<

1. '-' means that the memory system does not guarantee the ordering of the accesses.
'-' означает, что контроллер памяти не гарантирует порядок доступа.
'<' means that accesses are observed in program order, that is, A1 is always observed before A2.
'<' означает, что наблюдается доступ в соответсвии с кодом программы, то есть, **A1** всегда наблюдаются перед **A2**.

2.2.3 Behavior of memory accesses
Поведение при доступе к памяти

The behavior of accesses to each region in the memory map is:
Поведение при доступе к памяти для каждого региона будет следующим:

Table 12. Memory access behavior (Поведение при доступе к памяти)

Address range	Memory region	Memory type	XN	Description
0x00000000-0x1FFFFFFF	Code	Normal(1)	-	Executable region for program code. You can also put data here. Область памяти для кода программы. Вы можете также поместить здесь и данные.
0x20000000-0x3FFFFFFF	SRAM	Normal(1)	-	Executable region for data. You can also put code here. This region includes bit band and bit band alias areas, see Table 13 on page 27. Область памяти для данных с возможностью исполнения кода. Вы можете также поместить здесь и код. Этот регион включает также область данных с возможностью их побитовой обработки, см. таблицу 13.
0x40000000-0x5FFFFFFF	Peripheral	Device(1)	XN(1)	This region includes bit band and bit band alias areas, see Table 14 on page 27. Этот регион включает область данных с возможностью их побитовой обработки, см. таблицу 14.
0x60000000-0x9FFFFFFF	External RAM	Normal(1)	-	Executable region for data. Область памяти для данных с возможностью исполнения кода.
0xA0000000-0xDFFFFFFF	External device	Device(1)	XN(1)	External Device memory Память внешних устройств
0xE0000000-0xE00FFFFF	External Private Peripheral Bus	Strongly-ordered(1)	XN(1)	This region includes the NVIC, System timer, and system control block. Этот регион включает адреса системных регистров **NVIC**, системного таймера и блока **SCB**.
0xE0100000-0xFFFFFFFF	External Memory mapped peripherals	Device(1)	XN(1)	This region includes all the STM32 standard peripherals. Этот регион включает адреса регистров всей периферии **STM32**

1. See *Memory regions, types and attributes on page 24* for more information.
См. параграф "Регионы памяти, тип и атрибуты" для получения дополнительной информации.

The Code, SRAM, and external RAM regions can hold programs. However, it is recommended that programs always use the Code region. This is because the processor has separate buses that enable instruction fetches and data accesses to occur simultaneously.
Регионы памяти **Code**, **SRAM**, и внешней **RAM** могут содержать код программы. Однако, рекомендуется, чтобы код программы всегда размещался в регионе **Code**. Так как у процессора есть раздельные шины кода и данных, что допускает, в таком случае, одновременную выборку инструкции и доступ к данным.

2.2.4 Software ordering of memory accesses
Программное упорядочивание доступа к памяти

The order of instructions in the program flow does not always guarantee the order of the corresponding memory transactions. This is because:
Порядок инструкций в коде программы не всегда гарантирует порядок соответствующих транзакций в памяти. Это происходит потому, что:

- The processor can reorder some memory accesses to improve efficiency, providing this does not affect the behavior of the instruction sequence.
Процессор может переупорядочить ряд доступов к памяти, чтобы повысить эффективность, однако это никак не отражается на поведении самих инструкций.

- The processor has multiple bus interfaces
Процессор имеет несколько шин интерфейса

- Memory or devices in the memory map have different wait states
Собственно память и память устройств, отраженная на карту памяти, имеют различные тайминги

- Some memory accesses are buffered or speculative.
Некоторые доступы к памяти буферизируются или являются мнимыми

Section 2.2.2: Memory system ordering of memory accesses on page 25 describes the cases where the memory system guarantees the order of memory accesses. Otherwise, if the order of memory accesses is critical, software must include memory barrier instructions to force that ordering. The processor provides the following memory barrier instructions:
Раздел 2.2.2: "Как контроллер памяти регулирует порядок доступа к ней" описывает случаи, когда контроллер памяти гарантирует порядок доступа к памяти. В других случаях, если важен порядок доступа к памяти, в код программы нужно включать инструкции барьера памяти, чтобы обеспечить нужный порядок. Процессор предоставляет следующие инструкции барьера памяти:

DMB	The Data Memory Barrier (DMB) instruction ensures that outstanding memory transactions complete before subsequent memory transactions. See DMB on page 98. Инструкция **DMB** (*Data Memory Barrier*) гарантирует, что предыдущие транзакции памяти будут завершены перед последующими транзакциями памяти. См. описание инструкции **DMB**.
DSB	The Data Synchronization Barrier (DSB) instruction ensures that outstanding memory transactions complete before subsequent instructions execute. See DSB on page 98. Инструкция **DSB** (*Data Synchronization Barrier*) гарантирует, что предыдущие транзакции памяти будут завершены перед последующими инструкциями. См. описание инструкции **DSB**.
ISB	The Instruction Synchronization Barrier (ISB) ensures that the effect of all completed memory transactions is recognizable by subsequent instructions. See ISB on page 99. Инструкция **ISB** (*Instruction Synchronization Barrier*) гарантирует, что эффект от завершения всех транзакций памяти будет распознаваем всеми последующими инструкциями. См. описание инструкции **ISB**.

Use memory barrier instructions in, for example:
Примеры использования инструкций барьера памяти:

- **Vector table. (Таблица векторов.)**

If the program changes an entry in the vector table, and then enables the corresponding exception, use a DMB instruction between the operations. This ensures that if the exception is taken immediately after being enabled the processor uses the new exception vector.
Если программа изменяет элемент в таблице векторов, а затем разрешает соответствующее исключение, используйте инструкцию **DMB** между такими операциями. Это гарантирует, что, если будет запрос на исключение сразу после его разрешения, то процессор использует новый вектор исключения.

- **Self-modifying code. (Самомодифицирующийся код.)**

If a program contains self-modifying code, use an ISB instruction immediately after the code modification in the program. This ensures subsequent instruction execution uses the updated program.
Если программа содержит самомодифицирующийся код, используйте инструкцию **ISB** сразу после модификации кода в программе. Это гарантирует, что последующая инструкция будет из обновленной программы.

- **Memory map switching. (Переключение карты памяти)**

If the system contains a memory map switching mechanism, use a DSB instruction after switching the memory map in the program. This ensures subsequent instruction execution uses the updated memory map.
Если система содержит механизм переключения карты памяти, используйте инструкцию **DSB** после переключения карты памяти в программе. Это гарантирует, что последующая инструкция будут использовать обновленную карту памяти.

- **Dynamic exception priority change.**

Динамическое изменение приоритета исключения.
When an exception priority has to change when the exception is pending or active, use DSB instructions after the change. This ensures the change takes effect on completion of the DSB instruction.
Когда должен измениться приоритет исключения, когда исключение является отложенным или уже является активным, используйте инструкцию **DSB** после изменения приоритета. Это гарантирует, что изменение вступает в силу по завершению инструкции **DSB**.

- **Using a semaphore in multi-master system.**

Использование семафора в системе с несколькими мастерами.
If the system contains more than one bus master, for example, if another processor is present in the system, each processor must use a DMB instruction after any semaphore instructions, to ensure other bus masters see the memory transactions in the order in which they were executed.
Если система содержит более одного мастера на шине, например, если в системе присутствует другой процессор, каждый процессор должен использовать инструкцию **DMB** после любых семафорных инструкций, чтобы гарантировать, что другие мастера шины видят транзакции памяти в том порядке, в котором они выполнялись.

Memory accesses to Strongly-ordered memory, such as the system control block, do not require the use of DMB instructions.
Доступ к памяти типа **Strongly-ordered**, такой как блок **SCB**, не требуют использования инструкций **DMB**.

2.2.5 Bit-banding (Побитовые операции в памяти)

A bit-band region maps each word in a bit-band alias region to a single bit in the bit-band region. The bit-band regions occupy the lowest 1 MB of the SRAM and peripheral memory regions.
Область побитовых операций отражает каждое слово из области хранения бит **bit-band alias** на единственный бит в области доступа к битам **bit-band**.

The memory map has two 32 MB alias regions that map to two 1 MB bit-band regions:
Карта памяти имеет две области хранения бит **bit-band alias** по 32 МБайта, которые отображаются на две области доступа к битам **bit-band** по 1 МБайту:

- Accesses to the 32 MB SRAM alias region map to the 1 MB SRAM bit-band region, as shown in Table 13
Доступ к 32 МБайтной области **bit-band alias** отражается на 1 МБайтную область **bit-band** в регионе **SRAM**, как показано в таблице 13
- Accesses to the 32 MB peripheral alias region map to the 1 MB peripheral bit-band region, as shown in Table 14.
Доступ к 32 МБайтной области **bit-band alias** отражается на 1 МБайтную область **bit-band** в регионе памяти периферии, как показано в таблице 14

Table 13. SRAM memory bit-banding regions (Область **bit-banding** в регионе памяти **SRAM)**

Address range	Memory region	Instruction and data accesses
0x20000000-0x200FFFFF	SRAM bit-band region	Direct accesses to this memory range behave as SRAM memory accesses, but this region is also bit addressable through bit-band alias. Прямой доступ к этой области памяти ведет себя как доступ к памяти **SRAM**, но эта область доступна также побитно, с помощью ссылок из области **bit-band alias**.
0x22000000-0x23FFFFFF	SRAM bit-band alias	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not remapped. Доступ к данным в этой области переназначается на область **bit band**. Операция записи данных выполняется в стиле "чтение-изменение-запись". При выборке инструкции операция переназначения не производится.

Table 14. Peripheral memory bit-banding regions
Область **bit-banding** в регионе памяти периферии

Address range	Memory region	Instruction and data accesses
0x40000000-0x400FFFFF	Peripheral bit-band alias	Direct accesses to this memory range behave as peripheral memory accesses, but this region is also bit addressable through bit-band alias. Прямой доступ к этой области памяти ведет себя как доступ к памяти периферии, но эта область доступна также побитно, с помощью ссылок из области **bit-band alias**.
0x42000000-0x43FFFFFF	Peripheral bit-band region	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not permitted. Доступ к данным в этой области переназначается на область **bit band**. Операция записи данных выполняется в стиле "чтение-изменение-запись". При выборке инструкции операция переназначения не производится.

A word access to the SRAM or peripheral bit-band alias regions map to a single bit in the SRAM or peripheral bit-band region.
Доступ чтения слова в области **bit-band alias** в **SRAM** или памяти периферии отображается на единственный бит в **SRAM** или памяти периферии.

The following formula shows how the alias region maps onto the bit-band region:
Следующая формула показывает, как область **bit-band alias** отображает на область **bit-band**:

```
bit_word_offset = (byte_offset x 32) + (bit_number x 4)
bit_word_addr = bit_band_base + bit_word_offset
```

Where: (Где:)

- Bit_word_offset is the position of the target bit in the bit-band memory region.
Bit_word_offset - позиция целевого бита в области **bit-band**.

- Bit_word_addr is the address of the word in the alias memory region that maps to the targeted bit.
Bit_word_addr - адрес слова в области **bit-band alias**, которое отображается на целевой бит.

- Bit_band_base is the starting address of the alias region.
Bit_band_base - стартовый адрес области **bit-band alias**.

- Byte_offset is the number of the byte in the bit-band region that contains the targeted bit.
Byte_offset - номер байта в области **bit-band**, который содержит целевой бит.

- Bit_number is the bit position, 0-7, of the targeted bit.
Bit_number - позиция целевого бита бита в диапазоне 0-7.

Figure 10 on page 28 shows examples of bit-band mapping between the SRAM bit-band alias region and the SRAM bit-band region:
Рисунок 10 показывает пример побитового отображения между областью **bit-band alias** и областью **bit-band** в регионе **SRAM**:

- The alias word at 0x23FFFFE0 maps to bit[0] of the bit-band byte at

Слово по адресу **0x23FFFFE0** в области **bit-band alias** отображается на **bit[0]** в области **bit-band**

```
0x200FFFFF: 0x23FFFFE0 = 0x22000000 + (0xFFFFF*32) + (0*4).
```

- The alias word at 0x23FFFFFC maps to bit[7] of the bit-band byte at

Слово по адресу **0x23FFFFFC** в области **bit-band alias** отображается на **bit[7]** в области **bit-band**

```
0x200FFFFF: 0x23FFFFFC = 0x22000000 + (0xFFFFF*32) + (7*4).
```

- The alias word at 0x22000000 maps to bit[0] of the bit-band byte at

Слово по адресу **0x22000000** в области **bit-band alias** отображается на **bit[0]** в области **bit-band**

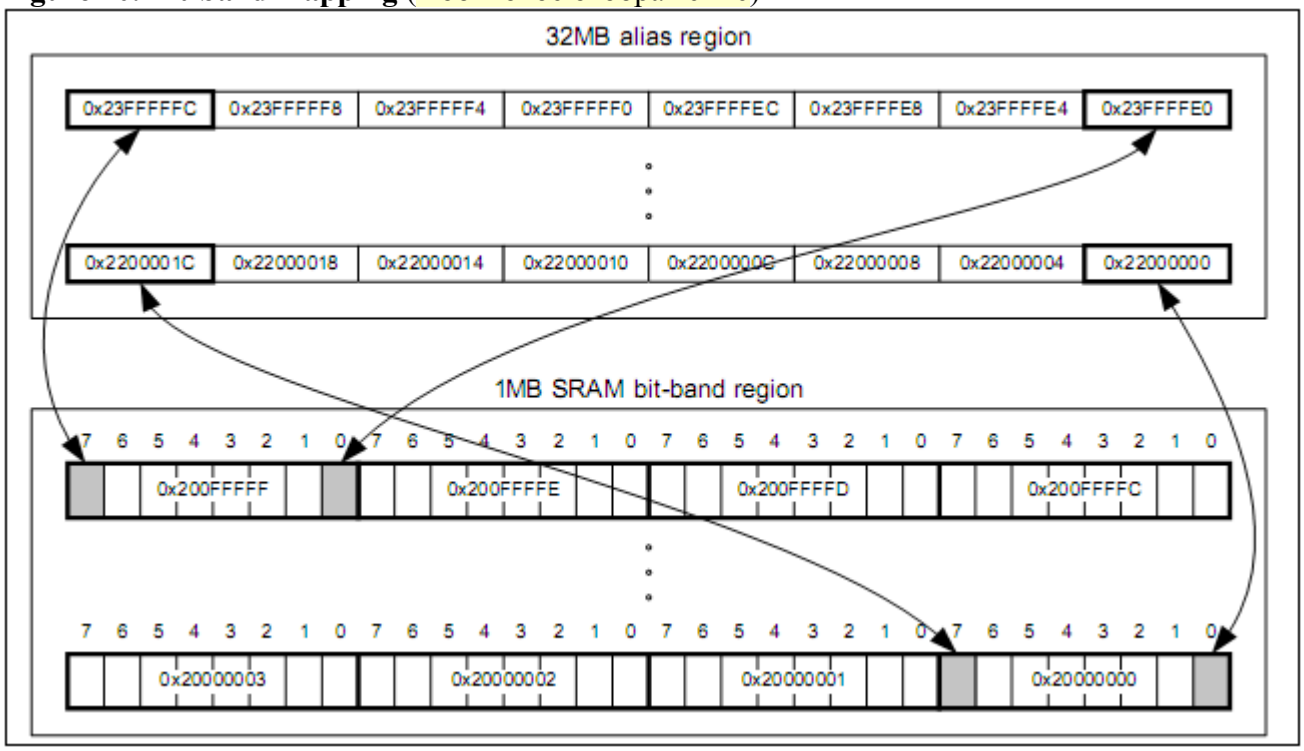
```
0x20000000: 0x22000000 = 0x22000000 + (0*32) + (0 *4).
```

- The alias word at 0x2200001C maps to bit[7] of the bit-band byte at

Слово по адресу **0x2200001C** в области **bit-band alias** отображается на **bit[7]** в области **bit-band**

```
0x20000000: 0x2200001C = 0x22000000+ (0*32) + (7*4).
```

Figure 10. Bit-band mapping (Побитовое отображение)



Directly accessing an alias region (Прямой доступ в бласти bit-band alias)

Writing to a word in the alias region updates a single bit in the bit-band region.
Запись слова в эту область обновляет единственный бит в области **bit-band**.

Bit[0] of the value written to a word in the alias region determines the value written to the targeted bit in the bit-band region. Writing a value with bit[0] set to 1 writes a 1 to the bit-band bit, and writing a value with bit[0] set to 0 writes a 0 to the bit-band bit.
Бит **[0]** значения, записанного в область **bit-band alias**, определяет значение, записанное в целевой бит в области **bit-band**. Запись значения с битом **[0]** установленного в '1' запишет '1' в бит области **bit-band**, а запись значения с битом [0], сброшенного в '0' запишет '0' в бит области **bit-band**.

Bits[31:1] of the alias word have no effect on the bit-band bit.
Writing 0x01 has the same effect as writing 0xFF.
Writing 0x00 has the same effect as writing 0x0E.
Биты **[31:1]** слова в области **bit-band alias** не имеют эффекта на бит в области **bit-band**.
Запись значения **0x01** имеет такой же эффект, что и запись **0xFF**.
Запись значения **0x00** имеет такой же эффект, что и запись **0x0E**.

Reading a word in the alias region: (Чтение слова в области **bit-band alias**)

- 0x00000000 indicates that the targeted bit in the bit-band region is set to zero

Значение **0x00000000** говорит о том, что бит в области **bit-band** установлен в '0'.

- 0x00000001 indicates that the targeted bit in the bit-band region is set to 1

Значение **0x00000001** говорит о том, что бит в области **bit-band** установлен в '1'.

Directly accessing a bit-band region (Прямой доступ в бласти bit-band)

Behavior of memory accesses on page 25 describes the behavior of direct byte, halfword, or word accesses to the bit-band regions.
Раздел 2.2.3 "Поведение при доступе к памяти" описывает поведение при прямом доступе к байту, полуслову, или слову в области **bit-band**.

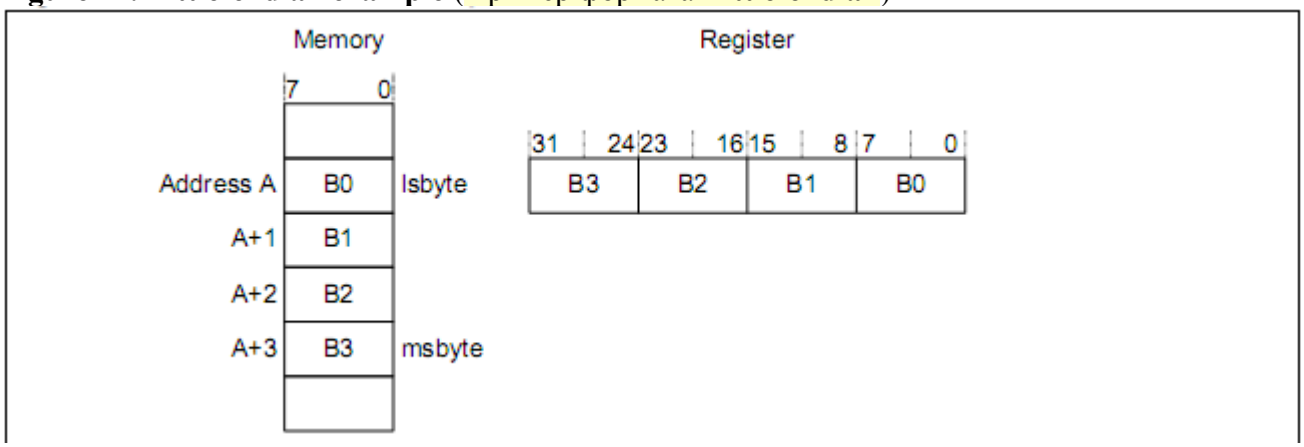
2.2.6 Memory endianness (Порядок байтов endianness)

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word.
Процессор рассматривает память как линейную коллекцию байтов, пронумерованных в порядке возрастания от нуля. Например, байты 0-3 содержат первое сохраненное слово, а байты 4-7 содержат второе сохраненное слово.

Little-endian format (Формат Little-endian)

In little-endian format, the processor stores the least significant byte of a word at the lowest-numbered byte, and the most significant byte at the highest-numbered byte. See Figure 11 for an example.
В формате **Little-endian** процессор хранит младший байт слова в байте с более низким номером, а старший байт слова в байте с более высоким номером. См. пример на рисунке 11.

Figure 11. Little-endian example (Пример формата **Little-endian**)



2.2.7 Synchronization primitives (Примитивы синхронизации)

The Cortex-M3 instruction set includes pairs of synchronization primitives. These provide a non-blocking mechanism that a thread or process can use to obtain exclusive access to a memory location. Software can use them to perform a guaranteed read-modify-write memory update sequence, or for a semaphore mechanism.
Набор команд **Cortex-M3** включает пары примитивов синхронизации. Они обеспечивают неблокирующий механизм, который могут использовать треды или процессы, чтобы получить монопольный доступ к области в памяти. Программа может использовать их, чтобы гарантированно выполнить операцию "чтение-изменение-запись", или для организации семафора.

A pair of synchronization primitives comprises: (Пары примитивов синхронизации включают:)

A Load-Exclusive instruction Инструкции монопольной загрузки	Used to read the value of a memory location, requesting exclusive access to that location. Используются для чтения значения в памяти, когда требуется монопольный доступ к данным.
A Store-Exclusive instruction Инструкции монопольного сохранения	Used to attempt to write to the same memory location, returning a status bit to a register. If this bit is: Используются для попытки записи в ту же самую область памяти, возвращая бит состояния в статусном регистре. Если этот бит равен: **0:** it indicates that the thread or process gained exclusive access to the memory, and the write succeeds это это указывает, что тред или процесс получили монопольный доступ к памяти, и запись была успешной **1:** it indicates that the thread or process did not gain exclusive access to the memory, and no write is performed это это указывает, что тред или процесс не получили монопольный доступ к памяти, и запись не была выполнена

The pairs of Load-Exclusive and Store-Exclusive instructions are:
Пары инструкций для монопольной загрузки/сохранения следующие:

- The word instructions LDREX and STREX
Инструкции операций со словами **LDREX** и **STREX**
- The halfword instructions LDREXH and STREXH
Инструкции операций с полусловами **LDREXH** и **STREXH**
- The byte instructions LDREXB and STREXB.
Инструкции операций с байтами **LDREXB** и **STREXB**.

Software must use a Load-Exclusive instruction with the corresponding Store-Exclusive instruction.
Программа должна использовать инструкцию монопольной загрузки с соответствующей парой монопольного сохранения.

To perform a guaranteed read-modify-write of a memory location, software must:
Чтобы гарантированно выполнить процедуру "чтение-изменение-запись" в памяти, программа должна:

1. Use a Load-Exclusive instruction to read the value of the location.
Использовать инструкцию монопольной загрузки, чтобы прочитать значение в памяти.
2. Update the value, as required. (Обновить это значение, как затребовано).

3. Use a Store-Exclusive instruction to attempt to write the new value back to the memory location, and tests the returned status bit. If this bit is:
Использовать инструкцию монопольного сохранения, чтобы попытаться записать новое значение в память, и проверить возвращенный бит состояния. Если этот бит равен:

0: The read-modify-write completed successfully,
Процедура "чтение-изменение-запись" завершилась успешно,
1: No write was performed. This indicates that the value returned at step 1 might be out of date. The software must retry the read-modify-write sequence.
Запись не была выполнена. Это указывает, что значение, возвращенное на шаге 1, might be out of date. Программа должна повторить процедуру "чтение-изменение-запись".

Software can use the synchronization primitives to implement a semaphores as follows:
Программа может использовать примитивы синхронизации, чтобы организовать семафоры следующим образом:

1. Use a Load-Exclusive instruction to read from the semaphore address to check whether the semaphore is free.
Использовать инструкцию монопольной загрузки, чтобы прочитать по адресу семафора, чтобы проверить, является ли семафор свободным.

2. If the semaphore is free, use a Store-Exclusive to write the claim value to the semaphore address.
Если семафор является свободным, использовать инструкцию монопольного сохранения, чтобы записать требуемое значение по адресу семафора.

3. If the returned status bit from step 2 indicates that the Store-Exclusive succeeded then the software has claimed the semaphore. However, if the Store-Exclusive failed, another process might have claimed the semaphore after the software performed step 1.
Если возвращенный бит состояния на шаге 2 указывает, что инструкция монопольного сохранения выполнена успешно, то программа освобождает семафор. Однако, если эта инструкция вернула ошибку, то возможно другой процесс занял семафор после того, как программа выполнила шаг 1.

The Cortex-M3 includes an exclusive access monitor, that tags the fact that the processor has executed a Load-Exclusive instruction.
Cortex-M3 включает монитор монопольного доступа, который отмечает факт, что процессор выполнил инструкцию монопольной загрузки.

The processor removes its exclusive access tag if:
Процессор удаляет свой флаг монопольного доступа, если:

- It executes a CLREX instruction (Он выполнил инструкцию **CLREX**)
- It executes a Store-Exclusive instruction, regardless of whether the write succeeds.
Он выполнил инструкцию монопольного сохранения, независимо от того, прошла ли она успешно.
- An exception occurs. This means the processor can resolve semaphore conflicts between different threads. Происходит исключение. Это означает, что процессор может разрешить конфликты семафора между различными тредами.

For more information about the synchronization primitive instructions, see LDREX and STREX on page 69 and CLREX on page 70.
Для получения дополнительной информации об инструкциях примитивов синхронизации, см. описание инструкций **LDREX** и **STREX** и **CLREX**.

2.2.8 Programming hints for the synchronization primitives
Рекомендации по программированию примитивов синхронизации

ANSI C cannot directly generate the exclusive access instructions. Some C compilers provide intrinsic functions for generation of these instructions:
Стандарт **ANSI C** не может непосредственно генерировать инструкции монопольного доступа. Некоторые **C** компиляторы обеспечивают встроенные функции для генерирования этих инструкций:

Table 15. C compiler intrinsic functions for exclusive access instructions
Встроенные функции в **C** компилятор для инструкций монопольного доступа

Instruction	Intrinsic function
LDREX, LDREXH, or LDREXB	unsigned int __ldrex(volatile void *ptr)
STREX, STREXH, or STREXB	int __strex(unsigned int val, volatile void *ptr)
CLREX	void __clrex(void)

The actual exclusive access instruction generated depends on the data type of the pointer passed to the intrinsic function. For example, the following C code generates the require LDREXB operation:
Фактичеси сгенерированная инструкция монопольного доступа зависит от типа данных, которые передают встроенной функции с помощью указателя. Например, следующий **C** код генерирует требуемую инструкцию **LDREXB**:

```
__ldrex((volatile char *) 0xFF);
```

2.3 Exception model (Модель исключений)

This section describes the exception model. (Этот раздел описывает модель исключений.)

2.3.1 Exception states (Статус исключения)
2.3.2 Exception types (Типы исключений)
2.3.3 Exception handlers (Обработчики исключений)
2.3.4 Vector table (Таблица векторов)
2.3.5 Exception priorities (Приоритеты исключений)
2.3.6 Interrupt priority grouping (Группировка приоритетов исключений)
2.3.7 Exception entry and return (Вход в исключение и выход из него)

2.3.1 Exception states (Статус исключения)

Each exception is in one of the following states:
Каждое исключение может находиться в одном из следующих состояний:

Inactive Неактивное	The exception is not active and not pending. Исключение неактивно и не запрошено.
Pending Запрошенное или Отложенное	The exception is waiting to be serviced by the processor. An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending. Исключение ждет обслуживания от процессора. Запрос на прерывание от периферии или от программы может изменить состояние соответствующего прерывания на Запрошенное.
Active Активное	An exception that is being serviced by the processor but has not completed. Note: An exception handler can interrupt the execution of another exception handler. In this case both exceptions are in the active state. Исключение, которое обслуживается процессором, и еще не завершилось. **Note**: обработчик одного исключения может прервать выполнение обработчика другого исключения. В этом случае оба исключения находятся в активном состоянии.
Active and pending Активное и отложенное	The exception is being serviced by the processor and there is a pending exception from the same source. Исключение уже обслуживается процессором и есть отложенное исключение от того же самого источника.

2.3.2 Exception types (Типы исключений)

The exception types are: (Есть следующие типы исключений:)

Reset Reset is invoked on power up or a warm reset. The exception model treats reset as a special form of exception. When reset is asserted, the operation of the processor stops, potentially at any point in an instruction. When reset is deasserted, execution restarts from the address provided by the reset entry in the vector table. Execution restarts as privileged execution in Thread mode.
Сброс, вызыванный подачей питания или кнопкой "горячий" сброс. Модель исключений обрабатывает **Reset**, как специальную форму исключения. Когда выставляется сигнал **Reset**, операции процессора останавливаются, потенциально, в любой момент обработки инструкции на конвеере. Когда сигнал сброса отпускается, выполнение возобновляется с того адреса, что указан в элементе **Reset** таблицы векторов. После рестарта процессор переходит в привилегированный режим **Thread**.

NMI A NonMaskable Interrupt (NMI) can be signalled by a peripheral or triggered by software. This is the highest priority exception other than reset. It is permanently enabled and has a fixed priority of -2. NMIs cannot be:
Немаскируемое прерывание **NMI** может запросить внешнее устройство или оно может быть вызвано программно. Это исключение с самом высоким приоритетом, кроме сброса. Оно всегда разрешено и имеет фиксированный приоритет '-2'. **NMI** не может быть:

- Masked or prevented from activation by any other exception
  Отключено или заблокировано другим исключением
- Preempted by any exception other than Reset.
  Прервано каким либо другим исключением, кроме **Reset**.

Hard fault A hard fault is an exception that occurs because of an error during exception processing, or because an exception cannot be managed by any other exception mechanism. Hard faults have a fixed priority of -1, meaning they have higher priority than any exception with configurable priority.
Hard fault, это исключение аппаратной ошибки, которое происходит из-за ошибки во время обработки исключения, или потому что исключением не может управлять никакой другой механизм исключения. Аппаратная ошибка имеет фиксированный приоритет '-1', это означает, что она имеет более высокий приоритет, чем любое исключение с конфигурируемым приоритетом.

Memory management fault A memory management fault is an exception that occurs because of a memory protection related fault. The fixed memory protection constraints determines this fault, for both instruction and data memory transactions. This fault is used to abort instruction accesses to Execute Never (XN) memory regions.
Ошибка управления памятью - это исключение, которое происходит из-за ошибки, связанной с защитой памяти. Эту ошибку определяют установленные ограничения для защиты памяти, как для инструкций, так и для транзакций в памяти данных. Эта ошибка используется, чтобы прервать выборку инструкций в регионе памяти типа **XN** (*Никогда не Выполнять*).

Bus fault A bus fault is an exception that occurs because of a memory related fault for an instruction or data memory transaction. This might be from an error detected on a bus in the memory system. Шинная ошибка - это исключение, которое происходит из-за ошибки, связанной с памятью, при выборке инструкций или при транзакции в памяти данных. Оно может быть вызвано ошибкой, обнаруженной на шине контроллером памяти.

Usage fault A usage fault is an exception that occurs because of a fault related to instruction execution. This includes:
Пользовательская ошибка - это исключение, которое происходит из-за ошибки, связанной с выполнением команды. Оно включает:

- An undefined instruction (Неопознанную инструкцию)
- An illegal unaligned access (Невыровненный доступ)
- Invalid state on instruction execution
  Неверное состояние после выполнения инструкции
- An error on exception return. (Ошибка возврата из исключения)

The following can cause a usage fault when the core is configured to report them:
Следующее может вызвать пользовательская ошибку при конфигурировании ядра, чтобы сообщить об этом:

- An unaligned address on word and halfword memory access
  Невыровненный адрес слова или полуслова при доступе к памяти
- Division by zero (Деление на ноль)

SVCall A supervisor call (SVC) is an exception that is triggered by the SVC instruction. In an OS environment, applications can use SVC instructions to access OS kernel functions and device drivers.
Вызов супервизора (**SVC**) является исключением, которое вызвано инструкцией **SVC**. В среде **OS** приложения могут использовать инструкции **SVC**, чтобы обратиться к функциям ядра **OS** и драйверам устройств.

PendSV PendSV is an interrupt-driven request for system-level service. In an OS environment, use PendSV for context switching when no other exception is active.
PendSV - это управляемый прерываниями запрос на обслуживание системного уровня. В среде **OS**, используйте **PendSV** для переключения контекста, когда никакое другое исключение не является активным.

SysTick A SysTick exception is an exception the system timer generates when it reaches zero. Software can also generate a SysTick exception. In an OS environment, the processor can use this exception as system tick.
SysTick - это исключение, которое генерирует системный таймер, когда его счетчик достигает нуля. Программное обеспечение также может генерировать исключение **SysTick**. В среде **OS** процессор может использовать это исключение, как системный тактовый сигнал.

Interrupt (IRQ) A interrupt, or IRQ, is an exception signalled by a peripheral, or generated by a software request. All interrupts are asynchronous to instruction execution. In the system, peripherals use interrupts to communicate with the processor.
Прерывание, или **IRQ** - это исключение, которое генерируется внешним устройством или программно. Все прерывания являются асинхронными к выполнению инструкции. В системе внешние устройства используют прерывания, чтобы общаться с процессором.

Table 16. Properties of the different exception types (Свойства различных типов исключений)

Exception number[1]	IRQ number[1]	Exception type	Priority	Vector address or offset[2]	Activation
1	-	Reset	-3, the highest	0x00000004	Asynchronous
2	14	NMI	-2	0x00000008	Asynchronous
3	13	Hard fault	-1	0x0000000C	-
4	12	Memory management fault	Configurable[3]	0x00000010	Synchronous
5	11	Bus fault	Configurable[3]	0x00000014	Synchronous when precise, asynchronous when imprecise
6	10	Usage fault	Configurable[3]	0x00000018	Synchronous
7-10	-	-	-	Reserved	-
11	-5	SVCall	Configurable[3]	0x0000002C	Synchronous
12-13	-	-	-	Reserved	-
14	-2	PendSV	Configurable[3]	0x00000038	Asynchronous
15	-1	SysTick	Configurable[3]	0x0000003C	Asynchronous
16-83	0-67	Interrupt (IRQ)	Configurable[4]	0x00000040 and above[5]	Asynchronous

1. To simplify the software layer, the CMSIS only uses IRQ numbers and therefore uses negative values for exceptions other than interrupts. The IPSR returns the Exception number, see *Interrupt program status register on page 17*.
Чтобы упростить программирование, **CMSIS** использует только номера **IRQ** и, поэтому, использует отрицательные значения для тех исключений, что не являются прерываниями. Инструкция **IPSR** возвращает номер исключения, см. параграф "Регистр статуса прерываний" в разделе 2.1.3.

2. See *Vector table on page 35* for more information.
См. раздел 2.3.4 "Таблица векторов" для получения дополнительной информации.

3. See *System handler priority registers (SHPRx) on page 122*.
См. раздел 4.3.7 "Регистры приоритетов системных обработчиков (**SHPRx**)"

4. See *Interrupt priority registers (NVIC_IPRx) on page 111*.
См. раздел 4.2.7 "Регистр приоритетов прерываний (**NVIC_IPRx**)"

5. Increasing in steps of 4. (Увеличивается с шагом 4.)

For an asynchronous exception, other than reset, the processor can execute another instruction between when the exception is triggered and when the processor enters the exception handler.
Для асинхронного исключения, кроме сброса, процессор может успеть выполнить другую инструкцию между тем моментом, когда вызвано исключение, и тем, когда процессор входит в обработчик исключения.

Privileged software can disable the exceptions that Table 16 on page 33 shows as having configurable priority, see: Привилегированный код может отключить те исключения, что в Таблице 16 указаны как имеющие конфигурируемый приоритет, см.:

- *System handler control and state register (SCB_SHCSR) on page 124*

Раздел 4.3.8 "Регистр управления и состояния системных обработчиков (**SCB_SHCSR**)"

- *Interrupt clear-enable registers (NVIC_ICERx) on page 107*

Раздел 4.2.3 "Регистр выключения прерываний (**NVIC_ICERx**)"

For more information about hard faults, memory management faults, bus faults, and usage faults, see *Section 2.4: Fault handling on page 39*.
Для получения дополнительной информации об аппаратных ошибках, ошибках управления памятью, ошибках шины, и пользовательских ошибках, см. раздел 2.4 **"Обработка ошибок"**.

2.3.3 Exception handlers (Обработчики исключений)

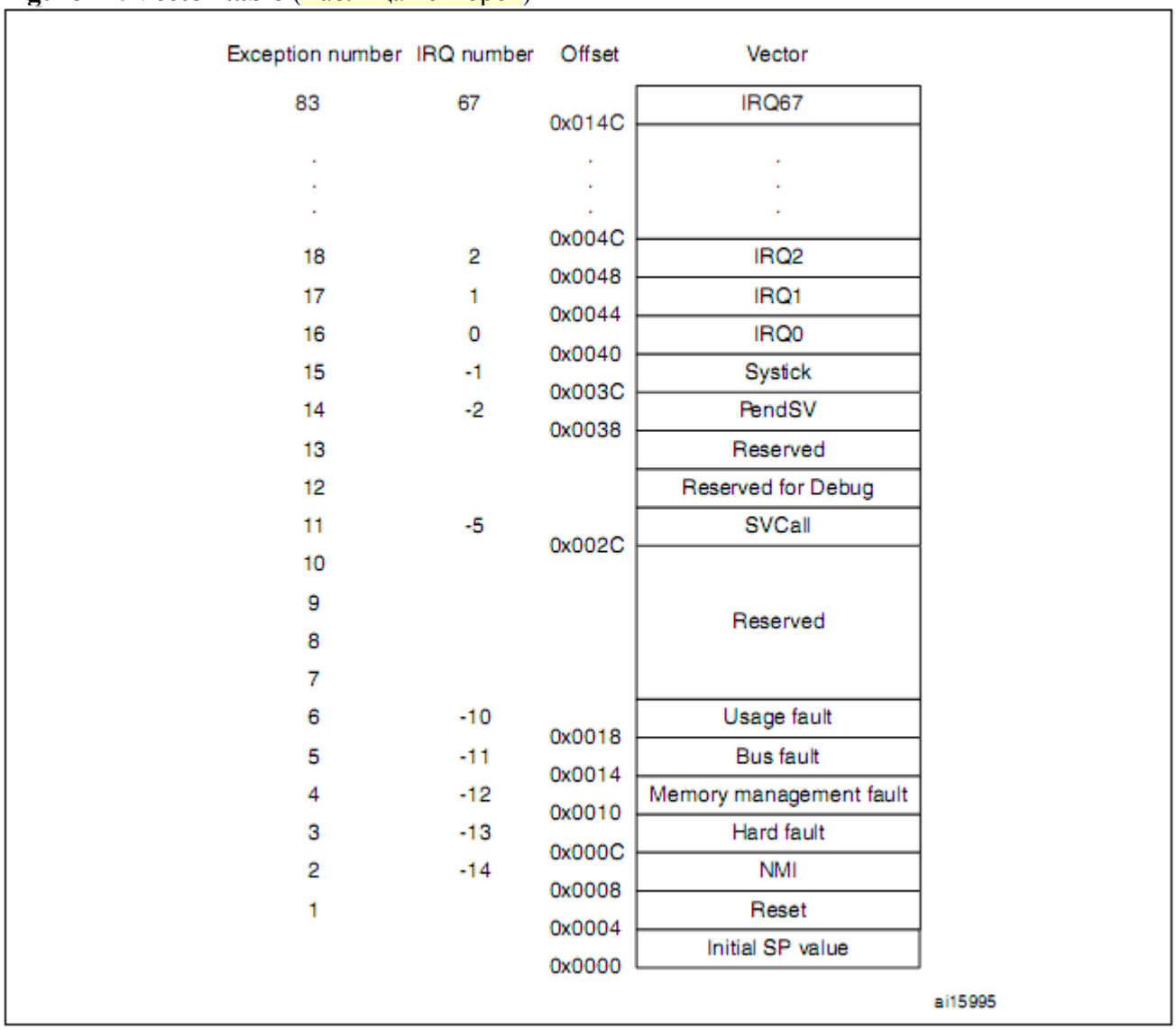
The processor handles exceptions using:
Процессор обрабатывает исключения, используя следующее:

Interrupt Service Routines (ISRs)	Interrupts IRQ0 to IRQ67 are the exceptions handled by ISRs. Прерывания от **IRQ0** до **IRQ67** являются исключениями, которые обрабатываются с помощью **ISR**.
Fault handlers	Hard fault, memory management fault, usage fault, bus fault are fault exceptions handled by the fault handlers. Аппаратные ошибки, ошибки управления памятью, ошибки шины и пользовательские ошибки являются исключениями, которые обрабатываются с помощью обработчиков ошибок.
System handlers	NMI, PendSV, SVCall SysTick, and the fault exceptions are all system exceptions that are handled by system handlers. **NMI**, **PendSV**, **SVCall**, **SysTick** и исключения ошибок - это все системные исключения, которые обрабатываются с помощью системных обработчиков.

2.3.4 Vector table (Таблица векторов)

The vector table contains the reset value of the stack pointer, and the start addresses, also called exception vectors, for all exception handlers. Figure 12 on page 35 shows the order of the exception vectors in the vector table. The least-significant bit of each vector must be 1, indicating that the exception handler is Thumb code.
Таблица векторов содержит значение указателя стека после сброса, и начальные адреса для всех обработчиков исключений, которые называются также векторами исключений. Рисунок 12 показывает порядок векторов исключений в таблице векторов. Младший бит каждого вектора должен быть равен '1', указывая на то, что код обработчика исключений использует набор команд **Thumb**.

Figure 12. Vector table (Таблица векторов)



On system reset, the vector table is fixed at address 0x00000000. Privileged software can write to the VTOR to relocate the vector table start address to a different memory location, in the range 0x00000080 to 0x3FFFFF80, see *Vector table offset register (SCB_VTOR) on page 118*.
После сброса системы таблица векторов находится по фиксированному адресу **0x00000000**. Привилегированный код может записать в регистр **VTOR**, чтобы переместить начало таблицы векторов в другое место в памяти, в диапазоне от **0x00000080** до **0x3FFFFF80**, см. раздел 4.3.3 "Регистр смещения таблицы векторов (**SCB_VTOR**)".

2.3.5 Exception priorities (Приоритеты исключений)

As Table 16 on page 33 shows, all exceptions have an associated priority, with:
Как показывает таблица 16, все исключения имеют ассоциированные с ними приоритеты.

- A lower priority value indicating a higher priority
Более низкое значение приоритета показывает более высокий приоритет
- Configurable priorities for all exceptions except Reset, Hard fault, and NMI.
Все исключения, кроме **Reset**, **Hard fault** и **NMI**, имеют конфигурируемый приоритет.

If software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities see
Если программа не конфигурирует никаких приоритетов, то все исключения с конфигурируемым приоритетом имеет приоритет 0. Для информации о конфигурировании приоритетов исключений смотри:

- *System handler priority registers (SHPRx) on page 122*
Раздел 4.3.7 "Регистры приоритетов системных обработчиков (**SHPRx**)"
- *Interrupt priority registers (NVIC_IPRx) on page 111*
Раздел 4.2.7 "Регистр приоритетов прерываний (**NVIC_IPRx**)"

Configurable priority values are in the range 0-15. This means that the Reset, Hard fault, and NMI exceptions, with fixed negative priority values, always have higher priority than any other exception.
Значения конфигурируемых приоритетов находятся в диапазоне 0-15. Это означает, что исключения **Reset**, **Hard fault** и **NMI**, с фиксированными отрицательными приоритетными значениями, всегда имеют более высокий приоритет, чем любое другое исключение.

For example, assigning a higher priority value to IRQ[0] and a lower priority value to IRQ[1] means that IRQ[1] has higher priority than IRQ[0]. If both IRQ[1] and IRQ[0] are asserted, IRQ[1] is processed before IRQ[0].
Например, присвоение более высокого значения приоритета для прерывания **IRQ[0]** и более низкого значения приоритета для прерывания **IRQ[1]** означает, что **IRQ[1]** имеет более высокий приоритет, чем **IRQ[0]**. Если одновременно будет выставлен запрос на прерывание **IRQ[1]** и **IRQ[0]**, то прерывание **IRQ[1]** будет обработано перед прерыванием **IRQ[0]**.

If multiple pending exceptions have the same priority, the pending exception with the lowest exception number takes precedence. For example, if both IRQ[0] and IRQ[1] are pending and have the same priority, then IRQ[0] is processed before IRQ[1].
Если есть несколько ожидающих запросов исключений с тем же самым приоритетом, то приоритет получит то исключение, что имеет самый низкий номер исключения. Например, если есть два отложенных запроса на прерывание **IRQ[0]** и **IRQ[1]** и они имеют один приоритет, то прерывание **IRQ[0]** будет обработано перед прерыванием **IRQ[1]**.

When the processor is executing an exception handler, the exception handler is preempted if a higher priority exception occurs. If an exception occurs with the same priority as the exception being handled, the handler is not preempted, irrespective of the exception number. However, the status of the new interrupt changes to pending.
Когда процессор выполняет обработку исключения, и приходит запрос от исключений с более высоким приоритетом, то процессор приостанавливает и выгружает текущее исключение. Если приходит запрос от исключения с тем же приоритетом, что у обрабатываемого исключение, то обработчик продолжает работу, независимо от номера этого исключения. Только состояние нового прерывания изменяется на отложенное.

2.3.6 Interrupt priority grouping (Группировка приоритетов исключений)

To increase priority control in systems with interrupts, the NVIC supports priority grouping. This divides each interrupt priority register entry into two fields:
Чтобы усилить контроль над приоритетами в системах с множеством прерываний, **NVIC** поддерживает группировку приоритетов. Она выполняется за счет деления каждого поля приоритета в регистре приоритетов прерываний на два субполя:

- An upper field that defines the group priority
Старшее субполе, которое определяет группу приоритета
- A lower field that defines a subpriority within the group.
И младшее субполе, которое определяет подприоритет в пределах группы.

Only the group priority determines preemption of interrupt exceptions. When the processor is executing an interrupt exception handler, another interrupt with the same group priority as the interrupt being handled does not preempt the handler.
Только приоритет группы определяет, будет ли прервана обработка текущего исключения. Когда процессор выполняет обработку исключения, то другое прерывание с той же самой группой приоритета, не сможет прервать текущий обработчик.

If multiple pending interrupts have the same group priority, the subpriority field determines the order in which they are processed. If multiple pending interrupts have the same group priority and subpriority, the interrupt with the lowest IRQ number is processed first.
Если несколько отложенных прерываний имеют одну группу приоритета, то поле подприоритета определяет порядок, в котором они будут обработаны. Если несколько отложенных прерываний имеют одну группу приоритета и одинаковый подприоритет, то обработка прерываний будет производиться в соответствии с их номерами, чем меньше номер, тем раньше обработка.

For information about splitting the interrupt priority fields into group priority and subpriority, see *Application interrupt and reset control register (SCB_AIRCR) on page 119*.
Для дополнительной информации о разбиении поля приоритета прерывания на субполя группы приоритета и подприоритета, см. раздел 4.3.4 "Регистр прерываний приложения и управления сбросом (**SCB_AIRCR**)".

2.3.7 Exception entry and return (Вход в исключение и выход из него)

Descriptions of exception handling use the following terms:
При описании обработки исключений используются следующие термины:

Preemption Прерывание прерывания	When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled. See *Section 2.3.6: Interrupt priority grouping* for more information about preemption by an interrupt. Когда процессор выполняет обработку исключения, то другое исключение может выгрузить текущий обработчик, если его приоритет выше. См. Раздел 2.3.6 "Группировка приоритетов исключений" для получения дополнительной информации о приоритетном прерывании прерываний. When one exception preempts another, the exceptions are called nested exceptions. See *Exception entry on page 37* more information. Когда одно исключение выгружает другое, то такое исключение называют вложенным. См. параграф "Вход в исключение" в этом разделе.

Return Возврат (из обработчика)	This occurs when the exception handler is completed, and: Это происходит, когда обработчик исключения закончил работу, и: • There is no pending exception with sufficient priority to be serviced Нет никакого отложенного исключения с достаточным приоритетом, который необходимо обслужить • The completed exception handler was not handling a late-arriving exception. Завершенный обработчик исключения не был запущен как "Чуть позже пришедшее" исключение. The processor pops the stack and restores the processor state to the state it had before the interrupt occurred. See *Exception return on page 38* for more information. Процессор выталкивает стек и восстанавливает состояние процессора на тот момент, что был до входа в прерывание. Для получения дополнительной информации см. параграф "Возврат из исключения" в этом разделе.
Tail-chaining "Сесть на хвост"	This mechanism speeds up exception servicing. On completion of an exception handler, if there is a pending exception that meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler. Этот механизм ускоряет обслуживание исключения. По завершению обработки исключения, если есть отложенное исключение, которое отвечает требованиям для входа в исключение, то процедура выталкивания из стека пропускается, и управление переходит к новому обработчику исключения.
Late-arriving Чуть позже пришедшее	This mechanism speeds up preemption. If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher priority exception and initiates the vector fetch for that exception. State saving is not affected by late arrival because the state saved is the same for both exceptions. Therefore the state saving continues uninterrupted. The processor can accept a late arriving exception until the first instruction of the exception handler of the original exception enters the execute stage of the processor. On return from the exception handler of the late-arriving exception, the normal tail-chaining rules apply. Этот механизм ускоряет прерывание прерывания. Если приходит более высоко-приоритетное исключение на стадии сохранения состояния для предыдущего исключения, то процессор сразу передает управление обработчику этого высоко-приоритетного исключения и инициализирует выборку вектора для этого исключения. Процесс сохранения состояния не затрагивается этим событием, так как сохраненное состояние будет одинаковым для обоих исключений. Поэтому процесс сохранения состояния не прерывается. Процессор может принять "Чуть позже пришедшее" исключение, пока первая инструкция обработчика текущего исключения не успела поступить на исполняющую стадию конвеера. По

Exception entry (Вход в исключение)

Exception entry occurs when there is a pending exception with sufficient priority and either:
Вход в исключение происходит, когда есть запрошенное исключение с достаточным приоритетом и одно из следующих условий:

- The processor is in Thread mode (Процессор находится в режиме **Thread**)
- The new exception is of higher priority than the exception being handled, in which case the new exception preempts the original exception.
Новое исключение имеет более высокий приоритет чем обрабатываемое исключение, тогда

новое исключение прерывает оригинальное исключение.

When one exception preempts another, the exceptions are nested.
Когда одно исключение прерывает другое, то оно является вложенным.

Sufficient priority means the exception has more priority than any limits set by the mask registers, see *Exception mask registers on page 18*. An exception with less priority than this is pending but is not handled by the processor.
Достаточный приоритет означает, что исключение имеет более высокий приоритет, чем любые ограничения, установленные регистрами маски, см. параграф "Регистры масок исключений" в разделе 2.1.3. Исключение с приоритетом ниже достаточного выставляется, но не обрабатывается процессором.

When the processor takes an exception, unless the exception is a tail-chained or a late-arriving exception, the processor pushes information onto the current stack. This operation is referred as stacking and the structure of eight data words is referred as stack frame. The stack frame contains the following information:
Когда процессор берет исключение, если это не процедура "Посадки на хвост" или "Чуть позже пришедшее" исключение, то он заталкивает информацию на текущий стек. Эта операция называется **stacking**, а структура из восьми слов данных называется "стековым фреймом". "Стековый фрейм" содержит следующую информацию:

R0-R3, R12
Return address
PSR
LR.

Immediately after stacking, the stack pointer indicates the lowest address in the stack frame. Unless stack alignment is disabled, the stack frame is aligned to a double-word address. If the STKALIGN bit of the Configuration Control Register (CCR) is set to 1, stack align adjustment is performed during stacking.
Немедленно, после сохранения в стеке, указатель стека указывает на самый младший адрес в "Стековом фрейме". Если опция выравнивания стека не заблокирована, то адрес "Стекового фрейма" выровнен по двойному слову. Если бит **STKALIGN** регистра **CCR** (*Configuration Control Register*) установлен в 1, то опция выравнивания стека включена, и оно выполняется во время операции сохранения в стеке.

The stack frame includes the return address. This is the address of the next instruction in the interrupted program. This value is restored to the PC at exception return so that the interrupted program resumes.
"Стековый фрейм" включает адрес возврата. Это адрес следующей инструкции в прерванной программе. Это значение восстановливает регистр **PC** при возвращении из исключения так, чтобы прерванная программа возобновила работу.

In parallel to the stacking operation, the processor performs a vector fetch that reads the exception handler start address from the vector table. When stacking is complete, the processor starts executing the exception handler. At the same time, the processor writes an EXC_RETURN value to the LR. This indicates which stack pointer corresponds to the stack frame and what operation mode the was processor was in before the entry occurred.
Параллельно с операцией сохранения в стеке, процессор выполняет выборку вектора, то есть читает начальный адрес обработчика исключения в таблице векторов. Когда сохранение в стеке закончено, процессор начинает выполнять обработчик исключения. В то же самое время, процессор записывает значение **EXC_RETURN** в регистр **LR**. Оно показывает, какой указатель стека соответствует "Стековому фрейму" и в каком режиме работал процессор до того, как произошел вход в исключение.

If no higher priority exception occurs during exception entry, the processor starts executing the exception handler and automatically changes the status of the corresponding pending interrupt to active.
Если, в процессе входа в исключение, не было запроса на исключение с более высоким приоритетом, процессор начинает выполнять текущий обработчик и автоматически изменяет его состояние с отложенного на активное.

If another higher priority exception occurs during exception entry, the processor starts executing the exception handler for this exception and does not change the pending status of the earlier exception. This is the late arrival case.
Если, в процессе входа в исключение, пришло новое исключение с более высоким приоритетом, то процессор начинает выполнять новый обработчик и не меняет статус ранее запрошенного исключения. Это случай "Чуть позже пришедшего" исключения.

Exception return (Выход из исключения)

Exception return occurs when the processor is in Handler mode and executes one of the following instructions to load the EXC_RETURN value into the PC: Возвращение из исключения происходит, когда процессор находится в режиме **Handler** и выполняет одну из следующих инструкций, чтобы загрузить значение **EXC_RETURN** в регистр **PC**:

- A POP instruction that includes the PC

Инструкция **POP** (восстанавливает регистры, включая **PC**)

- A BX instruction with any register.

Инструкция **BX** с любым регистром в качестве аргумента

- An LDR or LDM instruction with the PC as the destination

Инструкция **LDR** или **LDM** с аргументом **PC** в качестве цели

EXC_RETURN is the value loaded into the LR on exception entry. The exception mechanism relies on this value to detect when the processor has completed an exception handler. The lowest four bits of this value provide information on the return stack and processor mode. Table 17 shows the EXC_RETURN[3:0] values with a description of the exception return behavior.
EXC_RETURN - это значение, загруженное в **LR** при входе в исключение. Механизм исключений полагается на это значение, чтобы обнаружить, когда процессор завершил обработку исключения. Младшие четыре бита этого значения предоставляют информацию о стеке и режиме процессора по возвращению. Таблица 17 показывает значения **EXC_RETURN[3:0]** и описание поведения при возвращении из исключения.

The processor sets EXC_RETURN bits[31:4] to 0xFFFFFFF. When this value is loaded into the PC it indicates to the processor that the exception is complete, and the processor initiates the exception return sequence. Процессор устанавливает биты **EXC_RETURN[31:4]** в **0xFFFFFFF**. Когда это значение загружено в **PC**, это указывает процессору, что исключение закончено, и процессор инициализирует последовательность возвращения из исключения.

Table 17. Exception return behavior (Поведение при возвращении из исключения)

EXC_RETURN[3:0]	Description
0bxxx0	Reserved.
0b0001	Return to Handler mode. (Возврат в режим **Handler**) Exception return gets state from MSP. (Восстановление состояния из **MSP**) Execution uses MSP after return. (Использование **MSP** после возврата.)
0b0011	Reserved.

0b01x1	Reserved.
0b1001	Return to Thread mode. (Возврат в режим **Thread**) Exception return gets state from MSP. (Восстановление состояния из **MSP**) Execution uses MSP after return. (Использование **MSP** после возврата.)
0b1101	Return to Thread mode. (Возврат в режим **Thread**) Exception return gets state from PSP. (Восстановление состояния из **PSP**) Execution uses PSP after return. (Использование **PSP** после возврата.)
0b1x11	Reserved.

2.4 Fault handling (Обработка ошибок)

2.4.1 Fault types (Типы ошибок)
2.4.2 Fault escalation and hard faults (Усиление ошибки и аппаратные ошибки)
2.4.3 Fault status registers and fault address registers
Регистры статуса ошибок и Регистр адреса ошибки
2.4.4 Lockup (Блокировка)

Faults are a subset of the exceptions, see Exception model on page 32. The following generate a fault: Ошибки - это подмножество исключений, см. раздел 2.3 "**Модель исключений**". Следующие условия генерируют ошибку:

- A bus error on: (Ошибка шины при:)
  - An instruction fetch or vector table load
    Выборке инструкции или при загрузке вектора из таблицы
  - A data access (Доступе к данным)
- An internally-detected error such as an undefined instruction or an attempt to change state with a BX instruction
  Внутренне-обнаруженная ошибка, такая как неопознанная инструкция или попытка изменить состояние инструкцией **BX**
- Attempting to execute an instruction from a memory region marked as Non-Executable (XN).
  Попытка выполнить инструкцию из области памяти, помеченной как **XN** (*Non-Executable*).

2.4.1 Fault types (Типы ошибок)

Table 18 shows the types of fault, the handler used for the fault, the corresponding fault status register, and the register bit that indicates that the fault has occurred. See *Configurable fault status register (SCB_CFSR) on page 126* for more information about the fault status registers.
Таблица 18 показывает: типы ошибок, обработчик, используемый для ошибки, соответствующий регистр статуса ошибки, и бит регистра, который указывает, что ошибка произошла. См. раздел 4.3.9 "Регистр статуса конфигурируемых ошибок (**SCB_CFSR**)" для получения дополнительной информации о регистрах статуса ошибки.

Table 18. Faults (Ошибки)

Fault	Handler	Bit name	Fault status register
Bus error on a vector read Ошибка шины при чтении вектора	Hard fault	VECTTBL	*Hard fault status register (SCB_HFSR) on page 129* 4.3.10 "Регистр статуса аппаратных ошибок **(SCB_HFSR)**"
Fault escalated to a hard fault Усиление ошибки до аппаратной		FORCED	
Bus error: (Ошибка шины:)	Bus fault	-	-
During exception stacking В процессе сохранения в стеке для исключения		STKERR	*Configurable fault status register (SCB_CFSR) on page 126* 4.3.9 "Регистр статуса конфигурируемых ошибок **(SCB_CFSR)**"
During exception unstacking В процессе восстановления из стека после исключения		UNSTKERR	
During instruction prefetch В процессе предварительной выборки инструкции		IBUSERR	
Precise data bus error Определенная ошибка шины данных		PRECISERR	
Imprecise data bus error Неопределенная ошибка шины данных		IMPRECISERR	
Attempt to access a coprocessor Попытка доступа к сопроцессору	Usage fault	NOCP	*Configurable fault status register (SCB_CFSR) on page 126* 4.3.9 "Регистр статуса конфигурируемых ошибок **(SCB_CFSR)**"
Undefined instruction Неопознанная инструкция		UNDEFINSTR	
Attempt to enter an invalid instruction set state[1] Попытка войти в некорректное состояние для текущего набора каманд		INVSTATE	
Invalid EXC_RETURN value Некорректное значение **EXC_RETURN**		INVPC	
Illegal unaligned load or store Загрузка/сохранение с некорректным выравниванием		UNALIGNED	
Divide By 0 (Деление на ноль)		DIVBYZERO	

1. Attempting to use an instruction set other than the Thumb instruction set.
Попытка использовать инструкцию из набора команд отличного от **Thumb**.

2.4.2 Fault escalation and hard faults (Усиление ошибки и аппаратные ошибки)

All faults exceptions except for hard fault have configurable exception priority, see *System handler priority registers (SHPRx) on page 122*. Software can disable execution of the handlers for these faults, see *System handler control and state register (SCB_SHCSR) on page 124*.
Все исключения ошибок, кроме аппаратной ошибки, имеют конфигурируемый приоритет, см. раздел 4.3.7 "Регистры приоритетов системных обработчиков (SHPRx)". Программа может отключить выполнение обработки таких ошибок, см. раздел 4.3.8 "Регистр управления и состояния системных обработчиков (**SCB_SHCSR**)".

Usually, the exception priority, together with the values of the exception mask registers, determines whether the processor enters the fault handler, and whether a fault handler can preempt another fault handler, as described in Section 2.3: Exception model on page 32.
Обычно, приоритет исключения, вместе со значениями регистров масок исключений, определяют, войдет ли процессор в обработчик ошибки, и может ли обработчик ошибки выгрузить другой обработчик ошибки, как это описано в разделе 2.3 "**Модель исключений**".

In some situations, a fault with configurable priority is treated as a hard fault. This is alled priority escalation, and the fault is described as escalated to hard fault. Escalation to hard fault occurs when:
В некоторых ситуациях, ошибка с конфигурируемым приоритетом рассматривается как аппаратная ошибка. Это называется усилением ошибки, и, при описании ошибки, говорится об усилении до аппаратной ошибки. Усиление до аппаратной ошибки происходит, когда:

- A fault handler causes the same kind of fault as the one it is servicing. This escalation to hard fault occurs because a fault handler cannot preempt itself because it must have the same priority as the current priority level. В обработчике ошибки происходит тот же самый тип ошибки, который сейчас обслуживается. Это усиление до аппаратной ошибки происходит потому, что обработчик ошибки не может прервать сам себя, ввиду одинаковых приоритетов.

- A fault handler causes a fault with the same or lower priority as the fault it is servicing. This is because the handler for the new fault cannot preempt the currently executing fault handler. В обработчике ошибки происходит ошибка с тем же самым или более низким приоритетом. Это усиление происходит потому, что обработчик для новой ошибки не может прервать текущий обработчик ошибки.

- An exception handler causes a fault for which the priority is the same as or lower than the currently executing exception. В обработчике исключения происходит ошибка с тем же самым или более низким приоритетом.

- A fault occurs and the handler for that fault is not enabled.
Происходит ошибка, обработчик которой отключен.

If a bus fault occurs during a stack push when entering a bus fault handler, the bus fault does not escalate to a hard fault. This means that if a corrupted stack causes a fault, the fault handler executes even though the stack push for the handler failed. The fault handler operates but the stack contents are corrupted. Если происходит шинная ошибка во время сохранения в стеке, при входе в обработчик ошибки шины, то шинная ошибка не усиливается до аппаратной ошибки. Это означает, что если эту ошибку вызывает разрушенный стек, то обработчик ошибки продолжает выполняется даже при том, что операция сохранения в стеке для этого обработчика была некорректной. Обработчик ошибки работает, но содержимое стека разрушено.

Only Reset and NMI can preempt the fixed priority hard fault. A hard fault can preempt any exception other than Reset, NMI, or another hard fault. Только **Reset** и **NMI** могут прервать

обработчик аппаратной ошибки с фиксированным приоритетом. Аппаратная ошибка может прервать любое исключение кроме **Reset**, **NMI**, или обработчика другой аппаратная ошибки.

2.4.3 Fault status registers and fault address registers
Регистры статуса ошибок и Регистр адреса ошибки

The fault status registers indicate the cause of a fault. For bus faults and memory management faults, the fault address register indicates the address accessed by the operation that caused the fault, as shown in Table 19. Регистры статуса ошибок указывают причину ошибки. Для ошибок шины и ошибок управления памятью, регистр адресов ошибок указывает адрес, обращение к которому вызвало ошибку, как показано в таблице 19.

Table 19. Fault status and fault address registers (Регистры статуса и адресов ошибок)

Handler	Status register name	Address register name	Register description
Hard fault	HFSR	-	*Hard fault status register (SCB_HFSR) on page 129* 4.3.10 "Регистр статуса аппаратных ошибок (**SCB_HFSR**)"
Memory management fault	MMFSR	MMFAR	*Configurable fault status register (SCB_CFSR) on page 126* 4.3.9 "Регистр статуса конфигурируемых ошибок (**SCB_CFSR**)". *Memory management fault address register (SCB_MMFAR) on page 130* (4.3.11 "Регистр адреса ошибки управления памятью (**SCB_MMFAR**)")
Bus fault	BFSR	BFAR	*Configurable fault status register (SCB_CFSR) on page 126* 4.3.9 "Регистр статуса конфигурируемых ошибок (**SCB_CFSR**)". *Bus fault address register (SCB_BFAR) on page 130* 4.3.12 "Регистр адреса ошибки шины (**SCB_BFAR**)"
Usage fault	UFSR	-	*Configurable fault status register (SCB_CFSR) on page 126* 4.3.9 "Регистр статуса конфигурируемых ошибок (**SCB_CFSR**)".

2.4.4 Lockup (Блокировка)

The processor enters a lockup state if a hard fault occurs when executing the NMI or hard fault handlers. When the processor is in lockup state it does not execute any instructions. The processor remains in lockup state until either:
Процессор входит в состояние блокировки, если Аппаратная ошибка происходит при выполнении обработчика **NMI** или Аппаратной ошибки. Когда процессор находится в состоянии блокировки, он не выполняет никаких инструкций. Процессор остается в состоянии блокировки до одного из следующих событий:

- It is reset (Произошел сброс)
- An NMI occurs (Случилось исключение **NMI**)

If lockup state occurs from the NMI handler a subsequent NMI does not cause the processor to leave lockup state.
Если состояние блокировки происходит из-за обработчика **NMI**, то последующее исключение **NMI** не может деблокировать процессор.

2.5 Power management (Управление потреблением)

2.5.1 Entering sleep mode (Вход в режим сна)
2.5.2 Wakeup from sleep mode (Пробуждение из режима сна)
2.5.3 The external event input (Вход внешнего события)
2.5.4 Power management programming hints
Рекомендации по программированию управлением потреблением

The STM32 and Cortex-M3 processor sleep modes reduce power consumption:
Режимы сна для процессоров **STM32** и **Cortex-M3** уменьшают общее потребление.

- Sleep mode stops the processor clock. All other system and peripheral clocks may still be running. Режим сна останавливает тактовый сигнал процессора. Все другие системные и периферийные тактовые сигналы все еще могут выполняться.
- Deep sleep mode stops most of the STM32 system and peripheral clocks. At product level, this corresponds to either the Stop or the Standby mode. For more details, please refer to the “Power modes” Section in the STM32 reference manual.
Режим глубокого сна останавливает большинство системных и периферийных тактовых сигналов **STM32**. На уровне изделия это соответствует режиму **Stop** (*Остановка*) или **Standby** (*Ожидание*). За дополнительной информацией, пожалуйста, обратитесь к разделу "**Power modes**" в документе "***STM32 reference manual***".

The SLEEPDEEP bit of the SCR selects which sleep mode is used, see *System control register (SCB_SCR) on page 120*. For more information about the behavior of the sleep modes see the STM32 product reference manual.
Бит **SLEEPDEEP** регистра **SCR** выбирает, какой режим сна используется, см. раздел 4.3.5 "Регистр управления системой (SCB_SCR)". Для получения дополнительной информации о поведении режимов ожидания см. документ "***STM32 reference manual***".

This section describes the mechanisms for entering sleep mode, and the conditions for waking up from sleep mode. Этот раздел описывает механизмы входа в режим сна, и условия для пробуждения из этого режима.

2.5.1 Entering sleep mode (Вход в режим сна)

This section describes the mechanisms software can use to put the processor into sleep mode.
Этот раздел описывает механизмы, которые может использовать программа, чтобы поместить процессор в режим сна.

The system can generate spurious wakeup events, for example a debug operation wakes up the processor. Therefore software must be able to put the processor back into sleep mode after such an event. A program might have an idle loop to put the processor back to sleep mode.
Система может генерировать фиктивные события пробуждения, например операция отладки разбудит процессор. Поэтому программа должна быть в состоянии вернуть процессор в режим сна после такого события. Программа может иметь пустой (*idle*) цикл, чтобы вернуть процессор в режим сна.

Wait for interrupt (Ожидание прерывания)

The wait for interrupt instruction, WFI, causes immediate entry to sleep mode. When the processor executes a WFI instruction it stops executing instructions and enters sleep mode. See WFI on page 103 for more information.
Инструкция ожидания прерывания **WFI** вызывает немедленный вход в режим сна. Когда

процессор выполняет инструкцию **WFI**, он прекращает выполнять инструкции и входит в режим сна. См. описание инструкции **WFI** для получения дополнительной информации.

Wait for event (Ожидание события)

The wait for event instruction, WFE, causes entry to sleep mode conditional on the value of an one-bit event register. When the processor executes a WFE instruction, it checks this register:
Инструкция ожидания события **WFE** вызывает условный вход в режим сна, который зависит от значения однобитного регистра события. Когда процессор выполняет инструкцию **WFE**, он проверяет этот регистр:

- If the register is 0 the processor stops executing instructions and enters sleep mode
Если значение регистра равно '0', то процессор останавливает выполнение инструкций и входит в режим сна
- If the register is 1 the processor clears the register to 0 and continues executing instructions without entering sleep mode.
Если значение регистра равно '1', то процессор очищает регистр в '0' и продолжает выполнение инструкций без входа в режим сна

See WFE on page 102 for more information.
См. описание инструкции **WFE** для получения дополнительной информации.

If the event register is 1, this indicate that the processor must not enter sleep mode on execution of a WFE instruction. Typically, this is because an external event signal is asserted, or a processor in the system has executed an SEV instruction, see SEV on page 101. Software cannot access this register directly.
Если значение регистра события равно '1', то это указывает, что процессор не должен входить в режим сна при выполнении инструкции **WFE**. Как правило, это происходит потому что был выставлен сигнал внешнего события, или процессор в системе выполнил инструкцию **SEV**, см. описание инструкции **SEV**. Программа не может обратиться к этому регистру непосредственно.

Sleep-on-exit (Засыпание по выходу)

If the SLEEPONEXIT bit of the SCR is set to 1, when the processor completes the execution of an exception handler it returns to Thread mode and immediately enters sleep mode. Use this mechanism in applications that only require the processor to run when an exception occurs.
Если бит **SLEEPONEXIT** в регистре **SCR** установлен в '1', когда процессор завершает выполнение обработчика исключения, он возвращается в режим **Thread** и немедленно входит в режим сна. Используйте этот механизм в приложениях, которые требуют, чтобы процессор выполнял код только тогда, когда происходит исключение.

2.5.2 Wakeup from sleep mode (Пробуждение из режима сна)

The conditions for the processor to wakeup depend on the mechanism that cause it to enter sleep mode. Условия для пробуждения процессора зависят от механизма, который вызвал вход в режим сна.

Wakeup from WFI or sleep-on-exit (Пробуждение от WFI или условия sleep-on-exit)

Normally, the processor wakes up only when it detects an exception with sufficient priority to cause exception entry. Обычно, процессор просыпается только тогда, когда он обнаруживает исключение с достаточным приоритетом, чтобы вызвать вход в исключение.

Some embedded systems might have to execute system restore tasks after the processor wakes up, and before it executes an interrupt handler. To achieve this set the PRIMASK bit to 1 and the FAULTMASK bit to 0. If an interrupt arrives that is enabled and has a higher priority than current exception priority, the processor wakes up but does not execute the interrupt handler until the processor sets PRIMASK to zero. For more information about PRIMASK and FAULTMASK see Exception mask registers on page 18.
Некоторым встроенным системам, возможно, придется выполнить системные задачи восстановления после того, как процессор просыпается, и прежде, чем он выполнит программу обработки прерывания. Чтобы выполнить это, установите бит **PRIMASK** в '1', а бит **FAULTMASK** в '0'. Если приходит прерывание, которое разрешено и имеет более высокий приоритет, чем приоритет текущего исключения, то процессор просыпается, но не выполняет программу обработки прерывания, пока процессор не обнулит **PRIMASK**. Для получения дополнительной информации о **PRIMASK** и **FAULTMASK** см. параграф "Регистры масок исключений" в разделе 2.1.3.

Wakeup from WFE (Пробуждение от WFE)

The processor wakes up if: (Процессор просыпается если:)

- it detects an exception with sufficient priority to cause exception entry
он обнаруживает исключение с достаточным приоритетом, чтобы вызвать вход в исключение
- it detects an external event signal, see *The external event input on page 43*
он обнаруживает сигнал внешнего события, см. раздел 2.5.3 "Вход внешнего события"

In addition, if the SEVONPEND bit in the SCR is set to 1, any new pending interrupt triggers an event and wakes up the processor, even if the interrupt is disabled or has insufficient priority to cause exception entry. For more information about the SCR see *System control register (SCB_SCR) on page 120*.
Кроме того, если бит **SEVONPEND** в регистре **SCR** установлен в '1', то любое новое выставленное прерывание вызывает событие и будит процессор, даже если прерывание выключено или имеет недостаточный приоритет, чтобы вызвать вход в исключение. Для получения дополнительной информации о **SCR** см. раздел 4.3.5 "Регистр управления системой (**SCB_SCR**)".

2.5.3 The external event input (Вход внешнего события)

The processor provides an external event input signal. This signal can be generated by the up to 16 external input lines, by the PVD, RTC alarm or by the USB wakeup event, configured through the external interrupt/event controller (EXTI).
Процессор предоставляет входы для сигнала внешнего события. Этот сигнал может быть сгенерирован следующим: внешними входными линиями (до 16-ти), детектором напряжения **PVD**, сигналом тревоги часов реального времени **RTC** или событием пробуждения от **USB**, сконфигурированных с помощью контроллера внешних прерываний/событий (**EXTI**).

This signal can wakeup the processor from WFE, or set the internal WFE event register to one to indicate that the processor must not enter sleep mode on a later WFE instruction, see *Wait for event on page 42*. Fore more details please refer to the *STM32 reference manual*, section *4.3 Low power modes*.
Этот сигнал может пробуждать процессор, который заснул от **WFE**, или ставить внутренний регистр события для **WFE** в '1', чтобы указать, что процессор не должен входить в режим сна от последующей инструкции **WFE**, см., параграф "Ожидание события". Для дополнительной информации, пожалуйста, обратитесь к документу "***STM32 reference manual***", в раздел 4.3 "**Low power modes**".

2.5.4 Power management programming hints
Рекомендации по программированию управлением потреблением

ANSI C cannot directly generate the WFI and WFE instructions. The CMSIS provides the following intrinsic functions for these instructions:
Стандарт **ANSI C** не может непосредственно генерировать инструкции **WFI** И **WFE**. Библиотека **CMSIS** обеспечивает следующие встроенные функции для этих инструкций:

```
void __WFE(void) // Wait for Event
void __WFE(void) // Wait for Interrupt
```

3 The Cortex-M3 instruction set

Этот раздел пока не переведен.

4 Core peripherals (Периферия ядра)

4.1 About the STM32 core peripherals (О периферии ядра для **STM32**)
4.2 Nested vectored interrupt controller (NVIC)
(Контроллер вложенных векторов прерываний)
4.3 System control block (SCB) (Блок регистров управления системой)
4.4 SysTick timer (STK) (Системный таймер)

4.1 About the STM32 core peripherals (О периферии ядра для STM32)

The address map of the Private peripheral bus (PPB) is:
Адресное пространство Частной Периферийной Шины (**PPB**):

Table 33. STM32 core peripheral register regions

Address	Core peripheral	Description
0xE000E010-0xE000E01F	System timer	Table 41 on page 135
0xE000E100-0xE000E4EF	Nested vectored interrupt controller	Table 37 on page 114
0xE000ED00-0xE000ED3F	System control block	Table 40 on page 131
0xE000ED90-0xE000ED93	MPU type register Регистр типа MPU (модуль защиты памяти)	Reads as zero, indicating no MPU is implemented(1) При чтении дает нуль, что показывает отсутствие **MPU**
0xE000EF00-0xE000EF03	Nested vectored interrupt controller	Table 37 on page 114

1. Software can read the MPU Type Register at 0xE000ED90 to test for the presence of a memory protection unit (MPU).
Программа может читать регистр типа **MPU** по адресу **0xE000ED90**, чтобы проверить наличие модуля защиты памяти.

In register descriptions: (При описании регистров:)

- The required privilege gives the privilege level required to access the register, as follows:
Уровень привилегий, необходимый при обращении к регистру, указывается следующим образом:

`Privileged` Only privileged software can access the register.
Только код, работающий в привилегерованном режиме, может иметь доступ к регистру.

`Unprivileged` Both unprivileged and privileged software can access the register.
Любой код может иметь доступ к регистру.

4.2 Nested vectored interrupt controller (NVIC)
Контроллер вложенных векторов прерываний

4.2.1 The CMSIS mapping of the Cortex-M3 NVIC registers
Как **CMSIS** отображает регистры **Cortex-M3 NVIC**
4.2.2 Interrupt set-enable registers (NVIC_ISERx) (Регистр разрешения прерываний)
4.2.3 Interrupt clear-enable registers (NVIC_ICERx) (Регистр выключения прерываний)
4.2.4 Interrupt set-pending registers (NVIC_ISPRx)
(Регистры установки запроса на прерывание)
4.2.5 Interrupt clear-pending registers (NVIC_ICPRx)
(Регистр очистки флагов прерываний)
4.2.6 Interrupt active bit registers (NVIC_IABRx)
(Регистр битов активности прерываний)
4.2.7 Interrupt priority registers (NVIC_IPRx) (Регистр приоритетов прерываний)
4.2.8 Software trigger interrupt register (NVIC_STIR)
(Регистр программного запуска прерываний)
4.2.9 Level-sensitive and pulse interrupts
(Чувствительность прерываний к уровню и фронту)
4.2.10 NVIC design hints and tips (Работа с **NVIC**, рекомендации и советы)
4.2.11 NVIC register map (Карта регистров **NVIC**)

This section describes the Nested Vectored Interrupt Controller (NVIC) and the registers it uses. The NVIC supports: Этот раздел описывает Контроллер Вложенных Векторов Прерываний (**NVIC**) и регистры, которые он использует. **NVIC** поддерживает :

- up to 68 interrupts (до 68 прерываний)
- A programmable priority level of 0-15 for each interrupt. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority
Программируемые уровни приоритетов, от 0 до 15 для каждого прерывания. Высший уровень соответствует низшему приоритету, так уровень 0 - это высший приоритет прерывания.
- Level and pulse detection of interrupt signals
Детектирование прерывания по уровню или перепаду сигнала
- Dynamic reprioritization of interrupts (Динамическую смену приоритетов прерываний)
- Grouping of priority values into group priority and subpriority fields
Группировку значений приоритетов по группам и субгруппам
- Interrupt tail-chaining (Функцию "посадки на хвост" предыдущему прерыванию)
- An external Non-maskable interrupt (NMI) (Внешние немаскируемые прерывания)

The processor automatically stacks its state on exception entry and unstacks this state on exception exit, with no instruction overhead. This provides low latency exception handling.
Процессор автоматически сохраняет в стеке свое состояние при входе в исключение и восстанавливает это состояние из стека при выходе из исключения, без дополнительных инструкций. Это обеспечивает низкую задержку обработки исключений.

The hardware implementation of the NVIC registers is:
NVIC включает в себя следующие регистры:

4.2.1 The CMSIS mapping of the Cortex-M3 NVIC registers
Как CMSIS отображает регистры Cortex-M3 NVIC

To improve software efficiency, the CMSIS simplifies the NVIC register presentation. In the CMSIS: Чтобы улучшить эффективность программирования, **CMSIS** упрощает представление регистров **NVIC**. В **CMSIS**:

- The Set-enable, Clear-enable, Set-pending, Clear-pending and Active Bit registers map to arrays of 32-bit integers, so that:
Регистры разрешения и запрета прерываний, установки и очистки флагов прерываний, и битов активности, отражаются на массивы 32-х битных целых так, чтобы:
  - The array ISER[0] to ISER[2] corresponds to the registers ISER0-ISER2
Массив **ISER[0]...ISER[2]** соответствует регистрам **ISER0-ISER2**
  - The array ICER[0] to ICER[2] corresponds to the registers ICER0-ICER2
  - The array ISPR[0] to ISPR[2] corresponds to the registers ISPR0-ISPR2
  - The array ICPR[0] to ICPR[2] corresponds to the registers ICPR0-ICPR2
  - The array IABR[0] to IABR[2] corresponds to the registers IABR0-IABR2.
- The 8-bit fields of the Interrupt Priority Registers map to an array of 8-bit integers, so that the array IP[0] to IP[67] corresponds to the registers IPR0-IPR67, and the array entry IP[n] holds the interrupt priority for interrupt n.
8-ми битовые поля регистров Приоритетов Прерываний отображаются на массив 8-ми битовых целых чисел, так, чтобы массив **IP[0]...[67]** соответствовал регистрам **IPR0-IPR67**, а элемент массива **IP[n]** содержит приоритет для прерывания **n**.

The CMSIS provides thread-safe code that gives atomic access to the Interrupt Priority Registers. For more information see the description of the NVIC_SetPriority function in NVIC programming hints on page 113. Table 34 shows how the interrupts, or IRQ numbers, map onto the interrupt registers and corresponding CMSIS variables that have one bit per interrupt.
CMSIS обеспечивает атомарный доступ к Регистрам Приоритетов Прерываний, который является безопасным при использовании мульти-тредов. Для получения дополнительной информации см. описание функции **NVIC_SetPriority()** в советах по программированию **NVIC**. Таблица 34 показывает, как номер прерывания отображается на регистры прерывания и соответствующие переменные **CMSIS**, которые содержат по одному биту на прерывание.

Table 34. Mapping of interrupts to the interrupt variables
Отображение прерываний на свои переменные

Interrupts	CMSIS array elements(1)				
	Set-enable	Clear-enable	Set-pending	Clear-pending	Active Bit
0-31	ISER[0]	ICER[0]	ISPR[0]	ICPR[0]	IABR[0]
32-63	ISER[1]	ICER[1]	ISPR[1]	ICPR[1]	IABR[1]
64-67	ISER[2]	ICER[2]	ISPR[2]	ICPR[2]	IABR[2]

1. Each array element corresponds to a single NVIC register, for example the element ICER[1] corresponds to the ICER1 register.
Каждый элемент массива соответствует единственному регистру **NVIC**, например, элемент **ICER[1]** соответствует регистру **ICER1**.

4.2.2 Interrupt set-enable registers (NVIC_ISERx)
Регистр разрешения прерываний

Address offset: 0x00 - 0x0B
Reset value: 0x0000 0000
Required privilege: Privileged

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SETENA[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETENA[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 SETENA[31:0]: Interrupt set-enable bits. (Биты разрешения прерываний.)

Write (При записи):

`0:` No effect (Нет эффекта)

`1:` Enable interrupt (Разрешает прерывание)

Read (При чтении):

`0:` Interrupt disabled (Прерывание отключено)

`1:` Interrupt enabled. (Прерывание разрешено)

See *Table 34: Mapping of interrupts to the interrupt variables on page 105* for the correspondence of interrupts to each register bit.
См. Таблицу 34: "Отображение прерываний на свои переменные" при поиске соответствия прерывания каждому биту регистра.

If a pending interrupt is enabled, the NVIC activates the interrupt based on its priority. If an interrupt is not enabled, asserting its interrupt signal changes the interrupt state to pending, but the NVIC never activates the interrupt, regardless of its priority.
Если запрошенное прерывание разрешено, то **NVIC** активизирует прерывание в соответствии с его приоритетом. Если прерывание отключено, то сигнал, соответствующий запросу на прерывание, изменяет состояние прерывания, но **NVIC** никогда не активизирует такое прерывание, независимо от его приоритета.

4.2.3 Interrupt clear-enable registers (NVIC_ICERx)
Регистр выключения прерываний

Address offset: 0x00 - 0x0B
Reset value: 0x0000 0000
Required privilege: Privileged

The ICER0-ICER2 registers disable interrupts, and show which interrupts are enabled.
Регистры **ICER0-ICER2** отключают прерывания и показывают, какие из прерываний сейчас отключены.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLRENA[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRENA[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 CLRENA[31:0]: Interrupt clear-enable bits. (Биты отключения прерываний)

Write (При записи):

0: No effect (Нет эффекта)

1: Disable interrupt (Отключает прерывание)

Read (При чтении):

0: Interrupt disabled (Прерывание отключено)

1: Interrupt enabled. (Прерывание разрешено)

See *Table 34: Mapping of interrupts to the interrupt variables on page 105* for the correspondence of interrupts to each register bit.
См. Таблицу 34: "Отображение прерываний на свои переменные" при поиске соответствия прерывания каждому биту регистра.

4.2.4 Interrupt set-pending registers (NVIC_ISPRx)
Регистры установки запроса на прерывание

Address offset: 0x00 - 0x0B
Reset value: 0x0000 0000
Required privilege: Privileged

The ISPR0-ISPR2 registers force interrupts into the pending state, and show which interrupts are pending. Регистры **ISPR0-ISPR2** переводят прерывания в состояние запроса, и показывают, какое из прерываний находится в состоянии запроса.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SETPEND[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETPEND[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 SETPEND[31:0]: Interrupt set-pending bits

Write (При записи):

0: No effect (Нет эффекта)

1: Changes interrupt state to pending (Ставит флаг запроса прерывания)

Read (При чтении):

0: Interrupt is not pending (Нет состояния запроса прерывания)

1: Interrupt is pending (Есть состояние запроса прерывания)

See *Table 34: Mapping of interrupts to the interrupt variables on page 105* for the correspondence of interrupts to each register bit.
См. Таблицу 34: "Отображение прерываний на свои переменные" при поиске соответствия прерывания каждому биту регистра.

Writing 1 to the ISPR bit corresponding to an interrupt that is pending:
- has no effect.

Запись '1' в бит регистра **ISPR** с уже установленным флагом запроса не имеет эффекта.

Writing 1 to the ISPR bit corresponding to a disabled interrupt:
- sets the state of that interrupt to pending.

Запись '1' в чистый бит регистра **ISPR** ставит флаг запроса прерывания.

4.2.5 Interrupt clear-pending registers (NVIC_ICPRx)
Регистр очистки флагов прерываний

Address offset: 0x00 - 0x0B
Reset value: 0x0000 0000
Required privilege: Privileged

The ICPR0-ICPR2 registers remove the pending state from interrupts, and show which interrupts are pending. Регистры **ICPR0-ICPR2** очищают состояние запроса прерываний и показывают, какое из прерываний находится в состоянии запроса.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLRPEND[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRPEND[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 CLRPEND[31:0]: Interrupt clear-pending bits (Биты очистки запроса прерывания)

Write (При записи):

`0:` No effect (Нет эффекта)

`1:` Removes the pending state of an interrupt (Очищает состояние запроса прерывания)

Read (При чтении):

`0:` Interrupt is not pending (Нет состояния запроса у прерывания)

`1:` Interrupt is pending (Есть состояние запроса у прерывания)

See *Table 34: Mapping of interrupts to the interrupt variables on page 105* for the correspondence of interrupts to each register bit.
См. Таблицу 34: "Отображение прерываний на свои переменные" при поиске соответствия прерывания каждому биту регистра.

Writing 1 to an ICPR bit does not affect the active state of the corresponding interrupt.
Запись '1' в бит регистра **ICPR** не имеет эффекта на состояние активности соответствующего прерывания.

4.2.6 Interrupt active bit registers (NVIC_IABRx)
Регистр битов активности прерываний

Address offset: 0x00 - 0x0B
Reset value: 0x0000 0000
Required privilege: Privileged

The IABR0-IABR2 registers indicate which interrupts are active. The bit assignments are:
Регистры **IABR0-IABR2** показывают, какое из прерываний сейчас активно.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACTIVE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACTIVE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 ACTIVE[31:0]: Interrupt active flags (Флаги активности прерываний)

- **0:** Interrupt not active (Прерывание неактивно)
- **1:** Interrupt active (Прерывание активно)

See *Table 34: Mapping of interrupts to the interrupt variables on page 105* for the correspondence of interrupts to each register bit. См. Таблицу 34: "Отображение прерываний на свои переменные" при поиске соответствия прерывания каждому биту регистра.

A bit reads as 1 if the status of the corresponding interrupt is active or active and pending.
Бит читается как 1, если соответствующее прерывание активно, или активно и имеет статус отложенного запроса.

4.2.7 Interrupt priority registers (NVIC_IPRx)
Регистр приоритетов прерываний

Address offset: 0x00- 0x0B
Reset value: 0x0000 0000
Required privilege: Privileged

The IPR0-IPR16 registers provide a 4-bit priority field for each interrupt. These registers are byte-accessible. Each register holds four priority fields, that map to four elements in the CMSIS interrupt priority array IP[0] to IP[67], as shown in Figure 18. Регистры **IPR0-IPR16** обеспечивают 4-х битовое поле приоритета для каждого прерывания. Эти регистры имеют байтовый доступ. Каждый регистр содержит четыре поля приоритета, которые отображают четыре элемента в **CMSIS** массиве приоритетов прерываний **IP[0]...IP[67]**, как показано на рис. 18.

Figure 18. NVIC__IPRx register mapping (Отображение регистров **NVIC__IPRx**)

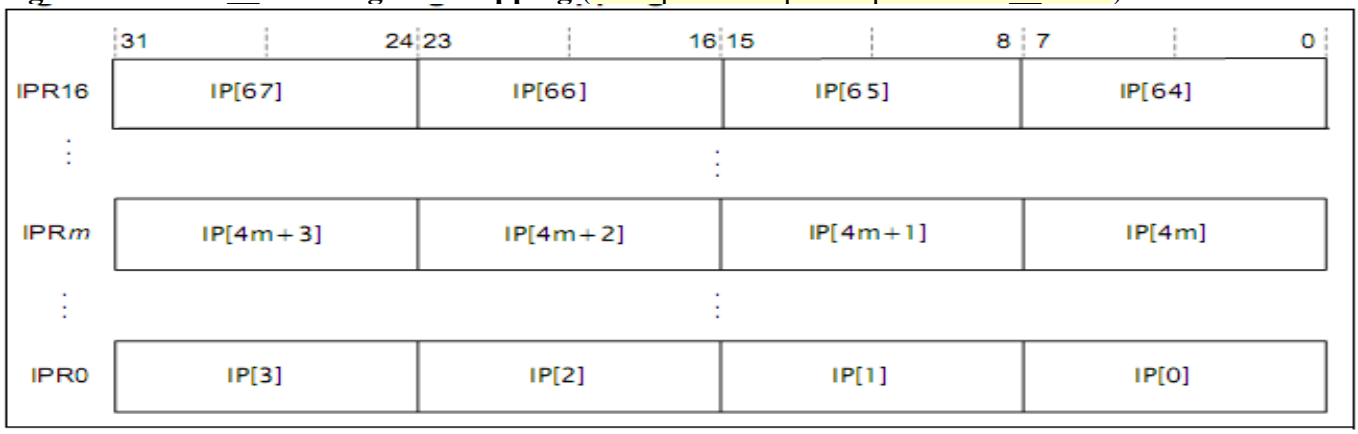


Table 35. IPR bit assignments (Назначения битов **IPR**)

Interrupts	Name	Function
[31:24]	Priority, byte offset 3	Each priority field holds a priority value, 0-255. The lower the value, the greater the priority of the corresponding interrupt. The processor implements only bits[7:4] of each field, bits[3:0] read as zero and ignore writes. Каждое поле приоритета содержит его значение, 0-255. Чем ниже значение, тем выше приоритет соответствующего прерывания. Процессор принимает только биты [7:4] каждого поля, биты [3:0] читаются как нуль и запись в них игнорируется. (*Семейство **CL** принимает все поле*)
[23:16]	Priority, byte offset 2	
[15:8]	Priority, byte offset 1	
[7:0]	Priority, byte offset 0	

See *The CMSIS mapping of the Cortex-M3 NVIC registers on page 105* for more information about the IP[0] to IP[67] interrupt priority array, that provides the software view of the interrupt priorities.
Смотри раздел 4.2.1 "Как **CMSIS** отображает регистры **Cortex-M3 NVIC**" для дополнительной информация о массиве приоритетов прерываний **IP[0]...IP[67]**, который обеспечивает программное представление приоритетов прерываний.

Find the IPR number and byte offset for interrupt N as follows:
Чтобы найти число **IPR** и смещение байта для прерывания **N**, надо:

- The corresponding IPR number, M, is given by M = N DIV 4
Соответствующий **IPR** номер **M** вычисляется как **N** деленное на 4

- The byte offset of the required Priority field in this register is N MOD 4, where:
Байтовое смещение для необходимого поля приоритета в этом регистре определяется как **N** (по модулю 4), где:

  - byte offset 0 refers to register bits[7:0]
  байтовое смещение 0 относится к битам **[7:0]** регистра

  - byte offset 1 refers to register bits[15:8]
  байтовое смещение 1 относится к битам **[15:8]** регистра

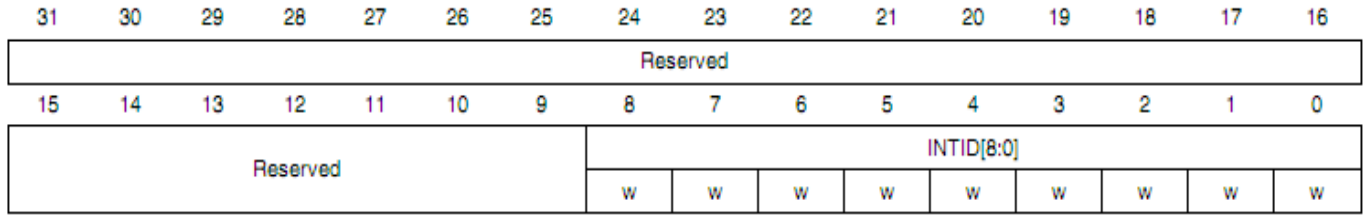
  - byte offset 2 refers to register bits[23:16]
  байтовое смещение 2 относится к битам **[23:16]** регистра

  - byte offset 3 refers to register bits[31:24].
  байтовое смещение 3 относится к битам **[31:24]** регистра

4.2.8 Software trigger interrupt register (NVIC_STIR)
Регистр программного запуска прерываний

Address offset: 0xE00
Reset value: 0x0000 0000

Required privilege: When the USERSETMPEND bit in the SCR is set to 1, unprivileged software can access the STIR, see *Section 4.3.5: System control register (SCB_SCR)*. Only privileged software can enable unprivileged access to the STIR.
Необходимая привилегия: Когда бит **USERSETMPEND** в **SCR** установлен в 1, то код программы в непривилегированном режиме может обратиться к **STIR**, см. раздел 4.3.5: "Регистр управления системой (**SCB_SCR**)". Только привилегированный код может разрешить непривилегированному доступу к **STIR**.



Bits 31:9 Reserved
must be kept cleared. (При записи всегда 0)

Bits 8:0 NTID[8:0] Software generated interrupt ID
(Идентификатор программно генерируемого прерывания)

Write to the STIR to generate a Software Generated Interrupt (SGI). The value to be written is the Interrupt ID of the required SGI, in the range 0-239. For example, a value of 0b000000011 specifies interrupt IRQ3.
Запишите в **STIR**, чтобы сгенерировать программное прерывание (**SGI**). Значение, которое будет записано, является идентификатором **ID** необходимого **SGI**, и может быть в диапазоне 0-239. Например, значение **0b000000011** определяет прерывание **IRQ3**.

4.2.9 Level-sensitive and pulse interrupts
Чувствительность прерываний к уровню и фронту

STM32 interrupts are both level-sensitive and pulse-sensitive. Pulse interrupts are also described as edge-triggered interrupts.
Прерывания в **STM32** могут быть чувствительны как к уровню, так и к импульсу сигнала. Прерывания от импульса именуются также как прерывания, запускаемые перепадом сигнала.

A level-sensitive interrupt is held asserted until the peripheral deasserts the interrupt signal. Typically this happens because the ISR accesses the peripheral, causing it to clear the interrupt request. A pulse interrupt is an interrupt signal sampled synchronously on the rising edge of the processor clock. To ensure the NVIC detects the interrupt, the peripheral must assert the interrupt signal for at least one clock cycle, during which the NVIC detects the pulse and latches the interrupt.
Для прерывания, чувствительного к уровню, запрос удерживается до тех пор, пока внешнее устройство не снимет сигнал прерывания. Обычно это случается потому, что **ISR** обращается к внешнему устройству, заставляя его очистить запрос на прерывание. Прерывание от импульса - это сигнал прерывания, выборка которого производится синхронно, по фронту тактового сигнала процессора. Для гарантированного обнаружения прерывания **NVIC**, внешнее устройство должно удерживать сигнал прерывания по крайней мере в течении одного цикла тактового сигнала, во время которого **NVIC** обнаруживает импульс и защелкивает прерывание.

When the processor enters the ISR, it automatically removes the pending state from the interrupt, see *Hardware and software control of interrupts*. For a level-sensitive interrupt, if the signal is not deasserted before the processor returns from the ISR, the interrupt becomes pending again, and the processor must execute its ISR again. This means that the peripheral can hold the interrupt signal asserted until it no longer needs servicing.
Когда процессор входит в обработчик прерывания, он автоматически удаляет состояние запроса от прерывания, см. параграф "Аппаратное и программное управление прерываниями". Для прерывания, чувствительного к уровню, если сигнал не был снят до возвращения процессора из **ISR**, то снова появляется запрос на прерывание, и процессор должен выполнить этот **ISR** снова. Это означает, что внешнее устройство может удерживать сигнал прерывания до тех пор, пока не пропадет необходимость в его обслуживании.

Hardware and software control of interrupts
Аппаратное и программное управление прерываниями

The Cortex-M3 latches all interrupts. A peripheral interrupt becomes pending for one of the following reasons: Ядро **Cortex-M3** защелкивает все прерывания. Запрос на прерывание от внешнего устройства становится по одной из следующих причин:

- The NVIC detects that the interrupt signal is HIGH and the interrupt is not active
NVIC детектирует, что сигнал прерывания имеет высокий уровень, а прерывание пока неактивно

- The NVIC detects a rising edge on the interrupt signal

NVIC детектирует наличие фронта сигнала прерывания

- Software writes to the corresponding interrupt set-pending register bit, see *Section 4.2.4: Interrupt set-pending registers (NVIC_ISPRx)*, or to the STIR to make an SGI pending, see *Section 4.2.8: Software trigger interrupt register (NVIC_STIR)*.

Программа записывает '1' в соответствующий бит регистра запроса на прерывание, см. раздел 4.2.4 "Регистры установки запроса на прерывание (**NVIC_ISPRx**)", или регистра **STIR**, чтобы вызвать программное прерывание, см. раздел 4.2.8 "Регистр программного запуска прерываний (**NVIC_STIR**)".

A pending interrupt remains pending until one of the following:

Флаг вызова прерывания остается выставленным до одного из следующих событий:

- The processor enters the ISR for the interrupt. This changes the state of the interrupt from pending to active. Then:

Процессор входит в **ISR** для этого прерывания. Это изменяет состояние прерывания с отложенного на активный. Тогда:

  - For a level-sensitive interrupt, when the processor returns from the ISR, the NVIC samples the interrupt signal. If the signal is asserted, the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR. Otherwise, the state of the interrupt changes to inactive.

Для прерывания, чувствительного к уровню, когда процессор возвращается из **ISR**, **NVIC** считывает сигнал прерывания. Если сигнал выставлен, то состояние прерывания изменяется на отложенное, что может привести к тому, что процессор немедленно повторно войдет в **ISR**. В противном случае, состояние прерывания изменяется на неактивное.

  - For a pulse interrupt, the NVIC continues to monitor the interrupt signal, and if this is pulsed the state of the interrupt changes to pending and active. In this case, when the processor returns from the ISR the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR. If the interrupt signal is not pulsed while the processor is in the ISR, when the processor returns from the ISR the state of the interrupt changes to inactive.

Для прерывания от импульса **NVIC** продолжает контролировать сигнал прерывания, и если на нем появляется импульс, то сотояние прерывания изменяется на активное и отложенное. В этом случае, когда процессор возвращается из **ISR**, то состояние прерывания изменяется на отложенное, что может привести к тому, что процессор немедленно повторно войдет в **ISR**. Если на сигнале прерывания не было импульса, пока процессор находится в **ISR**, то, по выходу процессора из **ISR**, состояние прерывания изменяется на неактивное.

- Software writes to the corresponding interrupt clear-pending register bit. For a level-sensitive interrupt, if the interrupt signal is still asserted, the state of the interrupt does not change. Otherwise, the state of the interrupt changes to inactive. For a pulse interrupt, state of the interrupt changes to:

Программа пишет '1' в соответствующий бит регистра очистки запроса прерывания. Для прерывания, чувствительного к уровню, если сигнал прерывания все еще выставлен, то состояние прерывания не изменяется. Иначе, состояние прерывания изменяется на неактивное. Для прерывания от импульса состояние прерывания изменяется на:

  - Inactive, if the state was pending

Неактивное, если состояние было отложенным

  - Active, if the state was active and pending.

Активное, если если состояние было активным и отложенным.

4.2.10 NVIC design hints and tips (Работа с NVIC, рекомендации и советы)

Ensure software uses correctly aligned register accesses. The processor does not support unaligned accesses to NVIC registers. See the individual register descriptions for the supported access sizes.
Убедитесь, что программа получает доступ к регистрам с корректным выравниваем и размером. Процессор не поддерживает невыровненный доступ к регистрам **NVIC**. См. описания на индивидуальные регистры для определения нужного размера доступа.

A interrupt can enter pending state even it is disabled.
Прерывание может получить статус отложенного, даже если оно отключено.

Before programming VTOR to relocate the vector table, ensure the vector table entries of the new vector table are setup for fault handlers, NMI and all enabled exception like interrupts. For more information see *Section 4.3.3: Vector table offset register (SCB_VTOR) on page 118*.
Прежде, чем программировать **VTOR**, чтобы переместить таблицу векторов, убедитесь, что заданы векторные входы новой таблицы для обработчиков ошибок, немаскируемого прерывания и всех исключений, разрешенных как прерывания. Для получения дополнительной информации см. раздел 4.3.3: "Регистр смещения таблицы векторов (**SCB_VTOR**)".

NVIC programming hints (Советы по программированию NVIC)

Software uses the CPSIE I and CPSID I instructions to enable and disable interrupts. The CMSIS provides the following intrinsic functions for these instructions:
Программи использует инструкции '**CPSIE I**' и '**CPSID I**', чтобы разрешить и отключить прерывания. **CMSIS** предоставляет следующие встроенные функции для этих команд:

```
void __disable_irq(void)  // Disable Interrupts
void __enable_irq(void)  // Enable Interrupts
```

In addition, the CMSIS provides a number of functions for NVIC control, including:
Кроме того, **CMSIS** предоставляет кучу функций для управления **NVIC**, включая:

Table 36. CMSIS functions for NVIC control (CMSIS функции для управления NVIC)

CMSIS interrupt control function	Description
void NVIC_SetPriorityGrouping(uint32_t priority_grouping)	Set the priority grouping Задание группы приритета
void NVIC_EnableIRQ(IRQn_t IRQn)	Enable IRQn (Разрешение прерывания **n**)
void NVIC_DisableIRQ(IRQn_t IRQn)	Disable IRQn (Запрещение прерывания **n**)
uint32_t NVIC_GetPendingIRQ (IRQn_t IRQn)	Return true (IRQ-Number) if IRQn is pending (Возвращает **true** (номер **IRQ**), если есть флаг запроса)
void NVIC_SetPendingIRQ (IRQn_t IRQn)	Set IRQn pending (Ставит запрос прерывания **n**)
void NVIC_ClearPendingIRQ (IRQn_t IRQn)	Clear IRQn pending status Очищает запрос прерывания **n**
uint32_t NVIC_GetActive (IRQn_t IRQn)	Return the IRQ number of the active interrupt (Возвращает номер активного прерывания)
void NVIC_SetPriority (IRQn_t IRQn, uint32_t priority)	Set priority for IRQn Задает приоритет прерывания **n**
uint32_t NVIC_GetPriority (IRQn_t IRQn)	Read priority of IRQn Читает приоритет прерывания **n**
void NVIC_SystemReset (void)	Reset the system (Программный сброс системы)

For more information about these functions see the CMSIS documentation.
Для получения дополнительной информации об этих функциях см. документацию на **CMSIS**.

4.2.11 NVIC register map (Карта регистров NVIC)

The table provides shows the NVIC register map and reset values. The base address of the main NVIC register block is 0xE000E100. The NVIC_STIR register is located in a separate block at 0xE000EF00. Таблица показывает карту регистров **NVIC** и их значение после сброса. Базовый адрес главного блока регистров **NVIC - 0xE000E100**. Регистр **NVIC_STIR** расположен в отдельном блоке по адресу **0xE000EF00**.

Table 37. NVIC register map and reset values
Карта регистров **NVIC** и их значение после сброса

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	NVIC_ISER0	SETENA[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	NVIC_ISER1	SETENA[63:32]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	NVIC_ISER2	Reserved																												SETENA [67:64]			
	Reset Value																													0	0	0	0
0x080	NVIC_ICER0	CLRENA[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	NVIC_ICER1	CLRENA[63:32]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	NVIC_ICER2	Reserved																												CLRENA [67:64]			
	Reset Value																													0	0	0	0
0x200	NVIC_ISPR0	SETPEND[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x204	NVIC_ISPR1	SETPEND[63:32]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x208	NVIC_ISPR2	Reserved																												SETPEND [67:64]			
	Reset Value																													0	0	0	0
0x280	NVIC_ICPR0	CLRPEND[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x284	NVIC_ICPR1	CLRPEND[63:32]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x288	NVIC_ICPR2	Reserved																												CLRPEND [67:64]			
	Reset Value																													0	0	0	0
0x300	NVIC_IABR0	ACTIVE[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x304	NVIC_IABR1	ACTIVE[63:32]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x308	NVIC_IABR2	Reserved																												ACTIVE [67:64]			
	Reset Value																													0	0	0	0
0x400	NVIC_IPR0	IP[3]								IP[2]								IP[1]								IP[0]							
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
:	:	:																															
0x410	NVIC_IPR16	IP[67]								IP[66]								IP[65]								IP[64]							
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SCB registers																																	
Reserved																																	
0xE00	NVIC_STIR	Reserved																												INTID[8:0]			
	Reset Value																													0	0	0	0

4.3 System control block (SCB)
Блок регистров управления системой

4.3.1 CPUID base register (SCB_CPUID) (Регистр идентификатора ЦПУ)
4.3.2 Interrupt control and state register (SCB_ICSR)
Регистр управления прерываниями и их состоянием (**SCB_ICSR**)
4.3.3 Vector table offset register (SCB_VTOR) (Регистр смещения таблицы векторов)
4.3.4 Application interrupt and reset control register (SCB_AIRCR)
Регистр прерываний приложения и управления сбросом (**SCB_AIRCR**)
4.3.5 System control register (SCB_SCR) (Регистр управления системой)
4.3.6 Configuration and control register (SCB_CCR)
(Регистр конфигурации и управления)
4.3.7 System handler priority registers (SHPRx)
(Регистры приоритетов системных обработчиков)
4.3.8 System handler control and state register (SCB_SHCSR)
Регистр управления и состояния системных обработчиков (**SCB_SHCSR**)
4.3.9 Configurable fault status register (SCB_CFSR)
(Регистр статуса конфигурируемых ошибок)
4.3.10 Hard fault status register (SCB_HFSR) (Регистр статуса аппаратных ошибок)
4.3.11 Memory management fault address register (SCB_MMFAR)
Регистр адреса ошибки управления памятью (**SCB_MMFAR**)
4.3.12 Bus fault address register (SCB_BFAR) (Регистр адреса ошибки шины)
4.3.13 System control block design hints and tips
Программирование регистров блока **SCB**: рекомендации и советы
4.3.14 SCB register map (Карта регистров блока **SCB**)

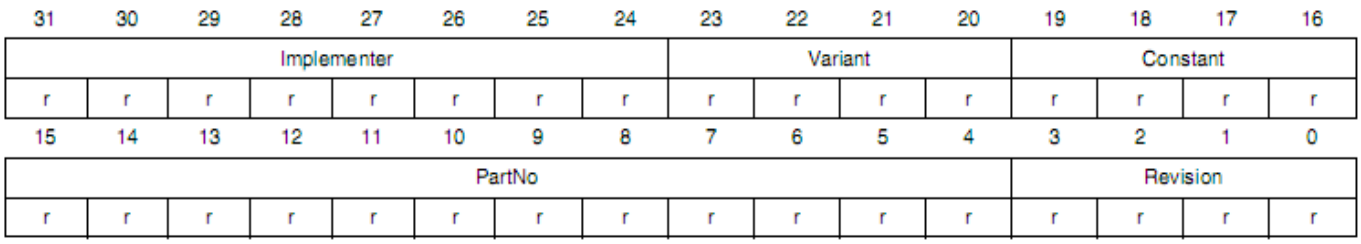
The System control block (SCB) provides system implementation information, and system control. This includes configuration, control, and reporting of the system exceptions. The CMSIS mapping of the Cortex-M3 SCB registers.
Блок управления системой (**SCB**) предоставляют информацию о реализации системы, и возможность управления системой. Это включает конфигурацию, управление, и сообщение системных исключений. **CMSIS** отображает регистры **Cortex-M3** на блок **SCB**.

To improve software efficiency, the CMSIS simplifies the SCB register presentation. In the CMSIS, the byte array SHP[0] to SHP[12] corresponds to the registers SHPR1-SHPR3.
Чтобы улучшить эффективность программирования, **CMSIS** упрощает представление регистров **SCB**. В **CMSIS** байтовый массив **SHP[0]...SHP[12]** соответствует регистрам **SHPR1-SHPR3**.

4.3.1 CPUID base register (SCB_CPUID) (Регистр идентификатора ЦПУ)

Address offset: 0x00
Reset value: 0x411F C231
Required privilege: Privileged

The CPUID register contains the processor part number, version, and implementation information.
Регистр **CPUID** содержит номер процессора, его версию, и информацию о реализации.



Bits 31:24 Implementer: Implementer code (Код реализации)
`0x41:` ARM

Bits 23:20 Variant: Variant number (Номер вариации)
The r value in the rnpn product revision identifier
Значение **r** в идентификаторе **rnpn** ревизии продукта
`0x1:` r1p1

Bits 19:16 Constant: (Константа)
Reads as 0xF (Читается как **0xF**)

its 15:4 PartNo: Part number of the processor (Номер процессора)
`0xC23:` = Cortex-M3

Bits 3:0 Revision: Revision number (Номер ревизии)
The p value in the rnpn product revision identifier
Значение **p** в идентификаторе **rnpn** ревизии продукта
`0x1:` = r1p1

4.3.2 Interrupt control and state register (SCB_ICSR)
Регистр управления прерываниями и их состоянием

Address offset: 0x04
Reset value: 0x0000 0000
Required privilege: Privileged

Регистр **ICSR**:

- Provides: (предоставляет:)
  - A set-pending bit for the Non-Maskable Interrupt (NMI) exception
  Бит установки запроса исключения для немаскируемого прерывания
  - Set-pending and clear-pending bits for the PendSV and SysTick exceptions
  Биты установки и очистки запроса для исключений **PendSV** и **SysTick**
- Indicates: (Показывает:)
  - The exception number of the exception being processed
  Номер обрабатываемого исключения
  - Whether there are preempted active exceptions
  Есть ли прерванные активные исключения
  - The exception number of the highest priority pending exception
  Номер исключения для отложенного исключения с высшим приоритетом
  - Whether any interrupts are pending. (Есть ли отложенные прерывания)

Caution: When you write to the ICSR, the effect is unpredictable if you:
Предупреждение: Когда вы пишете в **ICSR**, эффект непредсказуем, если вы:

- Write 1 to the PENDSVSET bit and write 1 to the PENDSVCLR bit
Пишите '1' в бит **PENDSVSET** и бит **PENDSVCLR**

- Write 1 to the PENDSTSET bit and write 1 to the PENDSTCLR bit.

Пишите '1' в бит **PENDSTSET** и бит **PENDSTCLR**.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NMIPENDSET	Reserved		PENDSVSET	PENDSVCLR	PENDSTSET	PENDSTCLR	Reserved		ISRPENDING	VECTPENDING[9:4]					
w			rw	w	rw	w			r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VECTPENDING[3:0]				RETOBASE	Reserved		VECTACTIVE[8:0]								
r	r	r	r	r			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 NMIPENDSET: NMI set-pending bit. (Бит установки запроса на NMI)

Write (При записи):

0: No effect (Нет эффекта)

1: Change NMI exception state to pending. (Ставит бит запроса на исключение **NMI**)

Read (При чтении):

0: NMI exception is not pending (Нет запроса на исключение **NMI**)

1: NMI exception is pending (Есть запрос на исключение **NMI**)

Because NMI is the highest-priority exception, normally the processor enter the NMI exception handler as soon as it registers a write of 1 to this bit, and entering the handler clears this bit to 0. A read of this bit by the NMI exception handler returns 1 only if the NMI signal is reasserted while the processor is executing that handler.

Так как исключение **NMI** имеет высший приоритет, обычно, процессор входит в обработчик **NMI** сразу, как только пишется '1' в этот бит регистра, а вход в обработчик очищает этот бит. Чтение этого бита в обработчике **NMI** даст '1' только в том случае, если сигнал **NMI** повторно выставлен в то время, пока процессор выполняет код обработчика.

Bits 30:29 Reserved

must be kept cleared (При записи всегда 0)

Bit 28 PENDSVSET: PendSV set-pending bit. (Бит установки запроса на PendSV)

Write (При записи):

0: No effect (Нет эффекта)

1: Change PendSV exception state to pending. (Ставит бит запроса на исключение **PendSV**)

Read (При чтении):

0: PendSV exception is not pending (Нет запроса на исключение **PendSV**)

1: PendSV exception is pending (Есть запрос на исключение **PendSV**)

Writing 1 to this bit is the only way to set the PendSV exception state to pending.

Запись '1' в этот бит - это единственный способ запросить исключение **PendSV**

Bit 27 PENDSVCLR: PendSV clear-pending bit. (Бит очистки запроса на PendSV)

Write (При записи):

0: No effect (Нет эффекта)

1: Removes the pending state from the PendSV exception.

Удаляет состояние запроса на исключение **PendSV**.

Bit 26 PENDSTSET: SysTick exception set-pending bit. (Бит установки запроса на SysTick)

Write (При записи):

0: No effect (Нет эффекта)

1: Change SysTick exception state to pending (Ставит бит запроса на исключение **SysTick**)

Read (При чтении):

0: SysTick exception is not pending (Нет запроса на исключение **SysTick**)

1: SysTick exception is pending (Есть запрос на исключение **SysTick**)

Bit 25 PENDSTCLR: SysTick exception clear-pending bit. (Бит очистки запроса на SysTick)

Write (При записи):

`0:` No effect (Нет эффекта)

`1:` Removes the pending state from the SysTick exception.

Удаляет состояние запроса на исключение **SysTick**

This bit is write-only. On a register read its value is unknown.

Это бит только для записи. При попытке чтения значение этого бита неопределено.

Bit 24 Reserved

must be kept cleared. (При записи всегда 0)

Bit 23 Reserved

This bit is reserved for Debug use and reads-as-zero when the processor is not in Debug.

Этот бит зарезервирован для отладочных целей и читается как '0', если процессор не находится в режиме **Debug**.

Bit 22 ISRPENDING: Interrupt pending flag (Флаг запроса на прерывание)

excluding NMI and Faults (включая от **NMI** и от ошибок)

`0:` Interrupt not pending (Нет запроса на прерывание)

`1:` Interrupt pending (Есть запрос на прерывание)

Bits 21:12 VECTPENDING[9:0] Pending vector (Вектор запроса)

Indicates the exception number of the highest priority pending enabled exception.

Показывает номер отложенного исключения с высшим приоритетом

`0:` No pending exceptions (Нет запроса на исключение)

`Other values:` The exception number of the highest priority pending enabled exception.

Номер исключения того из отложенных исключений, что имеет высший приоритет

The value indicated by this field includes the effect of the BASEPRI and FAULTMASK registers, but not any effect of the PRIMASK register.

Значение, показанное в этом поле, включает эффект регистров **BASEPRI** и **FAULTMASK**, и часть эффектов регистра **PRIMASK**.

Bit 11 RETOBASE: Return to base level (Возврат на базовый уровень)

Indicates whether there are preempted active exceptions:

Указывает, есть ли прерванные активные исключения:

`0:` There are preempted active exceptions to execute

Есть выгруженное активное исключения, которое следует завершить

`1:` There are no active exceptions, or the currently-executing exception is the only active exception. Нет никаких активных исключений, или то исключение, что сейчас выполняется - является единственным активным исключением.

Bits 10:9 Reserved

must be kept cleared (При записи всегда 0)

Bits 8:0 VECTACTIVE[8:0] Active vector (Активный вектор)

Contains the active exception number: (Содержит номер активного исключения)

`0:` Thread mode (Режим **Thread**)

`Other values:` The exception number[1] of the currently active exception.

`Другое значение:` Номер того исключения, что активно в текущий момент.

Note: Subtract 16 from this value to obtain the IRQ number required to index into the Interrupt Clear-Enable, Set-Enable, Clear-Pending, Set-Pending, or Priority Registers, see Table 5 on page 17.

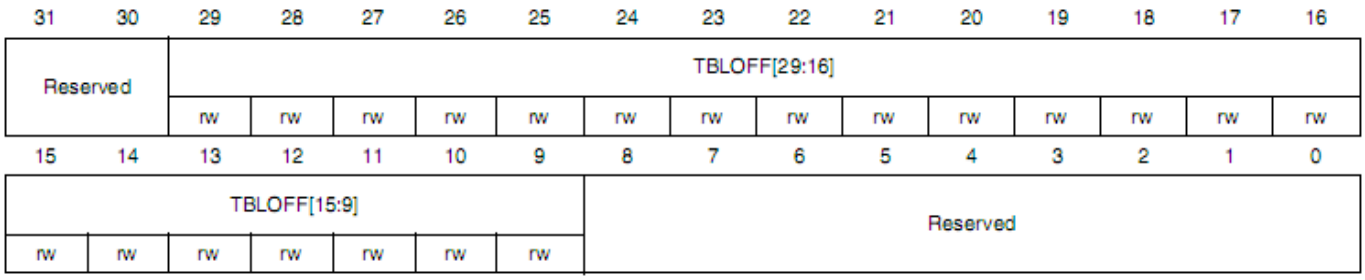
Note: Вычтите 16 из этого значения, чтобы получить номер **IRQ**, который требуется для индексирования бита в регистрах разрешения и отключения прерываний, установки и

очистки запроса прерывания, или в регистрах приоритета, см. Таблицу 5.

1. This is the same value as IPSR bits[8:0], see *Interrupt program status register on page 17*.
Это то же самое значение, что и поле **IPSR[8:0]**, см. параграф "Регистр статуса прерываний" в разделе 2.1.3.

4.3.3 Vector table offset register (SCB_VTOR)
Регистр смещения таблицы векторов

Address offset: 0x08
Reset value: 0x0000 0000
Required privilege: Privileged



Bits 31:30 Reserved
must be kept cleared (При записи всегда 0)

Bits 29:11 TBLOFF[29:9]: Vector table base offset field. (Смещение базы таблицы векторов)
It contains bits [29:9] of the offset of the table base from memory address 0x00000000. When setting TBLOFF, you must align the offset to the number of exception entries in the vector table. The minimum alignment is 128 words. Table alignment requirements mean that bits[8:0] of the table offset are always zero. Bit 29 determines whether the vector table is in the code or SRAM memory region.
Эти биты **[29:9]** содержат смещение базы таблицы относительно адреса памяти **0x00000000**. При задании этого значения, вы должны выровнить смещение по числу исключений в таблице векторов. Минимальное выравнивание - 128 слов. Требования выравнивания таблицы означают, что биты **[8:0]** смещения таблицы всегда являются нулем. Бит 29 определяет, находится ли таблица векторов в области кода или в области **SRAM** памяти.
`0:` Code
`1:` SRAM
Note: Bit 29 is sometimes called the TBLBASE bit.
Бит 29 иногда называют битом **TBLBASE**.

Bits 10:0 Reserved
must be kept cleared (При записи всегда 0)

4.3.4 Application interrupt and reset control register (SCB_AIRCR)
Регистр прерываний приложения и управления сбросом

Address offset: 0x0C
Reset value: 0xFA05 0000
Required privilege: Privileged

The AIRCR provides priority grouping control for the exception model, endian status for data accesses, and reset control of the system.
AIRCR обеспечивает управление группировкой приоритетов для модели исключений, статус **endian** (порядок упаковки байт в слове) для доступа к данным, и управление сбросом системы.

To write to this register, you must write 0x5VA to the VECTKEY field, otherwise the processor ignores the write. При записи в этот регистр, вы должны записать значение **0x5VA** в поле **VECTKEY**, иначе процессор проигнорирует запись.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VECTKEYSTAT[15:0](read)/ VECTKEY[15:0](write)															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDIANESS	Reserved				PRIGROUP			Reserved					SYSRESETREQ	VECTCLRACTIVE	VECTRESET
r					rw	rw	rw						w	w	w

Bits 31:16 VECTKEYSTAT[15:0]/VECTKEY[15:0] Register key
Reads as 0x05FA (Читается как **0x05FA**)
При записи, пишите значение **0x5FA** в **VECTKEY**, иначе запись будет проигнорирована.

Bit 15 ENDIANESS Data endianness bit (Бит статуса endian)
Reads as 0. (Читается как 0)
`0:` Little-endian

Bits 14:11 Reserved
must be kept cleared (При записи всегда 0)

Bits 10:8 PRIGROUP[2:0]: Interrupt priority grouping field
Поле группировки приоритетов прерываний
This field determines the split of group priority from subpriority, see Binary point on page 119.
Это поле определяет точку отделения группы приоритета от субприоритета, см. параграф "Бинарная точка" чуть ниже.

Bits 7:3 Reserved
must be kept cleared (При записи всегда 0)

Bit 2 SYSRESETREQ System reset request (Требование на сброс системы)
This is intended to force a large system reset of all major components except for debug.
Этот бит предназначен для инициации "большого" сброса системы, всех ее главных компонентов, за исключением модуля отладки.
This bit reads as 0. (Этот бит читается как 0)
`0:` No system reset request (Нет запроса на сброс системы)
`1:` Asserts a signal to the outer system that requests a reset.
Выставляет сигнал сброса тем частям системы, что требуют сброса

Bit 1 VECTCLRACTIVE

Reserved for Debug use. This bit reads as 0. When writing to the register you must write 0 to this bit, otherwise behavior is unpredictable.
Зарезервирован для целей отладки. Этот бит читается как '0'. При записи в регистр вы должны писать '0' в этот бит, иначе поведение будет непредсказуемым.

Bit 0 VECTRESET

Reserved for Debug use. This bit reads as 0. When writing to the register you must write 0 to this bit, otherwise behavior is unpredictable
Зарезервирован для целей отладки. Этот бит читается как '0'. При записи в регистр вы должны писать '0' в этот бит, иначе поведение будет непредсказуемым.

Binary point (Бинарная точка)

The PRIGROUP field indicates the position of the binary point that splits the PRI_n fields in the Interrupt Priority Registers into separate group priority and subpriority fields. Table 38 shows how the PRIGROUP value controls this split.
Поле **PRIGROUP** указывает позицию бинарной точки в двоичном числе, которая разбивает поля **PRI_n** в регистре **IPR** на отдельные субполя группы приоритета и субприоритета. Таблица 38 показывает, как значение **PRIGROUP** управляет этим разбиением.

Table 38. Priority grouping (Группировка приоритетов)

PRIGROUP [2:0]	Interrupt priority level value, PRI_N[7:4]			Number of	
	Binary point [1]	Group priority bits	Subpriority bits	Group priorities	Sub priorities
0b011	0bxxxx	[7:4]	None	16	None
0b100	0bxxx.y	[7:5]	[4]	8	2
0b101	0bxx.yy	[7:6]	[5:4]	4	4
0b110	0bx.yyy	[7]	[6:4]	2	8
0b111	0b.yyyy	None	[7:4]	None	16

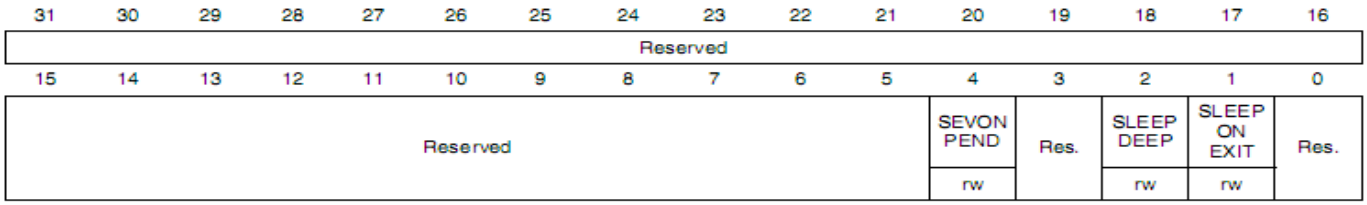
1. PRI_n[7:4] field showing the binary point. x denotes a group priority field bit, and y denotes a subpriority field bit.
Поле **PRI_n[7:4]** показано с бинарной точкой. **x** обозначает бит субполя группы приоритета, а **y** обозначает бит субполя субприоритета.

Determining preemption of an exception uses only the group priority field, see *Section 2.3.6: Interrupt priority grouping on page 36*. При определении приоритета исключения используется только поле группы приоритета, см. раздел 2.3.6 "Группировка приоритетов исключений".
(*В семействе **CL** используются все 8 бит поля.*)

4.3.5 System control register (SCB_SCR) (Регистр управления системой)

Address offset: 0x10
Reset value: 0x0000 0000
Required privilege: Privileged

The SCR controls features of entry to and exit from low power state. Регистр **SCR** управляет особенностями входа в состояние потребления малой мощности и выхода из него.



Bits 31:5 Reserved
must be kept cleared (При записи всегда 0)

Bit 4 SEVEONPEND Send Event on Pending bit (Считать запрос IRQ событием пробуждения)
When an event or interrupt enters pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE. The processor also wakes up on execution of an SEV instruction or an external event.
Когда событие или прерывание ставят флаг запроса, то это пробуждает процессор из состояния **WFE**. Если процессор не ждет этого события, то оно регистрируется и имеет эффект на то, что следует после состояния **WFE**. Процессор также просыпается при выполнении инструкции **SEV** или от внешнего события.
`0:` Only enabled interrupts or events can wakeup the processor, disabled interrupts are excluded
Только разрешенные прерывания или события могут разбудить процессор.
`1:` Enabled events and all interrupts, including disabled interrupts, can wakeup the processor.
Разрешенные события и все прерывания, включая запрещенные, могут разбудить процессор.

Bit 3 Reserved
must be kept cleared (При записи всегда 0)

Bit 2 SLEEPDEEP
Controls whether the processor uses sleep or deep sleep as its low power mode:
Управляет, будет ли процессор использовать простой сон или глубокий сон при входе в режим малого потребления:
`0:` Sleep
`1:` Deep sleep. (Глубокий сон)

Bit 1 SLEEPONEXIT
Configures sleep-on-exit when returning from Handler mode to Thread mode. Setting this bit to 1 enables an interrupt-driven application to avoid returning to an empty main application.
Конфигурация режима **сон-по-выходу**, при возврате из обработчика прерывания в режим **Thread**. Установка этого бита в '1' дает возможность приложению, управляемому только от прерываний, избежать возвращения в пустую функцию `main()`.
`0:` Do not sleep when returning to Thread mode.
Нет засыпания при возврате в режим **Thread**.
`1:` Enter sleep, or deep sleep, on return from an interrupt service routine.
Вход в режим сна или глубогого сна по возвращению из процедуры обработки прерывания.

Bit 0 Reserved
must be kept cleared (При записи всегда 0)

4.3.6 Configuration and control register (SCB_CCR)
Регистр конфигурации и управления

Address offset: 0x14
Reset value: 0x0000 0200
Required privilege: Privileged

The CCR controls entry to Thread mode and enables:
Регистр **CCR** управляет входом в режим **Thread** и позволяет:

- The handlers for NMI, hard fault and faults escalated by FAULTMASK to ignore bus faults Обработчикам для **NMI**, аппаратных ошибок и ошибок, усиленных до аппаратных с помощью **FAULTMASK**, игнорировать ошибки шины

- Trapping of divide by zero and unaligned accesses
Перехватывать события деления на ноль и невыровненного доступа

- Access to the STIR by unprivileged software, see *Software trigger interrupt register (NVIC_STIR) on page 112*. Давать доступ к **STIR** для непривилегированного кода, см. раздел 4.2.8 "Регистр программного запуска прерываний (NVIC_STIR)".

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						STK ALIGN	BFHF NMIGN	Reserved			DIV_0_ TRP	UN ALIGN_ TRP	Res.	USER SET MPEND	NON BASE THRD ENA
						rw	rw				rw	rw		rw	rw

Bits 31:10 Reserved
must be kept cleared (При записи всегда 0)

Bit 9 STKALIGN
Configures stack alignment on exception entry. On exception entry, the processor uses bit 9 of the stacked PSR to indicate the stack alignment. On return from the exception it uses this stacked bit to restore the correct stack alignment.
Конфигурация выравнивания стека при входе в исключение. При входе в исключение процессор использует бит 9, из сохраненного в стеке **PSR**, чтобы задать выравнивание стека. По возвращению из исключения он использует этот бит из регистра, сохраненного в стеке, чтобы восстановить правильное выравнивание стека.
`0:` 4-byte aligned
`1:` 8-byte aligned

Bit 8 BFHFNMIGN
Enables handlers with priority -1 or -2 to ignore data bus faults caused by load and store instructions. This applies to the hard fault, NMI, and FAULTMASK escalated handlers. Set this bit to 1 only when the handler and its data are in absolutely safe memory. The normal use of this bit is to probe system devices and bridges to detect control path problems and fix them.
Дает возможность обработчикам с приоритетом '-1' или '-2' проигнорировать ошибки шины данных, вызванные инструкциями загрузки (*load*) и сохранения (*store*). Это относится к обработчикам исключений от аппаратных ошибок, немаскируемого прерывания, и обработчикам, усиленным до аппаратных с помощью **FAULTMASK**. Устанавливайте этот бит в '1' только тогда, когда обработчик и его данные находятся в абсолютно безопасной области памяти. Обычно, этот бит используют для тестовых образцов устройств и мостов системы, чтобы обнаружить пути обхода проблемы и устранить их.
`0:` Data bus faults caused by load and store instructions cause a lock-up
Ошибки шины данных, вызванные инструкциями загрузки и сохранения, приводят к

блокировке системы.

1: Handlers running at priority -1 and -2 ignore data bus faults caused by load and store instructions. Обработчики, выполняющиеся с приоритетом '-1' и '-2', игнорируют ошибки шины данных, вызванные инструкциями загрузки и сохранения.

Bits 7:5 Reserved

must be kept cleared (При записи всегда 0)

Bit 4 DIV_0_TRP

Enables faulting or halting when the processor executes an SDIV or UDIV instruction with a divisor of 0: Допускает свой обработчик, когда процессор выполняет инструкции **SDIV** или **UDIV** с делителем равным 0:

0: Do not trap divide by 0 (Не перехватывать событие деления на ноль)

1: Trap divide by 0. (Перехватывать событие деления на ноль)

When this bit is set to 0, a divide by zero returns a quotient of 0.
Когда этот бит установлен в '0', то деление на ноль возвращает частное '0'.

Bit 3 UNALIGN_ TRP

Enables unaligned access traps: (Разрешает перехват невыровненного доступа)

0: Do not trap unaligned halfword and word accesses

Не перехватывать событие невыровненного доступа к слову или полуслову

1: Trap unaligned halfword and word accesses.

Перехватывать событие невыровненного доступа к слову или полуслову

If this bit is set to 1, an unaligned access generates a usage fault. Unaligned LDM, STM, LDRD, and STRD instructions always fault irrespective of whether UNALIGN_TRP is set to 1.
Если этот бит установлен в '1', то невыровненный доступ генерирует пользовательскую ошибку. Невыровненный доступ инструкций **LDM**, **STM**, **LDRD**, и **STRD** всегда вызывает системную ошибку, независимо от того, установлен ли **UNALIGN_TRP** в '1'.

Bit 2 Reserved

must be kept cleared (При записи всегда 0)

Bit 1 USERSETMPEND

Enables unprivileged software access to the STIR, see *Software trigger interrupt register (NVIC_STIR) on page 112*:
Разрешает непривилегированному коду доступ к **STIR**, см. раздел 4.2.8 "Регистр программного запуска прерываний (**NVIC_STIR**)".

0: Disable (Доступ запрещен)

1: Enable. (Доступ разрешен)

Bit 0 NONBASETHRDENA

Configures how the processor enters Thread mode.
Конфигурирует, как процессор входит в режим **Thread**.

0: Processor can enter Thread mode only when no exception is active.

Процессор может войти в режим **Thread** только когда нет активных исключений.

1: Processor can enter Thread mode from any level under the control of an EXC_RETURN value, see *Exception return on page 38*.

Процессор может войти в режим **Thread** с любого уровня, заданного значением **EXC_RETURN**, см. параграф "Возврат из исключения" в разделе 2.3.7.

4.3.7 System handler priority registers (SHPRx)
Регистры приоритетов системных обработчиков

The SHPR1-SHPR3 registers set the priority level, 0 to 15 of the exception handlers that have configurable priority.
Регистры **SHPR1-SHPR3** устанавливают уровень приоритета от 0 до 15 для тех обработчиков исключений, которые имеют конфигурируемый приоритет.

SHPR1-SHPR3 are byte accessible. (Эти регистры имеют побайтовый доступ.)

The system fault handlers and the priority field and register for each handler are:
Обработчики системных ошибок и поле приоритета и регистр для каждого обработчика, следующие:

Table 39. System fault handler priority fields
Поля приоритета для обработчиков системных ошибок

Handler	Field	Register description
Memory management fault	PRI_4	"System handler priority register 1 (SCB_SHPR1)"
Bus fault	PRI_5	
Usage fault	PRI_6	
SVCall	PRI_11	"System handler priority register 2 (SCB_SHPR2)" on page 123
PendSV	PRI_14	"System handler priority register 3 (SCB_SHPR3)" on page 124
SysTick	PRI_15	

Each PRI_N field is 8 bits wide, but the processor implements only bits[7:4] of each field, and bits[3:0] read as zero and ignore writes.
Каждое поле **PRI_N** шириной 8 битов, но процессор принимает только биты [7:4] каждого поля, а биты [3:0] читаются как нуль и запись в них игнорируется.
(*В семействе **CL** используются все 8 бит поля.*)

System handler priority register 1 (SCB_SHPR1)

Address offset: 0x18
Reset value: 0x0000 0000
Required privilege: Privileged

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								PRI_6[7:4]				PRI_6[3:0]			
								rw	rw	rw	rw	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRI_5[7:4]				PRI_5[3:0]				PRI_4[7:4]				PRI_4[7:4]			
rw	rw	rw	rw	r	r	r	r	rw	rw	rw	rw	r	r	r	r

Bits 31:24 Reserved
must be kept cleared (При записи всегда 0)

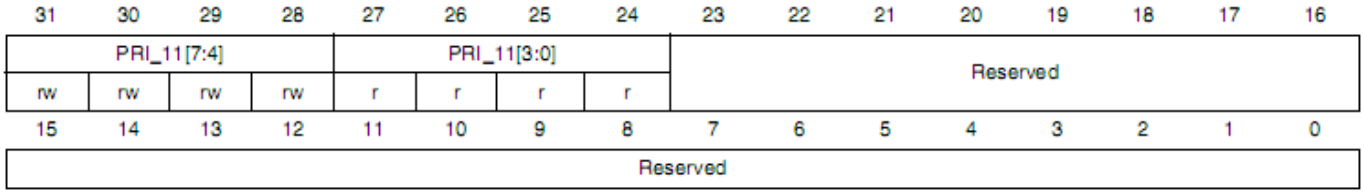
Bits 23:16 PRI_6[7:0]: Priority of system handler 6
usage fault (Приоритет пользовательской ошибки)

Bits 15:8 PRI_5[7:0]: Priority of system handler 5
bus fault (Приоритет ошибки шины)

Bits 7:0 PRI_4[7:0]: Priority of system handler 4
memory management fault (Приоритет ошибки управления памятью)

System handler priority register 2 (SCB_SHPR2)

Address offset: 0x1C
Reset value: 0x0000 0000
Required privilege: Privileged

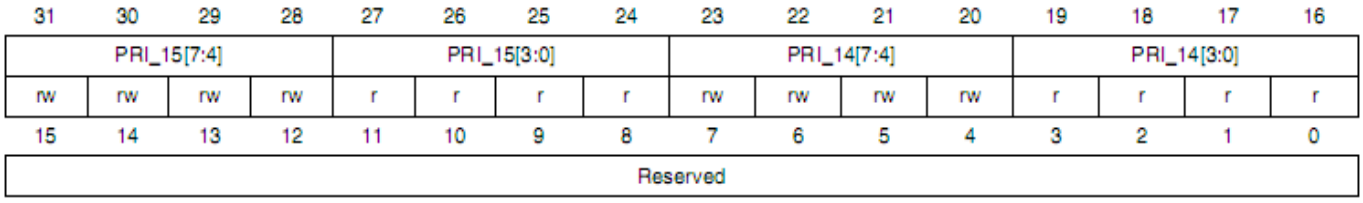


Bits 31:24 PRI_11[7:0]: Priority of system handler 11
SVCall (Приоритет исключения **SVCall**)

Bits 23:0 Reserved,
must be kept cleared (При записи всегда 0)

System handler priority register 3 (SCB_SHPR3)

Address: 0xE000 ED20
Reset value: 0x0000 0000
Required privilege: Privileged



Bits 31:24 PRI_15[7:0]: Priority of system handler 15
SysTick exception (Приоритет системного таймера)

Bits 23:16 PRI_14[7:0]: Priority of system handler 14
PendSV (Приоритет исключения **PendSV**)

Bits 15:0 Reserved
must be kept cleared (При записи всегда 0)

4.3.8 System handler control and state register (SCB_SHCSR)
Регистр управления и состояния системных обработчиков

Address offset: 0x24
Reset value: 0x0000 0000
Required privilege: Privileged

The SHCSR enables the system handlers, and indicates:
Регистр **SHCSR** разрешает системные обработчики и предоставляет следующую информацию:

- The pending status of the bus fault, memory management fault, and SVC exceptions
Статус запроса от ошибки шины, ошибки управления памятью, и исключения **SVC**
- The active status of the system handlers.
Состояние активности системных обработчиков.

If you disable a system handler and the corresponding fault occurs, the processor treats the fault as a hard fault. Если вы отключили системный обработчик, и происходит соответствующая ошибка, процессор обрабатывает такую ошибку, как аппаратную.

You can write to this register to change the pending or active status of system exceptions. An OS kernel can write to the active bits to perform a context switch that changes the current exception type.
Вы можете произвести запись в этот регистр, чтобы изменить статус запроса или активности системных исключений. Ядро операционной системы может писать в биты активности, чтобы выполнить переключение контекста, что изменяет тип текущего исключения.

- Software that changes the value of an active bit in this register without correct adjustment to the stacked content can cause the processor to generate a fault exception. Ensure software that writes to this register retains and subsequently restores the current active status.
Программа, которая изменяет значение бита активности в этом регистре без соответствующей настройки содержимого, сохраненного в стеке, может привести к тому, что процессор сгенерирует ошибку. Убедитесь, что программа, которая делает запись в этот регистр, сохраняет и, впоследствии, восстанавливает текущее состояние активности.
- After you have enabled the system handlers, if you have to change the value of a bit in this register you must use a read-modify-write procedure to ensure that you change only the required bit. После того, как вы разрешили системные обработчики, если вы должны изменить значение бита в этом регистре, вы должны использовать процедуру "чтение-изменение-запись", чтобы гарантировать, что вы изменяете только необходимый бит.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													USG FAULT ENA	BUS FAULT ENA	MEM FAULT ENA
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SV CALL PEND ED	BUS FAULT PEND ED	MEM FAULT PEND ED	USG FAULT PEND ED	SYS TICK ACT	PEND SV ACT	Res.	MONIT OR ACT	SV CALL ACT	Reserved			USG FAULT ACT	Res.	BUS FAULT ACT	MEM FAULT ACT
rw	rw	rw	rw	rw	rw		rw	rw				rw		rw	rw

Bits 31:19 Reserved
must be kept cleared (При записи всегда 0)

Bit 18 USGFAULTENA: Usage fault enable bit
Бит разрешения пользовательской ошибки
set to 1 to enable[1] (поставьте в '1', чтобы разрешить)

Bit 17 BUSFAULTENA: Bus fault enable bit (Бит разрешения ошибки шины)
set to 1 to enable[1] (поставьте в '1', чтобы разрешить)

Bit 16 MEMFAULTENA: Memory management fault enable bit
Бит разрешения ошибки управления памятью
set to 1 to enable[1] (поставьте в '1', чтобы разрешить)

Bit 15 SVCALLPENDED: SVC call pending bit (Бит запроса вызова SVC)
reads as 1 if exception is pending[2] (Читается как '1', если есть запрос исключения)

Bit 14 BUSFAULTPENDED: Bus fault exception pending bit
Бит запроса исключения ошибки шины
reads as 1 if exception is pending[2] (Читается как '1', если есть запрос исключения)

Bit 13 MEMFAULTPENDED: Memory management fault exception pending bit
Бит запроса исключения ошибки управления памятью
reads as 1 if exception is pending[2] (Читается как '1', если есть запрос исключения)

Bit 12 USGFAULTPENDED: Usage fault exception pending bit
Бит запроса исключения ошибки пользовательской ошибки
reads as 1 if exception is pending[2] (Читается как '1', если есть запрос исключения)

Bit 11 SYSTICKACT: SysTick exception active bit
Бит активности системного таймера
reads as 1 if exception is active[3] (Читается как '1', если исключение активно)

Bit 10 PENDSVACT: PendSV exception active bit
Бит активности исключения PendSV
reads as 1 if exception is active (Читается как '1', если исключение активно)

Bit 9 Reserved
must be kept cleared (При записи всегда 0)

Bit 8 MONITORACT: Debug monitor active bit (Бит активности монитора отладчика)
reads as 1 if Debug monitor is active (Читается как '1', если активен монитор отладчика)

Bit 7 SVCALLACT: SVC call active bit (Бит активности вызова SVC)
reads as 1 if SVC call is active (Читается как '1', если вызов **SVC** активен)

Bits 6:4 Reserved
must be kept cleared (При записи всегда 0)

Bit 3 USGFAULTACT: Usage fault exception active bit
Бит активности исключения пользовательской ошибки
reads as 1 if exception is active (Читается как '1', если исключение активно)

Bit 2 Reserved
must be kept cleared (При записи всегда 0)

Bit 1 BUSFAULTACT: Bus fault exception active bit

Бит активности исключения ошибки шины
reads as 1 if exception is active (Читается как '1', если исключение активно)

Bit 0 MEMFAULTACT: Memory management fault exception active bit
Бит активности исключения управления памятью
reads as 1 if exception is active (Читается как '1', если исключение активно)

1. Enable bits, set to 1 to enable the exception, or set to 0 to disable the exception.
Биты разрешения, установите бит в '1', чтобы разрешить исключение, или установите его в '0', чтобы выключить исключение.

2. Pending bits, read as 1 if the exception is pending, or as 0 if it is not pending. You can write to these bits to change the pending status of the exceptions.
Биты запроса, читается как '1', если есть запрос от исключения, или как '0', если нет запроса. Вы можете производить запись в эти биты, чтобы изменить статус запроса от исключения.

3. Active bits, read as 1 if the exception is active, or as 0 if it is not active. You can write to these bits to change the active status of the exceptions, but see the Caution in this section.
Биты активности, читаются как '1', если исключение является активным, или как '0', если оно не активно. Вы можете производить запись в эти биты, чтобы изменить статус активности исключений, но см. **Предостережение** ниже в этом разделе.

4.3.9 Configurable fault status register (SCB_CFSR)
Регистр статуса конфигурируемых ошибок

Address offset: 0x28
Reset value: 0x0000 0000
Required privilege: Privileged

The CFSR is byte accessible. You can access the CFSR or its subregisters as follows:
Регистр **CFSR** имеет побайтовый доступ. Вы можете обратиться к **CFSR** или его субрегистрам следующим образом:

- Access the complete CFSR with a word access to 0xE000ED28
Пословный доступ к полному **CFSR** по адресу **0xE000ED28**

- Access the MMFSR with a byte access to 0xE000ED28
Байтовый доступ к **MMFSR** по адресу **0xE000ED28**

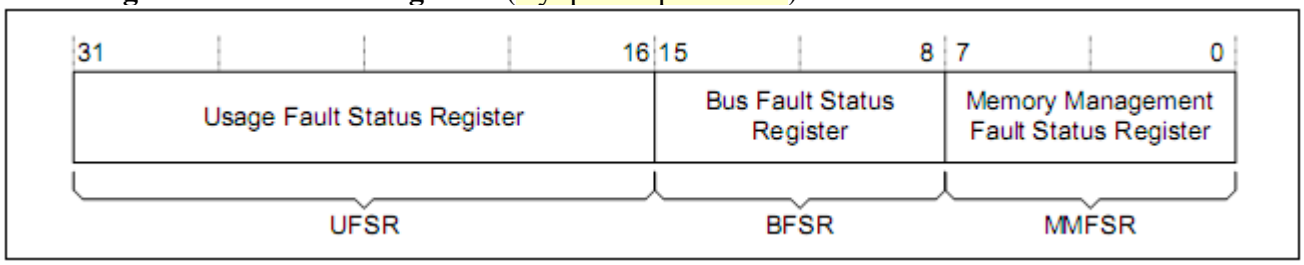
- Access the MMFSR and BFSR with a halfword access to 0xE000ED28
Полусловный доступ к **MMFSR** и **BFSR** по адресу **0xE000ED28**

- Access the BFSR with a byte access to 0xE000ED29
Байтовый доступ к **BFSR** по адресу **0xE000ED29**

- Access the UFSR with a halfword access to 0xE000ED2A.
Полусловный доступ к **UFSR** по адресу **0xE000ED2A**

The CFSR indicates the cause of a memory management fault, bus fault, or usage fault.
CFSR указывает причину ошибки управления памятью, ошибки шины, или пользовательской ошибки.

Figure 19. CFSR subregisters (Субрегистры **CFSR**)



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						DIVBY ZERO	UNALI GNED	Reserved				NOCP	INVPC	INV STATE	UNDEF INSTR
						rc_w1	rc_w1					rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BFARV ALID	Reserved		STK ERR	UNSTK ERR	IMPRE CIS ERR	PRECI S ERR	IBUS ERR	MMAR VALID	Reserved		MSTK ERR	M UNSTK ERR	Res.	DACC VIOL	IACC VIOL
rw			rw	rw	rw	rw	rw	rw			rw	rw		rw	rw

Bits 31:26 Reserved

must be kept cleared (При записи всегда 0)

Bit 25 DIVBYZERO: Divide by zero usage fault

Пользовательская ошибка деления на ноль

When the processor sets this bit to 1, the PC value stacked for the exception return points to the instruction that performed the divide by zero. Enable trapping of divide by zero by setting the DIV_0_TRP bit in the CCR to 1, see *Configuration and control register (SCB_CCR) on page 121*.

Когда процессор устанавливает этот бит в '1', то значение **PC** сохраненное в стеке как точка возвращения из исключения, указывает на инструкцию, которая выполнила деление на ноль. Разрешите ловушку для деления на ноль, установив в '1' бит **DIV_0_TRP** в регистре **CCR**, см. раздел 4.3.6 "Регистр конфигурации и управления (**SCB_CCR**)".

0: No divide by zero fault, or divide by zero trapping not enabled

Нет деления на ноль или ловушка для деления на ноль не разрешена

1: The processor has executed an SDIV or UDIV instruction with a divisor of 0.

Процессор выполнил инструкцию **SDIV** или **UDIV** с нулем в качестве делителя.

Bit 24 UNALIGNED: Unaligned access usage fault:

Пользовательская ошибка невыровненного доступа

Enable trapping of unaligned accesses by setting the UNALIGN_TRP bit in the CCR to 1, see *Configuration and control register (SCB_CCR) on page 121*. Unaligned LDM, STM, LDRD, and STRD instructions always fault irrespective of the setting of UNALIGN_TRP.

Разрешите ловушку для невыровненного доступа, устанавив бит в регистре **CCR**, см. раздел 4.3.6 "Регистр конфигурации и управления (**SCB_CCR**)". Невыровненные инструкции **LDM**, **STM**, **LDRD** и **STRD** всегда приводят к ошибке, независимо от установки бита **UNALIGN_TRP**.

0: No unaligned access fault, or unaligned access trapping not enabled

Нет ошибки невыровненного доступа или ловушка для невыровненного доступа не разрешена

1: the processor has made an unaligned memory access.

Процессор выполнил инструкцию невыровненного доступа

Bits 23:20 Reserved

must be kept cleared (При записи всегда 0)

Bit 19 NOCP: No coprocessor usage fault.
Пользовательская ошибка - нет сопроцессора

The processor does not support coprocessor instructions:
Процессор не поддерживает инструкции сопроцессора:

`0:` No usage fault caused by attempting to access a coprocessor
Нет пользовательской ошибки, вызванной попыткой доступа к сопроцессору

`1:` the processor has attempted to access a coprocessor.
Процессор выполнил попытку доступа к сопроцессору.

Bit 18 INVPC: Invalid PC load usage fault:
Пользовательская ошибка - неверная загрузка в PC

Caused by an invalid PC load by EXC_RETURN
When this bit is set to 1, the PC value stacked for the exception return points to the instruction that tried to perform the illegal load of the PC.
Вызвана недопустимым значением **PC** в поле **EXC_RETURN**
Когда этот бит установлен в "1', то значение **PC**, сохраненное в стеке, как точка возвращения из исключения, указывает на инструкцию, которая попыталась выполнить незаконную загрузку **PC**.

`0:` No invalid PC load usage fault
Нет пользовательской ошибки недопустимой загрузки **PC**

`1:` The processor has attempted an illegal load of EXC_RETURN to the PC, as a result of an invalid context, or an invalid EXC_RETURN value.
Процессор выполнил попытку недопустимой загрузки поля **EXC_RETURN** в регистре **PC**, как результат нарушения контекста, или просто неверное значение **EXC_RETURN**.

Bit 17 INVSTATE: Invalid state usage fault:
Пользовательская ошибка - недопустимое состояние

When this bit is set to 1, the PC value stacked for the exception return points to the instruction that attempted the illegal use of the EPSR. This bit is not set to 1 if an undefined instruction uses the EPSR. Когда этот бит установлен в '1', то значение **PC**, сохраненное в стеке, как точка возвращения из исключения, указывает на инструкцию, которая попыталась выполнить незаконное использование **EPSR**. Этот бит не ставится в '1', если неопределенная инструкция использует **EPSR**.

`0:` No invalid state usage fault
Нет пользовательской ошибки недопустимого состояния

`1:` The processor has attempted to execute an instruction that makes illegal use of the EPSR.
Процессор пытался выполнить инструкцию, которая привела к невозможности использовать **EPSR**.

Bit 16 UNDEFINSTR: Undefined instruction usage fault:
Пользовательская ошибка - неопознанная инструкция

When this bit is set to 1, the PC value stacked for the exception return points to the undefined instruction. An undefined instruction is an instruction that the processor cannot decode.
Когда этот бит установлен в '1', то значение **PC**, сохраненное в стеке, как точка возвращения из исключения, указывает на неопознанную инструкцию. Неопознанная инструкция - это инструкция, которую не может декодировать процессор.

`0:` No undefined instruction usage fault
Нет пользовательской ошибки - неопознанная инструкция

`1:` The processor has attempted to execute an undefined instruction.
Процессор пытался выполнить неопознанную инструкцию.

Bit 15 BFARVALID: Bus Fault Address Register (BFAR) valid flag:
Флаг валидности адресного регистра при ошибке шины

The processor sets this bit to 1 after a bus fault where the address is known. Other faults can set this bit to 0, such as a memory management fault occurring later. If a bus fault occurs and is escalated to a hard fault because of priority, the hard fault handler must set this bit to 0. This prevents problems if returning to a stacked active bus fault handler whose BFAR value has been overwritten.
Процессор устанавливает этот бит в '1' после ошибки шины, когда адрес известен. Другие ошибки могут установить этот бит в '0', такие как ошибка управления памятью, случившаяся позже. Если происходит ошибка шины, которая, из-за приоритета, усиливается до аппаратной ошибки, то обработчик аппаратной ошибки должен установить этот бит в '0'. Это предотвращает проблемы, когда идет возврат в прерванный обработчик ошибки шины, где значение **BFAR** было переписано.

`0:` Value in BFAR is not a valid fault address
Значение в **BFAR** не является правильным адресом ошибки

`1:` BFAR holds a valid fault address. (**BFAR** содержит правильный адрес ошибки.)

Bits 14:13 Reserved

must be kept cleared (При записи всегда 0)

Bit 12 STKERR: Bus fault on stacking for exception entry
Ошибка шины при сохранении в стек, при входе в исключение

When the processor sets this bit to 1, the SP is still adjusted but the values in the context area on the stack might be incorrect. The processor does not write a fault address to the BFAR.
Когда процессор устанавливает этот бит в '1', то **SP** все еще корректен, но значения контекста в стеке могут быть неправильными. Процессор не записывает адрес ошибки в **BFAR**.

`0:` No stacking fault (Нет ошибки сохранения в стеке)

`1:` Stacking for an exception entry has caused one or more bus faults.

Операция сохранения в стеке при входе в исключение привела к одной или более ошибкам шины.

Bit 11 UNSTKERR: Bus fault on unstacking for a return from exception
Ошибка шины при восстановлении из стека, при выходе из исключения

This fault is chained to the handler. This means that when the processor sets this bit to 1, the original return stack is still present. The processor does not adjust the SP from the failing return, does not performed a new save, and does not write a fault address to the BFAR.
Обработчик этой ошибки "садится на хвост" текущему обработчику. Это означает, что, когда процессор устанавливает этот бит в '1', оригинальный стек возврата все еще присутствует. Процессор не корректирует **SP** при неудаче возврата, не делает нового сохранения в стек, и не записывает адрес ошибки в **BFAR**.

`0:` No unstacking fault (Нет ошибки при восстановлении из стека)

`1:` Unstack for an exception return has caused one or more bus faults.

Операция восстановления из стека привела к одной или более ошибкам шины.

Bit 10 IMPRECISERR: Imprecise data bus error
Неопределенная ошибка шины данных

When the processor sets this bit to 1, it does not write a fault address to the BFAR. This is an asynchronous fault. Therefore, if it is detected when the priority of the current process is higher than the bus fault priority, the bus fault becomes pending and becomes active only when the processor returns from all higher priority processes. If a precise fault occurs before the processor enters the handler for the imprecise bus fault, the handler detects both IMPRECISERR set to 1 and one of the precise fault status bits set to 1.
Когда процессор устанавливает этот бит в '1', он не записывает адрес ошибки в **BFAR**. Это ошибка асинхронного типа. Поэтому, если она обнаружена, когда приоритет текущего процесса выше чем приоритет ошибки шины, то ошибка шины откладывается и становится активной только тогда, когда процессор возвращается из всех более высоко-приоритетных процессов. Если явная ошибка шины происходит прежде, чем процессор войдет в

обработчик для неопределенной ошибки шины данных, то обработчик обнаруживает обе причины, и бит **IMPRECISERR** установлен в '1' и один из явных битов ошибки состояния установлен в '1'.

0: No imprecise data bus error (Нет неопределенной ошибки шины данных)

1: A data bus error has occurred, but the return address in the stack frame is not related to the instruction that caused the error.

Есть ошибка шины данных, но адрес возврата, сохраненный в стеке, не связан с инструкцией, которая вызвала ошибку.

Bit 9 PRECISERR: Precise data bus error (Определенная ошибка шины данных)

When the processor sets this bit is 1, it writes the faulting address to the BFAR.

Когда процессор устанавливает этот бит в '1', он записывает адрес ошибки в **BFAR**

0: No precise data bus error (Нет определенной ошибки шины данных)

1: A data bus error has occurred, and the PC value stacked for the exception return points to the instruction that caused the fault.

Есть ошибка шины данных, и значение **PC**, сохраненное в стеке, как точка возвращения из исключения, указывает на инструкцию, которая вызвала ошибку.

Bit 8 IBUSERR: Instruction bus error (Ошибка шины инструкций)

The processor detects the instruction bus error on prefetching an instruction, but it sets the IBUSERR flag to 1 only if it attempts to issue the faulting instruction. When the processor sets this bit is 1, it does not write a fault address to the BFAR.

Процессор обнаруживает ошибку шины инструкций при предварительной выборке инструкции, но это устанавливает флаг **IBUSERR** в '1' только при попытке запустить ошибочную инструкцию. Когда процессор устанавливает этот бит в '1', он не пишет адрес ошибки в **BFAR**.

0: No instruction bus error (Нет ошибки шины инструкций)

1: Instruction bus error. (Есть ошибка шины инструкций)

Bit 7 MMARVALID: Memory Management Fault Address Register (MMAR) valid flag
Флаг валидности адреса ошибки управления памятью

If a memory management fault occurs and is escalated to a hard fault because of priority, the hard fault handler must set this bit to 0. This prevents problems on return to a stacked active memory management fault handler whose MMAR value has been overwritten.

Если происходит ошибка управления памятью и она усиливается до аппаратной, вследствии ее приоритета, то обработчик аппаратной ошибки должен установить этот бит в '0'. Это предотвращает проблемы, когда идет возврат в прерванный обработчик ошибки управления памятью, где значение **MMAR** было переписано.

0: Value in MMAR is not a valid fault address

Значение в **MMAR** не является правильным адресом ошибки

1: MMAR holds a valid fault address. (**MMAR** содержит правильный адрес ошибки.)

Bits 6:5 Reserved

must be kept cleared (При записи всегда 0)

Bit 4 MSTKERR: Memory manager fault on stacking for exception entry
Ошибка управления памятью при сохранении в стек при входе в исключение

When this bit is 1, the SP is still adjusted but the values in the context area on the stack might be incorrect. The processor has not written a fault address to the MMAR.

Когда этот бит '1', **SP** все еще корректен, но значения контекста в стеке могут быть неправильными. Процессор не записывает адрес ошибки в **MMAR**.

0: No stacking fault (Нет ошибки при сохранении в стек)

1: Stacking for an exception entry has caused one or more access violations.

Операция сохранения в стеке, при входе в исключение, привела к одной или более ошибкам управления памятью

Bit 3 MUNSTKERR: Memory manager fault on unstacking for a return from exception
Ошибка управления памятью при восстановлении из стека, при выходе из исключения

This fault is chained to the handler. This means that when this bit is 1, the original return stack is still present. The processor has not adjusted the SP from the failing return, and has not performed a new save. The processor has not written a fault address to the MMAR.
Обработчик этой ошибки "садится на хвост" текущему обработчику. Это означает, что, когда процессор устанавливает этот бит в '1', оригинальный стек возврата все еще присутствует. Процессор не корректирует **SP** при неудаче возврата, и не делает нового сохранения в стек. Процессор не записывает адрес ошибки в **MMAR**.

`0:` No unstacking fault (Нет ошибки при восстановлении из стека)

`1:` Unstack for an exception return has caused one or more access violations.
Операция восстановления из стека, при выходе из исключения, привела к одной или более ошибкам управления памятью.

Bit 2 Reserved

must be kept cleared (При записи всегда 0)

Bit 1 DACCVIOL: Data access violation flag (Флаг нарушения правил доступа к данным)

When this bit is 1, the PC value stacked for the exception return points to the faulting instruction. The processor has loaded the MMAR with the address of the attempted access.
Когда этот бит установлен в '1', то значение **PC**, сохраненное в стеке, как точка возвращения из исключения, указывает на инструкцию, которая вызвала ошибку. Процессор загружает в **MMAR** адрес попытки доступа.

`0:` No data access violation fault
Нет ошибки нарушения правил доступа к данным

`1:` The processor attempted a load or store at a location that does not permit the operation.
Процессор пытался загрузить или сохранить данные в области, которая не разрешает эти операции.

Bit 0 IACCVIOL: Instruction access violation flag
Флаг нарушения правил доступа для инструкций

This fault occurs on any access to an XN region. When this bit is 1, the PC value stacked for the exception return points to the faulting instruction. The processor has not written a fault address to the MMAR.
Эта ошибка происходит при любом доступе к области **XN**. Когда этот бит '1', то значение **PC**, сохраненное в стеке, как точка возвращения из исключения, указывает на инструкцию, которая вызвала ошибку. Процессор не записывает адрес ошибки в **MMAR**.

`0:` No instruction access violation fault
Нет ошибки нарушения правил для инструкций

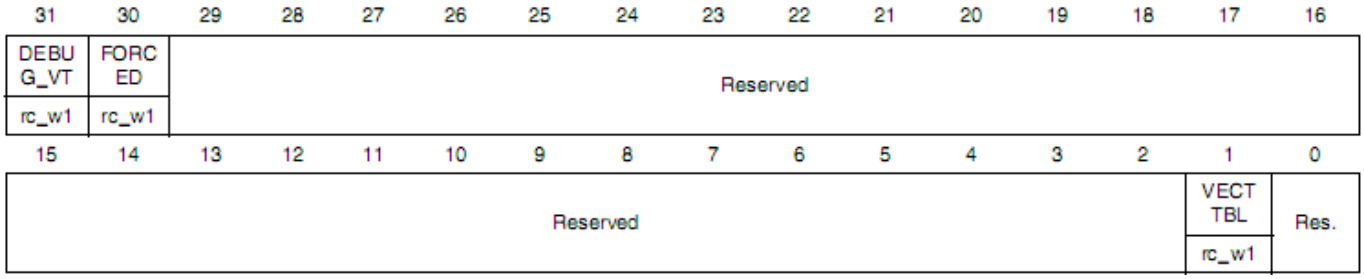
`1:` The processor attempted an instruction fetch from a location that does not permit execution.
Процессор пытался запустить инструкцию из области, которая не допускает исполнение.

4.3.10 Hard fault status register (SCB_HFSR) (Регистр статуса аппаратных ошибок)

Address offset: 0x2C
Reset value: 0x0000 0000
Required privilege: Privileged

The HFSR gives information about events that activate the hard fault handler. This register is read, write to clear. **HFSR** дает информацию о событиях, которые активизируют обработчик аппаратных ошибок. Этот регистр для чтения и записи, но только в целях очистки.

This means that bits in the register read normally, but writing 1 to any bit clears that bit to 0. Это означает, что биты в регистре читаются нормально, но запись '1' в любой бит, очищает этот бит в '0'.



Bit 31 DEBUG_VT: Reserved for Debug use. (Зарезервирован в целях отладки)
When writing to the register you must write 0 to this bit, otherwise behavior is unpredictable. При записи в этот регистр вы должны писать '0' в этот бит, иначе поведение будет непредсказуемым.

Bit 30 FORCED: Forced hard fault (Вызывает аппаратную ошибку)
Indicates a forced hard fault, generated by escalation of a fault with configurable priority that cannot be handles, either because of priority or because it is disabled: When this bit is set to 1, the hard fault handler must read the other fault status registers to find the cause of the fault.
Указывает на принудительную аппаратную ошибку, вызванную усилением другой ошибки с конфигурируемым приоритетом, которую нельзя обработать иначе, либо вследствии ее низкого приоритета, либо потому, что она отключена: Когда этот бит установлен в '1', то обработчик аппаратной ошибки должен читать другие регистры состояния ошибок, чтобы найти реальную причину ошибки.
- `0:` No forced hard fault (Нет эффекта)
- `1:` Forced hard fault. (Вызывает аппаратную ошибку)

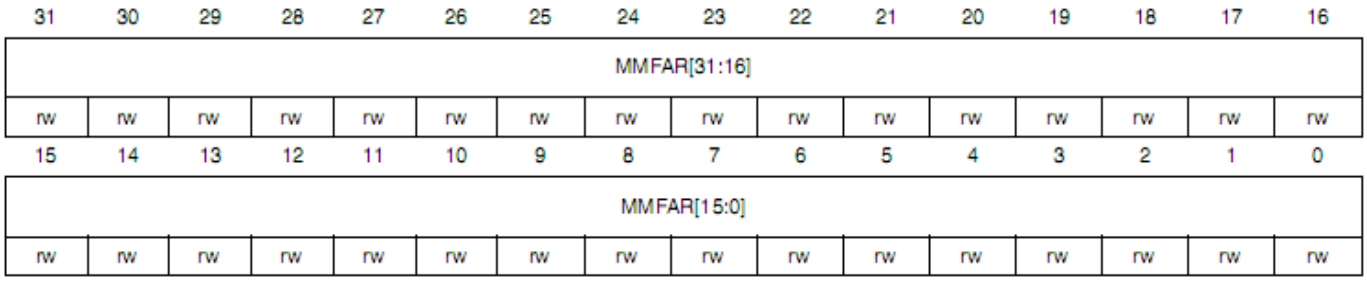
Bits 29:2 Reserved
must be kept cleared (При записи всегда 0)

Bit 1 VECTTBL: Vector table hard fault (Ошибка шины от чтения таблицы векторов)
Indicates a bus fault on a vector table read during exception processing: This error is always handled by the hard fault handler. When this bit is set to 1, the PC value stacked for the exception return points to the instruction that was preempted by the exception.
Указывает на ошибку шины, вызванную чтением таблицы векторов во время обработки исключения: Эта ошибка всегда обрабатывается обработчиком аппаратной ошибки. Когда этот бит установлен в '1', то значение **PC**, сохраненное в стеке, как точка возвращения из исключения, указывает на инструкцию, которая вызвала исключение.
- `0:` No bus fault on vector table read (Нет ошибки шины от чтения таблицы векторов)
- `1:` Bus fault on vector table read. (Есть ошибка шины от чтения таблицы векторов)

Bit 0 Reserved
must be kept cleared (При записи всегда 0)

4.3.11 Memory management fault address register (SCB_MMFAR)
Регистр адреса ошибки управления памятью

Address offset: 0x34
Reset value: undefined
Required privilege: Privileged



Bits 31:0 MMFAR[31:0]: Memory management fault address
Адрес ошибки управления памятью

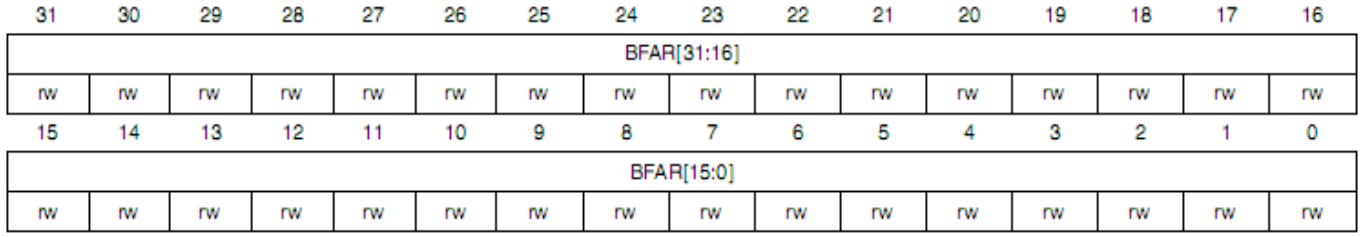
When the MMARVALID bit of the MMFSR is set to 1, this field holds the address of the location that generated the memory management fault.
Когда бит **MMARVALID** регистра **MMFSR** установлен в '1', то это поле содержит адрес, который сгенерировал ошибку управления памятью.

When an unaligned access faults, the address is the actual address that faulted. Because a single read or write instruction can be split into multiple aligned accesses, the fault address can be any address in the range of the requested access size.
Когда есть ошибка невыровненного доступа, то это реальный адрес, который вызвал ошибку. Поскольку инструкция одиночного чтения или записи может быть расчленена на доступы к отдельным частям элемента, то адрес ошибки может быть любым адресом в диапазоне требуемого размера доступа.

Flags in the MMFSR register indicate the cause of the fault, and whether the value in the MMFAR is valid. See *Configurable fault status register (SCB_CFSR) on page 126*.
Флаги в регистре **MMFSR** указывают причину ошибки, и является ли значение в **MMFAR** правильным. См. раздел 4.3.9 "Регистр статуса конфигурируемых ошибок (**SCB_CFSR**)".

4.3.12 Bus fault address register (SCB_BFAR)
Регистр адреса ошибки шины

Address offset: 0x38
Reset value: undefined
Required privilege: Privileged



Bits 31:0 BFAR[31:0]: Bus fault address (адрес ошибки шины)

When the BFARVALID bit of the BFSR is set to 1, this field holds the address of the location that generated the bus fault. When an unaligned access faults the address in the BFAR is the one requested by the instruction, even if it is not the address of the fault. Flags in the BFSR register indicate the cause of the fault, and whether the value in the BFAR is valid. See *Configurable fault status register (SCB_CFSR) on page 126*.

Когда бит **BFARVALID** в регистре **BFSR** установлен в '1', это поле содержит адрес, который сгенерировал ошибку шины. Когда запись адреса в **BFAR** вызвала ошибка невыровненного доступа, то это тот адрес, который затребовала инструкция, а не тот, что реально вызвал ошибку. Флаги в регистре **BFSR** указывают причину ошибки, и является ли значение в **BFAR** правильным. См. раздел 4.3.9 "Регистр статуса конфигурируемых ошибок (SCB_CFSR)".

4.3.13 System control block design hints and tips
Программирование регистров блока SCB: рекомендации и советы

Ensure software uses aligned accesses of the correct size to access the system control block registers (Убедитесь, что программа использует выровненный доступ правильного размера для обращения к регистрам блока **SCB)**:

- except for the CFSR and SHPR1-SHPR3, it must use aligned word accesses
за исключением **CFSR** и **SHPR1-SHPR3** нужно использовать пословный выровненный доступ
- for the CFSR and SHPR1-SHPR3 it can use byte or aligned halfword or word accesses.
для **CFSR** и **SHPR1-SHPR3** можно использовать побайтовый или полусловный выровненный доступ.

The processor does not support unaligned accesses to system control block registers.
Процессор не поддерживает невыровненный доступ к регистрам **SCB**.

In a fault handler. to determine the true faulting address:
В обработчике ошибки, чтобы определить истинный адрес ошибки:

1. Read and save the MMFAR or BFAR value.
Прочитайте и сохраните значение **MMFAR** или **BFAR**.
2. Read the MMARVALID bit in the MMFSR, or the BFARVALID bit in the BFSR. The MMFAR or BFAR address is valid only if this bit is 1.
Прочитайте бит **MMARVALID** в регистре **MMFSR**, или бит **BFARVALID** в регистре **BFSR**. Адрес **MMFAR** или **BFAR** правилен только тогда, когда этот бит '1'.

Software must follow this sequence because another higher priority exception might change the MMFAR or BFAR value. For example, if a higher priority handler preempts the current fault handler, the other fault might change the MMFAR or BFAR value.
Программа должно следовать этой последовательности, так как другое более высоко-приоритетное исключение может изменить значение **MMFAR** или **BFAR**. Например, если более высоко-приоритетный обработчик прерывает текущий обработчик ошибки, то другая ошибка может изменить значение **MMFAR** или **BFAR**.

4.3.14 SCB register map (Карта регистров блока SCB)

The table provides shows the System control block register map and reset values. The base address of the SCB register block is 0xE000 ED00. (Таблица показывает карту регистров блока **SCB** и их значения после сброса. Базовый адрес блока регистров **SCB - 0xE000 ED00)**.

Table 40. SCB register map and reset values (Карта регистров **SCB** и их значения после сброса)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SCB_CPUID	Implementer								Variant				Constant				PartNo												Revision			
	Reset Value	0	1	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	0	0	0	0	1	0	0	0	1	1	0	0	0	1
0x04	SCB_ICSR	NMIPENDSET	Reserved		PENDSVSET	PENDSVCLR	PENDSTSET	PENDSTCLR	Reserved	ISRPENDING	VECTPENDING[9:0]											RETOBASE	Reserved		VECTACTIVE[8:0]								
	Reset Value	0			0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0
0x08	SCB_VTOR	Reserved	TABLEOFF[29:9]																						Reserved								
	Reset Value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											
0x0C	SCB_AIRCR	VECTKEY[15:0]																ENDIANESS	Reserved				PRIGROUP[2:0]		Reserved					SYSRESETREQ	VECTCLRACTIVE	VECTRESET
	Reset Value	1	1	1	1	1	0	1	0	0	0	0	0	0	1	0	1	0					0	0	0					0	0	0
0x10	SCB_SCR	Reserved																											SEVONPEND	Reserved	SLEEPDEEP	SLEEPONEXIT	Reserved
	Reset Value																												0		0	0	
0x14	SCB_CCR	Reserved																								STKALIGN	BFHFNIGN	Reserved	DIV_0_TRP	UNALIGN_TRP	Reserved	USERSETMPEND	NONBASETHRDENA
	Reset Value																									1	0		0	0		0	0
0x18	SCB_SHPR1	Reserved				PRI6								PRI5												PRI4							
	Reset Value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	SCB_SHPR2	PRI11								Reserved																							
	Reset Value	0	0	0	0	0	0	0	0																								
0x20	SCB_SHPR3	PRI15								PRI14								Reserved															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x24	SCB_SHCRS	Reserved													USG FAULT ENA	BUS FAULT ENA	MEM FAULT ENA	SV CALL PENDED	BUS FAULT PENDED	MEM FAULT PENDED	USG FAULT PENDED	SYS TICK ACT	PENDSV ACT	Reserved	MONITOR ACT	SV CALL ACT	Reserved			USG FAULT ACT	Reserved	BUS FAULT ACT	MEM FAULT ACT
	Reset Value														0	0	0	0	0	0	0	0	0		0	0				0		0	0
0x28	SCB_CFSR	UFSR																BFSR								MMFSR							
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	SCB_HFSR	DEBUG_VT	FORCED	Reserved																												VECTTBL	Reserved
	Reset Value	0	0																													0	
0x34	SCB_MMAR	MMAR[31:0]																															
	Reset Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x38	SCB_BFAR	BFAR[31:0]																															
	Reset Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

4.4 SysTick timer (STK) (Системный таймер)

4.4.1 SysTick control and status register (STK_CTRL)
Регистр управления и статуса системного таймера
4.4.2 SysTick reload value register (STK_LOAD)
(Регистр значения перегрузки системного таймера)
4.4.3 SysTick current value register (STK_VAL)
(Регистр текущего значения системного таймера)
4.4.4 SysTick calibration value register (STK_CALIB)
Регистр значения калибровки системного таймера
4.4.5 SysTick design hints and tips
Программирование системного таймера: рекомендации и советы
4.4.6 SysTick register map (Карта регистров системного таймера)

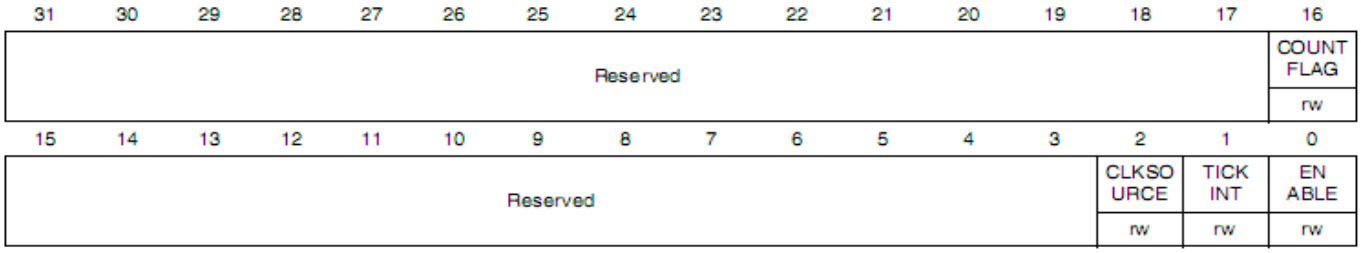
The processor has a 24-bit system timer, SysTick, that counts down from the reload value to zero, reloads (wraps to) the value in the LOAD register on the next clock edge, then counts down on subsequent clocks.
Процессор имеет 24-х битовый системный таймер **SysTick**, который считает в обратном порядке, от значения перезагрузки до нуля, перегружает значение в регистре **LOAD** на следующем фронте синхроимпульса, затем опять считает в обратном порядке на последующих тактах.

When the processor is halted for debugging the counter does not decrement.
Когда процессор остановлен для отладки, счетчик тоже останавливается.

4.4.1 SysTick control and status register (STK_CTRL)
Регистр управления и статуса системного таймера

Address offset: 0x00
Reset value: 0x0000 0004
Required privilege: Privileged

The SysTick CTRL register enables the SysTick features.
Регистр **CTRL** системного таймера разрешает его.



Bits 31:17 Reserved
must be kept cleared. (При записи всегда 0)

Bit 16 COUNTFLAG:
Returns 1 if timer counted to 0 since last time this was read.
Возвращает '1', если таймер успел досчитать до нуля, с тех пор, как последний раз было считано его значение.

Bits 15:3 Reserved
must be kept cleared. (При записи всегда 0)

Bit 2 CLKSOURCE: Clock source selection

Selects the clock source. (Выбор источника тактового сигнала)

`0:` AHB/8

`1:` Processor clock (AHB)

Bit 1 TICKINT: SysTick exception request enable

Разрешение запроса исключения от системного таймера

`0:` Counting down to zero does not assert the SysTick exception request (Нет разрешения)

`1:` Counting down to zero to asserts the SysTick exception request.

Счет вниз до нуля вызывает запрос исключения **SysTick**.

Note: Software can use COUNTFLAG to determine if SysTick has ever counted to zero.

Программа может использовать **COUNTFLAG**, чтобы определить, было ли когда-либо обнуление таймера.

Bit 0 ENABLE: Counter enable (Разрешение счетчика)

Enables the counter. When ENABLE is set to 1, the counter loads the RELOAD value from the LOAD register and then counts down. On reaching 0, it sets the COUNTFLAG to 1 and optionally asserts the SysTick depending on the value of TICKINT. It then loads the RELOAD value again, and begins counting.

Когда бит **ENABLE** установлен в '1', счетчик загружается значением **RELOAD** из регистрира **LOAD**, а затем считает в обратном порядке. При достижении нуля, бит **COUNTFLAG** ставится в '1' и возможно ставится запрос на **SysTick**, в зависимости от значения бита **TICKINT**. Счетчик загружает значение **RELOAD** снова, и начинает счет.

`0:` Counter disabled (Таймер выключен)

`1:` Counter enabled (Таймер включен)

4.4.2 SysTick reload value register (STK_LOAD)

Регистр значения перегрузки системного таймера

Address offset: 0x04
Reset value: 0x0000 0000
Required privilege: Privileged

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								RELOAD[23:16]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RELOAD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved

must be kept cleared. (При записи всегда 0)

Bits 23:0 RELOAD[23:0]: RELOAD value (Значение перегрузки)

The LOAD register specifies the start value to load into the VAL register when the counter is enabled and when it reaches 0. Регистр **LOAD** определяет начальное значение, которое будет загружаться в регистр **VAL**, когда счетчик разрешен и когда он достигает 0.

Calculating the RELOAD value (Расчет значения перегрузки)

The RELOAD value can be any value in the range 0x00000001-0x00FFFFFF. A start value of 0 is possible, but has no effect because the SysTick exception request and COUNTFLAG are activated when counting from 1 to 0. Значение поля **RELOAD** может быть любым в диапазоне **0x00000001-0x00FFFFFF**. Начальное значение 0 также возможно, но оно не имеет никакого эффекта, потому что запрос исключения **SysTick** и флаг **COUNTFLAG** активизируются при переходе счета с 1 на 0.

The RELOAD value is calculated according to its use:
Значение перегрузки расчитывается в соответствии с использованием таймера

- To generate a multi-shot timer with a period of N processor clock cycles, use a RELOAD value of N-1. For example, if the SysTick interrupt is required every 100 clock pulses, set RELOAD to 99.
  Чтобы таймер генерировал многократный сигнал с периодом **N** тактовых циклов процессора, используйте значение **RELOAD = N-1**. Например, если прерывание от **SysTick** требуется каждые 100 тактовых циклов, установите **RELOAD = 99**.
- To deliver a single SysTick interrupt after a delay of N processor clock cycles, use a RELOAD of value N. For example, if a SysTick interrupt is required after 400 clock pulses, set RELOAD to 400.
  Чтобы получить единственное прерывание от **SysTick** после задержки в **N** тактовых циклов процессора, используйте значение **RELOAD = N**. Например, если прерывание от **SysTick** требуется после 400 тактовых циклов, установите **RELOAD = 400**.

4.4.3 SysTick current value register (STK_VAL)
Регистр текущего значения системного таймера

Address offset: 0x08
Reset value: 0x0000 0000
Required privilege: Privileged

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CURRENT[23:16]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURRENT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

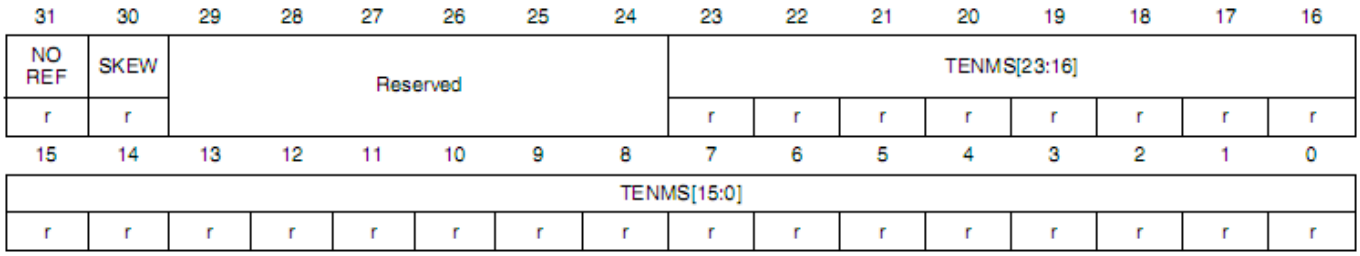
Bits 31:24 Reserved
must be kept cleared. (При записи всегда 0)

Bits 23:0 CURRENT[23:0]: Current counter value (Текущее значение таймера)
The VAL register contains the current value of the SysTick counter. Reads return the current value of the SysTick counter. A write of any value clears the field to 0, and also clears the COUNTFLAG bit in the STK_CTRL register to 0.
Регистр **VAL** содержит текущее значение счетчика **SysTick**. Чтение регистра возвращает это текущее значение. Запись любого значения очищает поле в '0', а также очищает бит **COUNTFLAG** в регистре **STK_CTRL**.

4.4.4 SysTick calibration value register (STK_CALIB)
Регистр значения калибровки системного таймера

Address offset: 0x0C
Reset value: 0x0002328
Required privilege: Privileged

The CALIB register indicates the SysTick calibration properties.
Регистр **CALIB** показывает свойства калибровки **SysTick**.



Bit 31 NOREF: NOREF flag

Reads as zero. Indicates that a separate reference clock is provided. The frequency of this clock is HCLK/8.
Читается как ноль. Показывает, что предоставлен отдельный тактовый сигнал. Частота этого сигнала **HCLK/8**.

Bit 30 SKEW: SKEW flag

Reads as one. Calibration value for the 1 ms inexact timing is not known because TENMS is not known. This can affect the suitability of SysTick as a software real time clock.
Читается как единица. Значение калибровки для условно точного периода в 1 ms не известно, так как не известно значение **TENMS**. Это может повлиять на возможность использовать **SysTick** в качестве программных часов реального времени.

Bits 29:24 Reserved

must be kept cleared. (При записи всегда 0)

Bits 23:0 TENMS[23:0]: Calibration value (Значение калибровки)

Reads as 9000. Indicates the calibration value when the SysTick counter runs on HCLK max/8 as external clock. For 72 MHz HCLK clock, the SysTick period is 1ms. If calibration information is not known, calculate the calibration value required from the frequency of the processor clock or external clock.
Читается как 9000. Указывает значение калибровки, когда таймер **SysTick** работает от внешнего тактового сигнала, в качестве которого используется 1/8 сигнала **HCLK**. Для тактового сигнала **HCLK** в 72 МГц это соответствует периоду **SysTick** в 1ms. Если информация о калибровке не известна, расчитайте требуемое значение калибровки по частоте тактового сигнала процессора или внешнего сигнала.

4.4.5 SysTick design hints and tips
Программирование системного таймера: рекомендации и советы

The SysTick counter runs on the processor clock. If this clock signal is stopped for low power mode, the SysTick counter stops.
Таймер **SysTick** работает от процессора. Если этот сигнал остановлен в режиме потребления малой мощности, то останавливается и счетчик **SysTick**.

Ensure software uses aligned word accesses to access the SysTick registers.
Убедитесь, что программа использует выровненный пословный доступ при обращении к регистрам **SysTick**.

4.4.6 SysTick register map
Карта регистров системного таймера

The table provides shows the SysTick register map and reset values. The base address of the SysTick register block is 0xE000 E010.
Таблица показывает карту регистров **SysTick** и их значений после сброса. Базовый адрес блока регистров **SysTick - 0xE000 E010**.

Table 41. SysTick register map and reset values
Карта регистров **SysTick** и их значений после сброса

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	STK_CTRL	Reserved															COUNTFLAG	Reserved													CLKSOURCE	TICK INT	EN ABLE
	Reset Value																0														1	0	0
0x04	STK_LOAD	Reserved								RELOAD[23:0]																							
	Reset Value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	STK_VAL	Reserved								CURRENT[23:0]																							
	Reset Value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	STK_CALIB	Reserved								TENMS[23:0]																							
	Reset Value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5 Revision history (История изменений)

Table 42. Document revision history (История изменений документа)

Date	Revision	Changes
03-Apr-2009	1	Initial release. (Первичный документ)
26-Oct-2009	2	Table 38: Priority grouping modified. Изменение в таблице 38 "Группировка приоритетов"

STM32F105_107_Reference manual (Справочное руководство)

1 Documentation conventions (Соглашения по документации)
2 Memory and bus architecture (Архитектура шин и памяти)
3 CRC calculation unit (Модуль расчета CRC)
4 Power control (PWR) (Управление питанием)
5 Backup registers (BKP) (Резервные регистры)
6 Low-, medium- and high-density reset and clock control (RCC) (Пока нет перевода)
7 Connectivity line devices: reset and clock control (RCC)
Семейство **CL**: управление сбросом и тактовыми
8 General-purpose and alternate-function I/Os (GPIOs and AFIOs)
Основные и альтернативные функции **I/O**
9 Interrupts and events (Прерывания и события)
10 DMA controller (DMA) (Контроллер прямого доступа)
11 Analog-to-digital converter (ADC) (Пока нет перевода)
12 Digital-to-analog converter (DAC) (Пока нет перевода)
13 Advanced-control timers (TIM1&TIM8) (Пока нет перевода)
14 General-purpose timer (TIMx) (Пока нет перевода)
15 Basic timers (TIM6&TIM7) (Пока нет перевода)
16 Real-time clock (RTC) (Пока нет перевода)
17 Independent watchdog (IWDG) (Пока нет перевода)
18 Window watchdog (WWDG) (Пока нет перевода)
19 Flexible static memory controller (FSMC) (Пока нет перевода)
20 Secure digital input/output interface (SDIO) (Пока нет перевода)
21 Universal serial bus full-speed device interface (USB) (Пока нет перевода)
22 Controller area network (bxCAN) (Пока нет перевода)
23 Serial peripheral interface (SPI) (Пока нет перевода)
24 Inter-integrated circuit (I2C) interface (Интерфейс **I2C**)
25 Universal synchronous asynchronous receiver transmitter (USART)
Универсальный синхронный/асинхронный приемо-передатчик **USART**
26 USB on-the-go full-speed (OTG_FS) (Пока нет перевода)
27 Ethernet (ETH): media access control (MAC) with DMA controller
(Пока нет перевода)
28 Device electronic signature (Пока нет перевода)
29 Debug support (DBG) (Пока нет перевода)
30 Revision history (Пока нет перевода)

1 Documentation conventions (Соглашения по документации)

1.1 List of abbreviations for registers (Перечень аббревиатур для регистров)
1.2 Glossary (Глоссарий)
1.3 Peripheral availability (Возможности периферии)

1.1 List of abbreviations for registers (Перечень аббревиатур для регистров)

The following abbreviations are used in register descriptions:
При описании регистров используются следующие аббревиатуры:

read/write (rw)	Software can read and write to these bits. Программа может читать и записывать эти биты.
read-only (r)	Software can only read these bits. Программа может только читать эти биты.
write-only (w)	Software can only write to this bit. Reading the bit returns the reset value. Программа может только записывать в эти биты. При чтении бита возвращается значение после сброса.
read/clear (rc_w1)	Software can read as well as clear this bit by writing 1. Writing ‘0’ has no effect on the bit value. Программа может читать эти биты и очищать их записью '1'. Запись '0' не оказывает влияния на значение бита.
read/clear (rc_w0)	Software can read as well as clear this bit by writing 0. Writing ‘1’ has no effect on the bit value. Программа может читать эти биты и очищать их записью '0'. Запись '1' не оказывает влияния на значение бита.
read/clear by read (rc_r)	Software can read this bit. Reading this bit automatically clears it to ‘0’. Writing ‘0’ has no effect on the bit value. При чтении этих битов они автоматически очищаются в '0'. Запись '0' не оказывает влияния на значение бита.
read/set (rs)	Software can read as well as set this bit. Writing ‘0’ has no effect on the bit value. Программа может читать эти биты и устанавливать их. Запись '0' не оказывает влияния на значение бита.
read-only write trigger (rt_w)	Software can read this bit. Writing ‘0’ or ‘1’ triggers an event but has no effect on the bit value. Программа может читать этот бит. Запись '0' или '1' переключает событие но не оказывает влияния на значение бита.
toggle (t)	Software can only toggle this bit by writing ‘1’. Writing ‘0’ has no effect. Программа может только переключать значение этого бита записью '1'. Запись '0' не имеет эффекта.
Reserved (Res.)	Reserved bit, must be kept at reset value. Зарезервированный бит, его значение должно удерживаться таким, как оно было после сброса.

1.2 Glossary (Глоссарий)

Low-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.
Устройства семейства **Low-density** (низкой плотности, *далее LD*) - это микроконтроллеры **STM32F101xx**, **STM32F102xx** и **STM32F103xx**, у которых объем **Flash** памяти находится в диапазоне 16-32 КБайта.

Medium-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.
Устройства семейства **Medium-density** (средней плотности, *далее MD*) - это микроконтроллеры **STM32F101xx**, **STM32F102xx** и **STM32F103xx**, у которых объем **Flash** памяти находится в диапазоне 64-128 КБайт.

High-density devices are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.
Устройства семейства **High-density** (высокой плотности, *далее HD*) - это микроконтроллеры **STM32F101xx** и **STM32F103xx**, у которых объем **Flash** памяти находится в диапазоне 256-512 КБайт.

Connectivity line devices are STM32F105xx and STM32F107xx microcontrollers.
Устройства семейства **Connectivity line** (линия для коммутации, *далее CL*) - это микроконтроллеры **STM32F105xx** и **STM32F107xx**

This section applies to the whole STM32F10xxx family, unless otherwise specified.
Разделы этого документа применимы ко всем семействам **STM32F10xxx**, если не указано иное.

1.3 Peripheral availability (Возможности периферии)

For peripheral availability and number across all STM32F10xxx sales types, please refer to the low-, medium- and high-density STM32F101xx and STM32F103xx datasheets, to the low- and medium-density STM32F102xx datasheets and to the connectivity line devices, STM32F105xx/STM32F107xx.
Для информации о ниличии и числе периферии по всем коммерческим типам **STM32F10xxx**, пожалуйста обратитесь к datasheet, соответствующему вашему изделию.

2 Memory and bus architecture (Архитектура шин и памяти)

2.1 System architecture (Архитектура системы)
2.2 Memory organization (Организация памяти)
2.3 Memory map (Карта памяти)
2.4 Boot configuration (Конфигурация начальной загрузки)

2.1 System architecture (Архитектура системы)

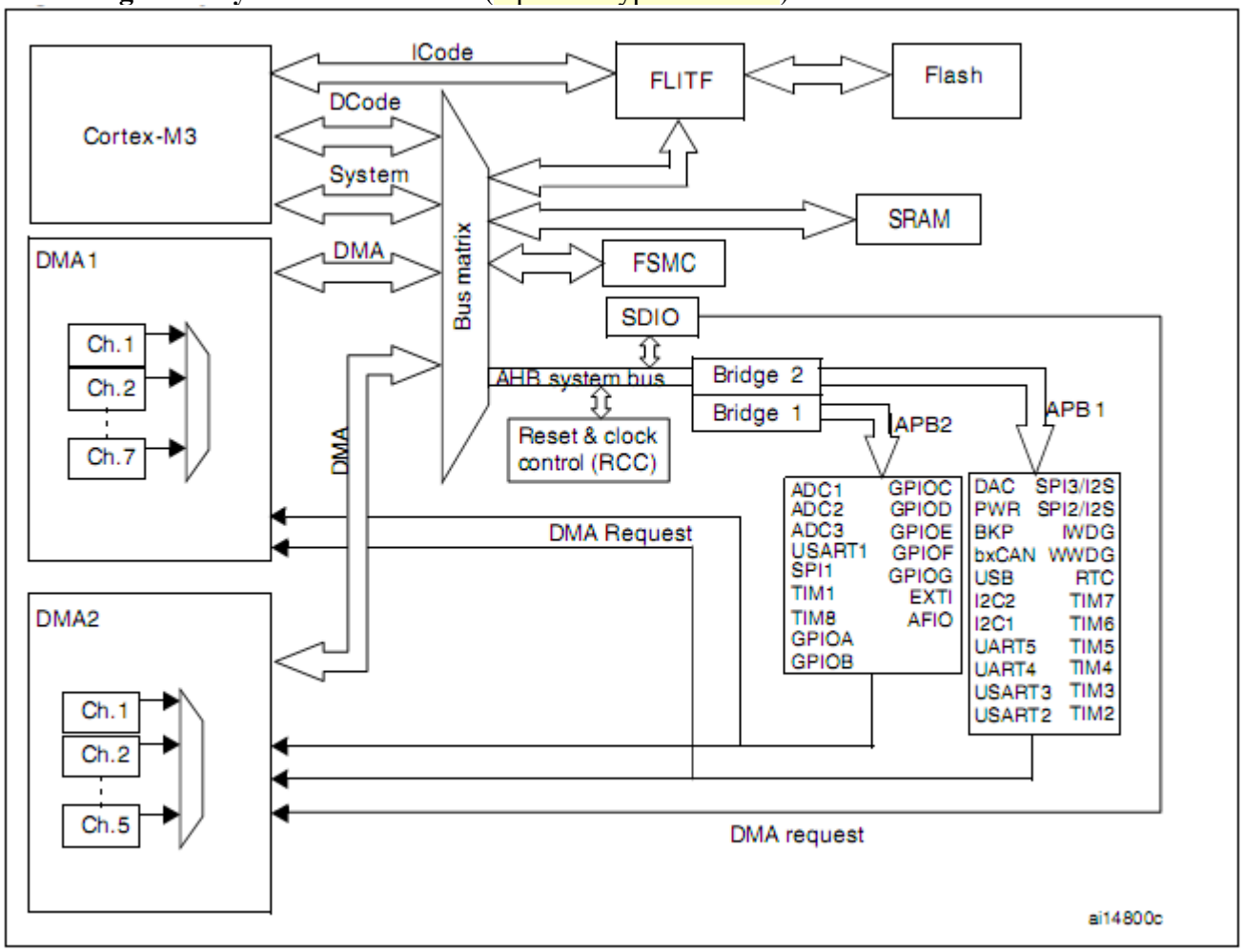
In low-, medium- and high-density devices, the main system consists of:
Для семейств **LD**, **MD** и **HD** основная система содержит следующее:

- Four masters (4 шинных мастера (ведущих контроллеров)):
  - Cortex™-M3 core DCode bus (D-bus) and System bus (S-bus)
    Шина данных (**D-bus**) и системная шина (**S-bus**) ядра **Cortex-M3**
  - GP-DMA1 & 2 (general-purpose DMA) (две **DMA** шины общего назначения)

- Four slaves (4 ведомых шинных контроллеров):
  - Internal (Внутренняя) SRAM
  - Internal Flash memory (Внутренняя **Flash** память)
  - FSMC
  - AHB to APB bridges (AHB2APBx), which connect all the APB peripherals
    Мосты с **AHB** на **APB** (**AHB2APBx**), которые объединяют все внешние устройства шины **APB**

These are interconnected using a multilayer AHB bus architecture as shown in Figure 1:
Эти модули соединены между собой с помощью **AHB** шины с многослойной архитектурой, как показано на рис. 1:

Figure 1. System architecture (Архитектура системы)

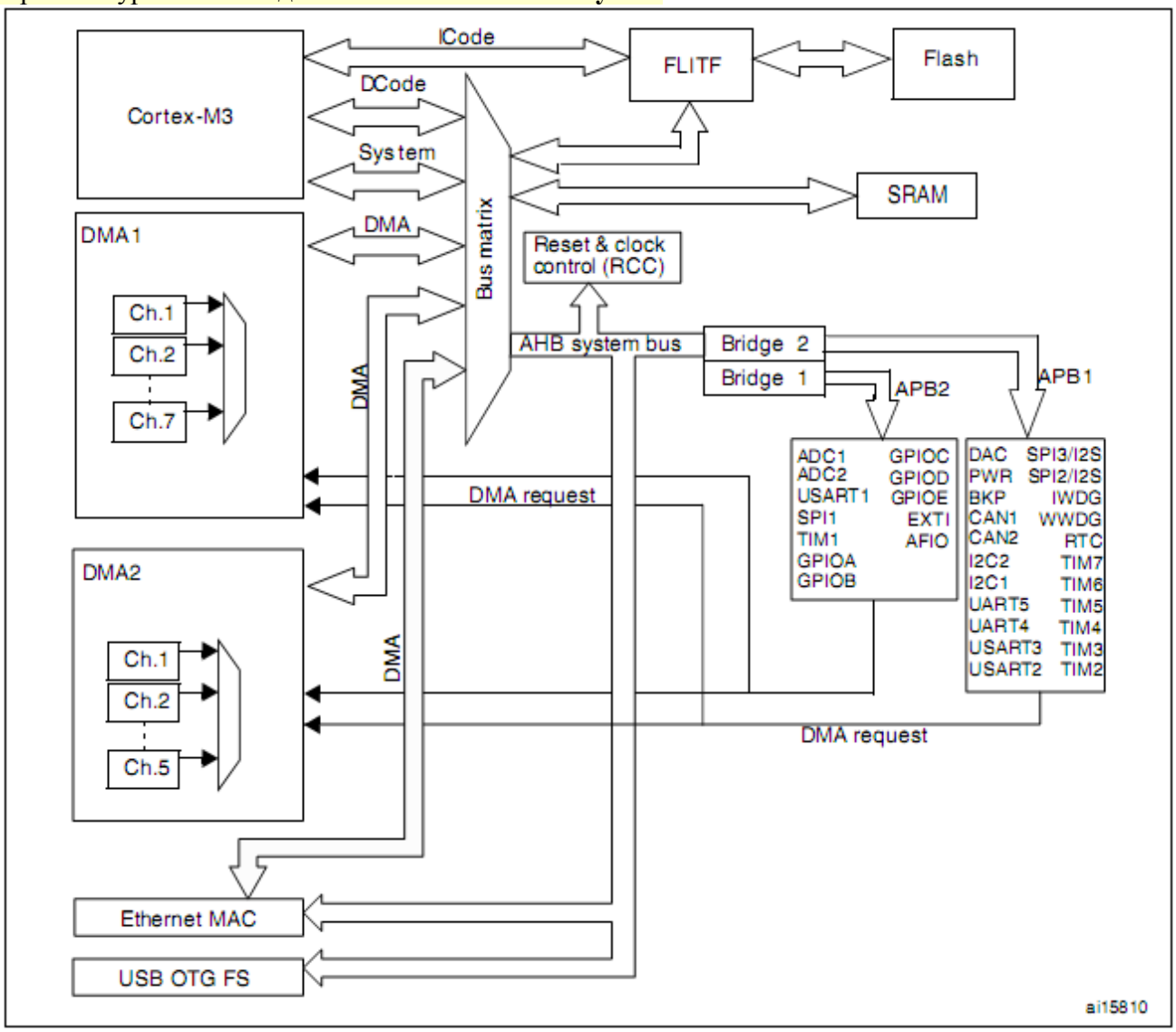


In connectivity line devices the main system consists of:
Для семейства **CL** основная система содержит следующее:

- Five masters (5 шинных мастеров (ведущих контроллеров)):
  - Cortex™-M3 core DCode bus (D-bus) and System bus (S-bus)
    Шина данных (**D-bus**) и системная шина (**S-bus**) ядра **Cortex-M3**
  - GP-DMA1 & 2 (general-purpose DMA) (две **DMA** шины общего назначения)
  - Ethernet DMA
- Three slaves (3 ведомых шинных контроллеров):
  - Internal (Внутренняя) SRAM
  - Internal Flash memory (Внутренняя **Flash** память)
  - AHB to APB bridges (AHB2APBx), which connect all the APB peripherals
    Мосты с **AHB** на **APB** (**AHB2APBx**), которые объединяют все устройства шины **APB**

These are interconnected using a multilayer AHB bus architecture as shown in Figure 2:
Эти модули соединены между собой с помощью **AHB** шины с многослойной архитектурой, как показано на рис. 2:

Figure 2. System architecture in connectivity line devices
Архитектура системы для семейства **connectivity line**



ICode bus (Шина инструкций)

This bus connects the Instruction bus of the Cortex™-M3 core to the Flash memory instruction interface. Prefetching is performed on this bus.
Эта шина подключает шину инструкций ядра **Cortex-M3** к интерфейсу **Flash** памяти инструкций. На этой шине выполняется предварительная выборка.

DCode bus (Шина данных)

This bus connects the DCode bus (literal load and debug access) of the Cortex™-M3 core to the Flash memory Data interface.
Эта шина подключает шину данных ядра **Cortex-M3** (есть доступ для отладчика и посимвольного загрузчика) к интерфейсу **Flash** памяти данных.

System bus (Системная шина)

This bus connects the system bus of the Cortex™-M3 core (peripherals bus) to a BusMatrix which manages the arbitration between the core and the DMA.
Эта шина подключает системную шину ядра **Cortex-M3** (шина периферии) к матрице шин, которая является арбитром между ядром и **DMA**.

DMA bus (Шина прямого доступа)

This bus connects the AHB master interface of the DMA to the BusMatrix which manages the access of CPU DCode and DMA to SRAM, Flash memory and peripherals.
Эта шина подключает мастер интерфейса **DMA** с **AHB** к матрице шин, которая управляет доступом ЦПУ и **DMA** к памяти **SRAM**, **Flash** памяти и памяти периферии.

BusMatrix (Матрица шин)

The BusMatrix manages the access arbitration between the core system bus and the DMA master bus. The arbitration uses a Round Robin algorithm. In connectivity line devices, the BusMatrix is composed of five masters (CPU DCode, System bus, Ethernet DMA, DMA1 and DMA2 bus) and three slaves (FLITF, SRAM and AHB2APB bridges). In other devices, the BusMatrix is composed of four masters (CPU DCode, System bus, DMA1 bus and DMA2 bus) and four slaves (FLITF, SRAM, FSMC and AHB2APB bridges).
Матрица шин "разруливает" конфликты доступа между системной шиной ядра и мастером **DMA** шины. Для арбитража используется алгоритм **Round Robin**. Для устройств семейства **CL**, матрица шин объединяет в себе пять мастеров шин (**CPU DCode**, **System bus**, **Ethernet DMA**, **DMA1** и **DMA2**) и три ведомых контроллера шин (**FLITF**, **SRAM** и **AHB2APB** мосты). Для других устройств матрица шин объединяет в себе четыре мастера шин (**CPU DCode**, **System bus**, **DMA1** и **DMA2**) и четыре ведомых контроллера шин (**FLITF**, **SRAM**, **FSMC** и **AHB2APB** мосты).

AHB peripherals are connected on system bus through a BusMatrix to allow DMA access.
Вся **AHB** периферия подключена к системной шине через матрицу шин, что позволяет иметь к ней прямой доступ (**DMA**).

AHB/APB bridges (APB) (AHB/APB мосты)

The two AHB/APB bridges provide full synchronous connections between the AHB and the 2 APB buses. APB1 is limited to 36 MHz, APB2 operates at full speed (up to 72 MHz depending on the device).
Два **AHB/APB** моста обеспечивают полностью синхронизированное соединение между шиной **AHB** и двумя шинами **APB**.

Refer to Table 1 on page 41 for the address mapping of the peripherals connected to each bridge.
Смотрите табл. 1, где приведена карта адресов периферии, подключенной к каждому мосту.

After each device reset, all peripheral clocks are disabled (except for the SRAM and FLITF). Before using a peripheral you have to enable its clock in the RCC_AHBENR, RCC_APB2ENR or RCC_APB1ENR register.
После каждого сброса устройства, все тактовые сигналы для периферии отключаются (кроме сигнала для **SRAM** и **FLITF**). Перед тем, как использовать периферию, вы должны разрешить ее собственный тактовый сигнал в регистре **RCC_AHBENR**, **RCC_APB2ENR** или **RCC_APB1ENR**.

***Note**: When a 16- or 8-bit access is performed on an APB register, the access is transformed into a 32-bit access: the bridge duplicates the 16- or 8-bit data to feed the 32-bit vector.*
***Note**: Когда выполняется 16-ти или 8-ми битовый доступ к регистру **APB**, такой доступ трансформируется в 32-х битный: мост дублирует 16-ти или 8-ми битовые данные, чтобы обеспечить 32-разрядный вектор.*

2.2 Memory organization (Организация памяти)

Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space. Память программы, память данных, пространство регистров и **I/O** портов организованы в единое 4-х ГигаБайтное адресное пространство.

The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word's least significant byte and the highest numbered byte the most significant.
Байты закодированы в памяти в формате **Little Endian**. Байт слова с самый низким номером считают самым младшим байтом этого слова, а байт с самым высоким номером считают самым старшим.

For the detailed mapping of peripheral registers, please refer to the related chapters.
Для дополнительной информации об адресах регистров периферии обратитесть к соответствующим главам этого руководства.

The addressable memory space is divided into 8 main blocks, each of 512 MB.
Адресуемое пространство памяти разделено на 8 главных блоков, каждый по 512 Мбайт.

All the memory areas that are not allocated to on-chip memories and peripherals are considered "Reserved". Refer to the Memory map figure in the corresponding product datasheet.
Все области памяти, которые не распределены на внутреннюю память чипа или периферию, считаются "Зарезервированными". Смотри карту памяти в **datasheet** на соответствующей продукт.

2.3 Memory map (Карта памяти)

2.3.1 Embedded SRAM (Встроенная **SRAM**)
2.3.2 Bit banding (Работа с отдельными битами)
2.3.3 Embedded Flash memory (Встроенная **Flash** память)

See the datasheet corresponding to your device for a comprehensive diagram of the memory map. Table 1 gives the boundary addresses of the peripherals available in all STM32F10xxx devices.
Смотри **datasheet**, соответствующий вашему устройству, где приведена всеобъемлющая карта памяти. Таблица 1 дает граничные адреса периферии, доступной на всех устройствах серии **STM32F10xxx**.

Table 1. Register boundary addresses (Граничные адреса регистров)

Boundary address	Peripheral	Bus	Register map
0x5000 0000 - 0x5000 03FF	USB OTG FS	AHB	Section 26.14.6 on page 778
0x4003 0000 - 0x4FFF FFFF	Reserved		
0x4002 8000 - 0x4002 9FFF	Ethernet		Section 27.8.5 on page 946
0x4002 3400 - 0x4002 7FFF	Reserved	AHB	
0x4002 3000 - 0x4002 33FF	CRC		Section 3.4.4 on page 52
0x4002 2000 - 0x4002 23FF	Flash memory interface		
0x4002 1400 - 0x4002 1FFF	Reserved		
0x4002 1000 - 0x4002 13FF	Reset and clock control RCC		Section 6.3.11 on page 102
0x4002 0800 - 0x4002 0FFF	Reserved		
0x4002 0400 - 0x4002 07FF	DMA2		Section 10.4.7 on page 196

0x4002 0000 - 0x4002 03FF	DMA1		Section 10.4.7 on page 196
0x4001 8400 - 0x4001 7FFF	Reserved		
0x4001 8000 - 0x4001 83FF	SDIO		Section 20.9.16 on page 510
0x4001 4000 - 0x4001 7FFF	Reserved		
0x4001 3C00 - 0x4001 3FFF	ADC3		Section 11.12.15 on page 231
0x4001 3800 - 0x4001 3BFF	USART1		Section 25.6.8 on page 693
0x4001 3400 - 0x4001 37FF	TIM8 timer		Section 13.4.21 on page 317
0x4001 3000 - 0x4001 33FF	SPI1		Section 23.5 on page 614
0x4001 2C00 - 0x4001 2FFF	TIM1 timer		Section 13.4.21 on page 317
0x4001 2800 - 0x4001 2BFF	ADC2		Section 11.12.15 on page 231
0x4001 2400 - 0x4001 27FF	ADC1		Section 11.12.15 on page 231
0x4001 2000 - 0x4001 23FF	GPIO Port G	APB2	Section 8.5 on page 167
0x4001 1C00 - 0x4001 1FFF	GPIO Port F		Section 8.5 on page 167
0x4001 1800 - 0x4001 1BFF	GPIO Port E		Section 8.5 on page 167
0x4001 1400 - 0x4001 17FF	GPIO Port D		Section 8.5 on page 167
0x4001 1000 - 0x4001 13FF	GPIO Port C		Section 8.5 on page 167
0x4001 0C00 - 0x4001 0FFF	GPIO Port B		Section 8.5 on page 167
0x4001 0800 - 0x4001 0BFF	GPIO Port A		Section 8.5 on page 167
0x4001 0400 - 0x4001 07FF	EXTI		Section 9.3.7 on page 181
0x4001 0000 - 0x4001 03FF	AFIO		Section 8.5 on page 167
0x4000 7800 - 0x4000 FFFF	Reserved	APB1	
0x4000 7400 - 0x4000 77FF	DAC		Section 12.5.14 on page 252
0x4000 7000 - 0x4000 73FF	Power control PWR		Section 4.4.3 on page 65
0x4000 6C00 - 0x4000 6FFF	Backup registers (BKP)		Section 5.4.5 on page 70
0x4000 6800 - 0x4000 6BFF	Reserved		
0x4000 6400 - 0x4000 67FF	bxCAN1		Section 22.9.5 on page 583
0x4000 6400 - 0x4000 67FF	bxCAN2		Section 22.9.5 on page 583
0x4000 6000[(1)] - 0x4000 63FF	Shared USB/CAN SRAM 512 bytes		
0x4000 5C00 - 0x4000 5FFF	USB device FS registers		Section 21.5.4 on page 540
0x4000 5800 - 0x4000 5BFF	I2C2		Section 24.6.10 on page 652
0x4000 5400 - 0x4000 57FF	I2C1		Section 24.6.10 on page 652
0x4000 5000 - 0x4000 53FF	UART5		Section 25.6.8 on page 693
0x4000 4C00 - 0x4000 4FFF	UART4		Section 25.6.8 on page 693
0x4000 4800 - 0x4000 4BFF	USART3		Section 25.6.8 on page 693
0x4000 4400 - 0x4000 47FF	USART2		Section 25.6.8 on page 693
0x4000 4000 - 0x4000 3FFF	Reserved		
0x4000 3C00 - 0x4000 3FFF	SPI3/I2S		Section 23.5 on page 614
0x4000 3800 - 0x4000 3BFF	SPI2/I2S		Section 23.5 on page 614
0x4000 3400 - 0x4000 37FF	Reserved		
0x4000 3000 - 0x4000 33FF	Independent watchdog (IWDG)		Section 17.4.5 on page 403

0x4000 2C00 - 0x4000 2FFF	Window watchdog (WWDG)		Section 18.6.4 on page 408
0x4000 2800 - 0x4000 2BFF	RTC		Section 16.4.7 on page 398
0x4000 1800 - 0x4000 27FF	Reserved		
0x4000 1400 - 0x4000 17FF	TIM7 timer		Section 15.4.9 on page 386
0x4000 1000 - 0x4000 17FF	TIM6 timer		Section 15.4.9 on page 386
0x4000 0C00 - 0x4000 0FFF	TIM5 timer		Section 14.4.19 on page 373
0x4000 0800 - 0x4000 0BFF	TIM4 timer		Section 14.4.19 on page 373
0x4000 0400 - 0x4000 07FF	TIM3 timer		Section 14.4.19 on page 373
0x4000 0000 - 0x4000 03FF	TIM2 timer		Section 14.4.19 on page 373

1. This shared SRAM can be fully accessed only in low-, medium- and high-density devices, not in connectivity line devices. (К этой **SRAM** совместного использования можно получить полный доступ только в семействах **LD**, **MD** и **HD**, но не в устройствах семейства **CL)**.

2.3.1 Embedded SRAM (Встроенная SRAM)

The STM32F10xxx features 64 Kbytes of static SRAM. It can be accessed as bytes, half-words (16 bits) or full words (32 bits). The SRAM start address is 0x2000 0000.
Серия **STM32F10xxx** подоставляет 64 КБайта статической **SRAM** памяти. К ней можно обращаться как к байту, полуслову (16 битов) или полному слову (32 бита). Адрес начала **SRAM - 0x2000 0000**.

2.3.2 Bit banding (Работа с отдельными битами)

The Cortex™-M3 memory map includes two bit-band regions. These regions map each word in an alias region of memory to a bit in a bit-band region of memory. Writing to a word in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.
Карта памяти ядра **Cortex-M3** включает две области типа **bit-band**. Эти области (*состоят из двух частей и*) отображают каждое слово из области доступа к битам (*alias region*) на один бит в области хранения бит (*bit-band region*). Запись слова в области доступа к битам имеет тот же самый эффект, что и операция "чтение - модификация - запись" над целевым битом в области хранения бит.

In the STM32F10xxx both peripheral registers and SRAM are mapped in a bit-band region. This allows single bit-band write and read operations to be performed.
В серии **STM32F10xxx** как регистры периферии, так и **SRAM** отображены на области типа **bit-band**. Это позволяет выполнять операции одиночной записи и чтения в этой области.

A mapping formula shows how to reference each word in the alias region to a corresponding bit in the bit-band region. The mapping formula is:
Формула отображения показывает, как сослаться с помощью слова в области доступа к битам на соответствующий бит в области хранения бит. Формула отображения следующая:

```
bit_word_addr = bit_band_base + (byte_offset x 32) + (bit_number × 4)
```

where (Где):
bit_word_addr is the address of the word in the alias memory region that maps to the targeted bit.
это адрес слова в области доступа к битам, которое отражается на целевой бит.

bit_band_base is the starting address of the alias region
это начальный адрес области доступа к битам

`byte_offset` is the number of the byte in the bit-band region that contains the targeted bit
это номер того байта в области хранения бит, который содержит целевой бит

`bit_number` is the bit position (0-7) of the targeted bit.
это позиция целевого бита (0-7) в байте.

Example (Пример):

The following example shows how to map bit 2 of the byte located at SRAM address 0x20000300 in the alias region:
Следующий пример показывает, как отобразить бит 2 из байта, расположенного в **SRAM** по адресу **0x20000300**, на область доступа к битам:
`0x22006008 = 0x22000000 + (0x300*32) + (2*4)`.

Writing to address 0x22006008 has the same effect as a read-modify-write operation on bit 2 of the byte at SRAM address 0x20000300.
Запись по адресу **0x22006008** имеет тот же самый эффект, что и операция "чтение-изменение-запись" для бита 2 из байта в **SRAM** по адресу **0x20000300**.

Reading address 0x22006008 returns the value (0x01 or 0x00) of bit 2 of the byte at SRAM address 0x20000300 (0x01: bit set; 0x00: bit reset).
Чтение по адресу **0x22006008** вернет значение бита 2 (**0x01** или **0x00**) для байта в **SRAM** по адресу **0x20000300** (**0x01**: бит установлен; **0x00**: бит сброшен).

For more information on Bit-Banding, please refer to the Cortex™-M3 Technical Reference Manual. Обратитесь к документу "***Cortex-M3 Technical Reference Manual***" за более детальной информацией о работе с битами.

2.3.3 Embedded Flash memory (Встроенная Flash память)

The high-performance Flash memory module has the following key features:
Модуль высокоэффективной **Flash**-памяти имеет следующие ключевые особенности:

- Density of up to 512 Kbytes (Объем памяти до 512 КБайт)
- Memory organization: the Flash memory is organized as a main block and an information block: Организация памяти: **Flash**-память организована как главный блок плюс блок информации:
    - Main memory block of size: (Главный блок памяти имеет размер:)
      up to 4 Kb x 64 bits divided into 32 pages of 1 Kbyte each
      до 4 Kb x 64 бита, разделенных на 32 страницы по 1 КБ каждая
      for low-density devices (see Table 2) (для устройств семейства **LD** (см. табл. 2))
      up to 16 Kb x 64 bits divided into 128 pages of 1 Kbyte each
      до 16 Kb x 64 бита, разделенных на 128 страниц по 1 КБ каждая
      for medium-density devices (see Table 3) (для устройств семейства **MD** (см. табл. 3))
      up to 64 Kb × 64 bits divided into 256 pages of 2 Kbytes each
      до 64 Kb x 64 бита, разделенных на 256 страниц по 2 КБ каждая
      (see Table 4) for high-density devices (для устройств семейства **HD** (см. табл. 4))
      up to 32 Kbit × 64 bits divided into 128 pages of 2 Kbytes each
      до 32 Kb x 64 бита, разделенных на 128 страниц по 2 КБ каждая
      (see Table 5) for connectivity line devices (для устройств семейства **CL** (см. табл. 5))
    - Information block of size: (Информационный блок памяти имеет размер:)

2360 x 64 bits for connectivity line devices (see Table 5)
2360 x 64 бита для устройств семейства **CL** (см. табл. 5)
258 x 64 bits for other devices (see Table 2, Table 3 and Table 4)
258 x 64 бита для устройств других семейств (см. таблицы 2, 3, 4)

The Flash memory interface (FLITF) features:
Интерфейс с **Flash**-памятью (**FLITF**) имеет следующие особенности:

- Read interface with prefetch buffer (2x64-bit words)
Интерфейс чтения с буфером предварительной выборки (2 слова по 64 бита)
- Option byte Loader (Опционный байтовый загрузчик)
- Flash Program/Erase operation (Операции Программирования/Стирания **Flash**-памяти)
- Read/Write protection (Защита от Чтения/Записи)

Table 2. Flash module organization (low-density devices)
Организация **Flash**-модуля (устройства семейства **LD**)

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 03FF	1 Kbyte
	Page 1	0x0800 0400 - 0x0800 07FF	1 Kbyte
	Page 2	0x0800 0800 - 0x0800 0BFF	1 Kbyte
	Page 3	0x0800 0C00 - 0x0800 0FFF	1 Kbyte
	Page 4	0x0800 1000 - 0x0800 13FF	1 Kbyte
	...	...	...
	Page 31	0x0800 7C00 - 0x0800 7FFF	1 Kbyte
Information block	System memory	0x1FFF F000 - 0x1FFF F7FF	2 Kbytes
	Option Bytes	0x1FFF F800 - 0x1FFF F80F	16
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Table 3. Flash module organization (medium-density devices)
Организация **Flash**-модуля (устройства семейства **MD**)

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 03FF	1 Kbyte
	Page 1	0x0800 0400 - 0x0800 07FF	1 Kbyte
	Page 2	0x0800 0800 - 0x0800 0BFF	1 Kbyte
	Page 3	0x0800 0C00 - 0x0800 0FFF	1 Kbyte
	Page 4	0x0800 1000 - 0x0800 13FF	1 Kbyte
	...	...	...
	Page 127	0x0801 FC00 - 0x0801 FFFF	1 Kbyte
Information block	System memory	0x1FFF F000 - 0x1FFF F7FF	2 Kbytes
	Option Bytes	0x1FFF F800 - 0x1FFF F80F	16
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Table 4. Flash module organization (high-density devices)
Организация **Flash**-модуля (устройства семейства **HD**)

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 07FF	1 Kbyte
	Page 1	0x0800 0800 - 0x0800 0FFF	2 Kbyte
	Page 2	0x0800 1000 - 0x0800 17FF	2 Kbyte
	Page 3	0x0800 1800 - 0x0800 1FFF	2 Kbyte
	...	...	...
	Page 255	0x0807 F800 - 0x0807 FFFF	2 Kbyte
Information block	System memory	0x1FFF F000 - 0x1FFF F7FF	2 Kbytes
	Option Bytes	0x1FFF F800 - 0x1FFF F80F	16
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4

Block	Name	Base addresses	Size (bytes)
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Table 5. Flash module organization (connectivity line devices)
Организация **Flash**-модуля (устройства семейства **CL**)

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 07FF	1 Kbyte
	Page 1	0x0800 0800 - 0x0800 0FFF	2 Kbyte
	Page 2	0x0800 1000 - 0x0800 17FF	2 Kbyte
	Page 3	0x0800 1800 - 0x0800 1FFF	2 Kbyte
	...	...	...
	Page 127	0x0803 F800 - 0x0803 FFFF	2 Kbyte
Information block	System memory	0x1FFF B000 - 0x1FFF F7FF	18 Kbytes
	Option Bytes	0x1FFF F800 - 0x1FFF F80F	16
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

***Note**: For further information on the Flash memory interface registers, please refer to the STM32F10xxx Flash programming manual.*
***Note**: Для дополнительной информацией об интерфейсе **Flash**-памяти обратитесь к документу "**STM32F10xxx Flash programming manual**".*

Reading the Flash memory (Чтение Flash-памяти)

Flash memory instructions and data access are performed through the AHB bus. The prefetch block is used for instruction fetches through the ICode bus. Arbitration is performed in the Flash memory interface, and priority is given to data access on the DCode bus.
Доступ к инструкциям и данным во **Flash**-памяти выполняется через шину **AHB**. Блок предварительной выборки используется при чтении инструкций через шину **ICode**. Арбитраж выполняется в интерфейсе **Flash**-памяти, и приоритет отдается доступу к данным на шине **DCode**.

Read accesses can be performed with the following configuration options:
Доступ для чтения может быть выполнен со следующими опциями конфигурации:

- Latency: number of wait states for a read operation programmed on-the-fly

Время ожидания: число состояний ожидания для операции чтения программируется "на лету"

- Prefetch buffer (2 x 64-bit blocks): it is enabled after reset; a whole block can be replaced with a single read from the Flash memory as the size of the block matches the bandwidth of the Flash memory. Thanks to the prefetch buffer, faster CPU execution is possible as the CPU fetches one word at a time with the next word readily available in the prefetch buffer

Буфер предварительной выборки (2 x 64 битовых блока): разрешен после сброса; целый блок команд чтения может быть заменен единственной командой чтения во Флэш-памяти, если размер блока соответствует пропускной способности Флэш-памяти. Благодаря буферу предварительной выборки возможна более быстрая работа ЦПУ, так как процессор выбирает одно слово одновременно со следующим словом, которое доступно для чтения в буфере предварительной выборки

- Half cycle: for power optimization

Половина цикла: для оптимизации потребляемой мощности

***Note**:*

1. *These options should be used in accordance with the Flash memory access time. The wait states represent the ratio of the SYSCLK (system clock) period to the Flash memory access time:*
   *Эти опции должны использоваться в соответствии с временем доступа к флэш-памяти. Состояния ожидания представляют собой отношение периода **SYSCLK** (системный тактовый сигнал) к времени доступа к флэш-памяти:*
   zero wait state, if 0 < SYSCLK ≤ 24 MHz
   one wait state, if 24 MHz < SYSCLK ≤ 48 MHz
   two wait states, if 48 MHz < SYSCLK ≤ 72 MHz

2. *Half cycle configuration is not available in combination with a prescaler on the AHB. The system clock (SYSCLK) should be equal to the HCLK clock. This feature can therefore be used only with a low-frequency clock of 8 MHz or less. It can be generated from the HSI or the HSE but not from the PLL.*
   *Конфигурирование полу-цикла не доступно в комбинации с делителем частоты шины **AHB**. Системный тактовый сигнал (**SYSCLK**) должны быть равен тактовому сигналу **HCLK**. Поэтому эту особенность можно использовать только с низкочастотными тактовым сигналом равным 8 МГц, или меньше. Такой сигнал можно сгенерировать с помощью **HSI** или **HSE**, но не **PLL**.*

3. *The prefetch buffer must be kept on when using a prescaler different from 1 on the AHB clock.*
   *Буфер предварительной выборки должен быть всегда включен, если для шины **AHB** используется делитель тактовой частоты, отличный от 1.*

4. *The prefetch buffer must be switched on/off only when SYSCLK is lower than 24 MHz. The prefetch buffer is usually switched on/off during the initialization routine, while the microcontroller is running on the internal 8 MHz RC (HSI) oscillator.*
   *Буфер предварительной выборки должен включаться или выключаться только при частоте **SYSCLK** ниже 24 МГц. Обычно буфер предварительной выборки переключается (вкл/выкл) во время инициализации программы, в это время микроконтроллер работает от внутреннего 8 МГц **RC** генератора (**HSI**).*

5. *Using DMA: DMA accesses Flash memory on the DCode bus and has priority over ICode instructions. The DMA provides one free cycle after each transfer. Some instructions can be performed together with DMA transfer.*
   *Использование **DMA**: **DMA** обращается к Флэш-памяти по шине **DCode** и имеет приоритет над командами шины **ICode**. **DMA** обеспечивает один пустой цикл после каждой транзакции. Некоторые инструкции можно выполненять одновременно с обменом по **DMA**.*

Programming and erasing the Flash memory
Программирование и стирание флэш-памяти

The Flash memory can be programmed 16 bits (half words) at a time.
Флэш-память можно программировать по 16 битов (полуслово) за один раз.

The Flash memory erase operation can be performed at page level or on the whole Flash area (mass-erase). The mass-erase does not affect the information blocks.
Операцию стирания Флэш-памяти можно выполнять постранично или стереть всю область памяти (массовое стирание). Массовое стирание не затрагивает информационные блоки.

To ensure that there is no over-programming, the Flash Programming and Erase Controller blocks are clocked by a fixed clock.
Чтобы гарантировать от случайного программирования, блоки контроллеров Программирования и Стирания тактируются отдельным тактовым сигналом.

The End of write operation (programming or erasing) can trigger an interrupt. This interrupt can be used to exit from WFI mode, only if the FLITF clock is enabled. Otherwise, the interrupt is served only after an exit from WFI.
Конец операции записи (программирование или стирание) может вызвать прерывание. Это прерывание можно использовать, чтобы выйти от режима ожидания **WFI**, если конечно разрешен тактовый для **FLITF**. Иначе, прерывание будет обслужено только после выхода из режима ожидания **WFI**.

***Note**: For further information on Flash memory operations and register configurations, please refer to the STM32F10xxx Flash programming manual.*
***Note**: Для дополнительной информации об операциях с флэш-памятью и конфигурационных регистрах обратитесь к документу "**STM32F10xxx Flash programming manual**".*

2.4 Boot configuration (Конфигурация начальной загрузки)

In the STM32F10xxx, 3 different boot modes can be selected through BOOT[1:0] pins as shown in Table 6.
Серия устройств **STM32F10xxx** имеет 3 различных режима начальной загрузки, которые могут быть выбраны с помощью выводов **BOOT[1:0]**, как показано в таблице 6.

Table 6. Boot modes (Режимы начальной загрузки)

Boot mode selection pins		**Boot mode**	**Aliasing**
BOOT 1	**BOOT 0**		
x	0	Main Flash memory	Main Flash memory is selected as boot space (В качестве области загрузки выбрана основная флэш-память)
0	1	System memory	System memory is selected as boot space (В качестве области загрузки выбрана системная память *(встроенный загрузчик)*)
1	1	Embedded SRAM	Embedded SRAM is selected as boot space (В качестве области загрузки выбрана встроенная **SRAM** память)

The values on the BOOT pins are latched on the 4th rising edge of SYSCLK after a Reset. It is up to the user to set the BOOT1 and BOOT0 pins after Reset to select the required boot mode.
Значения на **BOOT** выводах считыватываются на 4-ом фронте **SYSCLK** после сброса. Именно пользователь ставит уровни на выводах **BOOT1** и **BOOT0** после сброса, чтобы получить требуемый режим начальной загрузки.

The BOOT pins are also re-sampled when exiting from Standby mode. Consequently they must be kept in the required Boot mode configuration in Standby mode. After this startup delay has elapsed, the CPU fetches the top-of-stack value from address 0x0000 0000, then starts code execution from the boot memory starting from 0x0000 0004.
При выходе из режима ожидания **BOOT** выводы считыватываются повторно. Следовательно, в режиме ожидания они должны удерживаться в необходимой конфигурации для режима загрузки. После того, как истекла эта задержка запуска, ЦПУ выбирает значение вершины стека по адресу **0x0000 0000**, затем запускает выполнение кода с области памяти начальной загрузки, начиная со смещения **0x0000 0004**.

Due to its fixed memory map, the code area starts from address 0x0000 0000 (accessed through the ICode/DCode buses) while the data area (SRAM) starts from address 0x2000 0000 (accessed through the system bus). The Cortex-M3 CPU always fetches the reset vector on the ICode bus, which implies to have the boot space available only in the code area (typically, Flash memory). STM32F10xxx microcontrollers implement a special mechanism to be able to boot also from SRAM and not only from main Flash memory and System memory.
Вследствии фиксированной карты памяти, область кода начинается с адреса **0x0000 0000** (доступна через шины **ICode/DCode**), в то время как область данных (**SRAM**) начинается с адреса **0x2000 0000** (доступна через системную шину). ЦПУ ядра **Cortex-M3** всегда выбирает вектор сброса на шине **ICode**, что подразумевает, что область начальной загрузки доступна только в области кода (обычно, Флэш-память). Микроконтроллеры **STM32F10xxx** имеют специальный механизм, который дает возможность загрузиться также из **SRAM**, а не только из главной Флэш-памяти или Системной памяти.

Depending on the selected boot mode main Flash memory, System memory or SRAM is accessible as follows:
В зависимости от выбранного режима начальной загрузки, основной Флэш-памяти, системной памяти или **SRAM**, доступно следующие:

- Boot from main Flash memory: the main Flash memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x800 0000). In other words, the Flash memory contents can be accessed starting from address 0x0000 0000 or 0x800 0000.
Загрузка из главной Флэш-памяти: главная Флэш-память - отражается (*aliased*) на пространстве памяти начальной загрузки (**0x0000 0000**), но все еще доступна по ее оригинальной области памяти (**0x800 0000**). Другими словами, к контексту Флэш-памяти можно обратиться, начиная с адреса **0x0000 0000** или **0x800 0000**.

- Boot from System memory: the System memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x1FFF B000 in connectivity line devices, 0x1FFF F000 in other devices).
Загрузка из Системной памяти: Системная память - отражается (*aliased*) на пространстве памяти начальной загрузки (**0x0000 0000**), но все еще доступна по ее оригинальной области памяти (**0x1FFF B000** в семействе **CL**, **0x1FFF F000** в других семействах).

- Boot from the embedded SRAM: SRAM is accessible only at address 0x2000 0000.
Загрузка из встроенной **SRAM**: **SRAM** доступна только с адреса **0x2000 0000**.

***Note**: When booting from SRAM, in the application initialization code, you have to relocate the vector table in SRAM using the NVIC exception table and offset register. При загрузке из **SRAM**, в прикладном коде инициализации, вы должны переместить таблицу векторов в **SRAM** используя таблицу исключений **NVIC** и регистр смещения.*

Embedded boot loader (Встроенный загрузчик)

The embedded boot loader is located in the System memory, programmed by ST during production. It is used to reprogram the Flash memory with one of the available serial interfaces:
Встроенный загрузчик расположен в Системной памяти, и программируется фирмой **ST** при изготовлении продукции. Он используется для перепрограммирования Флэш-памяти с помощью одного из доступных последовательных интерфейсов:

- In low-, medium- and high-density devices the bootloader is activated through the USART1 interface. For further details please refer to AN2606.
В семействах **LD**, **MD** и **HD** загрузка производится через интерфейс **USART1**. Обратитесь к документу "***AN2606***" за более детальной информацией.

- In connectivity line devices the bootloader can be activated through one of the following interfaces: USART1, USART2 (remapped), CAN2 (remapped) or USB OTG FS in Device mode (DFU: device firmware upgrade).
The USART peripheral operates with the internal 8 MHz oscillator (HSI). The CAN and USB OTG FS, however, can only function if an external 8 MHz, 14.7456 MHz or 25 MHz clock (HSE) is present. For further details, please refer to AN2662.
В семействе **CL** загрузка может производится через один из следующих интерфейсов: **USART1**, **USART2** (переназначенный), **CAN2** (переназначенный) или **USB OTG FS** в режиме **Device** (**DFU**: *device firmware upgrade* (обновление программы МК)).
USART работает от внутреннего генератора на 8 МГц (**HSI**). Однако, **CAN** и **USB OTG FS** могут работать только тогда, когда представлен внешний резонатор на 8, 25 или 14.7456 МГц (**HSE**). Обратитесь к документу "***AN2662***" за более детальной информацией.

3 CRC calculation unit (Модуль расчета CRC)

(*Здесь находится ритуальное напоминание о наличии 4-х семейств для **STM32F10xxx**, и что этот документ обо всех семействах, если нет прямого указания. См. раздел* 1.2 Глоссарий*).*)

3.1 CRC introduction (Введение в **CRC** модуль)
3.2 CRC main features (Основные особенности **CRC** модуля)
3.3 CRC functional description (Функциональное описание **CRC** модуля)
3.4 CRC registers (Регистры **CRC** модуля)

3.1 CRC introduction (Введение в CRC модуль)

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from a 32-bit data word and a fixed generator polynomial.
Модуль вычисления **CRC** (контрольной суммы) используется, чтобы получить код **CRC** для 32-х разрядного слова данных с фиксированным полиномом.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the EN/IEC 60335-1 standard, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link-time and stored at a given memory location.
Технология использования **CRC**, это просто программа, которая используются для проверки целостности передачи данных или содержимого в памяти. В границах стандарта **EN/IEC 60335-1** эта технология предлагает средство подтверждения целостности Флэш-памяти. Модуль вычисления **CRC** помогает определить сигнатуру программного обеспечения во время ее выполнения и сравнить ее с той сигнатурой, что была сгенерированна во время линковки программы и была сохранена в определенном местоположении памяти.

3.2 CRC main features (Основные особенности CRC модуля)

- Uses CRC-32 (Ethernet) polynomial (32-х разрядный полином пользователя (для **Ethernet**)): 0x4C11DB7
`-X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 +X8 + X7 + X5 + X4 + X2+ X +1`

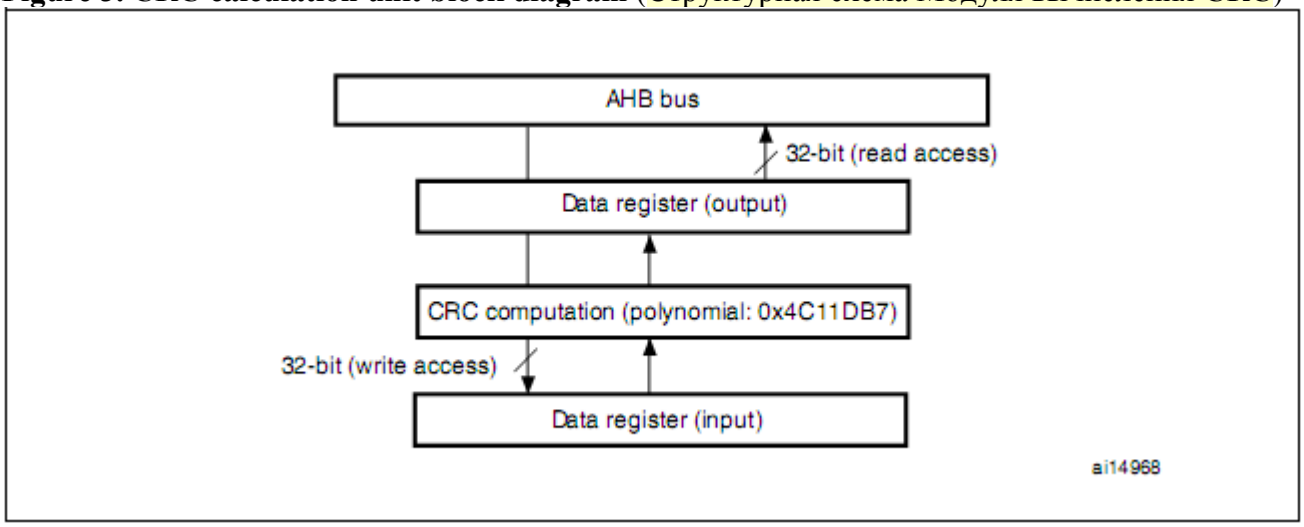
- Single input/output 32-bit data register (Единый 32-х разрядный регистр данных)

- CRC computation done in 4 AHB clock cycles (HCLK)
Расчет **CRC** выполняется за 4 такта шины **AHB** (**HCLK**)

- General-purpose 8-bit register (can be used for temporary storage) (8-ми битный регистр общего назначения (можно использовать для хранения временных данных))

The block diagram is shown in Figure 3. (Структурная схема представлена на рис. 3.)

Figure 3. CRC calculation unit block diagram (Структурная схема Модуля вычисления **CRC**)



3.3 CRC functional description
Функциональное описание CRC модуля

The CRC calculation unit mainly consists of a single 32-bit data register, which:
Модуль вычисления **CRC** состоит, в основном, из единственного 32-х разрядного регистра, который:

- is used as an input register to enter new data in the CRC calculator (when writing into the register)
используется как входной регистр, чтобы ввести новые данные в калькулятор **CRC** (при записи в регистр)
- holds the result of the previous CRC calculation (when reading the register)
содержит результат предыдущего вычисления **CRC** (при чтении регистра)

Each write operation into the data register creates a combination of the previous CRC value and the new one (CRC computation is done on the whole 32-bit data word, and not byte per byte).
Каждая операция записи в регистр данных создает комбинацию предыдущего значения **CRC**, и его нового значения (вычисление **CRC** выполняется над целым 32-разрядным словом, а не побайтно).

The write operation is stalled until the end of the CRC computation, thus allowing back-to-back write accesses or consecutive write and read accesses.
Операция записи остановливается до завершения вычисления **CRC**, позволяя, таким образом, осуществлять запись "впритык" или осуществлять последовательный доступ записи и чтения.

The CRC calculator can be reset to FFFF FFFFh with the RESET control bit in the CRC_CR register. This operation does not affect the contents of the CRC_IDR register.
Модуль вычисления **CRC** можно сбросить в значение **FFFF FFFFh** с помощью бита **RESET** в регистре **CRC_CR**. Эта операция не затрагивает содержимое регистра **CRC_IDR**.

3.4 CRC registers (Регистры CRC модуля)

3.4.1 Data register (CRC_DR) (Регистр данных)
3.4.2 Independent data register (CRC_IDR) (Регистр независимых данных)
3.4.3 Control register (CRC_CR) (Регистр управления)
3.4.4 CRC register map (Карта регистров **CRC**)

The CRC calculation unit contains two data registers and a control register.
Модуль вычисления **CRC** содержит два регистра данных и управляющий регистр.

3.4.1 Data register (CRC_DR) (Регистр данных)

Address offset: 0x00
Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR [31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR [15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Data register bits (Биты регистра данных)
Used as an input register when writing new data into the CRC calculator. Holds the previous CRC calculation result when it is read.
Используется как входной регистр при записи новых данных в калькулятор **CRC**. При чтении содержит предыдущий результат вычисления **CRC**.

3.4.2 Independent data register (CRC_IDR) (Регистр независимых данных)

Address offset: 0x04
Reset value: 0x0000 0000

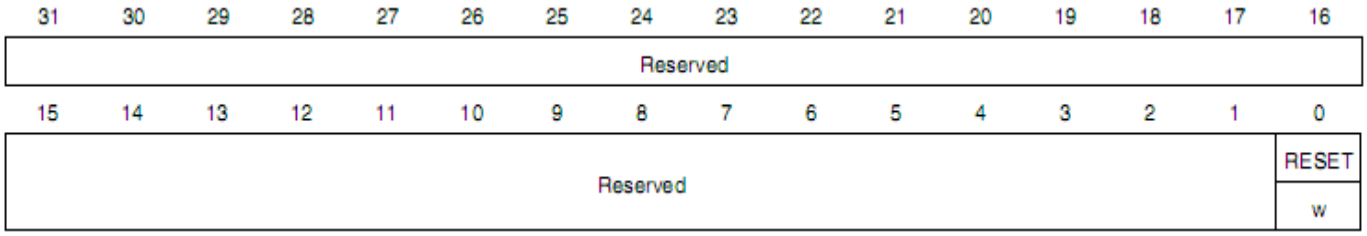
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								IDR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved

Bits 7:0 General-purpose 8-bit data register bits (8-ми битный регистр общего назначения)
Can be used as a temporary storage location for one byte. This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register.
Может использоваться для временного хранения одного байта. Этот регистр не затрагивается сбросом **CRC**, вызванного битом регистра **CRC_CR**.

3.4.3 Control register (CRC_CR) (Регистр управления)

Address offset: 0x08
Reset value: 0x0000 0000



Bits 31:1 Reserved

Bit 0 RESET bit (Бит сброса)

Resets the CRC calculation unit and sets the data register to FFFF FFFFh. This bit can only be set, it is automatically cleared by hardware.
Сбрасывает модуль вычисления **CRC** и устанавливает регистр данных в значение **FFFF FFFFh**. Этот бит можно только ставить, сбрасывается он автоматически, аппаратно.

3.4.4 CRC register map (Карта регистров CRC)

The following table provides the CRC register map and reset values.
Последующая таблица дает карту регистров модуля **CRC** и их значения после сброса.

Table 7. CRC calculation unit register map and reset values
Карта регистров **CRC** и их значение после сброса

Offset	Register	31-24	23-16	15-8	7	6	5	4	3	2	1	0
0x00	**CRC_DR** Reset value	Data register 0xFFFF FFFF										
0x04	**CRC_IDR** Reset value	Reserved			Independent data register 0x00							
0x08	**CRC_CR** Reset value	Reserved								Reserved 0		RESET 0

4 Power control (PWR) (Управление питанием)

(*Здесь находится ритуальное напоминание о наличии 4-х семейств для **STM32F10xxx**, и что этот документ обо всех семействах, если нет прямого указания. См. раздел 1.2 Глоссарий).*)

4.1 Power supplies (Подача питания)
4.2 Power supply supervisor (Супервизор подачи питания)
4.3 Low-power modes (Режимы низкого потребления)
4.4 Power control registers (Регистры управления питанием)

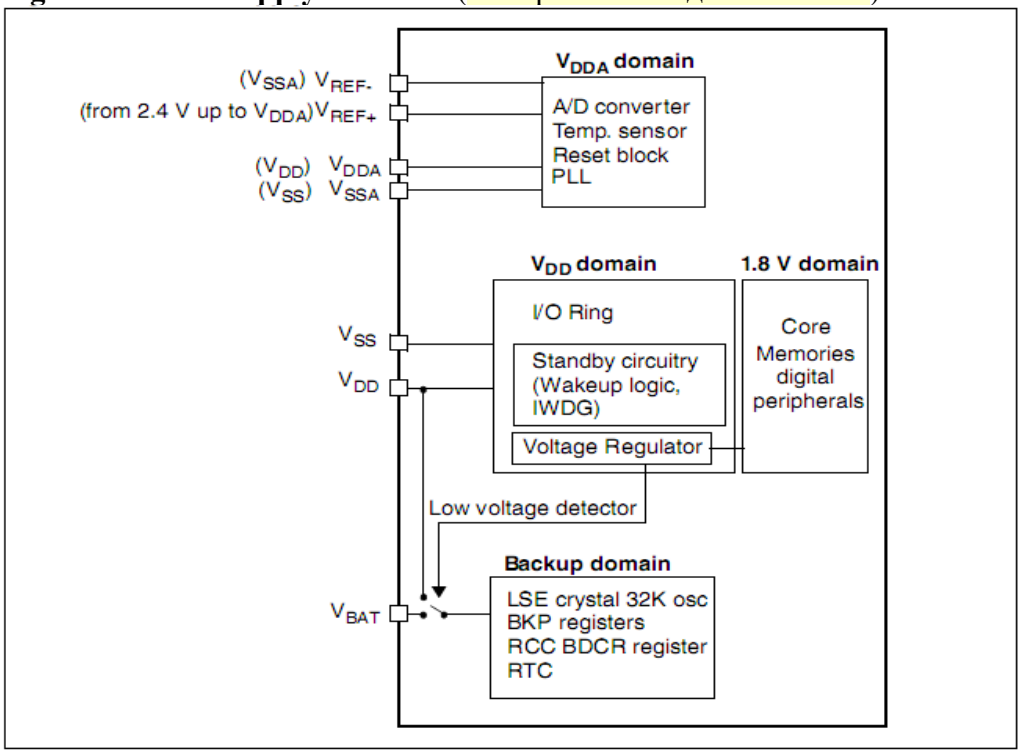
4.1 Power supplies (Подача питания)

4.1.1 Independent A/D converter supply and reference voltage
Независимое питание АЦП и опорное напряжение
4.1.2 Battery backup domain (Резервное питание)
4.1.3 Voltage regulator (Регулятор напряжения)

The device requires a 2.0-to-3.6 V operating voltage supply (V_{DD}). An embedded regulator is used to supply the internal 1.8 V digital power.
Устройству необходимо рабочее напряжение (**V_{DD}**) от 2.0 до 3.6 В. чтобы обеспечить внутреннее напряжение 1.8 В используется встроенный импульсный регулятор напряжения.

The real-time clock (RTC) and backup registers can be powered from the V_{BAT} voltage when the main V_{DD} supply is powered off. Тактовый сигнал часов реального времени (**RTC**) и резервные регистры можно запитывать от цепи **V_{BAT}**, когда основное питание на цепи **V_{DD}** отключено.

Figure 4. Power supply overview (Обзор схемы подачи питания)



***Note**: 1 V_{DDA} and V_{SSA} must be connected to V_{DD} and V_{SS}, respectively.*
*Цепи **V_{DDA}** и **V_{SSA}** должны быть подключены к **V_{DD}** и **V_{SS}**, соответственно.*

4.1.1 Independent A/D converter supply and reference voltage
Независимое питание АЦП и опорное напряжение

To improve conversion accuracy, the ADC has an independent power supply which can be separately filtered and shielded from noise on the PCB.
Чтобы улучшить точность преобразования, АЦП имеет независимую цепь питания, на которую можно "повесить" отдельный фильтр, чтобы оградить ее от шумов на плате.

- The ADC voltage supply input is available on a separate V_{DDA} pin.

Цепь питания АЦП имеет отдельный вывод $\mathbf{V_{DDA}}$.

- An isolated supply ground connection is provided on pin V_{SSA}.

Изолированная общая цепь питания предоставлена на выводе $\mathbf{V_{SSA}}$.

When available (according to package), V_{REF}- must be tied to V_{SSA}.
Когда есть вывод $\mathbf{V_{REF}}$ (зависит от корпуса), он должен быть связан с $\mathbf{V_{SSA}}$.

On 100-pin and 144- pin packages (Для корпусов на 100 и 144 вывода)

To ensure a better accuracy on low voltage inputs, the user can connect a separate external reference voltage ADC input on V_{REF+} and V_{REF-}. The voltage on V_{REF+} can range from 2.4 V to V_{DDA}.
Чтобы гарантировать лучшую точность преобразования низко-вольтного сигнала, пользователь может подключить отдельный внешний источник опорного напряжения к выводам $\mathbf{V_{REF+}}$ и $\mathbf{V_{REF-}}$. Напряжение на $\mathbf{V_{REF+}}$ может быть в пределах от 2.4 В до $\mathbf{V_{DDA}}$.

On 64-pin packages (Для корпусов на 64 вывода)

The VREF+ and V_{REF}- pins are not available, they are internally connected to the ADC voltage supply (V_{DDA}) and ground (V_{SSA}).
Выводы $\mathbf{V_{REF+}}$ и $\mathbf{V_{REF-}}$ недоступны, они внутренне подключены к источнику питания АЦП ($\mathbf{V_{DDA}}$) и общей цепи ($\mathbf{V_{SSA}}$).

4.1.2 Battery backup domain (Резервное питание)

To retain the content of the Backup registers and supply the RTC function when V_{DD} is turned off, V_{BAT} pin can be connected to an optional standby voltage supplied by a battery or by another source.
Чтобы сохранить содержимое **BKP** регистров и функционирование **RTC**, когда снято питание $\mathbf{V_{DD}}$, можно подключить вывод $\mathbf{V_{BAT}}$ к дополнительным резервному источнику питания, которое поступает от батарейки или другого источника.

The V_{BAT} pin powers the RTC unit, the LSE oscillator and the PC13 to PC15 IOs, allowing the RTC to operate even when the main digital supply (V_{DD}) is turned off. The switch to the V_{BAT} supply is controlled by the Power Down Reset embedded in the Reset block.
Цепь $\mathbf{V_{BAT}}$ питает модуль **RTC**, **LSE** генератор и **I/O** выводы от **PC13** до **PC15**, что позволяет **RTC** работать даже тогда, когда снято основное питание цифровых цепей ($\mathbf{V_{DD}}$). Подключением питания от $\mathbf{V_{BAT}}$ управляет модуль **Power Down Reset** (*Сброс от пропадания напряжения*), который встроен в модуль Сброса.

***Warning**: During $t_{RSTTEMPO}$ (temporization at V_{DD} startup) or after a PDR is detected, the power switch between V_{BAT} and V_{DD} remains connected to V_{BAT}. During the startup phase, if V_{DD} is established in less than $t_{RSTTEMPO}$ (Refer to the datasheet for the value of $t_{RSTTEMPO}$) and $V_{DD} > V_{BAT} + 0.6$ V, a current may be injected into V_{BAT} through an internal diode connected between V_{DD} and the power switch (V_{BAT}). If the power supply/battery connected to the V_{BAT} pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the V_{BAT} pin.*

***Предупреждение**: Во время $t_{RSTTEMPO}$ (отсчитывается от подачи V_{DD}) или после того, как обнаружен **PDR**, переключатель питания питания между V_{BAT} и V_{DD} остается связанным с V_{BAT}. Во время фазы запуска, если V_{DD} установился раньше, чем закончилось $t_{RSTTEMPO}$ (см. **datasheet** относительно значения $t_{RSTTEMPO}$), и $V_{DD} > V_{BAT} + 0.6$ В, то может возникнуть ток, втекающий в цепь V_{BAT} через внутренний диод, связывающий V_{DD} и переключатель питания (V_{BAT}). Если источник питания/батарейка, подключенный к выводу V_{BAT}, не может обеспечить такой ток, то настоятельно рекомендуется подключить внешний диод с низким падением напряжения между таким источником питания и выводом V_{BAT}.*

If no external battery is used in the application, it is recommended to connect V_{BAT} externally to V_{DD} through a 100 nF external ceramic capacitor (for more details refer to AN2586).
Если внешняя батарейка не используется, то рекомендуется подключить цепь **V_{BAT}** к цепи **V_{DD}** через внешний керамический конденсатор на 100 **nF** (за дополнительной информацией обратитесь к документу "***AN2586***").

When the backup domain is supplied by V_{DD} (analog switch connected to V_{DD}), the following functions are available:
Когда модуль **BKP** питается от **V_{DD}** (аналоговый переключатель подключен к **V_{DD}**), то доступны следующие функции:

- PC14 and PC15 can be used as either GPIO or LSE pins
  Выводы **PC14** и **PC15** можно использовать как **GPIO** или как выводы **LSE**.
- PC13 can be used as GPIO, TAMPER pin, RTC Calibration Clock, RTC Alarm or second output (refer to *Section 5: Backup registers (BKP) on page 66*)
  Вывод **PC13** можно использовать как **GPIO**, вывод **TAMPER**, выход калибровки тактового для **RTC**, выход тревоги от **RTC** или как вторичный выход (см. раздел 5 "Резервные регистры").

***Note**: Due to the fact that the switch only sinks a limited amount of current (3 mA), the use of GPIOs PC13 to PC15 in output mode is restricted: the speed has to be limited to 2 MHz with a maximum load of 30 pF and these IOs must not be used as a current source (e.g. to drive an LED).*
***Note**: Вследствие того, что нагрузочная способность переключателя ограничена (3 **mA**), то ограничено и использование **PC13...PC15** в качестве выхода общего назначения: частота должна быть ограничена до 2 МГц с максимальной нагрузкой 30 **pF**, и эти выводы не должны использоваться в качестве текущего источника (например, для управления светодиодом).*

When the backup domain is supplied by V_{BAT} (analog switch connected to V_{BAT} because V_{DD} is not present), the following functions are available:
Когда модуль **BKP** питается от **V_{BAT}** (аналоговый переключатель подключен к **V_{BAT}**, так как **V_{DD}** отключен), то доступны следующие функции:

- PC14 and PC15 can be used as LSE pins only

Выводы **PC14** и **PC15** можно использовать только как выводы **LSE**

- PC13 can be used as TAMPER pin, RTC Alarm or Second output (refer to *Section 5.4.2: RTC clock calibration register (BKP_RTCCR) on page 68*).

Вывод **PC13** можно использовать как вывод **TAMPER**, выход тревоги от **RTC** или как вторичный выход (см. раздел 5.4.2 "Регистр калибровки **RTC (BKP_RTCCR)**").

4.1.3 Voltage regulator (Регулятор напряжения)

The voltage regulator is always enabled after Reset. It works in three different modes depending on the application modes. Регулятор напряжения всегда разрешен после Сброса. Он работает в трех различных режимах в зависимости от режимов приложения.

- In Run mode, the regulator supplies full power to the 1.8 V domain (core, memories and digital peripherals).

В Рабочем режиме регулятор снабжает полным питанием модули, работающие от напряжения 1.8 В (ядро, память и цифровая периферия).

- In Stop mode the regulator supplies low-power to the 1.8 V domain, preserving contents of registers and SRAM.

В режиме Приостановки (*Stop*) регулятор снабжает пониженным питанием модули, работающие от напряжения 1.8 В, при этом содержимое регистров и **SRAM** сохраняется.

- In Standby Mode, the regulator is powered off. The contents of the registers and SRAM are lost except for the Standby circuitry and the Backup Domain.

В режиме Ожидания (*Standby*) регулятор выключается. Содержимое регистров и **SRAM** теряется, за исключением модулей **Standby** и **BKP**.

4.2 Power supply supervisor (Супервизор подачи питания)

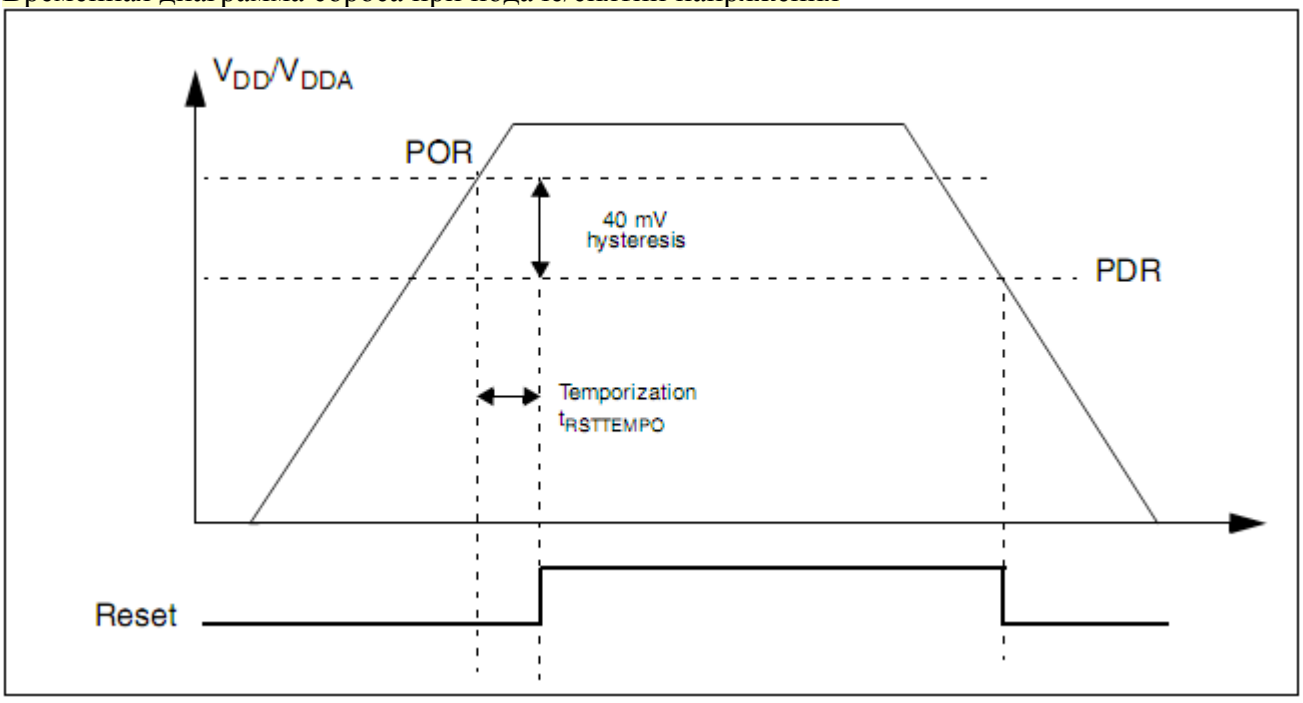
4.2.1 Power on reset (POR)/power down reset (PDR)
(Сброс при подаче/снятии напряжения)
4.2.2 Programmable voltage detector (PVD) (Программируемый детектор напряжения)

4.2.1 Power on reset (POR)/power down reset (PDR)
Сброс при подаче/снятии напряжения

The device has an integrated POR/PDR circuitry that allows proper operation starting from/down to 2 V.
Устройства имеют интегрированную схему **POR/PDR**, которая обеспечивает надлежащие операции при превышении/понижении порога напряжения 2 В.

The device remains in Reset mode when V_{DD}/V_{DDA} is below a specified threshold, $V_{POR/PDR}$, without the need for an external reset circuit. For more details concerning the power on/power down reset threshold, refer to the electrical characteristics of the datasheet.
Устройство остается в состоянии сброса, пока $\mathbf{V_{DD}/V_{DDA}}$ ниже указанного порога ($\mathbf{V_{POR/PDR}}$), без потребности во внешнем сигнале по цепи сброса. За дополнительной информацией относительно порога сброса, см. электрические характеристики в **datasheet**.

Figure 5. Power on reset/power down reset waveform
Временная диаграмма сброса при подаче/снятии напряжения



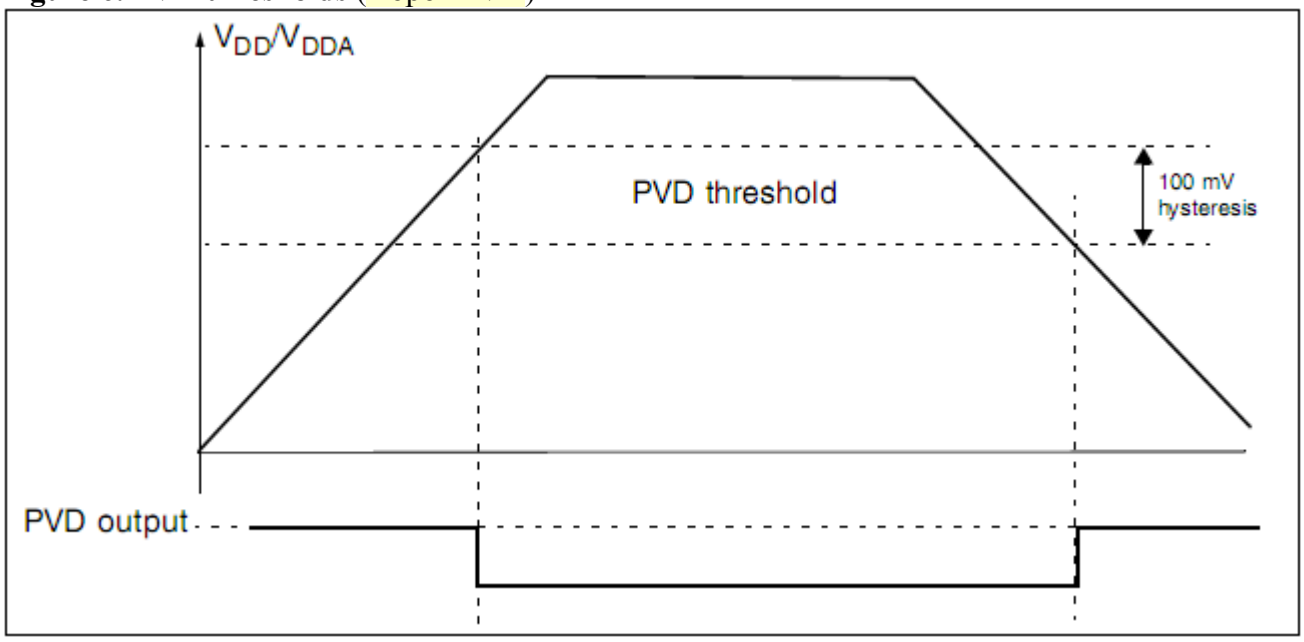
4.2.2 Programmable voltage detector (PVD)
Программируемый детектор напряжения

You can use the PVD to monitor the V_{DD}/V_{DDA} power supply by comparing it to a threshold selected by the PLS[2:0] bits in the Power control register (PWR_CR).
Вы можете использовать **PVD**, чтобы контролировать напряжение питания $\mathbf{V_{DD}/V_{DDA}}$, сравнивая его с порогом, выбранным битами поля **PLS[2:0]** в регистре управления питанием **PWR_CR**.

The PVD is enabled by setting the PVDE bit. (Разрешение **PVD** устанавливается битом **PVDE**.)

A PVDO flag is available, in the Power control/status register (PWR_CSR), to indicate if V_{DD}/V_{DDA} is higher or lower than the PVD threshold. This event is internally connected to the EXTI line16 and can generate an interrupt if enabled through the EXTI registers. The PVD output interrupt can be generated when V_{DD}/V_{DDA} drops below the PVD threshold and/or when V_{DD}/V_{DDA} rises above the PVD threshold depending on EXTI line16 rising/falling edge configuration. As an example the service routine could perform emergency shutdown tasks.
Флаг **PVDO**, доступный в регистре **PWR_CSR**, показавает, выше ли напряжение $\mathbf{V_{DD}/V_{DDA}}$ порога **PVD**, или ниже. Это событие внутренне связано с линией 16 **EXTI** и может генерировать прерывание, если оно разрешено в **EXTI** регистрах. Прерывание от **PVD** может быть сгенерировано, когда $\mathbf{V_{DD}/V_{DDA}}$ снижается ниже порога **PVD** и/или когда $\mathbf{V_{DD}/V_{DDA}}$ повышается выше порога **PVD**, в зависимости от конфигурации перепада (фронт/срез) на линии 16 **EXTI**. Например, обработчик прерывания может выполнить некие задачи при аварийном завершении задачи.

Figure 6. PVD thresholds (Порог **PVD**)



4.3 Low-power modes (Режимы низкого потребления)

4.3.1 Slowing down system clocks (Понижение частоты системных тактовых сигналов)
4.3.2 Peripheral clock gating (Стробирование тактовых сигналов периферии)
4.3.3 Sleep mode (Режим Сна)
4.3.4 Stop mode (Режим Приостановки)
4.3.5 Standby mode (Режим Ожидания)
4.3.6 Auto-wakeup (AWU) from low-power mode
Автопробуждение из режима низкого потребления

By default, the microcontroller is in Run mode after a system or a power Reset. Several low-power modes are available to save power when the CPU does not need to be kept running, for example when waiting for an external event. It is up to the user to select the mode that gives the best compromise between low-power consumption, short startup time and available wakeup sources.
По умолчанию, микроконтроллер находится в Рабочем режиме после системного Сброса или Сброса от подачи питания. Доступны несколько режимов с низким потреблением, для экономии питания, когда нет необходимости в работе ЦПУ, например, при ожидании внешнего события. Именно пользователь выбрает тот режим, который дает наилучший компромисс между уровнем потребления, временем восстановления и доступным источником события пробуждения.

The STM32F10xxx devices feature three low-power modes:
Устройства **STM32F10xxx** имеют три режима низкого потребления:

- Sleep mode (CPU clock off, all peripherals including Cortex-M3 core peripherals like NVIC, SysTick, etc. are kept running)
  Режим Сна (тактовый сигнал ЦПУ выключен, а вся периферия, включая периферию ядра **Cortex-M3**, такую как **NVIC**, **SysTick**, и т.д., продолжает работать)

- Stop mode (all clocks are stopped)
  Режим Приостановки (остановлены все тактовые сигналы)

- Standby mode (1.8 V domain powered-off)
  Режим Ожидания (снято питание с модулей, потребляющих 1.8 В)

In addition, the power consumption in Run mode can be reduce by one of the following means:
Кроме того, потребление в Рабочем режиме может быть уменьшено одним из следующих способов:

- Slowing down the system clocks (Уменьшение частоты системного тактового сигнала)
- Gating the clocks to the APB and AHB peripherals when they are unused.
  Стробирование (отключение) тактового сигнала периферии на шинах **APB** и **AHB**, когда они не используются.

Table 8. Low-power mode summary (Сводка по режимам низкого потребления)

Mode name	Entry (Вход в режим)	wakeup (Пробуждение)	Effect on 1.8V domain clocks	Effect on V_{DD} domain clocks	Voltage regulator
Sleep (Sleep now or Sleep-on-exit) (Заснуть сразу или по выходу из ISR)	WFI	Any interrupt (Любое прерывание)	CPU clock OFF no effect on other clocks or analog clock sources Тактовый ЦПУ выкл. на остальные тактовые не действует	None	ON
	WFE	Wakeup event (Событие побудки)			
Stop	PDDS and LPDS bits + SLEEPDEEP bit + WFI or WFE	Any EXTI line (configured in the EXTI registers) Любая **EXTI** линия	All 1.8V domain clocks OFF Выкл. тактовый всех 1.8 В модулей	HSI and HSE oscillators OFF	ON or in low-power mode (depends on Power control register (PWR_CR))
Standby	PDDS bit + SLEEPDEEP bit + WFI or WFE	WKUP pin rising edge, RTC alarm, external reset in NRST pin, IWDG reset			OFF

4.3.1 Slowing down system clocks
Понижение частоты системных тактовых сигналов

In Run mode the speed of the system clocks (SYSCLK, HCLK, PCLK1, PCLK2) can be reduced by programming the prescaler registers. These prescalers can also be used to slow down peripherals before entering Sleep mode.
В Рабочем режиме частоту системных тактовых сигналов (**SYSCLK**, **HCLK**, **PCLK1**, **PCLK2**) можно уменьшить, программируя регистры делителя частоты. Эти делители частоты могут также использоваться, чтобы замедлить периферию перед входом в режим Сна.

For more details refer to *Section 6.3.2: Clock configuration register (RCC_CFGR)*.
Для дополнительной информаци обратитесь к разделу 6.3.2 "**Clock configuration register (RCC_CFGR)**"

4.3.2 Peripheral clock gating
Стробирование тактовых сигналов периферии

In Run mode, the HCLK and PCLKx for individual peripherals and memories can be stopped at any time to reduce power consumption.
В Рабочем режиме тактовые **HCLK** и **PCLKx** для индивидуальных периферийных устройств и памяти можно остановить в любое время, чтобы уменьшить потребление.

To further reduce power consumption in Sleep mode the peripheral clocks can be disabled prior to executing the WFI or WFE instructions.
Чтобы еще уменьшить потребление мощность в режиме Сна, можно отключить тактовые сигналы

периферии до выполнения инструкций **WFI** или **WFE**.

Peripheral clock gating is controlled by the *AHB peripheral clock enable register (RCC_AHBENR)*, *APB1 peripheral clock enable register (RCC_APB1ENR)* and *APB2 peripheral clock enable register (RCC_APB2ENR)*.
Стробирование тактовых сигналов периферии управляется регистрами **RCC_AHBENR** (*Регистр разрешения тактовых сигналов **AHB** периферии*), **RCC_APB1ENR** (*Регистр разрешения тактовых сигналов **APB1** периферии*), и **RCC_APB2ENR** (*Регистр разрешения тактовых сигналов **APB2** периферии*).

4.3.3 Sleep mode (Режим Сна)

Entering Sleep mode (Вход в режим Сна)

The Sleep mode is entered by executing the WFI (Wait For Interrupt) or WFE (Wait for Event) instructions. Two options are available to select the Sleep mode entry mechanism, depending on the SLEEPONEXIT bit in the Cortex-M3 System Control register:
Чтобы войти в режим Сна, надо выполнить инструкцию **WFI** (ожидание прерывания), или **WFE** (ожидание события). Для входа в механизм сна доступны две опции, в зависимости от бита **SLEEPONEXIT** в регистре Управления Системой **Cortex-M3** (**SCB_SCR**):

- Sleep-now: if the SLEEPONEXIT bit is cleared, the MCU enters Sleep mode as soon as WFI or WFE instruction is executed.
  Заснуть немедленно: если бит **SLEEPONEXIT=0**, ЦПУ входит в режим Сна сразу после выполнения инструкции **WFI** или **WFE**.
- Sleep-on-exit: if the SLEEPONEXIT bit is set, the MCU enters Sleep mode as soon as it exits the lowest priority ISR.
  Заснуть по выходу: если бит **SLEEPONEXIT=1**, ЦПУ входит в режим Сна сразу по выходу из обработчика прерывания с низшим приоритетом.

In the Sleep mode, all I/O pins keep the same state as in the Run mode.
В режиме Сна все **I/O** выводы удерживаются в том состоянии, что были в Рабочем режиме.

Refer to Table 9 and Table 10 for details on how to enter Sleep mode.
См. таблицы 9 и 10 для дополнительной информации о входе в режим Сна.

Exiting Sleep mode (Выход из режима Сна)

If the WFI instruction is used to enter Sleep mode, any peripheral interrupt acknowledged by the nested vectored interrupt controller (NVIC) can wake up the device from Sleep mode. If the WFE instruction is used to enter Sleep mode, the MCU exits Sleep mode as soon as an event occurs. The wakeup event can be generated either by:
Если для входа в режим Сна используется инструкция **WFI**, то любое прерывание периферии, принятое контроллером векторов вложенных прерывания (**NVIC**), может разбудить устройство из этого Режима. Если для входа в режим Сна используется инструкция **WFE**, то ЦПУ выйдет из режима Сна, как только произойдет событие. Событие пробуждения может быть сгенерировано любым из следующих действий:

- enabling an interrupt in the peripheral control register but not in the NVIC, and enabling the SEVONPEND bit in the Cortex-M3 System Control register. When the MCU resumes from WFE, the peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.

разрешение прерывания в регистре управления периферией, но не в **NVIC**, и установка бита **SEVONPEND** в регистре Управления Системой **Cortex-M3** (**SCB_SCR**). Когда ЦПУ восстановится от **WFE**, то биты запроса прерываний от периферии и биты запроса прерываний от каналов **NVIC** (в регистре **NVIC_ICPRx**) необходимо очистить.

- or configuring an external or internal EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bit corresponding to the event line is not set.
или конфигурирование внешней или внутренней линии **EXTI** в режиме события. Когда ЦПУ восстановится от **WFE**, то нет необходимости в очистке битов запроса прерываний от периферии и битов запроса прерываний от каналов **NVIC**, так как биты запроса, соответствующие линиям событий, не установливаются.

This mode offers the lowest wakeup time as no time is wasted in interrupt entry/exit.
Этот режим предоставляет самое низкое время пробуждения, поскольку нет дополнительных ненужных затрат времени при входе/выходе из прерывания.

Refer to Table 9 and Table 10 for more details on how to exit Sleep mode.
См. таблицы 9 и 10 для дополнительной информации о выходе из режима Сна.

Table 9. Sleep-now (Заснуть немедленно)

Sleep-now mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: Ожидание прерывания или ожидание события - SLEEPDEEP = 0 and - SLEEPONEXIT = 0 Refer to the Cortex™-M3 System Control register. (См. регистр **SCB_SCR**)
Mode exit	If WFI was used for entry: (Если для входа используется **WFI**:) Interrupt: Refer to *Table 53: Vector table for other STM32F10xxx devices* Прерывание: См. табл. 53 If WFE was used for entry (Если для входа используется **WFE**:) Wakeup event: Refer to *Section 9.2.3: Wakeup event management* Событие пробуждения: См. раздел 9.2.3.
Wakeup latency	None

Table 10. Sleep-on-exit (Заснуть по выходу)

Sleep-on-exit	Description
Mode entry	WFI (wait for interrupt) while: (Ожидание прерывания) - SLEEPDEEP = 0 and - SLEEPONEXIT = 1 Refer to the Cortex™-M3 System Control register. (См. регистр **SCB_SCR**)
Mode exit	Interrupt: Refer to *Table 53: Vector table for other STM32F10xxx devices* Прерывание: См. табл. 53
Wakeup latency	None

4.3.4 Stop mode (Режим Приостановки)

The Stop mode is based on the Cortex-M3 deepsleep mode combined with peripheral clock gating. The voltage regulator can be configured either in normal or low-power mode. In Stop mode, all clocks in the 1.8 V domain are stopped, the PLL, the HSI and the HSE RC oscillators are disabled. SRAM and register contents are preserved.
Режим Приостановки основан на режиме глубокого сна для **Cortex-M3**, в комбинации со стробированием тактовых сигналов периферии. Регулятор напряжения может быть конфигурирован на нормальный режим или режим с низким потреблением. В режиме Приостановки остановлены все тактовые сигналы модулей с питанием от 1.8 В, а **PLL**, **HSI** и **RC HSE** генераторы отключены. Содержимое **SRAM** и регистров сохраняется.

In the Stop mode, all I/O pins keep the same state as in the Run mode.
В режиме Приостановки все **I/O** выводы сохраняют то состояние, что были в Рабочем режиме.

Entering Stop mode (Вход в режим Приостановки)

Refer to Table 11 for details on how to enter the Stop mode.
См. табл. 11 для дополнительной информации о входе в режим Приостановки.

To further reduce power consumption in Stop mode, the internal voltage regulator can be put in low-power mode. This is configured by the LPDS bit of the Power control register (PWR_CR).
Чтобы еще уменьшить потребление в режиме Приостановки, внутренний регулятор напряжения можно включить в режим с низким потреблением. Это конфигурируется битом **LPDS** регистра управления питанием (**PWR_CR**).

If Flash memory programming is ongoing, the Stop mode entry is delayed until the memory access is finished. Если идет процесс программирования Флэш-памяти, то вход в режим Приостановки откладывается, пока не завершиться доступ к памяти.

If an access to the APB domain is ongoing, The Stop mode entry is delayed until the APB access is finished. Если идет процесс доступа к домену **APB**, то вход в режим Приостановки откладывается, пока не завершиться доступ к **APB**.

In Stop mode, the following features can be selected by programming individual control bits:
В режиме Приостановки можно выбрать следующие особенности, программируя индивидуальные управляющие биты:

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a Reset. See Section 17.3 in Section 17: Independent watchdog (IWDG). Независимый **watchdog** (**IWDG**): **IWDG** запускается записью в его ключевой регистр или аппаратно. После запуска его нельзя остановить иначе как общим сбросом. См. подраздел 17.3 в разделе 17 "**Independent watchdog (IWDG)**".
- real-time clock (RTC): this is configured by the RTCEN bit in the Backup domain control register (RCC_BDCR) Часы реального времени (**RTC**): конфигурируются битом **RTCEN** регистра **RCC_BDCR**.
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the Control/status register (RCC_CSR). Внутренний **RC** генератор (**LSI RC**): конфигурируются битом **LSION** в регистре **RCC_CSR**.
- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the Backup domain control register (RCC_BDCR). Внешний генератор на 32.768 кГц (**LSE OSC**): конфигурируются битом **LSEON** в регистре **RCC_BDCR**.

The ADC or DAC can also consume power during the Stop mode, unless they are disabled before entering it. To disable them, the ADON bit in the ADC_CR2 register and the ENx bit in the DAC_CR

register must both be written to 0. АЦП или ЦАП также могут потреблять мощность во время режима Приостановки, если они не отключены перед входом в режим. Чтобы отключить их, надо сбросить два бита, **ADON** в регистре **ADC_CR2** и **ENx** в регистре **DAC_CR**.

Exiting Stop mode (Выход из режима Приостановки)

Refer to Table 11 for more details on how to exit Stop mode.
См. табл. 11 для дополнительной информации о выходе из режима Приостановки.

When exiting Stop mode by issuing an interrupt or a wakeup event, the HSI RC oscillator is selected as system clock. При выходе из режима Приостановки от прерывания или события, в качестве системного тактового сигнала выбирается **HSI RC** генератор.

When the voltage regulator operates in low-power mode, an additional startup delay is incurred when waking up from Stop mode. By keeping the internal regulator ON during Stop mode, the consumption is higher although the startup time is reduced. Когда регулятор напряжения находится в режиме с низким потреблением, то при выходе из режима Приостановки возникает дополнительная задержка запуска. Если в режиме Приостановки оставить внутренний регулятор включенным, то потребление будет выше, но время запуска сократиться.

Table 11. Stop mode (Режим Приостановки)

Stop mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: Ожидание прерывания или ожидание события - Set SLEEPDEEP bit in Cortex™-M3 System Control register Ставиться бит **SLEEPDEEP** в регистре **SCB_SCR** - Clear PDDS bit in Power Control register (PWR_CR) Очищается бит **PDDS** в регистре **PWR_CR** - Select the voltage regulator mode by configuring LPDS bit in PWR_CR Выбор режима регулятора битом **LPDS** в регистре **PWR_CR** **Note**: To enter Stop mode, all EXTI Line pending bits (in Pending register (EXTI_PR)) and RTC Alarm flag must be reset. Otherwise, the Stop mode entry procedure is ignored and program execution continues. **Note**: Чтобы войтм в режим Приостановки, надо сбросить все биты запроса по линиям **EXTI** в регистре **EXTI_PR** и сбросить флаг **RTC Alarm**.
Mode exit	If WFI was used for entry: (Если для входа используется **WFI**:) Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). Конфигурирование любой **EXTI** линии в режиме прерываний (должен быть разрешен соответствующий **EXTI** вектор). Refer to Table 53: Vector table for other STM32F10xxx devices on page 172. См. табл. 53. If WFE was used for entry: (Если для входа используется **WFE**:) Any EXTI Line configured in event mode. Refer to Section 9.2.3: Wakeup event management on page 175 Конфигурирование любой **EXTI** линии в режиме событий. См. раздел 9.2.3.
Wakeup latency	HSI RC wakeup time + regulator wakeup time from Low-power mode Задержка: время пробуждения **HSI RC** + время пробуждения регулятора из режима малого потребления.

4.3.5 Standby mode (Режим Ожидания)

The Standby mode allows to achieve the lowest power consumption. It is based on the Cortex-M3 deepsleep mode, with the voltage regulator disabled. The 1.8 V domain is consequently powered off. The PLL, the HSI oscillator and the HSE oscillator are also switched off. SRAM and register contents are lost except for registers in the Backup domain and Standby circuitry (see Figure 4).
Режим Ожидания позволяет достигать самого низкого потребления. Он основан на режиме глубокого сна для **Cortex-M3**, с выключенным регулятором напряжения. Все модули, питающиеся от 1.8 В последовательно выключаются. **HSI** и **HSE** генераторы и **PLL** также выключаются. Содержимое **SRAM** и регистров теряются, за исключением регистров модуля **BKP** и и схемного узла **Standby** (см. рис. 4).

Entering Standby mode (Вход в режим Ожидания)

Refer to Table 12 for more details on how to enter Standby mode. In Standby mode, the following features can be selected by programming individual control bits:
См. табл. 12 для дополнительной информации о входе в режим Ожидания. В Режиме Ожидания следующие особенности могут быть выбраны за счет программирования индивидуальных управляющих битов:

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a reset. See Section 17.3 in Section 17: Independent watchdog (IWDG).
Независимый **watchdog** (**IWDG**): **IWDG** запускается записью в его ключевой регистр или аппаратно. После запуска его нельзя остановить иначе как общим сбросом. См. подраздел 17.3 в разделе 17 "**Independent watchdog (IWDG)**".

- real-time clock (RTC): this is configured by the RTCEN bit in the Backup domain control register (RCC_BDCR)
Часы реального времени (**RTC**): конфигурируются битом **RTCEN** регистра **RCC_BDCR**.

- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the Control/status register (RCC_CSR).
Внутренний **RC** генератор (**LSI RC**): конфигурируются битом **LSION** в регистре **RCC_CSR**.

- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the Backup domain control register (RCC_BDCR)
Внешний генератор на 32.768 кГц (**LSE OSC**): конфигурируются битом **LSEON** в регистре **RCC_BDCR**.

Exiting Standby mode (Выход из режима Ожидания)

The microcontroller exits Standby mode when an external Reset (NRST pin), IWDG Reset, a rising edge on WKUP pin or an RTC alarm occurs. All registers are reset after wakeup from Standby except for Power control/status register (PWR_CSR).
Микроконтроллер выходит из режима Ожидания от внешнего сброса (на выводе **NRST**), сброса от **IWDG**, фронта сигнала на выводе **WKUP** или от тревоги **RTC**. После пробуждения из режима Ожидания все регистры сброшены, кроме регистра управления/состояния питания (**PWR_CSR**).

After waking up from Standby mode, program execution restarts in the same way as after a Reset (boot pins sampling, vector reset is fetched, etc.). The SBF status flag in the Power control/status register (PWR_CSR) indicates that the MCU was in Standby mode.
После пробуждения из режима Ожидания возникает рестарт выполнения программы, точно также, как от общего сброса (происходит считывание состояния выводов начальной загрузки, считывание

вектора сброса и т.д.). Статусный флаг **SBF** в регистре **PWR_CSR** показывает, что ЦПУ находился в режиме Ожидания.

Refer to Table 12 for more details on how to exit Standby mode.
См. табл. 12 для дополнительной информации о выходе из режима Ожидания.

Table 12. Standby mode (Режим Ожидания)

Standby mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: Ожидание прерывания или ожидание события - Set SLEEPDEEP in Cortex™-M3 System Control register Ставиться бит **SLEEPDEEP** в регистре **SCB_SCR** - Set PDDS bit in Power Control register (PWR_CR) Ставиться бит **PDDS** в регистре **PWR_CR** - Clear WUF bit in Power Control/Status register (PWR_CSR) Очищается бит **WUF** в регистре **PWR_CSR**
Mode exit	WKUP pin rising edge, RTC alarm, external Reset in NRST pin, IWDG Reset. По фронту на **WKUP**, от тревоги **RTC**, внешний сброс на **NRST**, сброс от **IWDG**.
Wakeup latency	Regulator start up. Reset phase Задержка: запуск регулятора на фазе сброса

I/O states in Standby mode (Состояние I/O в режиме Ожидания)

In Standby mode, all I/O pins are high impedance except:
В режиме Ожидания все **I/O** выводы находятся в высоко-импедансном состоянии, за исключением:

- Reset pad (still available) (Вход Сброса (все еще доступен))
- TAMPER pin if configured for tamper or calibration out
  Вывод **TAMPER**, если он сконфигурирован как вход сброса резервных данных или как выход калибровки.
- WKUP pin, if enabled (Вход пробуждения **WKUP**, если разрешен)

Debug mode (Режим Отладки)

By default, the debug connection is lost if the application puts the MCU in Stop or Standby mode while the debug features are used. This is due to the fact that the Cortex™-M3 core is no longer clocked.
По умолчанию, отладчик теряет подключение, если приложение помещает ЦПУ в режим Приостановки или Ожидания, а в тоже время особенности отладки используются. Это возникает вследствие того, что ядро **Cortex-M3** больше не тактируется.

However, by setting some configuration bits in the DBGMCU_CR register, the software can be debugged even when using the low-power modes extensively. For more details, refer to *Section 29.16.1: Debug support for low-power modes*.
Однако, если установить некоторые биты конфигурации в регистре **DBGMCU_CR**, то программное обеспечение, которое интенсивно использует режимы с низким потреблением, может быть отлажено. За дополнительной информацией обратитесь к Разделу 29.16.1 ***Debug support for low-power modes***.

4.3.6 Auto-wakeup (AWU) from low-power mode
Автопробуждение из режима низкого потребления

The RTC can be used to wakeup the MCU from low-power mode without depending on an external interrupt (Auto-wakeup mode). The RTC provides a programmable time base for waking up from Stop or Standby mode at regular intervals. For this purpose, two of the three alternative RTC clock sources can be selected by programming the RTCSEL[1:0] bits in the *Backup domain control register (RCC_BDCR)*:
Для пробуждения ЦПУ из режима с низким потреблением можно использовать **RTC**, это работает независимо от внешних прерываний (режим **Auto-wakeup**). **RTC** дает возможность программировать интервал времени для периодического пробуждения из режима Приостановки или Ожидания. Для этой цели можно выбрать два из трех альтернативных источников тактового сигнала для **RTC**, с помощью битов **RTCSEL[1:0]** регистра **RCC_BDCR**:

- Low-power 32.768 kHz external crystal oscillator (LSE OSC).

This clock source provides a precise time base with very low-power consumption (less than 1µA added consumption in typical conditions)
Генератор с низким потреблением от внешнего кварца на 32.768 кГц (**LSE OSC**). Этот источник предоставляет стабильный тактовый сигнал при очень низком потреблении (в типичных условиях он потребляет не более 1µA)

- Low-power internal RC Oscillator (LSI RC).

This clock source has the advantage of saving the cost of the 32.768 kHz crystal. This internal RC Oscillator is designed to add minimum power consumption.
Внутренний **RC** генератор с низким потреблением (**LSI RC**).
Этот источник тактового сигнала имеет то преимущество, что можно сэкономить на стоимости кварца на 32.768 кГц. Этот внутренний **RC** Генератор спроектирован так, чтобы потреблять минимальную мощность.

To wakeup from Stop mode with an RTC alarm event, it is necessary to:
Чтобы пробудить из режима Приостановки с помощью события тревоги **RTC**, необходимо следующее:

- Configure the EXTI Line 17 to be sensitive to rising edge.

Сконфигурируйте линию 17 **EXTI**, чтобы она срабатывала по фронту сигнала.

- Configure the RTC to generate the RTC alarm.

Сконфигурируйте **RTC** на генерацию сигнала тревоги.

To wakeup from Standby mode, there is no need to configure the EXTI Line 17.
Чтобы пробудить из режима Ожидания нет необходимости конфигурировать линию 17 **EXTI**.

4.4 Power control registers (Регистры управления питанием)

4.4.1 Power control register (PWR_CR) (Регистр управления питанием)
4.4.2 Power control/status register (PWR_CSR) (Регистр управления/статуса **PWR**)
4.4.3 PWR register map (Карта регистров **PWR**)

4.4.1 Power control register (PWR_CR) (Регистр управления питанием)

Address offset: 0x00
Reset value: 0x0000 0000 (reset by wakeup from Standby mode)
(сбрасывается при пробуждении из режима Ожидания)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
							rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw

Bits 31:9 Reserved
always read as 0. (Всегда читается как '0'.)

Bit 8 DBP: Disable backup domain write protection.
Отключение защиты от записи модуля BKP.
In reset state, the RTC and backup registers are protected against parasitic write access. This bit must be set to enable write access to these registers.
После сброса регистры **RTC** и **BKP** защищены от случайного доступа на запись. Чтобы разрешить запись в эти регистры нужно установить этот бит.
0: Access to RTC and Backup registers disabled
Доступ к регистрам **RTC** и **BKP** запрещен.
1: Access to RTC and Backup registers enabled
Доступ к регистрам **RTC** и **BKP** разрешен.

Note: If the HSE divided by 128 is used as the RTC clock, this bit must remain set to 1.
Если для тактового **RTC** используется сигнал от **HSE**, поделенный на 128, этот бит всегда должен быть установлен в '1'.

Bits 7:5 PLS[2:0]: PVD level selection. (Выбор уровня порога срабатывания.)
These bits are written by software to select the voltage threshold detected by the Power Voltage Detector Этот бит ставиться программно, чтобы выбрать порог срабатывания цепи **PVD** (*Power Voltage Detector*)
000: 2.2V
001: 2.3V
010: 2.4V
011: 2.5V
100: 2.6V
101: 2.7V
110: 2.8V
111: 2.9V
Note: Refer to the electrical characteristics of the datasheet for more details.
Для дополнительной информации см. раздел "Электрические характеристики" в **datasheet**.

Bit 4 PVDE: Power voltage detector enable. (Разрешение цепи PVD.)

This bit is set and cleared by software. (Этот бит ставиться и очищается программно.)

0: PVD disabled (цепь отключена)

1: PVD enabled (цепь включена)

Bit 3 CSBF: Clear standby flag. (Очистка флага нахождения в режиме Ожидания)

This bit is always read as 0. (Этот бит всегда читается как '0'.)

0: No effect (нет эффекта)

1: Clear the SBF Standby Flag (write). (При записи очищается флаг **CSBF**)

Bit 2 CWUF: Clear wakeup flag. (Очистка флага пробуждения)

This bit is always read as 0. (Этот бит всегда читается как '0'.)

0: No effect (нет эффекта)

1: Clear the WUF Wakeup Flag after 2 System clock cycles. (write)

При записи очищается флаг **CWUF** через два цикла тактового.

Bit 1 PDDS: Power down deepsleep. (Переход в режим глубокого сна.)

This bit is set and cleared by software. It works together with the LPDS bit.

Этот бит ставиться и очищается программно. Он работает совместно с битом **LPDS**.

0: Enter Stop mode when the CPU enters Deepsleep.

The regulator status depends on the LPDS bit.

Вход в режим Приостановки, когда ЦПУ входит в режим глубокого сна.

Состояние регулятора зависит от бита **LPDS**.

1: Enter Standby mode when the CPU enters Deepsleep.

Вход в режим Ожидания, когда ЦПУ входит в режим глубокого сна.

Bit 0 LPDS: Low-power deepsleep. (Режим глубокого сна с малым потреблением.)

This bit is set and cleared by software. It works together with the PDDS bit.

Этот бит ставиться и очищается программно. Он работает совместно с битом **PDDS**.

0: Voltage regulator on during Stop mode

В режиме Приостановки регулятор включен

1: Voltage regulator in low-power mode during Stop mode

В режиме Приостановки регулятор находится в режиме малого потребления

4.4.2 Power control/status register (PWR_CSR) (Регистр управления/статуса PWR)

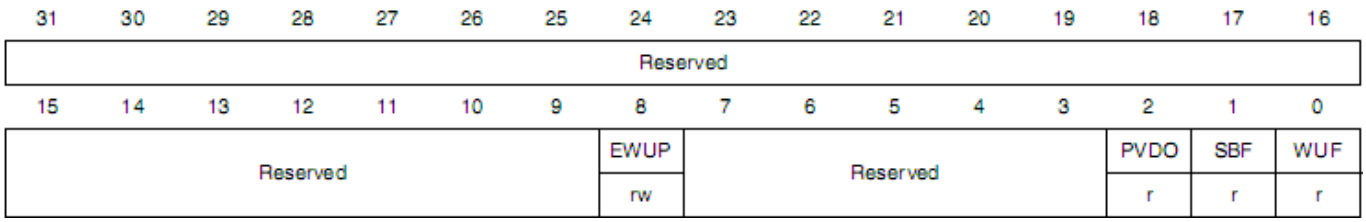
Address offset: 0x04

Reset value: 0x0000 0000 (not reset by wakeup from Standby mode)

(не сбрасывается при пробуждении из режима Ожидания)

Additional APB cycles are needed to read this register versus a standard APB read.

Чтобы прочитать этот регистр необходимы дополнительные циклы шины **APB**.



Bits 31:9 Reserved

always read as 0. (Всегда читается как '0'.)

Bit 8 EWUP: Enable WKUP pin (Разрешение вывода WKUP.)

This bit is set and cleared by software. (Этот бит ставиться и очищается программно.)

0: WKUP pin is used for general purpose I/O. An event on the WKUP pin does not wakeup the device from Standby mode.

Вывод **WKUP** используется как вывод общего назначения. Событие на этом выводе не пробуждает устройство из режима Ожидания.

1: WKUP pin is used for wakeup from Standby mode and forced in input pull down configuration (rising edge on WKUP pin wakes-up the system from Standby mode).

Вывод **WKUP** используется для пробуждения из режима Ожидания и принудительно ставится в режим входа притянутого к земле.

Note: This bit is reset by a system Reset. (Этот бит сбрасывается общим сбросом.)

Bits 7:3 Reserved

always read as 0. (Всегда читается как '0'.)

Bit 2 PVDO: PVD output (Выход детектора PVD.)

This bit is set and cleared by hardware. It is valid only if PVD is enabled by the PVDE bit.

Этот бит ставиться и очищается программно. Он работает только при разрешенном **PVD** битом **PVDE**.

0: V_{DD}/V_{DDA} is higher than the PVD threshold selected with the PLS[2:0] bits.

Напряжение на $\mathbf{V_{DD}/V_{DDA}}$ выше, чем порог, выбранный битами **PLS[2:0]**.

1: V_{DD}/V_{DDA} is lower than the PVD threshold selected with the PLS[2:0] bits.

Напряжение на $\mathbf{V_{DD}/V_{DDA}}$ ниже, чем порог, выбранный битами **PLS[2:0]**.

Note: The PVD is stopped by Standby mode. For this reason, this bit is equal to 0 after Standby or reset until the PVDE bit is set.

Работа детектора **PVD** останавливается в режиме Ожидания. По этой причине этот бит равен '0' после выхода из режима Ожидания или после сброса и остается в этом состоянии, пока не будет поставлен бит **PVDE**.

Bit 1 SBF: Standby flag (Флаг нахождения в режиме Ожидания.)

This bit is set by hardware and cleared only by a POR/PDR (power on reset/power down reset) or by setting the CSBF bit in the Power control register (PWR_CR)

Этот бит ставится аппаратно и очищается только сбросом от подачи/снятия питания или установкой бита **CSBF** в регистре **PWR_CR**.

0: Device has not been in Standby mode (Устройство не было в режиме Ожидания)

1: Device has been in Standby mode (Устройство было в режиме Ожидания)

Bit 0 WUF: Wakeup flag (Флаг события пробуждения.)

This bit is set by hardware and cleared only by a POR/PDR (power on reset/power down reset) or by setting the CWUF bit in the Power control register (PWR_CR)

Этот бит ставится аппаратно и очищается только сбросом от подачи/снятия питания или установкой бита **CWUF** в регистре **PWR_CR**.

0: No wakeup event occurred (Не было события пробуждения)

1: A wakeup event was received from the WKUP pin or from the RTC alarm

Было получено событие пробуждения на выводе **WKUP** или от сигнала тревоги **RTC**.

Note: An additional wakeup event is detected if the WKUP pin is enabled (by setting the EWUP bit) when the WKUP pin level is already high.

Note: Детектируется дополнительное событие пробуждения, если разрешается вывод **WKUP** (установкой бита **EWUP**) в то время, как уже есть высокий уровень на этом выводе.

4.4.3 PWR register map (Карта регистров PWR)

The following table summarizes the PWR registers.
Последующая таблица дает карту регистров модуля **PWR**.

Table 13. PWR register map and reset values
Карта регистров **PWR** и их значение после сброса

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PWR_CR	Reserved																							DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	Reset value																								0	0	0	0	0	0	0	0	0
0x004	PWR_CSR	Reserved																							EWUP	Reserved					PVDO	SBF	WUF
	Reset value																								0						0	0	0

Refer to Table 1 on page 41 for the register boundary addresses.
См. Табл. 1, где даны относительные адреса регистров.

5 Backup registers (BKP) (Резервные регистры)

(*Здесь находится ритуальное напоминание о наличии 4-х семейств для **STM32F10xxx**, и что этот документ обо всех семействах, если нет прямого указания. См. раздел 1.2 Глоссарий).*)

5.1 BKP introduction (Введение в **BKP**)
5.2 BKP main features (Основные особенности **BKP**)
5.3 BKP functional description (Функциональное описание **BKP**)
5.4 BKP registers (Регистры **BKP**)

5.1 BKP introduction (Введение в BKP)

The backup registers are forty two 16-bit registers for storing 84 bytes of user application data. They are implemented in the backup domain that remains powered on by VBAT when the VDD power is switched off. They are not reset when the device wakes up from Standby mode or by a system reset or power reset.
Регистры резервных данных **BKP** - это сорок два 16-ти разрядных регистра для хранения 84-х байт пользовательских данных. Они реализованы в домене **BKP**, который остается подключенным к выводу **VBAT**, когда питание на выводе **VDD** снимается. Они не сбрасываются при пробуждении устройства из режима **Standby**, системным сбросом или сбросом от подачи питания.

In addition, the BKP control registers are used to manage the Tamper detection feature and RTC calibration.
Кроме того, регистры управления доменом **BKP** используются также, чтобы управлять функцией обнаружения **Tamper** (*сброс резервных данных*) и калибровкой часов реального времени **RTC**.

After reset, access to the Backup registers and RTC is disabled and the Backup domain (BKP) is protected against possible parasitic write access. To enable access to the Backup registers and the RTC, proceed as follows:
После сброса, доступ к регистрам резервных данных и **RTC** заблокирован, и домен **BKP** защищен от возможного случайного доступа на запись. Чтобы разрешить доступ к **BKP** и **RTC** надо выполнить следующую процедуру:

- enable the power and backup interface clocks by setting the PWREN and BKPEN bits in the RCC_APB1ENR register
разрешите подачу тактового сигнала на блок управления питанием и домен **BKP** с помощью установки битов **PWREN** и **BKPEN** в регистре **RCC_APB1ENR**
- set the DBP bit the Power Control Register (PWR_CR) to enable access to the Backup registers and RTC.
поставьте бит **DPB** в регистре Управления потреблением **PWR_CR**, чтобы разрешить доступ к **BKP** и **RTC**.

5.2 BKP main features (Основные особенности BKP)

- 20-byte data registers (in medium-density and low-density devices) or 84-byte data registers (in high-density and connectivity line devices)
Регистры резервных данных объемом 20 байт (в семействах **MD** и **LD**) или объемом 84 байт (в семействах **HD** и **LC**)

- Status/control register for managing tamper detection with interrupt capability
Регистр состояния/управления для управления функцией обнаружения **Tamper** с возможностью прерывания

- Calibration register for storing the RTC calibration value
Регистр для хранения значения калибровки **RTC**

- Possibility to output the RTC Calibration Clock, RTC Alarm pulse or Second pulse on TAMPER pin PC13 (when this pin is not used for tamper detection)
Возможность вывести Калиброванный тактовый сигнал **RTC**, импульс "Тревоги" **RTC** или Вторичный импульс **RTC** на вывод **TAMPER** (**PC13**) (когда этот вывод не используется для обнаружения **Tamper**)

5.3 BKP functional description (Функциональное описание BKP)

5.3.1 Tamper detection (Детектирование события Tamper)

The TAMPER pin generates a Tamper detection event when the pin changes from 0 to 1 or from 1 to 0 depending on the TPAL bit in the Backup control register (BKP_CR). A tamper detection event resets all data backup registers. Вывод **TAMPER** генерирует событие обнаружения **Tamper** при изменении уровня на выводе с 0 до 1 или с 1 до 0, в зависимости от уставки бита **TPAL** в регистре **BKP_CR**. Событие обнаружения **Tamper** сбрасывает все регистры блока **BKP**.

However to avoid losing Tamper events, the signal used for edge detection is logically ANDed with the Tamper enable in order to detect a Tamper event in case it occurs before the TAMPER pin is enabled. Однако, чтобы избежать потери события **Tamper**, сигнал, используемый для обнаружения перепада, объединяется логически (**AND**) с разрешением **Tamper**, чтобы обнаружить событие **Tamper** в случае, когда оно происходит прежде, чем разрешен вывод **TAMPER**.

- When TPAL=0: If the TAMPER pin is already high before it is enabled (by setting TPE bit), an extra Tamper event is detected as soon as the TAMPER pin is enabled (while there was no rising edge on the TAMPER pin after TPE was set)
Когда **TPAL=0**: Если на выводе **TAMPER** уже присутствовал высокий уровень прежде, чем он был разрешен (установкой бита **TPE**), то детектируется дополнительное событие **Tamper**, как только разрешается вывод **TAMPER** (хотя не было никакого фронта сигнала на выводе **TAMPER** после того, как был установлен **TPE**)

- When TPAL=1: If the TAMPER pin is already low before it is enabled (by setting the TPE bit), an extra Tamper event is detected as soon as the TAMPER pin is enabled (while there was no falling edge on the TAMPER pin after TPE was set)
Когда **TPAL=1**: Если на выводе **TAMPER** уже присутствовал низкий уровень прежде, чем он будет разрешен (установкой бита **TPE**), то детектируется дополнительное событие **Tamper**, как только разрешается вывод **TAMPER** (хотя не было никакого среза сигнала на выводе **TAMPER** после того, как был установлен **TPE**)

By setting the TPIE bit in the BKP_CSR register, an interrupt is generated when a Tamper detection event occurs. Если устанавлен бит **TPIE** в регистре **BKP_CSR**, то, в случае обнаружения события **Tamper**, будет сгенерировано прерывание.

After a Tamper event has been detected and cleared, the TAMPER pin should be disabled and then re-enabled with TPE before writing to the backup data registers (BKP_DRx) again. This prevents software from writing to the backup data registers (BKP_DRx), while the TAMPER pin value still indicates a Tamper detection. This is equivalent to a level detection on the TAMPER pin.
После того, как событие **Tamper** было обнаружено и очищено, вывод **TAMPER** должен быть отключен, а затем вновь разрешен с помощью **TPE** прежде, чем снова что-то писать в регистры резервных данных (**BKP_DRx**). Это не дает программе что-то писать в регистры резервных данных в то время, как значение на входе **TAMPER** все еще показывает обнаружение события **Tamper**. Это эквивалентно наличию детектора уровня на входе **TAMPER**.

***Note**: Tamper detection is still active when VDD power is switched off. To avoid unwanted resetting of the data backup registers, the TAMPER pin should be externally tied to the correct level.*
***Note**: Детектирование **Tamper** остается активным, когда питание на выводе **VDD** снято. Чтобы избежать нежелательного сброса регистров резервных данных, вход **TAMPER** должен быть внешне подключен к правильному уровню.*

5.3.2 RTC calibration (Калибровка RTC)

For measurement purposes, the RTC clock with a frequency divided by 64 can be output on the TAMPER pin. This is enabled by setting the CCO bit in the *RTC clock calibration register (BKP_RTCCR)*.
Частота тактового сигнала **RTC**, поделенная на 64, может быть выведена на вывод **TAMPER**, в целях ее измерения. Это разрешается устанавкой бита **CCO** в регистре калибровки **RTC BKP_RTCCR**.

The clock can be slowed down by up to 121 ppm by configuring CAL[6:0] bits.
Частоту тактового сигнала можно уменьшить на величину до 121 ppm (*миллионной доли*), конфигурируя биты **CAL[6:0]**.

For more details about RTC calibration and how to use it to improve timekeeping accuracy, please refer to AN2604 "*STM32F101xx and STM32F103xx RTC calibration*".
Ддля дополнительной информации о калибровке **RTC** и как ее использовать для улучшения точности хронометрирования, пожалуйста обратитесь к документу **AN2604** "***STM32F101xx and STM32F103xx RTC calibration***".

5.4 BKP registers (Регистры BKP)

Refer to *Section 1.1 on page 37* for a list of abbreviations used in register descriptions.
См. раздел 1.1, где приведен перечень сокращений, используемых при описании регистров.

5.4.1 Backup data register x (BKP_DRx) (x = 1 ..42) (Регистр **x** резервных данных)
5.4.2 RTC clock calibration register (BKP_RTCCR) (Регистр калибровки **RTC**)
5.4.3 Backup control register (BKP_CR) (Регистр управления блоком **BKP**)
5.4.4 Backup control/status register (BKP_CSR)
(Регистр статуса/управления блока **BKP**)
5.4.5 BKP register map (Карта регистров **BKP**)

5.4.1 Backup data register x (BKP_DRx) (x = 1 ..42) (Регистр x резервных данных)

Address offset: 0x04 to 0x28, 0x40 to 0xBC
Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 D[15:0] Backup data (Резервные данные)
These bits can be written with user data.
В эти биты могут быть записаны пользовательские данные.

Note: The BKP_DRx registers are not reset by a System reset or Power reset or when the device wakes up from Standby mode. They are reset by a Backup Domain reset or by a TAMPER pin event (if the TAMPER pin function is activated).
Note: Регистры **BKP_DRx** не сбрасываются системным сбросом, или сбросом при подаче питания, или при пробуждении устройства из режима **Standby**. Они сбрасываются сбросом домена **BKP** или событием на входе **TAMPER** (если эта функция активирована).

5.4.2 RTC clock calibration register (BKP_RTCCR) (Регистр калибровки RTC)

Address offset: 0x2C
Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						ASOS	ASOE	CCO	CAL[6:0]						
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:10 Reserved
always read as 0. (Всегда читается как '0'.)

Bit 9 ASOS: Alarm or second output selection
Выбор выхода как "Тревога" или "Вторичный сигнал".

When the ASOE bit is set, the ASOS bit can be used to select whether the signal output on the TAMPER pin is the RTC Second pulse signal or the Alarm pulse signal:
Когда установлен бит **ASOE**, то можно использовать бит **ASOS**, чтобы выбрать, является ли выходной импульс на выводе **TAMPER** "Вторичным сигналом" **RTC** или сигналом "Тревоги" **RTC**:

- **0:** RTC Alarm pulse output selected (Выбран сигнал "Тревоги" **RTC**)
- **1:** RTC Second pulse output selected (Выбран "Вторичный сигнал" **RTC**)

Note: This bit is reset only by a Backup domain reset.
Этот бит сбрасывается только сбросом домена **BKP**.

Bit 8 ASOE: Alarm or second output enable
Разрешение выхода для "Тревоги" или "Вторичного сигнала"

Setting this bit outputs either the RTC Alarm pulse signal or the Second pulse signal on the TAMPER pin depending on the ASOS bit. The output pulse duration is one RTC clock period. The TAMPER pin must not be enabled while the ASOE bit is set.
Установка этого бита разрешает вывод импульсного сигнала "Тревоги" или "Вторичного сигнала" **RTC** на вывод **TAMPER**, в зависимости от бита **ASOS**.
Продолжительность импульсного сигнала - один период тактового **RTC**. В процессе установки бита **ASOE** вывод **TAMPER** должен быть отключен.

Note: This bit is reset only by a Backup domain reset.
Этот бит сбрасывается только сбросом домена **BKP**.

Bit 7 CCO: Calibration clock output (Выход калиброванного тактового)

- **0:** No effect (Нет эффекта)
- **1:** Setting this bit outputs the RTC clock with a frequency divided by 64 on the TAMPER pin.

The TAMPER pin must not be enabled while the CCO bit is set in order to avoid unwanted Tamper detection. Установка этого бита выводит тактовый сигнал **RTC**, поделенной на 64, на вывод **TAMPER**. В процессе установки бита **CCO** вывод **TAMPER** должен быть разрешен, чтобы избежать нежелательного события обнаружения **Tamper**.

Note: This bit is reset when the VDD supply is powered off.
Этот бит сбрасывается при снятии питания с вывода **VDD**.

Bit 6:0 CAL[6:0]: Calibration value (Значение калибровки)

This value indicates the number of clock pulses that will be ignored every 2^20 clock pulses. This allows the calibration of the RTC, slowing down the clock by steps of 1000000/2^20 PPM. The clock of the RTC can be slowed down from 0 to 121 PPM.
Это значение указывает число тактовых импульсов **RTC**, которые будут проигнорированы каждые 2^20 импульсов. Это позволяет калибровать **RTC**, замедляя часы с шагом 1000000/2^20 ppm. Тактовый сигнал **RTC** может быть замедлен от 0 до 121 ppm (*миллионной доли*).

5.4.3 Backup control register (BKP_CR) (Регистр управления блоком BKP)

Address offset: 0x30
Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														TPAL	TPE
														rw	rw

Bits 15:2 Reserved
always read as 0. (Всегда читается как '0'.)

Bit 1 TPAL: TAMPER pin active level (Активный уровень вывода TAMPER)
0: A high level on the TAMPER pin resets all data backup registers (if TPE bit is set).
Высокий уровень на вывода **TAMPER** сбрасывает все регистры резервных данных (если установлен бит **TPE**).
1: A low level on the TAMPER pin resets all data backup registers (if TPE bit is set).
Низкий уровень на вывода **TAMPER** сбрасывает все регистры резервных данных (если установлен бит **TPE**).

Bit 0 TPE: TAMPER pin enable (Разрешение вывода TAMPER)
0: The TAMPER pin is free for general purpose I/O
Вывод **TAMPER** свободен для использования как **I/O** общего назначения
1: Tamper alternate I/O function is activated.
Активирована альтернативная функция вывода **TAMPER**

Note: Setting the TPAL and TPE bits at the same time is always safe, however resetting both at the same time can generate a spurious Tamper event. For this reason it is recommended to change the TPAL bit only when the TPE bit is reset.
Note: Одновременная установка битов **TPAL** и **TPE** всегда безопасна, однако, одновременный сброс этих битов может генерировать ложное событие **Tamper**. По этой причине рекомендуется изменять бит **TPAL** только при сброшенном бите **TPE**.

5.4.4 Backup control/status register (BKP_CSR)
Регистр статуса/управления блока BKP

Address offset: 0x34
Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TIF	TEF	Reserved					TPIE	CTI	CTE
						r	r						rw	w	w

Bits 15:10 Reserved
always read as 0. (Всегда читается как '0'.)

Bit 9 TIF: Tamper interrupt flag (Флаг прерывания от Tamper)

This bit is set by hardware when a Tamper event is detected and the TPIE bit is set. It is cleared by writing 1 to the CTI bit (also clears the interrupt). It is also cleared if the TPIE bit is reset.
Этот бит ставится аппаратно, когда обнаружено событие **Tamper** и установлен бит **TPIE**. Он очищается при записи '1' в бит **CTI** (это также очищает прерывание). Он также очищается при сбросе бита **TPIE**.

0: No Tamper interrupt (Нет прерывания от **Tamper**)
1: A Tamper interrupt occurred (Случилось прерывание от **Tamper**)

Note: This bit is reset only by a system reset and wakeup from Standby mode.
Этот бит сбрасывается только системным сбросом и пробуждением из Режима **Standby**.

Bit 8 TEF: Tamper event flag (Флаг события Tamper)

This bit is set by hardware when a Tamper event is detected.
It is cleared by writing 1 to the CTE bit.
Этот бит ставится аппаратно, когда обнаружено событие **Tamper**.
Он очищается при записи '1' в бит **CTE**.

0: No Tamper event (Нет события **Tamper**)
1: A Tamper event occurred (Случилось событие **Tamper**)

Note: A Tamper event resets all the BKP_DRx registers. They are held in reset as long as the TEF bit is set. If a write to the BKP_DRx registers is performed while this bit is set, the value will not be stored.
Note: Событие **Tamper** сбрасывает все регистры **BKP_DRx**. Они удерживаются в состоянии сброса, пока установлен бит **TEF**. Если будет выполненяться запись в регистры **BKP_DRx**, пока этот бит установлен, то записанное значение не будет сохранено.

Bits 7:3 Reserved

always read as 0. (Всегда читается как '0'.)

Bit 2 TPIE: TAMPER pin interrupt enable
Разрешение прерывания от вывода TAMPER

0: Tamper interrupt disabled (Прерывание от **Tamper** отключено)
1: Tamper interrupt enabled (the TPE bit must also be set in the BKP_CR register)

Прерывание от **Tamper** разрешено (также должен быть установлен бит **TPE** в регистре **BKP_CR**)

Note:

1. A Tamper interrupt does not wake up the core from low-power modes.
   Прерывание от **Tamper** не пробуждает ядро из режима с низким потреблением.
2. This bit is reset only by a system reset and wakeup from Standby mode.
   Этот бит сбрасывается только системным сбросом и пробуждением из Режима **Standby**.

Bit 1 CTI: Clear tamper interrupt (Очищение прорывания от Tamper)

This bit is write only, and is always read as 0.
Это бит только для записи, при его чтении всегда получаем '0'.

0: No effect (Нет эффекта)
1: Clear the Tamper interrupt and the TIF Tamper interrupt flag.

Очищает прорывание от **Tamper** и его флаг **TIF**.

Bit 0 CTE: Clear tamper event
Очищение события Tamper

This bit is write only, and is always read as 0.
Это бит только для записи, при его чтении всегда получаем '0'.

0: No effect (Нет эффекта)
1: Reset the TEF Tamper event flag (and the Tamper detector)

Сбрасывает флаг события **Tamper - TEF** (и детектор **Tamper**)

5.4.5 BKP register map (Карта регистров BKP)

BKP registers are mapped as 16-bit addressable registers as described in the table below:
Регистры **BKP** отображены как 16-разрядно адресуемые регистры, как описано в таблице ниже:

Table 14. BKP register map and reset values (Карта регистров **BKP** и их значение после сброса)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	Reserved																																
0x04	BKP_DR1	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	BKP_DR2	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	BKP_DR3	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	BKP_DR4	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	BKP_DR5	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	BKP_DR6	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	**BKP_DR7**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**BKP_DR8**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**BKP_DR9**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**BKP_DR10**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2	**BKP_RTCCR**	Reserved																						ASOS	ASOE	CCO	CAL[6:0]						
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x30	**BKP_CR**	Reserved																														TPAL	TPE
	Reset value																															0	0
0x34	**BKP_CSR**	Reserved																						TIF	TEF	Reserved					TPIE	CTI	CTE
	Reset value																							0	0						0	0	0
0x38	Reserved																																
0x3C	Reserved																																
0x40	**BKP_DR11**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**BKP_DR12**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**BKP_DR13**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	**BKP_DR14**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**BKP_DR15**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	**BKP_DR16**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58	**BKP_DR17**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	**BKP_DR18**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	**BKP_DR19**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x64	**BKP_DR20**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	**BKP_DR21**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6C	**BKP_DR22**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70	**BKP_DR23**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	**BKP_DR24**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x78	**BKP_DR25**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7C	**BKP_DR26**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x80	**BKP_DR27**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	**BKP_DR28**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	**BKP_DR29**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	**BKP_DR30**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90	**BKP_DR31**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x94	**BKP_DR32**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98	**BKP_DR33**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9C	**BKP_DR34**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA0	**BKP_DR35**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4	**BKP_DR36**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA8	**BKP_DR37**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xAC	**BKP_DR38**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB0	**BKP_DR39**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB4	**BKP_DR40**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB8	**BKP_DR41**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBC	**BKP_DR42**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Table 1 on page 41* for the register boundary addresses.
См. Табл. 1, где даны относительные адреса регистров.

7 Connectivity line devices: reset and clock control (RCC)
Семейство CL: управление сбросом и тактовыми

(*Здесь находится ритуальное напоминание о наличии 4-х семейств для **STM32F10xxx**, и что этот документ обо всех семействах, если нет прямого указания. См. раздел* 1.2 Глоссарий*).*)

7.1 Reset (Сброс)
7.2 Clocks (Тактовые сигналы)
7.3 RCC registers (Регистры **RCC**)

7.1 Reset (Сброс)

7.1.1 System reset (Системный сброс)
7.1.2 Power reset (Сброс при подаче питания)
7.1.3 Backup domain reset (Сброс домена **BKP**)

There are three types of reset, defined as system Reset, power Reset and backup domain Reset.
Есть три типа сброса, определенных как "Системный сброс", "Сброс при подаче питания" и "Сброс домена **BKP**".

7.1.1 System reset (Системный сброс)

A system reset sets all registers to their reset values except the reset flags in the clock controller CSR register and the registers in the Backup domain (see *Figure 4*).
"Системный сброс" устанавливает все регистры в их значения после сброса, кроме флагов сброса в регистре управления тактовыми **CSR** и регистров в домене **BKP** (см. рис. 4).

A system reset is generated when one of the following events occurs:
"Системный сброс" генерируется, когда происходит одно из следующих событий:

1. A low level on the NRST pin (external reset)
Низкий уровень на выводе **NRST** (внешний сброс)

2. Window watchdog end of count condition (WWDG reset)
Условие окончания счета оконного **WatchDog** (сброс от **WWDG**)

3. Independent watchdog end of count condition (IWDG reset)
Условие окончания счета независимого **WatchDog** (сброс от **IWDG**)

4. A software reset (SW reset) (see *Section : Software reset*)
Программный сброс (сброс **SW**) (см. раздел "Программный сброс")

5. Low-power management reset (see *Section : Low-power management reset*)
Сброс от контроллера пониженного потребления (см. раздел "Сброс от контроллера пониженного потребления")

The reset source can be identified by checking the reset flags in the Control/Status register, RCC_CSR (see *Section 7.3.10: Control/status register (RCC_CSR)*).
Источник сброса может быть идентифицирован проверкой флагов сброса в регистре Управления/статуса **RCC_CSR** (см. раздел 7.3.10 "Регистр управления/статуса (**RCC_CSR**)")

Software reset (Программный сброс)

The SYSRESETREQ bit in Cortex™-M3 Application Interrupt and Reset Control Register must be set to force a software reset on the device. Refer to the Cortex™-M3 technical reference manual for more details. Чтобы вызвать программный сброс устройства надо поставить бит **SYSRESETREQ** в регистре **SCB_AIRCR**. Обратитесь к документу "***Cortex-M3 technical reference manual***" за более детальной информацией.

Low-power management reset (Сброс от контроллера пониженного потребления)

There are two ways to generate a low-power management reset:
Есть два способа генерировать сброс от контроллера пониженного потребления:

1. Reset generated when entering Standby mode: (Генерация сброса при входе в режим **Standby**:)
   This type of reset is enabled by resetting nRST_STDBY bit in User Option Bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode. Этот тип сброса разрешается сбросом бита **nRST_STDBY** в "**User Option Bytes**" (*Опционные Пользовательские Байты*). В этом случае, всякий раз, когда успешно выполняется последовательность входа в режим **Standby**, вместо того, чтобы войти в этот режим, устройство будет сброшено.
2. Reset when entering Stop mode: (Генерация сброса при входе в режим **Stop**:)
   This type of reset is enabled by resetting NRST_STOP bit in User Option Bytes. In this case, whenever a Stop mode entry sequence is successfully executed, the device is reset instead of entering Stop mode. Этот тип сброса разрешается сбросом бита **NRST_STOP** в "**User Option Bytes**". В этом случае, всякий раз, когда успешно выполняется последовательность входа в режим **Stop**, вместо того, чтобы войти в этот режим, устройство будет сброшено.

For further information on the User Option Bytes, refer to the STM32F10xxx Flash programming manual. Для дополнительной информации о "**User Option Bytes**", обратитесь к документу "***STM32F10xxx Flash programming manual***".

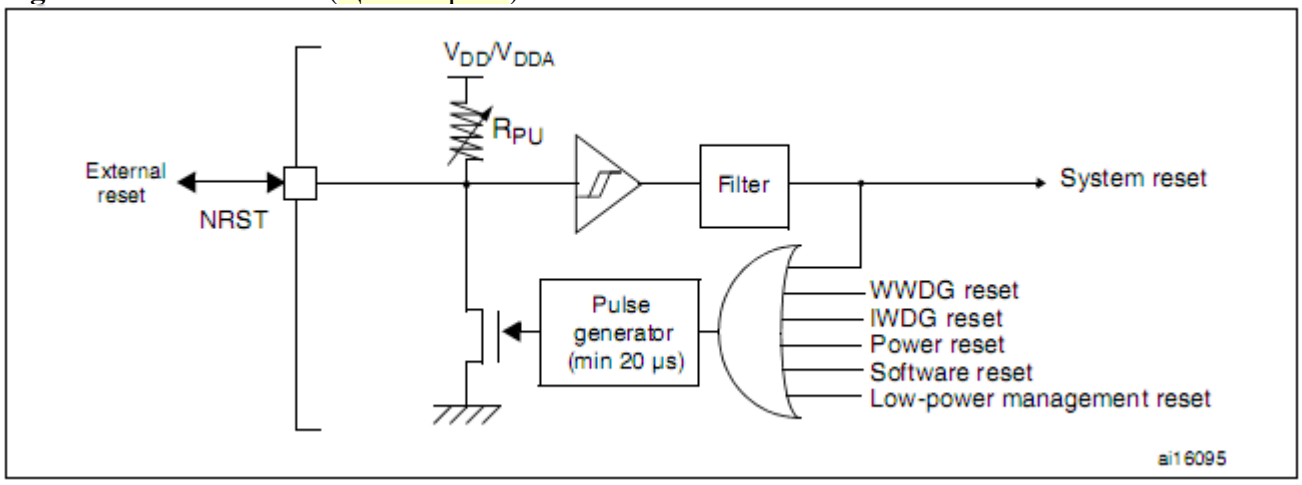
7.1.2 Power reset (Сброс при подаче питания)

A power reset is generated when one of the following events occurs:
"Сброс при подаче питания" генерируется, когда случается одно из следующих событий:

1. Power-on/power-down reset (POR/PDR reset)
   При "Подаче/Снятии питания" (сброс от **POR/PDR**)
2. When exiting Standby mode (При выходе из режима **Standby**)

A power reset sets all registers to their reset values except the Backup domain (see *Figure 4*) These sources act on the NRST pin and it is always kept low during the delay phase. The RESET service routine vector is fixed at address 0x0000_0004 in the memory map. For more details, refer to *Table 53: Vector table for other STM32F10xxx devices on page 172*.
Сброс при подаче питания устанавливает все регистры в их значения после сброса, кроме регистров домена **BKP** (см. рис. 4). Эти источники воздействуют на вывод **NRST** и удерживают его в низком состоянии во время фазы задержки. Вектор обработчика **RESET** фиксирован по адресу **0x0000_0004** в карте памяти. За дополнительной информацией обратитесь к Таблице 53: "***Vector table for other STM32F10xxx devices***".

Figure 10. Reset circuit (Цепи сброса)



7.1.3 Backup domain reset (Сброс домена BKP)

The backup domain has two specific resets that affect only the backup domain (see *Figure 4*).
Домен **BKP** имеет два специфичных сброса, которые воздействуют только на этот домен (см. рис. 4).

A backup domain reset is generated when one of the following events occurs:
Сброс домена **BKP** генерируется, когда случается одно из следующих событий:

1. Software reset, triggered by setting the BDRST bit in the *Backup domain control register (RCC_BDCR)*.
Программный сброс, вызванный установкой бита **BDRST** в регистре **RCC_BDCR**.
2. VDD or VBAT power on, if both supplies have previously been powered off.
Подача питания на вывод **VDD** или **VBAT**, если перед этим на обоих выводах не было питания.

7.2 Clocks (Тактовые сигналы)

7.2.1 HSE clock (Тактовый от **HSE**)
7.2.2 HSI clock (Тактовый от **HSI**)
7.2.3 PLLs
7.2.4 LSE clock (Тактовый от **LSE**)
7.2.5 LSI clock (Тактовый от **LSI**)
7.2.6 System clock (SYSCLK) selection (Выбор тактового системы)
7.2.7 Clock security system (CSS) (Система безопасности тактового сигнала)
7.2.8 RTC clock (Тактовый для **RTC**)
7.2.9 Watchdog clock (Тактовый для **Watchdog**)
7.2.10 Clock-out capability (Функция вывода тактового)

Three different clock sources can be used to drive the system clock (SYSCLK):
Для управления системным тактовым сигналом (**SYSCLK**) могут использоваться три различных источника:

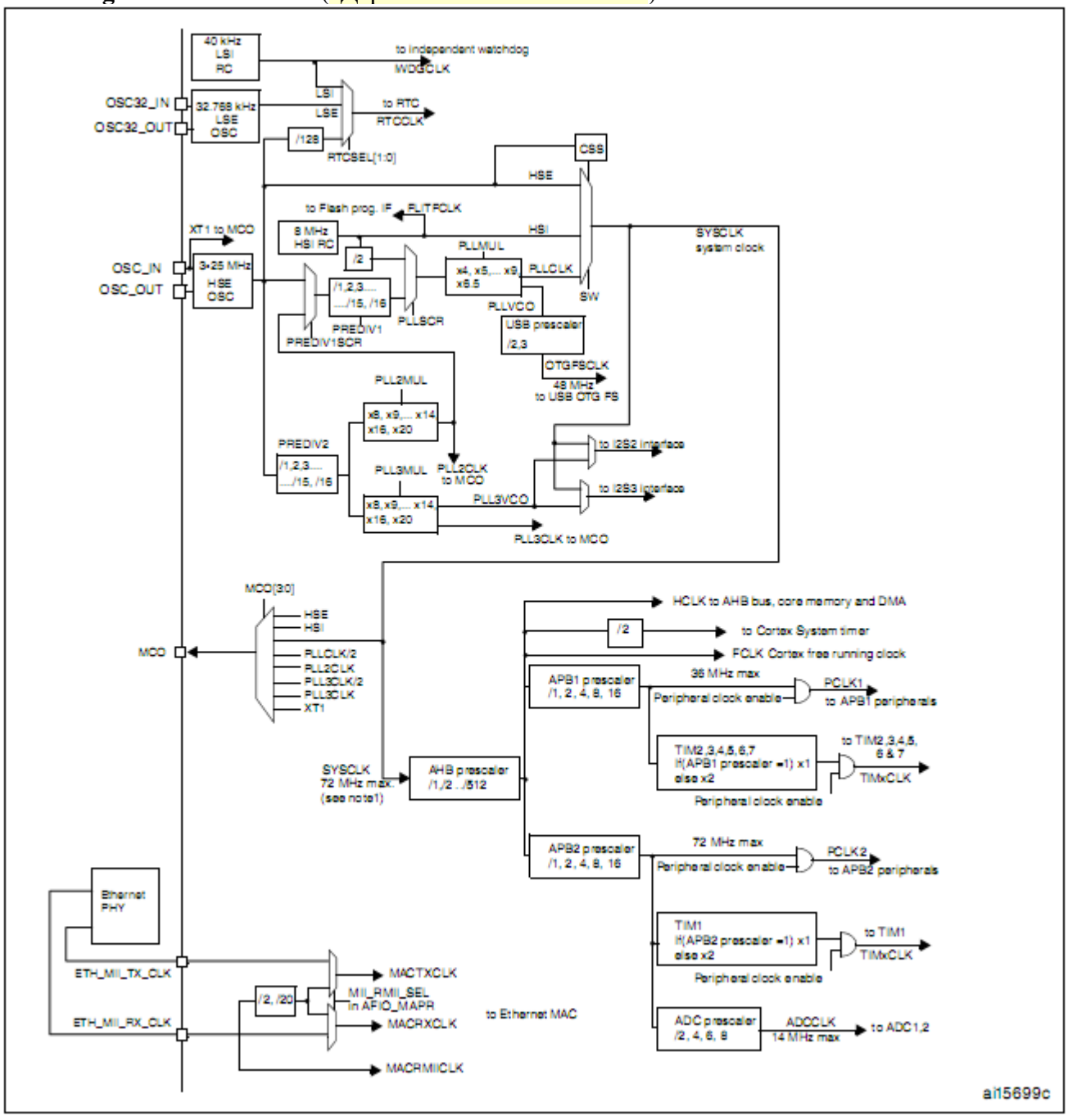
- HSI oscillator clock (Сигнал от резонатора **HSI**)
- HSE oscillator clock (Сигнал от резонатора **HSE**)
- PLL clock (Сигнал от **PLL**)

The devices have the following two secondary clock sources:
Устройства имеют два вторичных источника тактового сигнала:

- 40 kHz low speed internal RC (LSI RC) which drives the independent watchdog and optionally the RTC used for Auto-wakeup from Stop/Standby mode.
  Внутренний низко-частотный **RC** генератор на 40 кГц (**LSI RC**), который управляет независимым **WatchDog** и, по желанию, часами реального времени **RTC**, которые используются для автоматического пробуждения из режима **Stop/Standby**.
- 32.768 kHz low speed external crystal (LSE crystal) which optionally drives the real-time clock (RTCCLK)
  Внешний низко-частотный резонатор (**LSE crystal**) на 32.768 кГц, который, по желанию, управляет тактовым сигналом (**RTCCLK**) для **RTC**.

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.
Каждый источник тактового сигнала может быть независимо включен или выключен, если он не используется, чтобы оптимизировать потребляемую мощность.

Figure 11. Clock tree ("Дерево" тактовых сигналов)



1. When the HSI is used as a PLL clock input, the maximum system clock frequency that can be achieved is 36 MHz.
Когда, в качестве источника сигнала для **PLL** используется **HSI**, то максимальная системная тактовая частота, которая может быть достигнута, составляет 36 МГц.

2. For full details about the internal and external clock source characteristics, please refer to the “Electrical characteristics” section in your device datasheet. Для дополнительной информацией о характеристиках внутренних и внешних источников тактового сигнала, пожалуйста, обратитесь к разделу "***Electrical characteristics***" в **datasheet** на ваше устройство.

The advanced clock controller features 3 PLLs to provide a high degree of flexibility to the application in the choice of the external crystal or oscillator to run the core and peripherals at the highest frequency and guarantee the appropriate frequency for the Ethernet and USB OTG FS.
Улучшенный контроллер тактового сигнала имеет три **PLL**, чтобы обеспечить высокую степень гибкости для приложения в вопросе выбора внешнего кварца или резонатора, чтобы обеспечить ядро и внешние устройства самой высокой допустимой частотой и гарантировать при этом

необходимую частоту для **Ethernet** и **USB OTG FS**.

A single 25 MHz crystal can clock the entire system and all peripherals including the Ethernet and USB OTG FS peripherals. In order to achieve high-quality audio performance, an audio crystal can be used. In this case, the I2S master clock can generate all standard sampling frequencies from 8 kHz to 96 kHz with less than 0.5% accuracy. For more details about clock configuration for applications requiring Ethernet, USB OTG FS and/or I2S (audio), please refer to "Appendix A Applicative block diagrams" in your connectivity line device datasheet.
Единственный кварц на 25 МГц может обеспечить тактовыми сигналами всю систему и все внешние устройства, включая такую периферию, как **Ethernet** и **USB OTG FS**. Чтобы достигнуть высококачественной обработки аудио-сигнала, можно использовать звуковой кристалл. В этом случае, генератор тактовых сигналов для "Ведущего" **I2S** может генерировать все стандартные частоты сэмплирования от 8 до 96 кГц с точностью не менее 0.5%. За дополнительной информацией о конфигурации тактовых сигналов для приложений, требующих **Ethernet**, **USB OTG FS** и/или **I2S** (аудио), пожалуйста обратитесь к разделу "***Appendix A Applicative block diagrams***" в **datasheet** на ваше устройство семейства **Connectivity Line**.

Several prescalers allow the configuration of the AHB frequency, the high speed APB (APB2) and the low speed APB (APB1) domains. The maximum frequency of the AHB and the APB2 domains is 72 MHz. The maximum allowed frequency of the APB1 domain is 36 MHz.
Несколько делителей частоты позволяют конфигурировать частоту для доменов шины **AHB**, высоко-частотной шины **APB** (**APB2**) и низко-частотной шины **APB** (**APB1**). Максимальная частота доменов **AHB** и **APB2** составляет 72 МГц. Максимально допустимая частота домена **APB1** составляет 36 МГц.

All peripheral clocks are derived from the system clock (SYSCLK) except:
Все периферийные тактовые сигналы являются производными от системного тактового сигнала (**SYSCLK**), кроме:

- The Flash memory programming interface clock which is always the HSI clock
  Тактовый сигнал интерфейса с **Flash**-памятью программ, который всегда является тактовым сигналом **HSI**

- The USB OTG FS 48MHz clock which is derived from the PLL VCO clock
  Тактовый сигнал **USB OTG FS** на 48 МГц, который являются производным от **PLL VCO**

- The I2S2 and I2S3 clocks which can also be derived from the PLL3 VCO clock (selection by software)
  Тактовые сигналы для **I2S2** и **I2S3**, которые можно также получить из **PLL3 VCO** (выбирается программно)

- The Ethernet MAC clocks (TX, RX and RMII) which are provided from the external PHY. For further information on Ethernet configuration, please refer to *Section 27.4.4: MII/RMII selection*.
  Тактовые сигналы для **Ethernet MAC** (**TX**, **RX** и **RMII**), которые обеспечиваются внешним **PHY**. Для дополнительной информации о конфигурации **Ethernet**, пожалуйста, обратитесь к разделу 27.4.4 "***MII/RMII selection***".

The RCC feeds the Cortex System Timer (SysTick) external clock with the AHB clock (HCLK) divided by 8. The SysTick can work either with this clock or with the Cortex clock (HCLK), configurable in the SysTick Control and Status Register. The ADCs are clocked by the clock of the High Speed domain (APB2) divided by 2, 4, 6 or 8.
RCC снабжает системый таймер (**SysTick**) внешним тактовым сигналом **AHB** (**HCLK**), поделенным на 8. **SysTick** может работать либо с этим сигналом, либо с тактовым сигналом ядра (**HCLK**), что конфигурируется в его регистре **STK_CTRL**. АЦП тактируются сигналом высоко-частотного домена (**APB2**), поделенным на 2, 4, 6 или 8.

The timer clock frequencies are automatically fixed by hardware. There are two cases:
Частота тактового сигнала таймеров автоматически фиксируется, аппаратно. Есть два случая:

1. if the APB prescaler is 1, the timer clock frequencies are set to the same frequency as that of the APB domain to which the timers are connected.
если значение делителя частоты **APB = 1**, то частота тактового сигнала таймера соответствует частоте сигнала того домена **APB**, с которым он связан.

2. otherwise, they are set to twice (×2) the frequency of the APB domain to which the timers are connected.
в противном случае, она равна двукратной (**×2**) частоте домена **APB**, с которым таймер связан.

FCLK acts as Cortex™-M3 free running clock. For more details refer to the ***ARM Cortex™-M3 Technical Reference Manual***.
FCLK действует как тактовый сигнал ядра свободного хода. Обратитесь к документу "***Cortex-M3 technical reference manual***" за более детальной информацией.

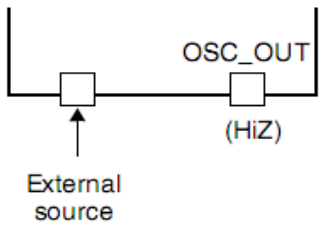
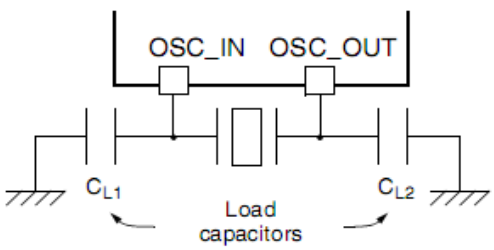
7.2.1 HSE clock (Тактовый от HSE)

The high speed external clock signal (HSE) can be generated from two possible clock sources:
Высоко-частотный внешний тактовый сигнал (**HSE**) может быть сгенерирован из двух возможных источников:

- HSE external crystal/ceramic resonator
С помощью внешнего кварцевого/керамического резонатора
- HSE user external clock (С помощью внешнего сигнала)

The resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected oscillator. Резонатор и нагрузочные конденсаторы должны быть помещены так близко к выводам генератора, насколько это возможно, чтобы минимизировать искажения и время стабилизации при запуске. Значения нагрузочных конденсаторов должны быть откорректированы в соответствии с выбранным резонатором.

Figure 12. HSE/LSE clock sources (Источники тактового сигнала **HSE/LSE**)

Clock source	Hardware configuration
External clock	
Crystal/ceramic resonators	

External source (HSE bypass) (Внешний источник (в обход HSE))

In this mode, an external clock source must be provided. It can have a frequency of up to 50 MHz. You select this mode by setting the HSEBYP and HSEON bits in the *Clock control register (RCC_CR)*. The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC_IN pin while the OSC_OUT pin should be left hi-Z. See *Figure 12*.
В этом режиме должен быть предоставлен внешний источник тактового сигнала. Его частота может быть до 50 МГц. Вы выбираете этот режим, устанавливая биты **HSEBYP** и **HSEON** в регистре **RCC_CR**. Внешний тактовый сигнал (прямоугольный, синусоилальный или треугольный), со скважностью около 50%, должен быть подан на вывод **OSC_IN**, в то время как вывод **OSC_OUT** нужно оставить в высоко-импедансном состоянии. См. рис 12.

External crystal/ceramic resonator (HSE crystal)
Внешний кварцевый/керамический резонатор (HSE)

The 3 to 25 MHz external oscillator has the advantage of producing a very accurate rate on the main clock.
Внешний резонатор от 3 до 25 МГц имеет то преимущество, что производит основной тактовый сигнал с очень высокой точностью.

The associated hardware configuration is shown in *Figure 12*. Refer to the electrical characteristics section of the datasheet for more details.
Соответствующая схема подключения приведена на рис. 12. Для дополнительной информацией обратитесь к разделу "***Electrical characteristics***" в **datasheet** на ваше устройство.

The HSERDY flag in the *Clock control register (RCC_CR)* indicates if the high-speed external oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *Clock interrupt register (RCC_CIR)*.
Флаг **HSERDY** в регистре **RCC_CR** показывает, стабилен **HSE** генератор, или нет. После запуска, тактового сигнала не будет до тех пор, пока не будет аппаратно установлен этот бит. При этом может быть сгенерировано прерывание, если оно разрешено в регистре **RCC_CIR**.

The HSE Crystal can be switched on and off using the HSEON bit in the *Clock control register (RCC_CR)*.
Режим **HSE Crystal** можно включить или выключить битом **HSEON** в регистре **RCC_CR**.

7.2.2 HSI clock (Тактовый от HSI)

The HSI clock signal is generated from an internal 8 MHz RC Oscillator and can be used directly as a system clock or divided by 2 to be used as PLL input.
Тактовый сигнал **HSI** генерируется с помощью внутреннего **RC** генератора на 8 МГц, и может использоваться непосредственно в качестве системного тактового сигнала, или, после деления на 2, использоваться в качестве входа для **PLL**.

The HSI RC oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the HSE crystal oscillator however, even with calibration the frequency is less accurate than an external crystal oscillator or ceramic resonator.
HSI RC генератор имеет то преимущество, то обеспечивает дешевый источник тактового сигнала (никаких внешних компонентов). Он также имеет более быстрое время запуска, чем **HSE** генератор на резонаторе, однако, даже после калибровки, его частота менее точна *и стабильна*, чем у генератора на внешнем кварцевом или керамическом резонаторе.

Calibration (Калибровка)

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by ST for 1% accuracy at TA=25°C.
Частота **RC** генератора может быть различной у разных чипов вследствии нестабильности производственного процесса, поэтому каждое **ST** устройство имеет фабричную калибровку 1%-ой точности при TA = 25°C.

After reset, the factory calibration value is loaded in the HSICAL[7:0] bits in the *Clock control register (RCC_CR)*.
После сброса, фабричное значение калибровки загружается в поле **HSICAL[7:0]** регистра **RCC_CR**.

If the application is subject to voltage or temperature variations this may affect the RC oscillator speed. You can trim the HSI frequency in the application using the HSITRIM[4:0] bits in the *Clock control register (RCC_CR)*.
Если ваше изделие подвержено заметным изменениям напряжения или температуры, то это может повлиять на частоту **RC** генератора. Вы можете программно немного подстроить частоту **HSI**, используя биты **HSITRIM[4:0]** регистра **RCC_CR**.

The HSIRDY flag in the *Clock control register (RCC_CR)* indicates if the HSI RC is stable or not. At startup, the HSI RC output clock is not released until this bit is set by hardware.
Флаг **HSIRDY** в регистре **RCC_CR** показывает, вышел ли **HSI RC** генератор на стабильный режим. После запуска, тактового сигнала **HSI RC** не будет до тех пор, пока не будет аппаратно установлен этот бит.

The HSI RC can be switched on and off using the HSION bit in the *Clock control register (RCC_CR)*.
Режим **HSI RC** можно включить или выключить битом **HSION** в регистре **RCC_CR**.

The HSI signal can also be used as a backup source (Auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 7.2.7: Clock security system (CSS) on page 112*.
Сигнал **HSI** можно также использовать в качестве резервного (дополнительного) источника тактового сигнала, если откажет **HSE** генератор на резонаторе. См. раздел 7.2.7 "Система безопасности тактового сигнала".

7.2.3 PLLs

The main PLL provides a frequency multiplier starting from one of the following clock sources:
Главный **PLL** имеет множитель частоты, который получает сигнал с одного из следующих источников тактового:

- HSI clock divided by 2 (Сигнал **HSI** поделенный на 2)
- HSE or PLL2 clock through a configurable divider

Сигнал **HSE** или **PLL2**, после конфигурируемого делителя

Refer to *Figure 11* and *Clock control register (RCC_CR)*.
См. рис 11 и раздел 7.3.1 "Регистр управления тактовыми (**RCC_CR**)"

PLL2 and PLL3 are clocked by HSE through a specific configurable divider. Refer to *Figure 11* and *Clock configuration register 2 (RCC_CFGR2)*
PLL2 и **PLL3** тактируются от **HSE** через специальный конфигурируемый делитель. См. рис 11 и раздел 7.3.12 "Регистр конфигурации тактового сигнала 2 (**RCC_CFGR2**)"

The configuration of each PLL (selection of clock source, predivision factor and multiplication factor) must be done before enabling the PLL. Each PLL should be enabled after its input clock becomes stable (ready flag). Once the PLL is enabled, these parameters can not be changed.
Конфигурация каждого **PLL** (выбор источника, коэффициента деления и коэффициента умножения), должен быть сделан прежде, чем разрешить **PLL**. Каждый **PLL** должен быть разрешен после того, как его входной сигнал становится устойчивыми (флаг готовности). Как только **PLL** разрешен, эти параметры не могут быть изменены.

When changing the entry clock source of the main PLL, the original clock source must be switched off only after the selection of the new clock source (done through bit PLLSRC in the *Clock configuration register (RCC_CFGR)*).
При изменении источника входного сигнала главного **PLL**, оригинальный источник сигнала должен быть выключен только после выбора нового источника (делается с помощью бита **PLLSRC** в регистре **RCC_CFGR**).

An interrupt can be generated when the PLL is ready if enabled in the *Clock interrupt register (RCC_CIR)*.
После готовности **PLL** может быть сгенерировано прерывание, если оно разрешено в регистре **RCC_CIR**.

7.2.4 LSE clock (Тактовый от LSE)

The LSE crystal is a 32.768 kHz Low Speed External crystal or ceramic resonator. It has the advantage providing a low-power but highly accurate clock source to the real-time clock peripheral (RTC) for clock/calendar or other timing functions.
Режим **LSE crystal** - это использование низко-частотного внешнего кварцевого или керамического резонатора на частоту 32.768 кГц. У этого режима есть то преимущество, что он предоставляет маломощный, но очень точный источник тактового сигнала для часов реального времени (**RTC**), для часов/календаря или других подобных задач.

The LSE crystal is switched on and off using the LSEON bit in *Backup domain control register (RCC_BDCR)*.
Режим **LSE crystal** можно включить или выключить битом **LSEON** в регистре **RCC_BDCR**.

The LSERDY flag in the Backup domain control register (RCC_BDCR) indicates if the LSE crystal is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *Clock interrupt register (RCC_CIR)*.
Флаг **LSERDY** в регистре **RCC_BDCR** показывает, вышел ли **LSE crystal** на стабильный режим. После запуска, тактового сигнала **LSE crystal** не будет до тех пор, пока не будет аппаратно установлен этот бит. При этом может быть сгенерировано прерывание, если оно разрешено в регистре **RCC_CIR**.

External source (LSE bypass) (Внешний источник (в обход LSE))

In this mode, an external clock source must be provided. It must have a frequency of 32.768 kHz. You select this mode by setting the LSEBYP and LSEON bits in the *Backup domain control register (RCC_BDCR)*. The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC32_IN pin while the OSC32_OUT pin should be left Hi-Z. See *Figure 12*.
В этом режиме должен быть предоставлен внешний источник тактового сигнала. Его частота должна быть 32.768 кГц. Вы выбираете этот режим, устанавливая биты **LSEBYP** и **LSEON** в регистре **RCC_BDCR**. Внешний тактовый сигнал (прямоугольный, синусоилальный или треугольный), со скважностью около 50%, должен быть подан на вывод **OSC32_IN**, в то время как вывод **OSC32_OUT** нужно оставить в высоко-импедансном состоянии. См. рис 12.

7.2.5 LSI clock (Тактовый от LSI)

The LSI RC acts as an low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG) and Auto-wakeup unit (AWU). The clock frequency is around 40 kHz (between 30 kHz and 60 kHz). For more details, refer to the electrical characteristics section of the datasheets.
LSI RC действует как источник тактового сигнала с низким потреблением, который можно оставить работающим в режимах **Stop** и **Standby** для независимого **WatchDog** (**IWDG**) и модуля Авто-пробуждения (**AWU**). Тактовая частота составляет приблизительно 40 кГц (между 30 кГц и 60 кГц).

The LSI RC can be switched on and off using the LSION bit in the *Control/status register (RCC_CSR)*.
Режим **LSI RC** можно включить или выключить битом **LSION** в регистре **RCC_CSR**.

The LSIRDY flag in the *Control/status register (RCC_CSR)* indicates if the low-speed internal oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *Clock interrupt register (RCC_CIR)*.
Флаг **LSIRDY** в регистре **RCC_CSR** показывает, вышел ли **LSI RC** генератор на стабильный режим. После запуска, тактового сигнала **LSI RC** не будет до тех пор, пока не будет аппаратно установлен этот бит. При этом может быть сгенерировано прерывание, если оно разрешено в регистре **RCC_CIR**.

LSI calibration (Калибровка LSI))

The frequency dispersion of the Low Speed Internal RC (LSI) oscillator can be calibrated to have accurate RTC time base and/or IWDG timeout (when LSI is used as clock source for these peripherals) with an acceptable accuracy.
Разброс частоты низко-частотного внутреннего **RC** генератора (**LSI**) можно откалибровать, чтобы иметь базу с приемлемой точностью для **RTC** и/или для независимого **WatchDog** (**IWDG**) (когда **LSI** используется как источник тактового сигнала для этой периферии).

This calibration is performed by measuring the LSI clock frequency with respect to TIM5 input clock (TIM5CLK). According to this measurement done at the precision of the HSE oscillator, the software can adjust the programmable 20-bit prescaler of the RTC to get an accurate time base or can compute accurate IWDG timeout.
Эта калибровка выполняется с помощью измерения тактовой частоты **LSI** относительно входного сигнала **TIM5** (**TIM5CLK**). Согласно этому измерению, сделанному с точностью **HSE** генератора, программа может скорректировать программируемый 20-ти битовый делитель **RTC**, чтобы получить точную основу времени для **RTC**, или может вычислить точный временной интервал **IWDG**.

Use the following procedure to calibrate the LSI:
Для калибровки **LSI** используйте следующую процедуру:

1. Enable TIM5 timer and configure channel4 in input capture mode
   Разрешите таймер **TIM5** и конфигурируйте **channel4** на режим захвата входа

2. Set the TIM5CH4_IREMAP bit in the AFIO_MAPR register to connect the LSI clock internally to TIM5 channel4 input capture for calibration purpose.
   Установите бит **TIM5CH4_IREMAP** в регистре **AFIO_MAPR**, чтобы внутренне подключить сигнал **LSI** ко входу захвата канала 4 таймера 5 в целях калибровки.

3. Measure the frequency of LSI clock using the TIM5 Capture/compare 4 event or interrupt.
   Имерьте частоту **LSI** сигнала, используя событие или прерывание **Capture/compare** 4

таймера 5.

4. Use the measured LSI frequency to update the 20-bit prescaler of the RTC depending on the desired time base and/or to compute the IWDG timeout.
В зависимости от того, что вы хотите получить, точную основу времени для **RTC**, или вычислить точный временной интервал **IWDG**, используйте измеренную частоту **LSI**, чтобы обновить 20-ти битовый делитель **RTC**.

7.2.6 System clock (SYSCLK) selection (Выбор тактового системы)

After a system reset, the HSI oscillator is selected as system clock. When a clock source is used directly or through the PLL as the system clock, it is not possible to stop it.
После "Системного сброса" в качестве тактового сигнала выбирается **HSI** генератор. Когда источник тактового сигнала используется в качестве системного тактового сигнала непосредственно, или через **PLL**, то нет способа остановить его.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source which is not yet ready is selected, the switch will occur when the clock source will be ready. Status bits in the *Clock control register (RCC_CR)* indicate which clock(s) is (are) ready and which clock is currently used as system clock.
Переключение с одного источника на другой происходит только тогда, когда целевой источник сигнала готов (сигнал стабилизировался после стартовой задержки или после сцепления **PLL**). Если выбран источник тактового сигнала, который еще не готов, то переключение произойдет по готовности источника. Биты состояния в регистре **RCC_CR** показывают, какие источники уже готовы и какой источник в настоящее время используются в качестве системного тактового сигнала.

7.2.7 Clock security system (CSS) (Система безопасности тактового сигнала)

Clock Security System can be activated by software. In this case, the clock detector is enabled after the HSE oscillator startup delay, and disabled when this oscillator is stopped.
Система безопасности тактового сигнала может быть активирована программно. В этом случае, детектор тактового сигнала разрешается после стартовой задержки **HSE** генератора, и отключается, когда этот генератор остановлен.

If a failure is detected on the HSE oscillator clock, this oscillator is automatically disabled, a clock failure event is sent to the break input of the TIM1 Advanced control timer and an interrupt is generated to inform the software about the failure (Clock Security System Interrupt CSSI), allowing the MCU to perform rescue operations. The CSSI is linked to the Cortex™-M3 NMI (Non-Maskable Interrupt) exception vector.
Если обнаружен отказ **HSE** генератора, то этот генератор автоматически отключается, посылается событие отказа генератора на прерывающий вход расширенного таймера **TIM1** и генерируется прерывание, чтобы сообщить программе об этом отказе (прерывание **CSSI**), позволяя ЦПУ выполнить операцию по спасению ситуации. Прерывание **CSSI** связано с вектором немаскируемого исключения (**NMI**) ядра **Cortex-M3**.

***Note**: Once the CSS is enabled and if the HSE clock fails, the CSS interrupt occurs and an NMI is automatically generated. The NMI will be executed indefinitely unless the CSS interrupt pending bit is cleared. As a consequence, in the NMI ISR user must clear the CSS interrupt by setting the CSSC bit in the **Clock interrupt register (RCC_CIR)**.*
***Note**: Как только **CSS** разрешен и, если есть отказ тактового сигнала **HSE**, то происходит прерывание **CSS**, и автоматически генерируется исключение **NMI**. Если бит запроса прерывания **CSS** не будет очищен, то прерывание **NMI** будет выполняться бесконечно. Следовательно, пользователь должен очистить запрос на прерывание **CSS** в бработчике **NMI**, установив бит **CSSC** в регистре **RCC_CIR**.*

If the HSE oscillator is used directly or indirectly as the system clock (indirectly means: it is used as PLL input clock directly or through PLL2, and the PLL clock is used as system clock), a detected failure causes a switch of the system clock to the HSI oscillator and the disabling of the external HSE oscillator. If the HSE oscillator clock is the clock entry of the PLL (directly or through PLL2) used as system clock when the failure occurs, the PLL is disabled too.
Если **HSE** генератор используется прямо или косвенно в качестве системнного (косвенно означает: что он используется как вход **PLL** напрямую, или через **PLL2**, а выход **PLL** уже используются как системный тактовый сигнал), то обнаруженный отказ вызывает переключение подачи системного тактового сигнала от **HSI** генератора и отключение внешнего **HSE** генератора. Если сигнал **HSE** генератора является входом **PLL** (непосредственно или через **PLL2**), выход которого используемый как системный тактовый сигнал, когда происходит отказ, то **PLL** также отключается.

7.2.8 RTC clock (Тактовый для RTC)

The RTCCLK clock source can be either the HSE/128, LSE or LSI clocks. This is selected by programming the RTCSEL[1:0] bits in the *Backup domain control register (RCC_BDCR)*. This selection cannot be modified without resetting the Backup domain.
Источником тактового сигнала **RTCCLK** могут быть сигналы **HSE/128**, **LSE** или **LSI**. Это выбирается программно битами **RTCSEL[1:0]** в регистре **RCC_BDCR**. Этот выбор не может быть изменен без сброса домена **BKP**.

The LSE clock is in the Backup domain, whereas the HSE and LSI clocks are not. Consequently:
Тактовый сигнал **LSE** присутствует в домене **BKP**, в то время как сигналы **HSE** и **LSI** - нет. Следовательно:

- If LSE is selected as RTC clock: (Если сигнал **LSE** выбран как тактовый для **RTC**, то:)
  - The RTC continues to work even if the VDD supply is switched off, provided the VBAT supply is maintained. Если есть питание на выводе **VBAT**, то **RTC** продолжают работать, даже когда снимается питание с вывода **VDD**.
- If LSI is selected as Auto-Wakeup unit (AWU) clock:

Если сигнал **LSI** выбран как тактовый для модуля Автопробуждения (**AWU**), то:

  - The AWU state is not guaranteed if the VDD supply is powered off. Refer to *Section 7.2.5: LSI clock on page 111* for more details on LSI calibration.
    Состояние **AWU** не гарантируется, если снимается питание с вывода **VDD**. Обратитесь к разделу 7.2.5 "Тактовый от **LSI**" за дополнительной информацией о калибровке **LSI**.
- If the HSE clock divided by 128 is used as RTC clock:

Если в качестве тактового для **RTC** используются сигнал **HSE**, поделенный на 128, то:

  - The RTC state is not guaranteed if the VDD supply is powered off or if the internal voltage regulator is powered off (removing power from the 1.8 V domain).
    Состояние **RTC** не гарантируется, если снимается питание с вывода **VDD** или с внутреннего регулятора напряжения (снятие питания с доменов, которые запитаны от 1.8 В).
  - The DPB bit (Disable backup domain write protection) in the Power controller register must be set to 1 (refer to *Section 4.4.1: Power control register (PWR_CR)*).
    Бит **DPB** (Отключает защиту от записи домена **BKP**) в регистре управления питанием должен быть установлен в '1' (обратитесь к разделу 4.4.1 ***"Power control register (PWR_CR)"***).

7.2.9 Watchdog clock (Тактовый для Watchdog)

If the Independent watchdog (IWDG) is started by either hardware option or software access, the LSI oscillator is forced ON and cannot be disabled. After the LSI oscillator temporization, the clock is provided to the IWDG.
Если запущен независимый **WatchDog** (**IWDG**), аппаратно или программно, то включается **LSI** генератор и не может быть отключен. После стартовой задержки **LSI** генератора тактовый сигнал поступает на **IWDG**.

7.2.10 Clock-out capability (Функция вывода тактового)

The microcontroller clock output (MCO) capability allows the clock to be output onto the external MCO pin. The configuration registers of the corresponding GPIO port must be programmed in alternate function mode. One of 8 clock signals can be selected as the MCO clock.
Функция **MCO** (*Microcontroller Clock Output*) позволяет вывести тактовый сигнал на внешний вывод **MCO**. Конфигурационные регистры соответствующего порта **GPIO** должны быть запрограммированы на режим альтернативных функций. Один из 8 тактовых сигналов может быть выбран как выход **MCO**.

- SYSCLK (Системный тактовый)
- HSI (Высоко-частотный внутренний сигнал)
- HSE (Высоко-частотный внешний сигнал)
- PLL clock divided by 2 selected (Выход **PLL**, деленный на 2)
- PLL2 clock selected (Выход **PLL2**)
- PLL3 clock divided by 2 selected (Выход **PLL3**, деленный на 2)
- XT1 external 3-25 MHz oscillator clock selected (for Ethernet)
  Сигнал 3-25 МГц с вывода **XT1** внешнего резонатора (для **Ethernet**)
- PLL3 clock selected (for Ethernet) (Выход **PLL3** (для **Ethernet**))

The selected clock to output onto MCO must not exceed 50 MHz (the maximum I/O speed).
Выбранный тактовый сигнал, для вывода на **MCO**, не должен превышать 50 МГц (максимальная частота для **I/O** ввода).

The selection is controlled by the MCO[3:0] bits of the *Clock configuration register (RCC_CFGR)*.
Выбором управляют биты **MCO[3:0]** в регистре **RCC_CFGR**.

7.3 RCC registers (Регистры RCC)

7.3.1 Clock control register (RCC_CR) (Регистр управления тактовыми)
7.3.2 Clock configuration register (RCC_CFGR) (Регистр конфигурации тактового сигнала)
7.3.3 Clock interrupt register (RCC_CIR) (Регистр прерывания от тактовых)
7.3.4 APB2 peripheral reset register (RCC_APB2RSTR) (Регистр сброса периферии **APB2**)
7.3.5 APB1 peripheral reset register (RCC_APB1RSTR) (Регистр сброса периферии **APB1**)
7.3.6 AHB Peripheral Clock enable register (RCC_AHBENR)
(Регистр разрешения периферии **AHB**)
7.3.7 APB2 peripheral clock enable register (RCC_APB2ENR)
(Регистр разрешения периферии **APB2**)
7.3.8 APB1 peripheral clock enable register (RCC_APB1ENR)
(Регистр разрешения периферии **APB1**)
7.3.9 Backup domain control register (RCC_BDCR) (Регистр управления доменом **BKP**)
7.3.10 Control/status register (RCC_CSR) (Регистр управления/статуса)
7.3.11 AHB peripheral clock reset register (RCC_AHBRSTR) (Регистр сброса периферии **AHB**)
7.3.12 Clock configuration register 2 (RCC_CFGR2)
(Регистр конфигурации тактового сигнала 2)
7.3.13 RCC register map (Карта регистров **RCC**)

Refer to *Section 1.1 on page 37* for a list of abbreviations used in register descriptions.
См. раздел 1.1, где приведен перечень сокращений, используемых при описании регистров.

7.3.1 Clock control register (RCC_CR)
Регистр управления тактовыми

Address offset: 0x00
Reset value: 0x0000 XX83 where X is undefined. (**X** - неопределенное значение)
Access: no wait state, word, half-word and byte access
Нет состояния ожидания, доступ словный, полусловный и байтовый

Reserved		PLL3 RDY	PLL3 ON	PLL2 RDY	PLL2 ON	PLLRDY	PLLON	Reserved				CSSON	HSEBYP	HSERDY	HSEON
		r	rw	r	rw	r	rw					rw	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res.	HSIRDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

Bits 31:30 Reserved
always read as 0. (Всегда читается как '0'.)

Bit 29 PLL3RDY: PLL3 clock ready flag (Флаг готовности сигнала от PLL3)
Set by hardware to indicate that the PLL3 is locked.
Ставится аппаратно, чтобы показать сцепление с **PLL3**
0: PLL3 unlocked (**PLL3** разомкнут)
1: PLL3 locked (**PLL3** сцеплен)

Bit 28 PLL3ON: PLL3 enable (Разрешение PLL3)

Set and cleared by software to enable PLL3.
Cleared by hardware when entering Stop or Standby mode.
Ставится и очищается программно, чтобы разрешить **PLL3**.
Очищается аппаратно при входе в режим **Stop** или **Standby**.

- **0:** PLL3 OFF (**PLL3** выключен)
- **1:** PLL3 ON (**PLL3** включен)

Bit 27 PLL2RDY: PLL2 clock ready flag (Флаг готовности сигнала от PLL2)

Set by hardware to indicate that the PLL2 is locked.
Ставится аппаратно, чтобы показать сцепление с **PLL2**

- **0:** PLL2 unlocked (**PLL2** разомкнут)
- **1:** PLL2 locked (**PLL2** сцеплен)

Bit 26 PLL2ON: PLL2 enable (Разрешение PLL2)

Set and cleared by software to enable PLL2. Cleared by hardware when entering Stop or Standby mode. This bit can not be cleared if the PLL2 clock is used indirectly as system clock (i.e. it is used as PLL clock entry that is used as system clock).
Ставится и очищается программно, чтобы разрешить **PLL2**. Очищается аппаратно при входе в режим **Stop** или **Standby**. Этот бит не может быть очищен, если сигнал **PLL2** косвенно используются в качестве системного тактового сигнала (то есть он используется как вход для **PLL**, который уже используется как системный тактовый).

- **0:** PLL2 OFF (**PLL2** выключен)
- **1:** PLL2 ON (**PLL2** включен)

Bit 25 PLLRDY: PLL clock ready flag (Флаг готовности сигнала от PLL)

Set by hardware to indicate that the PLL is locked.
Ставится аппаратно, чтобы показать сцепление с **PLL**

- **0:** PLL unlocked (**PLL** разомкнут)
- **1:** PLL locked (**PLL** сцеплен)

Bit 24 PLLON: PLL enable (Разрешение PLL)

Set and cleared by software to enable PLL. Cleared by hardware when entering Stop or Standby mode. This bit can not be reset if the PLL clock is used as system clock or is selected to become the system clock. Software must disable the USB OTG FS clock before clearing this bit.
Ставится и очищается программно, чтобы разрешить **PLL**. Очищается аппаратно при входе в режим **Stop** или **Standby**. Этот бит не может быть очищен, если сигнал **PLL** уже используются в качестве системного тактового сигнала, или выбран, чтобы стать таковым. Программа должна отключить тактовый **USB OTG FS**, прежде чем очистить этот бит.

- **0:** PLL OFF (**PLL** выключен)
- **1:** PLL ON (**PLL** включен)

Bits 23:20 Reserved

always read as 0. (Всегда читается как '0'.)

Bit 19 CSSON: Clock security system enable
Разрешение "Системы безопасности тактового сигнала"

Set and cleared by software to enable clock detector.
Ставится и очищается программно, чтобы разрешить детектор тактового сигнала.

- **0:** Clock detector OFF (Детектор тактового сигнала выключен)
- **1:** Clock detector ON if external 3-25 MHz oscillator is ready.

Детектор тактового сигнала включен, если готов сигнал от внешнего резонатора 3-25 МГц.

Bit 18 HSEBYP: External high-speed clock bypass
Обход внешнего высоко-частотного сигнала

Set and cleared by software in debug for bypassing the oscillator with an external clock.
This bit can be written only if the external 3-25 MHz oscillator is disabled.
Ставится и очищается программой отладчика, чтобы обойти генератор с внешним резонатором. Этот бит может быть записан только тогда, когда отключен внешний генератор на 3-25 МГц.

`0:` external 3-25 MHz oscillator not bypassed (Работает внешний генератор 3-25 МГц)
`1:` external 3-25 MHz oscillator bypassed with external clock

Внешний генератор 3-25 МГц обходится внешним сигналом

Bit 17 HSERDY: External high-speed clock ready flag
Флаг готовности внешнего высоко-частотного тактового сигнала

Set by hardware to indicate that the external 3-25 MHz oscillator is stable. This bit needs 6 cycles of external 3-25 MHz oscillator clock to fall down after HSEON reset.
Ставится аппаратно и показывает, что внешний 3-25 МГц генератор стабилизировался. Необходимо 6 циклов тактового сигнала от внешнего 3-25 МГц генератор, чтобы этот бит перешел в '0' после сброса бита **HSEON**.

`0:` external 3-25 MHz oscillator not ready (Внешний 3-25 МГц генератор не готов)
`1:` external 3-25 MHz oscillator ready (Внешний 3-25 МГц генератор готов)

Bit 16 HSEON: External high-speed clock enable
Разрешение внешнего высоко-частотного тактового сигнала

Set and cleared by software. Cleared by hardware to stop the external 3-25MHz oscillator when entering Stop or Standby mode. This bit can not be reset if the external 3-25 MHz oscillator is used directly or indirectly as system clock or is selected to become the system clock.
Ставится и очищается программно. Очищается аппаратно при входе в режим **Stop** или **Standby**, чтобы остановить внешний 3-25 МГц генератор. Этот бит не может быть очищен, если внешний 3-25 МГц генератор уже используются в качестве системного тактового сигнала, прямо или косвенно, или выбран, чтобы стать таковым.

`0:` HSE oscillator OFF (Высоко-частотный генератор выключен)
`1:` HSE oscillator ON (Высоко-частотный генератор включен)

Bits 15:8 HSICAL[7:0]: Internal high-speed clock calibration
Калибровка внутренего высоко-частотного тактового сигнала

These bits are initialized automatically at startup.
Эти биты инициируются автоматически, при запуске.

Bits 7:3 HSITRIM[4:0]: Internal high-speed clock trimming
Подстройка внутренего высоко-частотного тактового сигнала

These bits provide an additional user-programmable trimming value that is added to the HSICAL[7:0] bits. It can be programmed to adjust to variations in voltage and temperature that influence the frequency of the internal HSI RC. The default value is 16, which, when added to the HSICAL value, should trim the HSI to 8 MHz ± 1%. The trimming step (Fhsitrim) is around 40 kHz between two consecutive HSICAL steps.
Эти биты обеспечивают дополнительное, программируемое пользователем значение подстройки частоты, которое добавляется к битам **HSICAL[7:0]**. Это значение можно запрограммировать, чтобы скорректировать влияние изменения напряжения и температуры на частоту внутреннего **HSI RC**. Значение по умолчанию - 16, которое, когда добавляется к значению **HSICAL**, должно соответствовать частоте **HSI** 8 МГц ± 1%. Шаг подстройки, между двумя последовательными шагами **HSICAL**, составляет приблизительно 40 кГц.

Bits 2 Reserved

always read as 0. (Всегда читается как '0'.)

Bit 1 HSIRDY: Internal high-speed clock ready flag
Флаг готовности внутренего высоко-частотного тактового сигнала

Set by hardware to indicate that internal 8 MHz RC oscillator is stable. After the HSION bit is cleared, HSIRDY goes low after 6 internal 8 MHz RC oscillator clock cycles.
Ставится аппаратно и показывает, что внутренний 8 МГц **RC** генератор стабилизировался. Необходимо 6 циклов тактового сигнала от внутреннего 8 МГц **RC** генератора, чтобы этот бит перешел в '0' после сброса бита **HSEON**.

`0:` Internal 8 MHz RC oscillator not ready (Внутренний 8 МГц **RC** генератор не готов)
`1:` Internal 8 MHz RC oscillator ready (Внутренний 8 МГц **RC** генератор готов)

Bit 0 HSION: Internal high-speed clock enable
Разрешение внутренего высоко-частотного тактового сигнала

Set and cleared by software. Set by hardware to force the internal 8 MHz RC oscillator ON when leaving Stop or Standby mode or in case of failure of the external 3-25 MHz oscillator used directly or indirectly as system clock. This bit can not be cleared if the internal 8 MHz RC is used directly or indirectly as system clock or is selected to become the system clock.
Ставится и очищается программно. Ставится аппаратно, чтобы принудительно включить внутренний 8 МГц **RC** генератор при выходе из режима **Stop** или **Standby**, или в случае отказа внешнего 3-25 МГц генератора, который использовался в качестве системного тактового сигнала, прямо или косвенно. Этот бит не может быть очищен, если внутренний 8 МГц **RC** генератор уже используются в качестве системного тактового сигнала, прямо или косвенно, или выбран, чтобы стать таковым.

`0:` Internal 8 MHz RC oscillator OFF (Внутренний 8 МГц **RC** генератор выключен)
`1:` Internal 8 MHz RC oscillator ON (Внутренний 8 МГц **RC** генератор включен)

7.3.2 Clock configuration register (RCC_CFGR)
Регистр конфигурации тактового сигнала

Address offset: 0x04
Reset value: 0x0000 0000
`Access:` 0 <= wait state <= 2, word, half-word and byte access
0 <= состояние ожидания <= 2, доступ словный, полусловный и байтовый

Reserved				MCO[3:0]				Res.	OTGF SPRE	PLLMUL[3:0]				PLL XTPRE	PLL SRC
				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ADC PRE[1:0]		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

Bits 31:27 Reserved

always read as 0. (Всегда читается как '0'.)

Bits 26:24 MCO[3:0]: Microcontroller clock output (Выход тактового сигнала)

Set and cleared by software. (Ставится и очищается программно.)

00xx: No clock (Нет сигнала на выводе **MCO**)
0100: System clock (SYSCLK) selected (Выбран системный тактовый сигнал)
0101: HSI clock selected (Выбран высоко-частотный внутренний сигнал)
0110: HSE clock selected (Выбран высоко-частотный внешний сигнал)
0111: PLL clock divided by 2 selected (Выбран сигнал с выхода **PLL**, деленный на 2)
1000: PLL2 clock selected (Выбран сигнал с выхода **PLL2**)
1001: PLL3 clock divided by 2 selected (Выбран сигнал с выхода **PLL3**, деленный на 2)
1010: XT1 external 3-25 MHz oscillator clock selected (for Ethernet)
Выбран сигнал 3-25 МГц с вывода **XT1** внешнего резонатора (для **Ethernet**)
1011: PLL3 clock selected (for Ethernet) (Выбран сигнал с выхода **PLL3** (для **Ethernet**))

Note: This clock output may have some truncated cycles at startup or during MCO clock source switching. The selected clock to output onto the MCO pin must not exceed 50 MHz (the maximum I/O speed).
Note: У этого выходного сигнала может быть потеря некоторых циклов при запуске, или во время переключение источника сигнала для **MCO**. Выбранный тактовый сигнал, для вывода на **MCO**, не должен превышать 50 МГц (максимальная частота для **I/O** ввода).

Bit 22 OTGFSPRE: USB OTG FS prescaler (Делитель для USB OTG FS)

Set and cleared by software to generate the 48 MHz USB OTG FS clock. This bit must be valid before enabling the OTG FS clock in the RCC_APB1ENR register. This bit can not be cleared if the OTG FS clock is enabled.
Ставится и очищается программно, чтобы генерировать тактовый сигнал 48 МГц для **USB OTG FS**. Этот бит должен быть корректным перед разрешением подачи тактового на **USB OTG FS** в регистре **RCC_APB1ENR**. Этот бит нельзя очистить, если подача тактового на **USB OTG FS** разрешена.

0: PLL VCO clock is divided by 3 (Тактовый сигнал с выхода **PLL VCO** делится на 3)
1: PLL VCO clock is divided by 2 (Тактовый сигнал с выхода **PLL VCO** делится на 2)

Bits 21:18 PLLMUL[3:0]: PLL multiplication factor (Коэффициент умножения для PLL)

These bits are written by software to define the PLL multiplication factor. They can be written only when PLL is disabled.
Эти биты меняются программно, чтобы определить коэффициент умножения для **PLL**. Их можно менять только тогда, когда **PLL** выключен.

000x: Reserved
0010: PLL input clock x 4
0011: PLL input clock x 5
0100: PLL input clock x 6
0101: PLL input clock x 7
0110: PLL input clock x 8
0111: PLL input clock x 9
10xx: Reserved
1100: Reserved
1101: PLL input clock x 6.5
111x: Reserved

Caution: The PLL output frequency must not exceed 72 MHz.
Предупреждение: Выходная частота **PLL** не должна превышать 72 МГц.

Bit 17 PLLXTPRE: LSB of division factor PREDIV1 (Коэффициент деления для LSB)

Set and cleared by software to select the least significant bit of the PREDIV1 division factor. It is the same bit as bit(0) in the RCC_CFGR2 register, so modifying bit(0) in the RCC_CFGR2 register changes this bit accordingly.

If bits[3:1] in register RCC_CFGR2 are not set, this bit controls if PREDIV1 divides its input clock by 2 (PLLXTPRE=1) or not (PLLXTPRE=0).

This bit can be written only when PLL is disabled.

Ставится и очищается программно, чтобы выбрать младший бит коэффициента деления **PREDIV1**. Это тот же самый бит, что и **bit(0)** в регистре **RCC_CFGR2**, поэтому, изменение **bit(0)** в регистре **RCC_CFGR2** изменяет и этот бит соответственно.

Если биты **[3:1]** в регистре **RCC_CFGR2** не установлены, то этот бит управляет тем, будет ли **PREDIV1** делить входной сигнал на 2 (**PLLXTPRE=1**) или не будет (**PLLXTPRE=0**).

Этот бит можно менять только тогда, когда **PLL** выключен.

Bit 16 PLLSRC: PLL entry clock source (Источник входного сигнала для PLL)

Set and cleared by software to select PLL clock source.

This bit can be written only when PLL is disabled.

Ставится и очищается программно, чтобы выбрать источник входного сигнала для **PLL**.

Этот бит можно менять только тогда, когда **PLL** выключен.

0: HSI oscillator clock/2 selected as PLL input clock

Входом для **PLL** выбран высоко-частотный внутренний сигнал, деленный на 2

1: Clock from PREDIV1 selected as PLL input clock

Входом для **PLL** выбран выход делителя **PREDIV1**

Note: When changing the main PLL's entry clock source, the original clock source must be switched off only after the selection of the new clock source.

Когда изменяется источник сигнала для входа главного **PLL**, то оригинальный источник сигнала должен быть выключен только после выбора нового источника.

Bits 14:14 ADCPRE[1:0]: ADC prescaler (Делитель для ADC)

Set and cleared by software to select the frequency of the clock to the ADCs.

Ставятся и очищаются программно, чтобы выбрать тактовую частоту для АЦП.

00: PLCK2 divided by 2
01: PLCK2 divided by 4
10: PLCK2 divided by 6
11: PLCK2 divided by 8

Bits 13:11 PPRE2[2:0]: APB high-speed prescaler (APB2)
Делитель для высоко-частотной шины (APB2)

Set and cleared by software to control the division factor of the APB High speed clock (PCLK2).

Ставятся и очищаются программно, чтобы управлять коэффициентом деления тактового сигнала для высоко-частотной шины (**PCLK2**).

0xx: HCLK not divided
100: HCLK divided by 2
101: HCLK divided by 4
110: HCLK divided by 8
111: HCLK divided by 16

Bits 10:8 PPRE1[2:0]: APB Low-speed prescaler (APB1)
Делитель для низко-частотной шины (APB1)

Set and cleared by software to control the division factor of the APB Low speed clock (PCLK1).
Ставятся и очищаются программно, чтобы управлять коэффициентом деления тактового сигнала для низко-частотной шины (**PCLK1**).

`0xx:` HCLK not divided
`100:` HCLK divided by 2
`101:` HCLK divided by 4
`110:` HCLK divided by 8
`111:` HCLK divided by 16

Caution: Software must configure these bits ensure that the frequency in this domain does not exceed 36 MHz.
Предупреждение: Программа, при конфигурировании этих битов, должна убедиться, что частота в этом домене не превышает 36 МГц.

Bits 7:4 HPRE[3:0]: AHB prescaler (Делитель для AHB)

Set and cleared by software to control AHB clock division factor.
Ставятся и очищаются программно, чтобы управлять коэффициентом деления тактового сигнала для **AHB**.

`0xxx:` SYSCLK not divided
`1000:` SYSCLK divided by 2
`1001:` SYSCLK divided by 4
`1010:` SYSCLK divided by 8
`1011:` SYSCLK divided by 16
`1100:` SYSCLK divided by 64
`1101:` SYSCLK divided by 128
`1110:` SYSCLK divided by 256
`1111:` SYSCLK divided by 512

Note: The prefetch buffer must be kept on when using a prescaler different from 1 on the AHB clock. Refer to the section Reading the Flash memory on page 47 for more details.
Note: Когда используется делитель частоты, отличный от 1, для тактового сигнала **AHB**, то буфер предварительной выборки должен удерживаться в состоянии "Включен". Обратитесь к разделу ***"Reading the Flash memory"*** за дополнительной информацией.

Bits 3:2 SWS[1:0]: System clock switch status
Статус переключателя системного тактового сигнала

Set and cleared by hardware to indicate which clock source is used as system clock.
Ставится и очищается аппаратно, чтобы показать, какой из источников используется в качестве системного тактового сигнала.

`00:` HSI oscillator used as system clock (В качестве системного тактового сигнала используется внутренний высоко-частотный генератор)
`01:` HSE oscillator used as system clock (В качестве системного тактового сигнала используется внешний высоко-частотный генератор)
`10:` PLL used as system clock
В качестве системного тактового сигнала используется выход **PLL**
`11:` Not applicable (Не использовать)

Bits 1:0 SW[1:0]: System clock Switch (Переключатель системного тактового сигнала)

Set and cleared by software to select SYSCLK source. Set by hardware to force HSI selection when leaving Stop and Standby mode or in case of failure of the HSE oscillator used directly or indirectly as system clock (if the Clock Security System is enabled).

Ставится и очищается программно, чтобы выбрать источник для **SYSCLK**. Ставится аппаратно, чтобы принудительно включить **HSI** при выходе из режима **Stop** или **Standby**, или в случае отказа **HSE** генератора, который использовался в качестве системного тактового сигнала, прямо или косвенно (если разрешена "Система безопасности тактового сигнала").

`00:` HSI selected as system clock (Выбор внутреннего высоко-частотного генератора в качестве системного тактового сигнала)

`01:` HSE selected as system clock (Выбор внешнего высоко-частотного генератора в качестве системного тактового сигнала)

`10:` PLL selected as system clock

Выбор выхода **PLL** в качестве системного тактового сигнала

`11:` Not allowed (Не использовать)

7.3.3 Clock interrupt register (RCC_CIR)

Регистр прерывания от тактовых

Address offset: 0x08
Reset value: 0x0000 0000
Access: no wait state, word, half-word and byte access
Нет состояния ожидания, доступ словный, полусловный и байтовый

Reserved								CSSC	PLL3 RDYC	PLL2 RDYC	PLL RDYC	HSE RDYC	HSI RDYC	LSE RDYC	LSI RDYC
								w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PLL3 RDYIE	PLL2 RDYIE	PLL RDYIE	HSE RDYIE	HSI RDYIE	LSE RDYIE	LSI RDYIE	CSSF	PLL3 RDYF	PLL2 RDYF	PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF
	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r

Bits 31:24 Reserved

always read as 0. (Всегда читается как '0'.)

Bit 23 CSSC: Clock security system interrupt clear

Очистка прерывания от "Системы безопасности тактового сигнала"

This bit is set by software to clear the CSSF flag.

Этот бит ставится программно, чтобы очистить флаг **CSSF**.

`0:` No effect (Нет эффекта)

`1:` Clear CSSF flag (Очистка флага **CSSF**)

Bit 22 PLL3RDYC: PLL3 Ready Interrupt Clear

Очистка прерывания от готовности PLL3

This bit is set by software to clear the PLL3RDYF flag.

Этот бит ставится программно, чтобы очистить флаг **PLL3RDYF**.

`0:` No effect (Нет эффекта)

`1:` Clear PLL3RDYF flag (Очистка флага **PLL3RDYF**)

Bit 21 PLL2RDYC: PLL2 Ready Interrupt Clear
Очистка прерывания от готовности PLL2

This bit is set by software to clear the PLL2RDYF flag.
Этот бит ставится программно, чтобы очистить флаг **PLL2RDYF**.

- **0:** No effect (Нет эффекта)
- **1:** Clear PLL2RDYF flag (Очистка флага **PLL2RDYF**)

Bit 20 PLLRDYC: PLL ready interrupt clear
Очистка прерывания от готовности PLL

This bit is set by software to clear the PLLRDYF flag.
Этот бит ставится программно, чтобы очистить флаг **PLLRDYF**.

- **0:** No effect (Нет эффекта)
- **1:** Clear PLLRDYF flag (Очистка флага **PLLRDYF**)

Bit 19 HSERDYC: HSE ready interrupt clear
Очистка прерывания от готовности HSE

This bit is set by software to clear the HSERDYF flag.
Этот бит ставится программно, чтобы очистить флаг **HSERDYF**.

- **0:** No effect (Нет эффекта)
- **1:** Clear HSERDYF flag (Очистка флага **HSERDYF**)

Bit 18 HSIRDYC: HSI ready interrupt clear
Очистка прерывания от готовности HSI

This bit is set by software to clear the HSIRDYF flag.
Этот бит ставится программно, чтобы очистить флаг **HSIRDYF**.

- **0:** No effect (Нет эффекта)
- **1:** Clear HSIRDYF flag (Очистка флага **HSIRDYF**)

Bit 17 LSERDYC: LSE ready interrupt clear
Очистка прерывания от готовности LSE

This bit is set by software to clear the LSERDYF flag.
Этот бит ставится программно, чтобы очистить флаг **LSERDYF**.

- **0:** No effect (Нет эффекта)
- **1:** Clear LSERDYF flag (Очистка флага **LSERDYF**)

Bit 16 LSIRDYC: LSI ready interrupt clear
Очистка прерывания от готовности LSI

This bit is set by software to clear the LSIRDYF flag.
Этот бит ставится программно, чтобы очистить флаг **LSIRDYF**.

- **0:** No effect (Нет эффекта)
- **1:** Clear LSIRDYF flag (Очистка флага **LSIRDYF**)

Bit 15 Reserved

always read as 0. (Всегда читается как '0'.)

Bit 14 PLL3RDYIE: PLL3 Ready Interrupt Enable
Разрешение прерывания от готовности PLL3

Set and cleared by software to enable/disable interrupt caused by PLL3 lock.
Ставится и очищается программно, чтобы разрешить/запретить прерывание, вызванное сцеплением с **PLL3**.

- **0:** PLL3 lock interrupt disabled (Прерывание от сцепления с **PLL3** запрещено)
- **1:** LL3 lock interrupt enabled (Прерывание от сцепления с **PLL3** разрешено)

Bit 13 PLL2RDYIE: PLL2 Ready Interrupt Enable
Разрешение прерывания от готовности PLL2

Set and cleared by software to enable/disable interrupt caused by PLL2 lock.
Ставится и очищается программно, чтобы разрешить/запретить прерывание, вызванное сцеплением с **PLL2**.

- `0:` PLL2 lock interrupt disabled (Прерывание от сцепления с **PLL2** запрещено)
- `1:` PLL2 lock interrupt enabled (Прерывание от сцепления с **PLL2** разрешено)

Bit 12 PLLRDYIE: PLL ready interrupt enable
Разрешение прерывания от готовности PLL

Set and cleared by software to enable/disable interrupt caused by PLL lock.
Ставится и очищается программно, чтобы разрешить/запретить прерывание, вызванное сцеплением с **PLL**.

- `0:` PLL lock interrupt disabled (Прерывание от сцепления с **PLL** запрещено)
- `1:` PLL lock interrupt enabled (Прерывание от сцепления с **PLL** разрешено)

Bit 11 HSERDYIE: HSE ready interrupt enable
Разрешение прерывания от готовности HSE

Set and cleared by software to enable/disable interrupt caused by the external 3-25 MHz oscillator stabilization.
Ставится и очищается программно, чтобы разрешить/запретить прерывание, вызванное стабилизацией внешнего 3-25 МГц генератора.

- `0:` HSE ready interrupt disabled (Прерывание от готовности **HSE** запрещено)
- `1:` HSE ready interrupt enabled (Прерывание от готовности **HSE** разрешено)

Bit 10 HSIRDYIE: HSI ready interrupt enable
Разрешение прерывания от готовности HSI

Set and cleared by software to enable/disable interrupt caused by the internal 8 MHz RC oscillator stabilization.
Ставится и очищается программно, чтобы разрешить/запретить прерывание, вызванное стабилизацией внутреннего 8 МГц **RC** генератора.

- `0:` HSI ready interrupt disabled (Прерывание от готовности **HSI** запрещено)
- `1:` HSI ready interrupt enabled (Прерывание от готовности **HSI** разрешено)

Bit 9 LSERDYIE: LSE ready interrupt enable
Разрешение прерывания от готовности LSE

Set and cleared by software to enable/disable interrupt caused by the external 32 kHz oscillator stabilization.
Ставится и очищается программно, чтобы разрешить/запретить прерывание, вызванное стабилизацией внешнего 32 кГц генератора.

- `0:` LSE ready interrupt disabled (Прерывание от готовности **LSE** запрещено)
- `1:` LSE ready interrupt enabled (Прерывание от готовности **LSE** разрешено)

Bit 8 LSIRDYIE: LSI ready interrupt enable
Разрешение прерывания от готовности LSI

Set and cleared by software to enable/disable interrupt caused by internal RC 40 kHz oscillator stabilization.
Ставится и очищается программно, чтобы разрешить/запретить прерывание, вызванное стабилизацией внутреннего 40 кГц **RC** генератора.

- `0:` LSI ready interrupt disabled (Прерывание от готовности **LSI** запрещено)
- `1:` LSI ready interrupt enabled (Прерывание от готовности **LSI** разрешено)

Bit 7 CSSF: Clock security system interrupt flag
Флаг запроса прерывания от "Системы безопасности тактового сигнала"

Set by hardware when a failure is detected in the external 3-25 MHz oscillator. It is cleared by software setting the CSSC bit.
Ставится аппаратно, когда обнаружен отказ внешнего 3-25 МГц генератора. Он очищается программно, установкой бита **CSSC**.

`0:` No clock security interrupt caused by HSE clock failure
Нет прерывания от "Системы безопасности тактового сигнала", вызванного отказом **HSE**

`1:` Clock security interrupt caused by HSE clock failure
Есть прерывание от "Системы безопасности тактового сигнала", вызванного отказом **HSE**

Bit 6 PLL3RDYF: PLL3 Ready Interrupt flag
Флаг запроса прерывания от готовности PLL3

Set by hardware when the PLL3 locks and PLL3RDYIE is set. It is cleared by software setting the PLL3RDYC bit.
Ставится аппаратно, когда есть сцепление с **PLL3** и ставится бит **PLL3RDYIE**. Он очищается программно, установкой бита **PLL3RDYC**.

`0:` No clock ready interrupt caused by PLL3 lock
Нет прерывания, вызванного готовностью **PLL3**

`1:` Clock ready interrupt caused by PLL3 lock
Есть прерывание, вызванное готовностью **PLL3**

Bit 5 PLL2RDYF: PLL2 Ready Interrupt flag
Флаг запроса прерывания от готовности PLL2

Set by hardware when the PLL2 locks and PLL2RDYDIE is set. It is cleared by software setting the PLL2RDYC bit.
Ставится аппаратно, когда есть сцепление с **PLL2** и ставится бит **PLL2RDYIE**. Он очищается программно, установкой бита **PLL2RDYC**.

`0:` No clock ready interrupt caused by PLL2 lock
Нет прерывания, вызванного готовностью **PLL2**

`1:` Clock ready interrupt caused by PLL2 lock
Есть прерывание, вызванное готовностью **PLL2**

Bit 4 PLLRDYF: PLL ready interrupt flag
Флаг запроса прерывания от готовности PLL

Set by hardware when the PLL locks and PLLRDYDIE is set. It is cleared by software setting the PLLRDYC bit.
Ставится аппаратно, когда есть сцепление с **PLL** и ставится бит **PLLRDYIE**. Он очищается программно, установкой бита **PLLRDYC**.

`0:` No clock ready interrupt caused by PLL lock
Нет прерывания, вызванного готовностью **PLL**

`1:` Clock ready interrupt caused by PLL lock
Есть прерывание, вызванное готовностью **PLL**

Bit3 HSERDYF: HSE ready interrupt flag
Флаг запроса прерывания от готовности HSE

Set by hardware when External Low (*High*) Speed clock becomes stable and HSERDYIE is set. It is cleared by software setting the HSERDYC bit.
Ставится аппаратно, когда внешний высоко-частотный тактовый сигнал становиться стабильным и ставится бит **HSERDYIE**. Он очищается программно, установкой бита **HSERDYC**.

`0:` No clock ready interrupt caused by the external 3-25 MHz oscillator
Нет прерывания, вызванного готовностью тактового сигнала от внешнего 3-25 МГц генератора

`1:` Clock ready interrupt caused by the external 3-25 MHz oscillator
Есть прерывание, вызванное готовностью тактового сигнала от внешнего 3-25 МГц генератора

Bit 2 HSIRDYF: HSI ready interrupt flag
Флаг запроса прерывания от готовности HSI

Set by hardware when the Internal High Speed clock becomes stable and HSIRDYIE is set. It is cleared by software setting the HSIRDYC bit.
Ставится аппаратно, когда внутренний высоко-частотный тактовый сигнал становиться стабильным и ставится бит **HSIRDYIE**. Он очищается программно, установкой бита **HSIRDYC**.

0: No clock ready interrupt caused by the internal 8 MHz RC oscillator
Нет прерывания, вызванного готовностью тактового сигнала от внутреннего 8 МГц RC генератора

1: Clock ready interrupt caused by the internal 8 MHz RC oscillator
Есть прерывание, вызванное готовностью тактового сигнала от внутреннего 8 МГц RC генератора

Bit 1 LSERDYF: LSE ready interrupt flag
Флаг запроса прерывания от готовности LSE

Set by hardware when the External Low Speed clock becomes stable and LSERDYIE is set. It is cleared by software setting the LSERDYC bit.
Ставится аппаратно, когда внешний низко-частотный тактовый сигнал становиться стабильным и ставится бит **LSERDYIE**. Он очищается программно, установкой бита **LSERDYC**.

0: No clock ready interrupt caused by the external 32 kHz oscillator
Нет прерывания, вызванного готовностью тактового сигнала от внешнего 32 кГц генератора

1: Clock ready interrupt caused by the external 32 kHz oscillator
Есть прерывание, вызванное готовностью тактового сигнала от внешнего 32 кГц генератора

Bit 0 LSIRDYF: LSI ready interrupt flag (Флаг запроса прерывания от готовности LSI)

Set by hardware when Internal Low Speed clock becomes stable and LSIRDYIE is set. It is cleared by software setting the LSIRDYC bit.
Ставится аппаратно, когда внутренний низко-частотный тактовый сигнал становиться стабильным и ставится бит **LSIRDYIE**. Он очищается программно, установкой бита **LSIRDYC**.

0: No clock ready interrupt caused by the internal RC 40 kHz oscillator
Нет прерывания, вызванного готовностью тактового сигнала от внутреннего 40 кГц **RC** генератора

1: Clock ready interrupt caused by the internal RC 40 kHz oscillator
Есть прерывание, вызванное готовностью тактового сигнала от внутреннего 40 кГц **RC** генератора

7.3.4 APB2 peripheral reset register (RCC_APB2RSTR)
Регистр сброса периферии APB2

Address offset: 0x0C
Reset value: 0x00000 0000
Access: no wait state, word, half-word and byte access
Нет состояния ожидания, доступ словный, полусловный и байтовый

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 RST	Res.	SPI1 RST	TIM1 RST	ADC2 RST	ADC1 RST	Reserved		IOPE RST	IOPD RST	IOPC RST	IOPB RST	IOPA RST	Res.	AFIO RST
	rw		rw	rw	rw	rw			rw	rw	rw	rw	rw		rw

Bits 31:15, 13, 8, 7, 1 Reserved: always read as 0. (Зарезервированы. Всегда читаются как '0'.)
Bit 14 USART1RST: USART1 reset (Сброс **USART1**)
Bit 12 SPI1RST: SPI 1 reset (Сброс **SPI 1**)
Bit 11 TIM1RST: TIM1 timer reset (Сброс таймера **TIM1**)
Bit 10 ADC2RST: ADC 2 interface reset (Сброс интерфейса **ADC 2**)
Bit 9 ADC1RST: ADC 1 interface reset (Сброс интерфейса **ADC 1**)
Bit 6 IOPERST: I/O port E reset (Сброс **I/O** порта **E**)
Bit 5 IOPDRST: I/O port D reset (Сброс **I/O** порта **D**)
Bit 4 IOPCRST: IO port C reset (Сброс **I/O** порта **C**)
Bit 3 IOPBRST: IO port B reset (Сброс **I/O** порта **B**)
Bit 2 IOPARST: I/O port A reset (Сброс **I/O** порта **A**)
Bit 0 AFIORST: Alternate function I/O reset (Сброс альтернативной функции **I/O**)

Set and cleared by software. (Ставятся и очищаются программно.)
- `0:` No effect (Нет эффекта)
- `1:` Reset interface (Сброс указанного интерфейса)

7.3.5 APB1 peripheral reset register (RCC_APB1RSTR)
Регистр сброса периферии APB1

Address offset: 0x10
Reset value: 0x0000 0000
Access: no wait state, word, half-word and byte access
Нет состояния ожидания, доступ словный, полусловный и байтовый

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DAC RST	PWR RST	BKP RST	CAN2 RST	CAN1 RST	Reserved		I2C2 RST	I2C1 RST	UART 5 RST	UART 4 RST	USART 3 RST	USART 2 RST	Res.
		rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 RST	SPI2 RST	Reserved		WWD GRST	Reserved					TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw			rw						rw	rw	rw	rw	rw	rw

Bits 31:30, 24:23, 16, 13:12, 10:6 Reserved: always read as 0.
Зарезервированы. Всегда читаются как '0'.
Bit 29 DACRST: DAC interface reset (Сброс интерфейса **DAC**)
Bit 28 PWRRST: Power interface reset (Сброс интерфейса **Power**)
Bit 27 BKPRST: Backup interface reset (Сброс интерфейса **BKP**)
Bit 26 CAN2RST: CAN2 reset (Сброс **CAN2**)
Bit 25 CAN1RST: CAN1 reset (Сброс **CAN1**)
Bit 22 I2C2RST: I2C 2 reset (Сброс **I2C 2**)
Bit 21 I2C1RST: I2C 1 reset (Сброс **I2C 1**)
Bit 20 UART5RST: USART 5 reset (Сброс **UART 5**)
Bit 19 UART4RST: USART 4 reset (Сброс **UART 4**)
Bit 18 USART3RST: USART 3 reset (Сброс **UART 3**)
Bit 17 USART2RST: USART 2 reset (Сброс **UART 2**)
Bit 15 SPI3RST: SPI 3 reset (Сброс **SPI 3**)
Bit 14 SPI2RST: SPI 2 reset (Сброс **SPI 2**)
Bit 11 WWDGRST: Window watchdog reset (Сброс **Window watchdog**)
Bit 5 TIM7RST: Timer 7 reset (Сброс **Timer 7**)
Bit 4 TIM6RST: Timer 6 reset (Сброс **Timer 6**)
Bit 3 TIM5RST: Timer 5 reset (Сброс **Timer 5**)
Bit 2 TIM4RST: Timer 4 reset (Сброс **Timer 4**)

Bit 1 TIM3RST: Timer 3 reset (Сброс **Timer 3**)
Bit 0 TIM2RST: Timer 2 reset (Сброс **Timer 2**)

Set and cleared by software. (Ставятся и очищаются программно.)
- `0:` No effect (Нет эффекта)
- `1:` Reset interface (Сброс указанного интерфейса)

7.3.6 AHB Peripheral Clock enable register (RCC_AHBENR)
Регистр разрешения периферии AHB

Address offset: 0x14
Reset value: 0x0000 0014
Access: no wait state, word, half-word and byte access
Нет состояния ожидания, доступ словный, полусловный и байтовый

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															ETHMACRXEN
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETHMACTXEN	ETHMACEN	Res.	OTGFSEN	Reserved					CRCEN	Res.	FLITFEN	Res.	SRAMEN	DMA2EN	DMA1EN
rw	rw		rw						rw		rw		rw	rw	rw

Bits 31:17, 13, 11-7, 5, 3 Reserved: always read as 0. (Зарезервированы. Всегда читаются как '0'.)

Bit 16 ETHMACRXEN: Ethernet MAC RX clock enable
Разрешение подачи тактового сигнала на приемник **Ethernet**

> **Note**: In the RMII mode, if this clock is enabled, the RMII clock of the MAC is also enabled.
> В режиме **RMII**, если подан тактовый сигнал на приемник **Ethernet**, то сигнал **RMII** для **MAC** также разрешен.

Bit 15 ETHMACTXEN: Ethernet MAC TX clock enable
Разрешение подачи тактового сигнала на передатчик **Ethernet**

> **Note**: In the RMII mode, if this clock is enabled, the RMII clock of the MAC is also enabled.
> В режиме **RMII**, если подан тактовый сигнал на передатчик **Ethernet**, то сигнал **RMII** для **MAC** также разрешен.

Bit 14 ETHMACEN: Ethernet MAC clock enable
Разрешение подачи тактового сигнала на **MAC Ethernet**

> **Note**: Selection of PHY interface (MII/RMII) must be done before enabling the MAC clock.
> Выбор **PHY** интерфейса (**MII/RMII**) должен быть сделан до разрешения этого сигнала.

Bit 12 OTGFSEN: USB OTG FS clock enable
Разрешение подачи тактового сигнала на **USB OTG FS**

Bit 6 CRCEN: CRC clock enable (Разрешение подачи тактового сигнала на **CRC**)

Bit 4 FLITFEN: FLITF clock enable
Разрешение подачи тактового сигнала на **FLITF** во время сна

Bit 2 SRAMEN: SRAM interface clock enable
Разрешение подачи тактового сигнала на интерфейс **SRAM** в режиме **Sleep**

Bit 1 DMA2EN: DMA2 clock enable (Разрешение подачи тактового сигнала на **DMA2**)

Bit 0 DMA1EN: DMA1 clock enable (Разрешение подачи тактового сигнала на **DMA1**)

Set and cleared by software. (Ставятся и очищаются программно.)
0: Interface clock disabled (Отключен тактовый сигнал интерфейса)
1: nterface clock enable (Подан тактовый сигнал на указанный интерфейс)

7.3.7 APB2 peripheral clock enable register (RCC_APB2ENR)
Регистр разрешения периферии APB2

Address: 0x18
Reset value: 0x0000 0000
Access: word, half-word and byte access (Доступ словный, полусловный и байтовый)

No wait states, except if the access occurs while an access to a peripheral in the APB2 domain is on going. In this case, wait states are inserted until the access to APB2 peripheral is finished.
Нет состояния ожидания, кроме случая, когда доступ происходит в то время, как уже есть другой доступ к домену **APB2**. В этом случае вставляется состояние ожидания, пока текущий доступ к периферии **APB2** не будет завершен.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1EN	Res.	SPI1EN	TIM1EN	ADC2EN	ADC1EN	Reserved		IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	Res.	AFIOEN
	rw		rw	rw	rw	rw			rw	rw	rw	rw	rw		rw

Bits 31:15, 13, 8, 7, 1 Reserved: always read as 0. (Зарезервированы. Всегда читаются как '0'.)
Bit 14 USART1EN: USART1 clock enable (Разрешение подачи тактового сигнала на **USART1**)
Bit 12 SPI1EN: SPI 1 clock enable (Разрешение подачи тактового сигнала на **SPI 1**)

Bit 11 TIM1EN: TIM1 Timer clock enable
Разрешение подачи тактового сигнала на таймер **TIM1**

Bit 10 ADC2EN: ADC 2 interface clock enable
Разрешение подачи тактового сигнала на интерфейс **ADC 2**

Bit 9 ADC1EN: ADC 1 interface clock enable
Разрешение подачи тактового сигнала на интерфейс **ADC 1**

Bit 6 IOPEEN: I/O port E clock enable (Разрешение подачи тактового сигнала на **I/O** порт **E**(
Bit 5 IOPDEN: I/O port D clock enable (Разрешение подачи тактового сигнала на **I/O** порт **D**)
Bit 4 IOPCEN: I/O port C clock enable (Разрешение подачи тактового сигнала на **I/O** порт **C**)
Bit 3 IOPBEN: I/O port B clock enable (Разрешение подачи тактового сигнала на **I/O** порт **B**)
Bit 2 IOPAEN: I/O port A clock enable (Разрешение подачи тактового сигнала на **I/O** порт **A**)

Bit 0 AFIOEN: Alternate function I/O clock enable
Разрешение подачи тактового сигнала на альтернативные функции **I/O**

Set and cleared by software. (Ставятся и очищаются программно.)
0: Interface clock disabled (Отключен тактовый сигнал интерфейса)
0: Interface clock enable (Подан тактовый сигнал на указанный интерфейс)

7.3.8 APB1 peripheral clock enable register (RCC_APB1ENR)
Регистр разрешения периферии APB1

Address: 0x1C
Reset value: 0x0000 0000
Access: word, half-word and byte access (Доступ словный, полусловный и байтовый)

No wait state, except if the access occurs while an access to a peripheral on APB1 domain is on going. In this case, wait states are inserted until this access to APB1 peripheral is finished.
Нет состояния ожидания, кроме случая, когда доступ происходит в то время, как уже есть другой доступ к домену **APB1**. В этом случае вставляется состояние ожидания, пока текущий доступ к периферии **APB1** не будет завершен.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DAC EN	PWR EN	BKP EN	CAN2 EN	CAN1 EN	Reserved		I2C2 EN	I2C1 EN	UART5E N	UART4 EN	USART 3EN	USART 2EN	Res.
		rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Reserved		WWD GEN	Reserved					TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw						rw	rw	rw	rw	rw	rw

Bits 31:30, 24, 23, 16, 13, 12, 10:6 Reserved: always read as 0.
Зарезервированы. Всегда читаются как '0'.

Bit 29 DACEN: DAC interface clock enable
Разрешение подачи тактового сигнала на интерфейс **DAC**

Bit 28 PWREN: Power interface clock enable
Разрешение подачи тактового сигнала на интерфейс **Power**

Bit 27 BKPEN: Backup interface clock enable
Разрешение подачи тактового сигнала на интерфейс **BKP**

Bit 26 CAN2EN: CAN2 clock enable (Разрешение подачи тактового сигнала на **CAN2**)
Bit 25 CAN1EN: CAN1 clock enable (Разрешение подачи тактового сигнала на **CAN1**)
Bit 22 I2C2EN: I2C 2 clock enable (Разрешение подачи тактового сигнала на **I2C 2**)
Bit 21 I2C1EN: I2C 1 clock enable (Разрешение подачи тактового сигнала на **I2C 1**)
Bit 20 UART5EN: USART 5 clock enable (Разрешение подачи тактового сигнала на **USART 5**)
Bit 19 UART4EN: USART 4 clock enable (Разрешение подачи тактового сигнала на **USART 4**)
Bit 18 USART3EN: USART 3 clock enable (Разрешение подачи тактового сигнала на **USART 3**)
Bit 17 USART2EN: USART 2 clock enable (Разрешение подачи тактового сигнала на **USART 2**)
Bit 15 SPI3EN: SPI 3 clock enable (Разрешение подачи тактового сигнала на **SPI 3**)
Bit 14 SPI2EN: SPI 2 clock enable (Разрешение подачи тактового сигнала на **SPI 2**)

Bit 11 WWDGEN: Window watchdog clock enable
Разрешение подачи тактового сигнала на **Window watchdog**

Bit 5 TIM7EN: Timer 7 clock enable (Разрешение подачи тактового сигнала на **Timer 7**)
Bit 4 TIM6EN: Timer 6 clock enable (Разрешение подачи тактового сигнала на **Timer 6**)
Bit 3 TIM5EN: Timer 5 clock enable (Разрешение подачи тактового сигнала на **Timer 5**)
Bit 2 TIM4EN: Timer 4 clock enable (Разрешение подачи тактового сигнала на **Timer 4**)
Bit 1 TIM3EN: Timer 3 clock enable (Разрешение подачи тактового сигнала на **Timer 3**)

Bit 0 TIM2EN: Timer 2 clock enable (Разрешение подачи тактового сигнала на **Timer 2**)

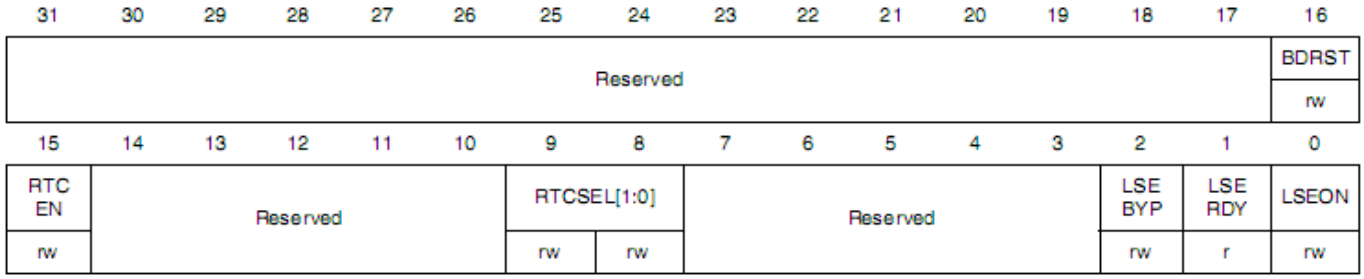
Set and cleared by software. (Ставятся и очищаются программно.)
0: Interface clock disabled (Отключен тактовый сигнал интерфейса)
0: Interface clock enable (Подан тактовый сигнал на указанный интерфейс)

7.3.9 Backup domain control register (RCC_BDCR)
Регистр управления доменом BKP

Address: 0x20
Reset value: 0x0000 0000, reset by Backup domain Reset. (сбрасывается от "Сброса домена **BKP**")
Access: 0 <= wait state <= 3, word, half-word and byte access
0 <= состояние ожидания <= 3, доступ словный, полусловный и байтовый

Wait states are inserted in the case of successive accesses to this register.
Состояние ожидания вставляется в случае успешного доступа к этому регистру.

***Note**: LSEON, LSEBYP, RTCSEL and RTCEN bits of the **Backup domain control register (RCC_BDCR)** are in the Backup domain. As a result, after Reset, these bits are write-protected and the DBP bit in the **Power control register (PWR_CR)** has to be set before these can be modified. Refer to **Section 5 on page 66** for further information. These bits are only reset after a Backup domain Reset (see **Section 7.1.3: Backup domain reset**). Any internal or external Reset will not have any effect on these bits.*
***Note**: Биты **LSEON**, **LSEBYP**, **RTCSEL** и **RTCEN** регистра **RCC_BDCR** находятся в домене **BKP**. В результате, после сброса, эти биты защищены от записи, и, прежде чем изменять эти биты, надо установить бит **DPB** в регистре **PWR_CR**. Обратитесь к разделу 5 для дополнительной информации. Эти биты сбрасываются только при сбросе домена **BKP** (см. раздел 7.1.3 "Сброс домена **BKP**"). Любой внутренний или внешний Сброс не будет иметь никакого эффекта на эти биты.*



Bits 31:17 Reserved
always read as 0. (Всегда читается как '0'.)

Bit 16 BDRST: Backup domain software reset (Программный сброс домена BKP)
Set and cleared by software. (Ставится и очищается программно.)
0: Reset not activated (Сброс не активирован)
1: Resets the entire Backup domain (Сброс всего домена **BKP**)

Bit 15 RTCEN: RTC clock enable (Разрешение подачи тактового сигнала на RTC)

Set and cleared by software. (Ставится и очищается программно.)

0: RTC clock disabled (Отключен тактовый сигнал **RTC**)

1: RTC clock enabled (Подан тактовый сигнал на **RTC**)

Bits 14:10 Reserved

always read as 0. (Всегда читается как '0'.)

Bits 9:8 RTCSEL[1:0]: RTC clock source selection
Выбор источника тактового сигнала для RTC

Set by software to select the clock source for the RTC.
Once the RTC clock source has been selected, it cannot be changed anymore unless the Backup domain is reset. The BDRST bit can be used to reset the RTCSEL[1:0] bits.
Ставится программно, чтобы выбрать источник тактового сигнала для **RTC**.
Как только источник для **RTC** был выбран, он не может быть изменен иначе, как только через сброс домена **BKP**. Можно использовать бит **BDRST**, чтобы сбросить биты **RTCSEL[1:0]**.

00: No clock (Нет тактового)

01: LSE oscillator clock used as RTC clock (В качестве тактового сигнала для **RTC** используется внешний низко-частотный генератор)

10: LSI oscillator clock used as RTC clock (В качестве тактового сигнала для **RTC** используется внутренний низко-частотный **RC** генератор)

11: HSE oscillator clock divided by 128 used as RTC clock
В качестве тактового сигнала для **RTC** используется сигнал внешнего высоко-частотного генератора, поделенный на 128

Bits 7:3 Reserved

always read as 0. (Всегда читается как '0'.)

Bit 2 LSEBYP: External Low Speed oscillator bypass
Обход внешнего низко-частотного генератора

Set and cleared by software to bypass oscillator in debug mode.
This bit can be written only when the external 32 kHz oscillator is disabled.
Ставится и очищается программно, чтобы обойти генератор в режиме отладки. Этот бит можно менять только тогда, когда внешний 32 кГц генератор выключен.

0: LSE oscillator not bypassed (Работает **LSE** генератор)

1: LSE oscillator bypassed (Обход **LSE** генератора)

Bit 1 LSERDY: External Low Speed oscillator ready
Готовность внешнего низко-частотного генератора

Set and cleared by hardware to indicate when the external 32 kHz oscillator is stable. After the LSEON bit is cleared, LSERDY goes low after 6 external low speed oscillator clock cycles
Ставится и очищается аппаратно, чтобы показать, что внешний 32 кГц генератор стабилизировался. Необходимо 6 циклов тактового сигнала от внешнего низко-частотного генератора, чтобы этот бит перешел в '0' после сброса бита **LSEON**.

0: External 32 kHz oscillator not ready (Внешний 32 кГц генератор не готов)

1: External 32 kHz oscillator ready (Внешний 32 кГц генератор готов)

Bit 0 LSEON: External Low Speed oscillator enable
Разрешение внешнего низко-частотного генератора

Set and cleared by software. (Ставится и очищается программно.)

0: External 32 kHz oscillator OFF (Внешний 32 кГц генератор выключен)

1: External 32 kHz oscillator ON (Внешний 32 кГц генератор включен)

7.3.10 Control/status register (RCC_CSR)
Регистр управления/статуса

Address: 0x24
Reset value: 0x0C00 0000, reset by system Reset, except reset flags by power Reset only.
Сбрасывается "Системным сбросом", за исключением флагов сброса, которые сбрасываются только "Сбросом от подачи питания".
Access: 0 <= wait state <= 3, word, half-word and byte access
0 <= состояние ожидания <= 3, доступ словный, полусловный и байтовый

Wait states are inserted in the case of successive accesses to this register.
Состояние ожидания вставляется в случае успешного доступа к этому регистру.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	Res.	RMVF	Reserved							
rw	rw	rw	rw	rw	rw		rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														LSI RDY	LSION
														r	rw

Bit 31 LPWRRSTF: Low-power reset flag (Флаг сброса от "Пониженного потребления")
Set by hardware when a Low-power management reset occurs.
It is cleared by writing to the RMVF bit.
Ставится аппаратно, когда случается сброс от контроллера пониженного потребления, а очищается записью в бит **RMVF**.
0: No Low-power management reset occurred
Нет сброса от контроллера пониженного потребления
1: Low-power management reset occurred
Случился сброс от контроллера пониженного потребления
For further information on Low-power management reset, refer to Section : Low-power management reset.
Для дополнительной информацией о сбросе от контроллера пониженного потребления см. раздел "Сброс от контроллера пониженного потребления".

Bit 30 WWDGRSTF: Window watchdog reset flag (Флаг сброса от оконного WatchDog)
Set by hardware when a window watchdog reset occurs.
It is cleared by writing to the RMVF bit.
Ставится аппаратно, когда случается сброс от оконного **WatchDog**, а очищается записью в бит **RMVF**.
0: No window watchdog reset occurred (Нет сброса от оконного **WatchDog**)
1: Window watchdog reset occurred (Случился сброс от оконного **WatchDog**)

Bit 29 IWDGRSTF: Independent watchdog reset flag
Флаг сброса от независимого WatchDog
Set by hardware when an independent watchdog reset from VDD domain occurs.
It is cleared by writing to the RMVF bit.
Ставится аппаратно, когда случается сброс от независимого **WatchDog**, а очищается записью в бит **RMVF**.
0: No watchdog reset occurred (Нет сброса от независимого **WatchDog**)
1: Watchdog reset occurred (Случился сброс от независимого **WatchDog**)

Bit 28 SFTRSTF: Software reset flag (Флаг сброса от "Программного сброса")

Set by hardware when a software reset occurs. It is cleared by writing to the RMVF bit.
Ставится аппаратно, когда случается "Программный сброс", а очищается записью в бит **RMVF**.

- **0:** No software reset occurred (Нет сброса от "Программного сброса")
- **1:** Software reset occurred (Случился сброс от "Программного сброса")

Bit 27 PORRSTF: POR/PDR reset flag (Флаг сброса от "Подачи/Снятия питания")

Set by hardware when a POR/PDR reset occurs. It is cleared by writing to the RMVF bit.
Ставится аппаратно, когда случается от "Подачи/Снятия питания", а очищается записью в бит **RMVF**.

- **0:** No POR/PDR reset occurred (Нет сброса от **POR/PDR**)
- **1:** POR/PDR reset occurred (Случился сброс от **POR/PDR**)

Bit 26 PINRSTF: PIN reset flag (Флаг сброса от вывода "Сброса")

Set by hardware when a reset from the NRST pin occurs.
It is cleared by writing to the RMVF bit.
Ставится аппаратно, когда случается от вывода **NRST**, а очищается записью в бит **RMVF**.

- **0:** No reset from NRST pin occurred (Нет сброса от вывода **NRST**)
- **1:** Reset from NRST pin occurred (Случился сброс от вывода **NRST**)

Bit 25 Reserved

always read as 0. (Всегда читается как '0'.)

Bit 24 RMVF: Remove reset flag (Очистка флагов сброса)

Set by software to clear the reset flags.
Ставится программно, чтобы очистить флаги сброса.

- **0:** No effect (Нет эффекта)
- **1:** Clear the reset flags (Очистка флагов сброса)

Bits 23:2 Reserved

always read as 0. (Всегда читается как '0'.)

Bit 1 LSIRDY: Internal low speed oscillator ready
Готовность внутреннего низко-частотного генератора

Set and cleared by hardware to indicate when the internal RC 40 kHz oscillator is stable. After the LSION bit is cleared, LSIRDY goes low after 3 internal 40 kHz RC oscillator clock cycles.
Ставится и очищается аппаратно, чтобы показать, что внутренний 40 кГц **RC** генератор стабилизировался. Необходимо 3 цикла тактового сигнала от внутреннего 40 кГц **RC** генератора, чтобы этот бит перешел в '0' после сброса бита **LSION**.

- **0:** Internal RC 40 kHz oscillator not ready (Внутренний 40 кГц **RC** генератор не готов)
- **1:** Internal RC 40 kHz oscillator ready (Внутренний 40 кГц **RC** генератор готов)

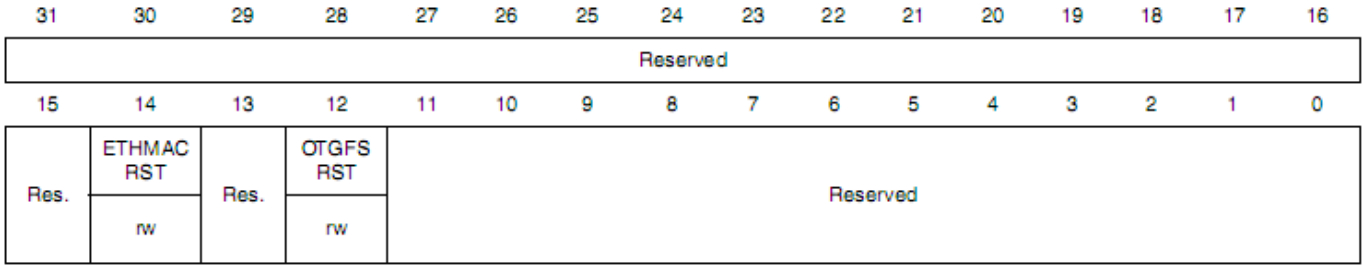
Bit 0 LSION: Internal low speed oscillator enable
Разрешение внутреннего низко-частотного генератора

Set and cleared by software. (Ставится и очищается программно.)

- **0:** Internal RC 40 kHz oscillator OFF (Внутренний 40 кГц **RC** генератор выключен)
- **1:** Internal RC 40 kHz oscillator ON (Внутренний 40 кГц **RC** генератор включен)

7.3.11 AHB peripheral clock reset register (RCC_AHBRSTR)
Регистр сброса периферии AHB

Address offset: 0x28
Reset value: 0x0000 0000
Access: no wait state, word, half-word and byte access
Нет состояния ожидания, доступ словный, полусловный и байтовый



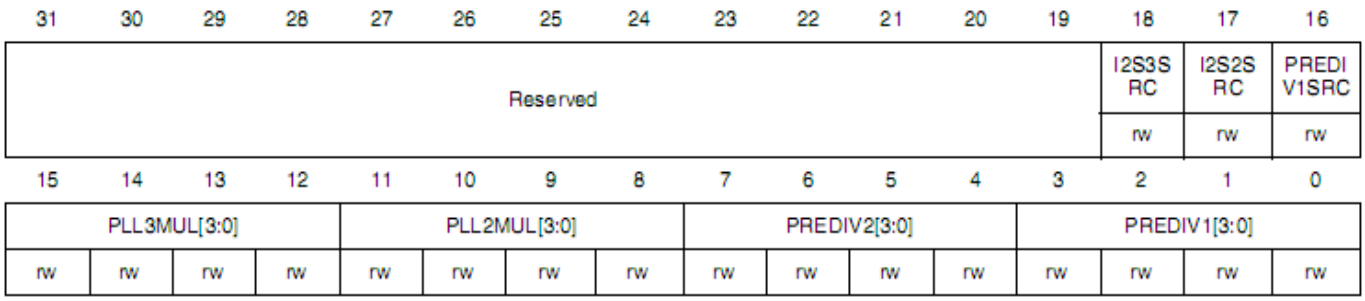
Bits 31:15, 13, 11:0 Reserved: always read as 0. (Всегда читаются как '0'.)
Bit 14 ETHMACRST Ethernet MAC reset (Сброс **Ethernet MAC**)

Bit 12 OTGFSRST USB OTG FS reset (Сброс **USB OTG FS**)

Set and cleared by software. (Ставятся и очищаются программно.)
`0:` No effect (Нет эффекта)
`1:` Reset interface (Сброс указанного интерфейса)

7.3.12 Clock configuration register 2 (RCC_CFGR2)
Регистр конфигурации тактовых 2

Address offset: 0x2C
Reset value: 0x0000 0000
Access: no wait state, word, half-word and byte access
Нет состояния ожидания, доступ словный, полусловный и байтовый



Bits 31:19 Reserved
always read as 0. (Всегда читается как '0'.)

Bit 18 I2S3SRC: I2S3 clock source (Источник тактового сигнала для I2S3)
Set and cleared by software to select I2S3 clock source. This bit must be valid before enabling I2S3 clock. Ставится и очищается программно, чтобы выбрать источник тактового сигнала для **I2S3**. Этот бит должен быть корректным перед подачей тактового на **I2S3**.
`0:` System clock selected as I2S3 clock entry
В качестве тактового для **I2S3** выбран системный тактовый сигнал (**SYSCLK**)
`1:` PLL3 VCO clock selected as I2S3 clock entry
В качестве тактового для **I2S3** выбран сигнал с **PLL3 VCO**

Bit 17 I2S2SRC: I2S2 clock source (Источник тактового сигнала для I2S2)

Set and cleared by software to select I2S2 clock source. This bit must be valid before enabling I2S2 clock. Ставится и очищается программно, чтобы выбрать источник тактового сигнала для **I2S2**. Этот бит должен быть корректным перед подачей тактового на **I2S2**.

0: System clock (SYSCLK) selected as I2S2 clock entry

В качестве тактового для **I2S2** выбран системный тактовый сигнал (**SYSCLK**)

1: PLL3 VCO clock selected as I2S2 clock entry

В качестве тактового для **I2S2** выбран сигнал с **PLL3 VCO**

Bit 16 PREDIV1SRC: PREDIV1 entry clock source

Источник тактового сигнала для входа PREDIV1

Set and cleared by software to select PREDIV1 clock source. This bit can be written only when PLL is disabled. Ставится и очищается программно, чтобы выбрать источник тактового сигнала для делителя **PREDIV1**. Этот бит можно менять только тогда, когда **PLL** отключен.

0: HSE oscillator clock selected as PREDIV1 clock entry

В качестве сигнала для входа делителя **PREDIV1** выбран **HSE** генератор

1: PLL2 selected as PREDIV1 clock entry

В качестве сигнала для входа делителя **PREDIV1** выбран **PLL2**

Bits 15:12 PLL3MUL[3:0]: PLL3 Multiplication Factor (Коэффициент умножения для PLL3)

Set and cleared by software to control PLL3 multiplication factor. These bits can be written only when PLL3 is disabled.

Ставятся и очищаются программно, чтобы управлять коэффициентом умножения для **PLL3**. Эти биты можно менять только тогда, когда **PLL3** отключен.

00xx: Reserved
010x: Reserved
0110: PLL3 clock entry x 8
0111: PLL3 clock entry x 9
1000: PLL3 clock entry x 10
1001: PLL3 clock entry x 11
1010: PLL3 clock entry x 12
1011: PLL3 clock entry x 13
1100: PLL3 clock entry x 14
1101: Reserved
1110: PLL3 clock entry x 16
1111: PLL3 clock entry x 20

Bits 11:8 PLL2MUL[3:0]: PLL2 Multiplication Factor (Коэффициент умножения для PLL2)

Set and cleared by software to control PLL2 multiplication factor. These bits can be written only when PLL2 is disabled. Ставятся и очищаются программно, чтобы управлять коэффициентом умножения для **PLL2**. Эти биты можно менять только тогда, когда **PLL2** отключен.

00xx: Reserved
010x: Reserved
0110: PLL2 clock entry x 8
0111: PLL2 clock entry x 9
1000: PLL2 clock entry x 10
1001: PLL2 clock entry x 11
1010: PLL2 clock entry x 12
1011: PLL2 clock entry x 13
1100: PLL2 clock entry x 14
1101: Reserved
1110: PLL2 clock entry x 16
1111: PLL2 clock entry x 20

Bits 7:4 PREDIV2[3:0]: PREDIV2 division factor (Коэффициент деления для PREDIV2)

Set and cleared by software to select PREDIV2 division factor. These bits can be written only when both PLL2 and PLL3 are disabled.

Ставятся и очищаются программно, чтобы управлять коэффициентом деления **PREDIV2**. Эти биты можно менять только тогда, когда **PLL2** и **PLL3** отключены.

- **0000:** PREDIV2 input clock not divided
- **0001:** PREDIV2 input clock divided by 2
- **0010:** PREDIV2 input clock divided by 3
- **0011:** PREDIV2 input clock divided by 4
- **0100:** PREDIV2 input clock divided by 5
- **0101:** PREDIV2 input clock divided by 6
- **0110:** PREDIV2 input clock divided by 7
- **0111:** PREDIV2 input clock divided by 8
- **1000:** PREDIV2 input clock divided by 9
- **1001:** PREDIV2 input clock divided by 10
- **1010:** PREDIV2 input clock divided by 11
- **1011:** PREDIV2 input clock divided by 12
- **1100:** PREDIV2 input clock divided by 13
- **1101:** PREDIV2 input clock divided by 14
- **1110:** PREDIV2 input clock divided by 15
- **1111:** PREDIV2 input clock divided by 16

Bits 3:0 PREDIV1[3:0]: PREDIV1 division factor (Коэффициент деления для PREDIV1)

Set and cleared by software to select PREDIV1 division factor. These bits can be written only when PLL is disabled.

Note: Bit(0) is the same as bit(17) in the RCC_CFGR register, so modifying bit(17) in the RCC_CFGR register changes Bit(0) accordingly.

Ставятся и очищаются программно, чтобы управлять коэффициентом деления **PREDIV1**. Эти биты можно менять только тогда, когда **PLL** отключен.

Note: **Bit(0)** - это тот же самый бит, что и **bit(17)** в регистре **RCC_CFGR**, поэтому, изменение **bit(17)** в регистре **RCC_CFGR** изменяет и **bit(0)** соответственно.

- **0000:** PREDIV1 input clock not divided
- **0001:** PREDIV1 input clock divided by 2
- **0010:** PREDIV1 input clock divided by 3
- **0011:** PREDIV1 input clock divided by 4
- **0100:** PREDIV1 input clock divided by 5
- **0101:** PREDIV1 input clock divided by 6
- **0110:** PREDIV1 input clock divided by 7
- **0111:** PREDIV1 input clock divided by 8
- **1000:** PREDIV1 input clock divided by 9
- **1001:** PREDIV1 input clock divided by 10
- **1010:** PREDIV1 input clock divided by 11
- **1011:** PREDIV1 input clock divided by 12
- **1100:** PREDIV1 input clock divided by 13
- **1101:** PREDIV1 input clock divided by 14
- **1110:** PREDIV1 input clock divided by 15
- **1111:** PREDIV1 input clock divided by 16

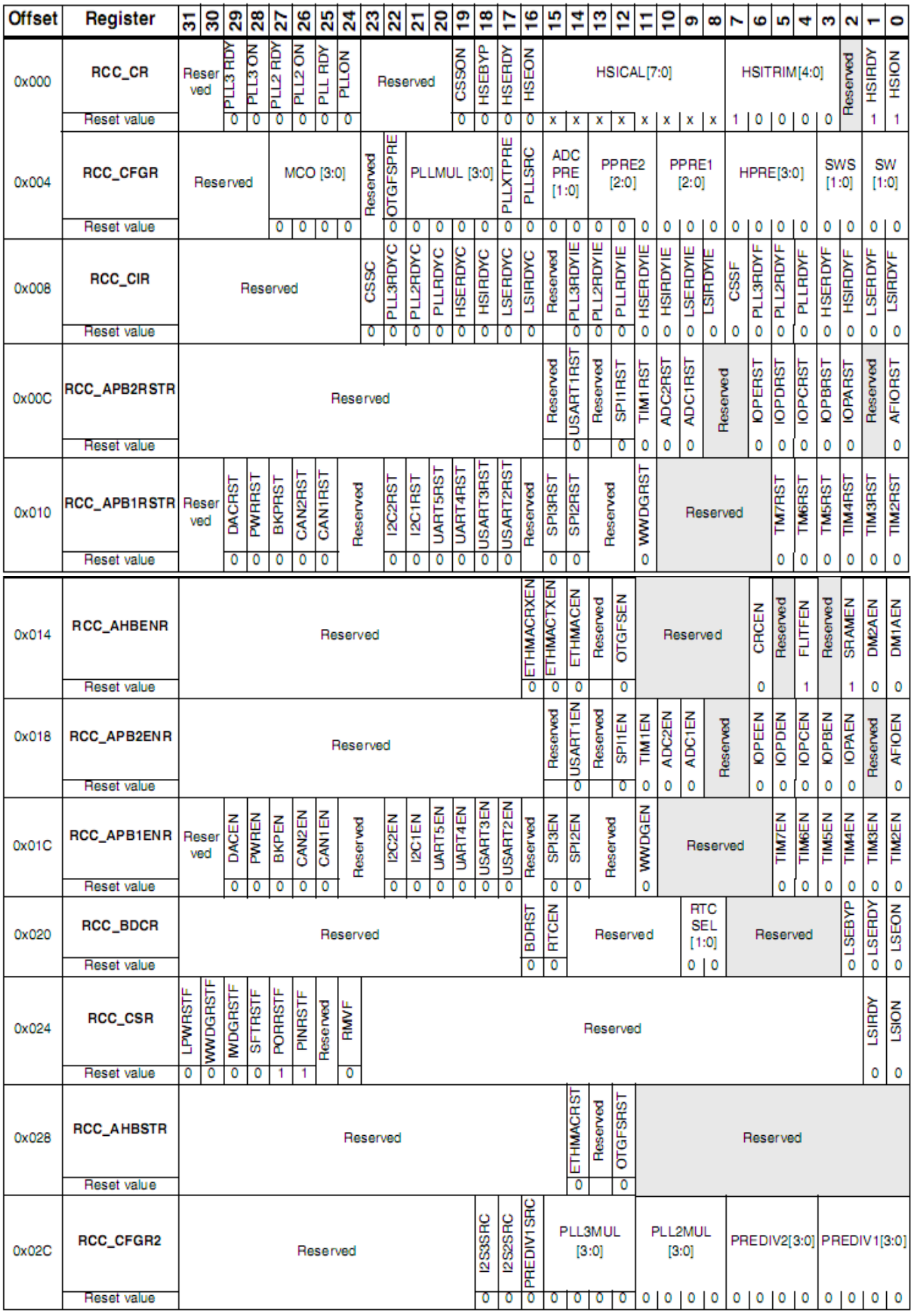
7.3.13 RCC register map (Карта регистров RCC)

The following table gives the RCC register map and the reset values.

В таблице ниже дана карта регистров **RCC** и их значение после сброса.

Table 16. RCC register map and reset values

Карта регистров **RCC** и их значение после сброса



Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RCC_CR	Reserved	PLL3RDY	PLL3ON	PLL2RDY	PLL2ON	PLLRDY	PLLON	Reserved					CSSON	HSEBYP	HSERDY	HSEON	HSICAL[7:0]								HSITRIM[4:0]					Reserved	HSIRDY	HSION
	Reset value		0	0	0	0	0	0						0	0	0	0	x	x	x	x	x	x	x	x	1	0	0	0	0		1	1
0x004	RCC_CFGR	Reserved				MCO [3:0]				Reserved	OTGFSPRE	PLLMUL [3:0]				PLLXTPRE	PLLSRC	ADC PRE [1:0]		PPRE2 [2:0]			PPRE1 [2:0]			HPRE[3:0]				SWS [1:0]		SW [1:0]	
	Reset value					0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	RCC_CIR	Reserved								CSSC	PLL3RDYC	PLL2RDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	Reserved		PLL3RDYIE	PLL2RDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	PLL3RDYF	PLL2RDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	Reset value									0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	RCC_APB2RSTR	Reserved																Reserved		USART1RST	Reserved	SPI1RST	TIM1RST	ADC2RST	ADC1RST	Reserved	IOPERST	IOPDRST	IOPCRST	IOPBRST	IOPARST	Reserved	AFIORST	
	Reset value																		0			0	0	0	0		0	0	0	0	0		0	
0x010	RCC_APB1RSTR	Reserved	DACRST	PWRRST	BKPRST	CAN2RST	CAN1RST	Reserved		I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Reserved		SPI3RST	SPI2RST	Reserved		WWDGRST	Reserved				TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST		TIM2RST	
	Reset value		0	0	0	0	0			0	0	0	0	0	0			0	0			0					0	0	0	0	0		0	
0x014	RCC_AHBENR	Reserved															ETHMACRXEN	ETHMACTXEN	ETHMACEN	Reserved	OTGFSEN	Reserved				CRCEN	Reserved	FLITFEN	Reserved	SRAMEN	DM2AEN	DM1AEN		
	Reset value																0	0	0		0					0		1		1	0	0		
0x018	RCC_APB2ENR	Reserved																Reserved		USART1EN	Reserved	SPI1EN	TIM1EN	ADC2EN	ADC1EN	Reserved	IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	Reserved	AFIOEN	
	Reset value																		0		0	0	0	0			0	0	0	0	0		0	
0x01C	RCC_APB1ENR	Reserved	DACEN	PWREN	BKPEN	CAN2EN	CAN1EN	Reserved		I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Reserved		SPI3EN	SPI2EN	Reserved		WWDGEN	Reserved				TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN		TIM2EN	
	Reset value		0	0	0	0	0			0	0	0	0	0	0			0	0			0					0	0	0	0	0		0	
0x020	RCC_BDCR	Reserved																BDRST	RTCEN	Reserved			RTC SEL [1:0]		Reserved						LSEBYP	LSERDY	LSEON	
	Reset value																	0	0				0	0							0	0	0	
0x024	RCC_CSR	LPWRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	Reserved	RMVF	Reserved																						LSIRDY	LSION	
	Reset value	0	0	0	0	1	1		0																							0	0	
0x028	RCC_AHBSTR	Reserved																	ETHMACRST	Reserved	OTGFSRST	Reserved												
	Reset value																		0		0													
0x02C	RCC_CFGR2	Reserved													I2S3SRC	I2S2SRC	PREDIV1SRC	PLL3MUL [3:0]				PLL2MUL [3:0]				PREDIV2[3:0]				PREDIV1[3:0]			
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to Table 1 on page 41 for the register boundary addresses.

См. Табл. 1, где даны относительные адреса регистров.

8 General-purpose and alternate-function I/Os (GPIOs and AFIOs)
Основные и альтернативные функции I/O

(*Здесь находится ритуальное напоминание о наличии 4-х семейств для **STM32F10xxx**, и что этот документ обо всех семействах, если нет прямого указания. См. раздел* 1.2 Глоссарий*).*)

8.1 GPIO functional description (Функциональное описание **GPIO**)
8.2 GPIO registers (Регистры **GPIO**)
8.3 Alternate function I/O and debug configuration (AFIO)
Альтернативные функции **I/O** и конфигурация порта отладки
8.4 AFIO registers (Регистры **AFIO**)
8.5 GPIO and AFIO register maps (Карта регистров **GPIO** и **AFIO**)

8.1 GPIO functional description (Функциональное описание GPIO)

8.1.1 General-purpose I/O (GPIO) (**I/O** общего назначения (**GPIO**))
8.1.2 Atomic bit set or reset (Атомарный сброс и установка бита)
8.1.3 External interrupt/wakeup lines (Линии внешнего прерывания/пробуждения)
8.1.4 Alternate functions (AF) (Альтернативные функции)
8.1.5 Software remapping of I/O alternate functions (Перназначение альтернативных функций)
8.1.6 GPIO locking mechanism (Механизм блокировки **GPIO**)
8.1.7 Input configuration (Конфигурация вывода как "Вход")
8.1.8 Output configuration (Конфигурация вывода как "Выход")
8.1.9 Alternate function configuration (Конфигурация вывода как "Альтернативная функция")
8.1.10 Analog input configuration (Конфигурация вывода как "Аналоговый вход")
8.1.11 Peripherals' GPIO configurations (Конфигурация **GPIO** для интерфейсов чипа)

Each of the general-purpose I/O ports has two 32-bit configuration registers (GPIOx_CRL, GPIOx_CRH), two 32-bit data registers (GPIOx_IDR, GPIOx_ODR), a 32-bit set/reset register (GPIOx_BSRR), a 16-bit reset register (GPIOx_BRR) and a 32-bit locking register (GPIOx_LCKR).
Каждый из портов **I/O** общего назначения имеет два 32-х разрядных регистра конфигурации (**GPIOx_CRL**, **GPIOx_CRH**), два 32-х разрядных регистра данных (**GPIOx_IDR**, **GPIOx_ODR**), 32-х разрядный регистр установки/сброса (**GPIOx_BSRR**), 16-ти разрядный регистр сброса (**GPIOx_BRR**) и 32-х разрядный регистр блокировки (**GPIOx_LCKR**).

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the General Purpose IO (GPIO) Ports, can be individually configured by software in several modes:
Аппаратные характеристики каждого **I/O** порта приведены в таблице в **datasheet**. Каждый бит **I/O** порта общего назначения (**GPIO**) может быть индивидуально сконфигурирован программой в один из нескольких режимов:

– Input floating ("Плавающий" вход)
– Input pull-up (Вход, притянутый к цепи питания)
– Input-pull-down (Вход, притянутый к общей цепи)
– Analog Input (Аналоговый вход)
– Output open-drain (Выход с открытым коллектором)
– Output push-pull (Двухтактный выход)
– Alternate function push-pull (Двухтактный выход альтернативной функции)
-- Alternate function open-drain (Выход с открытым коллектором альтернативной функции)

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words (half-word or byte accesses are not allowed). The purpose of the GPIOx_BSRR and GPIOx_BRR registers is to allow atomic read/modify accesses to any of the GPIO registers. This way,

there is no risk that an IRQ occurs between the read and the modify access.
Каждый бит **I/O** порта программируется независимо, однако, к регистрам порта нужно обращаться 32-х разрядными словами (полусловный или байтовый доступ не разрешен). Назначение регистров **GPIOx_BSRR** и **GPIOx_BRR** состоит в том, чтобы обеспечить атомарный доступ чтения-модификации к любому из **GPIO** регистров. При этом способе нет никакого риска, что произойдет запрос на прерывание между чтением и модификацией.

Figure 13 shows the basic structure of an I/O Port bit.
Рисунок 13 показывает блок-схему для одного бита **I/O** порта.

Figure 13. Basic structure of a standard I/O port bit (Блок-схема стандартного **I/O** порта)

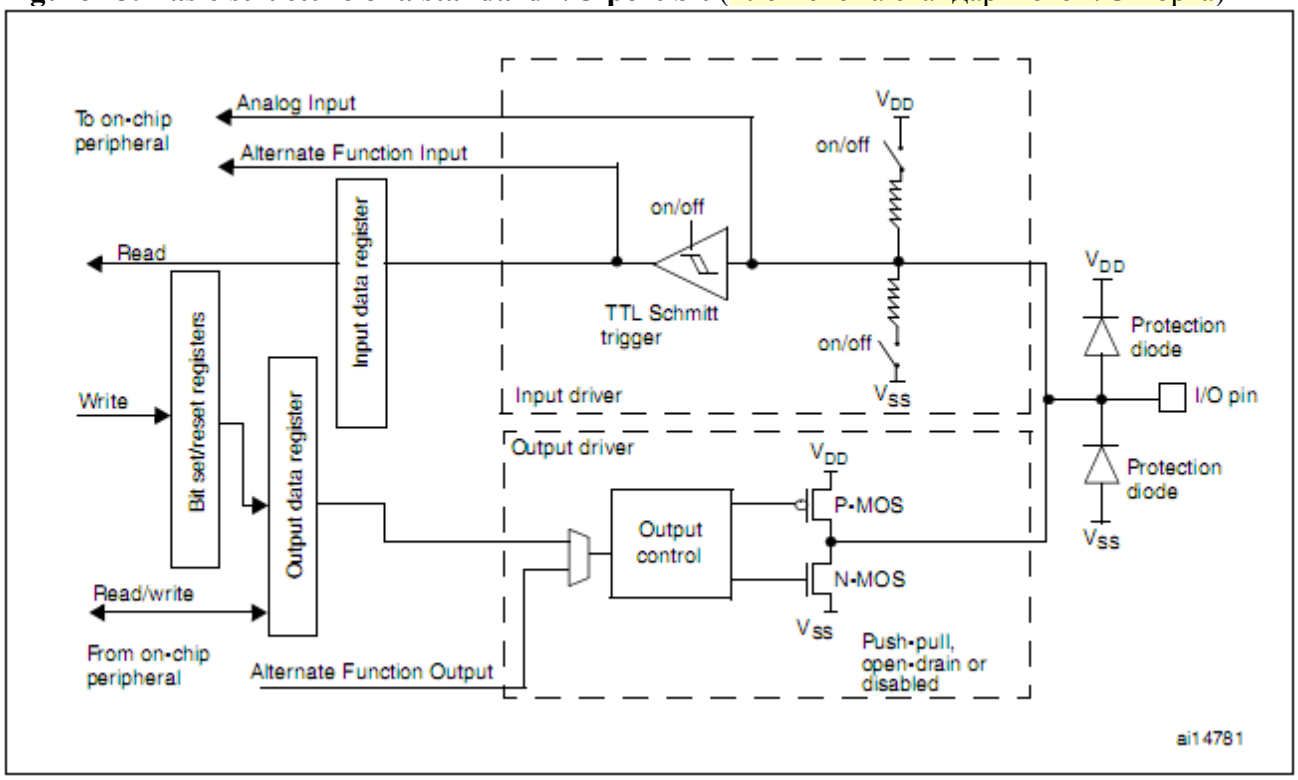
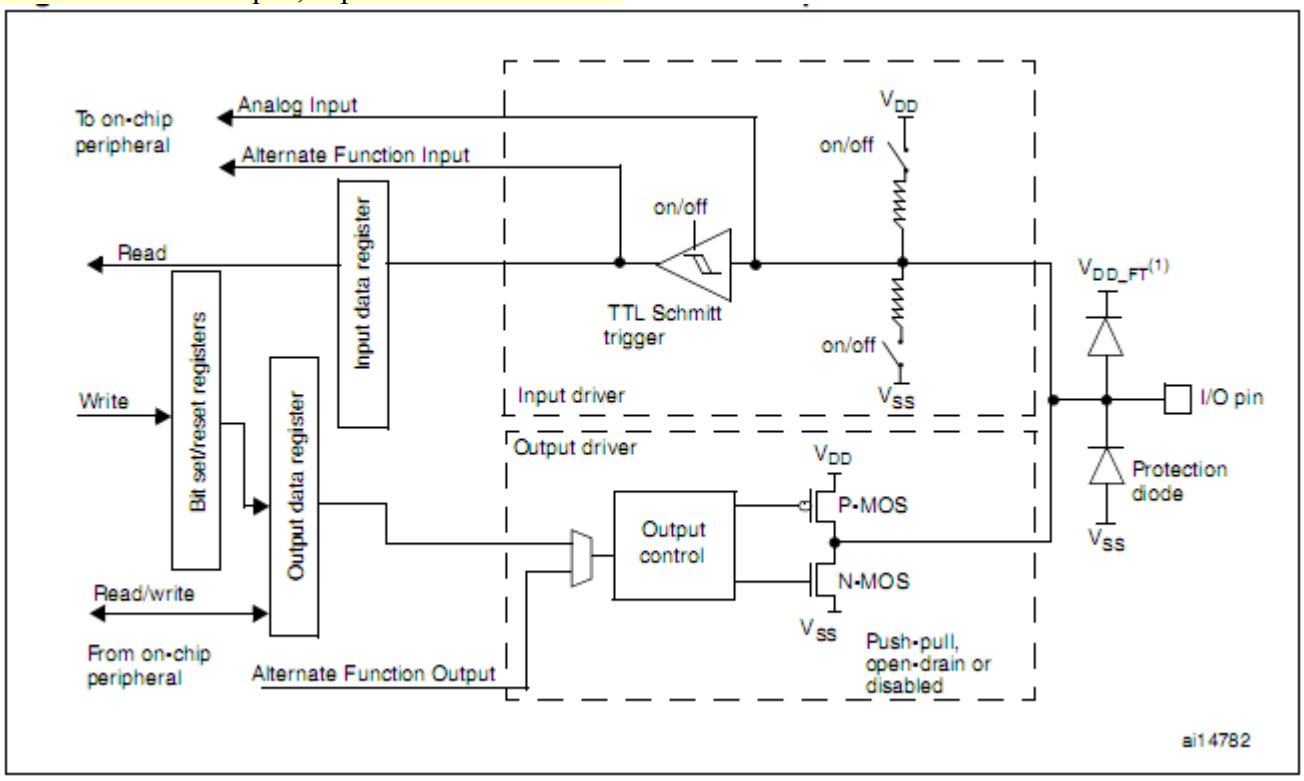


Figure 14. Basic structure of a five-volt tolerant I/O port bit
Блок-схема **I/O** порта, терпимого к 5 вольтам



1. VDD_FT is a potential specific to five-volt tolerant I/Os and different from VDD.
Цепь **VDD_FT** специфична для **I/O**, терпимых к напряжению 5 В, и она отличается от **VDD**.

Table 17. Port bit configuration table (Биты конфигурации порта)

Configuration mode		CNF1	CNF0	MODE1	MODE0	PxODR register
General purpose output	Push-pull	0	0	01 10 11 see *Table 18*		0 or 1
	Open-drain		1			0 or 1
Alternate Function output	Push-pull	1	0			don't care
	Open-drain		1			don't care
Input	Analog input	0	0	00		don't care
	Input floating		1			don't care
	Input pull-down	1	0			0
	Input pull-up					1

Table 18. Output MODE bits (Биты режимов выхода)

MODE[1:0]	Meaning
00	Reserved
01	Max. output speed 10 MHz
10	Max. output speed 2 MHz
11	Max. output speed 50 MHz

8.1.1 General-purpose I/O (GPIO) (I/O общего назначения)

During and just after reset, the alternate functions are not active and the I/O ports are configured in Input Floating mode (CNFx[1:0]=01b, MODEx[1:0]=00b).
В процессе сброса и сразу после него, альтернативные функции не являются активными, и выводы **I/O** портов сконфигурированы как "плавающие" входы (*Input Floating*) (**CNFx[1:0]=01b**, **MODEx[1:0]=00b**).

The JTAG pins are in input PU/PD after reset:
Выводы интерфейса **JTAG** находятся в режимах **input PU/PD** после сброса.

PA15: JTDI in PU (Притянутый вверх)
PA14: JTCK in PD (Притянутый вниз)
PA13: JTMS in PU (Притянутый вверх)
PB4: JNTRST in PU (Притянутый вверх)

When configured as output, the value written to the Output Data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in Push-Pull mode or Open-Drain mode (only the N-MOS is activated when outputting 0). При конфигурировании выводов как выходов, значение, записанное в регистр выходных данных **GPIOx_ODR**, поступает на выход. Выходной драйвер можно использовать в двухтактном режиме (*Push-Pull*) или в режиме с открытым коллектором (*Open-Drain*) (при выводе 0 активизируется только нижний n-МОП транзистор ключа).

The Input Data register (GPIOx_IDR) captures the data present on the I/O pin at every APB2 clock cycle. Регистр входных данных **GPIOx_IDR** защелкивает данные, представленные на

входных выводах на каждом тактовом цикле шины **APB2**.

All GPIO pins have an internal weak pull-up and weak pull-down which can be activated or not when configured as input. Все выводы **GPIO** имеют внутренние резисторы, подтягивающие цепь вверх или вниз, которые можно активировать, когда вывод конфигурируется как вход.

8.1.2 Atomic bit set or reset (Атомарный сброс и установка бита)

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: it is possible to modify only one or several bits in a single atomic APB2 write access. This is achieved by programming to ‘1’ the Bit Set/Reset Register (GPIOx_BSRR, or for reset only GPIOx_BRR) to select the bits you want to modify. The unselected bits will not be modified.
Нет никакой необходимости в программном отключении прерываний на время программирования **GPIOx_ODR** на уровне битов: вполне возможно изменить один или несколько битов в единственном атомарном доступе на запись к шине **APB2**. Это достигается установкой в '1' нужных битов в регистрах установки/сброса (регистр **GPIOx_BSRR**, или, если нужен только сброс, то **GPIOx_BRR**), чтобы выбрать те биты, которые вы хотите изменить. Невыбранные биты не будут модифицироваться.

8.1.3 External interrupt/wakeup lines (Линии внешнего прерывания/пробуждения)

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode. For more information on external interrupts, refer to:
Выводы всех портов можно использовать как входы внешних прерываний. Чтобы использовать вывод в этом режиме, он должен быть сконфигурирован как вход. Для получения дополнительной информации о внешних прерываниях, обратитесь к:

- *Section 9.2: External interrupt/event controller (EXTI) on page 174* and
Разделу 9.2 "***External interrupt/event controller (EXTI)***"
- *Section 9.2.3: Wakeup event management on page 175*.
Разделу 9.2.3 ***"Wakeup event management"***

8.1.4 Alternate functions (AF) (Альтернативные функции)

It is necessary to program the Port Bit Configuration Register before using a default alternate function. Прежде чем использовать альтернативную функцию по умолчанию, необходимо программировать регистр конфигурации битов порта.

- For alternate function inputs, the port must be configured in Input mode (floating, pull-up or pull-down) and the input pin must be driven externally.
Для альтернативных входных функций порт должен быть сконфигурирован в режиме альтернативного входа (плавающего, подтянутого вверх или притянутого вниз), и на вход должен подаваться внешний сигнал.

Note: It is also possible to emulate the AFI input pin by software by programming the GPIO controller. In this case, the port should be configured in Alternate Function Output mode. And obviously, the corresponding port should not be driven externally as it will be driven by the software using the GPIO controller.
***Note**: Возможно также программно эмулировать сигнал на альтернативном входе, программируя контроллер **GPIO**. В этом случае, порт должен быть конфигурирован как альтернативный выход. И очевидно, что на соответствующий вывод не должен подаваться внешний сигнал, поскольку он будет управляться программно, с помощью контроллера **GPIO**.*

- For alternate function outputs, the port must be configured in Alternate Function Output mode (Push-Pull or Open-Drain).
Для альтернативных выходных функций порт должен быть сконфигурирован в режиме альтернативного выхода (двухтактного или с открытым коллектором).

- For bidirectional Alternate Functions, the port bit must be configured in Alternate Function Output mode (Push-Pull or Open-Drain). In this case the input driver is configured in input floating mode
Для двунаправленных альтернативного функций порт должен быть сконфигурирован также в режиме альтернативного выхода (двухтактного или с открытым коллектором). В этом случае конфигурируется и входной драйвер на режим плавающего входа.

If you configure a port bit as Alternate Function Output, this disconnects the output register and connects the pin to the output signal of an on-chip peripheral.
Если вы конфигурируете бит порта для альтернативной выходной функции, это отключит его от регистра выходных данных и подключит к выходному сигналу периферии чипа.

If software configures a GPIO pin as Alternate Function Output, but peripheral is not activated, its output is not specified.
Если программа конфигурирует **GPIO** вывод для альтернативной выходной функции, но периферия не активирована, то ее выходной сигнал не определен.

8.1.5 Software remapping of I/O alternate functions
Программное перназначение альтернативных функций

To optimize the number of peripheral I/O functions for different device packages, it is possible to remap some alternate functions to some other pins. This is achieved by software, by programming the corresponding registers (refer to *AFIO registers on page 158*. In that case, the alternate functions are no longer mapped to their original assignations.
Чтобы оптимизировать число интерфейсов, подключенных к чипам в различных корпусах, возможно переназначить некоторые альтернативные функции на другие выводы. Это достигается программно, конфигурацией соответствующих регистров (обратитесь к разделу 8.4 "Регистры **AFIO**". В этом случае, альтернативные функции больше не подключены к их исходным выводам.

8.1.6 GPIO locking mechanism (Механизм блокировки GPIO)

The locking mechanism allows the IO configuration to be frozen. When the LOCK sequence has been applied on a port bit, it is no longer possible to modify the value of the port bit until the next reset.
Механизм блокировки позволяет "заморозить" конфигурацию **I/O** портов. Когда, к биту порта была применена последовательность процедуры блокировки, то его конфигурацию больше нельзя изменить до следующего сброса.

8.1.7 Input configuration (Конфигурация вывода как "Вход")

When the I/O Port is programmed as Input: (Когда **I/O** порт программируется как "Вход":)

- The Output Buffer is disabled. (Отключается буфер выходных данных.)

- The Schmitt Trigger Input is activated. (Активизируется входной триггер Шмидта.)

- The weak pull-up and pull-down resistors are activated or not depending on input configuration (pull-up, pull-down or floating).

В зависимости от конфигурации входа может активироваться **pull-up** и/или **pull-down** резистор.

- The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle.

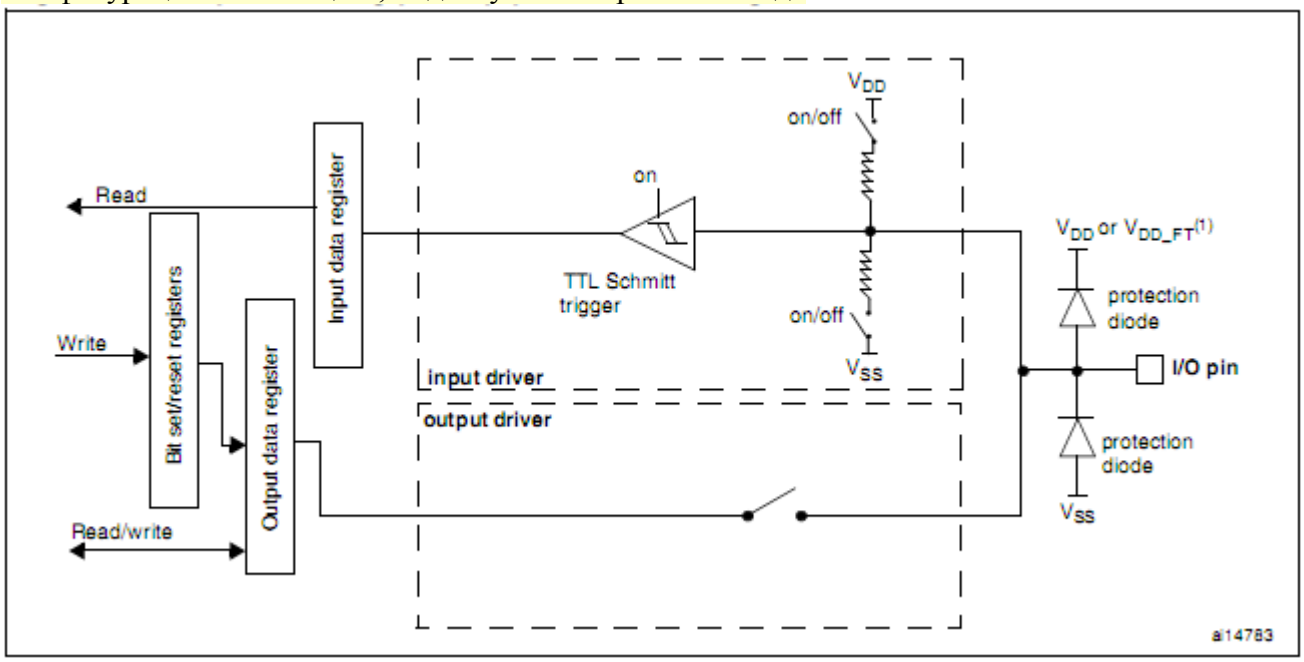
Данные, представленные на **I/O** выводе, защелкиваются в регистр входных данных (**IDR**) каждый такт шины **APB2**.

- A read access to the Input Data Register obtains the I/O State.

Доступ чтения к регистру **IDR** возвращает состояние **I/O** выводов порта.

The *Figure 15 on page 142* shows the Input Configuration of the I/O Port bit.
Рис. 15 показывает конфигурацию вывода **I/O** порта как входа.

Figure 15. Input floating/pull up/pull down configurations
Конфигурация плавающего, подтянутого вверх/вниз входа



1. VDD_FT is a potential specific to five-volt tolerant I/Os and different from VDD.
Цепь **VDD_FT** специфична для **I/O**, терпимых к напряжению 5 В, и она отличается от **VDD**.

8.1.8 Output configuration (Конфигурация вывода как "Выход")

When the I/O Port is programmed as Output: (Когда **I/O** порт программируется как "Выход":)

- The Output Buffer is enabled: (Подключается буфер выходных данных)
  - Open Drain Mode: A “0” in the Output register activates the N-MOS while a “1” in the Output register leaves the port in Hi-Z. (the P-MOS is never activated)
    В режиме "с открытым коллектором": значение '0' в регистре выходных данных (**ODR**) активизирует транзистор n-МОП ключа, а значение '1' оставляет ключ порта в высоко-импедансном состоянии (транзистор p-МОП никогда не активизируется).
  - Push-Pull Mode: A “0” in the Output register activates the N-MOS while a “1” in the Output register activates the P-MOS.
    В режиме "двухтактный": значение '0' в регистре выходных данных (**ODR**) активизирует транзистор n-МОП ключа, а значение '1' - p-МОП.

- The Schmitt Trigger Input is activated. (Активизируется входной триггер Шмидта.)

- The weak pull-up and pull-down resistors are disabled.
pull-up и **pull-down** резисторы отключаются.

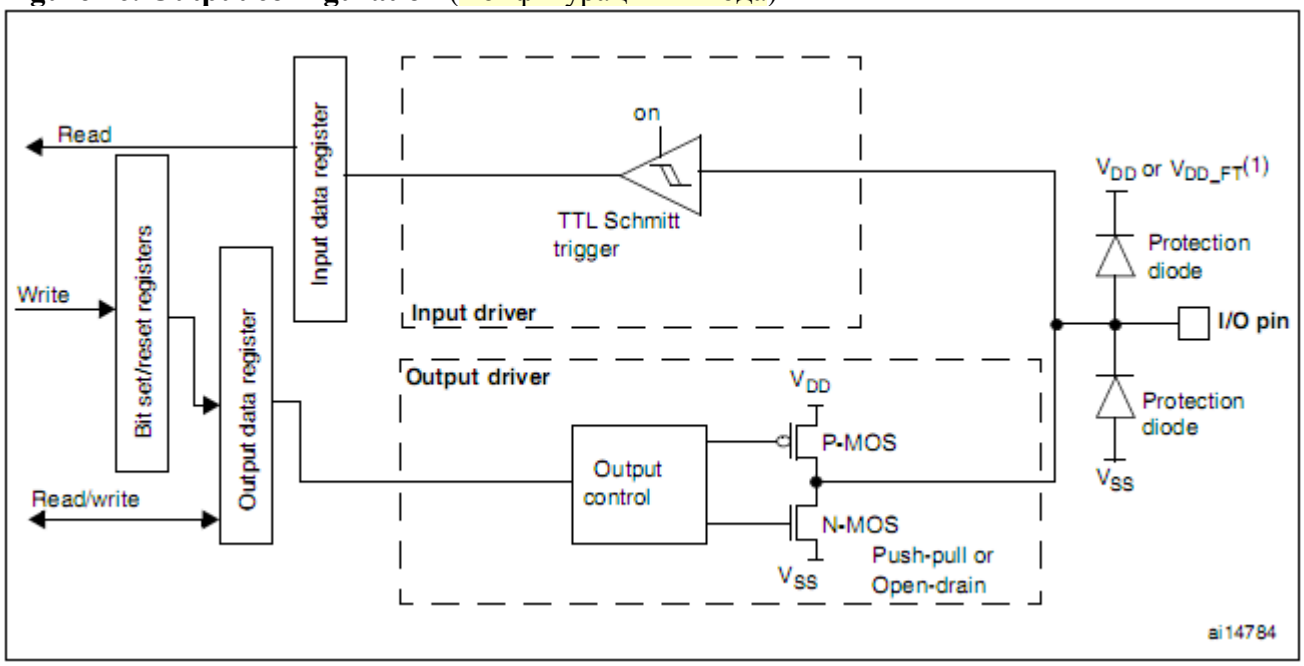
- The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle. Данные, представленные на **I/O** выводе, защелкиваются в регистр входных данных (**IDR**) каждый такт шины **APB2**.

- A read access to the Input Data Register gets the I/O state in open drain mode
Доступ чтения к регистру **IDR** возвращает состояние **I/O** выводов порта в режиме "с открытым коллектором".

- A read access to the Output Data register gets the last written value in Push-Pull mode
Доступ чтения к регистру **ODR** возвращает последнее записанное в него значение в режиме "двухтактный".

The *Figure 16 on page 143* shows the Output configuration of the I/O Port bit.
Рис. 16 показывает конфигурацию вывода **I/O** порта как выхода.

Figure 16. Output configuration (Конфигурация выхода)



1. VDD_FT is a potential specific to five-volt tolerant I/Os and different from VDD.
Цепь **VDD_FT** специфична для **I/O**, терпимых к напряжению 5 В, и она отличается от **VDD**.

8.1.9 Alternate function configuration
Конфигурация вывода как "Альтернативная функция"

When the I/O Port is programmed as Alternate Function:
Когда **I/O** порт программируется как "Альтернативная функция":

- The Output Buffer is turned on in Open Drain or Push-Pull configuration
Подключается буфер выходных данных в режиме "с открытым коллектором" или "двухтактный"

- The Output Buffer is driven by the signal coming from the peripheral (alternate function out) Буфер выходных данных управляется сигналом, приходящим из интерфейса (функция альтернативного выхода)

- The Schmitt Trigger Input is activated (Активизируется входной триггер Шмидта)

- The weak pull-up and pull-down resistors are disabled.
pull-up и **pull-down** резисторы отключаются.

- The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle Данные, представленные на **I/O** выводе, защелкиваются в регистр входных данных (**IDR**) каждый такт шины **APB2**

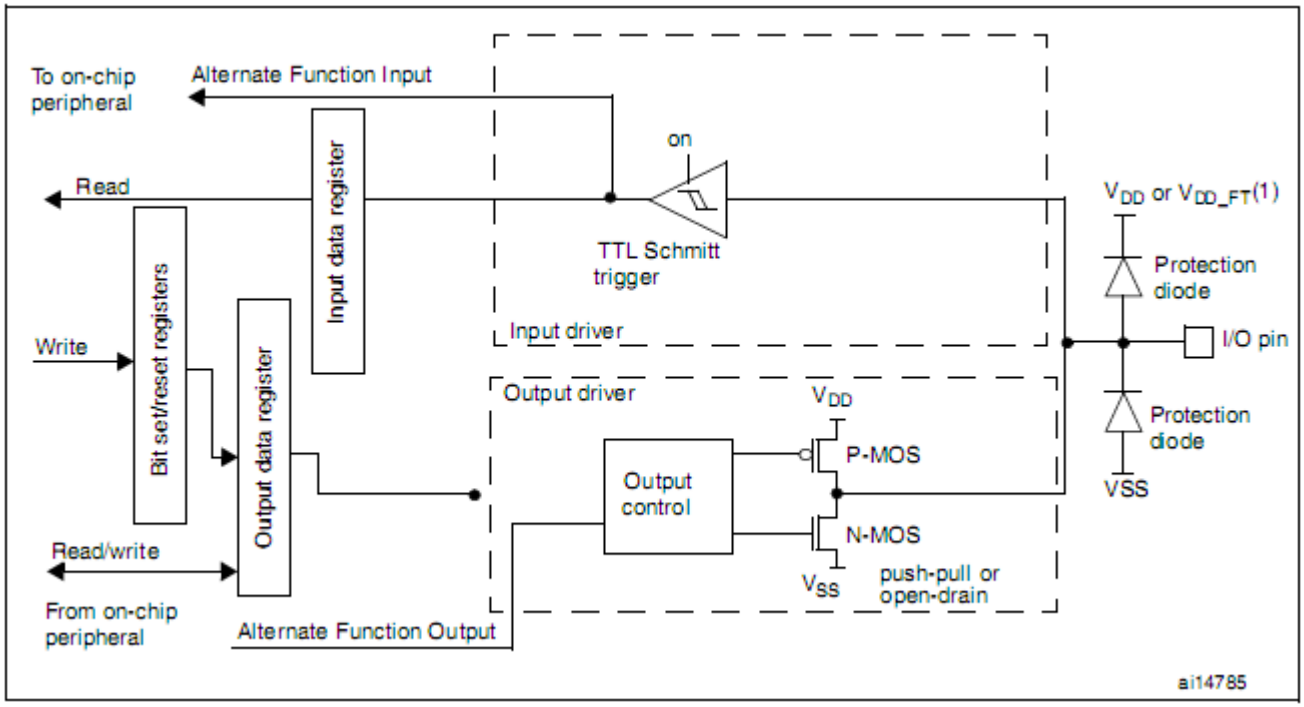
- A read access to the Input Data Register gets the I/O state in open drain mode
Доступ чтения к регистру **IDR** возвращает состояние **I/O** выводов порта в режиме "с открытым коллектором".

- A read access to the Output Data register gets the last written value in Push-Pull mode
Доступ чтения к регистру **ODR** возвращает последнее записанное в него значение в режиме "двухтактный".

The *Figure 17 on page 144* shows the Alternate Function Configuration of the I/O Port bit. Also, refer to *Section 8.4: AFIO registers on page 158* for further information.
Рис. 17 показывает конфигурацию вывода **I/O** порта как альтернативной функции. Кроме того, обратитесь к разделу 8.4 "Регистры **AFIO**" за дополнительной информацией.

A set of Alternate Function I/O registers allow you to remap some alternate functions to different pins. Refer to ..?? (*Ребята торопились очень...*)
Набор регистров альтернативных функций позволяет вам переназначать некоторые альтернативные функции на другие выводы.

Figure 17. Alternate function configuration (Конфигурация альтернативной функции)



1. VDD_FT is a potential specific to five-volt tolerant I/Os and different from VDD.
Цепь **VDD_FT** специфична для **I/O**, терпимых к напряжению 5 В, и она отличается от **VDD**.

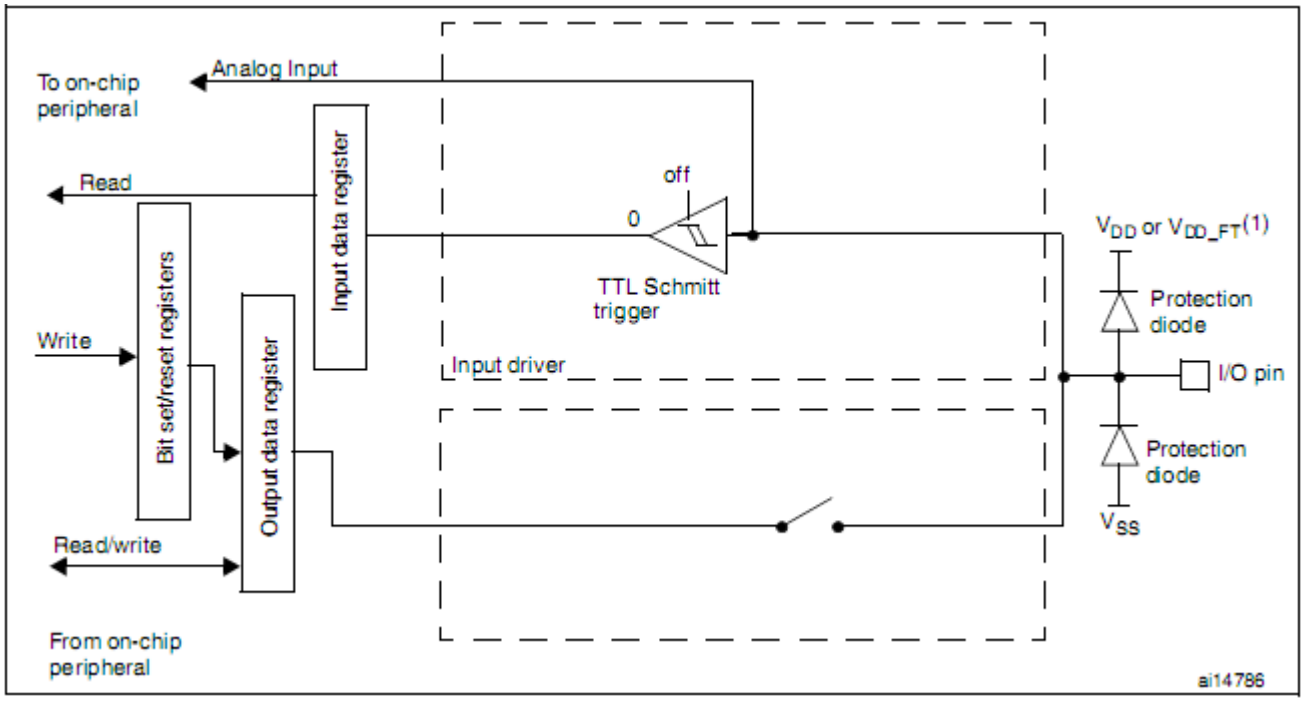
8.1.10 Analog input configuration (Конфигурация вывода как "Аналоговый вход")

When the I/O Port is programmed as Analog Input Configuration:
Когда **I/O** порт программируется как "Аналоговый вход":

- The Output Buffer is disabled. (Отключается буфер выходных данных.)
- The Schmitt Trigger Input is de-activated providing zero consumption for every analog value of the I/O pin. The output of the Schmitt Trigger is forced to a constant value (0). Деактивируется входной триггер Шмидта, обеспечивая нулевое потребление для каждого аналогового входа.
- The weak pull-up and pull-down resistors are disabled.
pull-up и **pull-down** резисторы отключаются.
- Read access to the Input Data Register gets the value “0”.
Доступ чтения к регистру **IDR** возвращает '0'.

The Figure 18 on page 144 shows the High impedance-Analog Input Configuration of the I/O Port bit.
Рис. 18 показывает конфигурацию вывода **I/O** порта как "Высоко-импедансный Аналоговый вход"

Figure 18. High impedance-analog input configuration
Конфигурация высоко-импедансного аналогового входа



8.1.11 Peripherals' GPIO configurations
Конфигурация GPIO для интерфейсов чипа

Table 19 to Table 29 give the GPIO configurations of the device peripherals.
Таблицы, с 19-й по 29-ю, показывают конфигурацию выводов **GPIO** для интерфейсов чипа.

Table 19. Advanced timers TIM1/TIM8 (Улучшенные таймеры)

TIM1/8 pinout	Configuration	GPIO configuration
TIM1/8_CHx	Input capture channel x (Вход захвата канала **x**)	Input floating
	Output compare channel x Выход сравнения с каналом **x**	Alternate function push-pull
TIM1/8_CHxN	Complementary output channel x Дополнительный выход канала **x**	Alternate function push-pull
TIM1/8_BKIN	Break input (Вход прерывания счета)	Input floating
TIM1/8_ETR	External trigger timer input Вход внешнего запуска таймера	Input floating

Table 20. General-purpose timers TIM2/3/4/5 (Таймеры общего назначения)

TIM2/3/4 pinout	Configuration	GPIO configuration
TIM2/3/4/5_CHx	Input capture channel x (Вход захвата канала **x**)	Input floating
	Output compare channel x Выход сравнения с каналом **x**	Alternate function push-pull
TIM2/3/4/5_ETR	External trigger timer input Вход внешнего запуска таймера	Input floating

Table 21. USARTs

USART pinout	Configuration	GPIO configuration
USARTx_TX	Full duplex	Alternate function push-pull
	Half duplex synchronous mode	Alternate function push-pull
USARTx_RX	Full duplex	Input floating / Input pull-up
	Half duplex synchronous mode	Not used. Can be used as a general IO
USARTx_CK	Synchronous mode	Alternate function push-pull
USARTx_RTS	Hardware flow control	Alternate function push-pull
USARTx_CTS	Hardware flow control	Input floating/ Input pull-up

Table 22. SPI

SPI pinout	Configuration	GPIO configuration
SPIx_SCK	Master	Alternate function push-pull
	Slave	Input floating
SPIx_MOSI	Full duplex / Master	Alternate function push-pull
	Full duplex / slave	Input floating / Input pull-up
	Simplex bidirectional data wire / master	Alternate function push-pull
	Simplex bidirectional data wire / slave	Not used. Can be used as a GPIO
SPIx_MISO	Full duplex / Master	Input floating / Input pull-up
	Full duplex / slave	Alternate function push-pull
	Simplex bidirectional data wire / master	Not used. Can be used as a GPIO
	Simplex bidirectional data wire / slave	Alternate function push-pull
SPIx_NSS	Hardware Master / Slave	Input floating/ Input pull-up / Input pull-down
	Hardware Master/ NSS output enabled	Alternate function push-pull
	Software	Not used. Can be used as a GPIO

Table 23. I2S

I2S pinout	Configuration	GPIO configuration
I2Sx_WS	Master	Alternate function push-pull
	Slave	Input floating
I2Sx_CK	Master	Alternate function push-pull
	Slave	Input floating
I2Sx_SD	Transmitter	Alternate function push-pull
	Receiver	Input floating/ Input pull-up/ Input pull-down
I2Sx_MCK	Master	Alternate function push-pull
	Slave	Not used. Can be used as a GPIO

Table 24. I2C interface

I2C pinout	Configuration	GPIO configuration
I2Cx_SCL	I2C clock	Alternate function open drain
I2Cx_SDA	I2C Data I/O	Alternate function open drain

Table 25. BxCAN

BxCAN pinout	GPIO configuration
CAN_TX (Transmit data line)	Alternate function push-pull
CAN_RX (Receive data line)	Input floating / Input pull-up

Table 26. USB

USB pinout	GPIO configuration
USB_DM / USB_DP	As soon as the USB is enabled, these pins are connected to the USB internal transceiver automatically. (Как только есть разрешение **USB**, эти выводы автоматически подключается к внутреннему приемо-передатчику **USB**.)

Table 27. SDIO

SDIO pinout	GPIO configuration
SDIO_CK	Alternate function push-pull
SDIO_CMD	Alternate function push-pull
SDIO[D7:D0]	Alternate function push-pull

Figure 19. ADC / DAC

ADC/DAC pinout	GPIO configuration
ADC/DAC	Analog input

Table 28. FSMC

FSMC pinout	GPIO configuration	FSMC pinout	GPIO configuration
FSMC_A[25:0] FSMC_D[15:0]	Alternate function push-pull	FSMC_NWAIT FSMC_CD	Input floating/ Input pull-up
FSMC_CK	Alternate function push-pull	FSMC_NIOS16, FSMC_INTR FSMC_INT[3:2]	Input floating
FSMC_NOE FSMC_NWE	Alternate function push-pull	FSMC_NL FSMC_NBL[1:0]	Alternate function push-pull
FSMC_NE[4:1] FSMC_NCE[3:2] FSMC_NCE4_1 FSMC_NCE4_2	Alternate function push-pull	FSMC_NIORD, FSMC_NIOWR FSMC_NREG	Alternate function push-pull

Table 29. Other IOs (Другие входы/выходы)

Pins	Alternate function	GPIO configuration
TAMPER-RTC pin	RTC output	Forced by hardware when configuring the BKP_CR and BKP_RTCCR registers Ставятся аппаратно при конфигурированиии регистров **BKP_CR** и **BKP_RTCCR**
	Tamper event input	
MCO	Clock output	Alternate function push-pull
EXTI input lines	External input interrupts	Input floating / input pull-up / input pull-down

8.2 GPIO registers (Регистры GPIO)

8.2.1 Port configuration register low (GPIOx_CRL) (x=A..G)
Регистр конфигурации младшей половины порта
8.2.2 Port configuration register high (GPIOx_CRH) (x=A..G)
Регистр конфигурации старшей половины порта
8.2.3 Port input data register (GPIOx_IDR) (x=A..G) (Регистр входных данных)
8.2.4 Port output data register (GPIOx_ODR) (x=A..G) (Регистр выходных данных)
8.2.5 Port bit set/reset register (GPIOx_BSRR) (x=A..G) (Регистр установки/сброса битов)
8.2.6 Port bit reset register (GPIOx_BRR) (x=A..G) (Регистр сброса битов)
8.2.7 Port configuration lock register (GPIOx_LCKR) (x=A..G)
Регистр блокировки конфигурации порта

Refer to *Section 1.1 on page 37* for a list of abbreviations used in register descriptions.
См. раздел 1.1, где приведен перечень сокращений, используемых при описании регистров.

8.2.1 Port configuration register low (GPIOx_CRL) (x=A..G)
Регистр конфигурации младшей половины порта

Address offset: 0x00
Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30, 27:26, 23:22, 19:18, 15:14, 11:10, 7:6, 3:2 **CNFy[1:0]:** Port x configuration bits (y= 0 .. 7) Биты конфигурирования порта **x**
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 17: Port bit configuration table on page 140*.
Эти биты пишутся программно, чтобы конфигурировать соответствующий порт.
См. Табл. 17 "Конфигурация битов порта".

In input mode (MODE[1:0]=00):

- **00:** Analog input mode
- **01:** Floating input (reset state)
- **10:** Input with pull-up / pull-down
- **11:** Reserved

In output mode (MODE[1:0] > 00):

- **00:** General purpose output push-pull
- **01:** General purpose output Open-drain
- **10:** Alternate function output Push-pull
- **11:** Alternate function output Open-drain

Bits 29:28, 25:24, 21:20, 17:16, 13:12, 9:8, 5:4, 1:0 **MODEy[1:0]:** Port x mode bits (y= 0 .. 7) (Биты режима порта **x**.)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 17: Port bit configuration table on page 140*.
Эти биты пишутся программно, чтобы конфигурировать соответствующий порт.
См. Табл. 17 "Конфигурация битов порта".

- **00:** Input mode (reset state)
- **01:** Output mode, max speed 10 MHz.
- **10:** Output mode, max speed 2 MHz.
- **11:** Output mode, max speed 50 MHz.

8.2.2 Port configuration register high (GPIOx_CRH) (x=A..G)
Регистр конфигурации старшей половины порта

Address offset: 0x04
Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30, 27:26, 23:22, 19:18, 15:14, 11:10, 7:6, 3:2 **CNFy[1:0]:** Port x configuration bits (y= 0 .. 7)
Биты конфигурирования порта **x**
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 17: Port bit configuration table on page 140*.
In input mode (MODE[1:0]=00):
Эти биты пишутся программно, чтобы конфигурировать соответствующий порт.
См. Табл. 17 "Конфигурация битов порта".
- **00:** Analog input mode
- **01:** Floating input (reset state)
- **10:** Input with pull-up / pull-down
- **11:** Reserved

In output mode (MODE[1:0] > 00):
- **00:** General purpose output push-pull
- **01:** General purpose output Open-drain
- **10:** Alternate function output Push-pull
- **11:** Alternate function output Open-drain

Bits 29:28, 25:24, 21:20, 17:16, 13:12, 9:8, 5:4, 1:0 **MODEy[1:0]:** Port x mode bits (y= 0 .. 7) (Биты режима порта **x**)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 17: Port bit configuration table on page 140*.
Эти биты пишутся программно, чтобы конфигурировать соответствующий порт.
См. Табл. 17 "Конфигурация битов порта".
- **00:** Input mode (reset state)
- **01:** Output mode, max speed 10 MHz.
- **10:** Output mode, max speed 2 MHz.
- **11:** Output mode, max speed 50 MHz.

8.2.3 Port input data register (GPIOx_IDR) (x=A..G) (Регистр входных данных)

Address offset: 0x08
Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved
always read as 0. (Всегда читается как '0'.)

Bits 15:0 IDRy[15:0]: Port input data (y= 0 .. 15) (Порт входных данных)
These bits are read only and can be accessed in Word mode only. They contain the input value of the corresponding I/O port.
Это биты только для чтения и обращаться к ним можно только пословно. Они содержат входные значения соответствующего **I/O** порта.

8.2.4 Port output data register (GPIOx_ODR) (x=A..G) (Регистр выходных данных)

Address offset: 0x0C
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved
always read as 0. (Всегда читается как '0'.)

Bits 15:0 IDRy[15:0]: Port output data (y= 0 .. 15) (Порт выходных данных)
These bits can be read and written by software and can be accessed in Word mode only.
Эти биты можно читать и писать программно и обращаться к ним можно только пословно

Note: For atomic bit set/reset, the ODR bits can be individually set and cleared by writing to the GPIOx_BSRR register (x = A .. G).
Для атомарной операции сброса/установки, биты регистра **ODR** можно индивидуально ставить и очищать записью в регистр **GPIOx_BSRR** (где **x = A .. G**).

8.2.5 Port bit set/reset register (GPIOx_BSRR) (x=A..G)
Регистр установки/сброса битов

Address offset: 0x10
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 BRy: Port x Reset bit y (y= 0 .. 15) (Сброс бита y порта x)
These bits are write-only and can be accessed in Word mode only.
Это биты только для записи и обращаться к ним можно только пословно.
`0:` No action on the corresponding ODRx bit (Нет воздействия на бит **ODRx**)
`1:` Reset the corresponding ODRx bit (Сброс соответствующего бита **ODRx**)
Note: If both BSx and BRx are set, BSx has priority.
Если установлены оба бита, **BSx** и **BRx**, то приоритет имеет **BSx**.

Bits 15:0 BSy: Por t x Set bit y (y= 0 .. 15) (Установка бита y порта x)
These bits are write-only and can be accessed in Word mode only.
Это биты только для записи и обращаться к ним можно только пословно.
`0:` No action on the corresponding ODRx bit (Нет воздействия на бит **ODRx**)
`1:` Set the corresponding ODRx bit (Установка соответствующего бита **ODRx**)

8.2.6 Port bit reset register (GPIOx_BRR) (x=A..G) (Регистр сброса битов)

Address offset: 0x14
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved
always read as 0. (Всегда читается как '0'.)

Bits 15:0 BRy: Port x Reset bit y (y= 0 .. 15) (Сброс бита y порта x)
These bits are write-only and can be accessed in Word mode only.
Это биты только для записи и обращаться к ним можно только пословно.
`0:` No action on the corresponding ODRx bit (Нет воздействия на бит **ODRx**)
`1:` Reset the corresponding ODRx bit (Сброс соответствующего бита **ODRx**)

8.2.7 Port configuration lock register (GPIOx_LCKR) (x=A..G)
Регистр блокировки конфигурации порта

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit it is no longer possible to modify the value of the port bit until the next reset.
Этот регистр используется для блокировки конфигурации битов порта, когда применена правильная последовательность записи к биту 16 (**LCKK**). Для блокировки конфигурации **GPIO** используется значение битов [15:0]. Во время последовательности записи значение **LCKR** [15:0] не должно изменяться. Когда последовательность процедуры блокировки была применена к биту порта, то его конфигурацию больше нельзя изменить до следующего сброса.

Each lock bit freezes the corresponding 4 bits of the control register (CRL, CRH).
Каждый бит блокировки "замораживает" 4 соответствующих бита регистра управления (**CRL**, **CRH**).

Address offset: 0x18
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved
Bit 16 LCKK[16]: Lock key (Ключ блокировки)

This bit can be read anytime. It can only be modified using the Lock Key Writing Sequence.
Этот бит можно читать в любое время. Его можно модифицировать только правильной последовательностью записи.

0: Port configuration lock key not active (Блокировка не активирована)
1: Port configuration lock key active. GPIOx_LCKR register is locked until an MCU reset occurs. (Ключ блокировки конфигурации порта активирован. Регистр **GPIOx_LCKR** заблокирован до сброса ЦПУ.)

LOCK key writing sequence: (последовательностью записи блокировки)
Write 1, Write 0, Write 1, Read 0, Read 1 (this read is optional but confirms that the lock is active
это чтение необязательно, но оно подтверждает, что блокировка активирована.)

Note: During the LOCK Key Writing sequence, the value of LCK[15:0] must not change.
Any error in the lock sequence will abort the lock.
Note: В процессе последовательности записи блокировки значение **LCK[15:0]** не должно изменяться. Любая ошибка в процессе последовательности приведет к прерыванию процедуры.

Bits 15:0 LCKy: Port x Lock bit y (y= 0 .. 15) (Блокировка бита y порта x)

These bits are read write but can only be written when the LCKK bit is 0.
Это биты для чтения и записи, но запись в них будет только тогда, когда бит **LCKK = 0**.

0: Port configuration not locked (Нет блокировки конфигурации порта)
1: Port configuration locked. (Конфигурация порта заблокирована)

8.3 Alternate function I/O and debug configuration (AFIO)
Альтернативные функции I/O и конфигурация порта отладки

8.3.1 Using OSC32_IN/OSC32_OUT pins as GPIO ports PC14/PC15
Использование выводов **OSC32_IN/OSC32_OUT** как **GPIO** порты **PC14/PC15**
8.3.2 Using OSC_IN/OSC_OUT pins as GPIO ports PD0/PD1
Использование выводов **OSC_IN/OSC_OUT** как **GPIO** порты **PD0/PD1**
8.3.3 CAN1 alternate function remapping (Переназначение альтернативной функции **CAN1**)
8.3.4 CAN2 alternate function remapping (Переназначение альтернативной функции **CAN2**)
8.3.5 JTAG/SWD alternate function remapping
(Переназначение альтернативной функции **JTAG/SWD**)
8.3.6 ADC alternate function remapping (Переназначение альтернативной функции **ADC**)
8.3.7 Timer alternate function remapping (Переназначение альтернативной функции **Timer**)
8.3.8 USART Alternate function remapping (Переназначение альтернативной функции **USART**)
8.3.9 I2C1 alternate function remapping (Переназначение альтернативной функции **I2C1**)
8.3.10 SPI1 alternate function remapping (Переназначение альтернативной функции **SPI1**)
8.3.11 SPI3 alternate function remapping (Переназначение альтернативной функции **SPI3**)
8.3.12 Ethernet alternate function remapping
(Переназначение альтернативной функции **Ethernet**)

To optimize the number of peripherals available for the 64-pin or the 100-pin or the 144-pin package, it is possible to remap some alternate functions to some other pins. This is achieved by software, by programming the *AF remap and debug I/O configuration register (AFIO_MAPR) on page 159*. In this case, the alternate functions are no longer mapped to their original assignations.
Чтобы оптимизировать число интерфейсов, подключенных к чипам в различных корпусах на 64, 100 или 144 вывода, возможно переназначить некоторые альтернативные функции на другие выводы. Это достигается программно, конфигурацией регистра AFIO_MAPR. В этом случае, альтернативные функции больше не подключены к их исходным выводам.

8.3.1 Using OSC32_IN/OSC32_OUT pins as GPIO ports PC14/PC15
Использование выводов OSC32_IN/OSC32_OUT как GPIO порты PC14/PC15

The LSE oscillator pins OSC32_IN and OSC32_OUT can be used as general-purpose I/O PC14 and PC15, respectively, when the LSE oscillator is off. The LSE has priority over the GPIOs function.
Выводы **OSC32_IN** и **OSC32_OUT LSE** генератора можно использовать как **I/O PC14** и **PC15** общего назначения соответственно, когда **LSE** генератор выключен. У **LSE** есть приоритет над функцией **GPIO**.

Note:

1. *The PC14/PC15 GPIO functionality is lost when the 1.8 V domain is powered off (by entering standby mode) or when the backup domain is supplied by VBAT (VDD no more supplied). In this case the IOs are set in analog input mode.*
   *Функциональные возможности **PC14/PC15 GPIO** теряются, когда отключают домены, использующие напряжение 1.8 В (вход в режим **Standby**) или когда домен **BKP** питается от **VBAT** (**VDD** отсутствует). В этом случае эти выводы установливаются в режим "Аналоговый вход".*

2. *Refer to the note on IO usage restrictions in Section 4.1.2 on page 54.*
   *См. примечание раздела 4.1.2 относительно ограничений использования **I/O**.*

8.3.2 Using OSC_IN/OSC_OUT pins as GPIO ports PD0/PD1
Использование выводов OSC_IN/OSC_OUT как GPIO порты PD0/PD1

The HSE oscillator pins OSC_IN/OSC_OUT can be used as general-purpose I/O PD0/PD1 by programming the PD01_REMAP bit in the *AF remap and debug I/O configuration register (AFIO_MAPR)*.
Выводы **OSC_IN/OSC_OUT HSE** генератора можно использовать как **I/O PD0/PD1** общего назначения, программируя бит **PD01_REMAP** в регистре AFIO_MAPR.

This remap is available only on 36-, 48- and 64-pin packages (PD0 and PD1 are available on 100-pin and 144-pin packages, no need for remapping).
Это переназначение возможно только для корпусов на 36, 48 и 64 выводов (выводы **PD0** и **PD1** доступны в корпусах на 100 и 144 выводов, и не нуждаются в переназначении).

***Note**: The external interrupt/event function is not remapped. PD0 and PD1 cannot be used for external interrupt/event generation on 36-, 48- and 64-pin packages.*
***Note**: Функция внешнего прерывания/события не переназначается. Выводы **PD0** и **PD1** нельзя использовать для генерации внешнего прерывания/события в корпусах на 36, 48 и 64 вывода.*

8.3.3 CAN1 alternate function remapping
Переназначение альтернативной функции CAN1

The CAN signals can be mapped on Port A, Port B or Port D as shown in Table 30. For port D, remapping is not possible in devices delivered in 36-, 48- and 64-pin packages.
Сигналы порта **CAN** можно назначить на порт **A**, **B** или **D**, как показано в таблице 30. Переназначение для порта **D** невозможно в корпусах на 36, 48 и 64 вывода.

Table 30. CAN1 alternate function remapping (Переназначение альтернативной функции **CAN1**)

Alternate function(1)	**CAN_REMAP[1:0] = “00”**	**CAN_REMAP[1:0] = “10”**(2)	**CAN_REMAP[1:0] = “11”**(3)
CAN1_RX or CAN_RX	PA11	PB8	PD0
CAN1_TX or CAN_RX	PA12	PB9	PD1

1. CAN1_RX and CAN1_TX in connectivity line devices; CAN_RX and CAN_TX in other devices with a single CAN interface.
CAN1_RX и **CAN1_TX** в семействе **Connectivity Line**; **CAN_RX** и **CAN_TX** в других семействах, где только один **CAN**.

2. Remap not available on 36-pin package (Переназначение невозможно в корпусе на 36 выводов.)

3. This remapping is available only on 100-pin and 144-pin packages, when PD0 and PD1 are not remapped on OSC-IN and OSC-OUT.
Это переназначение возможно только для корпусов на 100 и 144 вывода, когда нет переназначения **PD0** и **PD1** на **OSC-IN** и **OSC-OUT**.

8.3.4 CAN2 alternate function remapping
Переназначение альтернативной функции CAN2

CAN2 is available in connectivity line devices. The external signal can be remapped as shown in Chapter Table 31. (**CAN2** доступно в семействе **Connectivity Line**. Внешний сигнал можно переназначить, как показано в таблице 31.)

Table 31. CAN2 alternate function remapping (Переназначение альтернативной функции **CAN2**)

Alternate function	CAN2_REMAP = “0”	CAN2_REMAP = “1”
CAN2_RX	PA12	PB5
CAN2_TX	PA13	PB6

8.3.5 JTAG/SWD alternate function remapping
Переназначение альтернативной функции JTAG/SWD

The debug interface signals are mapped on the GPIO ports as shown in *Table 32*. Сигналы интерфейса отладчика назначены на **GPIO** порты, как показано в таблице 32.

Table 32. Debug interface signals (Сигналы интерфейса отладчика)

Alternate function	GPIO port
JTMS / SWDIO	PA13
JTCK / SWCLK	PA14
JTDI	PA15
JTDO / TRACESWO	PB3
JNTRST	PB4
TRACECK	PE2
TRACED0	PE3
TRACED1	PE4
TRACED2	PE5
TRACED3	PE6

To optimize the number of free GPIOs during debugging, this mapping can be configured in different ways by programming the SWJ_CFG[1:0] bits in the *AF remap and debug I/O configuration register (AFIO_MAPR)*. Refer to *Table 33*. Это назначение может быть сконфигурировано различным образом, чтобы оптимизировать число свободных **GPIO** во время отладки, с помощью битов **SWJ_CFG[1:0]** в регистре AFIO_MAPR. См. таблицу 33.

Table 33. Debug port mapping (Назначения выводов порта отладчика)

SWJ _CFG [2:0	Available debug ports	SWJ I/O pin assigned				
		PA13 / JTMS/ SWDIO	PA14 / JTCK/S WCLK	PA15 / JTDI	PB3 / JTDO/ TRACE SWO	PB4/ NJTRST
000	Full SWJ (JTAG-DP + SW-DP) (Reset state)	X	X	X	X	X
001	Full SWJ (JTAG-DP + SW-DP) but without JNTRST	X	X	X	x	free

010	JTAG-DP Disabled and SW-DP Enabled	X	X	free	free[1]	free
100	JTAG-DP Disabled and SW-DP Disabled	free	free	free	free	free
Other	Forbidden					

1. Released only if not using asynchronous trace.

Реализуется только в случае, когда не используется асинхронная трассировка.

8.3.6 ADC alternate function remapping

Переназначение альтернативной функции ADC

Refer to AF remap and debug I/O configuration register (AFIO_MAPR).

См. регистр переназначения AFIO_MAPR.

Table 34. ADC1 external trigger injected conversion alternate function remapping[1]

Переназначение альтернативной функции внешнего запуска **ADC** в режиме **injected conversion**

Alternate function	ADC1_ETRGINJ_REMAP = 0	ADC1_ETRGINJ_REMAP = 1
ADC1 external trigger injected conversion	ADC1 external trigger injected conversion is connected to EXTI15	ADC1 external trigger injected conversion is connected to TIM8_CH4

1. Remap available only for high-density devices.

Переназначение возможно только для семейства **High-Density**.

Table 35. ADC1 external trigger regular conversion alternate function remapping[1]

Переназначение альтернативной функции внешнего запуска **ADC1** в режиме **regular conversion**

Alternate function	ADC1_ETRGREG_REMAP = 0	ADC1_ETRGREG_REMAP = 1
ADC1 external trigger regular conversion	ADC1 external trigger regular conversion is connected to EXTI11	ADC1 external trigger regular conversion is connected to TIM8_CH0

1. Remap available only for high-density devices.

Переназначение возможно только для семейства **High-Density**.

Table 36. ADC2 external trigger injected conversion alternate function remapping[1]

Переназначение альтернативной функции внешнего запуска **ADC2** в режиме **injected conversion**

Alternate function	ADC2_ETRGINJ_REMAP = 0	ADC2_ETRGINJ_REMAP = 1
ADC2 external trigger injected conversion	ADC2 external trigger injected conversion is connected to EXTI15	ADC2 external trigger injected conversion is connected to TIM8_CH4

1. Remap available only for high-density devices.

Переназначение возможно только для семейства **High-Density**.

Table 37. ADC2 external trigger regular conversion alternate function remapping(1)
Переназначение альтернативной функции внешнего запуска **ADC2** в режиме **regular conversion**

Alternate function	ADC2_ETRGREG_REMAP = 0	ADC2_ETRGREG_REMAP = 1
ADC2 external trigger regular conversion	ADC2 external trigger regular conversion is connected to EXTI11	ADC2 external trigger regular conversion is connected to TIM8_CH0

1. Remap available only for high-density devices.
Переназначение возможно только для семейства **High-Density**.

8.3.7 Timer alternate function remapping
Переназначение альтернативной функции Timer

Timer 4 channels 1 to 4 can be remapped from Port B to Port D. Other timer remapping possibilities are listed in *Table 40* to *Table 42*. Refer to *AF remap and debug I/O configuration register (AFIO_MAPR)*. Каналы с 1-го по 4-й таймера 4 можно переназначить с порта **B** на порт **D**. Возможности переназначения для других таймеров перечислены в таблицах с 40 по 42. См. регистр переназначения AFIO_MAPR.

Table 38. TIM5 alternate function remapping(1)
Переназначение альтернативной функции **TIM5**

Alternate function	TIM5CH4_IREMAP = 0	TIM5CH4_IREMAP = 1
TIM5_CH4	TIM5 Channel4 is connected to PA3	LSI internal clock is connected to TIM5_CH4 input for calibration purpose.

1. Remap available only for high-density and connectivity line devices.
Переназначение возможно только для семейств **High-Density** и **Connectivity Line**.

Table 39. TIM4 alternate function remapping (Переназначение альтернативной функции **TIM4**)

Alternate function	TIM4_REMAP = 0	TIM4_REMAP = 1(1)
TIM4_CH1	PB6	PD12
TIM4_CH2	PB7	PD13
TIM4_CH3	PB8	PD14
TIM4_CH4	PB9	PD15

1. Remap available only for 100-pin and for 144-pin package.
Переназначение возможно только для корпусов на 100 и 144 вывода.

Table 40. TIM3 alternate function remapping (Переназначение альтернативной функции **TIM3**)

Alternate function	TIM3_REMAP[1:0] = “00” (no remap)	TIM3_REMAP[1:0] = “10” (partial remap)	TIM3_REMAP[1:0] = “11” (full remap)(1)
TIM3_CH1	PB6	PB4	PC6
TIM3_CH2	PB7	PB5	PC7
TIM3_CH3	PB0		PC8
TIM3_CH4	PB1		PC9

1. Remap available only for 64-pin, 100-pin and 144-pin packages.
Переназначение возможно только для корпусов на 64, 100 и 144 вывода.

Table 41. TIM2 alternate function remapping (Переназначение альтернативной функции **TIM2**)

Alternate function	TIM2_REMAP[1:0] = “00” (no remap)	TIM2_REMAP[1:0] = “01” (partial remap)	TIM2_REMAP[1:0] = “10” (partial remap)(1)	TIM2_REMAP[1:0] = “11” (full remap)(1)
TIM2_CH1_ETR (2)	PA0	PA15	PA0	PA15
TIM2_CH2	PA1	PA3	PA1	PA3
TIM2_CH3	PA2		PB10	
TIM2_CH4	PA3		PB11	

1. Remap not available on 36-pin package.
Переназначение возможно только для корпусов на 36 выводов.

2. TIM_CH1 and TIM_ETR share the same pin but cannot be used at the same time (which is why we have this notation: TIM2_CH1_ETR).
Выводы **TIM_CH1** и **TIM_ETR** разделяют один вывод и не могут использоваться одновременно (вот почему мы используем такую нотацию: **TIM2_CH1_ETR**).

Table 42. TIM1 alternate function remapping (Переназначение альтернативной функции **TIM1**)

Alternate function mapping	TIM1_REMAP[1:0] = “00” (no remap)	TIM1_REMAP[1:0] = “01” (partial remap)	TIM1_REMAP[1:0] = “11” (full remap)(1)
TIM1_ETR	PA12		PE7
TIM1_CH1	PA8		PE9
TIM1_CH2	PA9		PE11
TIM1_CH3	PA10		PE13
TIM1_CH4	PA11		PE14
TIM1_BKIN	PB12(2)	PA6	PE15
TIM1_CH1N	PB13(2)	PA7	PE8
TIM1_CH2N	PB14(2)	PB0	PE10
TIM1_CH3N	PB15(2)	PB1	PE12

1. Remap available only for 100-pin and 144-pin packages.
Переназначение возможно только для корпусов на 100 и 144 вывода.

2. Remap not available on 36-pin package.
Переназначение невозможно для корпусов на 36 выводов.

8.3.8 USART Alternate function remapping
Переназначение альтернативной функции USART

Refer to AF remap and debug I/O configuration register (AFIO_MAPR).
См. регистр переназначения AFIO_MAPR.

Table 43. USART3 remapping (Переназначение **USART3**)

Alternate function	USART3_REMAP[1:0] = “00” (no remap)	USART3_REMAP[1:0] = “01” (partial remap)(1)	USART3_REMAP[1:0] = “11” (full remap)(2)
USART3_TX	PB10	PC10	PD8
USART3_RX	PB11	PC11	PD9
USART3_CK	PB12	PC12	PD10
USART3_CTS	PB13		PD11
USART3_RTS	PB14		PD12

1. Remap available only for 64-pin, 100-pin and 144-pin packages.
Переназначение возможно только для корпусов на 64, 100 и 144 вывода.
2. Remap available only for 100-pin and 144-pin packages.
Переназначение возможно только для корпусов на 100 и 144 вывода.

Table 44. USART2 remapping (Переназначение **USART2**)

Alternate function	USART2_REMAP = 0	USART2_REMAP = 1(1)
USART2_CTS	PA0	PD3
USART2_RTS	PA1	PD4
USART2_TX	PA2	PD5
USART2_RX	PA3	PD6
USART2_CK	PA4	PD7

1. Remap available only for 100-pin and 144-pin packages.
Переназначение возможно только для корпусов на 100 и 144 вывода.

Table 45. USART1 remapping (Переназначение **USART1**)

Alternate function	USART1_REMAP = 0	USART1_REMAP = 1
USART1_TX	PA9	PB6
USART1_RX	PA10	PB7

8.3.9 I2C1 alternate function remapping
Переназначение альтернативной функции I2C1

Refer to AF remap and debug I/O configuration register (AFIO_MAPR)
См. регистр переназначения AFIO_MAPR.

Table 46. I2C1 remapping (Переназначение **I2C1**)

Alternate function	I2C1_REMAP = 0	I2C1_REMAP = 1(1)
I2C1_SCL	PB6	PB8
I2C1_SDA	PB7	PB9

1. Remap available only for 36-pin packages.
Переназначение возможно только для корпусов на 36 выводов.

8.3.10 SPI1 alternate function remapping
Переназначение альтернативной функции SPI1

Refer to AF remap and debug I/O configuration register (AFIO_MAPR)
См. регистр переназначения AFIO_MAPR.

Table 47. SPI1 remapping (Переназначение **SPI1**)

Alternate function	SPI1_REMAP = 0	SPI1_REMAP = 1
SPI1_NSS	PA4	PA15
SPI1_SCK	PA5	PB3
SPI1_MISO	PA6	PB4
SPI1_MOSI	PA7	PB5

8.3.11 SPI3 alternate function remapping
Переназначение альтернативной функции SPI3

Refer to AF remap and debug I/O configuration register (AFIO_MAPR). This remap is available only in connectivity line devices. (См. регистр переназначения AFIO_MAPR. Это переназначение возможно только для семейства **Connectivity Line**.)

Table 48. SPI3 remapping (Переназначение **SPI3**)

Alternate function	SPI3_REMAP = 0	SPI3_REMAP = 1
SPI3_NSS	PA15	PA4
SPI3_SCK	PB3	PC10
SPI3_MISO	PB4	PC11
SPI3_MOSI	PB5	PC12

8.3.12 Ethernet alternate function remapping
Переназначение альтернативной функции Ethernet

Refer to AF remap and debug I/O configuration register (AFIO_MAPR). This remap is available only in connectivity line devices. (См. регистр переназначения AFIO_MAPR. Это переназначение возможно только для семейства **Connectivity Line**.)

Table 49. ETH remapping (Переназначение **ETH**)

Alternate function	ETH_REMAP = 0	ETH_REMAP = 1
RX_DV-CRS_DV	PA7	PD8
RXD0	PC4	PD9
RXD1	PC5	PD10
RXD2	PB0	PD11
RXD3	PB1	PD12

8.4 AFIO registers (Регистры AFIO)

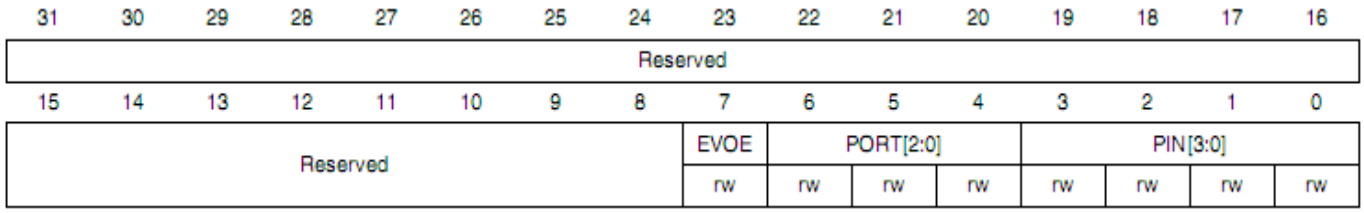
8.4.1 Event control register (AFIO_EVCR) (Регистр управления событиями)
8.4.2 AF remap and debug I/O configuration register (AFIO_MAPR) (Регистр переназначения)
8.4.3 External interrupt configuration register 1 (AFIO_EXTICR1)
(Регистр конфигурации прерываний 1)
8.4.4 External interrupt configuration register 2 (AFIO_EXTICR2)
(Регистр конфигурации прерываний 2)
8.4.5 External interrupt configuration register 3 (AFIO_EXTICR3)
(Регистр конфигурации прерываний 3)
8.4.6 External interrupt configuration register 4 (AFIO_EXTICR4)
(Регистр конфигурации прерываний 4)

Refer to *Section 1.1 on page 37* for a list of abbreviations used in register descriptions.
См. раздел 1.1, где приведен перечень сокращений, используемых при описании регистров.

***Note**: To read/write the AFIO_EVCR,AFIO_MAPR and AFIO_EXTICRX registers, the AFIO clock should first be enabled. Refer to Section 6.3.7: APB2 peripheral clock enable register (RCC_APB2ENR).*
***Note**: Чтобы производить операции чтения/записи регистров **AFIO_EVCR**, **AFIO_MAPR** и **AFIO_EXTICRX**, надо сначала разрешить тактовый сигнал для **AFIO**. См. раздел 6/7.3.7 "Регистр разрешения периферии **APB2**"*

8.4.1 Event control register (AFIO_EVCR) (Регистр управления событиями)

Address offset: 0x00
Reset value: 0x0000 0000



Bits 31:8 Reserved

Bit 7 EVOE: Event output enable (Разрешение вывода события)
Set and cleared by software. When set the EVENTOUT Cortex output is connected to the I/O selected by the PORT[2:0] and PIN[3:0] bits.
Ставится и очищается программно. Когда установлен бит **EVENTOUT**, то выход **Cortex** подключен к **I/O** выводу, выбранному битами **PORT[2:0]** и **PIN[3:0]**.

Bits 6:4 PORT[2:0]: Port selection (Выбор порта)
Set and cleared by software. Select the port used to output the Cortex EVENTOUT signal.
Ставится и очищается программно. Выбирает порт, который будет использоваться для вывода сигнала **Cortex EVENTOUT**.
Note: The EVENTOUT signal output capability is not extended to ports PF and PG.
Возможность вывода сигнала **Cortex EVENTOUT** не распространяется на порты **PF** и **PG**.
000: PA selected
001: PB selected
010: PC selected
011: PD selected
100: PE selected

Bits 3:0 PIN[3:0]: Pin selection (x = A .. E) (Выбор ввода порта)

Set and cleared by software. Select the pin used to output the Cortex EVENTOUT signal.
Ставится и очищается программно. Выбирает вывод порта, который будет использоваться для вывода сигнала **Cortex EVENTOUT**.

`0000:` Px0 selected
`0001:` Px1 selected
`0010:` Px2 selected
`0011:` Px3 selected
`...`
`1111:` Px15 selected

8.4.2 AF remap and debug I/O configuration register (AFIO_MAPR)

Регистр переназначения

Address offset: 0x04
Reset value: 0x0000 0000

Memory map and bit definitions for low-, medium- and high-density devices:
Определяет карту памяти и назначения битов для семейств **Low-**, **Medium-** и **High-Density**:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					SWJ_CFG[2:0]			Reserved			ADC2_ETRGREG_REMAP	ADC2_ETRGINJ_REMAP	ADC1_ETRGREG_REMAP	ADC1_ETRGINJ_REMAP	TIM5CH4_IREMAP
					w	w	w				rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_REMAP	CAN_REMAP[1:0]		TIM4_REMAP	TIM3_REMAP[1:0]		TIM2_REMAP[1:0]		TIM1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved

Bits 26:24 SWJ_CFG[2:0]: Serial wire JTAG configuration (Конфигурация цепей JTAG)

These bits are write-only (when read, the value is undefined). They are used to configure the SWJ and trace alternate function I/Os. The SWJ (Serial Wire JTAG) supports JTAG or SWD access to the Cortex debug port. The default state after reset is SWJ ON without trace. This allows JTAG or SW mode to be enabled by sending a specific sequence on the JTMS / JTCK pin.
Это биты только для записи (значение при чтении не определено). Они используются для конфигурирования альтернативных функций **SWJ** и трассировки. **SWJ** (Последовательный **JTAG**) поддерживает доступ к порту отладки **Cortex** по интерфейсу **JTAG**, или по **SWD**. После сброса порт отладки находится в режиме: **SWJ** включен без трассировки. Это позволяет разрешать режим **JTAG** или **SW**, посылая определенную последовательность на вывод **JTMS/JTCK**.

`000:` Full SWJ (JTAG-DP + SW-DP): Reset State
`001:` Full SWJ (JTAG-DP + SW-DP) but without JNTRST
`010:` JTAG-DP Disabled and SW-DP Enabled
`100:` JTAG-DP Disabled and SW-DP Disabled
`Other combinations:` no effect

Bits 23:21 Reserved

Bits 20 ADC2_ETRGREG_REMAP: ADC 2 external trigger regular conversion remapping
Переназначение вывода внешнего запуска ADC2 в режиме regular conversion

Set and cleared by software. This bit controls the trigger input connected to ADC2 external trigger regular conversion. When this bit is reset, the ADC2 external trigger regular conversion is connected to EXTI11. When this bit is set, the ADC2 external event regular conversion is connected to TIM8_TRGO.

Ставится и очищается программно. Этот бит управляет подключением входа запуска в режиме **regular conversion** для **ADC2**. Когда этот бит сброшен, то этот вход подключен к выводу **EXTI11**, иначе - к выводу **TIM8_TRGO**.

Bits 19 ADC2_ETRGINJ_REMAP: ADC 2 external trigger injected conversion remapping
Переназначение вывода внешнего запуска ADC2 в режиме injected conversion

Set and cleared by software. This bit controls the trigger input connected to ADC2 external trigger injected conversion. When this bit is reset, the ADC2 external trigger injected conversion is connected to EXTI15. When this bit is set, the ADC2 external event injected conversion is connected to TIM8_Channel4.

Ставится и очищается программно. Этот бит управляет подключением входа запуска в режиме **injected conversion** для **ADC2**. Когда этот бит сброшен, то этот вход подключен к выводу **EXTI15**, иначе - к выводу **TIM8_Channel4**.

Bits 18 ADC1_ETRGREG_REMAP: ADC 1 external trigger regular conversion remapping
Переназначение вывода внешнего запуска ADC1 в режиме regular conversion

Set and cleared by software. This bit controls the trigger input connected to ADC1 External trigger regular conversion. When reset the ADC1 External trigger regular conversion is connected to EXTI11. When set the ADC1 External Event regular conversion is connected to TIM8_TRGO.

Ставится и очищается программно. Этот бит управляет подключением входа запуска в режиме **regular conversion** для **ADC1**. Когда этот бит сброшен, то этот вход подключен к выводу **EXTI11**, иначе - к выводу **TIM8_TRGO**.

Bits 17 ADC1_ETRGINJ_REMAP: ADC 1 External trigger injected conversion remapping
Переназначение вывода внешнего запуска ADC1 в режиме injected conversion

Set and cleared by software. This bit controls the trigger input connected to ADC1 External trigger injected conversion. When reset the ADC1 External trigger injected conversion is connected to EXTI15. When set the ADC1 External Event injected conversion is connected to TIM8 Channel4.

Ставится и очищается программно. Этот бит управляет подключением входа запуска в режиме **injected conversion** для **ADC1**. Когда этот бит сброшен, то этот вход подключен к выводу **EXTI15**, иначе - к выводу **TIM8_Channel4**.

Bits 16 TIM5CH4_IREMAP: TIM5 channel4 internal remap
Переназначение вывода канала 4 таймера 5

Set and cleared by software. This bit controls the TIM5_CH4 internal mapping. When reset the timer TIM5_CH4 is connected to PA3. When set the LSI internal clock is connected to TIM5_CH4 input for calibration purpose.

Ставится и очищается программно. Этот бит управляет подключением входа **TIM5_CH4**. Когда этот бит сброшен, то этот вход подключен к выводу **PA3**. Когда этот бит установлен, то ко входу **TIM5_CH4** подключен внутренний тактовый сигнал от **LSI** в целях его калибровки.

Bit 15 PD01_REMAP: Port D0/Port D1 mapping on OSC_IN/OSC_OUT
Переназначение выводов PD0/PD1 на OSC_IN/OSC_OUT

This bit is set and cleared by software. It controls the mapping of PD0 and PD1 GPIO functionality. When the HSE oscillator is not used (application running on internal 8 MHz RC) PD0 and PD1 can be mapped on OSC_IN and OSC_OUT. This is available only on 36-, 48- and 64-pin packages (PD0 and PD1 are available on 100-pin and 144-pin packages, no need for remapping).
Этот бит ставится и очищается программно. Он управляет функциональным назначением выводов **PD0** и **PD1 GPIO**. Когда **HSE** генератор не используется (приложение работает от внутреннего 8 МГц **RC** генератора), то на место выводов **OSC_IN** и **OSC_OUT** можно назначить **PD0** и **PD1**. Это возможно только для корпусов на 36, 48 и 64 выводов (выводы **PD0** и **PD1** доступны в корпусах на 100 и 144 выводов, и не нуждаются в переназначении).

`0:` No remapping of PD0 and PD1 (Нет переназначения для **PD0** и **PD1**)
`1:` PD0 remapped on OSC_IN, PD1 remapped on OSC_OUT,
PD0 переназначен на **OSC_IN**, а **PD1** - на **OSC_OUT**.

Bits 14:13 CAN_REMAP[1:0]: CAN alternate function remapping
Переназначение альтернативной функции CAN

These bits are set and cleared by software. They control the mapping of alternate functions CAN_RX and CAN_TX in devices with a single CAN interface.
Эти биты ставятся и очищаются программно. Они управляют назначением альтернативных функций выводов **CAN_RX** и **CAN_TX** в устройствах с одним **CAN** интерфейсом.

`00:` CAN_RX mapped to PA11, CAN_TX mapped to PA12
CAN_RX назначен на **PA11**, **CAN_TX** - на **PA12**
`01:` Not used (Не использовать)
`10:` CAN_RX mapped to PB8, CAN_TX mapped to PB9 (not available on 36-pin package)
CAN_RX назначен на **PB8**, **CAN_TX** - на **PB9** (недоступно для корпусов на 36 выводов)
`11:` CAN_RX mapped to PD0, CAN_TX mapped to PD1
CAN_RX назначен на **PD0**, **CAN_TX** - на **PD1**

Bit 12 TIM4_REMAP: TIM4 remapping (Переназначение для TIM4)

This bit is set and cleared by software. It controls the mapping of TIM4 channels 1 to 4 onto the GPIO ports.
Этот бит ставится и очищается программно. Он управляет назначением выводов для каналов с 1-го по 4-й таймера **TIM4** на порты **GPIO**.

`0:` No remap (Нет переназначения)
(TIM4_CH1/PB6, TIM4_CH2/PB7, TIM4_CH3/PB8, TIM4_CH4/PB9)
`1:` Full remap (Полное переназначение)
(TIM4_CH1/PD12, TIM4_CH2/PD13, TIM4_CH3/PD14, TIM4_CH4/PD15)

Note: TIM4_ETR on PE0 is not re-mapped. (Вывод **TIM4_ETR/PE0** не переназначается.)

Bits 11:10 TIM3_REMAP[1:0]: TIM3 remapping (Переназначение для TIM3)

These bits are set and cleared by software. They control the mapping of TIM3 channels 1 to 4 on the GPIO ports.
Эти биты ставятся и очищаются программно. Они управляют назначением выводов для каналов с 1-го по 4-й таймера **TIM3** на порты **GPIO**.

`00:` No remap (Нет переназначения) (CH1/PA6, CH2/PA7, CH3/PB0, CH4/PB1)
`01:` Not used (Не использовать)
`10:` Partial remap (Частичное переназначение) (CH1/PB4, CH2/PB5, CH3/PB0, CH4/PB1)
`11:` Full remap (Полное переназначение) (CH1/PC6, CH2/PC7, CH3/PC8, CH4/PC9)

Note: TIM3_ETR on PE0 is not re-mapped. (Вывод **TIM3_ETR/PE0** не переназначается.)

Bits 9:8 TIM2_REMAP[1:0]: TIM2 remapping (Переназначение для TIM2)

These bits are set and cleared by software. They control the mapping of TIM2 channels 1 to 4 and external trigger (ETR) on the GPIO ports.

Эти биты ставятся и очищаются программно. Они управляют назначением выводов для таймера **TIM2**: каналы с 1-го по 4-й и вход внешнего запуска (**ETR**), на порты **GPIO**.

00: No remap (Нет переназначения) (CH1/ETR/PA0, CH2/PA1, CH3/PA2, CH4/PA3)

01: Partial remap (Частичное переназначение) (CH1/ETR/PA15, CH2/PB3, CH3/PA2, CH4/PA3)

10: Partial remap (Частичное переназначение) (CH1/ETR/PA0, CH2/PA1, CH3/PB10, CH4/PB11)

11: Full remap (Полное переназначение) (CH1/ETR/PA15, CH2/PB3, CH3/PB10, CH4/PB11)

Bits 7:6 TIM1_REMAP[1:0]: TIM1 remapping (Переназначение для TIM1)

These bits are set and cleared by software. They control the mapping of TIM2 channels 1 to 4, 1N to 3N, external trigger (ETR) and Break input (BKIN) on the GPIO ports.

Эти биты ставятся и очищаются программно. Они управляют назначением выводов для таймера **TIM1**: каналы 1-4, **1N-3N**, вход внешнего запуска (**ETR**) и вход прерывания счета (**BKIN**), на порты **GPIO**.

00: No remap (Нет переназначения) (ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PB12, CH1N/PB13, CH2N/PB14, CH3N/PB15)

01: Partial remap (Частичное переназначение) (ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PA6, CH1N/PA7, CH2N/PB0, CH3N/PB1)

10: not used (Не использовать)

11: Full remap (Полное переназначение) (ETR/PE7, CH1/PE9, CH2/PE11, CH3/PE13, CH4/PE14, BKIN/PE15, CH1N/PE8, CH2N/PE10, CH3N/PE12)

Bits 5:4 USART3_REMAP[1:0]: USART3 remapping (Переназначение для USART3)

These bits are set and cleared by software. They control the mapping of USART3 CTS, RTS, CK, TX and RX alternate functions on the GPIO ports.

Эти биты ставятся и очищаются программно. Они управляют назначением альтернативных функций выводов **CTS**, **RTS**, **CK**, **TX** и **RX** для **USART3** на порты **GPIO**.

00: No remap (Нет переназначения) (TX/PB10, RX/PB11, CK/PB12, CTS/PB13, RTS/PB14)

01: Partial remap (Частичное переназначение) (TX/PC10, RX/PC11, CK/PC12, CTS/PB13, RTS/PB14)

10: not used (Не использовать)

11: Full remap (Полное переназначение) (TX/PD8, RX/PD9, CK/PD10, CTS/PD11, RTS/PD12)

Bit 3 USART2_REMAP: USART2 remapping (Переназначение для USART2)

This bit is set and cleared by software. It controls the mapping of USART2 CTS, RTS, CK, TX and RX alternate functions on the GPIO ports.

Этот бит ставится и очищается программно. Он управляет назначением альтернативных функций выводов **CTS**, **RTS**, **CK**, **TX** и **RX** для **USART2** на порты **GPIO**.

0: No remap (Нет переназначения) (CTS/PA0, RTS/PA1, TX/PA2, RX/PA3, CK/PA4)

1: Remap (Переназначение) (CTS/PD3, RTS/PD4, TX/PD5, RX/PD6, CK/PD7)

Bit 2 USART1_REMAP: USART1 remapping (Переназначение для USART1)

This bit is set and cleared by software. It controls the mapping of USART1 TX and RX alternate functions on the GPIO ports. Этот бит ставится и очищается программно. Он управляет назначением альтернативных функций выводов **TX** и **RX** для **USART1** на порты **GPIO**.

0: No remap (Нет переназначения) (TX/PA9, RX/PA10)

1: Remap (Переназначение) (TX/PB6, RX/PB7)

Bit 1 I2C1_REMAP: I2C1 remapping (Переназначение для I2C1)

This bit is set and cleared by software. It controls the mapping of I2C1 SCL and SDA alternate functions on the GPIO ports.

Этот бит ставится и очищается программно. Он управляет назначением альтернативных функций выводов **SCL** и **SDA** для **I2C1** на порты **GPIO**.

`0:` No remap (Нет переназначения) (SCL/PB6, SDA/PB7)

`1:` Remap (Переназначение) (SCL/PB8, SDA/PB9)

Bit 0 SPI1_REMAP: SPI1 remapping (Переназначение для SPI1)

This bit is set and cleared by software. It controls the mapping of SPI1 NSS, SCK, MISO, MOSI alternate functions on the GPIO ports.

Этот бит ставится и очищается программно. Он управляет назначением альтернативных функций выводов **NSS**, **SCK**, **MISO**, **MOSI** для **SPI1** на порты **GPIO**.

`0:` No remap (Нет переназначения) (NSS/PA4, SCK/PA5, MISO/PA6, MOSI/PA7)

`1:` Remap (Переназначение) (NSS/PA15, SCK/PB3, MISO/PB4, MOSI/PB5)

Memory map and bit definitions for connectivity line devices:

Определяет карту памяти и назначения битов для семейства **Connectivity Line**.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PTP_PPS_REMAP	TIM2ITR1_IREMAP	SPI3_REMAP	Res.	SWJ_CFG[2:0]			MII_RMII_SEL	CAN2_REMAP	ETH_REMAP	Reserved				TIM5CH4_IREMAP
	rw	rw	rw		w	w	w	rw	rw	rw					rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_REMAP	CAN1_REMAP[1:0]		TIM4_REMAP	TIM3_REMAP[1:0]		TIM2_REMAP[1:0]		TIM1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved

Bit 30 PTP_PPS_REMAP: Ethernet PTP PPS remapping
Переназначение для Ethernet PTP PPS

This bit is set and cleared by software. It enables the Ethernet MAC PPS_PTS to be output on the PB5 pin. (Этот бит ставится и очищается программно. Он разрешает вывод сигнала **Ethernet MAC PPS_PTS** на вывод **PB5**.)

`0:` PTP_PPS not output on PB5 pin. (Нет вывода сигнала **PTP_PPS** на вывод **PB5**)

`1:` PTP_PPS is output on PB5 pin. (Сигнала **PTP_PPS** выводится на вывод **PB5**)

Note: This bit is available only in connectivity line devices and is reserved otherwise.
Этот бит доступен только для семейства **Connectivity Line**, в противном случает он зарезервирован.

Bit 29 TIM2ITR1_IREMAP: TIM2 internal trigger 1 remapping
Переназначение внутреннего подключения входа запуска 1 для TIM2

This bit is set and cleared by software. It controls the TIM2_ITR1 internal mapping.

Этот бит ставится и очищается программно. Он управляет внутренним подключением **TIM2_ITR1**.

`0:` Connect TIM2_ITR1 internally to the Ethernet PTP output for calibration purposes.
Внутренне подключает **TIM2_ITR1** к выходу **Ethernet PTP** в целях калибровки.

`1:` Connect USB OTG SOF (Start of Frame) output to TIM2_ITR1 for calibration purposes.
Внутренне подключает **TIM2_ITR1** к выходу **USB OTG SOF** в целях калибровки.

Note: This bit is available only in connectivity line devices and is reserved otherwise.
Этот бит доступен только для семейства **Connectivity Line**, в противном случает он зарезервирован.

Bit 28 SPI3_REMAP: SPI3 remapping (Переназначение для SPI3)

This bit is set and cleared by software. It controls the mapping of SPI3 NSS, SCK, MISO, MOSI alternate functions on the GPIO ports.

Этот бит ставится и очищается программно. Он управляет назначением альтернативных функций выводов **NSS**, **SCK**, **MISO**, **MOSI** для **SPI3** на порты **GPIO**.

`0:` No remap (Нет переназначения) (NSS/PA15, SCK/PB3, MISO/PB4, MOSI/PB5)

`1:` Remap (Переназначение) (NSS/PA4, SCK/PC10, MISO/PC11, MOSI/PC12)

Note: This bit is available only in connectivity line devices and is reserved otherwise.

Этот бит доступен только для семейства **Connectivity Line**, в противном случает он зарезервирован.

Bit 27 Reserved

Bits 26:24 SWJ_CFG[2:0]: Serial wire JTAG configuration (Конфигурация JTAG)

These bits are write-only (when read, the value is undefined). They are used to configure the SWJ and trace alternate function I/Os. The SWJ (Serial Wire JTAG) supports JTAG or SWD access to the Cortex debug port. The default state after reset is SWJ ON without trace. This allows JTAG or SW mode to be enabled by sending a specific sequence on the JTMS / JTCK pin.

Это биты только для записи (при чтении их значение не определено). Они используются для конфигурации альтернативных функций **SWJ** и трассировки. После сброса порт отладки находится в режиме: **SWJ** включен без трассировки. Это позволяет разрешать режим **JTAG** или **SW**, посылая определенную последовательность на вывод **JTMS/JTCK**.

`000:` Full SWJ (JTAG-DP + SW-DP): Reset State

`001:` Full SWJ (JTAG-DP + SW-DP) but without JNTRST

`010:` JTAG-DP Disabled and SW-DP Enabled

`100:` JTAG-DP Disabled and SW-DP Disabled

`Other combinations:` no effect

Bit 23 MII_RMII_SEL: MII or RMII selection (Выбор MII или RMII)

This bit is set and cleared by software. It configures the Ethernet MAC internally for use with an external MII or RMII PHY.

Этот бит ставится и очищается программно. Он конфигурирует **Ethernet MAC** на использование с внешним **MII PHY** или с **RMII PHY**.

`0:` Configure Ethernet MAC for connection with an MII PHY

Ethernet MAC сконфигурован на подключение к **MII PHY**

`1:` Configure Ethernet MAC for connection with an RMII PHY

Ethernet MAC сконфигурован на подключение к **RMII PHY**

Note: This bit is available only in connectivity line devices and is reserved otherwise.

Этот бит доступен только для семейства **Connectivity Line**, в противном случает он зарезервирован.

Bit 22 CAN2_REMAP: CAN2 I/O remapping (Переназначение для CAN2)

This bit is set and cleared by software. It controls the CAN2_TX and CAN2_RX pins.

Этот бит ставится и очищается программно. Он управляет назначением выводов **CAN2_RX** и **CAN2_TX**

`0:` No remap (Нет переназначения) (CAN2_RX/PB12, CAN2_TX/PB13)

`1:` Remap (Переназначение) (CAN2_RX/PB5, CAN2_TX/PB6)

Note: This bit is available only in connectivity line devices and is reserved otherwise.

Этот бит доступен только для семейства **Connectivity Line**, в противном случает он зарезервирован.

Bit 21 ETH_REMAP: Ethernet MAC I/O remapping (Переназначение для Ethernet MAC)

This bit is set and cleared by software. It controls the Ethernet MAC connections with the PHY.
Этот бит ставится и очищается программно. Он управляет подключением **Ethernet MAC** к **PHY**.

`0:` No remap (Нет переназначения)
(RX_DV-CRS_DV/PA7, RXD0/PC4, RXD1/PC5, RXD2/PB0, RXD3/PB1)

`0:` Remap (Переназначение)
(RX_DV-CRS_DV/PD8, RXD0/PD9, RXD1/PD10, RXD2/PD11, RXD3/PD12)

Note: This bit is available only in connectivity line devices and is reserved otherwise.
Этот бит доступен только для семейства **Connectivity Line**, в противном случает он зарезервирован.

Bits 20:17 Reserved

Bits 16 TIM5CH4_IREMAP: TIM5 channel4 internal remap
Переназначение вывода канала 4 таймера 5

Set and cleared by software. This bit controls the TIM5_CH4 internal mapping. When reset the timer TIM5_CH4 is connected to PA3. When set the LSI internal clock is connected to TIM5_CH4 input for calibration purpose.
Ставится и очищается программно. Этот бит управляет подключением входа **TIM5_CH4**. Когда этот бит сброшен, то этот вход подключен к выводу **PA3**. Когда этот бит установлен, то ко входу **TIM5_CH4** подключен внутренний тактовый сигнал от **LSI** в целях его калибровки.

Bit 15 PD01_REMAP: Port D0/Port D1 mapping on OSC_IN/OSC_OUT
Переназначение выводов PD0/PD1 на OSC_IN/OSC_OUT

This bit is set and cleared by software. It controls the mapping of PD0 and PD1 GPIO functionality. When the HSE oscillator is not used (application running on internal 8 MHz RC) PD0 and PD1 can be mapped on OSC_IN and OSC_OUT. This is available only on 36-, 48- and 64-pin packages (PD0 and PD1 are available on 100-pin and 144-pin packages, no need for remapping).
Этот бит ставится и очищается программно. Он управляет функциональным назначением выводов **PD0** и **PD1 GPIO**. Когда **HSE** генератор не используется (приложение работает от внутреннего 8 МГц **RC** генератора), то на место выводов **OSC_IN** и **OSC_OUT** можно назначить **PD0** и **PD1**. Это возможно только для корпусов на 36, 48 и 64 выводов (выводы **PD0** и **PD1** доступны в корпусах на 100 и 144 выводов, и не нуждаются в переназначении).

`0:` No remapping of PD0 and PD1 (Нет переназначения для **PD0** и **PD1**)

`1:` PD0 remapped on OSC_IN, PD1 remapped on OSC_OUT,
PD0 переназначен на **OSC_IN**, а **PD1** - на **OSC_OUT**.

Bits 14:13 CAN1_REMAP[1:0]: CAN1 alternate function remapping
Переназначение альтернативной функции CAN1

These bits are set and cleared by software. They control the mapping of alternate functions CAN1_RX and CAN1_TX.
Эти биты ставятся и очищаются программно. Они управляют назначением альтернативных функций выводов **CAN1_RX** и **CAN1_TX**.

`00:` CAN1_RX mapped to PA11, CAN1_TX mapped to PA12
CAN1_RX назначен на **PA11**, **CAN1_TX** - на **PA12**

`01:` Not used (Не использовать)

`10:` CAN1_RX mapped to PB8, CAN1_TX mapped to PB9 (not available on 36-pin package)
CAN1_RX назначен на **PB8**, **CAN1_TX** - на **PB9** (недоступно для корпусов на 36 выводов)

`11:` CAN1_RX mapped to PD0, CAN1_TX mapped to PD1
CAN1_RX назначен на **PD0**, **CAN1_TX** - на **PD1**

Bit 12 TIM4_REMAP: TIM4 remapping (Переназначение для TIM4)

This bit is set and cleared by software. It controls the mapping of TIM4 channels 1 to 4 onto the GPIO ports.

Этот бит ставится и очищается программно. Он управляет назначением выводов для каналов с 1-го по 4-й таймера **TIM4** на порты **GPIO**.

0: No remap (Нет переназначения)
(TIM4_CH1/PB6, TIM4_CH2/PB7, TIM4_CH3/PB8, TIM4_CH4/PB9)

1: Full remap (Полное переназначение)
(TIM4_CH1/PD12, TIM4_CH2/PD13, TIM4_CH3/PD14, TIM4_CH4/PD15)

Note: TIM4_ETR on PE0 is not re-mapped. (Вывод **TIM4_ETR/PE0** не переназначается.)

Bits 11:10 TIM3_REMAP[1:0]: TIM3 remapping (Переназначение для TIM3)

These bits are set and cleared by software. They control the mapping of TIM3 channels 1 to 4 on the GPIO ports.

Эти биты ставятся и очищаются программно. Они управляют назначением выводов для каналов с 1-го по 4-й таймера **TIM3** на порты **GPIO**.

00: No remap (Нет переназначения) (CH1/PA6, CH2/PA7, CH3/PB0, CH4/PB1)

01: Not used (Не использовать)

10: Partial remap (Частичное переназначение) (CH1/PB4, CH2/PB5, CH3/PB0, CH4/PB1)

11: Full remap (Полное переназначение) (CH1/PC6, CH2/PC7, CH3/PC8, CH4/PC9)

Note: TIM3_ETR on PE0 is not re-mapped. (Вывод **TIM3_ETR/PE0** не переназначается.)

Bits 9:8 TIM2_REMAP[1:0]: TIM2 remapping (Переназначение для TIM2)

These bits are set and cleared by software. They control the mapping of TIM2 channels 1 to 4 and external trigger (ETR) on the GPIO ports.

Эти биты ставятся и очищаются программно. Они управляют назначением выводов для таймера **TIM2**: каналы с 1-го по 4-й и вход внешнего запуска (**ETR**), на порты **GPIO**.

00: No remap (Нет переназначения) (CH1/ETR/PA0, CH2/PA1, CH3/PA2, CH4/PA3)

01: Partial remap (Частичное переназначение)
(CH1/ETR/PA15, CH2/PB3, CH3/PA2, CH4/PA3)

10: Partial remap (Частичное переназначение)
(CH1/ETR/PA0, CH2/PA1, CH3/PB10, CH4/PB11)

11: Full remap (Полное переназначение)
(CH1/ETR/PA15, CH2/PB3, CH3/PB10, CH4/PB11)

Bits 7:6 TIM1_REMAP[1:0]: TIM1 remapping (Переназначение для TIM1)

These bits are set and cleared by software. They control the mapping of TIM2 channels 1 to 4, 1N to 3N, external trigger (ETR) and Break input (BKIN) on the GPIO ports.

Эти биты ставятся и очищаются программно. Они управляют назначением выводов для таймера **TIM1**: каналы 1-4, **1N-3N**, вход внешнего запуска (**ETR**) и вход прерывания счета (**BKIN**), на порты **GPIO**.

00: No remap (Нет переназначения) (ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PB12, CH1N/PB13, CH2N/PB14, CH3N/PB15)

01: Partial remap (Частичное переназначение) (ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PA6, CH1N/PA7, CH2N/PB0, CH3N/PB1)

10: not used (Не использовать)

11: Full remap (Полное переназначение) (ETR/PE7, CH1/PE9, CH2/PE11, CH3/PE13, CH4/PE14, BKIN/PE15, CH1N/PE8, CH2N/PE10, CH3N/PE12)

Bits 5:4 USART3_REMAP[1:0]: USART3 remapping (Переназначение для USART3)

These bits are set and cleared by software. They control the mapping of USART3 CTS, RTS, CK, TX and RX alternate functions on the GPIO ports.

Эти биты ставятся и очищаются программно. Они управляют назначением альтернативных функций выводов **CTS**, **RTS**, **CK**, **TX** и **RX** для **USART3** на порты **GPIO**.

`00:` No remap (Нет переназначения)
(TX/PB10, RX/PB11, CK/PB12, CTS/PB13, RTS/PB14)

`01:` Partial remap (Частичное переназначение)
(TX/PC10, RX/PC11, CK/PC12, CTS/PB13, RTS/PB14)

`10:` not used (Не использовать)

`11:` Full remap (Полное переназначение)
(TX/PD8, RX/PD9, CK/PD10, CTS/PD11, RTS/PD12)

Bit 3 USART2_REMAP: USART2 remapping (Переназначение для USART2)

This bit is set and cleared by software. It controls the mapping of USART2 CTS, RTS, CK, TX and RX alternate functions on the GPIO ports.

Этот бит ставится и очищается программно. Он управляет назначением альтернативных функций выводов **CTS**, **RTS**, **CK**, **TX** и **RX** для **USART2** на порты **GPIO**.

`0:` No remap (Нет переназначения) (CTS/PA0, RTS/PA1, TX/PA2, RX/PA3, CK/PA4)

`1:` Remap (Переназначение) (CTS/PD3, RTS/PD4, TX/PD5, RX/PD6, CK/PD7)

Bit 2 USART1_REMAP: USART1 remapping (Переназначение для USART1)

This bit is set and cleared by software. It controls the mapping of USART1 TX and RX alternate functions on the GPIO ports.

Этот бит ставится и очищается программно. Он управляет назначением альтернативных функций выводов **TX** и **RX** для **USART1** на порты **GPIO**.

`0:` No remap (Нет переназначения) (TX/PA9, RX/PA10)

`1:` Remap (Переназначение) (TX/PB6, RX/PB7)

Bit 1 I2C1_REMAP: I2C1 remapping (Переназначение для I2C1)

This bit is set and cleared by software. It controls the mapping of I2C1 SCL and SDA alternate functions on the GPIO ports.

Этот бит ставится и очищается программно. Он управляет назначением альтернативных функций выводов **SCL** и **SDA** для **I2C1** на порты **GPIO**.

`0:` No remap (Нет переназначения) (SCL/PB6, SDA/PB7)

`1:` Remap (Переназначение) (SCL/PB8, SDA/PB9)

Bit 0 SPI1_REMAP: SPI1 remapping (Переназначение для SPI1)

This bit is set and cleared by software. It controls the mapping of SPI1 NSS, SCK, MISO, MOSI alternate functions on the GPIO ports.

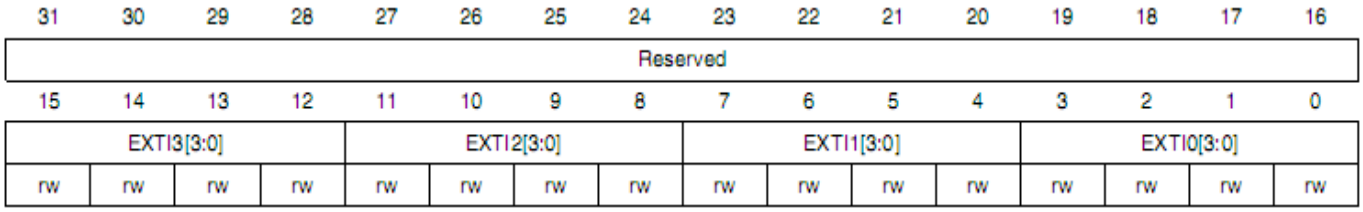
Этот бит ставится и очищается программно. Он управляет назначением альтернативных функций выводов **NSS**, **SCK**, **MISO**, **MOSI** для **SPI1** на порты **GPIO**.

`0:` No remap (Нет переназначения) (NSS/PA4, SCK/PA5, MISO/PA6, MOSI/PA7)

`1:` Remap (Переназначение) (NSS/PA15, SCK/PB3, MISO/PB4, MOSI/PB5)

8.4.3 External interrupt configuration register 1 (AFIO_EXTICR1)
Регистр конфигурации прерываний 1

Address offset: 0x08
Reset value: 0x0000



Bits 31:16 Reserved

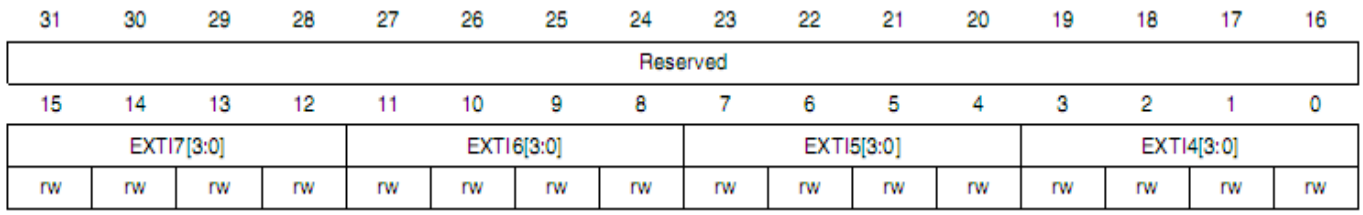
Bits 15:0 EXTIx[3:0]: EXTI x configuration (x= 0 to 3)
Конфигурация внешнего прерывания EXTIx

These bits are written by software to select the source input for EXTIx external interrupt. Refer to Section 9.2.5: External interrupt/event line mapping on page 176
Эти биты пишутся программно, чтобы выбрать порт для внешнего прерывания **EXTIx**.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin

8.4.4 External interrupt configuration register 2 (AFIO_EXTICR2)
Регистр конфигурации прерываний 2

Address offset: 0x0C
Reset value: 0x0000



Bits 31:16 Reserved

Bits 15:0 EXTIx[3:0]: EXTI x configuration (x= 4 to 7)

These bits are written by software to select the source input for EXTIx external interrupt.
Эти биты пишутся программно, чтобы выбрать порт для внешнего прерывания **EXTIx**.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin

8.4.5 External interrupt configuration register 3 (AFIO_EXTICR3)
Регистр конфигурации прерываний 3

Address offset: 0x10
Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved

Bits 15:0 EXTIx[3:0]: EXTI x configuration (x= 8 to 11)

These bits are written by software to select the source input for EXTIx external interrupt.
Эти биты пишутся программно, чтобы выбрать порт для внешнего прерывания **EXTIx**.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin

8.4.6 External interrupt configuration register 4 (AFIO_EXTICR4)
Регистр конфигурации прерываний 4

Address offset: 0x14
Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved

Bits 15:0 EXTIx[3:0]: EXTI x configuration (x= 12 to 15)

These bits are written by software to select the source input for EXTIx external interrupt.
Эти биты пишутся программно, чтобы выбрать порт для внешнего прерывания **EXTIx**.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin

8.5 GPIO and AFIO register maps (Карта регистров GPIO и AFIO)

Refer to Table 1 on page 41 for the register boundary addresses. The following tables give the GPIO and AFIO register map and the reset values.
См. Табл. 1, где даны относительные адреса регистров. Последующие таблицы дают карту регистров **GPIO** и **AFIO** и их значения после сброса.

Table 50. GPIO register map and reset values
Карта регистров **GPIO** и их значение после сброса

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOx_CRL	CNF7 [1:0]		MODE7 [1:0]		CNF6 [1:0]		MODE6 [1:0]		CNF5 [1:0]		MODE5 [1:0]		CNF4 [1:0]		MODE4 [1:0]		CNF3 [1:0]		MODE3 [1:0]		CNF2 [1:0]		MODE2 [1:0]		CNF1 [1:0]		MODE1 [1:0]		CNF0 [1:0]		MODE0 [1:0]	
	Reset value	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0x04	GPIOx_CRH	CNF 15 [1:0]		MODE1 5 [1:0]		CNF 14 [1:0]		MODE1 4 [1:0]		CNF 13 [1:0]		MODE1 3 [1:0]		CNF 12 [1:0]		MODE1 2 [1:0]		CNF 11 [1:0]		MODE1 1 [1:0]		CNF 10 [1:0]		MODE1 0 [1:0]		CNF 9 [1:0]		MODE9 [1:0]		CNF 8 [1:0]		MODE8 [1:0]	
	Reset value	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0x08	GPIOx_IDR	Reserved																IDR[15:0]I																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOx_ODR	Reserved																ODR[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPIOx_BSRR	BR[15:0]																BSR[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	GPIOx_BRR	Reserved																BR[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	GPIOx_LCKR	Reserved															LCKK	LCK[15:0]																
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51. AFIO register map and reset values

Карта регистров **AFIO** и их значение после сброса

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	AFIO_EVCR	Reserved																									EVOE	PORT[2:0]			PIN[3:0]		
	Reset value																										0	0	0	0	0	0	0
0x04	AFIO_MAPR low-, medium- and high-density devices	Reserved					SWJ_CFG[2]	SWJ_CFG[1]	SWJ_CFG[0]	Reserved			ADC2_ETRGREG_REMAP	ADC2_ETRGINJ_REMAP	ADC1_ETRGREG_REMAP	ADC1_ETRGINJ_REMAP	TIM5CH4_IREMAP	PD01_REMAP	CAN1_REMAP[1]	CAN1_REMAP[0]	TIM4_REMPAP	TIM3_REMPAP[1]	TIM3_REMPAP[0]	TIM2_REMPAP[1]	TIM2_REMPAP[0]	TIM1_REMPAP[1]	TIM1_REMPAP[0]	USART3_REMAP[1]	USART3_REMAP[0]	USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP
	Reset value						0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	AFIO_MAPR connectivity line devices	Reserved	PTP_PPS_REMAP	TIM2ITR1_IREMAP	SPI3_REMAP	Reserved	SWJ_CFG[2]	SWJ_CFG[1]	SWJ_CFG[0]	MII_RMII_SEL	CAN2_REMAP	ETH_REMAP	Reserved				TIM5CH4_IREMAP	PD01_REMAP	CAN1_REMAP[1]	CAN1_REMAP[0]	TIM4_REMPAP	TIM3_REMPAP[1]	TIM3_REMPAP[0]	TIM2_REMPAP[1]	TIM2_REMPAP[0]	TIM1_REMPAP[1]	TIM1_REMPAP[0]	USART3_REMAP[1]	USART3_REMAP[0]	USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP
	Reset value		0	0	0		0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	AFIO_EXTICR1	Reserved																EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	AFIO_EXTICR2	Reserved																EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	AFIO_EXTICR3	Reserved																EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	AFIO_EXTICR4	Reserved																EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Table 1 on page 41* for the register boundary addresses.

См. Табл. 1, где даны относительные адреса регистров.

9 Interrupts and events (Прерывания и события)

(*Здесь находится ритуальное напоминание о наличии 4-х семейств для **STM32F10xxx**, и что этот документ обо всех семействах, если нет прямого указания. См. раздел* 1.2 Глоссарий*).*)

9.1 Nested vectored interrupt controller (NVIC) (Контроллер вложенных векторов прерываний)
9.2 External interrupt/event controller (EXTI) (Контроллер внешних прерываний/событий)
9.3 EXTI registers (Регистры **EXTI**)

9.1 Nested vectored interrupt controller (NVIC)
Контроллер вложенных векторов прерываний

9.1.1 SysTick calibration value register (Регистр калибровки значения для системного таймера)
9.1.2 Interrupt and exception vectors (Вектора исключений и прерываний)

Features (Особенности)

- 68 maskable interrupt channels (not including the 16 interrupt lines of Cortex™-M3)
68 маскируемых каналов прерываний (не считая 16 линий прерываний **Cortex-M3**)
- 16 programmable priority levels (4 bits of interrupt priority are used)
16 программируемых уровней приоритетов
(используется 4 бита приоритетов прерываний (*кроме семейства **CL***))
- Low-latency exception and interrupt handling
малая задержка обработки исключений и прерываний
- Power management control (Управление потреблением)
- Implementation of System Control Registers
Реализация регистров управления системой

The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts.
Модуль **NVIC** и интерфейс с ядром процессора сильно взаимосвязаны, вследствии чего имеем малую задержку обработки прерываний и эффективную обработку прерываний, пришедших чуть позже.

All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming see Chap 5 Exceptions & Chap 8 Nested Vectored Interrupt Controller of the ARM *Cortex™-M3 Technical Reference Manual*.
Все прерывания, включая исключения ядра, управляются модулем **NVIC**. Дополнительную информацию по исключениям и программированию **NVIC** см. в главе 5 "***Exceptions***" и главе 8 "***Nested Vectored Interrupt Controller***" документа "***ARM Cortex-M3 Technical Reference Manual***".

9.1.1 SysTick calibration value register
Регистр калибровки значения для системного таймера

The SysTick calibration value is fixed to 9000, which gives a reference time base of 1 ms with the SysTick clock set to 9 MHz (max HCLK/8).
Значение калибровки системного таймера **SysTick** предустановлено и равно 9000, что дает интервал осчета в 1 мс при значении тактового сигнала **SysTick** - 9 МГц (72/8).
(*См. раздел 4.4.4 документа "**STM32F10xxx Cortex-M3 programming manual**".*)

9.1.2 Interrupt and exception vectors (Вектора исключений и прерываний)

Table 52 and Table 53 are the vector tables for connectivity line and other STM32F10xxx devices, respectively. Таблицы 52 и 53 - это таблицы векторов для семейства **Connectivity Line** и других семейств **STM32F10xxx**, соответственно.

Table 52. Vector table for connectivity line devices
Таблица векторов для семейства **Connectivity Line**

Pos	Pri	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000_0000
	-3	fixed	Reset	Reset	0x0000_0004
	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
	-1	fixed	HardFault	All class of fault	0x0000_000C
	0	settable	MemManage	Memory management	0x0000_0010
	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000_0014
	2	settable	UsageFault	Undefined instruction or illegal state	0x0000_0018
	-	-	-	Reserved	0x0000_001C-0x0000_002B
	3	settable	SVCall	System service call via SWI instruction	0x0000_002C
	4	settable	Debug Monitor	Debug Monitor	0x0000_0030
	-	-	-	Reserved	0x0000_0034
	5	settable	PendSV	Pendable request for system service	0x0000_0038
	6	settable	SysTick	System tick timer	0x0000_003C

0	7	settable	WWDG	Window Watchdog interrupt	0x0000_0040
1	8	settable	PVD	PVD through EXTI Line detection interrupt	0x0000_0044
2	9	settable	TAMPER	Tamper interrupt	0x0000_0048
3	10	settable	RTC	RTC global interrupt	0x0000_004C
4	11	settable	FLASH	Flash global interrupt	0x0000_0050
5	12	settable	RCC	RCC global interrupt	0x0000_0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000_0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068
11	18	settable	DMA1_Channel1	DMA1 Channel1 global interrupt	0x0000_006C
12	19	settable	DMA1_Channel2	DMA1 Channel2 global interrupt	0x0000_0070
13	20	settable	DMA1_Channel3	DMA1 Channel3 global interrupt	0x0000_0074
14	21	settable	DMA1_Channel4	DMA1 Channel4 global interrupt	0x0000_0078
15	22	settable	DMA1_Channel5	DMA1 Channel5 global interrupt	0x0000_007C
16	23	settable	DMA1_Channel6	DMA1 Channel6 global interrupt	0x0000_0080
17	24	settable	DMA1_Channel7	DMA1 Channel7 global interrupt	0x0000_0084
18	25	settable	ADC1_2	ADC1 and ADC2 global interrupt	0x0000_0088
19	26	settable	CAN1_TX	CAN1 TX interrupts	0x0000_008C
20	27	settable	CAN1_RX0	CAN1 RX0 interrupts	0x0000_0090
21	28	settable	CAN1_RX1	CAN1 RX1 interrupts	0x0000_0094
22	29	settable	CAN1_SCE	CAN1 SCE interrupts	0x0000_0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_00

					9C
24	31	settable	TIM1_BRK	TIM1 Break interrupt	0x0000_00A0
25	32	settable	TIM1_UP	TIM1 Update interrupt	0x0000_00A4
26	33	settable	TIM1_TRG_COM	TIM1 Trigger and Commutation interrupts	0x0000_00A8
27	34	settable	TIM1_CC	TIM1 Capture Compare interrupt	0x0000_00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000_00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000_00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000_00B8
31	38	settable	I2C1_EV	I2C1 event interrupt	0x0000_00BC
32	39	settable	I2C1_ER	I2C1 error interrupt	0x0000_00C0
33	40	settable	I2C2_EV	I2C2 event interrupt	0x0000_00C4
34	41	settable	I2C2_ER	I2C2 error interrupt	0x0000_00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000_00D0
37	44	settable	USART1	USART1 global interrupt	0x0000_00D4
38	45	settable	USART2	USART2 global interrupt	0x0000_00D8
39	46	settable	USART3	USART3 global interrupt	0x0000_00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00E0
41	48	settable	RTCAlarm	RTC alarm through EXTI line interrupt	0x0000_00E4
42	49	settable	OTG_FS_WKUP	USB On-The-Go FS Wakeup through EXTI line interrupt	0x0000_00E8
-	-	-		Reserved	0x0000_00EC-0x0000_0104
50	57	settable	TIM5	TIM5 global interrupt	0x0000_0108
51	58	settable	SPI3	SPI3 global interrupt	0x0000_010C

52	59	settable	UART4	UART4 global interrupt	0x0000_0110
53	60	settable	UART5	UART5 global interrupt	0x0000_0114
54	61	settable	TIM6	TIM6 global interrupt	0x0000_0118
55	62	settable	TIM7	TIM7 global interrupt	0x0000_011C
56	63	settable	DMA2_Channel1	DMA2 Channel1 global interrupt	0x0000_0120
57	64	settable	DMA2_Channel2	DMA2 Channel2 global interrupt	0x0000_0124
58	65	settable	DMA2_Channel3	DMA2 Channel3 global interrupt	0x0000_0128
59	66	settable	DMA2_Channel4	DMA2 Channel4 global interrupt	0x0000_012C
60	67	settable	DMA2_Channel5	DMA2 Channel5 global interrupt	0x0000_0130
61	68	settable	ETH	Ethernet global interrupt	0x0000_0134
62	69	settable	ETH_WKUP	Ethernet Wakeup through EXTI line interrupt	0x0000_0138
63	70	settable	CAN2_TX	CAN2 TX interrupts	0x0000_013C
64	71	settable	CAN2_RX0	CAN2 RX0 interrupts	0x0000_0140
65	72	settable	CAN2_RX1	CAN2 RX1 interrupt	0x0000_0144
66	73	settable	CAN2_SCE	CAN2 SCE interrupt	0x0000_0148
67	74	settable	OTG_FS	USB On The Go FS global interrupt	0x0000_014C

Table 53. Vector table for other STM32F10xxx devices
Таблица векторов для других семейств **STM32F10xxx**

Pos	Pri	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000_0000
	-3	fixed	Reset	Reset	0x0000_0004
	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
	-1	fixed	HardFault	All class of fault	0x0000_000C
	0	settable	MemManage	Memory management	0x0000_0010

	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000_0014
	2	settable	UsageFault	Undefined instruction or illegal state	0x0000_0018
	-	-	-	Reserved	0x0000_001C-0x0000_002B
	3	settable	SVCall	System service call via SWI instruction	0x0000_002C
	4	settable	Debug Monitor	Debug Monitor	0x0000_0030
	-	-	-	Reserved	0x0000_0034
	5	settable	PendSV	Pendable request for system service	0x0000_0038
	6	settable	SysTick	System tick timer	0x0000_003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000_0040
1	8	settable	PVD	PVD through EXTI Line detection interrupt	0x0000_0044
2	9	settable	TAMPER	Tamper interrupt	0x0000_0048
3	10	settable	RTC	RTC global interrupt	0x0000_004C
4	11	settable	FLASH	Flash global interrupt	0x0000_0050
5	12	settable	RCC	RCC global interrupt	0x0000_0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000_0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068
11	18	settable	DMA1_Channel1	DMA1 Channel1 global interrupt	0x0000_006C
12	19	settable	DMA1_Channel2	DMA1 Channel2 global interrupt	0x0000_0070
13	20	settable	DMA1_Channel3	DMA1 Channel3 global interrupt	0x0000_0074
14	21	settable	DMA1_Channe	DMA1 Channel4 global interrupt	0x0000_00

			l4		78
15	22	settable	DMA1_Channel5	DMA1 Channel5 global interrupt	0x0000_007C
16	23	settable	DMA1_Channel6	DMA1 Channel6 global interrupt	0x0000_0080
17	24	settable	DMA1_Channel7	DMA1 Channel7 global interrupt	0x0000_0084
18	25	settable	ADC1_2	ADC1 and ADC2 global interrupt	0x0000_0088
19	26	settable	USB_HP_CAN_TX	USB High Priority or CAN TX interrupts	0x0000_008C
20	27	settable	USB_LP_CAN_RX0	USB Low Priority or CAN RX0 interrupts	0x0000_0090
21	28	settable	CAN_RX1	CAN RX1 interrupts	0x0000_0094
22	29	settable	CAN_SCE	CAN SCE interrupts	0x0000_0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_009C
24	31	settable	TIM1_BRK	TIM1 Break interrupt	0x0000_00A0
25	32	settable	TIM1_UP	TIM1 Update interrupt	0x0000_00A4
26	33	settable	TIM1_TRG_COM	TIM1 Trigger and Commutation interrupts	0x0000_00A8
27	34	settable	TIM1_CC	TIM1 Capture Compare interrupt	0x0000_00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000_00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000_00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000_00B8
31	38	settable	I2C1_EV	I2C1 event interrupt	0x0000_00BC
32	39	settable	I2C1_ER	I2C1 error interrupt	0x0000_00C0
33	40	settable	I2C2_EV	I2C2 event interrupt	0x0000_00C4
34	41	settable	I2C2_ER	I2C2 error interrupt	0x0000_00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000_00D0
37	44	settable	USART1	USART1 global interrupt	0x0000_00D4

38	45	settable	USART2	USART2 global interrupt	0x0000_00D8
39	46	settable	USART3	USART3 global interrupt	0x0000_00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00E0
41	48	settable	RTCAlarm	RTC alarm through EXTI line interrupt	0x0000_00E4
42	49	settable	USBWakeup	USB wakeup from suspend through EXTI line interrupt	0x0000_00E8
43	50	settable	TIM8_BRK	TIM8 Break interrupt	0x0000_00EC
44	51	settable	TIM8_UP	TIM8 Update interrupt	0x0000_00F0
45	52	settable	TIM8_TRG_COM	TIM8 Trigger and Commutation interrupts	0x0000_00F4
46	53	settable	TIM8_CC	TIM8 Capture Compare interrupt	0x0000_00F8
47	54	settable	ADC3	ADC3 global interrupt	0x0000_00FC
48	55	settable	FSMC	FSMC global interrupt	0x0000_0100
49	56	settable	SDIO	SDIO global interrupt	0x0000_0104
50	57	settable	TIM5	TIM5 global interrupt	0x0000_0108
51	58	settable	SPI3	SPI3 global interrupt	0x0000_010C
52	59	settable	UART4	UART4 global interrupt	0x0000_0110
53	60	settable	UART5	UART5 global interrupt	0x0000_0114
54	61	settable	TIM6	TIM6 global interrupt	0x0000_0118
55	62	settable	TIM7	TIM7 global interrupt	0x0000_011C
56	63	settable	DMA2_Channel1	DMA2 Channel1 global interrupt	0x0000_0120
57	64	settable	DMA2_Channel2	DMA2 Channel2 global interrupt	0x0000_0124
58	65	settable	DMA2_Channel3	DMA2 Channel3 global interrupt	0x0000_0128
59	66	settable	DMA2_Channel4_5	DMA2 Channel4 and DMA2 Channel5 global interrupts	0x0000_012C

9.2 External interrupt/event controller (EXTI)
Контроллер внешних прерываний/событий

9.2.1 Main features (Основные особенности)
9.2.2 Block diagram (Структурная схема)
9.2.3 Wakeup event management (Управление событиями пробуждения)
9.2.4 Functional description (Функциональное описание)
9.2.5 External interrupt/event line mapping (Назначения цепей для внешних прерываний/событий)

The external interrupt/event controller consists of up to 20 edge detectors in connectivity line devices, or 19 edge detectors in other devices for generating event/interrupt requests. Each input line can be independently configured to select the type (pulse or pending) and the corresponding trigger event (rising or falling or both). Each line can also masked independently. A pending register maintains the status line of the interrupt requests.
Контроллер внешних прерываний/событий имеет 20 детекторов перепада для генерации запросов прерываний/событий. Каждая входная линия может быть независимо сконфигурирована на выбор типа сигнала (импульс/уровень) и типа перепада (фронт/срез/оба) для запуска события. Каждая линия может быть независимо маскирована. Регистр запросов прерываний содержит флаги состояния запросов на прерывание.

9.2.1 Main features (Основные особенности)

The EXTI controller main features are the following: (Основные особенности **EXTI** контроллера:)

- Independent trigger and mask on each interrupt/event line
Независимый флаг запроса и маска для каждой линии прерывания/события

- Dedicated status bit for each interrupt line
Выделенный бит статуса для каждой линии прерывания

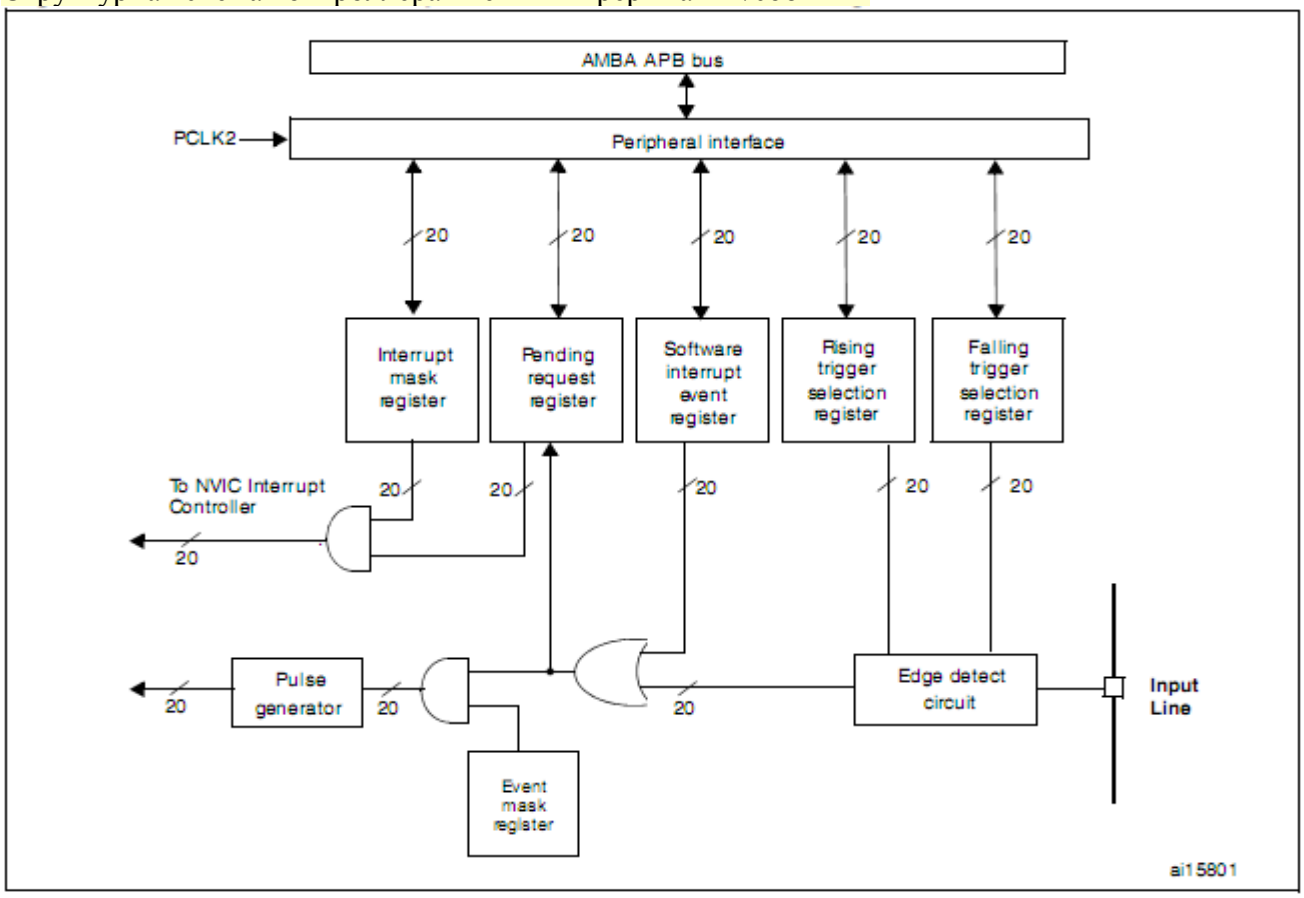
- Generation of up to 20 software event/interrupt requests
Генерирование до 20 программных запросов прерываний/событий

- Detection of external signal with pulse width lower than APB2 clock period. Refer to the electrical characteristics section of the datasheet for details on this parameter.
Детектирование внешних сигналов, ширина импульсов которых меньше, чем период тактового сигнала шины **APB2**.

9.2.2 Block diagram (Структурная схема)

The block diagram is shown in *Figure 20*. (Структурная схема приведена на рис. 20.)

Figure 20. External interrupt/event controller block diagram
Структурная схема контроллера внешних прерываний/событий



9.2.3 Wakeup event management (Управление событиями пробуждения)

The STM32F10xxx is able to handle external or internal events in order to wake up the core (WFE). The wakeup event can be generated either by:
STM32F10xxx способен использовать внешние или внутренние события для пробуждения ядра (**WFE**). Событие пробуждения может исходить:

- enabling an interrupt in the peripheral control register but not in the NVIC, and enabling the SEVONPEND bit in the Cortex-M3 System Control register. When the MCU resumes from WFE, the peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
при разрешении прерывания в регистре управления периферией, но не в модуле **NVIC**, при разрешенном бите **SEVONPEND** в регистре управления системой **SCB_SCR**. Когда ЦПУ проснется от **WFE**, то бит запроса прерывания от периферии (***ISRPENDING ?***) и бит запроса прерывания от **IRQ** канала модуля **NVIC** (в **NVIC** регистре очистки запроса прерываний (***NVIC_ICPRx ?***)) надо очистить.

- or configuring an external or internal EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bit corresponding to the event line is not set.
при конфигурировании внешней или внутренней линии **EXTI** на режим "событие". Когда

ЦПУ проснется от **WFE**, то нет необходимости очищать бит запроса прерывания от периферии или бит запроса прерывания от **IRQ** канала модуля **NVIC**, так как бит запроса соответствующей линии события не устанавливается.

In connectivity line devices, Ethernet wakeup events also have the WFE wakeup capability.
В устройствах семейства **Connectivity Line**, события от **Ethernet** также имеют способность **WFE** пробуждения.

To use an external line as a wakeup event, refer to Section 9.2.4: Functional description.
Чтобы использовать внешние линии как событие пробуждения, см. раздел 9.2.4: "Функциональное описание".

9.2.4 Functional description (Функциональное описание)

To generate the interrupt, the interrupt line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the interrupt request by writing a ‘1’ to the corresponding bit in the interrupt mask register. When the selected edge occurs on the external interrupt line, an interrupt request is generated. The pending bit corresponding to the interrupt line is also set. This request is reset by writing a ‘1’ in the pending register.
Чтобы генерировать прерывание, линия прерывания должна быть сконфигурирована и разрешена. Это выполняется программированием двух регистров запуска, где выбирается нужный перепад сигнала прерывания, и разрешением прерывания записью '1' в соответствующий бит регистра маскирования прерывания. Когда случился выбранный перепад на линии внешнего прерывания, генерируется запрос на прерывание. Также устанавливается флаг запроса от соответствующей линии прерывания. Этот флаг запроса можно сбросить записью '1' в регистр сброса прерывания.

To generate the event, the event line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the event request by writing a ‘1’ to the corresponding bit in the event mask register. When the selected edge occurs on the event line, an event pulse is generated. The pending bit corresponding to the event line is not set.
Чтобы сгенерировать событие, линия события должна быть сконфигурирована и разрешена. Это выполняется программированием двух регистров запуска, где выбирается нужный перепад сигнала события, и разрешением события записью '1' в соответствующий бит регистра маскирования события. Когда случился выбранный перепад на линии события, генерируется импульс события. Флаг запроса для соответствующей линии события не устанавливается.

An interrupt/event request can also be generated by software by writing a ‘1’ in the software interrupt/event register. Запрос прерывания/события можно также сгенерировать программно, записью '1' в регистр "программного прерывания/события".

Hardware interrupt selection (Выбор аппаратного источника прерывания)

To configure the 20 lines as interrupt sources, use the following procedure:
Чтобы сконфигурировать 20 линий, как источников прерываний, используйте следующую процедуру:

- Configure the mask bits of the 20 Interrupt lines (EXTI_IMR)
Конфигурируйте биты маскирования 20-ти линий прерывания (**EXTI_IMR**).

- Configure the Trigger Selection bits of the Interrupt lines (EXTI_RTSR and EXTI_FTSR)
Конфигурируйте биты выбора типа запуска этих линий (**EXTI_RTSR** и **EXTI_FTSR**).

- Configure the enable and mask bits that control the NVIC IRQ channel mapped to the External Interrupt Controller (EXTI) so that an interrupt coming from one of the 20 lines can be correctly acknowledged.

Конфигурируйте биты разрешения и биты маскирования, которые управляют назначением **NVIC IRQ** каналов на линии контроллера внешних прерываний (**EXTI**), так чтобы прерывание, пришедшее от одной из 20-ти линий, могло быть корректно опознано.

Hardware event selection (Выбор аппаратного источника события)

To configure the 20 lines as event sources, use the following procedure:
Чтобы сконфигурировать 20 линий, как источников событий, используйте следующую процедуру:

- Configure the mask bits of the 20 Event lines (EXTI_EMR)
Конфигурируйте биты маскирования 20-ти линий событий (**EXTI_EMR**).
- Configure the Trigger Selection bits of the Event lines (EXTI_RTSR and EXTI_FTSR)
Конфигурируйте биты выбора типа запуска этих линий (**EXTI_RTSR** и **EXTI_FTSR**).

Software interrupt/event selection
Выбор программного источника прерывания/события

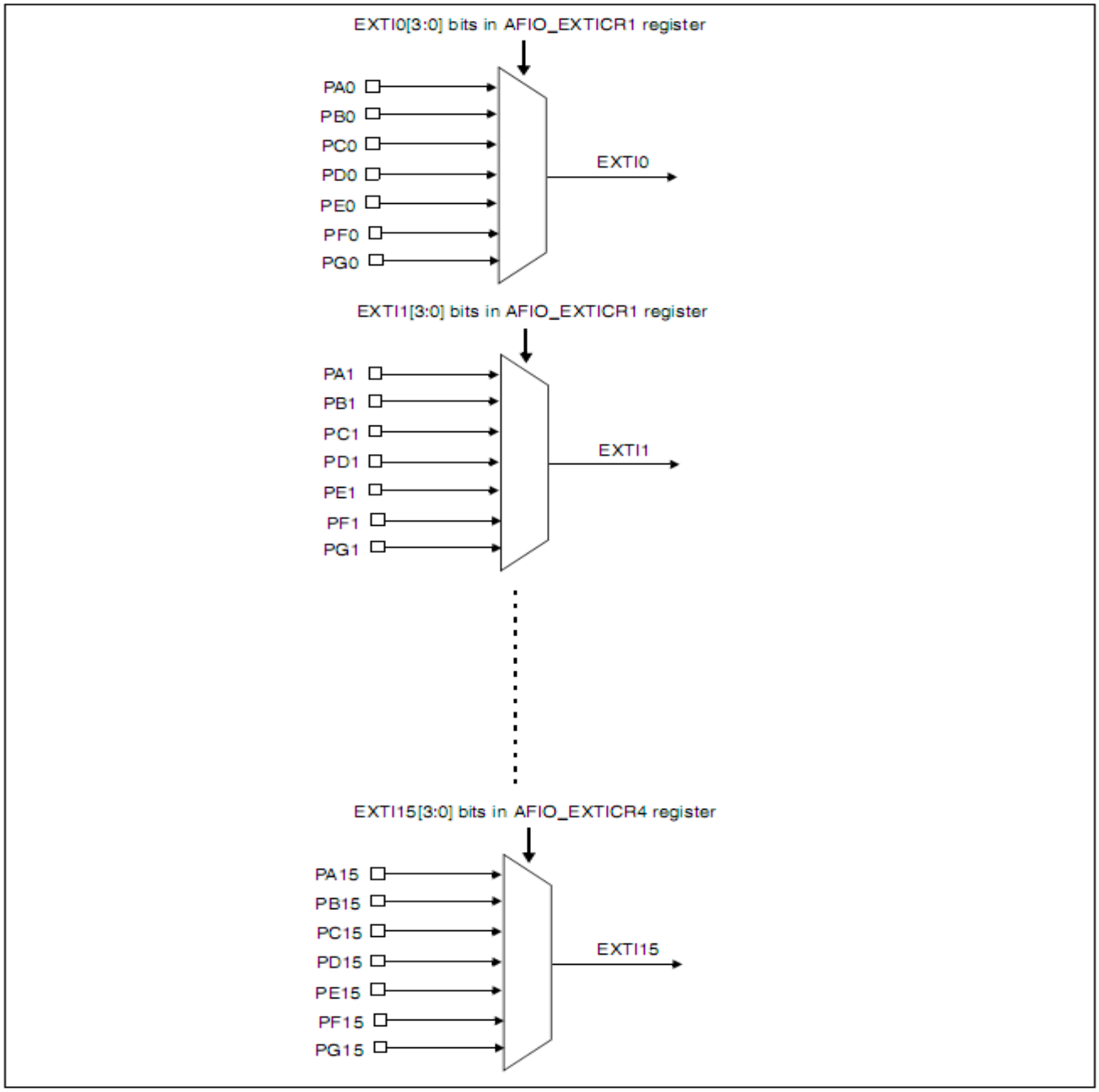
The 20 lines can be configured as software interrupt/event lines. The following is the procedure to generate a software interrupt.
Эти 20 линий можно сконфигурировать, как линии программного прерывания/события.
Используйте следующую процедуру для генерации программного прерывания.

- Configure the mask bits of the 20 Interrupt/Event lines (EXTI_IMR, EXTI_EMR)
Конфигурируйте биты маскирования 20-ти линий прерывания/события (**EXTI_IMR**, **EXTI_EMR**).
- Set the required bit of the software interrupt register (EXTI_SWIER)
Установите требуемый бит запроса в регистре программного прерывания (**EXTI_SWIER**).

9.2.5 External interrupt/event line mapping (Назначения цепей для внешних прерываний/событий)

The 112 GPIOs are connected to the 16 external interrupt/event lines in the following manner:
112 линий **GPIO** подключены к 16 внешним линиям прерываний/событий следующим образом:

Figure 21. External interrupt/event GPIO mapping
Назначения цепей для внешних прерываний/событий



1. To configure the AFIO_EXTICRx for the mapping of external interrupt/event lines onto GPIOs, the AFIO clock should first be enabled. Refer to *Section 6.3.7: APB2 peripheral clock enable register (RCC_APB2ENR)* for low-, medium- and high-density devices and, to *Section 7.3.7: APB2 peripheral clock enable register (RCC_APB2ENR)* for connectivity line devices.
1. Чтобы сконфигурировать **AFIO_EXTICRx** для назначения внешних линий прерываний/событий на выводы **GPIO**, сначала надо разрешить подачу тактового сигнала на **AFIO**. См. раздел 7.3.7 "Регистр разрешения периферии **APB2 (RCC_APB2ENR)**" (для семейства **Connectivity Line**) или раздел 6.3.7 (для остальных семейств).

The four other EXTI lines are connected as follows:
Четыре других линии **EXTI** подключены следующим образом:

EXTI line 16 is connected to the PVD output (Линия 16 **EXTI** подключена к выходу **PVD**)

EXTI line 17 is connected to the RTC Alarm event (подключена к событию **RTC Alarm**)

EXTI line 18 is connected to the USB Wakeup event (подключена к событию **USB Wakeup**)

EXTI line 19 is connected to the Ethernet Wakeup event (available only in connectivity line devices) (подключена к событию **Ethernet Wakeup** (только для семейства **CL**))

9.3 EXTI registers (Регистры EXTI)

9.3.1 Interrupt mask register (EXTI_IMR) (Регистр маскирования прерываний)
9.3.2 Event mask register (EXTI_EMR) (Регистр маскирования событий)
9.3.3 Rising trigger selection register (EXTI_RTSR)
(Регистр выбора линий, запускающихся по фронту)
9.3.4 Falling trigger selection register (EXTI_FTSR)
(Регистр выбора линий, запускающихся по срезу)
9.3.5 Software interrupt event register (EXTI_SWIER)
(Регистр программных прерываний/событий)
9.3.6 Pending register (EXTI_PR) (Регистр флагов запроса)
9.3.7 EXTI register map (Карта регистров **EXTI**)

Refer to Section 1.1 on page 37 for a list of abbreviations used in register descriptions.
См. раздел 1.1, где приведен перечень сокращений, используемых при описании регистров.

9.3.1 Interrupt mask register (EXTI_IMR) (Регистр маскирования прерываний)

Address offset: 0x00
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												MR19	MR18	MR17	MR16
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved
must be kept at reset value (0). (При записи всегда 0)

Bits 19:0 MRx: Interrupt Mask on line x (Маскирование прерывания на линии x)
0: Interrupt request from Line x is masked (Запрос прерывания от линии **x** маскирован)
1: Interrupt request from Line x is not masked (Запрос прерывания от линии **x** разрешен)

Note: Bit 19 is used in connectivity line devices only and is reserved otherwise.
Вит 19 используется только в семействе **CL** и зарезервирован у других семейств.

9.3.2 Event mask register (EXTI_EMR) (Регистр маскирования событий)

Address offset: 0x04
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												MR19	MR18	MR17	MR16
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved

must be kept at reset value (0). (При записи всегда 0)

Bits 19:0 MRx: Event Mask on line x (Маскирование события на линии x)

0: Event request from Line x is masked (Запрос события от линии **x** маскирован)

1: Event request from Line x is not masked (Запрос события от линии **x** разрешен)

Note: Bit 19 is used in connectivity line devices only and is reserved otherwise.

Вит 19 используется только в семействе **CL** и зарезервирован у других семейств.

9.3.3 Rising trigger selection register (EXTI_RTSR)

Регистр выбора линий, запускающихся по фронту

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												TR19	TR18	TR17	TR16
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved

must be kept at reset value (0). (При записи всегда 0)

Bits 19:0 TRx: Rising trigger event configuration bit of line x

Конфигурирует биты запуска линии x по фронту

0: Rising trigger disabled (for Event and Interrupt) for input line

Запуск по фронту на линии отключен (для события и прерывания)

1: Rising trigger enabled (for Event and Interrupt) for input line.

Разрешен запуск по фронту на линии (для события и прерывания)

Note: Bit 19 is used in connectivity line devices only and is reserved otherwise.

Вит 19 используется только в семействе **CL** и зарезервирован у других семейств.

***Note**: The external wakeup lines are edge triggered, no glitches must be generated on these lines. If a rising edge on external interrupt line occurs during writing of EXTI_RTSR register, the pending bit will not be set. Rising and Falling edge triggers can be set for the same interrupt line. In this configuration, both generate a trigger condition.*

***Note**: Внешние линии пробуждения запускаются по перепаду, никаких помех не должно допускаться на этих линиях. Если фронт сигнала произойдет на внешней линии прерывания во время записи регистра **EXTI_RTSR**, то бит запроса прерывания не будет установлен. Оба разрешения, на запуск и от фронта и от среза, можно устанавливать для одной линии. В этой конфигурации оба перепада генерируют условие запуска.*

9.3.4 Falling trigger selection register (EXTI_FTSR)
Регистр выбора линий, запускающихся по срезу

Address offset: 0x0C
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												TR19	TR18	TR17	TR16
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved
must be kept at reset value (0). (При записи всегда 0)

Bits 19:0 TRx: Falling trigger event configuration bit of line x
Конфигурирует биты запуска линии x по срезу
`0:` Falling trigger disabled (for Event and Interrupt) for input line
Запуск по срезу на линии отключен (для события и прерывания)
`1:` Falling trigger enabled (for Event and Interrupt) for input line.
Разрешен запуск по срезу на линии (для события и прерывания)
Note: Bit 19 used in connectivity line devices and is reserved otherwise.
Вит 19 используется только в семействе **CL** и зарезервирован у других семейств.

***Note**: The external wakeup lines are edge triggered, no glitches must be generated on these lines. If a falling edge on external interrupt line occurs during writing of EXTI_FTSR register, the pending bit will not be set. Rising and Falling edge triggers can be set for the same interrupt line. In this configuration, both generate a trigger condition.*
*Внешние линии пробуждения запускаются по перепаду, никаких помех не должно допускаться на этих линиях. Если срез сигнала произойдет на внешней линии прерывания во время записи регистра **EXTI_RTSR**, то бит запроса прерывания не будет установлен. Оба разрешения, на запуск и от фронта и от среза, можно устанавливать для одной линии. В этой конфигурации оба перепада генерируют условие запуска.*

9.3.5 Software interrupt event register (EXTI_SWIER)
Регистр программных прерываний/событий

Address offset: 0x10
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												SWIER 19	SWIER 18	SWIER 17	SWIER 16
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIER 15	SWIER 14	SWIER 13	SWIER 12	SWIER 11	SWIER 10	SWIER 9	SWIER 8	SWIER 7	SWIER 6	SWIER 5	SWIER 4	SWIER 3	SWIER 2	SWIER 1	SWIER 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved
must be kept at reset value (0). (При записи всегда 0)

Bits 19:0 SWIERx: Software interrupt on line x (Программное прерывание на линии x)

Writing a 1 to this bit when it is at 0 sets the corresponding pending bit in EXTI_PR. If the interrupt is enabled on this line on the EXTI_IMR and EXTI_EMR, an interrupt request is generated.
This bit is cleared by clearing the corresponding bit of EXTI_PR (by writing a 1 into the bit).
Запись '1' в этот бит, когда он был равен '0', ставит соответствующий флаг запроса в **EXTI_PR**. Если разрешено прерывание от этой линии в регистрах **EXTI_IMR** и **EXTI_EMR**, то будет сгенерирован запрос на прерывание.
Этот бит очищается записью '1' в соответствующий бит **EXTI_PR**.

Note: Bit 19 used in connectivity line devices and is reserved otherwise.
Вит 19 используется только в семействе **CL** и зарезервирован у других семейств.

9.3.6 Pending register (EXTI_PR) (Регистр флагов запроса)

Address offset: 0x14
Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												PR19	PR18	PR17	PR16
												rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:20 Reserved

must be kept at reset value (0). (При записи всегда 0)

Bits 19:0 PRx: Pending bit (Бит запроса)

0: No trigger request occurred (Нет запроса)
1: elected trigger request occurred (Случилось выбранное условие запуска)

This bit is set when the selected edge event arrives on the external interrupt line.
This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.
Этот бит ставится, когда на внешней линии прерывания случилось выбранное условие запуска. Этот бит очищается записью в него '1' или изменением чувствительности детектора перепада.
Note: Bit 19 is used in connectivity line devices only and is reserved otherwise.
Вит 19 используется только в семействе **CL** и зарезервирован у других семейств.

9.3.7 EXTI register map (Карта регистров EXTI)

The following table gives the EXTI register map and the reset values. Bits 19 in all registers, are used in connectivity line devices and is reserved otherwise.
Таблица ниже дает карту регистров **EXTI** и их значения после сброса. Вит 19 во всех регистрах используется только в семействе **CL** и зарезервирован у других семейств.

Table 54. External interrupt/event controller register map and reset values
Карта регистров контроллера внешних прерываний/событий и их значения после сброса

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	EXTI_IMR	Reserved												MR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	EXTI_EMR	Reserved												MR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	EXTI_RTSR	Reserved												TR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	EXTI_FTSR	Reserved												TR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	EXTI_SWIER	Reserved												SWIER[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	EXTI_PR	Reserved												PR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to Table 1 on page 41 for the register boundary addresses.
См. Табл. 1, где даны относительные адреса регистров.

10 DMA controller (DMA) (Контроллер прямого доступа)

(*Здесь находится ритуальное напоминание о наличии 4-х семейств для **STM32F10xxx**, и что этот документ обо всех семействах, если нет прямого указания. См. раздел 1.2 Глоссарий).*)

10.1 DMA introduction (Введение в **DMA**)
10.2 DMA main features (Основные особенности **DMA**)
10.3 DMA functional description (Функциональное описание **DMA**)
10.4 DMA registers (Регистры **DMA**)

10.1 DMA introduction (Введение в DMA)

Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory as well as memory to memory. Data can be quickly moved by DMA without any CPU actions. This keeps CPU resources free for other operations.
Прямой доступ к памяти (**DMA**) используется, чтобы обеспечить высокоскоростную передачу данных между внешними устройствами (периферией) и памятью, а так же передачу типа память-память. Данные могут быть быстро перемещены с помощью **DMA** без каких любо действий ЦПУ. Это оставляет ресурсы ЦПУ свободными для других операций.

The two DMA controllers have 12 channels in total (7 for DMA1 and 5 for DMA2), each dedicated to managing memory access requests from one or more peripherals. It has an arbiter for handling the priority between DMA requests. Два **DMA** контроллера имеют в общем 12 каналов (7 для **DMA1** и 5 для **DMA2**), каждый из которых специализируется на управлении запросами доступа к памяти от одного или более внешних устройств. **DMA** контроллеры имеют арбитр для обработки приоритетов между запросами на прямой доступ к памяти.

10.2 DMA main features (Основные особенности DMA)

- 12 independently configurable channels (requests): 7 for DMA1 and 5 for DMA2
12 независимо конфигурируемых каналов (запросов): 7 для **DMA1** и 5 для **DMA2**

- Each of the 12 channels is connected to dedicated hardware DMA requests, software trigger is also supported on each channel. This configuration is done by software.
Каждый из этих 12 каналов связан со выделенными аппаратными **DMA** запросами, также на каждом канале поддерживается программный пусковой механизм. Конфигурация задается программно.

- Priorities between requests from channels of one DMA are software programmable (4 levels consisting of very high, high, medium, low) or hardware in case of equality (request 1 has priority over request 2, etc.) Приоритеты между запросами от каналов одного **DMA** контроллера задаются программно (возможны 4 уровня: очень высокий, высокий, средний, низкий) а в случае равенства программных приоритетов, разруливаются аппаратно (запрос с номером 1 имеет приоритет над запросом 2, и т.д.)

- Independent source and destination transfer size (byte, half word, word), emulating packing and unpacking. Source/destination addresses must be aligned on the data size.
Независимые размеры элементов обмена для источника и адресата (байт, полуслово, слово), эмуляция упаковки и распаковки. Адреса источника/приемника обмена должны быть выровнены по размеру (*элемента*) данных.

- Support for circular buffer management (Поддержка кольцевых буферов)

- 3 event flags (DMA Half Transfer, DMA Transfer complete and DMA Transfer Error) logically ORed together in a single interrupt request for each channel Три флага события (Половина обмена **DMA**, Завершение обмена **DMA** и Ошибка обмена **DMA**) логически объединяются вместе (по **OR**) в едный запрос на прерывание для каждого канала

- Memory-to-memory transfer (Обмен типа "память-память")

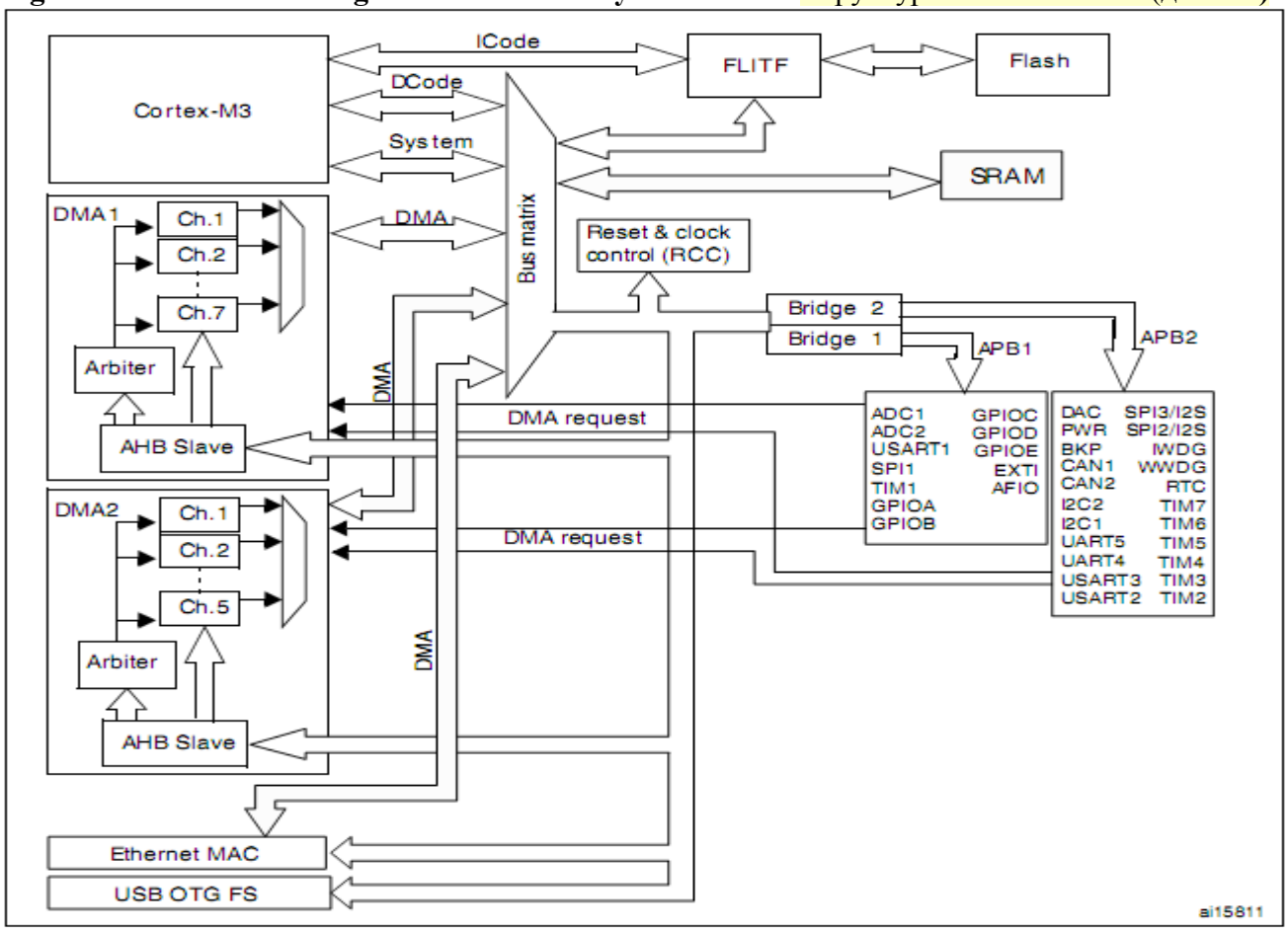
- Peripheral-to-memory and memory-to-peripheral, and peripheral-to-peripheral transfers Обмены типа "периферия-память", "память-периферия" и "периферия-периферия"

- Access to Flash, SRAM, peripheral SRAM, APB1, APB2 and AHB peripherals as source and destination Доступ к **Flash**, **SRAM**, памяти периферии, шинам периферии **APB1**, **APB2** и **AHB**, как к источнику и как к приемнику данных

- Programmable number of data to be transferred: up to 65536
Программируемый размер данных для обмена: до 65536

The block diagram is shown in Figure 22. (Структурная схема приведена на рис. 22.)

Figure 22. DMA block diagram in connectivity line devices Структурная схема **DMA** (для **CL**)



1. The DMA2 controller is available only in high-density and connectifity line devices. Контроллер **DMA2** доступен только для семейств **HD** и **CL**.

2. SPI/I2S3, UART4, TIM5, TIM6, TIM7 and DAC DMA requests are available only in high-density and connectivity line devices.
Запросы на **DMA** от **SPI/I2S3**, **UART4**, **TIM5**, **TIM6**, **TIM7** и **DAC** возможны только для семейств **HD** и **CL**.

3. ADC3, SDIO and TIM8 DMA requests are available only in high-density devices.
Запросы на **DMA** от **ADC3**, **SDIO** и **TIM8** возможны только для семейства **HD**.

10.3 DMA functional description (Функциональное описание DMA)

10.3.1 DMA transactions (**DMA** транзакции)
10.3.2 Arbiter (Арбитр)
10.3.3 DMA channels (**DMA** каналы)
10.3.4 Programmable data width, data alignment and endians
Программирование выравнивания данных и порядка следования байтов
10.3.5 Error management (Обработка ошибок)
10.3.6 Interrupts (Прерывания)
10.3.7 DMA request mapping (Отображение **DMA** запросов)

The DMA controller performs direct memory transfer by sharing the system bus with the Cortex™-M3 core. The DMA request may stop the CPU access to the system bus for some bus cycles, when the CPU and DMA are targeting the same destination (memory or peripheral). The bus matrix implements round-robin scheduling, thus ensuring at least half of the system bus bandwidth (both to memory and peripheral) for the CPU. **DMA** контроллер выполняет прямой обмен с памятью, разделяя системную шину с ядром **Cortex-M3**. **DMA** запрос может приостановить доступ ЦПУ к системной шине на несколько тактов шинны, если ЦПУ и **DMA** работают с одним адресатом (память или внешнее устройство). Матрица шин работает по алгоритму **round-robin** (циклическое планирование), гарантируя, таким образом, по крайней мере половину пропускной способности системной шины для ЦПУ (при обращении как к памяти, так и к периферии).

10.3.1 DMA transactions (DMA транзакции)

After an event, the peripheral sends a request signal to the DMA Controller. The DMA controller serves the request depending on the channel priorities. As soon as the DMA Controller accesses the peripheral, an Acknowledge is sent to the peripheral by the DMA Controller. The peripheral releases its request as soon as it gets the Acknowledge from the DMA Controller. Once the request is deasserted by the peripheral, the DMA Controller release the Acknowledge. If there are more requests, the peripheral can initiate the next transaction. После события периферия посылает сигнал запроса на **DMA** контроллер. **DMA** контроллер обслуживает этот запрос в зависимости от приоритета канала. Как только **DMA** контроллер получает доступ к периферии, он посылает ей сигнал уведомления. Внешнее устройство снимает свой запрос, как только оно получает сигнал уведомления от **DMA** контроллера. Как только снимается сигнал запроса от периферии, **DMA** контроллер, в свою очередь, снимает сигнал уведомления. Если есть дополнительные запросы, периферия может инициализировать следующую транзакцию.

In summary, each DMA transfer consists of three operations:
В итоге, каждый **DMA** обмен состоит из трех операций:

- The loading of data from the peripheral data register or a location in memory addressed through an internal current peripheral/memory address register. The start address used for the first transfer is the base peripheral/memory address programmed in the DMA_CPARx or DMA_CMARx register Загрузка данных из регистров данных периферии или из области памяти, производится с помощью внутреннего регистра текущего адреса периферии/памяти. Стартовый адрес, который используется для первого обмена, является базовым адресом периферии/памяти, и программируется в регистре **DMA_CMARx** или **DMA_CPARx**

- The storage of the data loaded to the peripheral data register or a location in memory addressed through an internal current peripheral/memory address register. The start address used for the first transfer is the base peripheral/memory address programmed in the DMA_CPARx or DMA_CMARx register Сохранение данных, загрузкой в регистры периферии или в область памяти, адресуя их через внутренний регистр текущего адреса периферии/памяти. Стартовый адрес, который используется для первого обмена, является базовым адресом периферии/памяти, и программируется в регистре **DMA_CMARx** или **DMA_CPARx**

- The post-decrementing of the DMA_CNDTRx register, which contains the number of transactions that have still to be performed.
Пост-декремент регистра **DMA_CNDTRx**, который содержит число транзакций, которые еще надо выполнить.

10.3.2 Arbiter (Арбитр)

The arbiter manages the channel requests based on their priority and launches the peripheral/memory access sequences.
Арбитр управляет запросами каналов, на основе их приоритетов, и запускает последовательность доступа к периферии/памяти.

The priorities are managed in two stages: (Приоритетами управляют на двух уровнях:)

- Software: each channel priority can be configured in the DMA_CCRx register. There are four levels:
Программный: приоритет каждого канала может быть сконфигурирован в регистре **DMA_CCRx**. Есть четыре уровня:

  – Very high priority (Очень высокий приоритет)
  – High priority (Высокий приоритет)
  – Medium priority (Средний приоритет)
  – Low priority (Низкий приоритет)

- Hardware: if 2 requests have the same software priority level, the channel with the lowest number will get priority versus the channel with the highest number. For example, channel 2 gets priority over channel 4.
Аппаратный: если два запроса имеют одинаковый программный приоритет, то канал с более низким порядковым числом получит приоритет над каналом с более высоким порядковым числом. Например, канал 2 получит приоритет над каналом 4.

***Note**: In high-density and connectivity line devices, the DMA1 controller has priority over the DMA2 controller. (В устройствах семейства **HD** и **CL**, контроллер **DMA1** имеет приоритет над контроллером **DMA2**.)*

10.3.3 DMA channels (DMA каналы)

Each channel can handle DMA transfer between a peripheral register located at a fixed address and a memory address. The amount of data to be transferred (up to 65535) is programmable. The register which contains the amount of data items to be transferred is decremented after each transaction.
Каждый канал может обработать **DMA** обмен между регистром периферии, имеющему фиксированный адрес, и адресом в памяти. Количество данных, которые будет передано (до 65535 элементов), программируется. Регистр, который содержит количество данных, которые надо передать, будет декрементироваться после каждой транзакции.

Programmable data sizes (Программируемый размер обмена)

Transfer data sizes of the peripheral and memory are fully programmable through the PSIZE and MSIZE bits in the DMA_CCRx register.
Размеры элементов данных обмена для периферии и памяти полностью программируется с помощью битовых полей **PSIZE** и **MSIZE** в регистре **DMA_CCRx**.

Pointer incrementation (Инкремент указателя)

Peripheral and memory pointers can optionally be automatically post-incremented after each transaction depending on the PINC and MINC bits in the DMA_CCRx register. If incremented mode is enabled, the address of the next transfer will be the address of the previous one incremented by 1, 2 or 4 depending on the chosen data size. The first transfer address is the one programmed in the DMA_CPARx/DMA_CMARx registers. During transfer operations, these registers keep the initially programmed value. The current transfer addresses (in the current internal peripheral/memory address register) are not accessible by software. Указатели для периферии и памяти могут, по желанию, автоматически инкрементироваться после каждой транзакции, в зависимости от битов **PINC** и **MINC** в регистре **DMA_CCRx**. Если разрешен режим инкремента, то адрес следующей транзакции будет адресом предыдущей транзакции, увеличенный на 1, 2 или 4, в зависимости от выбранного размера элемента данных. Первый адрес обмена - это тот адрес, что запрограммирован в регистрах **DMA_CPARx**, **DMA_CMARx**. Во время операций обмена эти регистры сохраняют первоначально запрограммированное значение. Текущие адреса обмена (во внутреннем регистре текущего адреса периферии/памяти) не доступны программе.

If the channel is configured in noncircular mode, no DMA request is served after the last transfer (that is once the number of data items to be transferred has reached zero). In order to reload a new number of data items to be transferred into the DMA_CNDTRx register, the DMA channel must be disabled. Если канал сконфигурирован в нециклическом режиме, то, после последней транзакции, никакой **DMA** запрос не обслуживается (это потому, что число элементов данных, которые надо передавать, достигло нуля). Чтобы загрузить в регистр **DMA_CNDTRx** новое число элементов данных, которые надо передавать, **DMA** канал должен быть заблокирован.

***Note**: If a DMA channel is disabled, the DMA registers are not reset. The DMA channel registers (DMA_CCRx, DMA_CPARx and DMA_CMARx) retain the initial values programmed during the channel configuration phase.*
***Note**: Если **DMA** канал блокируется, то **DMA** регистр не сбрасывается. Регистры **DMA** каналов (**DMA_CCRx**, **DMA_CPARx** и **DMA_CMARx**) сохраняют начальные значения, запрограммированные на этапе конфигурации канала.*

In circular mode, after the last transfer, the DMA_CNDTRx register is automatically reloaded with the initially programmed value. The current internal address registers are reloaded with the base address vaules from the DMA_CPARx/DMA_CMARx registers. В циклическом режиме, после последней транзакции, регистр **DMA_CNDTRx** автоматически перезагружается первоначально запрограммированным значением. Внутренние регистры текущего адреса периферии/памяти перезагружаются значениями из регистров базового адреса **DMA_CPARx/DMA_CMARx**.

Channel configuration procedure (Процедура конфигурации каналов)

The following sequence should be followed to configure a DMA channelx (where x is the channel number). Чтобы сконфигурировать **DMA** канал **channelx** (где **x** - номер канала), необходимо выполнить следующую последовательность.

1. Set the peripheral register address in the DMA_CPARx register. The data will be moved from/ to this address to/ from the memory after the peripheral event.
Задайте адрес регистра периферии в регистре **DMA_CPARx**. Данные будут перемещены из/по этому адресу в/из памяти после события от периферии.

2. Set the memory address in the DMA_CMARx register. The data will be written to or read from this memory after the peripheral event. Задайте адрес памяти в регистре **DMA_CMARx**. Данные будут записаны в или прочитаны из этой памяти после события на периферии.

3. Configure the total number of data to be transferred in the DMA_CNDTRx register. After each peripheral event, this value will be decremented. Задайте общее число данных (*в байтах?*), которые надо переместить, в регистре **DMA_CNDTRx**.

4. Configure the channel priority using the PL[1:0] bits in the DMA_CCRx register
Задайте приоритет канала с помощью битов **PL[1:0]** в регистре **DMA_CCRx**.

5. Configure data transfer direction, circular mode, peripheral & memory incremented mode, peripheral & memory data size, and interrupt after half and/or full transfer in the DMA_CCRx register Задайте направление перемещения данных, режим цикличности, режим инкремента периферии и памяти, размер элемента данных периферии и памяти, и источник прерывания (завершение половины или всего обмена) в регистре **DMA_CCRx**.

6. Activate the channel by setting the ENABLE bit in the DMA_CCRx register.
Активируйте канал установкой бита **ENABLE** в регистре **DMA_CCRx**.

As soon as the channel is enabled, it can serve any DMA request from the peripheral connected on the channel. Как только канал будет разрешен, он сможет обслуживать **DMA** запросы от периферии, подключенной к каналу.

Once half of the bytes are transferred, the half-transfer flag (HTIF) is set and an interrupt is generated if the Half-Transfer Interrupt Enable bit (HTIE) is set. At the end of the transfer, the Transfer Complete Flag (TCIF) is set and an interrupt is generated if the Transfer Complete Interrupt Enable bit (TCIE) is set.
Как только будет передана половина байтов, будет установлен флаг полуобмена **HTIF**, и будет сгенерировано прерывание, если такое прерывание разрешено битом **HTIE**. В конце обмена будет установлен флаг Завершения Обмена **TCIF**, и будет сгенерировано прерывание, если такое прерывание разрешено битом **TCIE**.

Circular mode (Режим цикличности)

Circular mode is available to handle circular buffers and continuous data flows (e.g. ADC scan mode). This feature can be enabled using the CIRC bit in the DMA_CCRx register. When circular mode is activated, the number of data to be transferred is automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.
Можно использовать режим цикличности для управления непрерывным потоком данных с помощью кольцевых буферов (например, от АЦП в режиме сканирования). Эта особенность может быть разрешена битом **CIRC** в регистре **DMA_CCRx**. Когда режим цикличности активизирован, то значение числа данных, которые будут переданы (*за один цикл*), автоматически перезагружается начальным значением, запрограммированным на этапе конфигурации канала, и **DMA** запросы продолжают обслуживаться.

Memory-to-memory mode (Режим "память-память")

The DMA channels can also work without being triggered by a request from a peripheral. This mode is called Memory to Memory mode. **DMA** каналы могут также работать без запроса от периферии. Такой режим называют "память-память".

If the MEM2MEM bit in the DMA_CCRx register is set, then the channel initiates transfers as soon as it is enabled by software by setting the Enable bit (EN) in the DMA_CCRx register. The transfer stops once the DMA_CNDTRx register reaches zero. Memory to Memory mode may not be used at the same time as Circular mode. Если установлен бит **MEM2MEM** в регистре **DMA_CCRx**, то канал инициализирует обмен, как только он разрешается в программе битом **EN** в регистре **DMA_CCRx**. Обмен останавливается, как только значение в регистре **DMA_CNDTRx** достигает нуля. Режим "память-память" не может использоваться одновременно с режимом цикличности.

10.3.4 Programmable data width, data alignment and endians
Программирование выравнивания данных и порядка следования байтов

When PSIZE and MSIZE are not equal, the DMA performs some data alignments as described in Table 55: Programmable data width & endian behavior (when bits PINC = MINC = 1).
Когда значения в битовых полях **PSIZE** и **MSIZE** различны, то **DMA** выполняет некоторое выравнивание данных, как это описано в табл. ниже.

Table 55. Programmable data width & endian behavior (when bits PINC = MINC = 1)
Поведение при программировании ширины данных и порядка следования байтов

Source port width	Destination port width	Number of data items to transfer (NDT)	Source content: address / data	Transfer operations	Destination content: address / data
8	8	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: READ B0[7:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B1[7:0] @0x1 then WRITE B1[7:0] @0x1 3: READ B2[7:0] @0x2 then WRITE B2[7:0] @0x2 4: READ B3[7:0] @0x3 then WRITE B3[7:0] @0x3	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3
8	16	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: READ B0[7:0] @0x0 then WRITE 00B0[15:0] @0x0 2: READ B1[7:0] @0x1 then WRITE 00B1[15:0] @0x2 3: READ B3[7:0] @0x2 then WRITE 00B2[15:0] @0x4 4: READ B4[7:0] @0x3 then WRITE 00B3[15:0] @0x6	@0x0 / 00B0 @0x2 / 00B1 @0x4 / 00B2 @0x6 / 00B3
8	32	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: READ B0[7:0] @0x0 then WRITE 000000B0[31:0] @0x0 2: READ B1[7:0] @0x1 then WRITE 000000B1[31:0] @0x4 3: READ B3[7:0] @0x2 then WRITE 000000B2[31:0] @0x8 4: READ B4[7:0] @0x3 then WRITE 000000B3[31:0] @0xC	@0x0 / 000000B0 @0x4 / 000000B1 @0x8 / 000000B2 @0xC / 000000B3
16	8	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: READ B1B0[15:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE B2[7:0] @0x1 3: READ B5B4[15:0] @0x4 then WRITE B4[7:0] @0x2 4: READ B7B6[15:0] @0x6 then WRITE B6[7:0] @0x3	@0x0 / B0 @0x1 / B2 @0x2 / B4 @0x3 / B6
16	16	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: READ B1B0[15:0] @0x0 then WRITE B1B0[15:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE B3B2[15:0] @0x2 3: READ B5B4[15:0] @0x4 then WRITE B5B4[15:0] @0x4 4: READ B7B6[15:0] @0x6 then WRITE B7B6[15:0] @0x6	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6
16	32	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: READ B1B0[15:0] @0x0 then WRITE 0000B1B0[31:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE 0000B3B2[31:0] @0x4 3: READ B5B4[15:0] @0x4 then WRITE 0000B5B4[31:0] @0x8 4: READ B7B6[15:0] @0x6 then WRITE 0000B7B6[31:0] @0xC	@0x0 / 0000B1B0 @0x4 / 0000B3B2 @0x8 / 0000B5B4 @0xC / 0000B7B6
32	8	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B4[7:0] @0x1 3: READ BBBAB9B8[31:0] @0x8 then WRITE B8[7:0] @0x2 4: READ BFBEBDBC[31:0] @0xC then WRITE BC[7:0] @0x3	@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC
32	16	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B1B0[7:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B5B4[7:0] @0x1 3: READ BBBAB9B8[31:0] @0x8 then WRITE B9B8[7:0] @0x2 4: READ BFBEBDBC[31:0] @0xC then WRITE BDBC[7:0] @0x3	@0x0 / B1B0 @0x2 / B5B4 @0x4 / B9B8 @0x6 / BDBC
32	32	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B3B2B1B0[31:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B7B6B5B4[31:0] @0x4 3: READ BBBAB9B8[31:0] @0x8 then WRITE BBBAB9B8[31:0] @0x8 4: READ BFBEBDBC[31:0] @0xC then WRITE BFBEBDBC[31:0] @0xC	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC

Addressing an AHB peripheral that does not support byte or halfword write operations
Адресация той периферии на шине AHB, что не поддерживает доступ байтом или полусловом

When the DMA initiates an AHB byte or halfword write operation, the data are duplicated on the unused lanes of the HWDATA[31:0] bus. So when the used AHB slave peripheral does not support byte or halfword write operations (when HSIZE is not used by the peripheral) and does not generate any error, the DMA writes the 32 HWDATA bits as shown in the two examples below:
Когда **DMA** инициализирует операцию записи байта или полуслова на шине **AHB**, то такие

данные дублируются на неиспользуемой ширине шины **HWDATA[31:0]**. Поэтому, когда используется **slave** периферия на шине **AHB**, которая не поддерживает доступ байтом или полусловом (когда она не "видит" значение **HSIZE**) и при этом она не генерирует ошибки, то **DMA** записывает в **HWDATA** 32 бита, как показано в двух примерах ниже:

- To write the halfword "0xABCD", the DMA sets the HWDATA bus to "0xABCDABCD" with HSIZE = HalfWord
Чтобы записать полуслово "**0xABCD**", **DMA** выставляет на шине **HWDATA** значение "**0xABCDABCD**" и ставит **HSIZE=HalfWord**.

- To write the byte "0xAB", the DMA sets the HWDATA bus to "0xABABABAB" with HSIZE = Byte
Чтобы записать байт "**0xAB**", **DMA** выставляет на шине **HWDATA** значение "**0xABABABAB**" и ставит **HSIZE=Byte**.

Assuming that the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take the HSIZE data into account, it will transform any AHB byte or halfword operation into a 32-bit APB operation in the following manner:
Предположим, что **AHB/APB** мост - это 32-х разрядная **slave AHB** периферия, которая не принимает во внимание значение **HSIZE**, тогда она преобразует любую операцию с байтом или полусловом на **AHB** в 32-х разрядную **APB** операцию следующим образом:

- an AHB byte write operation of the data "0xB0" to 0x0 (or to 0x1, 0x2 or 0x3) will be converted to an APB word write operation of the data "0xB0B0B0B0" to 0x0
операция перезаписи байта на **AHB** из **0xB0** в **0x0** (или любое другое значение) будет преобразована в операцию перезаписи слова из **0xB0B0B0B0** в **0x0**

- an AHB halfword write operation of the data "0xB1B0" to 0x0 (or to 0x2) will be converted to an APB word write operation of the data "0xB1B0B1B0" to 0x0
операция перезаписи полуслова на **AHB** из **0xB1B0** в **0x0** будет преобразована в операцию перезаписи слова из **0xB1B0B1B0** в **0x0**

For instance, if you want to write the APB backup registers (16-bit registers aligned to a 32-bit address boundary), you must configure the memory source size (MSIZE) to "16-bit" and the peripheral destination size (PSIZE) to "32-bit".
Например, если вы хотите записать в **APB BKP** регистры резервирования (это 16-ти разрядные регистры, выровненные по 32-х разрядной границе адреса), вы должны сконфигурировать размер памяти источника **MSIZE** как 16-бит, а размер периферии приемника **PSIZE** как 32 бита.

10.3.5 Error management (Обработка ошибок)

A DMA transfer error can be generated by reading from or writing to a reserved address space. When a DMA transfer error occurs during a DMA read or a write access, the faulty channel is automatically disabled through a hardware clear of its EN bit in the corresponding Channel configuration register (DMA_CCRx). The channel's transfer error interrupt flag (TEIF) in the DMA_IFR register is set and an interrupt is generated if the transfer error interrupt enable bit (TEIE) in the DMA_CCRx register is set.
Генерация ошибки обмена по **DMA** может возникнуть при операции чтении или записи в зарезервированном адресном пространстве. Когда происходит ошибка обмена по **DMA** во время доступа на чтение или запись, то такой канал автоматически отключается посредством аппаратного сброса бита **EN** в регистре конфигурации соответствующего канала (**DMA_CCRx**). Выставляется флаг запроса прерывания от ошибки обмена в канале (**TEIF**) в регистре **DMA_IFR**, и генерируется прерывание, если оно разрешено битом **TEIE** в регистре **DMA_CCRx**.

10.3.6 Interrupts (Прерывания)

An interrupt can be produced on a Half-transfer, Transfer complete or Transfer error for each DMA channel. Separate interrupt enable bits are available for flexibility.
Запрос на прерывание может генерироватся событиями "Завершения Половины Обмена", "Завершения Полного Обмена" или "Ошибкой обмена" в каждом **DMA** канале. Для удобства, доступны отдельное биты разрешения прерываний.

Table 56. DMA interrupt requests (Запросы прерываний от **DMA**)

Interrupt event	Event flag	Enable Control bit
Half-transfer	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE

***Note**: In high-density devices, DMA2 Channel4 and DMA2 Channel5 interrupts are mapped onto the same interrupt vector. In connectivity line devices, DMA2 Channel4 and DMA2 Channel5 interrupts have separate interrupt vectors. All other DMA1 and DMA2 Channel interrupts have their own interrupt vector.*
***Note**: В устройствах семейства **HD**, прерывания от каналов **DMA2 Channel4** и **DMA2 Channel5** отображаются на один вектор прерывания. В устройствах семейства **CL** прерывания от каналов **DMA2 Channel4** и **DMA2 Channel5** имеют отдельные векторы прерывания. Прерывания всех других каналов контроллеров **DMA1** и **DMA2** имеют свой собственный вектор прерывания.*

10.3.7 DMA request mapping (Отображение DMA запросов)

DMA1 controller (Контроллер DMA1)

The 7 requests from the peripherals (TIMx[1,2,3,4], ADC1, SPI1, SPI/I2S2, I2Cx[1,2] and USARTx[1,2,3]) are simply logically ORed before entering DMA1, this means that only one request must be enabled at a time. Refer to Figure 23: DMA1 request mapping.
Семь запросов от периферии (**TIMx[1,2,3,4]**, **ADC1**, **SPI1**, **SPI/I2S2**, **I2Cx[1,2]** и **USARTx[1,2,3]**) логически объединяются (по **OR**) прежде, чем поступить на **DMA1**, это означает, что в одно время может быть разрешен только один запрос. Обратитесь к рис. 23: "Отображение запросов **DMA1**".

The peripheral DMA requests can be independently activated/de-activated by programming the DMA control bit in the registers of the corresponding peripheral.
Запросы от периферии на **DMA** можно независимо активировать/деактивировать, программируя бит управления **DMA** в регистрах соответствующего внешнего устройства.

Figure 23. DMA1 request mapping (Отображение запросов на **DMA1**)

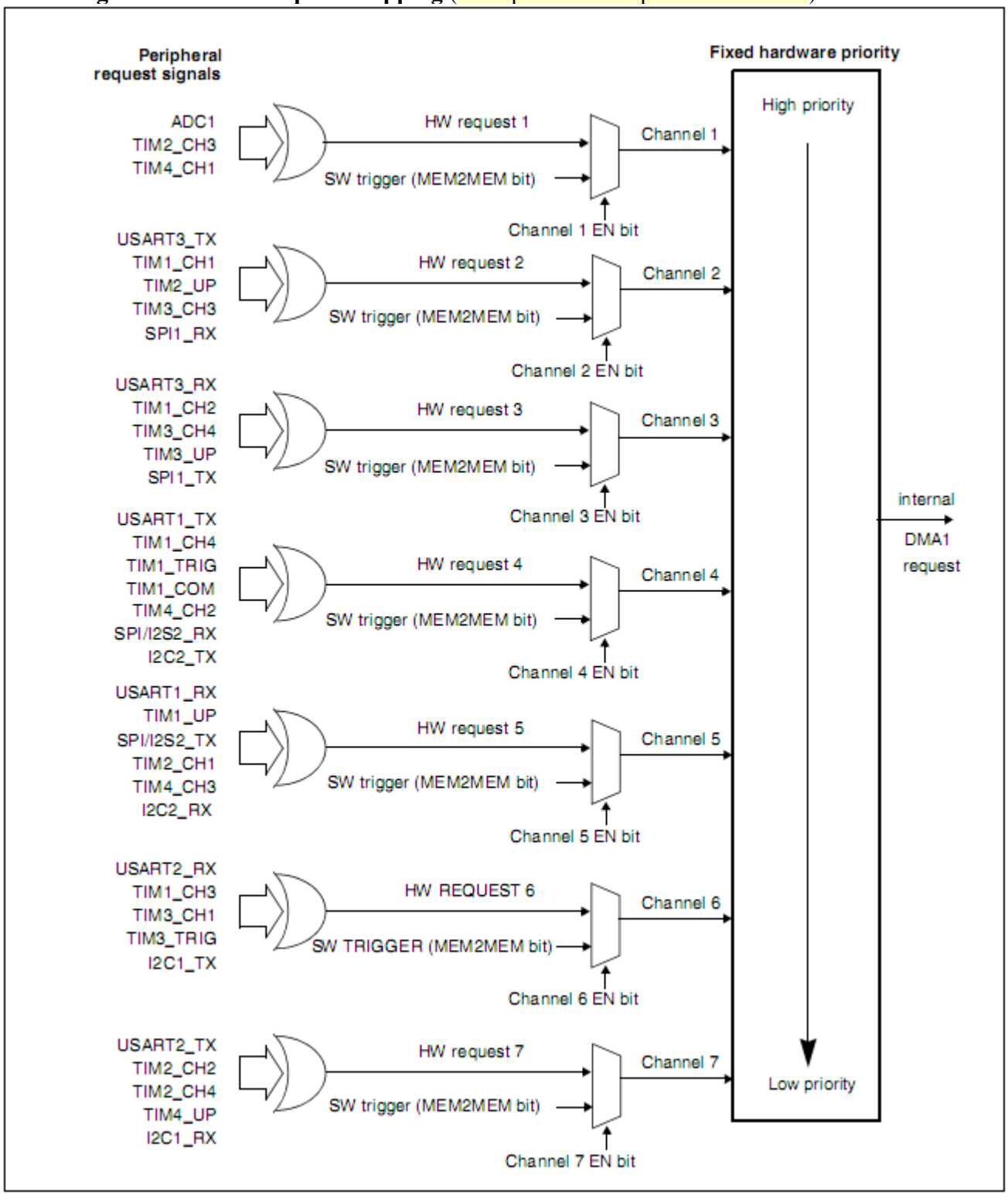


Table 57 lists the DMA requests for each channel.

В таблице 57 приведен список запросов на **DMA** для каждого канала.

Table 57. Summary of DMA1 requests for each channel
Сводная таблица запросов на **DMA1** для каждого канала

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
ADC1	ADC1						
SPI/I2S		SPI1_RX	SPI1_TX	SPI/I2S2_RX	SPI/I2S2_TX		
USART		USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I2C				I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1		TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
TIM2	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2 TIM2_CH4
TIM3		TIM3_CH3	TIM3_CH4 TIM3_UP			TIM3_CH1 TIM3_TRIG	
TIM4	TIM4_CH1			TIM4_CH2	TIM4_CH3		TIM4_UP

DMA2 controller (Контроллер DMA2)

The 5 requests from the peripherals (TIMx[5,6,7,8], ADC3, SPI/I2S3, UART4, DAC_Channel[1,2] and SDIO) are simply logically ORed before entering to the DMA2, this means that only one request must be enabled at a time. Refer to Figure 24: DMA2 request mapping.
Пять запросов от периферии (**TIMx[5,6,7,8]**, **ADC3**, **SPI/I2S3**, **USARTx[4]**, **DAC_Channel[1,2]** и **SDIO**) логически объединяются (по **OR**) прежде, чем поступить на **DMA2**, это означает, что в одно время может быть разрешен только один запрос. Обратитесь к рис. 24: "Отображение запросов **DMA2**".

The peripheral DMA requests can be independently activated/de-activated by programming the DMA control bit in the registers of the corresponding peripheral.
Запросы от периферии на **DMA** можно независимо активировать/деактивировать, программируя бит управления **DMA** в регистрах соответствующего внешнего устройства.

***Note**: The DMA2 controller and its relative requests are available only in high-density and connectivity line devices. (Контроллер **DMA2** и соответствующие ему запросы доступны только в семействах **HD** и **CL**.)*

Figure 24. DMA2 request mapping (Отображение запросов на **DMA2**)

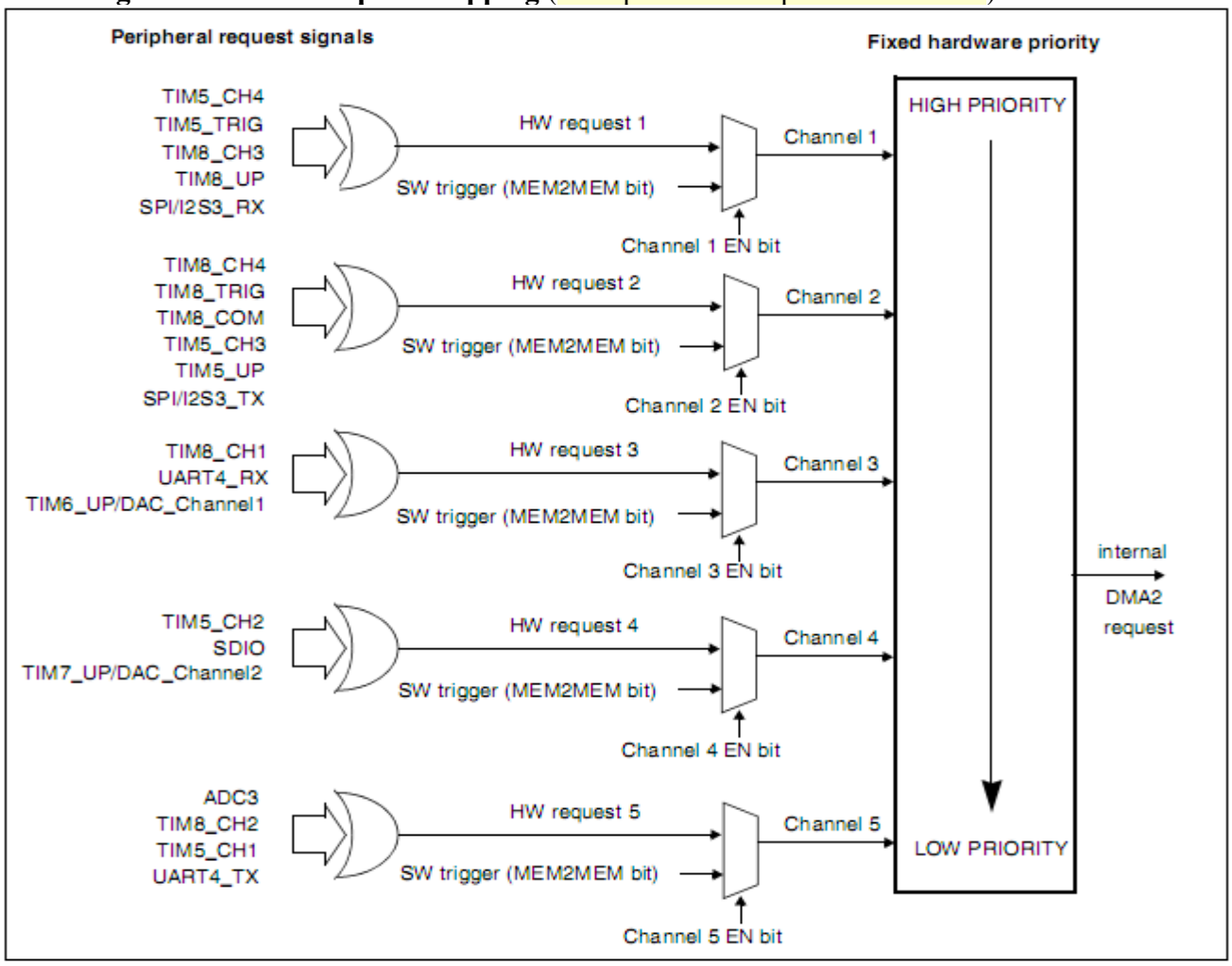


Table 58 lists the DMA2 requests for each channel.
В таблице 58 приведен список запросов на **DMA2** для каждого канала.

Table 58. Summary of DMA2 requests for each channel
Сводная таблица запросов на **DMA2** для каждого канала

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
ADC3(1)					ADC3
SPI/I2S3	SPI/I2S3_RX	SPI/I2S3_TX			
UART4			UART4_RX		UART4_TX
SDIO(1)				SDIO	
TIM5	TIM5_CH4 TIM5_TRIG	TIM5_CH3 TIM5_UP		TIM5_CH2	TIM5_CH1
TIM6/ DAC_Channel1			TIM6_UP/ DAC_Channel1		
TIM7/ DAC_Channel2				TIM7_UP/ DAC_Channel2	
TIM8(1)	TIM8_CH3 TIM8_UP	TIM8_CH4 TIM8_TRIG TIM8_COM	TIM8_CH1		TIM8_CH2

1. ADC3, SDIO and TIM8 DMA requests are available only in high-density devices.
Запросы на **DMA** от **ADC3**, **SDIO** и **TIM8** доступны только в семействе **HD**.

10.4 DMA registers (Регистры DMA)

10.4.1 DMA interrupt status register (DMA_ISR) (Регистр статуса прерываний **DMA**)
10.4.2 DMA interrupt flag clear register (DMA_IFCR)
(Регистр очистки флагов прерываний **DMA**)
10.4.3 DMA channel x configuration register (DMA_CCRx) (x = 1 ..7)
Регистр конфигурации канала **x DMA**
10.4.4 DMA channel x number of data register (DMA_CNDTRx) (x = 1 ..7)
Регистр размера данных канала **x DMA**
10.4.5 DMA channel x peripheral address register (DMA_CPARx) (x = 1 ..7)
Регистр адреса периферии для канала **x DMA**
10.4.6 DMA channel x memory address register (DMA_CMARx) (x = 1 ..7)
Регистр адреса памяти для канала **x DMA**
10.4.3 DMA register map (Карта регистров **DMA**)

Refer to Section 1.1 on page 37 for a list of abbreviations used in the register descriptions.
См. раздел 1.1, где приведен перечень сокращений, используемых при описании регистров.

***Note**: In the following registers, all bits relative to channel6 and channel7 are not relevant for DMA2 since it has only 5 channels. (В последующих регистрах все биты, относящиеся к каналам 6 и 7 не существенны для **DMA2**, так как он имеет только пять каналов.)*

10.4.1 DMA interrupt status register (DMA_ISR)
Регистр статуса прерываний DMA

Address offset: 0x00
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved
always read as 0. (Всегда читается как '0'.)

Bits 27, 23, 19, 15, 11, 7, 3 TEIFx: Channel x transfer error flag (x = 1 ..7)
Флаг ошибки обмена в канале x
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register. (Этот бит ставится аппаратно. Очищается программно записью '1' в соответствующий бит регистра **DMA_IFCR**.)
`0:` No transfer error (TE) on channel x (Нет ошибки обмена в канале **x**)
`1:` A transfer error (TE) occurred on channel x (Есть ошибка обмена в канале **x**)

Bits 26, 22, 18, 14, 10, 6, 2 HTIFx: Channel x half transfer flag (x = 1 ..7)
Флаг завершения половины обмена в канале x
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register. (Этот бит ставится аппаратно. Очищается программно записью '1' в соответствующий бит регистра **DMA_IFCR**.)
`0:` No half transfer (HT) event on channel x
Нет события завершения половины обмена в канале **x**
`1:` A half transfer (HT) event occurred on channel x
Есть событие завершения половины обмена в канале **x**

Bits 25, 21, 17, 13, 9, 5, 1 TCIFx: Channel x transfer complete flag (x = 1 ..7)
Флаг завершения обмена в канале x

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register. (Этот бит ставится аппаратно. Очищается программно записью '1' в соответствующий бит регистра **DMA_IFCR**.)

`0:` No transfer complete (TC) event on channel x

Нет события завершения обмена в канале **x**

`1:` A transfer complete (TC) event occurred on channel x

Есть событие завершения обмена в канале **x**

Bits 24, 20, 16, 12, 8, 4, 0 GIFx: Channel x global interrupt flag (x = 1 ..7)
Общий флаг прерывания в канале x

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register. (Этот бит ставится аппаратно. Очищается программно записью '1' в соответствующий бит регистра **DMA_IFCR**.)

`0:` No TE, HT or TC event on channel x

Нет ни одного события прерывания **TE**, **HT** или **TC**

`1:` A TE, HT or TC event occurred on channel x

Есть одно из событий прерывания **TE**, **HT** или **TC**

10.4.2 DMA interrupt flag clear register (DMA_IFCR)
Регистр очистки флагов прерываний DMA

Address offset: 0x04
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CTEIF 7	CHTIF 7	CTCIF 7	CGIF 7	CTEIF 6	CHTIF 6	CTCIF 6	CGIF 6	CTEIF 5	CHTIF 5	CTCIF 5	CGIF 5
				w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTEIF 4	CHTIF 4	CTCIF 4	CGIF 4	CTEIF 3	CHTIF 3	CTCIF 3	CGIF 3	CTEIF 2	CHTIF 2	CTCIF 2	CGIF 2	CTEIF 1	CHTIF 1	CTCIF 1	CGIF 1
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:28 Reserved

always read as 0. (Всегда читается как '0'.)

Bits 27, 23, 19, 15, 11, 7, 3 CTEIFx: Channel x transfer error clear (x = 1 ..7)
Очистка флага ошибки обмена в канале x

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

`0:` No effect (Нет эффекта)

`1:` Clears the corresponding TEIF flag in the DMA_ISR register

Очистка соответствующего **TEIF** флага в регистре **DMA_ISR**

Bits 26, 22, 18, 14, 10, 6, 2 CHTIFx: Channel x half transfer clear (x = 1 ..7)
Очистка флага завершения половины обмена в канале x

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

`0:` No effect (Нет эффекта)

`1:` Clears the corresponding HTIF flag in the DMA_ISR register

Очистка соответствующего **HTIF** флага в регистре **DMA_ISR**

Bits 25, 21, 17, 13, 9, 5, 1 CTCIFx: Channel x transfer complete clear (x = 1 ..7)
Очистка флага завершения обмена в канале x

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: No effect (Нет эффекта)
1: Clears the corresponding TCIF flag in the DMA_ISR register

Очистка соответствующего **TCIF** флага в регистре **DMA_ISR**

Bits 24, 20, 16, 12, 8, 4, 0 CGIFx: Channel x global interrupt clear (x = 1 ..7)
Глобальная очистка флагов прерывания канала x

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: No effect (Нет эффекта)
1: Clears the GIF, TEIF, HTIF and TCIF flags in the DMA_ISR register

Очистка флагов **GIF**, **TEIF**, **HTIF** и **TCIF** в регистре **DMA_ISR**

10.4.3 DMA channel x configuration register (DMA_CCRx) (x = 1 ..7)
Регистр конфигурации канала x DMA

Address offset: 0x08 + 20d × Channel number
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved

always read as 0. (Всегда читается как '0'.)

Bit 14 MEM2MEM: Memory to memory mode (Режим "память-память")

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: Memory to memory mode disabled (Режим "память-память" отключен)
1: Memory to memory mode enabled (Режим "память-память" разрешен)

Bits 13:12 PL[1:0]: Channel priority level (Уровень приоритета канала)

These bits are set and cleared by software. (Этот бит ставится и очищается программно.)

00: Low (Низкий)
01: Medium (Средний)
10: High (Высокий)
11: Very high (Очень высокий)

Bits 11:10 MSIZE[1:0]: Memory size (Размер элемента данных в памяти)

These bits are set and cleared by software. (Этот бит ставится и очищается программно.)

00: 00: 8-bits
01: 16-bits
10: 32-bits
11: Reserved

Bits 9:8 PSIZE[1:0]: Peripheral size (Размер элемента данных в периферии)

These bits are set and cleared by software. (Этот бит ставится и очищается программно.)

00: 8-bits
01: 16-bits
10: 32-bits
11: Reserved

Bit 7 MINC: Memory increment mode (Режим инкремента указателя в памяти)

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: Memory increment mode disabled (Режим инкремента указателя в памяти отключен)

1: Memory increment mode enabled (Режим инкремента указателя в памяти разрешен)

Bit 6 PINC: Peripheral increment mode (Режим инкремента указателя в периферии)

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: Peripheral increment mode disabled

Режим инкремента указателя в периферии отключен

1: Peripheral increment mode enabled

Режим инкремента указателя в периферии разрешен

Bit 5 CIRC: Circular mode (Режим цикличности)

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: Circular mode disabled (Режим цикличности отключен)

1: Circular mode enabled (Режим цикличности разрешен)

Bit 4 DIR: Data transfer direction (Направление обмена данных)

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: Read from peripheral (Чтение из периферии)

1: Read from memory (Чтение из памяти)

Bit 3 TEIE: Transfer error interrupt enable (Разрешение прерывания от ошибки обмена)

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: TE interrupt disabled (Прерывание от **TE** запрещено)

1: TE interrupt enabled (Прерывание от **TE** разрешено)

Bit 2 HTIE: Half transfer interrupt enable
Разрешение прерывания от события завершения половины обмена

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: HT interrupt disabled (Прерывание от **HT** запрещено)

1: HT interrupt enabled (Прерывание от **HT** разрешено)

Bit 1 TCIE: Transfer complete interrupt enable
Разрешение прерывания от события завершения обмена

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: TC interrupt disabled (Прерывание от **TC** запрещено)

1: TC interrupt enabled (Прерывание от **TC** разрешено)

Bit 0 EN: Channel enable (Разрешение канала)

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: Channel disabled (Канал отключен)

1: Channel enabled (Канал разрешен)

10.4.4 DMA channel x number of data register (DMA_CNDTRx) (x = 1 ..7)
Регистр размера данных канала x DMA

Address offset: 0x0C + 20d × Channel number
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDT															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved

always read as 0. (Всегда читается как '0'.)

Bits 15:0 NDT[15:0]: Number of data to transfer (Размер данных обмена)

Number of data to be transferred (0 up to 65535). This register can only be written when the channel is disabled. Once the channel is enabled, this register is read-only, indicating the remaining bytes to be transmitted. This register decrements after each DMA transfer.
Число данных (*в байтах?*), которые будут переданы (0 до 65535). В этот регистр можно производить запись только тогда, когда канал выключен. Как только канал будет разрешен, этот регистр можно будет только читать, чтобы определить число байт, оставшееся для обмена. Этот регистр декрементируется после каждой **DMA** транзакции.

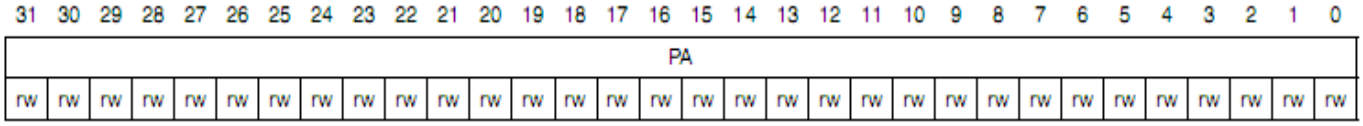
Once the transfer is completed, this register can either stay at zero or be reloaded automatically by the value previously programmed if the channel is configured in uto-reload mode.
Как только обмен завершен, этот регистр может либо остаться в нуле, либо автоматически перезагрузиться ранее запрограммированным значением, если канал сконфигурирован в режиме авто-перезагрузки (режим цикличности).

If this register is zero, no transaction can be served whether the channel is enabled or not.
Если значение этого регистра равно нулю, никакая транзакция не может быть обслужена, независимо от того, разрешен канал, или нет.

10.4.5 DMA channel x peripheral address register (DMA_CPARx) (x = 1 ..7)
Регистр адреса периферии для канала x DMA

Address offset: 0x10 + dx20 × Channel number
Reset value: 0x0000 0000

This register must not be written when the channel is enabled.
В этот регистр нельзя производить запись, когда канал разрешен.



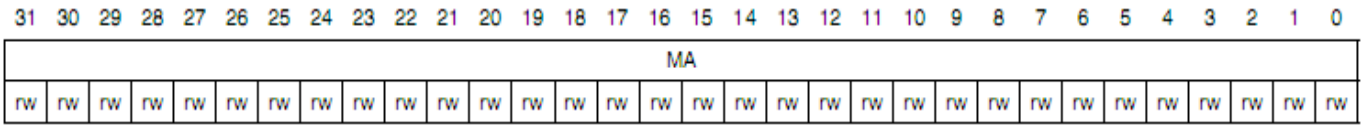
Bits 31:0 PA[31:0]: Peripheral address (Адрес периферии)

Base address of the peripheral data register from/to which the data will be read/written.
When PSIZE is 01 (16-bit), the PA[0] bit is ignored. Access is automatically aligned to a half-word address. When PSIZE is 10 (32-bit), PA[1:0] are ignored. Access is automatically aligned to a word address.
Регистр базового адреса данных в периферии, из/по которому будут читаться/записываться данные. Когда **PSIZE=01** (элемент 16-ти битный), бит **PA[0]** игнорируется. Доступ автоматически выравнивается по адресу полуслова. Когда **PSIZE=10** (элемент 32-х битный), биты PA[1:0] игнорируются. Доступ автоматически выравнивается по адресу слова.

10.4.6 DMA channel x memory address register (DMA_CMARx) (x = 1 ..7)
Регистр адреса памяти для канала x DMA

Address offset: 0x14 + dx20 × Channel number
Reset value: 0x0000 0000

This register must not be written when the channel is enabled.
В этот регистр нельзя производить запись, когда канал разрешен.



Bits 31:0 MA[31:0]: Memory address (Адрес в памяти)

Base address of the memory area from/to which the data will be read/written.
When MSIZE is 01 (16-bit), the MA[0] bit is ignored. Access is automatically aligned to a half-word address. When MSIZE is 10 (32-bit), MA[1:0] are ignored. Access is automatically aligned to a word address.
Регистр базового адреса данных в памяти, из/по которому будут читаться/записываться данные. Когда **MSIZE=01** (элемент 16-ти битный), бит **MA[0]** игнорируется. Доступ автоматически выравнивается по адресу полуслова. Когда **MSIZE=10** (элемент 32-х битный), биты MA[1:0] игнорируются. Доступ автоматически выравнивается по адресу слова.

10.4.7 DMA register map (Карта регистров DMA)

The following table gives the DMA register map and the reset values.
Последующая таблица дает карту регистров модуля **DMA** и их значение после сброса.

Table 59. DMA register map and reset values
Карта регистров **DMA** и их значение после сброса

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DMA_ISR	Reserved				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	DMA_IFCR	Reserved				CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DMA_CCR1	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZ E [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	DMA_CNDTR1	Reserved																	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	DMA_CPAR1	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DMA_CMAR1	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	Reserved																																
0x01C	DMA_CCR2	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZ E [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	DMA_CNDTR2	Reserved																	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	DMA_CPAR2	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	DMA_CMAR2	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	Reserved																																
0x030	DMA_CCR3	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZ E [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	DMA_CNDTR3	Reserved																	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	DMA_CPAR3	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	DMA_CMAR3	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	Reserved																																
0x044	DMA_CCR4	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZ E [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	DMA_CNDTR4	Reserved																	NDT[15:0]														

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x04C	DMA_CPAR4	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	DMA_CMAR4	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	Reserved																																
0x058	DMA_CCR5	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZ E [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	DMA_CNDTR5	Reserved																	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	DMA_CPAR5	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	DMA_CMAR5	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	Reserved																																
0x06C	DMA_CCR6	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZ E [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	DMA_CNDTR6	Reserved																	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	DMA_CPAR6	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078	DMA_CMAR6	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	Reserved																																
0x080	DMA_CCR7	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZ E [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	DMA_CNDTR7	Reserved																	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	DMA_CPAR7	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	DMA_CMAR7	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	Reserved																																

Refer to Table 1 on page 41 for the register boundary addresses.
См. Табл. 1, где даны относительные адреса регистров.

24 Inter-integrated circuit (I^2C) interface (Интерфейс I2C)

24.1 I2C introduction (Введение в **I2C**)
24.2 I2C main features (Основные особенности **I2C**)
24.3 I2C functional description (Функциональное описание **I2C**)
24.4 I2C interrupts (Прерывания от **I2C**)
24.5 I2C debug mode (Режим отладки и **I2C**)
24.6 I2C registers (**I2C** регистры)

(*Здесь находится ритуальное напоминание о наличии 4-х семейств для **STM32F10xxx**, и что этот документ обо всех семействах, если нет прямого указания. См. раздел 1.2 Глоссарий).*)

24.1 I2C introduction (Введение в I2C)

I2C (inter-integrated circuit) bus Interface serves as an interface between the microcontroller and the serial I2C bus. It provides multimaster capability, and controls all I2C bus-specific sequencing, protocol, arbitration and timing. It supports standard and fast speed modes. It is also SMBus 2.0 compatible. **I2C** интерфейс (*inter-integrated circuit*) работает как интерфейс между МК и последовательной **I2C** шиной. Он допускает несколько "Ведущих" на шине и управляет всеми особенностями шины **I2C**, протоколом, арбитражем, временными характеристиками. Он поддерживает высоко-скоростной режим. Он совместим также с **SMBus 2.0**.

It may be used for a variety of purposes, including CRC generation and verification, SMBus (system management bus) and PMBus (power management bus).
Его можно использовать для широкого спектра приложений, включая генерацию и верификацию контрольной суммы, **SMBus** (*system management bus*) и **PMBus** (*power management bus*).

Depending on specific device implementation DMA capability can be available for reduced CPU overload. В зависимости от специфики устройства, использование встроенного **DMA** может уменьшить нагрузку на ЦПУ.

24.2 I2C main features (Основные особенности I2C)

- Parallel-bus/I2C protocol converter (Конвертация параллельных данных в **I2C** протокол)

- Multimaster capability: the same interface can act as Master or Slave (Поддержка нескольких "Ведущих": один и тот же интерфейс может работать как "Ведущий" или как "Ведомый")

- I2C Master features: (Особенности "Ведущего" **I2C**:)
  – Clock generation (Генерирует тактовый сигнал)
  – Start and Stop generation (Генерирует старт- и стоп-состояние)

- I2C Slave features: (Особенности "Подчиненного" **I2C**:)
  – Programmable I2C Address detection
  (Детектирование программируемого **I2C** адреса)
  – Dual Addressing Capability to acknowledge 2 slave addresses
  Двойной адрес: возможность иметь 2 **slave** адреса
  – Stop bit detection (Детектирование стоп-бита)

- Generation and detection of 7-bit/10-bit addressing and General Call
Генерация и детектирование 7-bit/10-bit адреса и общего вызова

- Supports different communication speeds: (Поддерживает различные скорости обмена:)
  – Standard Speed (up to 100 kHz), (Стандартная скорость (до 100 kHz),)
  – Fast Speed (up to 400 kHz) (Высокая скорость (до 400 kHz))

- Status flags: (Флаги статуса:)
  – Transmitter/Receiver mode flag Флаг режима Передача/Прием
  – End-of-Byte transmission flag Флаг конца передачи байта
  – I2C busy flag Флаг готовности **I2C**

- Error flags: (Флаги ошибок:)
  – Arbitration lost condition for master mode (Потеря прав на шину)
  – Acknowledgement failure after address/ data transmission
  Ошибка уведомления после передачи адреса/данных
  – Detection of misplaced start or stop condition
  Обнаружения неуместного старт- или стоп-состояния
  – Overrun/Underrun if clock stretching is disabled
  Переполнение/Недостача, если отключена опция растяжения тактового

- 2 Interrupt vectors: (Два вектора прерываний:)
  – 1 Interrupt for successful address/ data communication
  Один для успешной передачи адреса/данных
  – 1 Interrupt for error condition (Еще один для ошибок)

- Optional Clock Stretching (Опционный режим растяжения тактового)

- 1-byte buffer with DMA capability
Однобайтный буфер, с возможностью использовать **DMA**

- Configurable PEC (Packet Error Checking) Generation or Verification:
Конфигурируемая генерация/проверка контрольной суммы пакета (**PEC**):
  – PEC value can be transmitted as last byte in Tx mode
  Значение **PEC** может быть передано, как последний байт в режиме **Tx**
  – PEC error checking for last received byte
  Проверка ошибки **PEC** в последнем принятом байте

- SMBus 2.0 Compatibility: (Совместимость с **SMBus 2.0**:)
  – 25 ms clock low timeout delay
  Контроль времени низкого состояния на шине 25 мс
  – 10 ms master cumulative clock low extend time Контроль времени времени низкого состояния на шине 10 мс, при растяжении тактового "Ведущим"
  – 25 ms slave cumulative clock low extend time Контроль времени времени низкого состояния на шине 25 мс, при растяжении тактового "Ведомым"
  – Hardware PEC generation/verification with ACK control
  Аппаратное генерирование/проверка контрольной суммы с уведомлением **ACK**
  -- Address Resolution Protocol (ARP) supported
  Поддержка протокола динамических адресов **ARP**

- PMBus Compatibility (Совместимость с **PMBus** устройствами)

***Note**: Some of the above features may not be available in certain products. The user should refer to the product data sheet, to identify the specific features supported by the I2C interface implementation.*
*Некоторые из выше перечисленных особенностей могут быть недоступны в некоторых изделиях. Вы должны посмотреть описание **I2C** на конкретное изделие.*

24.3 I2C functional description (Функциональное описание I2C)

24.3.1 Mode selection (Выбор режима)
24.3.2 I2C slave mode (**I2C** в режиме "Ведомый")
24.3.3 I2C master mode (**I2C** в режиме "Ведущий")
24.3.4 Error conditions (Условия для ошибки)
24.3.5 SDA/SCL line control (Управление линиями **SDA/SCL**)
24.3.6 **SMBus**
24.3.7 DMA requests (Запрос на использование **DMA**)
24.3.8 Packet error checking (Проверка ошибок пакетирования)

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I2C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz) or fast (up to 400 kHz) I2C bus.
Кроме приема и передачи данных, этот интерфейс конвертирует данные из параллельной формы в последовательную и наоборот. Прерывания разрешаются и отключаются программно. Интерфейс с **I2C** осуществляется подключением линии данных **SDA** и линии тактового **SCL**. Можно подключаться к **I2C** устройству со стандартной скоростью (до 100 кГц) или высокой скоростью (до 400 кГц).

24.3.1 Mode selection (Выбор режима)

The interface can operate in one of the four following modes:
Интерфейс может работать в одном из четырех следующих режимов:

– Slave transmitter (Подчиненный передатчик)
– Slave receiver (Подчиненный приемник)
– Master transmitter (Ведущий передатчик)
– Master receiver (Ведущий приемник)

By default, it operates in slave mode. The interface automatically switches from slave to master, after it generates a START condition and from master to slave, if an arbitration loss or a Stop generation occurs, allowing multimaster capability.
По умолчанию, он работает в режиме "Ведомый". Интерфейс автоматически переключается с "Ведомого" на "Ведущий", после генерирования старт-условия, и с "Ведущего" на "Ведомый", при потере арбитража или после генерирования стоп-условия, что позволяет работать нескольким "Ведущим".

Communication flow (Процедура обмена)

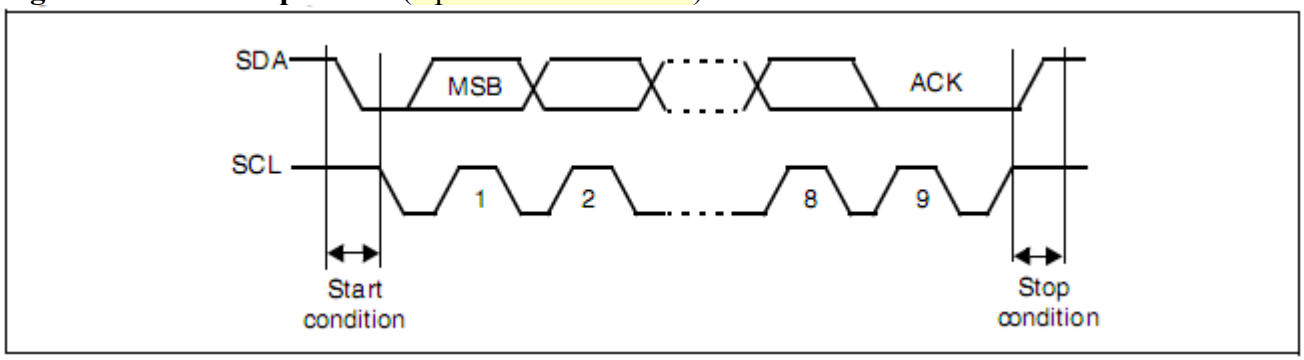
In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a start condition and ends with a stop condition. Both start and stop conditions are generated in master mode by software.
В режиме "Ведущий", **I2C** инициирует обмен данными и генерирует тактовый сигнал. Передаче последовательных данных всегда предшествует старт-условие, а завершается обмен всегда стоп-условием. Оба этих условия генерируются в режиме "Ведущий" программно.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection may be enabled or disabled by software.
В режиме "Ведомый", **I2C** способен распознать свой собственный адрес (7 или 10-bit), и адрес общего вызова. Определение наличия адреса общего вызова можно включить или отключить программно.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the start condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.
Адрес и данные передаются 8-ми битными байтами, старшим битом вперед. Первый байт(ы), следующий за старт-условием, содержит адрес (один байт в 7-ми битном режиме, и два байта в 10-ти битном режиме). Адрес всегда передается в режиме "Ведущий".

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure.
За восемью тактами передачи байта данных следует 9-й такт, в течении которого приемник должен послать бит уведомления. См. следующий рисунок.

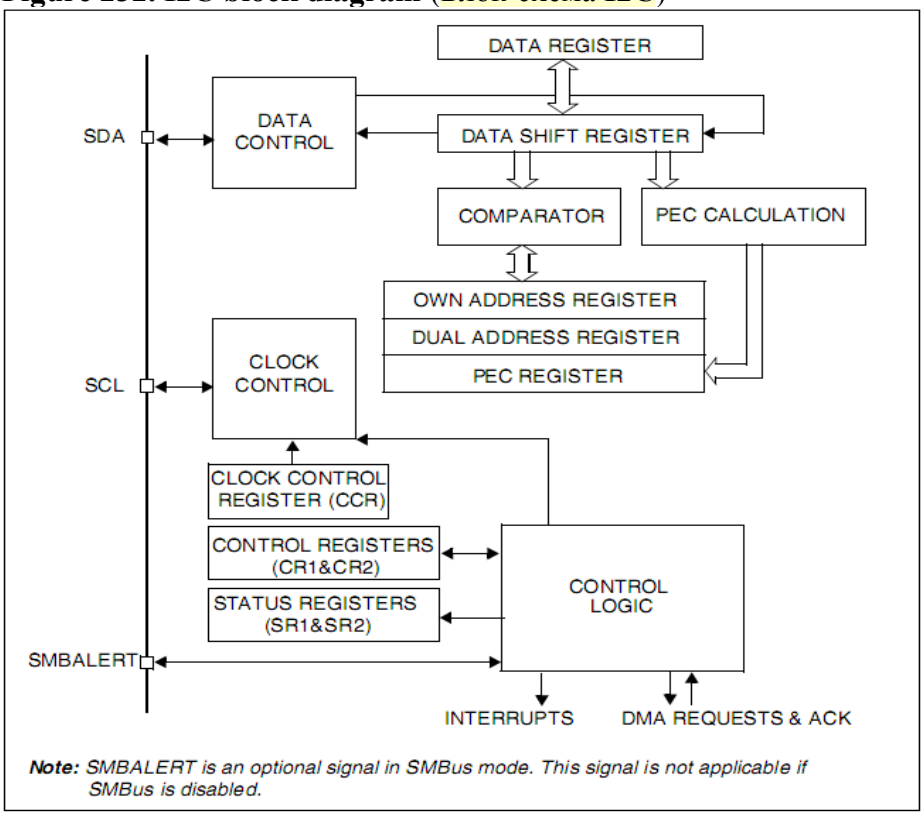
Figure 231. I2C bus protocol (Протокол **I2C** шины)



Acknowledge may be enabled or disabled by software. The I2C interface addresses (dual addressing 7-bit/ 10-bit and/or general call address) can be selected by software.
Наличие уведомления можно включить или отключить программно. Адреса **I2C** (двойной адрес 7/10 бит и/или адрес общего вызова) можно выбрать программно.

The block diagram of the I2C interface is shown in *Figure 232*.
Блок-схема интерфейса **I2C** приведена на рис. 232.

Figure 232. I2C block diagram (Блок-схема **I2C**)



24.3.2 I2C slave mode (I2C в режиме "Ведомый")

Slave transmitter (Ведомый передатчик)
Slave receiver (Ведомый приемник)
Closing slave communication (Закрытие обмена в режиме "Ведомый")

By default the I2C interface operates in Slave mode. To switch from default Slave mode to Master mode a Start condition generation is needed.
По умолчанию **I2C** работает в режиме "Ведомый". Чтобы переключиться из режима по умолчанию "Ведомый" в режим Ведущий", необходимо сгенерировать старт-условие.

The peripheral input clock must be programmed in the I2C_CR2 register in order to generate correct timings. The peripheral input clock frequency must be at least:
Входная частота периферии должна быть сконфигурирована в регистре **I2C_CR2** так, чтобы генерировать корректные временные соотношения (тайминги). Входная частота периферии должна быть не менее:

- 2 MHz in Standard mode (2 мГц в стандартном режиме)
- 4 MHz in Fast mode (4 мГц в быстром режиме)

As soon as a start condition is detected, the address is received from the SDA line and sent to the shift register. Then it is compared with the address of the interface (OAR1) and with OAR2 (if ENDUAL=1) or the General Call address (if ENGC = 1).
Как только обнаружено старт-условие, принимается адрес на входе **SDA** и поступает в сдвиговый регистр. Затем он сравнивается с адресом интерфейса **OAR1** и **OAR2** (если **ENDUAL=1**), или с адресом общего вызова (если **ENGC=1**).

***Note**: In 10-bit addressing mode, the comparison includes the header sequence (11110xx0), where xx denotes the two most significant bits of the address.*
*Если адресация в 10-ти битном режиме, то сравнение включает байт заголовка (**11110xx0**), где **xx** отмечает два старших бита адреса.*

Header or address not matched: the interface ignores it and waits for another Start condition.
Если заголовок или адрес не совпали: то интерфейс игнорирует обмен и ждет следующего старт-условия.

Header matched (10-bit mode only): the interface generates an acknowledge pulse if the ACK bit is set and waits for the 8-bit slave address.
Если заголовок совпал (только для 10-ти битного режима): то интерфейс генерирует бит уведомления (если стоит бит **ACK**) и ждет 8 бит адреса.

Address matched: the interface generates in sequence:
Если адрес совпал: то интерфейс генерирует следующее:

- An acknowledge pulse if the ACK bit is set
бит уведомления (если стоит бит **ACK**)
- The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.
Бит **ADDR** ставится аппаратно и генерируется прерывание, если есть бит **ITEVFEN**.
- If ENDUAL=1, the software has to read the DUALF bit to check which slave address has been acknowledged.
Если **ENDUAL=1**, то программа читает бит **DUALF**, чтобы проверить, какой из адресов для ведомых устройств был принят.

In 10-bit mode, after receiving the address sequence the slave is always in Receiver mode. It will enter Transmitter mode on receiving a repeated Start condition followed by the header sequence with matching address bits and the least significant bit set (11110xx1).
В 10-ти битном режиме, после получения адреса, "Ведомый" всегда находится в режиме приема. Оно войдет в режим передатчика после получения повторного старт-условия, с последующим заголовком с соответствующим адресом, где младший бит установлен в единицу (**11110xx1**).

The TRA bit indicates whether the slave is in Receiver or Transmitter mode.
Бит **TRA** показывает, в каком режиме находится "Ведомый", "передача" или "прием".

Slave transmitter (Ведомый передатчик)

Following the address reception and after clearing ADDR, the slave sends bytes from the DR register to the SDA line via the internal shift register.
После получения адреса и очистки бита **ADDR**, "Ведомый" посылает байты из регистра **DR** на линию **SDA** с помощью сдвигового регистра.

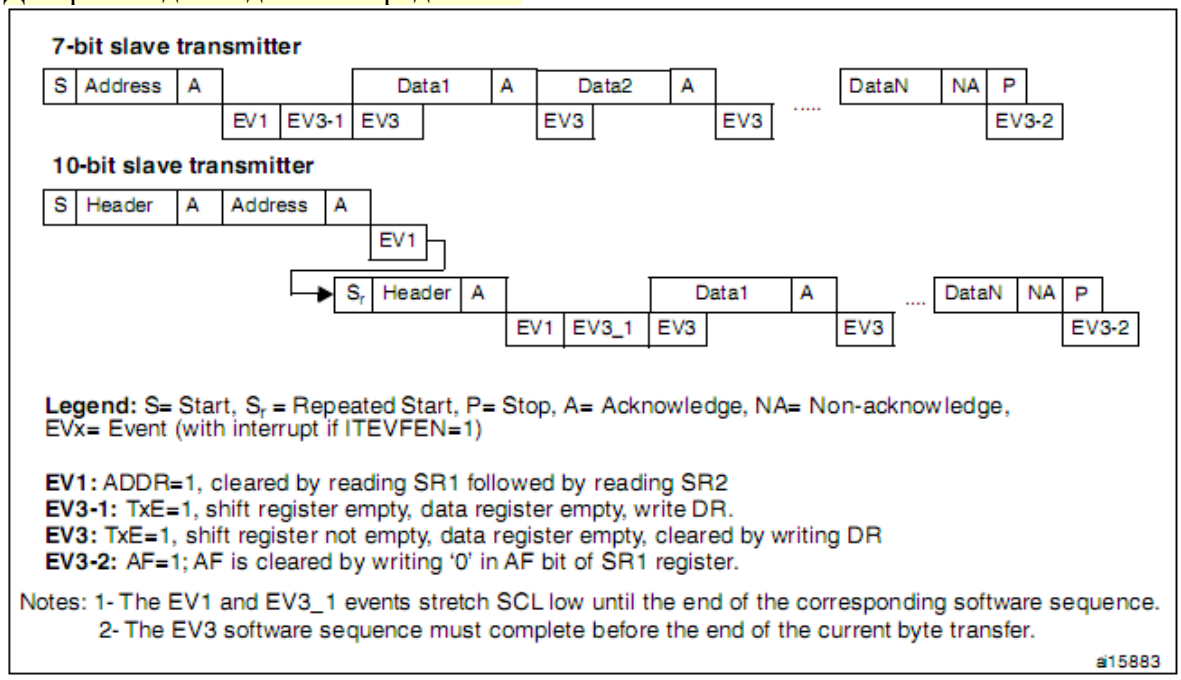
The slave stretches SCL low until ADDR is cleared and DR filled with the data to be sent (see *Figure 233* Transfer sequencing EV1 EV3).
"Ведомый" растягивает низкий уровень на линии **SCL** до тех пор, пока не будет очищен **ADDR** и **DR**, заполненный данными, не начнет передачу (см. состояния **EV1 EV3** на рис 233).

When the acknowledge pulse is received: (Когда получен бит уведомления:)

- The TxE bit is set by hardware with an interrupt if the ITEVFEN and the ITBUFEN bits are set. Бит прерывания **TxE** ставится аппаратно, если стоят биты **ITEVFEN** и **ITBUFEN**.

If TxE is set and some data were not written in the I2C_DR register before the end of the next data transmission, the BTF bit is set and the interface waits until BTF is cleared by a read to I2C_SR1 followed by a write to the I2C_DR register, stretching SCL low. Если бит **TxE** поставлен и некие данные еще не были записаны в регистр **I2C_DR** до окончания передачи текущих данных, ставится бит **BTF** и интерфейс ждет, пока бит **BTF** не будет очищен чтением **I2C_SR1**, с последующей записью в регистр **I2C_DR**, что затягивает низкий уровень на **SCL**.

Figure 233. Transfer sequence diagram for slave transmitter
Диаграмма для ведомого передатчика



Slave receiver (Ведомый приемник)

Following the address reception and after clearing ADDR, the slave receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence: После получения адреса и очистки бита **ADDR**, "Ведомый" принимает байты с линии **SDA** в регистр **DR** с помощью сдвигового регистра. После каждого байта интерфейс генерирует следующую последовательность:

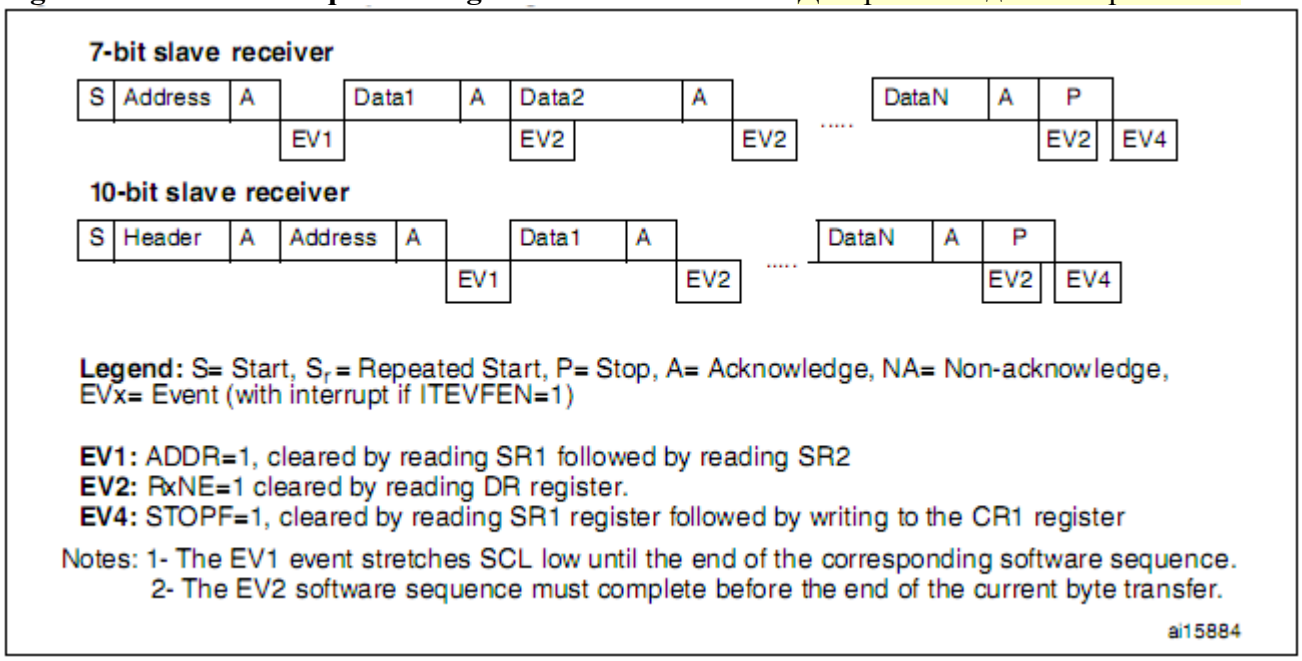
- An acknowledge pulse if the ACK bit is set

Если есть бит **ACK**, генерируется бит уведомления

- The RxNE bit is set by hardware and an interrupt is generated if the ITEVFEN and ITBUFEN bit is set. Бит **RxNE** ставится аппаратно и генерируется прерывание, если стоят биты **ITEVFEN** и **ITBUFEN**.

If RxNE is set and the data in the DR register is not read before the end of the next data reception, the BTF bit is set and the interface waits until BTF is cleared by a read from I2C_SR1 followed by a read from the I2C_DR register, stretching SCL low (see Figure 234 Transfer sequencing). Если бит **RxNE** поставлен и данные в регистре **DR** еще не прочитаны до окончания приема текущих данных, ставится бит **BTF** и интерфейс ждет, пока бит **BTF** не будет очищен чтением **I2C_SR1**, с последующим чтением регистра **I2C_DR**, что затягивает низкий уровень на **SCL** (см. рис. 234).

Figure 234. Transfer sequence diagram for slave receiver Диаграмма ведомого приемника



Closing slave communication (Закрытие обмена в режиме "Ведомый")

After the last data byte is transferred a Stop Condition is generated by the master. The interface detects this condition and sets, После того, как передан последний байт данных, "Ведущий" генерирует стоп-условие. Интерфейс детектирует это условие и ставит,

- The STOPF bit and generates an interrupt if the ITEVFEN bit is set.

Бит **STOPF** и генерируется прерывание, если стоит бит **ITEVFEN**.

Then the interface waits for a read of the SR1 register followed by a write to the CR1 register (see *Figure 234* Transfer sequencing EV4). Затем интерфейс ждет чтения регистра **SR1** с последующей записью в регистр **CR1** (см. этап **EV4** на рис. 234).

24.3.3 I2C master mode (I2C в режиме "Ведущий")

Start condition (Старт-условие)
Slave address transmission (Передача адреса ведомому устройству)
Master transmitter (Ведущий передатчик)
Closing the communication (Закрытие обмена)
Master receiver (Ведущий приемник)
Closing the communication (Закрытие обмена)

In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a Start condition and ends with a Stop condition. Master mode is selected as soon as the Start condition is generated on the bus with a START bit.
В режиме "Ведущий", интерфейс **I2C** инициирует обмен данными и генерирует тактовый сигнал. Передача последовательных данных всегда начинается со старт-условия и заканчивается стоп-условием. Интерфейс входит в режим "Ведущий", как только генерируется старт-условие на шине.

The following is the required sequence in master mode.
Следующая последовательность требуется для режима "Ведущий".

- Program the peripheral input clock in I2C_CR2 Register in order to generate correct timings Конфигурация входной частоты периферии в регистре **I2C_CR2**, так чтобы генерировать корректные тайминги
- Configure the clock control registers (Конфигурация регистра тактового сигнала **CCR**)
- Configure the rise time register (Конфигурация регистра скважности)
- Program the I2C_CR1 register to enable the peripheral
Конфигурация регистра **I2C_CR1**, чтобы разрешить периферию
- Set the START bit in the I2C_CR1 register to generate a Start condition
Установка бита **START** в регистре **I2C_CR1**, чтобы сгенерировать старт-условие

The peripheral input clock frequency must be at least:
Входная частота периферии должна быть не менее:
– 2 MHz in Standard mode (2 мГц в стандартном режиме)
– 4 MHz in Fast mode (4 мГц в быстром режиме)

Start condition (Старт-условие)

Setting the START bit causes the interface to generate a Start condition and to switch to Master mode (M/SL bit set) when the BUSY bit is cleared.
Установка бита **START** вызывает генерацию старт-условия и переключает интерфейс в режим "Ведущий" (ставится бит **M/SL**) когда очищен бит **BUSY**.

***Note**: In master mode, setting the START bit causes the interface to generate a ReStart condition at the end of the current byte transfer. В режиме "Ведущий", становка бита **START** в конце передачи текущего байта вызывает генерацию повторного старт-условия.*

Once the Start condition is sent: (Как только послано старт-условие:)

- The SB bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.
Бит **SB** ставится аппаратно и генерируется прерывание, если стоит бит **ITEVFEN**.

Then the master waits for a read of the SR1 register followed by a write in the DR register with the Slave address (see *Figure 235* & *Figure 236* Transfer sequencing EV5).

Затем "Ведущий" ждет чтения регистра **SR1** с последующей записью в регистр **DR** адреса ведомого устройства (см. этап **EV5** на рис. 235 и 236).

Slave address transmission (Передача адреса ведомому устройству)

Then the slave address is sent to the SDA line via the internal shift register.
Затем посылается адрес ведомого устройства на линию **SDA** с помощью сдвигового регистра.

- in 10-bit addressing mode, sending the header sequence causes the following event:

В режиме 10-ти битного адреса, посылка заголовка вызывает следующие события:

– The ADD10 bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set. Аппаратно ставится бит **ADD10** и генерируется прерывание, если стоит бит **ITEVFEN**.

Then the master waits for a read of the SR1 register followed by a write in the DR register with the second address byte (see *Figure 235* & *Figure 236* Transfer sequencing).
Затем "Ведущий" ждет чтения регистра **SR1** с последующей записью в регистр **DR** второго байта адреса (см. рис. 235 и 236).

– The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set. Аппаратно ставится бит **ADDR** и генерируется прерывание, если стоит бит **ITEVFEN**.

Then the master waits for a read of the SR1 register followed by a read of the SR2 register (see *Figure 235* & *Figure 236* Transfer sequencing). Затем "Ведущий" ждет чтения регистра **SR1** с последующим чтением регистра **SR2** (см. рис. 235 и 236).

- In 7-bit addressing mode, one address byte is sent.

В режиме 7-ми битного адреса посылается один байт адреса.
As soon as the address byte is sent, (Как только адресный байт послан,)

– The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set. Аппаратно ставится бит **ADDR** и генерируется прерывание, если стоит бит **ITEVFEN**.

Then the master waits for a read of the SR1 register followed by a read of the SR2 register (see *Figure 235* & *Figure 236* Transfer sequencing). Затем "Ведущий" ждет чтения регистра **SR1** с последующим чтением регистра **SR2** (см. рис. 235 и 236).

The master can decide to enter Transmitter or Receiver mode depending on the LSB of the slave address sent. "Ведущий" может решить перейти в режим передачи или приема в зависимости от младшего бита адресной посылки.

- In 7-bit addressing mode, (В режиме 7-ми битного адреса,)

– To enter Transmitter mode, a master sends the slave address with LSB reset.
Чтобы войти в режим передачи, "Ведущий" посылает адрес со сброшенным младшим битом.

– To enter Receiver mode, a master sends the slave address with LSB set. Чтобы войти в режим приема, "Ведущий" посылает адрес с установленным младшим битом.

- In 10-bit addressing mode, (В режиме 10-ти битного адреса,)

– To enter Transmitter mode, a master sends the header (11110xx0) and then the slave address with LSB reset, (where xx denotes the two most significant bits of the address).
Чтобы войти в режим передачи, "Ведущий" посылает заголовок (**11110xx0**), а затем адрес ведомого со сброшенным младшим битом, (где **xx** означает два старших бита адреса).

-- To enter Receiver mode, a master sends the header (11110xx0) and then the slave address with LSB reset. Then it should send a repeated Start condition followed by the header (11110xx1), (where xx denotes the two most significant bits of the address).
Чтобы войти в режим приема, "Ведущий" посылает заголовок (**11110xx0**), а затем адрес ведомого со сброшенным младшим битом. Затем он должен послать повторное старт-условие с последующим заголовком (**11110xx1**), (где **xx** означает два старших бита адреса).

- The TRA bit indicates whether the master is in Receiver or Transmitter mode.

Бит **TRA** показывает в каком режиме находится "Ведущий", "передача" или "прием".

Master transmitter (Ведущий передатчик)

Following the address transmission and after clearing ADDR, the master sends bytes from the DR register to the SDA line via the internal shift register. После передачи адреса и очистки бита **ADDR**, "Ведущий" посылает байты из регистра **DR** на линию **SDA** с помощью сдвигового регистра.

The master waits until the first data byte is written into I2C_DR (see Figure 235 Transfer sequencing EV8_1). Затем "Ведущий" ждет, пока первый байт данных не будет записан в **I2C_DR** (см. этап **EV8_1** на рис. 235).

When the acknowledge pulse is received: (Когда принят бит уведомления:)

- The TxE bit is set by hardware and an interrupt is generated if the ITEVFEN and ITBUFEN bits are set. Аппаратно ставится бит **TxE** и генерируется прерывание, если стят биты **ITEVFEN** и **ITBUFEN**.

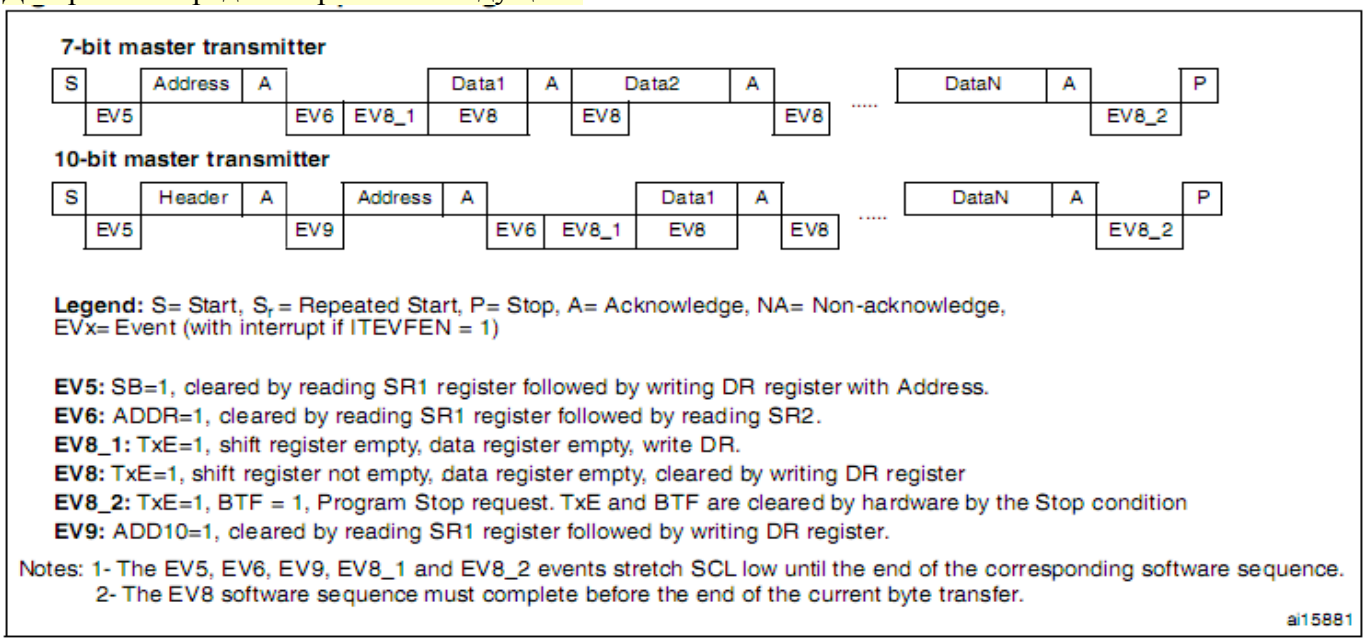
If TxE is set and a data byte was not written in the DR register before the end of the last data transmission, BTF is set and the interface waits until BTF is cleared by a read from I2C_SR1 followed by a write to I2C_DR, stretching SCL low.
Если бит **TxE** поставлен и байт данных не был записан в регистр **DR** до окончания передачи текущих данных, ставится бит **BTF** и интерфейс ждет, пока бит **BTF** не будет очищен чтением **I2C_SR1**, с последующей записью в регистр **I2C_DR**, что затягивает низкий уровень на **SCL**.

Closing the communication (Закрытие обмена)

After writing the last byte to the DR register, the STOP bit is set by software to generate a Stop condition (see *Figure 235* Transfer sequencing EV8_2). The interface goes automatically back to slave mode (M/SL bit cleared).
После записи последнего байта в регистр **DR**, бит **STOP** ставится программно, чтобы сгенерировать стоп-условие (см. этап **EV8_2** на рис.235). Интерфейс автоматически переходит в режим "Ведомый" (бит **M/SL** очищается).

***Note**: Stop condition should be programmed during EV8_2 event, when either TxE or BTF is set.*
*Стоп-условие должно быть установлено программно в течении этапа **EV8_2**, когда ставится бит **TxE** или **BTF**.*

Figure 235. Transfer sequence diagram for master transmitter
Диаграмма передачи в режиме "Ведущий"



Master receiver (Ведущий приемник)

Following the address transmission and after clearing ADDR, the I2C interface enters Master Receiver mode. In this mode the interface receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:
После передачи адреса и очистки бита **ADDR**, **I2C** интерфейс входит в режим "Ведущий" передатчик. В этом режиме "Ведущий" получает байты с линии **SDA** в регистр **DR** с помощью сдвигового регистра. После каждого байта интерфейс генерирует следующую последовательность:

- An acknowledge pulse if the ACK bit is set
Если есть бит **ACK**, генерируется бит уведомления
- The RxNE bit is set and an interrupt is generated if the ITEVFEN and ITBUFEN bits are set (see *Figure 236* Transfer sequencing EV7). Ставится бит **RxNE** и генерируется прерывание, если стят биты **ITEVFEN** и **ITBUFEN** (см. этап **EV7** на рис. 236).

If the RxNE bit is set and the data in the DR register is not read before the end of the last data reception, the BTF bit is set by hardware and the interface waits until BTF is cleared by a read in the SR1 register followed by a read in the DR register, stretching SCL low.
Если бит **RxNE** поставлен и данные в регистре **DR** еще не прочитаны до окончания приема текущих данных, ставится бит **BTF** и интерфейс ждет, пока бит **BTF** не будет очищен чтением **SR1**, с последующим чтением регистра **DR**, что затягивает низкий уровень на **SCL**.

Closing the communication (Закрытие обмена)

The master sends a NACK for the last byte received from the slave. After receiving this NACK, the slave releases the control of the SCL and SDA lines. Then the master can send a Stop/Re-Start condition. "Ведущий" посылает **NACK** по получении последнего байта от "Ведомого". После приема этого **NACK**, "Ведомый" отпускает линии **SCL** и **SDA**. Затем "Ведущий" может послать стоп-условие или повторное старт-условие.

- In order to generate the non-acknowledge pulse after the last received data byte, the ACK bit must be cleared just after reading the second last data byte (after second last RxNE event).

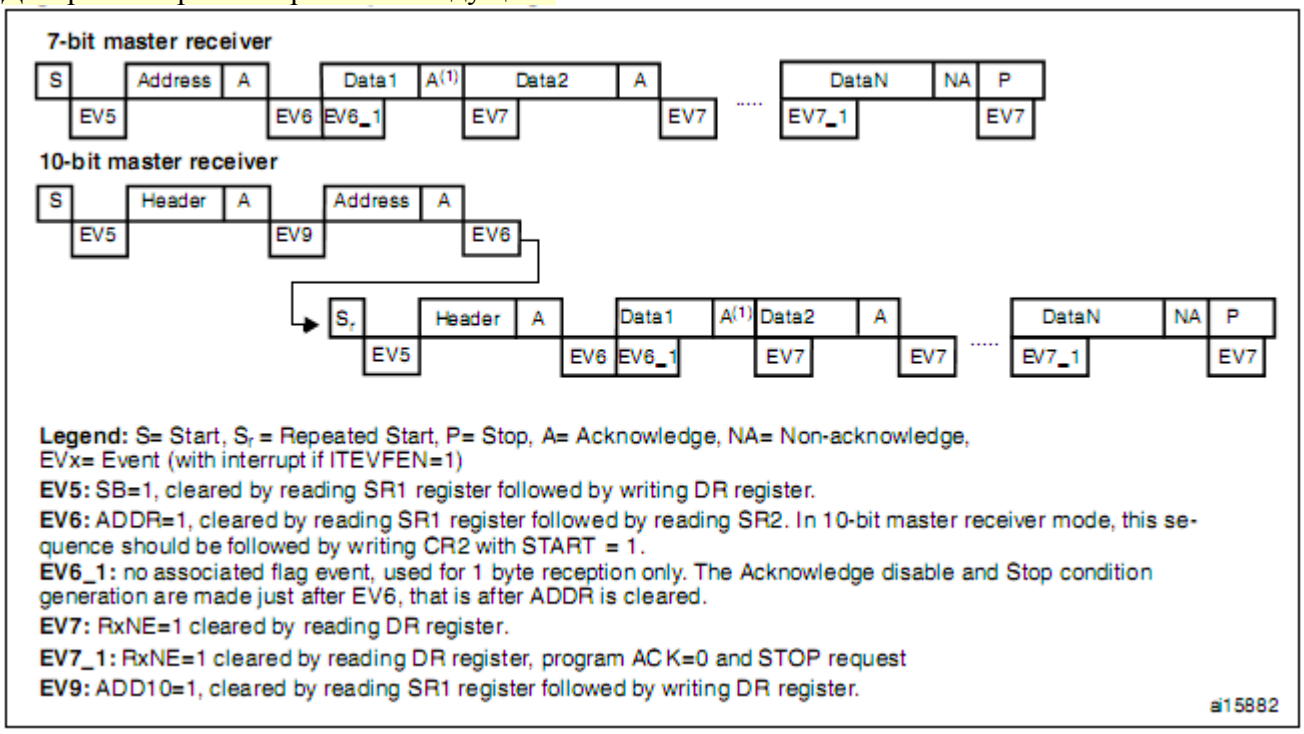
Чтобы не генерировать бит уведомления после приема последнего байта данных, бит **ACK** должен быть очищен сразу после чтения предпоследнего байта данных (после предпоследнего события **RxNE**).

- In order to generate the Stop/Re-Start condition, software must set the STOP/START bit just after reading the second last data byte (after the second last RxNE event).
Чтобы генерировать стоп-условие или повторное старт-условие, программа должна поставить бит **STOP** или **START** сразу после чтения предпоследнего байта данных (после предпоследнего события **RxNE**).

- In case a single byte is to be received, the Acknowledge disable and the Stop condition generation are made just after EV6 (in EV6_1, just after ADDR is cleared).
В случае, когда надо принять одиночный байт, отключение увеломления и генерация стоп-условия выполняются сразу после этапа **EV6** (на этапеп **EV6_1**, сразу после очистки бита **ADDR**).

After the Stop condition generation, the interface goes automatically back to slave mode (M/SL bit cleared). После генерации стоп-условия , интерфейс автоматически возвращается в режим "Ведомый" (бит **M/SL** очищается).

Figure 236. Transfer sequence diagram for master receiver
Диаграмма приема в режиме "Ведущий"



1. If a single byte is received, it is NA. Если принимается одиночный байт, это равно **NA**.

2. The EV5, EV6 and EV9 events stretch SCL low until the end of the corresponding software sequence.
События **EV5**, **EV6** и **EV9** удерживают **SCL** на низком уровне, чтобы завершить соответствующие программные операции.

3. The EV7 software sequence must complete before the end of the current byte transfer.
Программный этап **EV7** должен быть завершен до окончания передачи текущего байта.

4. The EV6_1 or EV7_1 software sequence must complete before the ACK pulse of the current byte transfer. Программный этап **EV6_1** или **EV7_1** должен быть завершен до бита уведомления **ACK** при передаче текущего байта.

24.3.4 Error conditions (Условия для ошибки)

The following are the error conditions which may cause communication to fail.
Следующие условия могут вызвать ошибку обмена.

Bus error (BERR) (Ошибка шины)
Acknowledge failure (AF) (Ошибка уведомления)
Arbitration lost (ARLO) (Потеря прав на шину)
Overrun/underrun error (OVR) (Ошибка переполнения/недостачи)

Bus error (BERR) (Ошибка шины)

This error occurs when the I2C interface detects a Stop or a Start condition during a byte transfer. In this case, Эта ошибка случается, когда **I2C** интерфейс детектирует стоп- или старт-условие в процессе передачи байта. В этом случае:

- The BERR bit is set and an interrupt is generated if the ITERREN bit is set
Ставится бит **BERR** и генерируется прерывание, если стоит бит **ITERREN**
- In case of Slave: data is discarded and the lines are released by hardware:
В случае режима "Ведомый": данные отвергаются и линии отпускаются аппаратно:
  - in case of misplaced start, the slave considers it is a restart and waits for address, or stop condition. В случае неуместного старт-условия, "Ведомый" рассматривает это, как рестарт и ждет адрес или стоп-условие.
  - in case of misplaced stop, the slave reacts like for a stop condition and the lines are released by hardware. В случае неуместного стоп-условия, "Ведомый" реагирует как на стоп-условие и линии отпускаются аппаратно.

Acknowledge failure (AF) (Ошибка уведомления)

This error occurs when the interface detects a non-acknowledge bit. In this case, Эта ошибка случается, когда интерфейс детектирует **NACK** (нет бита уведомления). В этом случае:

- The AF bit is set and an interrupt is generated if the ITERREN bit is set
Ставится бит **AF** и генерируется прерывание, если стоит бит **ITERREN**
- A transmitter which receives a NACK must reset the communication:
Передатчик, который получает **NACK**, должен сбросить обмен:
  - If Slave: lines are released by hardware
В режиме "Ведомый": линии отпускаются аппаратно
  - If Master: a Stop condition must be generated by software
В режиме "Ведущий": должно быть сгенерировано стоп-условие, программно

Arbitration lost (ARLO) (Потеря прав на шину)

This error occurs when the I2C interface detects an arbitration lost condition. In this case, Эта ошибка случается, когда интерфейс детектирует потерю прав на шину. В этом случае:

- the ARLO bit is set by hardware (and an interrupt is generated if the ITERREN bit is set)
Ставится бит **ARLO** и генерируется прерывание, если стоит бит **ITERREN**

- the I2C Interface goes automatically back to slave mode (the M/SL bit is cleared). When the I2C loses the arbitration, it is not able to acknowledge its slave address in the same transfer, but it can acknowledge it after a repeated Start from the winning master.
Интерфейс автоматически возвращается в режим "Ведомый" (бит **M/SL** очищается). Когда ведущее устройство **I2C** теряет права на шину, оно неспособно уведомить об этом "Ведомый" в этом же сеансе передачи, но оно может уведомить его после успешного повторного старта.

- lines are released by hardware (Линии отпускаются аппаратно)

Overrun/underrun error (OVR) (Ошибка переполнения/недостачи)

An overrun error can occur in slave mode when clock stretching is disabled and the I 2C interface is receiving data. The interface has received a byte (RxNE=1) and the data in DR has not been read, before the next byte is received by the interface. In this case,
Ошибка переполнения может случиться в режиме "Ведомый", когда отключена возможность растяжения такта, а **I2C** интерфейс принимает данные. Интерфейс принимает байт (**RxNE=1**) но данные в **DR** не были прочитаны до того, как начался прием следующего байта. В этом случае:

- The last received byte is lost. (Последний принятый байт теряется.)

- In case of Overrun error, software should clear the RxNE bit and the transmitter should re-transmit the last received byte.
В случае ошибки переполнения, программа должна очистить бит **RxNE** и передатчик должен повторно передать последний байт.

Underrun error can occur in slave mode when clock stretching is disabled and the I 2C interface is transmitting data. The interface has not updated the DR with the next byte (TxE=1), before the clock comes for the next byte. In this case, Ошибка недостачи может случиться в режиме "Ведомый", когда отключена возможность растяжения такта, а **I2C** интерфейс передает данные. Интерфейс не может обновить содержимое **DR** следующим байтом (**TxE=1**), до того, как придет тактовый для следующего байта. В этом случае:

- The same byte in the DR register will be sent again
Байт в регистре **DR** будет послан повторно

- The user should make sure that data received on the receiver side during an underrun error are discarded and that the next bytes are written within the clock low time specified in the I2C bus standard. Программа должна убедиться, что прием на стороне приемника, в процессе ошибки недостачи, отвергнут и что следующий байт записан в течении низкого уровня тактового, как определяет стандарт на **I2C**.

For the first byte to be transmitted, the DR must be written after ADDR is cleared and before the first SCL rising edge. If not possible, the receiver must discard the first data.
Для первого байта, который должен быть передан, регистр **DR** должен быть записан после очистки бита **ADDR** и до того, как появится первый фронт на линии **SCL**. Если это невозможно, приемник должен отвергнуть бервый байт.

24.3.5 SDA/SCL line control (Управление линиями SDA/SCL)

- If clock stretching is enabled: (Если разрешена возможность растяжения такта:)

    - Transmitter mode: If TxE=1 and BTF=1: the interface holds the clock line low before transmission to wait for the microcontroller to read SR1 and then write the byte in the Data Register (both buffer and shift register are empty).
    В режиме "передача": если бит **TxE=1** и бит **BTF=1**: интерфейс удерживает тактовую линию в низком состоянии, пока передатчик ожидает, что микроконтроллер прочтет **SR1**, а затем запишет байт в регистр **DR** (как буфер, так и сдвиговый регистр пустые).

    - Receiver mode: If RxNE=1 and BTF=1: the interface holds the clock line low after reception to wait for the microcontroller to read SR1 and then read the byte in the Data Register (both buffer and shift register are full).
    В режиме "прием": если бит **RxNE=1** и бит **BTF=1**: интерфейс удерживает тактовую линию в низком состоянии после приема, чтобы подождать, пока микроконтроллер прочтет **SR1**, а затем прочтет байт в регистре **DR** (как буфер, так и сдвиговый регистр пустые).

- If clock stretching is disabled in Slave mode:
Если в режиме "Ведомый" отключена возможность растяжения такта:

    - Overrun Error in case of RxNE=1 and no read of DR has been done before the next byte is received. The last received byte is lost.
    В случае **RxNE=1** будет ошибка переполнения и никакого чтения **DR** не будет до приема следующего байта. Последний принятый байт теряется.

    - Underrun Error in case TxE=1 and no write into DR has been done before the next byte must be transmitted. The same byte will be sent again.
    В случае **TxE=1** будет ошибка недостачи и никакой записи в **DR** не будет до передачи следующего байта. Один байт будет передан повторно.

    - Write Collision not managed. (Коллизии записи не разруливаются.)

24.3.6 SMBus

Introduction (Введение)
Similarities between SMBus and I2C (Сходство между **SMBus** и **I2C**)
Differences between SMBus and I2C (Различия между **SMBus** и **I2C**)
SMBus application usage (Использование **SMBus** приложением)
Device identification (Идентификация устройства)
Bus protocols (Шинный протокол)
Address resolution protocol (ARP) (Протокол динамического определения адресов **ARP**)
Unique device identifier (UDID) (Уникальный идентификатор устройства)
SMBus alert mode (Режим тревоги **SMBus**)
Timeout error (Ошибка превышения времени)
How to use the interface in SMBus mode (Как использовать интерфейс в режиме **SMBus**)

(*Я этим интерфейсом никогда не пользовался, поэтому, при переводе, мог переврать!!!*)

Introduction (Введение)

The System Management Bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I2C principles of operation. SMBus provides a control bus for system and power management related tasks. A system may use SMBus to pass messages to and from devices instead of toggling individual control lines.
SMBus (*System Management Bus*) - это двухпроводный интерфейс, посредством которого различные устройства могут обмениваться информацией друг с другом и с основной системой. Его работа основана на тех же принципах, что и **I2C**. **SMBus** предоставляет управление шиной системе, а управление потреблением соответствующим задачам. Система может использовать общую шину **SMBus** для передачи сообщений устройствам и от них, вместо использования индивидуальных соединений.

The System Management Bus Specification refers to three types of devices. A slave is a device that is receiving or responding to a command. A master is a device that issues commands, generates the clocks, and terminates the transfer. A host is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system. Спецификация **SMBus** определяет три типа устройств. "Ведомый" - это устройство, которое принимает или откликается на команду. "Ведущий" - это устройство, которое выдает команды, генерирует тактовые сигналы, и прекращает обмен. Хост - это специализированное "Ведущий", которое обеспечивает основной интерфейс с ЦПУ системы. Хост должен быть **Ведущий-Ведомый** и должен поддерживать **SMBus** протокол для Хоста. Только один Хост может быть в системе.

Similarities between SMBus and I2C (Сходство между SMBus и I2C)

- 2 wire bus protocol (1 Clk, 1 Data) + SMBus Alert line optional

2-х проводный протокол (1 для тактового, 1 для данных) + опциональная линия тревоги **SMBus Alert**

- Master-slave communication, Master provides clock

Обмен в стиле Ведущий-Ведомый, где "Ведущий" вырабатывает тактовый сигнал

- Multi master capability (Допускает несколько ведущих)
- SMBus data format similar to I2C 7-bit addressing format (*Figure 231*).

Формат данных **SMBus** похож на формат 7-ми битного адреса **I2C** (Рис. 231).

Differences between SMBus and I2C (Различия между SMBus и I2C)

The following table describes the differences between SMBus and I2C.
Следующая таблица описывает различия между **SMBus** и **I2C**.

Table 170. SMBus vs. I2C (SMBus против I2C)

SMBus	I2C
Max. speed 100 kHz Максимальная скорость 100 кГц	Max. speed 400 kHz Максимальная скорость 400 кГц
Min. clock speed 10 kHz Ограничение минимальной скорости в 10 кГц	No minimum clock speed Нет ограничения минимальной скорости
35 ms clock low timeout Ограничение времени низкого состояния 35 мс	No timeout Нет ограничения времени низкого состояния
Logic levels are fixed Логич. уровни фиксированы	Logic levels are V_{DD} dependent Логич. уровни зависят от $\mathbf{V_{DD}}$
Different address types (reserved, dynamic etc.) Различные типы адресов (**reserved**, **dynamic** и.т.д.)	7-bit, 10-bit and general call slave address types Типы **slave** адресов: 7/10 бит и общий вызов
Different bus protocols (quick command, process call etc.) Различные шинные протоколы (**quick command**, **process call** и.т.д.)	No bus protocols Нет шинного протокола

SMBus application usage (Использование SMBus приложением)

With System Management Bus, a device can provide manufacturer information, tell the system what its model/part number is, save its state for a suspend event, report different types of errors, accept control parameters, and return its status. SMBus provides a control bus for system and power management related tasks.
С помощью **SMBus** устройство может предоставить информацию изготовителя, сообщить системе, каков номер и субномер ее модели, сохранить свое состояние на случай приостановки, сообщить о различных типах ошибок, принять управляющие параметры, и возвратить их состояние. **SMBus** обеспечивает шину управления для задач, связанных с управлением системой и потреблением питания.

Device identification (Идентификация устройства)

Any device that exists on the System Management Bus as a slave has a unique address called the Slave Address. For the list of reserved slave addresses, refer to the SMBus specification ver. 2.0 (http://smbus.org/specs/).
У любого устройства, которое подключено к **SMBus** как "Ведомый", есть уникальный адрес, называемый **Slave** адресом. Чтобы ознакомиться со списком зарезервированных **Slave** адресов, обратитесь к спецификации **SMBus** версии **2.0** (**http://smbus.org/specs/**).

Bus protocols (Шинный протокол)

The SMBus specification supports up to 9 bus protocols. For more details of these protocols and SMBus address types, refer to SMBus specification ver. 2.0 (http://smbus.org/specs/). These protocols should be implemented by the user software.
Спецификация **SMBus** поддерживает до 9 шинных протоколов. За дополнительной информацией об этих протоколах и типах адресов в **SMBus**, обратитесь к спецификации **SMBus 2.0** (**http://smbus.org/specs/**). Эти протоколы должны быть встроены в пользовательские программны.

Address resolution protocol (ARP)
Протокол динамического определения адресов ARP

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. The Address Resolution Protocol (ARP) has the following attributes:
Конфликты адресов ведомых **SMBus** устройств могут быть разрешены динамически, с помощью назначения нового уникального адреса для каждого ведомого устройства. Протокол определения адресов **ARP** (*Address Resolution Protocol*) имеет следующие атрибуты:

- Address assignment uses the standard SMBus physical layer arbitration mechanism
Присвоение адресов использует стандартный **SMBus** механизм арбитража на физическом уровне

- Assigned addresses remain constant while device power is applied; address retention through device power loss is also allowed
Назначенные адреса остаются постоянными, пока подано питание на устройства; запоминание адресов на случай потери питания также возможно

- No additional SMBus packet overhead is incurred after address assignment. (i.e. subsequent accesses to assigned slave addresses have the same overhead as accesses to fixed address devices.)
После присвоения адреса никакого дополнительного заголовка в пакете **SMBus** не требуется. (То есть, последующий доступ к ведомым устройствам с назначенным адресом точно такой же, как доступ к устройствам с фиксированным адресом).

- Any SMBus master can enumerate the bus
Любое ведущее **SMBus** устройство может провести операцию назначения адресов на шине (*enumeration*)

Unique device identifier (UDID) (Уникальный идентификатор устройства)

In order to provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique device identifier (UDID).
Чтобы обеспечить механизм изоляции каждого устройства, с целью присваивания адреса, каждое устройство должно иметь свой уникальный идентификатор (**UDID**).

For the details on 128 bit UDID and more information on ARP, refer to SMBus specification ver. 2.0 (http://smbus.org/specs/). За дополнительной информацией о 128-ми битном **UDID** и о протоколе **ARP**, обратитесь к спецификации **SMBus 2.0** (**http://smbus.org/specs/**).

SMBus alert mode (Режим тревоги SMBus)

SMBus Alert is an optional signal with an interrupt line for devices that want to trade their ability to master for a pin. SMBALERT is a wired-AND signal just as the SCL and SDA signals are. SMBALERT is used in conjunction with the SMBus General Call Address. Messages invoked with the SMBus are 2 bytes long. Тревога **SMBus** - это опционная дополнительная сигнальная цепь прерывания для тех устройств, которые хотят "поторговаться" за право владеть шиной. **SMBALERT** - это цепь с открытым коллектором, такая же, как цепи **SCL** и **SDA**. Сигнал **SMBALERT** используется совместно с Общим Адресом Запроса в **SMBus**. **SMBus** сообщения, инициированные этим сигналом, имеют длину 2 байта.

A slave-only device can signal the host through SMBALERT that it wants to talk by setting ALERT bit in I2C_CR1 register. The host processes the interrupt and simultaneously accesses all SMBALERT devices through the Alert Response Address (known as ARA having a value 0001 100X). Only the device(s) which pulled SMBALERT low will acknowledge the Alert Response Address. This status is identified using SMBALERT Status flag in I2C_SR1 register. The host performs a modified Receive Byte operation. The 7 bit device address provided by the slave transmit device is placed in the 7 most significant bits of the byte. The eighth bit can be a zero or one.
Только ведомое устройство может сигнализировать Хосту через **SMBALERT**, что оно хочет "поговорить" насчет установки бита **ALERT** в регистре **I2C_CR1**. Хост обрабатывает прерывание и, одновременно, обращается ко всем устройствам, имеющим цепь **SMBALERT**, с помощью Адреса Тревоги со значением **0001 100X**, известного как **ARA** (*Alert Response Address*). Только устройство(а), которое выставило низкий уровень на **SMBALERT**, ответит на этот адрес. Это состояние (*что "Ведущий" передает свои права на шину*) идентифицировано с помощью флага статуса **SMBALERT** в регистре **I2C_SR1**. Хост выполняет модифицированную операцию приема байта. 7-ми битный адрес устройства, который предоставляет ведомый передатчик, помещается в 7 старших битов байта. Восьмой бит может быть нулем или единицей.

If more than one device pulls SMBALERT low, the highest priority (lowest address) device will win communication rights via standard arbitration during the slave address transfer. After acknowledging the slave address the device must disengage its SMBALERT pull-down. If the host still sees SMBALERT low when the message transfer is complete, it knows to read the ARA again.
Если больше чем одно устройство выставит низкий уровень на **SMBALERT**, то, с помощью стандартного арбитража, во время передачи адреса с ведомых устройств, право быть "Ведущим" выиграет то устройство, у которого самый высокий приоритет (самый низкий адрес). После подтверждения адреса от ведомого устройства, оно должно отпустить линию **SMBALERT**. Если Хост все еще видит низкий уровень на **SMBALERT**, когда передача сообщения завершена, он будет читать адрес **ARA** снова.

A host which does not implement the SMBALERT signal may periodically access the ARA. Хост, у которого нет встроенного сигнала **SMBALERT**, может периодически обращаться к устройствам с помощью **ARA**.

For more details on SMBus Alert mode, refer to SMBus specification ver. 2.0 (http://smbus.org/specs/). За дополнительной информацией о режиме тревоги в **SMBus**, обратитесь к спецификации **SMBus 2.0** (**http://smbus.org/specs/**).

Timeout error (Ошибка превышения времени)

There are differences in the timing specifications between I2C and SMBus. SMBus defines a clock low timeout, TIMEOUT of 35 ms. Also SMBus specifies TLOW: SEXT as the cumulative clock low extend time for a slave device. SMBus specifies TLOW: MEXT as the cumulative clock low extend time for a master device. For more details on these timeouts, refer to SMBus specification ver. 2.0 (http://smbus.org/specs/). Есть различия в спецификации временных соотношений (таймингов) между **I2C** и **SMBus**. **SMBus** определяет максимальное время нахождения линии в низком

состоянии - **TIMEOUT** равное 35 мс. Также **SMBus** определяет параметр **TLOW:SEXT**, как совокупное время нахождения линии в низком состоянии (*при растяжке бита ?*) для ведомого устройства. Также **SMBus** определяет **TLOW:MEXT**, как совокупное время нахождения линии в низком состоянии (*при растяжке бита ?*) для "Ведущего". За дополнительной информацией о превышении времени, обратитесь к спецификации **SMBus 2.0** (**http://smbus.org/specs/**).

The status flag Timeout or Tlow Error in I2C_SR1 shows the status of this feature.
Флаг состояния **Timeout** (или, по другому, **Tlow Error**) в регистре **I2C_SR1** показывает состояние этой функций.

How to use the interface in SMBus mode (Как использовать интерфейс в режиме SMBus)

To switch from I2C mode to SMBus mode, the following sequence should be performed.
Чтобы переключиться из режима **I2C** в режим **SMBus**, надо выполнить следующую последовательность.

- Set the SMBus bit in the I2C_CR1 register
  Установите бит **SMBus** в регистре **I2C_CR1**
- Configure the SMBTYPE and ENARP bits in the I2C_CR1 register as required for the application Сконфигурируйте биты **SMBTYPE** и **ENARP** в регистре **I2C_CR1**, как это требуется для приложения

If you want to configure the device as a master, follow the Start condition generation procedure in *Section 24.3.3: I2C master mode*. Otherwise, follow the sequence in *Section 24.3.2: I2C slave mode*.
Если Вы хотите сконфигурировать устройство как "Ведущий", выполните процедуру стартового условия из раздела 24.3.3 "**I2C** в режиме Ведущий". Иначе, выполните процедуру из раздела 24.3.2 "**I2C** в режиме "Ведомый".

The application has to control the various SMBus protocols by software.
Приложение должно управлять различными протоколами **SMBus** программно.

- SMB Device Default Address acknowledged if ENARP=1 and SMBTYPE=0
  SMB устройство откликается на свой адрес по умолчанию, если **ENARP=1** и **SMBTYPE=0**
- SMB Host Header acknowledged if ENARP=1 and SMBTYPE=1
  SMB Хост откликается на заголовок, если **ENARP=1** и **SMBTYPE=1**
- SMB Alert Response Address acknowledged if SMBALERT=1
  SMB устройство откликается на адрес тревоги, если **SMBALERT=1**

24.3.7 DMA requests (Запрос на использование DMA)

Transmission using DMA (Передача с использованием **DMA**)
Reception using DMA (Прием с использованием **DMA**)

DMA requests (when enabled) are generated only for data transfer. DMA requests are generated by Data Register becoming empty in transmission and Data Register becoming full in reception. The DMA request must be served before the end of the current byte transfer. When the number of data transfers which has been programmed for the corresponding DMA channel is reached, the DMA controller sends an End of Transfer EOT signal to the I2C interface and generates a Transfer Complete interrupt if enabled:
Запрос на использование **DMA** (когда разрешено) генерируется только обменом данных. Этот запрос генерируется регистром данных, при его опустошении при передаче, и его заполнением при приеме. Запрос от **DMA** должен быть обслужен до завершения обменом текущего байта. Когда достигнуто число байт обмена, которое было запрограммировано на соответствующий **DMA** канал, то **DMA** контроллер посылает сигнал **EOT** (*End of Transfer*) на **I2C** интерфейс и генерирует прерывание **TC** (*Transfer Complete*), если оно разрешено:

- Master transmitter: In the interrupt routine after the EOT interrupt, disable DMA requests then wait for a BTF event before programming the Stop condition.
Ведущий передатчик: В подпрограмме обработчика прерывания, после прерывания **EOT**, отключите запрос от **DMA**, затем ждите событие **BTF**, прежде чем запрограммировать стоп-условие.

- Master receiver: when the number of bytes to be received is equal to or greater than two, the DMA controller sends a hardware signal, EOT_1, corresponding to the last but one data byte (number_of_bytes – 1). If, in the I2C_CR2 register, the LAST bit is set, I2C automatically sends a NACK after the next byte following EOT_1. The user can generate a Stop condition in the DMA Transfer Complete interrupt routine if enabled.
Ведущий приемник: когда число байт, которые надо принять, не менее двух, то **DMA** контроллер посылает аппаратный сигнал **EOT_1** при достижении предпоследнего байта (*number_of_bytes - 1*). Если в регистре **I2C_CR2** стоит бит **LAST**, то **I2C** автоматически посылает **NACK** после байта, следующего за **EOT_1**. Программа может генерировать стоп-условие в обработчике прерывания **TC** (*Transfer Complete*) от **DMA**, если оно разрешено.

Transmission using DMA (Передача с использованием DMA)

DMA mode can be enabled for transmission by setting the DMAEN bit in the I2C_CR2 register. Data will be loaded from a Memory area configured using the DMA peripheral (refer to the DMA specification) to the I2C_DR register whenever the TxE bit is set. To map a DMA channel for I2C transmission, perform the following sequence. Here x is the channel number.
Режим **DMA** может быть разрешен для передачи установкой бита **DMAEN** в регистре **I2C_CR2**. Данные будут загружаться из области памяти, сконфигурированной на использование с **DMA** периферией (см. описание **DMA**) в регистр **I2C_DR** всякий раз, когда ставится бит **TxE**. Чтобы отразить **DMA** канал на **I2C** передатчик, выполните следующее. Здесь **x** - это номер канала.

1. Set the I2C_DR register address in the DMA_CPARx register. The data will be moved to this address from the memory after each TxE event.
Задайте адрес регистра **I2C_DR** в регистре **DMA_CPARx**. Данные будут перемещаться по этому адресу из памяти после каждого события **TxE**.

2. Set the memory address in the DMA_CMARx register. The data will be loaded into I2C_DR from this memory after each TxE event.
Задайте адрес памяти в регистре **DMA_CMARx**. Данные будут загружаться в регистр **I2C_DR** из этой памяти после каждого события **TxE**.

3. Configure the total number of bytes to be transferred in the DMA_CNDTRx register. After each TxE event, this value will be decremented.
Задайте общее число байт, которое будет перемещено, в регистре **DMA_CNDTRx**. После каждого события **TxE**, это значение будет декрементировано.

4. Configure the channel priority using the PL[0:1] bits in the DMA_CCRx register
Задайте приоритет канала, используя биты **PL[0:1]** в регистре **DMA_CCRx**

5. Set the DIR bit and, in the DMA_CCRx register, configure interrupts after half transfer or full transfer depending on application requirements.
Установите бит **DIR**, и задайте прерывание в регистре **DMA_CCRx**, после половины обмена, или полного обмена, в зависимости от требований приложения.

6. Activate the channel by setting the EN bit in the DMA_CCRx register.
Активируйте канал установкой бита **EN** в регистре **DMA_CCRx**.

When the number of data transfers which has been programmed in the DMA Controller registers is reached, the DMA controller sends an End of Transfer EOT/ EOT_1 signal to the I2C interface and the DMA generates an interrupt, if enabled, on the DMA channel interrupt vector.
Когда достигнуто число байт обмена, которое было запрограммировано в регистре **DMA** контроллера, то он посылает сигнал **EOT/EOT_1** (*End of Transfer*) на **I2C** интерфейс и **DMA** генерирует прерывание, если оно разрешено, на векторе прерывания от **DMA** канала.

***Note**: Do not enable the ITBUFEN bit in the I2C_CR2 register if DMA is used for transmission.*
*Не разрешайте бит **ITBUFEN** в регистре **I2C_CR2**, если **DMA** используется для передачи.*

Reception using DMA (Прием с использованием DMA)

DMA mode can be enabled for reception by setting the DMAEN bit in the I2C_CR2 register. Data will be loaded from the I2C_DR register to a Memory area configured using the DMA peripheral (refer to the DMA specification) whenever a data byte is received. To map a DMA channel for I2C reception, perform the following sequence. Here x is the channel number.
Режим **DMA** может быть разрешен для приема установкой бита **DMAEN** в регистре **I2C_CR2**. Данные будут загружаться из регистра **I2C_DR** в область памяти, сконфигурированной на использование с **DMA** периферией (см. описание **DMA**) всякий раз, когда принимается байт. Чтобы отразить **DMA** канал на **I2C** приемник, выполните следующее. Здесь **x** - это номер канала.

1. Set the I2C_DR register address in DMA_CPARx register. The data will be moved from this address to the memory after each RxNE event.
Задайте адрес регистра **I2C_DR** в регистре **DMA_CPARx**. Данные будут перемещаться из этого адреса в память после каждого события **RxNE**.

2. Set the memory address in the DMA_CMARx register. The data will be loaded from the I2C_DR register to this memory area after each RxNE event.
Задайте адрес памяти в регистре **DMA_CMARx**. Данные будут загружаться из регистра **I2C_DR** в эту память после каждого события **RxNE**.

3. Configure the total number of bytes to be transferred in the DMA_CNDTRx register. After each RxNE event, this value will be decremented.
Задайте общее число байт, которое будет перемещено, в регистре **DMA_CNDTRx**. После каждого события **RxNE**, это значение будет декрементировано.

4. Configure the channel priority using the PL[0:1] bits in the DMA_CCRx register
Задайте приоритет канала, используя биты **PL[0:1]** в регистре **DMA_CCRx**

5. Reset the DIR bit and configure interrupts in the DMA_CCRx register after half transfer or full transfer depending on application requirements.

Сбросьте бит **DIR** и задайте прерывание в регистре **DMA_CCRx**, после половины обмена, или полного обмена, в зависимости от требований приложения.

6. Activate the channel by setting the EN bit in the DMA_CCRx register.
Активируйте канал установкой бита **EN** в регистре **DMA_CCRx**.

When the number of data transfers which has been programmed in the DMA Controller registers is reached, the DMA controller sends an End of Transfer EOT/ EOT_1 signal to the I2C interface and DMA generates an interrupt, if enabled, on the DMA channel interrupt vector.
Когда достигнуто число байт обмена, которое было запрограммировано в регистре **DMA** контроллера, то он посылает сигнал **EOT/EOT_1** (*End of Transfer*) на **I2C** интерфейс и **DMA** генерирует прерывание, если оно разрешено, на векторе прерывания от **DMA** канала.

***Note**: Do not enable the ITBUFEN bit in the I2C_CR2 register if DMA is used for reception.*
*Не разрешайте бит **ITBUFEN** в регистре **I2C_CR2**, если **DMA** используется для приема.*

24.3.8 Packet error checking (Проверка ошибок пакетирования)

A PEC calculator has been implemented to improve the reliability of communication. The PEC is calculated by using the C(x) = x8 + x2 + x + 1 CRC-8 polynomial serially on each bit.
В интерфейс встроен калькулятор **PEC**, чтобы улучшить надежность обмена. **PEC** расчитывается по формуле **C(x) = x8 + x2 + x + 1** (полином **CRC-8**) последовательно на каждом бите.

- PEC calculation is enabled by setting the ENPEC bit in the I2C_CR1 register. PEC is a CRC-8 calculated on all message bytes including addresses and R/W bits.
Расчет **PEC** разрешается установкой бита **ENPEC** в регистре **I2C_CR1**. **PEC** - это **CRC-8**, расчитанное для всех байт сообщения, включая адрес и **R/W** биты.

  - In transmission: set the PEC transfer bit in the I2C_CR1 register after the TxE event corresponding to the last byte. The PEC will be transferred after the last transmitted byte.
  При передаче: поставьте бит **PEC** в регистре **I2C_CR1** после события **TxE**, которое соответствует последнему байту. **PEC** будет передано после вслед за последним байтом.

  - In reception: set the PEC bit in the I2C_CR1 register after the RxNE event corresponding to the last byte so that the receiver sends a NACK if the next received byte is not equal to the internally calculated PEC. In case of Master-Receiver, a NACK must follow the PEC whatever the check result. PEC must be set before the ACK pulse of the current byte reception.
  При приеме: поставьте бит **PEC** в регистре **I2C_CR1** после события **RxNE**, которое соответствует последнему байту, тогда приемник пошлет **NACK**, если полученный байт не равен тому, что получен в результате внутреннего расчета **PEC**. В случае ведущего приемника, **NACK** должен следовать за **PEC**, какой бы ни был результат проверки. Бит **PEC** должен ставиться до этапа **ACK** при приеме текущего байта.

- A PECERR error flag/interrupt is also available in the I2C_SR1 register.
В регистре **I2C_SR1** также доступен флаг ошибки/прерывания **PECERR**.

- If DMA and PEC calculation are both enabled:-
Если есть разрешение и на **DMA**, и на расчет **PEC**, то:

  - In transmission: when the I2C interface receives an EOT signal from the DMA controller, it automatically sends a PEC after the last byte.
  При передаче: когда **I2C** интерфейс получает сигнал **EOT** от **DMA** контроллера, он

автоматически посылает **PEC** после последнего байта.

- In reception: when the I2C interface receives an EOT_1 signal from the DMA controller, it will automatically consider the next byte as a PEC and will check it. A DMA request is generated after PEC reception.
При приеме: когда **I2C** интерфейс получает сигнал **EOT_1** от **DMA** контроллера, он автоматически будет рассматривать следующий байт как **PEC**, и будет проверять его. Запрос на **DMA** генерируется после получения **PEC**.

- To allow intermediate PEC transfers, a control bit is available in the I2C_CR2 register (LAST bit) to determine if it is really the last DMA transfer or not. If it is the last DMA request for a master receiver, a NACK is automatically sent after the last received byte.
Чтобы разрешить передачу промежуточного **PEC**, доступен управляющий бит в регистре **I2C_CR2** (бит **LAST**), который определяет, это действительно последний обмен по **DMA**, или нет. Если это последний запрос на **DMA** для ведущего приемника, то автоматически посылается **NACK** после получения последнего байта.

- PEC calculation is corrupted by an arbitration loss.
Расчет **PEC** некорректен при потере прав на шину.

24.4 I2C interrupts (Прерывания от I2C)

The table below gives the list of I2C interrupt requests.
В таблице ниже даны запросы на прерывание от **I2C**.

Table 171. I2C Interrupt requests (Запросы на прерывание от **I2C**)

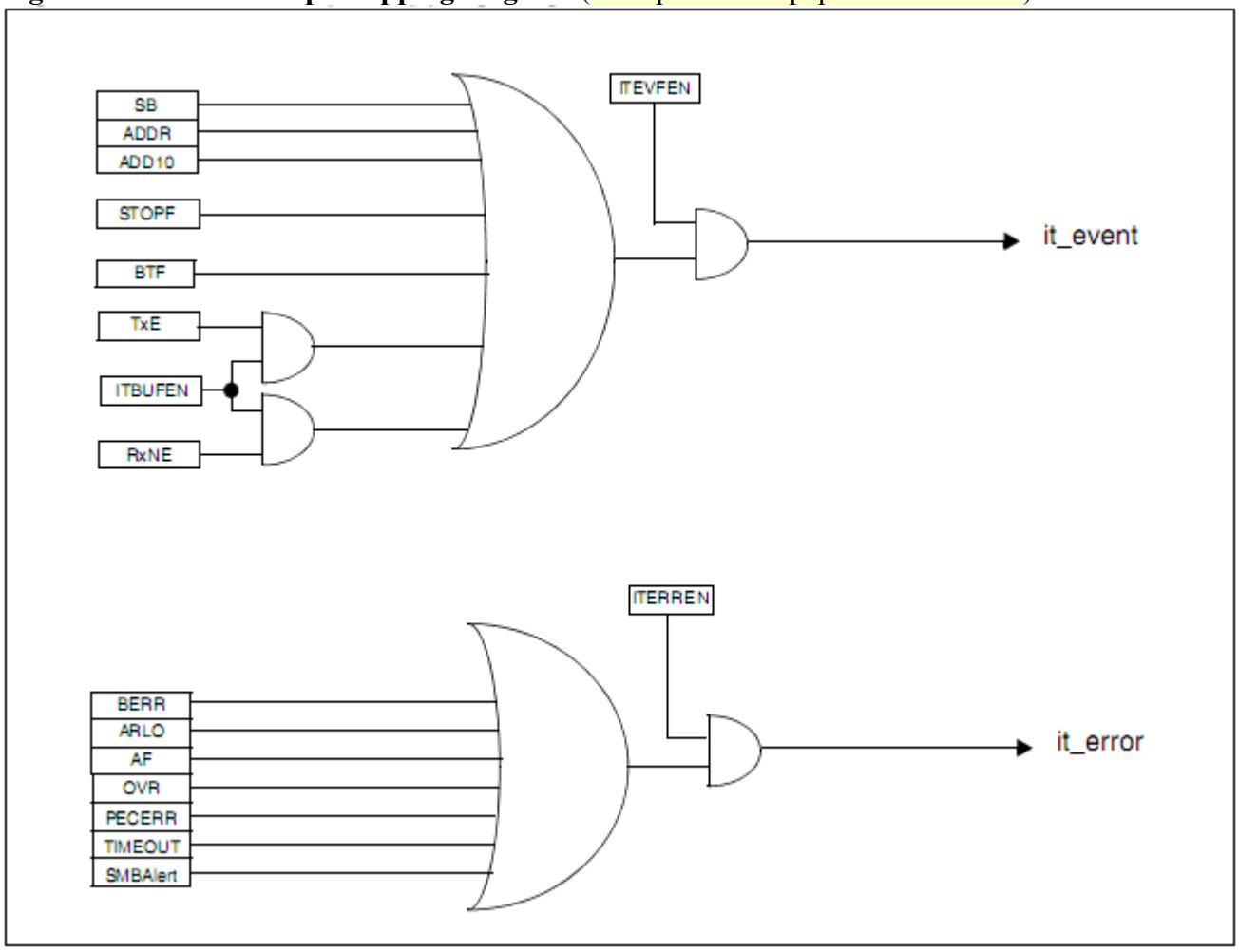
Interrupt event	Event flag	Enable Control bit
Start bit sent (Master) (Послано старт-условие)	SB	ITEVFEN
Address sent (Master) or Address matched (Slave) "Ведущий" послало адрес или "Ведомый" получило свой адрес	ADD	
10-bit header sent (Master) "Ведущий" послало заголовок (10 бит)	ADD10	
Stop received (Slave) ("Ведомый" получило стоп-условие)	STOPF	
Data byte transfer finished (Передача байта закончена)	BTF	
Receive buffer not empty (Буфер прима не пуст)	RxNE	ITEVFEN и ITBUFEN
Transmit buffer empty (Буфер передачи пуст)	TxE	
Bus error (Ошибка шины)	BERR	ITERREN
Arbitration loss (Master) (Потеря прав на шину)	ARLO	
Acknowledge failure (Ошибка уведомления)	AF	
Overrun/Underrun (Переполнение/недостача)	OVR	
PEC error (Ошибка пакета)	PECERR	
Timeout/Tlow error (Ошибка превышения времени)	TIMEOUT	
SMBus Alert (Тревога от **SMBus**)	SMBALERT	

***Note**:*

1. *SB, ADDR, ADD10, STOPF, BTF, RxNE and TxE are logically ORed on the same interrupt channel.*
   ***SB**, **ADDR**, **ADD10**, **STOPF**, **BTF**, **RxNE** и **TxE** логически объединены на один канал прерывания.*

2. *BERR, ARLO, AF, OVR, PECERR, TIMEOUT and SMBALERT are logically ORed on the same interrupt channel.*
   ***BERR**, **ARLO**, **AF**, **OVR**, **PECERR**, **TIMEOUT** и **SMBALERT** логически объединены на другой канал прерывания.*

Figure 237. I2C interrupt mapping diagram (Отображение прерываний от **I2C**)



24.5 I2C debug mode (Режим отладки и I2C)

When the microcontroller enters the debug mode (Cortex-M3 core halted), the SMBUS timeout either continues to work normally or stops, depending on the DBG_I2Cx_SMBUS_TIMEOUT configuration bits in the DBG module. For more details, refer to *Section 29.16.2: Debug support for timers, watchdog, bxCAN and I2C on page 972*.
Когда микроконтроллер входит в режим отладки (ядро **Cortex-M3** остановлено), контроль превышения времени **SMBUS** либо продолжает работать нормально, либо останавливается, в зависимости от конфигурации бита **DBG_I2Cx_SMBUS_TIMEOUT** в модуле **DBG**. Для дополнительной информации, см. раздел 29.16.2: "**Debug support for timers, watchdog, bxCAN and I2C**".

24.6 I2C registers (I2C регистры)

24.6.1 Control register 1 (I2C_CR1) (Управляющий регистр 1)
24.6.2 Control register 2 (I2C_CR2) (Управляющий регистр 2)
24.6.3 Own address register 1 (I2C_OAR1) Регистр собственного адреса 1
24.6.4 Own address register 2 (I2C_OAR2) Регистр собственного адреса 2
24.6.5 Data register (I2C_DR) Регистр данных
24.6.6 Status register 1 (I2C_SR1) Статусный регистр 1
24.6.7 Status register 2 (I2C_SR2) Статусный регистр 2
24.6.8 Clock control register (I2C_CCR) Регистр управления тактовым сигналом
24.6.9 TRISE register (I2C_TRISE) Регистр **TRISE**
24.6.10 I2C register map (Карта регистров **I2C**)

Refer to *Section 1.1 on page 37* for a list of abbreviations used in register descriptions.
См. раздел 1.1, где приведен перечень сокращений, используемых при описании регистров.

24.6.1 Control register 1 (I2C_CR1) (Управляющий регистр 1)

Address offset: 0x00
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW RST	Res.	ALERT	PEC	POS	ACK	STOP	START	NO STRETCH	ENGC	EN PEC	EN ARP	SMB TYPE	Res.	SM BUS	PE
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 15 SWRST: Software reset (Программный сброс)
When set, the I2C is under reset state. Before resetting this bit, make sure the I2C lines are released and the bus is free.
Когда установлен, то **I2C** находится в состоянии сброса. Перед сбросом этого бита убедитесь, что линии **I2C** отпущены и шина свободна.
`0:` I2C Peripheral not under reset (**I2C** периферия в нормальном состоянии)
`1:` I2C Peripheral under reset state (**I2C** периферия в состоянии сброса)
Note: This bit can be used in case the BUSY bit is set to '1' when no stop condition has been detected on the bus.
Этот бит можно использовать в случае, когда стоит бит **BUSY** и стоп-состояние на шине не обнаружено.

Bit 14 Reserved:
forced by hardware to 0. (Аппаратно ставится в 0.)

Bit 13 ALERT: SMBus alert (Тревога SMBus)
This bit is set and cleared by software, and cleared by hardware when PE=0.
Этот бит ставится и очищается программно, и очищается аппаратно, когда **PE=0**.
`0:` Releases SMBAlert pin high. Alert Response Address Header followed by NACK.
Вывод **SMBAlert** отпущен (высокое состояние). В ответ на заголовок Тревоги выставляется **NACK**.
`1:` Drives SMBAlert pin low. Alert Response Address Header followed by ACK.
Управляет выводом **SMBAlert** (низкое состояние). В ответ на заголовок Тревоги выставляется **ACK**.

Bit 12 PEC: Packet error checking (Функция проверки ошибки пакета)

This bit is set and cleared by software, and cleared by hardware when PEC is transferred or by a START or Stop condition or when PE=0.

Этот бит ставится и очищается программно, и очищается аппаратно, когда передается **PEC**, или старт- стоп-условие, или когда **PE=0**.

`0:` No PEC transfer (Байт **PEC** не передается)

`1:` PEC transfer (in Tx or Rx mode) (**PEC** передается (в режиме **Tx** или **Rx**))

Note: PEC calculation is corrupted by an arbitration loss.

Расчет **PEC** некорректен при потере прав на шину.

Bit 11 POS: Acknowledge/PEC Position (for data reception)

Позиция Acknowledge/PEC (при приеме)

This bit is set and cleared by software and cleared by hardware when PE=0.

Этот бит ставится и очищается программно, и очищается аппаратно, когда **PE=0**.

`0:` ACK bit controls the (N)ACK of the current byte being received in the shift register. The PEC bit indicates that current byte in shift register is a PEC.

бит **ACK** управляет этапом **(N)ACK** по приему текущего байта в сдвиговый регистр. Бит **PEC** показывает, что текущий байт в сдвиговом регистре является **PEC**.

`1:` ACK bit controls the (N)ACK of the next byte which will be received in the shift register. The PEC bit indicates that the next byte in the shift register is a PEC

бит **ACK** управляет этапом **(N)ACK** при приеме следующего байта в сдвиговый регистр. Бит **PEC** показывает, что следующий байт в сдвиговом регистре будет **PEC**

Note: The POS bit must be used only in 2-byte reception configuration and must be configured before data reception starts. To NACK the 2nd byte, the ACK bit must be cleared after ADDR is cleared. To check the 2nd byte as PEC, the PEC bit must be set during the ADDR stretch event after configuring the POS bit.

Бит **POS** должен использоваться только при приеме не менее 2-х байт и должен быть поставлен перед стартом приема данных. Чтобы поставить **NACK** на втором байте, бит **ACK** должен быть очищен после очистки бита **ADDR**. Чтобы проверить второй байт, как **PEC**, должен быть установлен бит **PEC** в процессе события растяжения **ADDR**, после установки бита **POS**.

Bit 10 ACK: Acknowledge enable (Разрешение уведомления)

This bit is set and cleared by software and cleared by hardware when PE=0.

Этот бит ставится и очищается программно, и очищается аппаратно, когда **PE=0**.

`0:` No acknowledge returned (Нет увеломления)

`1:` Acknowledge returned after a byte is received (matched address or data)

Возвращается бит увеломления после приема байта (корректного адреса или данных)

Bit 9 STOP: Stop generation (Генерация стоп-условия)

The bit is set and cleared by software, cleared by hardware when a Stop condition is detected, set by hardware when a timeout error is detected.

Этот бит ставится и очищается программно, очищается аппаратно, когда детектируется стоп-условие, ставится аппаратно, когда детектируется ошибка **timeout**.

In Master Mode: (В режиме "Ведущий":)

`0:` No Stop generation. (Нет генерации стоп-условия.)

`1:` Stop generation after the current byte transfer or after the current Start condition is sent.

Генерации стоп-условия после передачи текущего байта или после посылки текущего старт-условия.

In Slave mode: (В режиме "Ведомый":)

`0:` No Stop generation. (Нет генерации стоп-условия.)

`1:` Release the SCL and SDA lines after the current byte transfer.

Отпускает линии **SCL** и **SDA** после передачи текущего байта.

Note: When the STOP, START or PEC bit is set, the software must not perform any write access to I2C_CR1 before this bit is cleared by hardware. Otherwise there is a risk of setting a second STOP, START or PEC request.
Когда ставятся биты **STOP**, **START** или **PEC**, программа не должна выполнять какой либо записи в **I2C_CR1**, до того, как этот бит не будет очищен аппаратно. В противном случае есть риск установки повторного аналогичного бита.

Bit 8 START: Start generation (Генерация старт-условия)
This bit is set and cleared by software and cleared by hardware when start is sent or PE=0.
Этот бит ставится и очищается программно, очищается аппаратно, когда посылается старт-условие или **PE=0**.

In Master Mode: (В режиме "Ведущий":)
0: No Start generation (Нет генерации старт-условия)
1: Repeated start generation (Повторяющаяся генерация старт-условия)

In Slave mode: (В режиме "Ведомый":)
0: No Start generation (Нет генерации старт-условия)
1: Start generation when the bus is free (Генерация старт-условия, когда шина свободна)

Bit 7 NOSTRETCH: Clock stretching disable (Slave mode)
Отключение растяжки такта (в режиме "Ведомый")
This bit is used to disable clock stretching in slave mode when ADDR or BTF flag is set, until it is reset by software.
Этот бит используется для отключения растяжки такта в режиме "Ведомый", когда стоит флаг **ADDR** или **BTF**, до тех пор, пока бит не будет сброшен программно.
0: Clock stretching enabled (Растяжки такта разрешена)
1: Clock stretching disabled (Растяжки такта отключена)

Bit 6 ENGC: General call enable (Разрешение общего вызова)
0: General call disabled. Address 00h is NACKed.
Общий вызов отключен. В ответ на адрес **00h** посылается **NACK**.
1: General call enabled. Address 00h is ACKed.
Общий вызов разрешен. В ответ на адрес **00h** посылается **ACK**.

Bit 5 ENPEC: PEC enable (Разрешение PEC)
0: PEC calculation disabled (Расчет **PEC** отключен)
1: PEC calculation enabled (Расчет **PEC** разрешен)

Bit 4 ENARP: ARP enable (Разрешение ARP)
0: ARP disable (**ARP** отключен)
1: ARP enable (**ARP** разрешен)
SMBus Device default address recognized if SMBTYPE=0
SMBus Host address recognized if SMBTYPE=1
Распознается адрес устройства **SMBus**, если **SMBTYPE=0**
Распознается адрес хоста **SMBus**, **если SMBTYPE=1**

Bit 3 SMBTYPE: SMBus type (Тип SMBus устройства)
0: SMBus Device (Ведомый **SMBus** устройство)
1: SMBus Host (Ведущий **SMBus** устройство)

Bit 2 Reserved:
forced by hardware to 0. (Аппаратно ставится в 0.)

Bit 1 SMBUS: SMBus mode (Режим SMBus)
0: I2C mode (Режим **I2C**)
1: SMBus mode (Режим **SMBus**)

Bit 0 PE: Peripheral enable (Разрешение I2C интерфейса)

0: Peripheral disable (Интерфейс отключен)

1: Peripheral enable: the corresponding I/Os are selected as alternate functions depending on SMBus bit.

Интерфейс разрешен: соответствующий **I/O** выбирается как альтернативная функция, в зависимости от бита **SMBus**.

Note: If this bit is reset while a communication is on going, the peripheral is disabled at the end of the current communication, when back to IDLE state. All bit resets due to PE=0 occur at the end of the communication. In master mode, this bit must not be reset before the end of the communication.
Если этот бит сброшен в процессе коммуникации, то периферия отключается по окончанию текущего обмена, когда идет возврат в состояние **IDLE**. Вследствии того, что случилось **PE=0**, все биты сбрасываются по окончанию текущего обмена. В режиме "Ведущий", этот бит не должен сбрасываться до окончания обмена.

24.6.2 Control register 2 (I2C_CR2) Управляющий регистр 2

Address offset: 0x04
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			LAST	DMA EN	ITBUF EN	ITEVT EN	ITERR EN	Reserved		FREQ[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved:

forced by hardware to 0. (Аппаратно ставится в 0.)

Bit 12 LAST: DMA last transfer (последний обмен для DMA)

0: Next DMA EOT is not the last transfer

Следующее событие **DMA EOT** не является концом обмена

1: Next DMA EOT is the last transfer

Следующее событие **DMA EOT** означает конец обмена

Note: This bit is used in master receiver mode to permit the generation of a NACK on the last received data.

Этот бит используется в режиме ведущего приемника, чтобы позволить генерацию **NACK** по последнему принятому байту.

Bit 11 DMAEN: DMA requests enable (Разрешение запроса DMA)

0: DMA requests disabled (Запрос на **DMA** отключен)

1: DMA request enabled when TxE=1 or RxNE =1

Запрос на **DMA** разрешен, когда **TxE=1** или **RxNE=1**

Bit 10 ITBUFEN: Buffer interrupt enable (Разрешение прерывания от буфера)

0: TxE = 1 or RxNE = 1 does not generate any interrupt.

TxE=1 или **RxNE=1** не генерирует прерывание.

1: TxE = 1 or RxNE = 1 generates Event Interrupt (whatever the state of DMAEN)

TxE=1 или **RxNE=1** генерирует запрос на прерывание (при любом состоянии бита **DMAEN**)

Bit 9 ITEVTEN: Event interrupt enable (Разрешение прерывания от события)

`0:` Event interrupt disabled (Прерывание от события отключено)

`1:` Event interrupt enabled (Прерывание от события разрешено)

This interrupt is generated when: (Это прерывание генерируется, когда:)

- **SB=1 (Master)**
- **ADDR=1 (Master/Slave)**
- **ADD10=1 (Master)**
- **STOPF=1 (Slave)**
- **BTF=1** with no TxE or RxNE event (без наличия события от **TxE** или **RxNE**)
- **TxE=1** если **ITBUFEN=1**
- **RxNE=1** если **ITBUFEN=1**

Bit 8 ITERREN: Error interrupt enable (Разрешение прерывания от ошибки)

`0:` Error interrupt disabled (Прерывание от ошибки отключено)

`1:` Error interrupt enabled (Прерывание от ошибки разрешено)

This interrupt is generated when: (Это прерывание генерируется, когда:)

- **BERR=1**
- **ARLO=1**
- **AF=1**
- **OVR=1**
- **PECERR=1**
- **TIMEOUT=1**
- **SMBAlert=1**

Bits 7:6 Reserved:

forced by hardware to 0. (Аппаратно ставится в 0.)

Bits 5:0 FREQ[5:0]: Peripheral clock frequency (Входная частота периферии)

Input clock frequency must be programmed to generate correct timings The allowed range is between 2 MHz and 36 MHz

Входная частота периферии должна быть установлена так, чтобы корректно генерировать временные соотношения (тайминги) Допустимый диапазон от 2 до 36 мГц

`000000:` Not allowed (Значение недопустимо)

`000001:` Not allowed (Значение недопустимо)

`000010:` 2 MHz

`...`

`100100:` 36 MHz

`Higher than 100100:` Not allowed (Значение выше `100100` недопустимо)

24.6.3 Own address register 1 (I2C_OAR1) (Регистр собственного адреса 1)

Address offset: 0x08
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD MODE	Res.	Reserved				ADD[9:8]		ADD[7:1]							ADD0
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 ADDMODE: Addressing mode (slave mode)
Режим адресации в режиме "Ведомый")

`0:` 7-bit slave address (10-bit address not acknowledged)

7-ми битный режим адресации "Ведомого" (на 10-ти битный адрес нет уведомления)

`1:` 10-bit slave address (7-bit address not acknowledged)

10-ти битный режим адресации "Ведомого" (на 7-ми битный адрес нет уведомления)

Bit 14:
Must be configured and kept at 1. (При конфигурации всегда ставится в 1.)

Bits 13:10 Reserved:
forced by hardware to 0. (Аппаратно ставится в 0.)

Bits 9:8 ADD[9:8]: Interface address (Адрес интерфейса)
7-bit addressing mode: don't care (При 7-ми битном адресе: не имеет значения)
10-bit addressing mode: bits9:8 of address (При 10-ти битном адресе: 9,8 биты адреса

Bits 7:1 ADD[7:1]: Interface address (Адрес интерфейса)
bits 7:1 of address (7-1 биты адреса)

Bit 0 ADD0: Interface address (Адрес интерфейса)
7-bit addressing mode: don't care (При 7-ми битном адресе: не имеет значения)
10-bit addressing mode: bit 0 of address (При 10-ти битном адресе: 0 бит адреса)

24.6.4 Own address register 2 (I2C_OAR2) (Регистр собственного адреса 2)

Address offset: 0x0C
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ADD2[7:1]							ENDUAL
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved:
forced by hardware to 0. (Аппаратно ставится в 0.)

Bits 7:1 ADD2[7:1]: Interface address (Адрес интерфейса)
bits 7:1 of address in dual addressing mode (7-1 биты адреса в режиме двойной адресации)

Bit 0 ENDUAL: Dual addressing mode enable (Разрешение режима двойной адресации)
`0:` Only OAR1 is recognized in 7-bit addressing mode
Только **OAR1** распознается в режиме 7-ми битного адреса
`1:` Both OAR1 and OAR2 are recognized in 7-bit addressing mode
Оба регистра, **OAR1** и **OAR2** распознаются в режиме 7-ми битного адреса

24.6.5 Data register (I2C_DR) (Регистр данных)

Address offset: 0x10
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved:
forced by hardware to 0. (Аппаратно ставится в 0.)

Bits 7:0 DR[7:0] 8-bit data register (8-ми битный регистр данных)

Byte received or to be transmitted to the bus.
Байт принимается или передается на шину.

- **Transmitter mode**: Byte transmission starts automatically when a byte is written in the DR register. A continuous transmit stream can be maintained if the next data to be transmitted is put in DR once the transmission is started (TxE=1)
  Режим "передатчик": Передача байта стартует автоматически, когда он записывается в регистр **DR**. Можно поддерживать непрерывный поток передачи, если следующие данные для передачи помещаются в **DR** сразу после старта передачи (**TxE=1**)
- **Receiver mode**: Received byte is copied into DR (RxNE=1). A continuous transmit stream can be maintained if DR is read before the next data byte is received (RxNE=1).
  Режим "приемник": Принятый байт копируется в **DR** (**RxNE=1**). Можно поддерживать непрерывный поток приема, если **DR** читать перед приемом следующего байта данных (**RxNE=1**).

Note:

1. In slave mode, the address is not copied into DR.
   В режиме "Ведомый", адрес не копируется в **DR**.
2. Write collision is not managed (DR can be written if TxE=0).
   Коллизии записи не разруливаются (**DR** можно записывать, если **TxE=0**).
3. If an ARLO event occurs on ACK pulse, the received byte is not copied into DR and so cannot be read.
   Если произошло событие **ARLO**, на этапе **ACK**, то принятый байт не копируется в **DR**, и потому не может быть прочитан.

24.6.6 Status register 1 (I2C_SR1) (Статусный регистр 1)

Address offset: 0x14
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMB ALERT	TIME OUT	Res.	PEC ERR	OVR	AF	ARLO	BERR	TxE	RxNE	Res.	STOP F	ADD10	BTF	ADDR	SB
rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	r	r		r	r	r	r	r

Bit 15 SMBALERT: SMBus alert (Тревога SMBus)

In SMBus host mode: (В режиме Хоста **SMBus**:)

`0:` no SMBAlert (нет тревоги)
`1:` SMBAlert event occurred on pin (Событие тревоги на выводе)

In SMBus slave mode: (В ведомом режиме **SMBus** :)

`0:` no SMBAlert response address header (нет отклика)
`1:` SMBAlert response address header to SMBAlert LOW received

В ответ на низкий уровень на выводе **SMBAlert** откликается заголовком адреса Тревоги

- Cleared by software writing 0, or by hardware when PE=0.
  Очищается программно, записью нуля, или аппаратно, когда **PE=0**.

Bit 14 TIMEOUT: Timeout or Tlow error (Ошибка превышения времени)

`0:` No timeout error (Нет ошибки)

`1:` SCL remained LOW for 25 ms (Timeout)

SCL оставался в низком состоянии в течении более 25 ms (это **Timeout**)
or (или)
Master cumulative clock low extend time more than 10 ms (Tlow:mext)
"Ведущий" застягивал общее низкое состояние такта более чем на 10 ms (это **Tlow:mext**)
or (или)
Slave cumulative clock low extend time more than 25 ms (Tlow:sext)
"Ведомый" застягивал общее низкое состояние такта более чем на 25 ms (это **Tlow:sext**)

- When set in slave mode: slave resets the communication and lines are released by hardware
  Когда ставится в режиме "Ведомый": то он аппаратно сбрасывает обмен и отпускает линии
- When set in master mode: Stop condition sent by hardware
  Когда ставится в режиме "Ведущий": то аппаратно посылается стоп-условие
- Cleared by software writing 0, or by hardware when PE=0.
  Очищается программно, записью нуля, или аппаратно, когда **PE=0**.

Bit 13 Reserved:

forced by hardware to 0. (Аппаратно ставится в 0.)

Bit 12 PECERR: PEC Error in reception (Ошибка PEC при приеме)

`0:` no PEC error: receiver returns ACK after PEC reception (if ACK=1)
Нет ошибки: приемник возвращает **ACK** после приема **PEC** (если **ACK=1**)

`1:` PEC error: receiver returns NACK after PEC reception (whatever ACK)
Ошибка **PEC**: приемник возвращает **NACK** после приема **PEC** (независимо от состояния **ACK**)

- Cleared by software writing 0, or by hardware when PE=0.
  Очищается программно, записью нуля, или аппаратно, когда **PE=0**.

Bit 11 OVR: Overrun/Underrun (Ошибка Переполнения/Недостачи)

`0:` No overrun/underrun (Нет ошибки)

`1:` Overrun or underrun (Переполнение или недостача)

- Set by hardware in slave mode when NOSTRETCH=1 and:
  Ставится аппаратно в режиме "Ведомый", когда **NOSTRETCH=1** и:
  - In reception when a new byte is received (including ACK pulse) and the DR register has not been read yet, new received byte is lost.
    при приеме, когда получен новый байт (включая этап **ACK**) и регистр **DR** пока не был прочитан, новый принятый байт теряется.
  - In transmission when a new byte should be sent and the DR register has not been written yet, the same byte is sent twice.
    при передаче, когда новый байт должен быть послан и регистр **DR** еще не был записан, то тот же байт посылается повторно.
- Cleared by software writing 0, or by hardware when PE=0.
  Очищается программно, записью нуля, или аппаратно, когда **PE=0**.

Note: If the DR write occurs very close to SCL rising edge, the sent data is unspecified and a hold timing error occurs
Если запись в **DR** происходит очень близко с фронтом **SCL**, то посылаемые данные не определены и происходит ошибка **hold timing**.

Bit 10 AF: Acknowledge failure (Ошибка уведомления)

0: No acknowledge failure (Нет ошибки)

1: Acknowledge failure (Ошибка уведомления)

- Set by hardware when no acknowledge is returned.
  Ставится аппаратно, когда нет возврата уведомления.
- Cleared by software writing 0, or by hardware when PE=0.
  Очищается программно, записью нуля, или аппаратно, когда **PE=0**.

Bit 9 ARLO: Arbitration lost (master mode)
Ошибка потери прав на шину (в режиме "Ведущий")

0: No Arbitration Lost detected (Нет ошибки)

1: Arbitration Lost detected (Ошибка потери прав на шину)

Set by hardware when the interface loses the arbitration of the bus to another master
Ставится аппаратно, когда интерфейс теряет права на шину (захватывает другое устройство)

- Cleared by software writing 0, or by hardware when PE=0.
  Очищается программно, записью нуля, или аппаратно, когда **PE=0**.

After an ARLO event the interface switches back automatically to Slave mode (M/SL=0).
После события **ARLO**, интерфейс переключается автоматически обратно в режим "Ведомый" (**M/SL=0**).

Note: In SMBUS, the arbitration on the data in slave mode occurs only during the data phase, or the acknowledge transmission (not on the address acknowledge).
В режиме **SMBUS**, для "Ведомого" событие захвата шины может произойти только во время фазы данных, включая бит уведомления (но не на фазе адресации).

Bit 8 BERR: Bus error (Ошибка шины)

0: No misplaced Start or Stop condition (Нет ошибки)

1: Misplaced Start or Stop condition (Неуместное старт- или стоп-условие)

- Set by hardware when the interface detects a misplaced Start or Stop condition
  Ставится аппаратно, когда интерфейс определяет неуместное старт- или стоп-условие
- Cleared by software writing 0, or by hardware when PE=0.
  Очищается программно, записью нуля, или аппаратно, когда **PE=0**.

Bit 7 TxE: Data register empty (transmitters) (Регистр данных пуст (при передаче))

0: Data register not empty (Нет события)

1: Data register empty (Регистр данных пуст)

- Set when DR is empty in transmission. TxE is not set during address phase.
  Ставится, когда опустошается **DR** в процессе передачи. **TxE** не ставится на фазе адресации.
- Cleared by software writing to the DR register or by hardware after a start or a stop condition or when PE=0.
  Очищается программно, записью в регистр **DR**, или аппаратно, после старт- или стоп-условия, или когда **PE=0**.

TxE is not set if either a NACK is received, or if next byte to be transmitted is PEC (PEC=1)
TxE не ставится, если принят **NACK**, или если следующий байт для передачи является **PEC** (**PEC=1**)

Note: TxE is not cleared by writing the first data being transmitted, or by writing data when BTF is set, as in both cases the data register is still empty.
TxE не очищается записью первого байта данных для передачи, или записью данных, когда стоит бит **BTF**, в обоих случаях регистр данных все еще пуст.

Bit 6 RxNE: Data register not empty (receivers) (Регистр данных не пустой (при приеме))

`0:` Data register empty (Нет события)

`1:` Data register not empty (Регистр данных не пустой)

- Set when data register is not empty in receiver mode. RxNE is not set during address phase.
  Ставится, когда регистр данных не пустой при приеме. **RxNE** не ставится а фазе адресации.
- Cleared by software reading or writing the DR register or by hardware when PE=0.
  Очищается программно, чтением или записью в регистр **DR**, или аппаратно, когда **PE=0**.

RxNE is not set in case of ARLO event. (**RxNE** не ставится при наличии события **ARLO**.)

Note: RxNE is not cleared by reading data when BTF is set, as the data register is still full.
RxNE не очищается чтением данных, когда стоит **BTF**, так как регистр данных все еще полон.

Bit 5 Reserved:

forced by hardware to 0. (Аппаратно ставится в 0.)

Bit 4 STOPF: Stop detection (slave mode)
Детектирование стоп-условия (в режиме "Ведомый")

`0:` No Stop condition detected (Нет события)

`1:` Stop condition detected (Детектировано стоп-условие)

- Set by hardware when a Stop condition is detected on the bus by the slave after an acknowledge (if ACK=1).
  Ставится аппаратно, когда "Ведомый" детектировал стоп-условие на шине после уведомления (если **ACK=1**).
- Cleared by software reading the SR1 register followed by a write in the CR1 register, or by hardware when PE=0
  Очищается программно, чтением регистра **SR1** с последующей записью в регистр **CR1**, или аппаратно, когда **PE=0**

Note: The STOPF bit is not set after a NACK reception
Бит **STOPF** не ставится после получения **NACK**

Bit 3 ADD10: 10-bit header sent (Master mode)
Посылка заголовка при 10-ти битной адресации (в режиме "Ведущий")

`0:` No ADD10 event occurred. (Нет события.)

`1:` Master has sent first address byte (header).

"Ведущий" послал первый байт адреса (заголовок).

- Set by hardware when the master has sent the first byte in 10-bit address mode.
  Ставится аппаратно, когда посылает первый байт адреса при 10-ти битной адресации.
- Cleared by software reading the SR1 register followed by a write in the DR register of the second address byte, or by hardware when PE=0.
  Очищается программно, чтением регистра **SR1** с последующей записью в регистр **DR** второго байта адреса, или аппаратно, когда **PE=0**.

Note: ADD10 bit is not set after a NACK reception
Бит **ADD10** не ставится после получения **NACK**

Bit 2 BTF: Byte transfer finished (Завершение передачи байта)

0: Data byte transfer not done (Нет события)

1: Data byte transfer succeeded (Байт данных передан успешно)

- Set by hardware when NOSTRETCH=0 and:
  Ставится аппаратно, когда **NOSTRETCH=0** и:
  - In reception when a new byte is received (including ACK pulse) and DR has not been read yet (RxNE=1).
    при приеме, когда принят новый байт (включая этап **ACK**), а **DR** еще не был прочитан (**RxNE=1**).
  - In transmission when a new byte should be sent and DR has not been written yet (TxE=1).
    при передаче, когда надо передавать новый байт, а **DR** пока еще не был записан (**TxE=1**).
  - Cleared by software reading SR1 followed by either a read or write in the DR register or by hardware after a start or a stop condition in transmission or when PE=0.
    Очищается программно, чтением регистра **SR1** с последующим чтением или записью регистра **DR**, или аппаратно, после старт- или стоп-условия при передаче, или когда **PE=0**.

Note: The BTF bit is not set after a NACK reception
The BTF bit is not set if next byte to be transmitted is the PEC (TRA=1 in I2C_SR2 register and PEC=1 in I2C_CR1 register)
Бит **BTF** не ставится после получения **NACK**.
Бит **BTF** не ставится, если следующий байт для передачи является **PEC** (**TRA=1** в регистре **I2C_SR2** и **PEC=1** в регистре **I2C_CR1**)

Bit 1 ADDR: Address sent (master mode)/matched (slave mode)
Послан адрес (в режиме "Ведущий") или принят адрес (в режиме "Ведомый")

This bit is cleared by software reading SR1 register followed reading SR2, or by hardware when PE=0.
Этот бит очищается программно чтением регистра **SR1** с последующим чтением регистра **SR2**, или аппаратно, когда **PE=0**.

Address matched (Slave) (Адрес соответствует (у "Ведомого"))

0: Address mismatched or not received. (Нет адреса или он не соответствует.)

1: Received address matched. (Полученный адрес соответствует.)

- Set by hardware as soon as the received slave address matched with the OAR registers content or a general call or a SMBus Device Default Address or SMBus Host or SMBus Alert is recognized. (when enabled depending on configuration).
  Ставится аппаратно, как только получен корректный адрес для ведомого устройства (равный содержимому регистра **OAR**), или адрес общего вызова, или опознан один из адресов **SMBus** (**SMBus Device Default** или **SMBus Host** или **SMBus Alert**). (зависит от конфигурации).

Address sent (Master) (Посылка адреса "Ведущим"))

0: No end of address transmission (Нет события)

1: End of address transmission (Конец передачи адреса)

- For 10-bit addressing, the bit is set after the ACK of the 2nd byte.
  При 10-ти битной адресации, бит ставится после получения **ACK** для второго байта.
- For 7-bit addressing, the bit is set after the ACK of the byte.
  При 7-ми битной адресации, бит ставится после получения **ACK**.

Note: ADDR is not set after a NACK reception
Бит **ADDR** не ставится после получения **NACK**

Bit 0 SB: Start bit (Master mode) (Бит старт-условия (в режиме "Ведущий"))

0: No Start condition (Нет старт-условия)

1: Start condition generated. (Генерация старт-условия.)

- Set when a Start condition generated. (Ставится, когда генерируется старт-условие.)
- Cleared by software by reading the SR1 register followed by writing the DR register, or by hardware when PE=0
  Очищается программно, чтением регистра **SR1** с последующей записью в регистр **DR**, или аппаратно, когда **PE=0**

24.6.7 Status register 2 (I2C_SR2) (Статусный регистр 2)

Address offset: 0x18
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7:0]								DUALF	SMB HOST	SMB DEF AULT	GEN CALL	Res.	TRA	BUSY	MSL
r	r	r	r	r	r	r	r	r	r	r	r		r	r	r

Bits 15:8 PEC[7:0] Packet error checking register (Регистр контрольной суммы кадра)

This register contains the internal PEC when ENPEC=1.
Этот регистр содержит внутренний **PEC**, когда **ENPEC=1**

Bit 7 DUALF: Dual flag (Slave mode) (Флаг двойной адресации (в режиме "Ведомый"))

0: Received address matched with OAR1 (Полученный адрес сравнивается только с **OAR1**)

1: Received address matched with OAR2 (Полученный адрес сравнивается также с **OAR2**)

- Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.
  Очищается аппаратно, после стоп-условия, или повторного старт-условия, или когда **PE=0**.

Bit 6 SMBHOST: SMBus host header (Slave mode)
Принят заголовок SMBus host (в режиме "Ведомый")

0: No SMBus Host address (Нет **SMBus Host** адресации)

1: SMBus Host address received when SMBTYPE=1 and ENARP=1.
Был принят **SMBus Host** адрес, когда **SMBTYPE=1** и **ENARP=1**.

- Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.
  Очищается аппаратно, после стоп-условия, или повторного старт-условия, или когда **PE=0**.

Bit 5 SMBDEFAULT: SMBus device default address (Slave mode)
Принят адрес по умолчанию для SMBus устройства (в режиме "Ведомый")

0: No SMBus Device Default address (Нет адресации по умолчанию)

1: SMBus Device Default address received when ENARP=1
Был принят адрес по умолчанию для **SMBus** устройства, когда **ENARP=1**

- Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.
  Очищается аппаратно, после стоп-условия, или повторного старт-условия, или когда **PE=0**.

Bit 4 GENCALL: General call address (Slave mode)
Принят адрес общего вызова (в режиме "Ведомый")

0: No General Call (Нет общего вызова)
1: General Call Address received when ENGC=1

Был принят адрес общего вызова, когда **ENGC=1**

- Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.
  Очищается аппаратно, после стоп-условия, или повторного старт-условия, или когда **PE=0**.

Bit 3 Reserved:

forced by hardware to 0. (Аппаратно ставится в 0.)

Bit 2 TRA: Transmitter/receiver (Режим передатчик/приемник)

0: Data bytes received (Прием данных)
1: Data bytes transmitted (Передача данных)

This bit is set depending on the R/W bit of the address byte, at the end of total address phase. It is also cleared by hardware after detection of Stop condition (STOPF=1), repeated Start condition, loss of bus arbitration (ARLO=1), or when PE=0.
Этот бит ставится в зависимости от бита **R/W** в байте адреса, который является последним в общей фазе адресации. Он также чищается аппаратно, после стоп-условия (**STOPF=1**), повторного старт-условия, потери арбитража (**ARLO=1**), или когда **PE=0**.

Bit 1 BUSY: Bus busy (Шина занята)

0: No communication on the bus (Нет обмена на шине)
1: Communication ongoing on the bus (Шина в состоянии обмена)

- Set by hardware on detection of SDA or SCL low
  Ставится аппаратно при детектировании низкого уровня на **SDA** или **SCL**
- cleared by hardware on detection of a Stop condition.
  Очищается аппаратно при детектировании стоп-условия.

It indicates a communication in progress on the bus. This information is still updated when the interface is disabled (PE=0).
Он показывает, что шина находится в состоянии обмена. Эта информация продолжает обновляться, когда интерфейс отключен (**PE=0**).

Bit 0 MSL: Master/slave (Режим "Ведущий/Ведомый")

0: Slave Mode (Режим "Ведомый")
1: Master Mode (Режим "Ведущий")

- Set by hardware as soon as the interface is in Master mode (SB=1).
  Ставится аппаратно, как только интерфейс переходит в режим "Ведущий" (**SB=1**).
- Cleared by hardware after detecting a Stop condition on the bus or a loss of arbitration (ARLO=1), or by hardware when PE=0.
  Очищается аппаратно, после детектировании стоп-условия на шине или потери прав на шину (**ARLO=1**), или аппаратно, когда **PE=0**.

24.6.8 Clock control register (I2C_CCR)
Регистр управления тактовым сигналом

Address offset: 0x1C
Reset value: 0x0000

(*Здесь пара прмечаний, которые повторяются ниже.*)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	Reserved		CCR[11:0]											
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 F/S: I2C master mode selection (Выбор скорости для режима "Ведущий")
- `0:` Standard Mode I2C (Стандартный **I2C**)
- `1:` Fast Mode I2C (Быстрый **I2C**)

Bit 14 DUTY: Fast mode duty cycle (Скважность в быстром режиме)
- `0:` **Fast Mode tLow/tHigh = 2**
- `1:` **Fast Mode tLow/tHigh = 16/9** (см. **CCR**)

Bits 13:12 Reserved:
forced by hardware to 0. (Аппаратно ставится в 0.)

Bits 11:0 CCR[11:0]: Clock control register in Fast/Standard mode (Master mode)
Управление тактовым сигналом для Быср ой/Стандартной скорости (в режиме "Ведущий")
Controls the SCL clock in master mode.
Управляет сигналом на **SCL** в режиме "Ведущий".

Standard mode or SMBus: (Стандартный режим или **SMBus**:)

```
tHigh = CCR * TPCLK1
tLow  = CCR * TPCLK1
```

Fast mode: (Быстрый режим:)

```
If DUTY = 0:
   tHigh = CCR * TPCLK1
   tLow  = CCR * TPCLK1 * 2
If DUTY = 1: (чтобы получить 400 kHz)
   tHigh =  9 * CCR * TPCLK1
   tLow  = 16 * CCR * TPCLK1
```

For instance: in standard mode, to generate a 100 kHz SCL frequency:
If FREQR = 08, T = 125 ns so CCR must be programmed with 0x28
Например: в стандартном режиме, чтобы генерировать частоту **SCL** 100 kHz:
Если `FREQR=08, TPCLK1=125 ns` то в **CCR** должно быть запрограммировано значение **0x28**
`(0x28 <=> 40d x 125 ns = 5000 ns.)`

Note:
1. The minimum allowed value is 0x04, except in FAST DUTY mode where the minimum allowed value is 0x01
   Минимально допустимое значение **0x04**, за исключением режима **FAST DUTY**, где допускается значение **0x01**

2. t_{high} includes the SCLH rising edge (**tHigh** включает фронт **SCLH**)
3. t_{low} includes the SCLH falling edge (**tLow** включает срез **SCLH**)
4. These timings are without filters. (Это тайминги без фильтрации.)
5. The CCR register must be configured only when the I2C is disabled (PE = 0).
   Регистр **CCR** можно менять только тогда, когда **I2C** отключена (**PE=0**).
6. f_{CK} = a multiple of 10 MHz is required to generate the fast clock at 400 kHz.
   Чтобы генерировать частоту 400 кГц в быстром режиме, значение $\mathbf{f_{CK}}$ должно быть кратно 10 мГц.

24.6.9 TRISE register (I2C_TRISE) (Регистр TRISE)

Address offset: 0x20
Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										TRISE[5:0]					
Res.										rw	rw	rw	rw	rw	rw

Bits 15:6 Reserved:
forced by hardware to 0. (Аппаратно ставится в 0.)

Bits 5:0 TRISE[5:0]: Maximum rise time in Fast/Standard mode (Master mode)
Максимальное время фронта в режиме Fast/Standard (в режиме "Ведущий")
These bits must be programmed with the maximum SCL rise time given in the I2C bus specification, incremented by 1.
Эти биты должны быть запрограммированы на максимальное время фронта **SCL**, что дает спецификация шины **I2C**, плюс 1.

For instance: in standard mode, the maximum allowed SCL rise time is 1000 ns.
Например: в стандартном режиме, максимально допустимое время фронта **SCL** - 1000 ns.

If, in the I2C_CR2 register, the value of FREQ[5:0] bits is equal to 0x08 and TPCLK1 = 125 ns therefore the TRISE[5:0] bits must be programmed with 09h.
Если в регистре **I2C_CR2** значение бит **FREQ[5:0]** равно **0x08** и **TPCLK1=125 ns**, следовательно биты **TRISE[5:0]** должны быть запрограммированы значением **09h**.

```
(1000 ns / 125 ns = 8 + 1)
```

The filter value can also be added to TRISE[5:0].
К **TRISE[5:0]** может быть добавлено значение фильтра.

If the result is not an integer, TRISE[5:0] must be programmed with the integer part, in order to respect the t_{HIGH} parameter.
Если результат не является целым значением, в **TRISE[5:0]** должна быть запрограммирована целая часть, чтобы получить ожидаемое значение параметра **tHigh**.

Note: TRISE[5:0] must be configured only when the I2C is disabled (PE = 0).
TRISE[5:0] можно менять только тогда, когда **I2C** отключена (**PE=0**).

24.6.10 I2C register map (Карта регистров I2C)

The table below provides the I2C register map and reset values.
В таблице ниже дана карта регистров **I2C** и их значение после сброса.

Table 172. I2C register map and reset values
Карта регистров **I2C** и их значение после сброса

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	Reserved																SWRST	Reserved	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	Reserved	SMBUS	PE
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0		0	0
0x04	I2C_CR2	Reserved																			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Reserved	FREQ[5:0]						
	Reset value																				0	0	0	0	0		0	0	0	0	0	0	0
0x08	I2C_OAR1	Reserved																ADDMODE	Reserved	Reserved				ADD[9:8]		ADD[7:1]							ADD0
	Reset value																	0	1					0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2	Reserved																								ADD2[7:1]							ENDUAL
	Reset value																									0	0	0	0	0	0	0	0
0x10	I2C_DR	Reserved																								DR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	I2C_SR1	Reserved																SMBALERT	TIMEOUT	Reserved	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Reserved	STOPF	ADD10	BTF	ADDR	SB
	Reset value																	0	0		0	0	0	0	0	0	0		0	0	0	0	0
0x18	I2C_SR2	Reserved																PEC[7:0]								DUALF	SMBHOST	SMBDEFAULT	GENCALL	Reserved	TRA	BUSY	MSL
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x1C	I2C_CCR	Reserved																F/S	DUTY	Reserved	CCR[11:0]												
	Reset value																	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	I2C_TRISE	Reserved																								TRISE[5:0]							
	Reset value																									0	0	0	0	1	0		

25 Universal synchronous/asynchronous receiver transmitter (USART) (Универсальный синхронный/асинхронный приемо-передатчик USART)

25.1 USART introduction (Введение в **USART**)
25.2 USART main features (Основные особенности **USART**)
25.3 USART functional description (Функциональное описание **USART**)
25.4 USART interrupts (Прерывания от **USART**)
25.5 USART mode configuration (Конфигурация режимов **USART**)
25.6 USART registers (**USART** регистры)

(*Здесь находится ритуальное напоминание о наличии 4-х семейств для **STM32F10xxx**, и что этот документ обо всех семействах, если нет прямого указания. См. раздел 1.2 Глоссарий).*)

25.1 USART introduction (Введение в USART)

The universal synchronous asynchronous receiver transmitter (USART) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The USART offers a very wide range of baud rates using a fractional baud rate generator.
Универсальный синхронный/асинхронный приемо-передатчик **USART** (*Universal synchronous asynchronous receiver transmitter*) предоставляет гибкий способ полно-дуплексного обмена данными с внешними устройствами, требующими индустриального стандарта **NRZ** на асинхронный последовательный формат данных. **USART** предоставляет очень широкий спектр скоростей обмена (*baud rates*) за счет фракционного **baud rate** генератора.

It supports synchronous one-way communication and half-duplex single wire communication. It also supports the LIN (local interconnection network), Smartcard Protocol and IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). It allows multiprocessor communication.
Интерфейс поддерживает одностороннюю (*one-way*) коммуникацию и полу-дуплексную коммуникацию по одному проводу. Он поддерживает также протокол **LIN** (*local interconnection network*), протокол **Smartcard** и протокол **IrDA** (*infrared data association*) по спецификации **SIR ENDEC**, а также режим модема (**CTS/RTS**). Он позволяет также организовывать многопроцессорный обмен.

High speed data communication is possible by using the DMA for multibuffer configuration.
Возможен высоко-скоростной обмен данными за счет использования **DMA** в многобуферной конфигурации.

25.2 USART main features (Основные особенности USART)

- Full duplex, asynchronous communications

Полно-дуплексный, асинхронный обмен данными

- NRZ standard format (Mark/Space) (Стандартный формат **NRZ** (**Mark/Space**))

- Fractional baud rate generator systems (Фракционный **baud rate** генератор)
  – A common programmable transmit and receive baud rates up to 4.5 MBits/s
  наиболее употребимый ряд скоростей приема и передачи, до 4.5 MBits/s

- Programmable data word length (8 or 9 bits)

Программируемая длина слова данных (8 или 9 бит)

- Configurable stop bits - support for 1 or 2 stop bits

Программируемое число стоп битов (1 или 2)

- LIN Master Synchronous Break send capability and LIN slave break detection capability

Возможность посылки сигнала **LIN Break** "Ведущим" в синхронном режиме и детектирования сигнала **LIN Break** на стороне "Ведомого"

  – 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
  Генерация 13-ти битного **Break** символа и детектирование 10/11-ти битного **Break** символа, когда аппаратная часть **USART** сконфигурирована на работу с **LIN**

- Transmitter clock output for synchronous transmission

Передатчик выдает тактовый сигнал для синхронизации обмена

- IrDA SIR Encoder Decoder (**SIR** кодер/декодер для **IrDA**)
  – Support for 3/16 bit duration for normal mode
  Поддерживает длительность импульса 3/16 периода бита в нормальном режиме

- Smartcard Emulation Capability (Возможность эмуляции **Smartcard**)
  – The Smartcard interface supports the asynchronous protocol Smartcards as defined in ISO 7816-3 standards интерфейс **Smartcard** поддерживает асинхронный протокол, как определено в стандарте **ISO 7816-3**

  – 0.5, 1.5 Stop Bits for Smartcard operation
  Формат 0.5, 1.5 стоп бита для режима **Smartcard**

- Single wire half duplex communication

Однопроводный полу-дуплексный обмен данными

- Configurable multibuffer communication using DMA (direct memory access)

Конфигурируемый многобуферный обмен с использованием **DMA**

  – Buffering of received/transmitted bytes in reserved SRAM using centralized DMA
  Буфферизация принимаемых/посылаемых байт в зарезервированной **SRAM**

- Separate enable bits for Transmitter and Receiver

Раздельные биты разрешения для Приемника и Передатчика

- Transfer detection flags: (Флаги, выставляемые при обмене:)
  – Receive buffer full (Буфер Приемника полон)
  – Transmit buffer empty (Буфер Передатчика пуст)
  – End of Transmission flags (Конец передачи)

- Parity control: (Управление паритетом:)
  – Transmits parity bit (Установка бита паритета при передаче байта)
  -- Checks parity of received data byte (Проверка бита паритета при приеме байта)

- Four error detection flags: (Детектируются четыре ошибки (выставляются флаги):)
  – Overrun error (Ошибка Переполнения)
  – Noise error (Ошибка Шума)
  – Frame error (Ошибка Кадра)
  – Parity error (Ошибка Паритета)

- Ten interrupt sources with flags: (10 источников прерываний:)
  – CTS changes (Изменение на линии **CTS**)
  – LIN break detection (Обнаружение **LIN Break** символа)
  – Transmit data register empty (Опустошение регистра передачи)
  – Transmission complete (Завершение передачи)
  – Receive data register full (Регистр приема полон)
  – Idle line received (Получено состояние простоя)
  – Overrun error (Ошибка Переполнения)
  – Framing error (Ошибка Кадра)
  – Noise error (Ошибка Шума)
  – Parity error (Ошибка Паритета)

- Multiprocessor communication - enter into mute mode if address match does not occur
Многопроцессорный обмен - вход в "безмолвный" режим при несовпадении адреса

- Wake up from mute mode (by idle line detection or address mark detection)
Пробуждение из "безмолвного" режима (при обнаружении состояния **Idle** либо при обнаружении адресного маркера)

- Two receiver wakeup modes: Address bit (MSB, 9-th bit), Idle line
Два режима пробуждения приемника: "**Address bit**", и "**Idle line**"

25.3 USART functional description Функциональное описание USART

25.3.1 USART character description Описание кадра **USART**
25.3.2 Transmitter (Передатчик)
25.3.3 Receiver (Приемник)
25.3.4 Fractional baud rate generation (Фракционный **baud rate** генератор)
25.3.5 USART receiver's tolerance to clock deviation
Терпимость приемника **USART** к девиации тактовой частоты
25.3.6 Multiprocessor communication (Многопроцессорный обмен)
25.3.7 Parity control (Контроль паритета)
25.3.8 LIN (local interconnection network) mode (Режим **LIN** (Локальная сеть))
25.3.9 USART synchronous mode (Синхронный режим **USART**)
25.3.10 Single-wire half-duplex communication
(Обмен в однопроводном полу-дуплексном режиме)
25.3.11 Smartcard
25.3.12 IrDA SIR ENDEC block (Блок **SIR** кодера/декодера для **IrDA**)
25.3.13 Continuous communication using DMA (Непрерывный обмен с использованием **DMA**)
25.3.14 Hardware flow control (Аппаратное управление процессом обмена)

The interface is externally connected to another device by three pins (see *Figure 238*). Any USART bidirectional communication requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX): Интерфейс имеет внешнее подключение к другому устройству с помощью трех выводов (см. Рис. 238). Любой двунаправленный обмен по **USART** требует не менее двух выводов: Вход приема **RX** и выход передачи **TX**:

RX: Receive Data Input is the serial data input. Oversampling techniques are used for data recovery by discriminating between valid incoming data and noise. **RX**: Вход приема - это вход последовательных данных. Используется техника передискретизации (*Oversampling*) для распознавания данных за счет разделения (дискриминации) правильных данных на фоне шума.

TX: Transmit Data Output. When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire and smartcard modes, this I/O is used to transmit and receive the data (at USART level, data are then received on SW_RX). **TX**: Выход передачи. Когда передатчик выключен, его вывод принимает исходное назначение - **GPIO**. Когда передатчик разрешен, но нет данных для передачи, то вывод **TX** находится в состоянии высокого уровня. В режимах однопроводного обмена или **Smartcard**, этот вывод используется как для передачи, так и для приема данных (на стороне **USART** данные поступают на внутренний вход приемника **SW_RX**).

Through these pins, serial data is transmitted and received in normal USART mode as frames comprising: Через этот вывод последовательные данные передаются и принимаются в нормальном режиме **USART** в виде кадра, который содержит:

- An Idle Line prior to transmission or reception
Состояние покоя (простоя) **Idle**, которое предшествует передаче или приему
- A start bit (Стартовый бит)
- A data word (8 or 9 bits) least significant bit first
Слово данных (8 или 9 бит), младший бит идет первым
- 0.5,1, 1.5, 2 Stop bits indicating that the frame is complete
0.5,1, 1.5, 2 стоп бита, показывающих, что кадр завершен

- This interface uses a fractional baud rate generator - with a 12-bit mantissa and 4-bit fraction Этот интерфейс использует фракциональный **baud rate** генератор - с 12-ти битной мантиссой 4-х битной дробной частью

- A status register (USART_SR) (Статусный регистр (**USART_SR**))

- Data Register (USART_DR) (Регистр данных (**USART_DR**))

- A baud rate register (USART_BRR) - 12-bit mantissa and 4-bit fraction.
Регистр скорости обмена (**USART_BRR**) - 12-bit мантисса и 4-bit дробная часть.

- A Guardtime Register (USART_GTPR) in case of Smartcard mode.
Регистр задержки **Guardtime** (**USART_GTPR**) в режиме **Smartcard**.

Refer to *Section 25.6: USART registers on page 683* for the definitions of each bit.
См раздел 25.6 "**USART** регистры", где описаны эти биты.

The following pin is required to interface in synchronous mode:
Для обмена в синхронном режиме требуются следующие выводы:

- **SCLK**: Transmitter clock output. This pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel data can be received synchronously on RX. This can be used to control peripherals that have shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable. In smartcard mode, SCLK can provide the clock to the smartcard.
SCLK: выход тактового сигнала передачи. Этот выход дает тактовый сигнал для синхронного обмена в соответствии с режимом **SPI master** (нет тактовых импульсов во время передачи стартового и стопового бита, и программная опция для посылки тактового по последнему биту данных). Одновременно данные могут синхронно приниматься на выводе **RX**. Это можно использовать для управления периферией, которая имеет сдвиговые регистры (например **LCD** драйверы). Фаза тактового сигнала и его полярность программируются. В режиме **Smartcard**, вывод **SCLK** можно использовать как тактовый для **smartcard**.

The following pins are required to interface in IrDA mode:
Для обмена в **IrDA** режиме требуются следующие выводы:

- **IrDA_RDI**: Receive Data Input is the data input in IrDA mode.
Вход принимаемого сигнала в режиме **IrDA**.

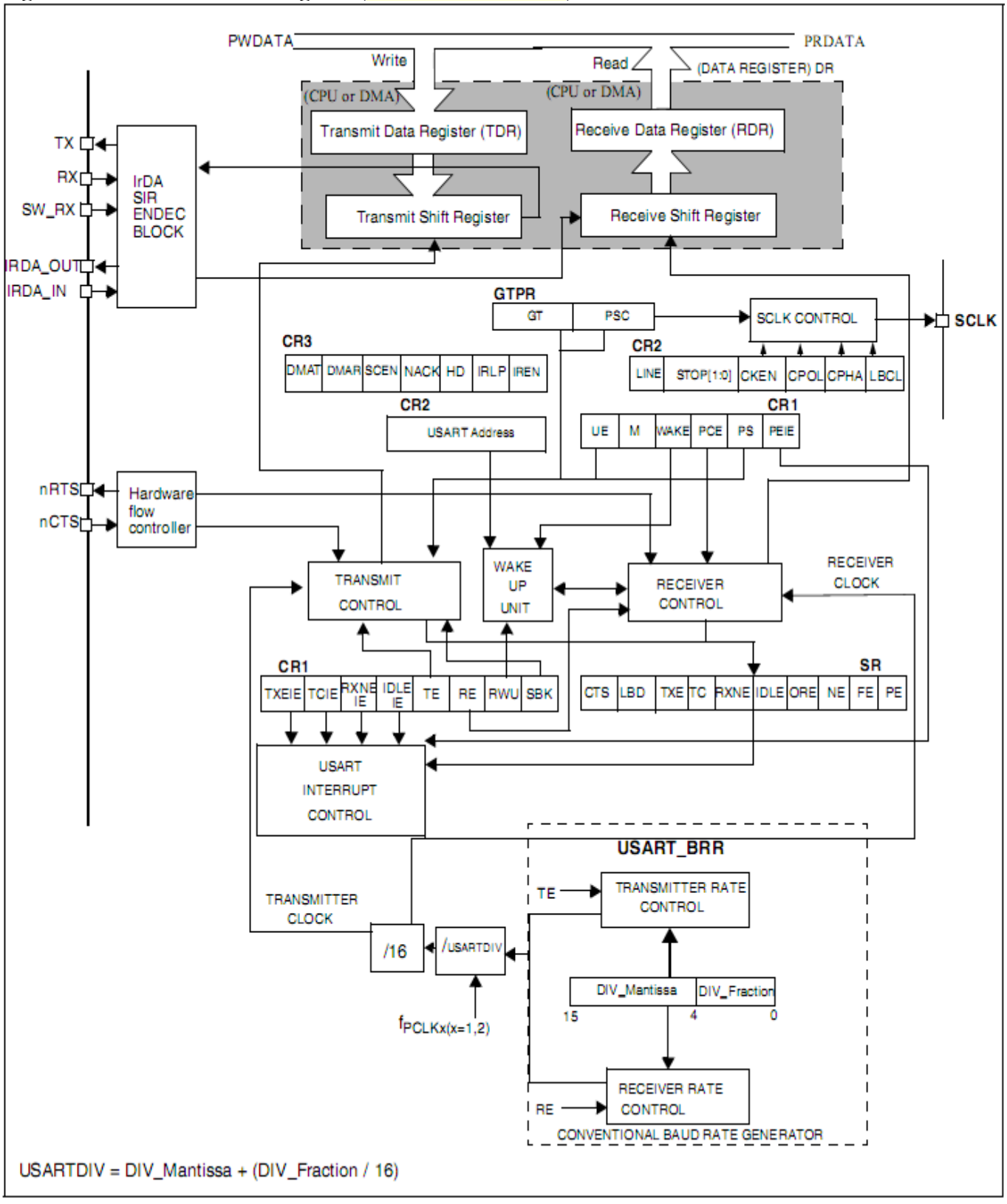
- **IrDA_TDO**: Transmit Data Output in IrDA mode.
Выход передаваемого сигнала в режиме **IrDA**.

the following pins are required in Hardware flow control mode:
Для обмена с аппаратным управлением потока данных требуются следующие выводы:

- **nCTS**: Clear To Send blocks the data transmission at the end of the current transfer when high
Высокий уровень этого сигнала выставляется по окончанию текущего обмена

- **nRTS**: Request to send indicates that the USART is ready to receive a data (when low).
Низкий уровень этого сигнала показывает, что **USART** готов принять данные.

Figure 238. USART block diagram (Блок схема **USART**)



25.3.1 USART character description (Описание кадра USART)

Word length may be selected as being either 8 or 9 bits by programming the M bit in the USART_CR1 register (see *Figure 239*).
Длина слова может быть выбрана как 8 или 9 бит программно в поле **M** регистра **USART_CR1**.

The TX pin is in low state during the start bit. It is in high state during the stop bit.
Вывод **TX** находится в низком состоянии в течении старт-бита. И он находится в высоком состоянии в течении стоп-бита.

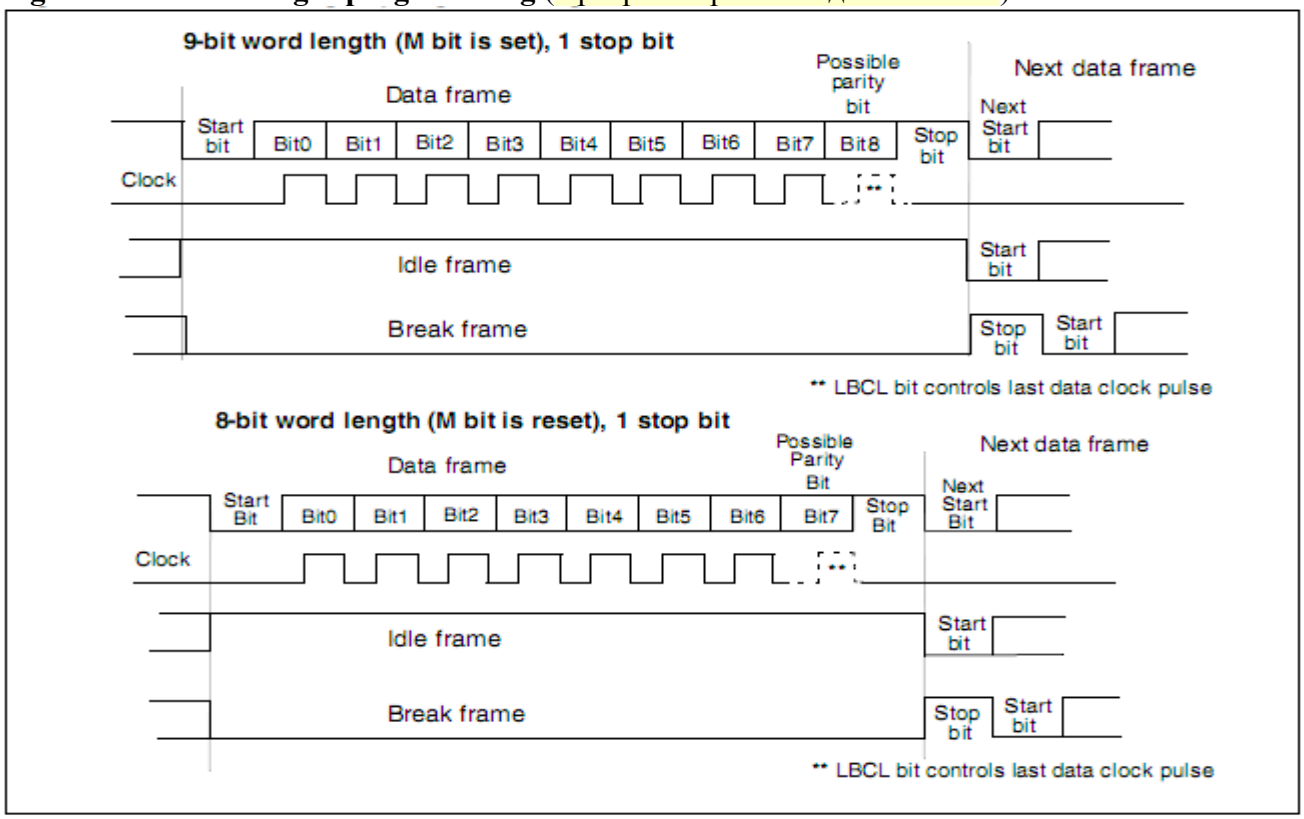
An ***Idle character*** is interpreted as an entire frame of "1"s followed by the start bit of the next frame which contains data (The number of "1" 's will include the number of stop bits).
Символ **Idle** интерпертируется как целый кадр из логических единиц, которые следуют после старт-бита, с последующим кадром, содержащим полезные данные (число '1' включает число стоп-бит).

A ***Break character*** is interpreted on receiving "0"s for a frame period. At the end of the break frame the transmitter inserts either 1 or 2 stop bits (logic "1" bit) to acknowledge the start bit.
Символ **Break** интерпертируется при приеме как нули '0' на протяжении всего кадра. В конце **Break** кадра передатчик вставляет 1 или 2 стоп-бита (логические '1'), чтобы правильно интерпретировать последующий старт-бит.

Transmission and reception are driven by a common baud rate generator, the clock for each is generated when the enable bit is set respectively for the transmitter and receiver.
Передача и прием управляются **baud rate** генератором с одной из общеприменимых скоростей обмена. Тактовый сигнал для каждого из них вырабатывается, если установлен соответствующий бит разрешения.

The details of each block is given below. (Детали каждого блока даны ниже.)

Figure 239. Word length programming (Программирование длины слова)



25.3.2 Transmitter (Передатчик)

Character transmission (Передача символа)
Configurable stop bits (Конфигурируемый стоп-бит)
Single byte communication (Обмен в режиме одиночного буфера)
Break characters (Символ **Break**)
Idle characters (Символ **Idle**)

The transmitter can send data words of either 8 or 9 bits depending on the M bit status. When the transmit enable bit (TE) is set, the data in the transmit shift register is output on the TX pin and the corresponding clock pulses are output on the SCLK pin. Передатчик может посылать слова данных длиной 8 или 9 бит, в зависимости от статуса битового поля **M**. Когда передача разрешена (поставлен бит **TE**), то данные, в сдвиговом регистре передатчика, выдвигаются на вывод **TX** и соответствующие тактовые импульсы выводятся на вывод **SCLK**.

Character transmission (Передача символа)

During an USART transmission, data shifts out least significant bit first on the TX pin. In this mode, the USART_DR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Figure 238*). В процессе передачи по **USART**, данные "выдвигаются" (*shifts out*) младшим битом вперед. В этом режиме, регистр **USART_DR** имеет буфер **TDR** между внутренней шиной и сдвиговым регистром передачи (см. рис. 238).

Every character is preceded by a start bit which is a logic level low for one bit period. The character is terminated by a configurable number of stop bits.
Каждому символу предшествует старт-бит, который имеет уровень логического нуля в течении одного бита. Символ завершается конфигурируемым числом стоп-бит.

The following stop bits are supported by USART: 0.5, 1, 1.5 and 2 stop bits.
Следующие стоп-биты поддерживаются **USART**: 0.5, 1, 1.5 и 2 стоп-бита.

***Note**:*

1. *The TE bit should not be reset during transmission of data. Resetting the TE bit during the transmission will corrupt the data on the TX pin as the baud rate counters will get frozen.*
   The current data being transmitted will be lost.
   *Бит **TE** (разрешение передатчика) не должен быть сброшен (переустановлен) в процессе передачи данных. Это вызовет разрушение данных на выводе **TX**, так как **baud rate** счетчик будет заморожен. Текущие данные будут утеряны.*

2. *An idle frame will be sent after the TE bit is enabled.*
   *После того, как будет поставлен бит **TE**, будет послан символ **Idle**.*

Configurable stop bits (Конфигурируемый стоп-бит)

The number of stop bits to be transmitted with every character can be programmed in Control register 2, bits 13,12.
Число стоп-битов, которые будут посланы с каждым символом, можно программировать в управляющем регистре (биты 13,12).

1. **1 stop bit**: This is the default value of number of stop bits.
   00 - 1 стоп-бит: Это значение по умолчанию.

2. **2 Stop bits**: This will be supported by normal USART, single-wire and modem modes.
   01 - 2 стоп-бита: Поддерживается нормальным **USART**, режимом **single-wire** и режимом **modem**.

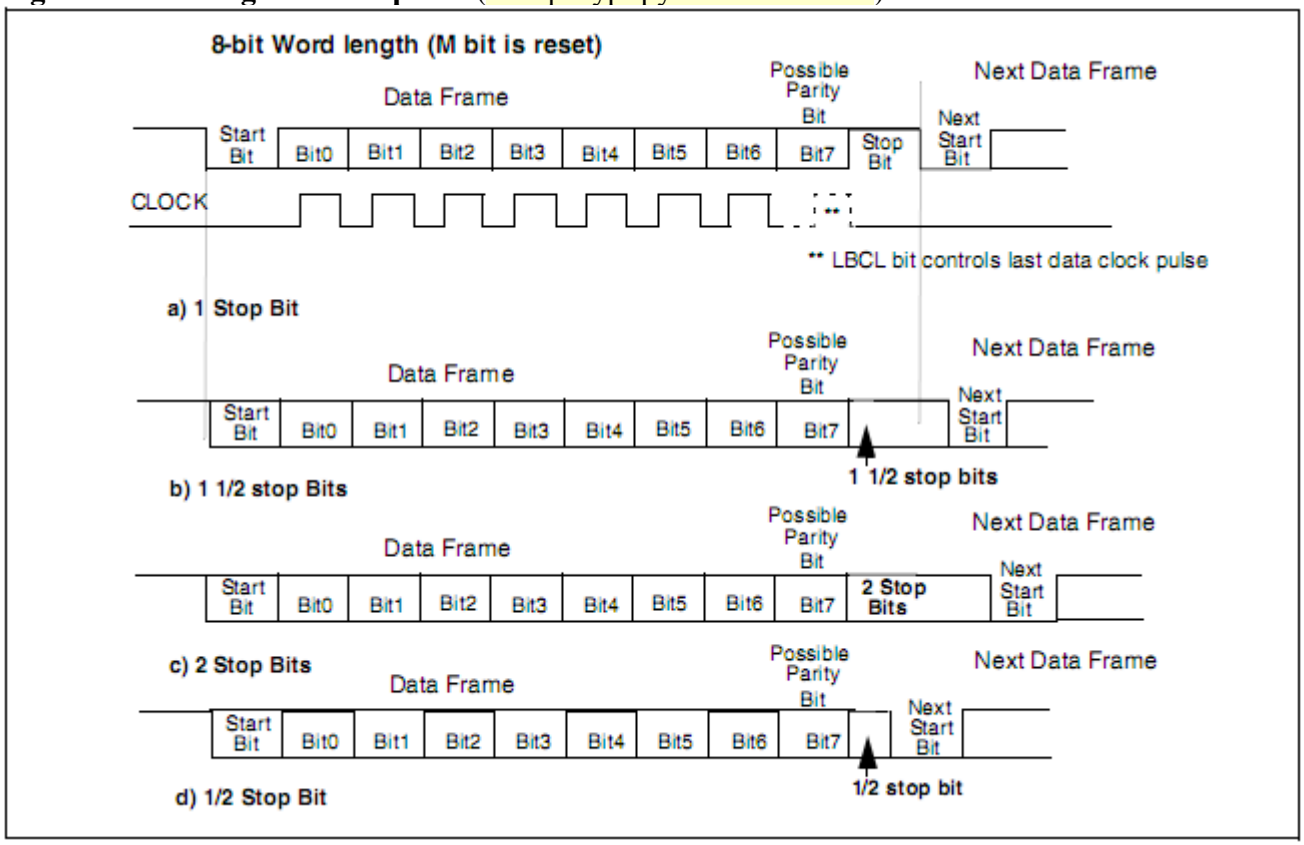
3. **0.5 stop bit**: To be used when receiving data in Smartcard mode.
10 - 0.5 стоп-бита: Используется при приеме данных в режиме **Smartcard**.

4. **1.5 stop bits**: To be used when transmitting and receiving data in Smartcard mode.
11 - 1.5 стоп-бита: Используется при передаче и приеме данных в режиме **Smartcard**.

An idle frame transmission will include the stop bits. (Кадр **Idle** включает в себя стоп-биты.)

A break transmission will be 10 low bits followed by the configured number of stop bits (when m = 0) and 11 low bits followed by the configured number of stop bits (when m = 1). It is not possible to transmit long breaks (break of length greater than 10/11 low bits).
Кадр **Break** содержит 10 "нулей", с последующим заданным числом стоп-бит (когда **M = 0**), и он содержит 11 "нулей", с последующим заданным числом стоп-бит (когда **M = 1**). Нет возможности передать длинный **Break** (длиной более 10/11 **low bits**).

Figure 240. Configurable stop bits (Конфигурируемый стоп-бит)



Procedure: (Процедура:)

1. Enable the USART by writing the UE bit in USART_CR1 register to 1.
Разрешаем **USART** записав '1' в **UE** бит регистра **USART_CR1**.

2. Program the M bit in USART_CR1 to define the word length.
Программируем **M** бит регистра **USART_CR1**, определяем длину слова.

3. Program the number of stop bits in USART_CR2.
Программируем число стоп-бит в регистре **USART_CR2**.

4. Select DMA enable (DMAT) in USART_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in multibuffer communication.
Выбираем разрешение **DMA** (**DMAT**) в регистре **USART_CR3**, если имеет место многобуферный обмен. Конфигурируем **DMA** регистр, как описано для многобуферного обмена.

5. Select the desired baud rate using the USART_BRR register.
Выбираем желаемый **baud rate** с помощью регистра **USART_BRR**.

6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.
Ставим **TE** бит в регистре **USART_CR1**, чтобы послать **Idle** кадр, как первый элемент обмена.

7. Write the data to send in the USART_DR register (this clears the TXE bit). Repeat this for each data to be transmitted in case of single buffer.
Записать данные, которые надо переслать, в регистр **USART_DR** (это очистит **TXE** бит). Повторить это для каждого байта данных в случае одиночного буфера.

8. After writing the last data into the USART_DR register, wait until TC=1. This indicates that the transmission of the last frame is complete. This is required for instance when the USART is disabled or enters the Halt mode to avoid corrupting the last transmission.
После записи последнего байта данных в регистр **USART_DR**, ждем, пока бит **TC** не станет равным '1'. Это говорит о том, что последний кадр завершен. Это требуется в случае, когда **USART** надо отключить или перевести в режим **Halt**, чтобы избежать разрушения последних данных.

Single byte communication (Обмен в режиме одиночного буфера)

Clearing the TXE bit is always performed by a write to the data register.
Очистка **TXE** бита всегда выполняется записью в регистр данных.

The TXE bit is set by hardware and it indicates: (Бит **TXE** ставится аппаратно и показывает:)

- The data has been moved from TDR to the shift register and the data transmission has started. Что данные начали перемещаться из **TDR** в сдвиговый регистр, то есть началась передача данных.
- The TDR register is empty. (Что регистр **TDR** пуст.)
- The next data can be written in the USART_DR register without overwriting the previous data. Что можно записывать следующие данные в регистр **USART_DR** без риска перезаписать предыдущие данные.

This flag generates an interrupt if the TXEIE bit is set.
Этот флаг генерирует прерывание, если стоит бит **TXEIE**.

When a transmission is taking place, a write instruction to the USART_DR register stores the data in the TDR register and which is copied in the shift register at the end of the current transmission.
Когда идет процесс передачи, то инструкция записи в регистр **USART_DR** сохраняет данные в регистре **TDR** и они копируются в сдвиговый регистр по окончанию текущей передачи.

When no transmission is taking place, a write instruction to the USART_DR register places the data directly in the shift register, the data transmission starts, and the TXE bit is immediately set.
Когда нет процесса передачи, то инструкция записи в регистр **USART_DR** помещает данные прямо в сдвиговый регистр, стартует процесс передачи и немедленно ставится бит **TXE**.

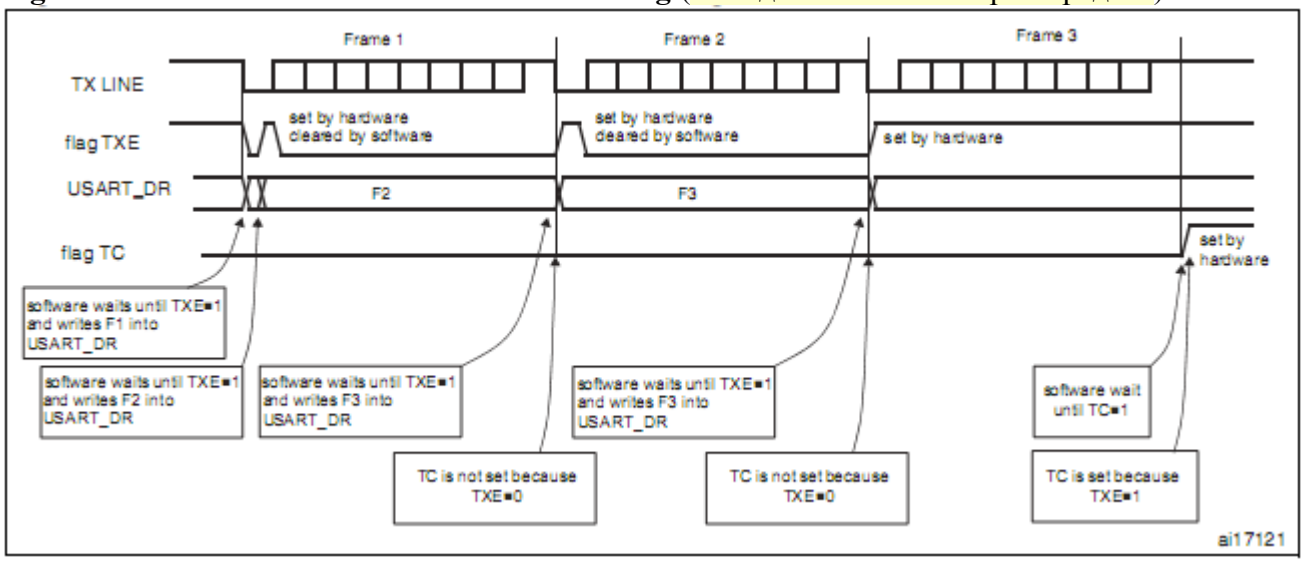
If a frame is transmitted (after the stop bit) and the TXE bit is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.
Когда кадр передан (после стоп-бита) и бит **TXE** поставлен, то бит **TC** переходит в высокое состояние. Если стоит бит **TCIE** в регистре **USART_CR1**, то генерирует прерывание.

After writing the last data into the USART_DR register, it is mandatory to wait for TC=1 before disabling the USART or causing the microcontroller to enter the low power mode (see *Figure 241: TC/TXE behavior when transmitting*).
После записи последнего байта данных в регистр **USART_DR**, обязательно подождите условия **TC=1**, перед тем, как отключить **USART** или перевести МК в режим малого потребления (см. рис. 241: Поведение **TC/TXE** при передаче).

Clearing the TC bit is performed by the following software sequence:
Очистка бита **TC** выполняется следующей программной процедурой:

1. A read from the USART_SR register (Чтение регистра **USART_SR**)
2. A write to the USART_DR register (Запись в регистр **USART_DR**)

Figure 241. TC/TXE behavior when transmitting (Поведение **TC/TXE** при передаче)



1. This example assumes that several other transmissions occured since TE was set. Otherwise, if USART_DR had been written for the first time, an IDLE preamble would have been transmitted first.
Этот пример предполагает, что уже были несколько других передач, вследствии чего был установлен бит **TE**. В противном случае, если регистр **USART_DR** был записан первый раз, то должна была сначала передана преамбула (кадр **IDLE**).

***Note**: The TC bit can also be cleared by writing a '0' to it. This clearing sequence is recommended only for Multibuffer communication.*
*Бит **TC** можно также очистить записью в него '0'. Но это рекомендуется только для многобуферного обмена.*

Break characters (Символ Break)

Setting the SBK bit transmits a break character. The break frame length depends on the M bit (see *Figure 239*).
Установка бита **SBK** вызывает передачу символа **Break**. Длина кадра **Break** зависит от бита **M** (см. рис. 239).

If the SBK bit is set to '1' a break character is sent on the TX line after completing the current character transmission. This bit is reset by hardware when the break character is completed (during the stop bit of the break character). The USART inserts a logic 1 bit at the end of the last break frame to guarantee the recognition of the start bit of the next frame.
Если поставить бит **SBK** в '1', то символ **Break** посылается на **TX** после завершения передачи

текущего символа. Этот бит сбрасывается аппаратно, когда завершается передача символа **Break** (в процессе передачи стоп-бита кадра **Break**). **USART** сам вставляет бит логической '1' в конец последнего кадра **Break**, чтобы гарантировать определение старт-бита следующего кадра.

***Note**: If the software resets the SBK bit before the commencement of break transmission, the break character will not be transmitted. For two consecutive breaks, the SBK bit should be set after the stop bit of the previous break.*
*Если программно сбросить бит **SBK** до того, как начнется передача кадра **Break**, то он не будет послан. Для второго, подряд идущего символа **Break**, бит **SBK** должен быть установлен после стоп-бита предыдущего кадра.*

Idle characters (Символ Idle)

Setting the TE bit drives the USART to send an idle frame before the first data frame.
Установка бита **TE** заставляет **USART** послать кадр символа **Idle** перед кадром первого байта данных.

25.3.3 Receiver (Приемник)

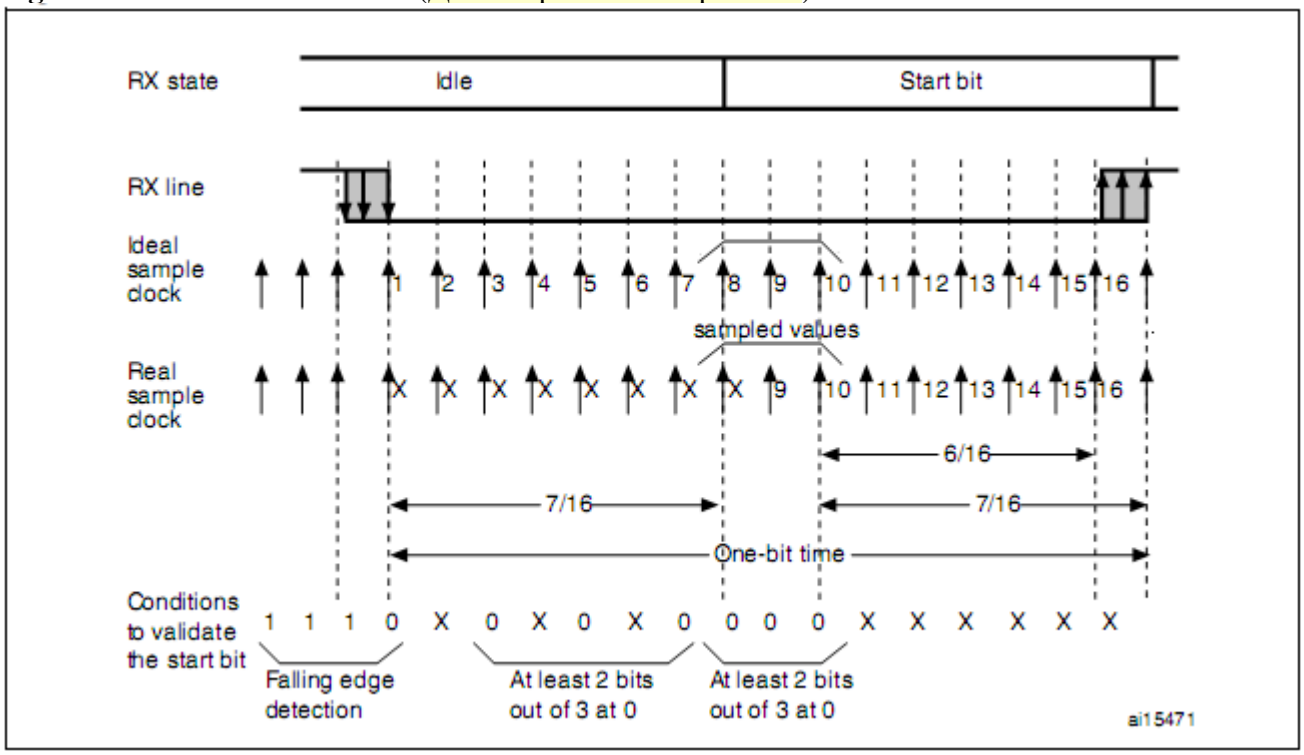
Start bit detection (Детектирование старт-бита)
Character reception (Прием символа)
Break character (Символ **Break**)
Idle character (Символ **Idle**)
Overrun error (Ошибка переполнения)
Noise error (Ошибка шума)
Framing error (Ошибка кадра)
Configurable stop bits during reception (Конфигурирование стоп-битов для приема)

The USART can receive data words of either 8 or 9 bits depending on the M bit in the USART_CR1 register.
USART может принять слово данных длиной 8 или 9 бит, в зависимости от наличия бита **M** в регистре **USART_CR1**.

Start bit detection (Детектирование старт-бита)

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.
USART детектирует старт-бит, когда, в процессе распознавания сэмплированных (захваченных) уровней, появляется специфическая последовательность: `1 1 1 0 x 0 x 0x 0x 0 x 0x 0.`

Figure 242. Start bit detection (Детектирование старт-бита)



***Note**: If the sequence is not complete, the start bit detection aborts and the receiver returns to idle state (no flag is set) waiting for a falling edge.*
Если последовательность не завершена, то процесс детектирования старт-бита прерывается и приемник возвращается в состояние покоя (на ставится никаких флагов) и ждет срез уровня.

If only 2 out of the 3 bits are at 0 (sampling on the 3-rd, 5-th and 7-th bits or sampling on the 8-th, 9-th and 10-th bits), the start bit is validated but the NE noise flag bit is set.
Если из трех бит только два являются нулем (например захват 3-го, 5-го и 7-го бита, или захват 8-го, 9-го и 10-го бита), то старт-бит детектируется, но ставится бит флага **NE** (наличие шума).

The start bit is confirmed if the last 3 samples are at 0 (sampling on the 8-th, 9-th, and 10-th bits.
Старт-бит подтверждается, если последние 3 сэмпла являются нулем (захват 8-го, 9-го и 10-го бита).

Character reception (Прием символа)

During an USART reception, data shifts in least significant bit first through the RX pin. In this mode, the USART_DR register consists of a buffer (RDR) between the internal bus and the received shift register.
В процессе приема на **RX**, **USART** "задвигает" данные младшим битом вперед. В этом режиме регистр **USART_DR** имеет буфер **RDR** между внутренней шиной и сдвиговым регистром приема.

Procedure: (Процедура:)

1. Enable the USART by writing the UE bit in USART_CR1 register to 1.
Разрешаем **USART**, записав '1' в **UE** бит регистра **USART_CR1**.

2. Program the M bit in USART_CR1 to define the word length.
Программируем **M** бит регистра **USART_CR1**, определяем длину слова.

3. Program the number of stop bits in USART_CR2.
Программируем число стоп-бит в регистре **USART_CR2**.

4. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in multibuffer communication. STEP 3
Выбираем разрешение **DMA** (**DMAT**) в регистре **USART_CR3**, если имеет место многобуферный обмен. Конфигурируем **DMA** регистр, как описано для многобуферного обмена.

5. Select the desired baud rate using the baud rate register USART_BRR
Выбираем желаемый **baud rate** с помощью регистра **USART_BRR**.

6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.
Ставим **RE** бит в регистре **USART_CR1**. Это разрешает приемник, который начинает поиск старт-бита.

When a character is received (Когда принимается символ)

- The RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
Ставится бит **RXNE**. Это говорит о том, что содержимое сдвигового регистра перемещено в буфер **RDR**. Другими словами, данные будут приняты и могут быть считаны (точно также, как и флаги ошибок, связанные с приемом).

- An interrupt is generated if the RXNEIE bit is set.
Когда ставится бит **RXNEIE**, генерируется прерывание.

- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.
Если в процессе приема детектируются ошибки Кадра, Шума или Переполнения, то ставятся соответствующие флаги.

- In multibuffer, RXNE is set after every byte received and is cleared by the DMA read to the Data Register.
Если используется мультибуферный прием, то бит **RXNE** ставится после приема каждого байта и очищается **DMA** при чтении им регистра данных.

- In single buffer mode, clearing the RXNE bit is performed by a software read to the USART_DR register. The RXNE flag can also be cleared by writing a zero to it. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
В режиме одиночного буфера очистка бита **RXNE** выполняется программно, чтением регистра **USART_DR**. Бит **RXNE** можно также очистить записью в него нуля. Этот бит должен быть очищен до конца приема следующего символа, чтобы избежать ошибки переполнения.

***Note**: The RE bit should not be reset while receiving data. If the RE bit is disabled during reception, the reception of the current byte will be aborted.*
*Бит **RE** (разрешение приемника) не должен быть сброшен в процессе приема данных. Если это сделать, то прием текущего символа будет прерван.*

Break character (Символ Break)

When a break character is received, the USART handles it as a framing error.
При приеме символа **Break**, **USART** обрабатывает его как ошибку кадра.

Idle character (Символ Idle)

When an idle frame is detected, there is the same procedure as a data received character plus an interrupt if the IDLEIE bit is set.
При обнаружении кадра **Idle**, следует та же процедура, что и при приеме символа данных, плюс генерируется прерывание, если стоит бит **IDLEIE**.

Overrun error (Ошибка Переполнения)

An overrun error occurs when a character is received when RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared.
Эта ошибка ставится, если принимается символ в то время, когда флаг **RXNE** не сброшен. Данные невозможно переместить из сдвигового регистра в буфер **RDR** до тех пор, пока не будет сброшен бит **RXNE**.

The RXNE flag is set after every byte received. An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:
Флаг **RXNE** ставится после приема каждого байта. Ошибка Переполнения ставится, если флаг **RXNE** ставится, когда получены следующие данные, или предыдущий запрос **DMA** не был обслужен. Когда случилась эта ошибка:

- The ORE bit is set. (Ставится бит **ORE**.)
- The RDR content will not be lost. The previous data is available when a read to USART_DR is performed.
  Содержимое **RDR** не будет потеряно. Предыдущие данные доступны, если прочитать **USART_DR**.
- The shift register will be overwritten. After that point, any data received during overrun is lost.
  Сдвиговый регистр не будет переписан. На этот момент, любые данные, полученные пока есть **overrun**, теряются.
- An interrupt is generated if either the RXNEIE bit is set or both the EIE and DMAR bits are set.
  Если поставлен бит **RXNEIE**, или поставлены оба бита, **EIE** и **DMAR**, то генерируется прерывание.
- The ORE bit is reset by a read to the USART_SR register followed by a USART_DR register read operation.
  Бит **ORE** сбрасывается чтением регистра **USART_SR** с последующим чтением регистра **USART_DR**.

***Note**: The ORE bit, when set, indicates that at least 1 data has been lost. There are two possibilities:*
*Поставленный бит **ORE** показывает, что по крайней мере 1 байт данных был потерян. Есть два варианта:*

- *if RXNE=1, then the last valid data is stored in the receive register RDR and can be read,*
  *если **RXNE=1**, то последние правильные данные хранятся в регистре приемника **RDR** и их можно прочитать,*
- *if RXNE=0, then it means that the last valid data has already been read and thus there is nothing to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received. It may also occur when the new data is received during the reading sequence (between the USART_SR register read access and the USART_DR read access).*

*если **RXNE=0**, то последние правильные данные уже были считаны, и, таким образом, в регистре **RDR** нечего читать. Этот случай может произойти, когда последние правильные данные читаются в **RDR** в то же время, как принимаются (и теряются) новые данные. Это может также случиться, когда новые данные принимаются в процессе последовательного чтения (между чтением регистра **USART_SR** и чтением регистра **USART_DR**).*

Noise error (Ошибка Шума)

Over-sampling techniques are used (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise.
Для распознавания данных используется техника передискретизации (кроме синхронного режима) за счет дискриминации корректных данных от шумы.

Figure 243. Data sampling for noise detection
Сэмплирование данных для детектирования шума

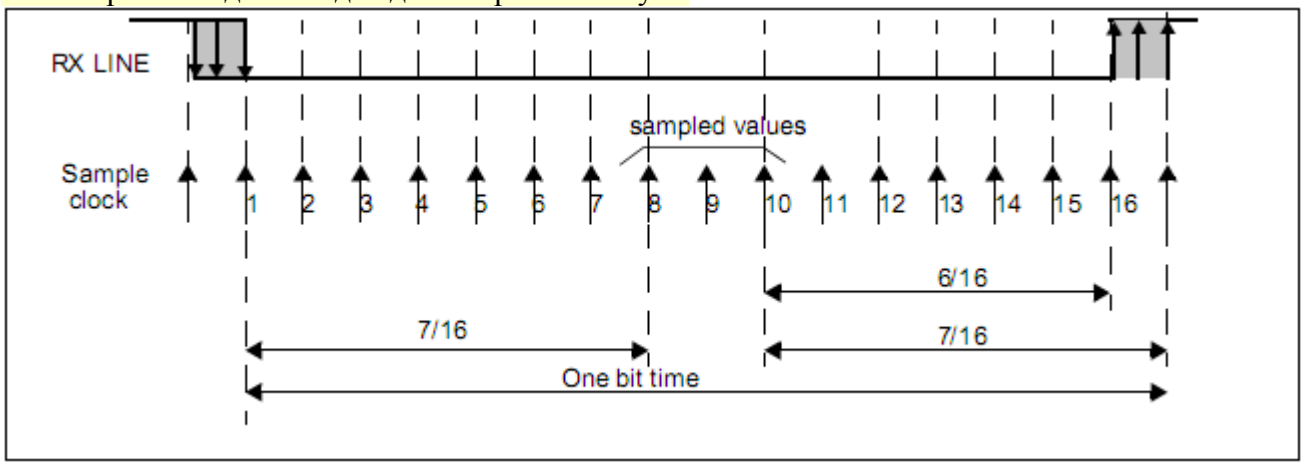


Table 173. Noise detection from sampled data
Детектирования шума в сэмплированных данных

Sampled value	NE status	Received bit value	Data validity
000	0	0	Valid
001	1	0	Not Valid
010	1	0	Not Valid
011	1	1	Not Valid
100	1	0	Not Valid
101	1	1	Not Valid
110	1	1	Not Valid
111	0	1	Valid

When noise is detected in a frame: (Когда в кадре обнаружен шум:)

- The NE is set at the rising edge of the RXNE bit. (Ставится бит **NE** по фронту бита **RXNE**.)
- The invalid data is transferred from the Shift register to the USART_DR register.
Некорректные данные перемещаются из сдвигового регистра в **USART_DR**.

- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USART_CR3 register.
В случае однобайтного обмена прерывание не генерируется. Однако этот бит ставится в то же время, что и бит **RXNE**, который сам по себе генерирует прерывание. В случае многобуферного обмена прерывание будет запрошено, если в регистре **USART_CR3** стоит бит **EIE**.

The NE bit is reset by a USART_SR register read operation followed by a USART_DR register read operation.
Бит **NE** сбрасывается чтением регистра **USART_SR** с последующим чтением регистра **USART_DR**.

Framing error (Ошибка Кадра)

A framing error is detected when: (Ошибка Кадра обнаруживается, когда:)

- The stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.
При приеме, за ожидаемое время, не обнаружен стоп-бит, с последующей рассинхронизацией, либо черезмерным шумом.

When the framing error is detected: (Когда обнаружена ошибка Кадра:)

- The FE bit is set by hardware (Бит **FE** ставится аппаратно)
- The invalid data is transferred from the Shift register to the USART_DR register.
Некорректные данные перемещаются из сдвигового регистра в **USART_DR**.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USART_CR3 register.
В случае однобайтного обмена прерывание не генерируется. Однако этот бит ставится в то же время, что и бит **RXNE**, который сам по себе генерирует прерывание. В случае многобуферного обмена прерывание будет запрошено, если в регистре **USART_CR3** стоит бит **EIE**.

The FE bit is reset by a USART_SR register read operation followed by a USART_DR register read operation. Бит **FE** сбрасывается чтением регистра **USART_SR** с последующим чтением регистра **USART_DR**.

Configurable stop bits during reception (Конфигурирование стоп-битов для приема)

The number of stop bits to be received can be configured through the control bits of Control Register 2 - it can be either 1 or 2 in normal mode and 0.5 or 1.5 in Smartcard mode.
Число стоп-бит, которое будет принято, можно конфигурировать в **USART_CR2**. Оно может быть 1 или 2 в нормальном режиме и 0.5 или 1.5 в режиме **Smartcard**.

1. ***0.5 stop bit (reception in Smartcard mode)***: No sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
0.5 стоп-бита (прием в режиме **Smartcard**): Никакого сэмплирования в этом случае не производится. Как следствие, никакой ошибки кадра и **Break** кадра не может быть обнаружено при таком выборе.

2. ***1 stop bit***: Sampling for 1 stop Bit is done on the 8th, 9th and 10th samples.
1 стоп-бит: Захват при 1 стоп-бите выполняется на 8-м, 9-м и 10-м сэмпле.

3. ***1.5 stop bits (Smartcard mode)***: When transmitting in smartcard mode, the device must check that the data is correctly sent. Thus the receiver block must be enabled (RE=1 in the USART_CR1 register) and the stop bit is checked to test if the smartcard has detected a parity error. In the event of a parity error, the smartcard forces the data signal low during the sampling -NACK signal-, which is flagged as a framing error. Then, the FE flag is set with the RXNE at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16-th, 17-th and 18-th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be decomposed into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through. Refer to *Section 25.3.11: Smartcard on page 675* for more details.
1.5 стоп-бита (режим **Smartcard**): Когда идет передача в режиме **Smartcard**, устройство должно проверять, что данные переданы корректно. Таким образом модуль приема должен быть разрешен (**RE=1** в регистре **USART_CR1**) и стоп-бит проверяется только на наличие ошибки паритета. В случае такой ошибки, **Smartcard** переводит линию данных в низкое состояние в процессе сэмплирования (сигнал **NACK**), что определяется как ошибка кадра. Ставится флаги **FE** и **RXNE** в конце 1.5 стоп-бита. Захват для 1.5 стоп-бита выполняется на 16-м, 17-и и 18-м сэмпле (1 период **baud clock** после начала стоп-бита). 1.5 стоп-бита можно разложить на две части: одну в 0.5 периода **baud clock**, в течении которого ничего не делается, с последующим периодом в 1 нормальный стоп-бит, в течении которого производится сэмплирование. См. раздел 25.3.11 "Smartcard".

4. ***2 stop bits***: Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit. If a framing error is detected during the first stop bit the framing error flag will be set. The second stop bit is not checked for framing error. The RXNE flag will be set at the end of the first stop bit.
2 стоп-бита: Захват для 2 стоп-битов выполняется на 8-м, 9-м и 10-м сэмпле первого стоп-бита. Если обнаруживается ошибка кадра в процессе приема первого стоп-бита, то будет выставлен соответствующий флаг. Второй стоп-бит не проверяется на наличие ошибки кадра. Флаг **RXNE** будет выставлен в конце первого стоп-бита.

25.3.4 Fractional baud rate generation (Фракционный baud rate генератор)

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the same value as programmed in the Mantissa and Fraction values of USARTDIV.
Скорость обмена для приемника и передатчика (**Rx** и **Tx**) имеет одно значение, что задано основной и дробной частью регистра **USART_BRR**.

```
Tx/ Rx baud = fCK / (16*USARTDIV)
```

legend: fCK - Input clock to the peripheral (PCLK1 for USART2, 3, 4, 5 or PCLK2 for USART1)
где: **fCK** - входной тактовый сигнал периферии (**PCLK1** для **USART2**, **3**, **4**, **5** или **PCLK2** для **USART1**)

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.
USARTDIV - это беззнаковое число с фиксированной точкой, которым закодирован регистр **USART_BRR**.

***Note**: The baud counters are updated with the new value of the Baud registers after a write to USART_BRR. Hence the Baud rate register value should not be changed during communication.*
*Счетчики **baud** обновляются новым значением скорости обмена из регистра после записи в **USART_BRR**. Следовательно, значение этого регистра не должно меняться в процессе обмена.*

How to derive USARTDIV from USART_BRR register values
Как получить USARTDIV из значения регистра USART_BRR

Example 1:

if (если) `DIV_Mantissa = 27 and (и) DIV_Fraction = 12 (USART_BRR = 0x1BC)`, then: (то:)
`Mantissa (USARTDIV) = 27`
`Fraction (USARTDIV) = 12/16 = 0.75`
Therefore: (Тогда:) `USARTDIV = 27.75`

Example 2:

To program (Чтобы запрограммировать) `USARTDIV = 25.62`
This leads to: (надо:)
`DIV_Fraction = 16*0.62 = 9.92`
The nearest real number is (Ближайшее целое) `0d10 = 0x0A`
`DIV_Mantissa = mantissa (25.620) = 25 = 0x19`
Then, (Тогда:) `USART_BRR = 0x19A`, hence (следовательно) `USARTDIV = 25.625`

Example 3:

To program (Чтобы запрограммировать) `USARTDIV = 50.99`
This leads to: (надо:)
`DIV_Fraction = 16*0.99 = 15.84`
The nearest real number is 0d16 = 0x10 => overflow of DIV_frac[3:0] => carry must be added up to the mantissa Ближайшее целое `= 16 = 0x10 =>` переполнение `DIV_frac[3:0] =>` должен быть выполнен перенос в мантиссу
`DIV_Mantissa = mantissa (50.990 + carry) = 51 = 0x33`
Then, (Тогда:) `USART_BRR = 0x330`, hence (следовательно) `USARTDIV = 51.000`

Table 174. Error calculation for programmed baud rates
Расчет ошибки установки частоты **baud rate** генератора

Baud rate		**fPCLK = 36 MHz**			**fPCLK = 72 MHz**		
S.No	**in Kbps**	**Actual**	**Value programmed in the Baud Rate register**	**% Error =(Calculated - Desired)B.Rate /Desired B.Rate**	**Actual**	**Value programmed in the Baud Rate register**	**% Error**
1.	2.4	2.400	937.5	0%	2.4	1875	0%
2.	9.6	9.600	234.375	0%	9.6	468.75	0%
3.	19.2	19.2	117.1875	0%	19.2	234.375	0%
4.	57.6	57.6	39.0625	0%	57.6	78.125	0%
5.	115.2	115.384	19.5	0.15%	115.2	39.0625	0%
6.	230.4	230.769	9.75	0.16%	230.769	19.5	0.16%
7.	460.8	461.538	4.875	0.16%	461.538	9.75	0.16%
8.	921.6	923.076	2.43755	0.16%	923.076	4.875	0.16%
9.	2250	2250	1	0%	2250	2	0%
10.	4500	NA	NA	0%	NA	4500	0%

***Note**:*

1. *The lower the CPU clock the lower will be the accuracy for a particular Baud rate. The upper limit of the achievable baud rate can be fixed with this data.*
   *При снижении частоты **CPU** получаем меньшую точность выставления нужной **Baud rate**. Максимально достижимый **baud rate** приведен в этой таблице.*

2. *Only USART1 is clocked with PCLK2 (72 MHz Max). Other USARTs are clocked with PCLK1 (36 MHz Max). Частоту тактирования **PCLK2** (72 **MHz Max**) можно давать только **USART1**. Другие **USART** тактируются от **PCLK1** (36 **MHz Max**).*

25.3.5 USART receiver's tolerance to clock deviation
Терпимость приемника USART к девиации тактовой частоты

The USART's asynchronous receiver works correctly only if the total clock system deviation is smaller than the USART receiver's tolerance. The causes which contribute to the total deviation are:
Асинхронный приемник **USART** работает корректно только тогда, когда суммарная девиация тактирующей системы меньше, чем терпимость приемника **USART**. Причины, влияющие на суммарную девиацию:

- **DTRA**: Deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator) Девиация вследствии ошибки передатчика (которая включает также девиацию локального генератора передатчика)

- **DQUANT**: Error due to the baud rate quantization of the receiver
Ошибки вследствии квантования **baud rate** приемника

- **DREC**: Deviation of the receiver's local oscillator
Девиация локального генератора приемника

- **DTCL**: Deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing) Девиация вследствии качества линии обмена (как правило, вследствии такой линии, которая вносит ассиметрию между временем фронта и спада сигнала)

DTRA + DQUANT + DREC + DTCL < receiver's tolerance (терпимости приемника)

The USART receiver's tolerance to properly receive data is equal to the maximum tolerated deviation and depends on the following choices:
Терпимость приемника **USART** к конкретным принимаемым данным равна максимальной терпимости к девиации и зависит от следующих причин:

- 10- or 11-bit character length defined by the M bit in the USART_CR1 register
длина кадра 10 или 11 бит, в зависимости от бита **M** в регистре **USART_CR1**

- use of fractional baud rate or not
использования целого значения частоты или фракционального

Table 175. USART receiver 's tolerance when DIV_Fraction is 0
Терпимость приемника **USART**, когда дробная часть равна 0

Бит M	NF is an error	NF is don't care
0	3.75%	4.375%
1	3.41%	3.97%

Table 176. USART receiver's tolerance when DIV_Fraction is different from 0
Терпимость приемника **USART**, когда дробная часть отлична от 0

Бит M	NF is an error	NF is don't care
0	3.33%	3.88%
1	3.03%	3.53%

***Note**: The figures specified in Table 175 and Table 176 may slighly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M=0 (11-bit times when M=1).*
*Значения, указанные в табл. 175 и 176 могут несколько отличаться в особых случаях, когда принимаемые кадры содержат несколько **Idle** кадров, длительностью точно 10 бит при **M=0** (или 11 бит при **M=1**).*

25.3.6 Multiprocessor communication (Многопроцессорный обмен)

Idle line detection (WAKE=0) (Определение **Idle** кадра (**WAKE=0**))
Address mark detection (WAKE=1) (Обнаружение адресного маркера (**WAKE=1**))

There is a possibility of performing multiprocessor communication with the USART (several USARTs connected in a network). For instance one of the USARTs can be the master, its TX output is connected to the RX input of the other USART. The others are slaves, their respective TX outputs are logically ANDed together and connected to the RX input of the master.
Есть возможность осуществлять многопроцессорный обмен через **USART** (несколько **USART** подключены к локальной сети). Например, один **USART** может быть "Ведущим" (мастером), его **TX** выход подключен к **RX** входам других **USART**. Остальные "Ведомые", их выходы **TX** логически объединены вместе и подключены к **RX** входу "Ведущего".

In multiprocessor configurations it is often desirable that only the intended message recipient should actively receive the full message contents, thus reducing redundant USART service overhead for all non addressed receivers. В многопроцессорной конфигурации часто желательно, чтобы только те сообщения, что назначены адресату, активировали приемник на прием полного сообщения, уменьшая таким образом, чрезмерное обслуживание служебной информации для всех неадресуемых **USART** приемников.

The non addressed devices may be placed in mute mode by means of the muting function. Неадресуемые устройства могут быть помещены в немой режим (**mute**).

In mute mode: (В режиме **mute**:)

- None of the reception status bits can be set.
Ни один из статусных битов приема не может быть поставлен.
- All the receive interrupts are inhibited. (Все прерывания приема подавлены.)
- The RWU bit in USART_CR1 register is set to 1. RWU can be controlled automatically by hardware or written by the software under certain conditions.
Бит **RWU** регистра **USART_CR1** ставится в 1. Битом **RWU** можно управлять автоматически, аппаратно, или записывать программно, при определенных условиях.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:
USART может входить в режим **mute** или выходить из него, используя один из двух методов, в зависимости от бита **WAKE** в регистре **USART_CR1**:

- Idle Line detection if the WAKE bit is reset,

Если **WAKE** бит сброшен, то выход при определении **Idle** кадра,

- Address Mark detection if the WAKE bit is set.

Если **WAKE** бит установлен, то выход при определении адресного маркера (**Address Mark**).

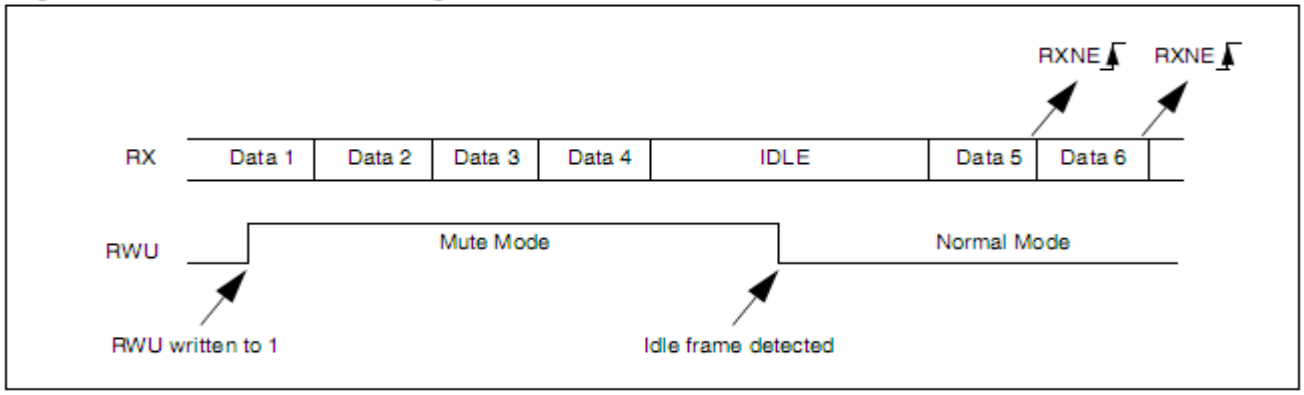
Idle line detection (WAKE=0) (Определение Idle кадра (WAKE=0))

The USART enters mute mode when the RWU bit is written to 1.
USART входит в режим **mute**, когда в бит **RWU** записывается 1.

It wakes up when an Idle frame is detected. Then the RWU bit is cleared by hardware but the IDLE bit is not set in the USART_SR register. RWU can also be written to 0 by software.
Он просыпается, когда обнаруживается кадр **Idle**. Затем бит **RWU** очищается аппаратно, но бит **IDLE** не ставится в регистре **USART_SR**. В бит **RWU** можно также записать 0 программно.

An example of mute mode behavior using idle line detection is given in *Figure 244*.
Пример поведения в режиме **mute** при обнаружении **Idle** кадра дается на рис. 244.

Figure 244. Mute mode using Idle line detection
Выход из **Mute** режима при обнаружении **Idle** кадра



Address mark detection (WAKE=1) Обнаружение адресного маркера (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a ‘1’ else they are considered as data. In an address byte, the address of the targeted receiver is put on the 4 LSB. This 4-bit word is compared by the receiver with its own address which is rogrammed in the ADD bits in the USART_CR2 register.
В этом режиме, байт распознается как адрес, если его старший бит равен 1, иначе байт считается данными. В адресном байте, адрес целевого приемника размещается в 4-х младших битах. Это 4-х битное слово сравнивается приемником с его собственным адресом, который программируется в битах поля **ADD** регистра **USART_CR2**.

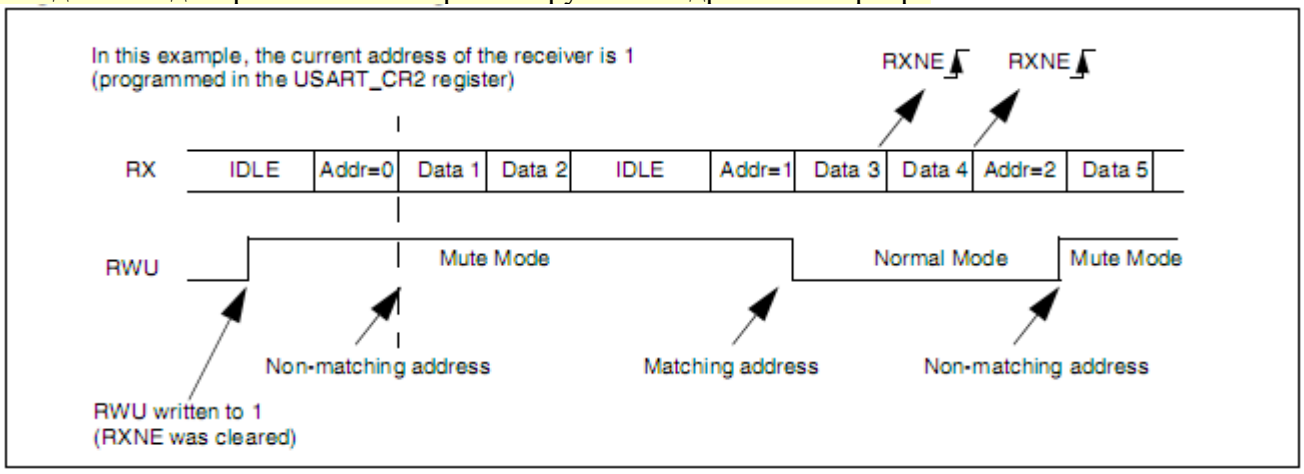
The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt nor DMA request is issued as the USART would have entered mute mode.
USART входит в режим **mute**, когда получен адресный байт, который не соответствует запрограммированному адресу. В этом случае, бит **RWU** ставится аппаратно. Флаг **RXNE** не ставится для этого адресного байта и никаких запросов, ни на прерывание, ни на обслуживание **DMA**, не выполняется, так как **USART** уже должен войти в режим **mute**.

It exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE bit is set for the address character since the RWU bit has been cleared.
Он выходит из режима **mute**, когда получен адресный байт, который соответствует запрограммированному адресу. Затем бит **RWU** очищается и последующие байты принимаются нормально. Бит **RXNE** ставится для адресного байта, так как бит **RWU** уже очищен.

The RWU bit can be written to as 0 or 1 when the receiver buffer contains no data (RXNE=0 in the USART_SR register). Otherwise the write attempt is ignored.
В бит **RWU** можно записать 0 или 1, когда буфер приема не содержит данных (**RXNE=0** в регистре **USART_SR**). В противном случае попытка записи будет проигнорирована.

An example of mute mode behavior using address mark detection is given in *Figure 245*.
Пример поведения в режиме **mute** при обнаружении адресного маркера дается на рис. 245.

Figure 245. Mute mode using Address mark detection
Вход и выход из режима **Mute** при обнаружении адресного маркера



25.3.7 Parity control (Контроль паритета)

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bit, the possible USART frame formats are as listed in *Table 177*.
Контроль паритета (генерация бита паритета передатчиком и проверка паритета приемником) может быть разрешен установкой бита **PCE** в регистре **USART_CR1**. В табл. 177 даны возможные форматы **USART** кадра в зависимости от длины кадра, определенного битом **M**.

Table 177. Frame formats(1) (Формат кадра)

M bit	PCE bit	USART frame
0	0	| SB | 8 bit data | STB |
0	1	| SB | 7-bit data | PB | STB |
1	0	| SB | 9-bit data | STB |
1	1	| SB | 8-bit data PB | STB |

1. Legends: SB: Start Bit, STB: Stop Bit, PB: Parity Bit

***Note**: n case of wake up by an address mark, the MSB bit of the data is taken into account and not the parity bit*
В случае просыпания от адресного маркера, старший бит данных берется во внимание, но не обсчитывается на паритет.

Even parity: the parity bit is calculated to obtain an even number of "1s" inside the frame made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.
Четный паритет: бит паритета расчитывается, чтобы получить четное число "единиц" в кадре, состоящего из младших 7 или 8 бит (в зависимости от бита **M**) и бита паритета.

Ex: data=00110101; 4 bits set => parity bit will be 0 if even parity is selected (PS bit in USART_CR1 = 0). **Пример**: `data=00110101`; установлено 4 бита **=>** бит паритета должен быть 0, если выбран четный паритет (бит **PS** в регистре **USART_CR1 = 0**).

Odd parity: the parity bit is calculated to obtain an odd number of "1s" inside the frame made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.
Нечетный паритет: бит паритета расчитывается, чтобы получить нечетное число "единиц" в кадре, состоящего из младших 7 или 8 бит (в зависимости от бита **M**) и бита паритета.

Ex: data=00110101; 4 bits set => parity bit will be 1 if odd parity is selected (PS bit in USART_CR1 = 1). **Пример**: `data=00110101`; установлено 4 бита **=>** бит паритета должен быть 1, если выбран нечетный паритет (бит **PS** в регистре **USART_CR1 = 1**).

Transmission mode: If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of "1s" if even parity is selected (PS=0) or an odd number of "1s" if odd parity is selected (PS=1)). If the parity check fails, the PE flag is set in the USART_SR register and an interrupt is generated if PEIE is set in the USART_CR1 register.
Режим обмена: Если установлен бит **PCE** в регистре **USART_CR1**, то старший бит данных в регистре данных - это бит обмена и он меняется битом паритета (четное число "единиц", если выбран четный паритет (**PS=0**), или нечетное число "единиц", если выбран нечетный паритет (**PS=1**)). Если проверка паритета не прошла, ставится **PE** флаг в регистре **USART_SR** и генерируется прерывание, если есть бит **PEIE** в регистре **USART_CR1**.

25.3.8 LIN (local interconnection network) mode
Режим LIN (локальная сеть)

LIN transmission (Передача в режиме **LIN**)
LIN reception (Прием в режиме **LIN**)

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:
Режим **LIN** выбирается установкой бита **LINEN** в регистре **USART_CR2**. В этом режиме следующие биты должны удерживаться чистыми:

- CLKEN in the USART_CR2 register, (бит **CLKEN** в регистре **USART_CR2**,)
- STOP[1:0], SCEN, HDSEL and IREN in the USART_CR3 register.

биты **STOP[1:0]**, **SCEN**, **HDSEL**, **IREN** в регистре **USART_CR3**.

LIN transmission (Передача в режиме LIN)

The same procedure explained in Section 25.3.2 has to be applied for LIN Master transmission than for normal USART transmission with the following differences:
Процедуры, описанные в разделе 25.3.2 для нормальной **USART** передачи, должны выполняться при передаче **LIN** мастера, но со следующими отличиями:

- Clear the M bit to configure 8-bit word length.
Очистите бит **M**, чтобы задать режим 8-ми битного слова.

- Set the LINEN bit to enter LIN mode. In this case, setting the SBK bit sends 13 ‘0’ bits as a break character. Then a bit of value ‘1’ is sent to allow the next start detection.
Установите бит **LINEN**, чтобы войти в режим **LIN**. В этом случае, установка бита **SBK** пошлет 13 "нулей" в качестве **Break** символа. Затем будет послан бит '1', что позволит определить следующий старт бит.

LIN reception (Прием в режиме LIN)

When the LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during idle state or during a frame.
Когда разрешен режим **LIN**, активирована цепь детектирования символа **Break**. Этот детектор полностью независим от нормального **USART** приемника. Символ **Break** можно обнаружить, когда бы он не случился, в состоянии **Idle** или в процессе приема кадра.

When the receiver is enabled (RE=1 in USART_CR1), the circuit looks at the RX input for astart signal. The method for detecting start bits is the same when searching break haracters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL=1 in USART_CR2) consecutive bits are detected as ‘0’, and are followed by a delimiter character, the LBD flag is set in USART_SR. If the LBDIE bit=1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.
Когда приемник включен (**RE=1** в **USART_CR1**), эта цепь следит за входом **RX**, чтобы засечь сигнал **start**. Метод обнаружения старт-бита точно такой же, что и для поиска **Break** символа или байта данных. После того, как старт-бит обнаружен, цепь сэмплирует последующие биты точно также, как для данных (на 8-м, 9-м и 10-м сэмпле). Если 10 (когда **LBDL=0** в **USART_CR2**) или 11 (когда **LBDL=1** в **USART_CR2**) последующих бит детектируются как '0', и затем следует разделитель, то ставится флаг **LBD** в **USART_SR**. Если **LBDIE=1**, то генерируется прерывание. Перед валидацией символа **Break**, проверяется наличие разделителя, то есть, что **RX** линия вернулась на высокий уровень.

If a ‘1’ is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.
Если захвачена '1' до 10-го или 11-го сэмпла, то цепь детектирования **Break** прекращает текущую работу и снава начинает поиск старт-бита.

If the LIN mode is disabled (LINEN=0), the receiver continues working as normal USART, without taking into account the break detection.
Если режим **LIN** отключить (**LINEN=0**), то приемник продолжает работу в нормальном **USART** режиме, без захода в детектирование **Break** символа.

If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (i.e. stop bit detected at ‘0’, which will be the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1’, if the break word was not complete, or a delimiter character if a break has been detected.

Если режим **LIN** включить (**LINEN=1**), как только случится ошибка кадра (то есть, детектируется стоп-бит '0', что будет в случае приема любого **Break** кадра), приемник остановится до тех пор, пока цепь детектирования **Break** не примет либо '1', если слово **Break** было некомплектным, либо разделитель, если был обнаружен **Break**.

The behavior of the break detector state machine and the break flag is shown on the *Figure 246: Break detection in LIN mode (11-bit break length - LBDL bit is set) on page 671*. Поведение детектора **Break** состояния и установка соответствующего флага показано на рис. 246.

Examples of break frames are given on Figure 247: *Break detection in LIN mode vs. Framing error detection on page 672*. Пример **Break** кадров дан на рис. 247.

Figure 246. Break detection in LIN mode (11-bit break length - LBDL bit is set)
Обнаружение **Break** в режиме **LIN** (длина **Break** 11-bit - есть бит **LBDL**)

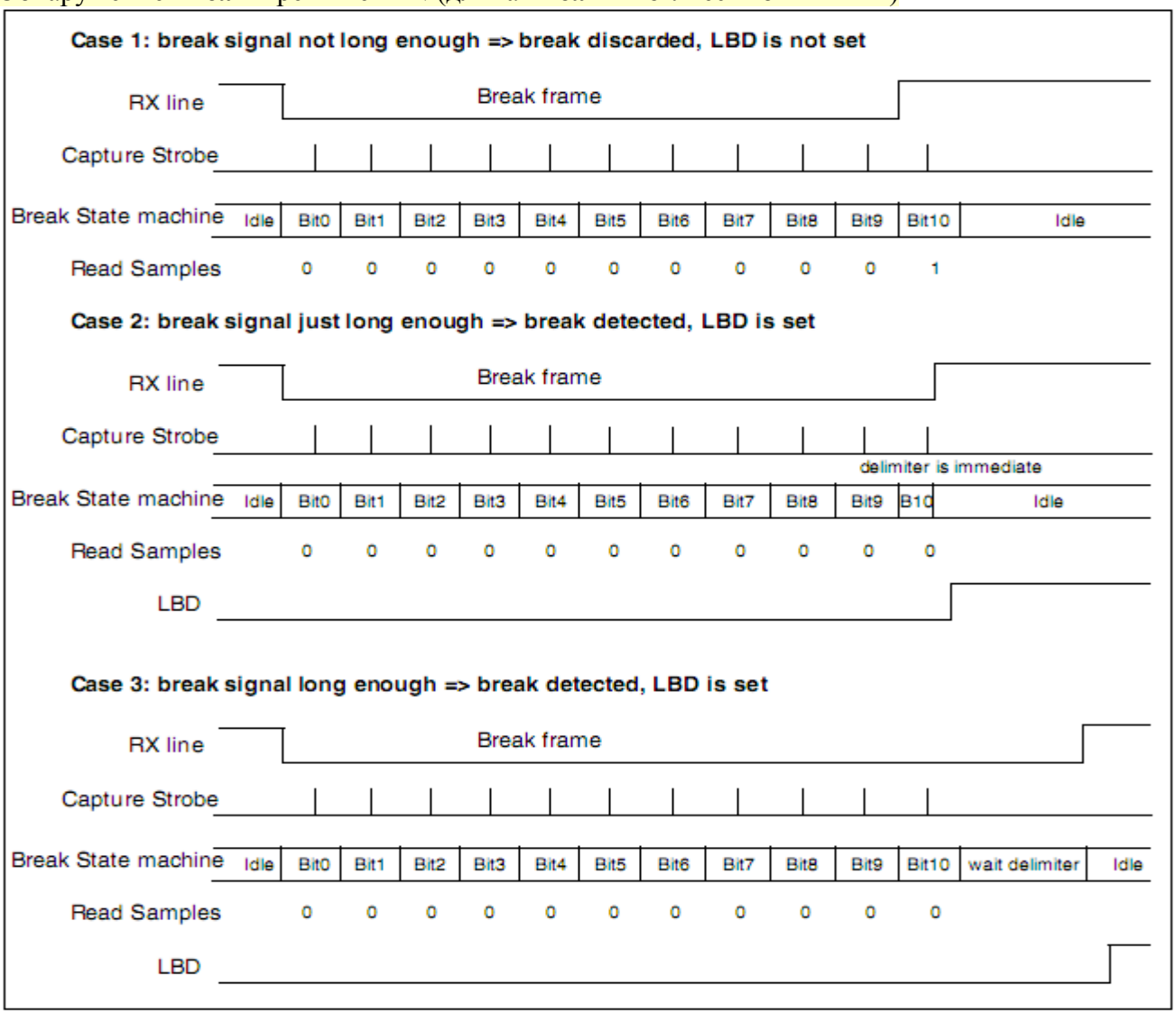
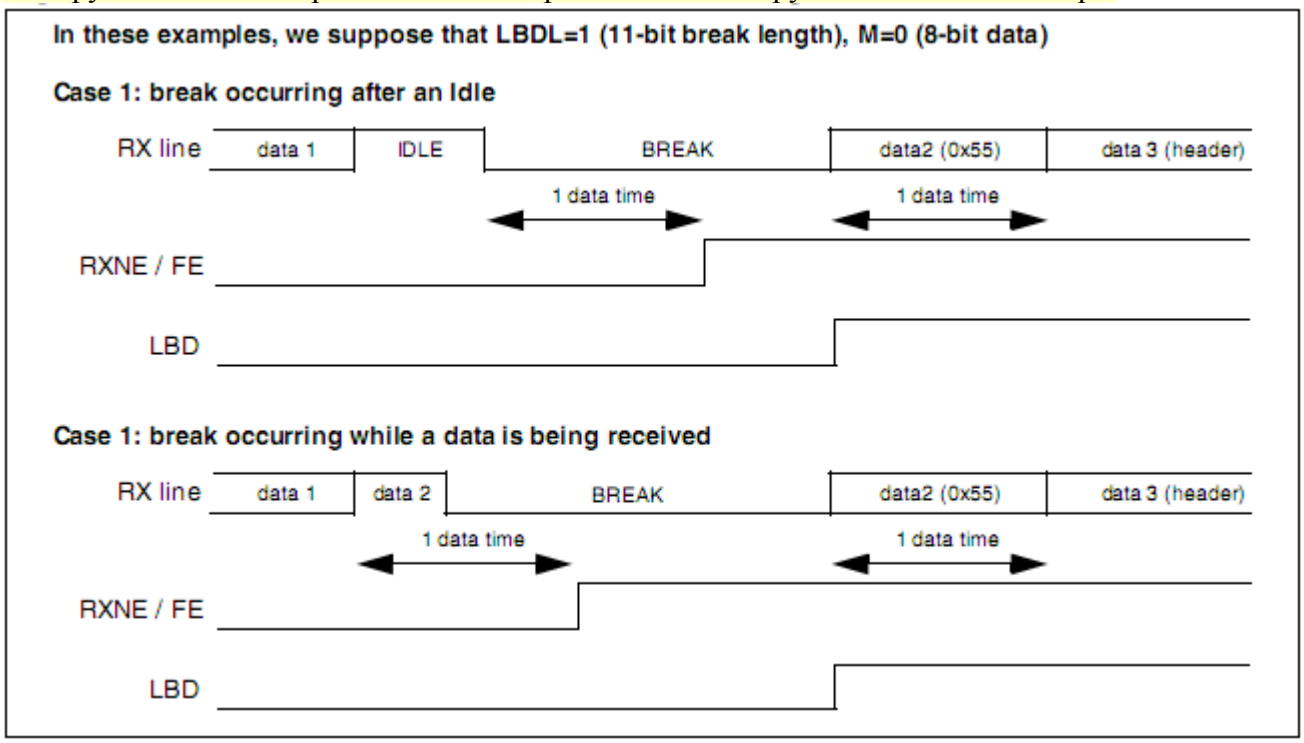


Figure 247. Break detection in LIN mode vs. Framing error detection
Обнаружение **Break** в режиме **LIN** по сравнению с обнаружением ошибки кадра.



25.3.9 USART synchronous mode (Синхронный режим USART)

The synchronous mode is selected by writing the CLKEN bit in the USART_CR2 register to 1. In synchronous mode, the following bits must be kept cleared:
Синхронный режим выбирается записью 1 в бит **CLKEN** в регистр **USART_CR2**. В синхронном режиме, следующие биты должны удерживаться очищенными:

- LINEN bit in the USART_CR2 register, (бит **LINEN** в регистре **USART_CR2**,)
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

биты **SCEN**, **HDSEL**, **IREN** в регистре **USART_CR3**.

The USART allows the user to control a bidirectional synchronous serial communications in master mode. The SCLK pin is the output of the USART transmitter clock. No clock pulses are sent to the SCLK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register clock pulses will or will not be generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register allows the user to select the clock polarity, and the CPHA bit in the USART_CR2 register allows the user to select the phase of the external clock (see *Figure 248*, *Figure 249* & *Figure 250*).
USART позволяет пользователю управлять двунаправленным синхронным последовательным обменом в режиме "Ведущий". Вывод **SCLK** - это выход тактового для **USART** передатчика. Во время передачи старт- и стоп-битов нет пульсаций тактового на выводе **SCLK**. В зависимости от состояния бита **LBCL** в регистре **USART_CR2**, тактовый импульс будет генерироваться, или отсутствовать в течении передачи последнего бита валидных данных (адресный маркер). Бит **CPOL** в регистре **USART_CR2** позволяет пользователю выбрать полярность тактового импульса, а бит **CPHA** в регистре **USART_CR2** позволяет выбрать фазу внешнего тактового импульса (см. рис. 248, 249 и 250).

During idle, preamble and send break, the external SCLK clock is not activated.
В процессе состояния **Idle** (покоя), преамбулы (кадра **Idle**) и символа **Break**, внешний **SCLK** исмпульс не активируется.

In synchronous mode the USART transmitter works exactly like in asynchronous mode. But as SCLK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.
В синхронном режиме **USART** передатчик работает точно также, как и в асинхронном режиме. Но, так как **SCLK** синхронизирован с **TX** (в соответствии с **CPOL** и **CPHA**), то данные на **TX** являются синхронными.

In this mode the USART receiver works in a different manner compared to the asynchronous mode. If RE=1, the data is sampled on SCLK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).
В этом режиме **USART** приемник работает в другом режиме, чем в асинхронном. Если **RE=1**, то данные сэмплируются по **SCLK** (по фронту или срезу, в зависимости от **CPOL** и **CPHA**), без какой либо дополнительной передискретизаци. Времени запуска (*setup*) и остановки (*hold*) приемника надо уделить должное внимание (оно зависит от **baud rate** и равно 1/16 периода бита).

***Note**:*

1. *The SCLK pin works in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE=1) and a data is being transmitted (the data register USART_DR has been written). This means that it is not possible to receive a synchronous data without transmitting data.*
   *Вывод **SCLK** работает в связке с выводом **TX**. Таким образом, тактовый выдается только тогда, когда передатчик включен (**TE=1**) и данные начали передаваться (регистр данных **USART_DR** был записан). Это означает, что невозможно принять синхронные данные без их передачи.*

2. *The LBCL, CPOL and CPHA bits have to be selected when both the transmitter and the receiver are disabled (TE=RE=0) to ensure that the clock pulses function correctly. These bits should not be changed while the transmitter or the receiver is enabled.*
   *Биты **LBCL**, **CPOL** и **CPHA** должны быть выбраны, пока оба, и приемник, и передатчик, отключены (**TE=RE=0**), чтобы гарантировать, что тактовый импульс будет функционировать корректно. Эти биты не должны меняться, когда приемник или передатчик включен.*

3. *It is advised that TE and RE are set in the same instruction in order to minimize the setup and the hold time of the receiver. Благоразумно устанавливать **TE** и **RE** одной инструкцией, чтобы минимизировать время запуска (setup) и остановки (hold) приемника.*

4. *The USART supports master mode only: it cannot receive or send data related to an input clock (SCLK is always an output).*
   *Синхронный **USART** поддерживает только режим **master**: он не может принять или послать данные относительно какого-то входного тактоваго сигнала (**SCLK** всегда является выходом).*

Figure 248. USART example of synchronous transmission
Пример синхронной передачи по **USART**

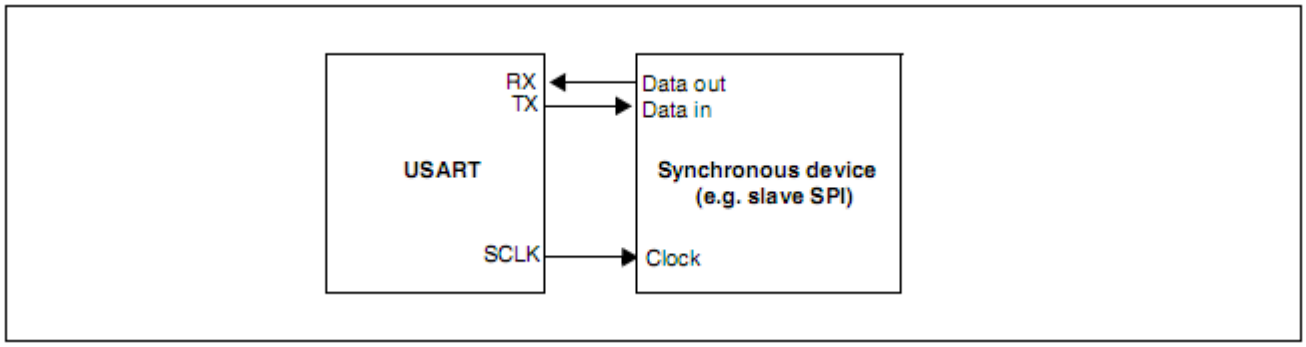


Figure 249. USART data clock timing diagram (M=0)
Временная диаграмма тактового сигнала **USART** (при **M=0**)

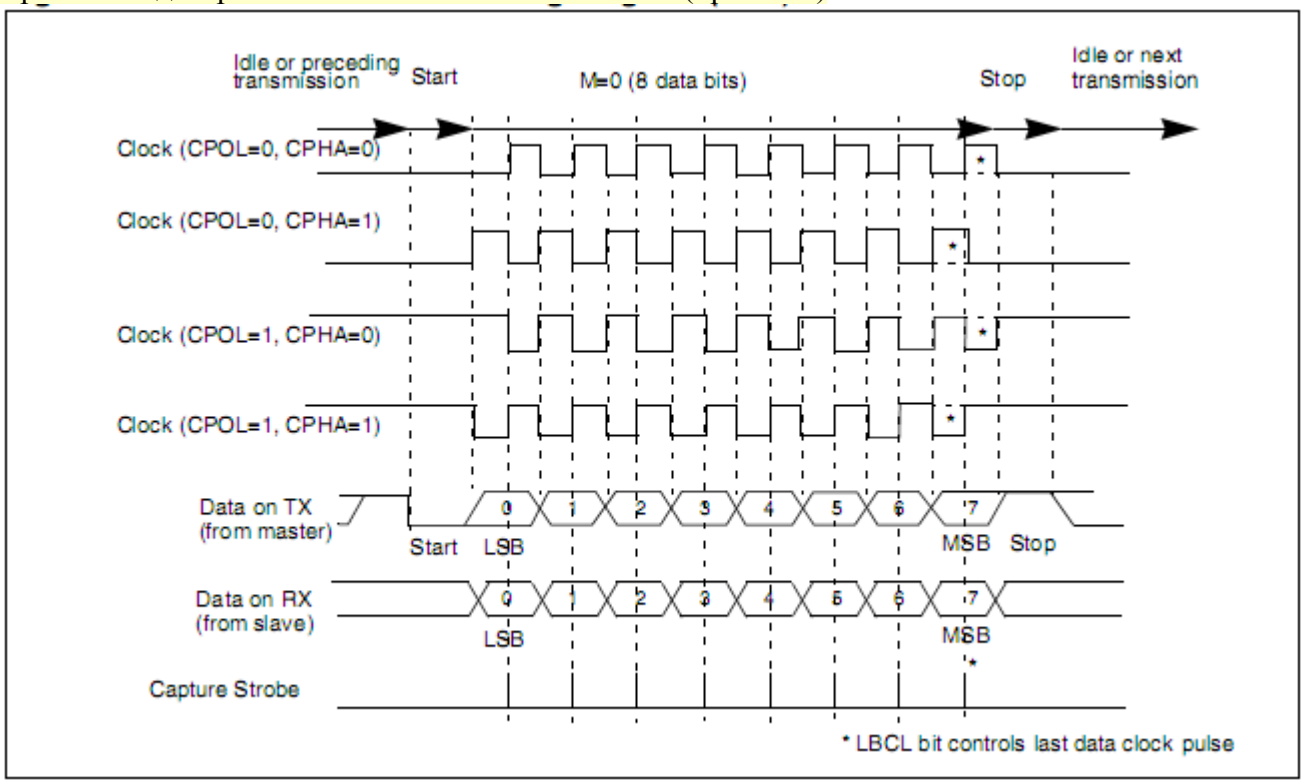


Figure 250. USART data clock timing diagram (M=1)
Временная диаграмма тактового сигнала **USART** (при **M=1**)

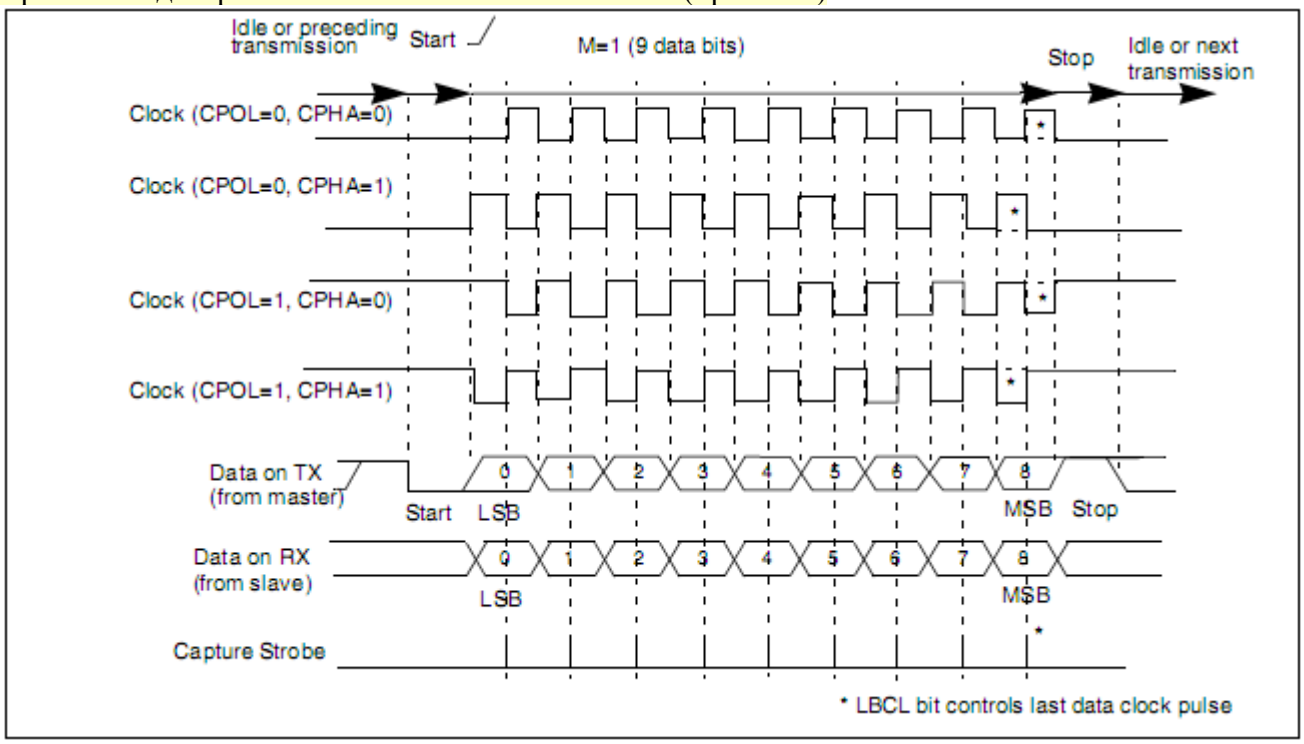
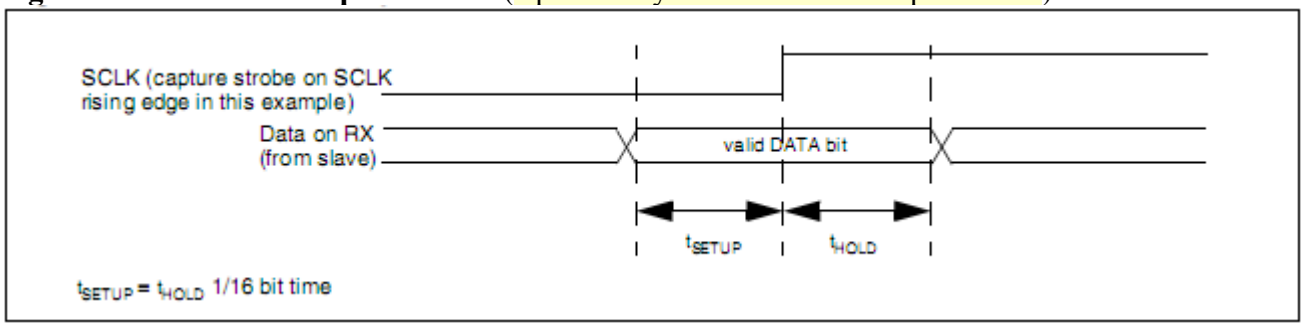


Figure 251. RX data setup/hold time (Время запуска и остановки приемника)



***Note**: The function of SCLK is different in Smartcard mode. Refer to the Smartcard mode chapter for more details.*
*Функциональность вывода **SCLK** отличается в режиме **Smartcard**. Обратитесь к разделу 25.3.11 "Smartcard" для дополнительной информации.*

25.3.10. Single-wire half-duplex communication
Обмен в однопроводном полу-дуплексном режиме

The single-wire half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:
Однопроводный полу-дуплексный режим выбирается установкой бита **HDSEL** в регистре **USART_CR3**. В этом режиме следующие биты должны удерживаться очищенными:

- LINEN and CLKEN bits in the USART_CR2 register
биты **LINEN** и **CLKEN** в регистре **USART_CR2**,
- SCEN and IREN bits in the USART_CR3 register.
биты **SCEN** и **IREN** в регистре **USART_CR3**.

The USART can be configured to follow a single-wire half-duplex protocol. In single-wire half-duplex mode, the TX and RX pins are connected internally. The selection between half- and full-duplex communication is made with a control bit ‘HALF DUPLEX SEL’ (HDSEL in USART_CR3).
USART можно конфигурировать на следующий однопроводный полу-дуплексный протокол. В этом режиме, выводы **TX** и **RX** соединены внутренне. Выбор между полу-дуплексным и полно-дуплексным обменом выполняется битом **HDSEL** в регистре **USART_CR3**).

As soon as HDSEL is written to 1: (Как только в бит **HDSEL** записана 1:)

- RX is no longer used, (**RX** более не используется,)
- TX is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as floating input (or output high open-drain) when not driven by the USART. **TX** всегда "свободен", когда данные не передаются. Таким образом, он действует как стандартный **I/O** вывод в состоянии покоя или приема. Это означает, что вывод **TX** должен быть сконфигурирован как "плавающий" вход (или как выход с открытым коллектором) когда он не управляется от **USART**.

Apart from this, the communications are similar to what is done in normal USART mode. The conflicts on the line must be managed by the software (by the use of a centralized arbiter, for instance). In particular, the transmission is never blocked by hardware and continue to occur as soon as a data is written in the data register while the TE bit is set.
Если отвлечся от этого, то обмен похож на нормальный **USART** режим. Конфликты на линии должны разруливаться программно (например, с использованием централизованного арбитра). в частности, передатчик никогда не блокируется аппаратно и продолжает работать, как только данные записаны в регистр данных, и бит **TE** установлен.

25.3.11 Smartcard

The Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In smartcard mode, the following bits must be kept cleared:
Режим **Smartcard** выбирается установкой бита **SCEN** в регистре **USART_CR3**. В этом режиме следующие биты должны удерживаться очищенными:

- LINEN bit in the USART_CR2 register (бит **LINEN** в регистре **USART_CR2**,)
- HDSEL and IREN bits in the USART_CR3 register.
биты **HDSEL** и **IREN** в регистре **USART_CR3**.

Moreover, the CLKEN bit may be set in order to provide a clock to the smartcard.
Сверх того, может быть установлен бит **CLKEN**, чтобы обеспечить тактирование **Smartcard**.

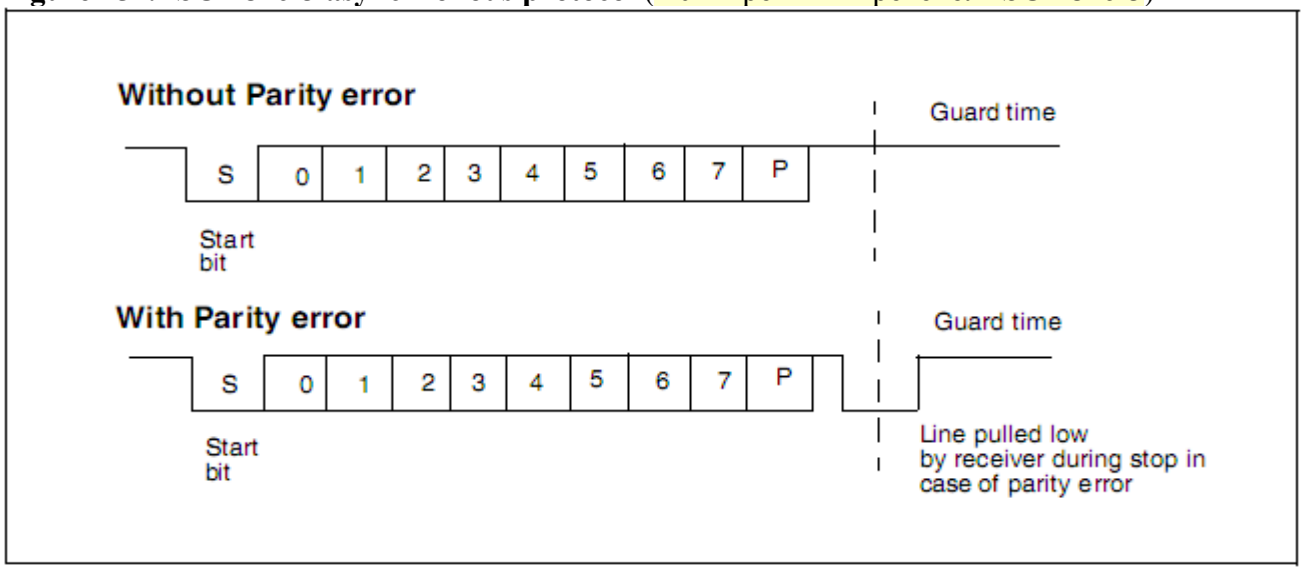
The Smartcard interface is designed to support asynchronous protocol Smartcards as defined in the ISO 7816-3 standard. The USART should be configured as:
Интерфейс **Smartcard** разработан для поддержки асинхронного протокола для **Smartcards**, что определен стандартом **ISO 7816-3**. **USART** должен быть сконфигурирован следующим образом:

- 8 bits plus parity: where M=1 and PCE=1 in the USART_CR1 register
8 бит плюс паритет: (**M=1** и **PCE=1** в регистре **USART_CR1**)
- 1.5 stop bits when transmitting and receiving: where STOP='11' in the USART_CR2 register.
1.5 стоп-бита на прием и передачу: (**STOP='11'** в регистре **USART_CR2**).

***Note**: It is also possible to choose 0.5 stop bit for receiving but it is recommended to use 1.5 stop bits for both transmitting and receiving to avoid switching between the two configurations.*
Возможно также выбрать 0.5 стоп-бита на прием, но рекомендуется использовать 1.5 стоп-бита для обоих каналов, чтобы избежать переключения между этими двумя конфигурациями.

Figure 252 shows examples of what can be seen on the data line with and without parity error.
Рис. 252 показывает пример, что можно увидеть на линии данных с ошибкой паритета и без нее.

Figure 252. ISO 7816-3 asynchronous protocol (Асинхронный протокол **ISO 7816-3**)



When connected to a smartcard, the TX output of the USART drives a bidirectional line that the smartcard also drives into. To do so, SW_RX must be connected on the same I/O than TX at product level. The Transmitter output enable TX_EN is asserted during the transmission of the start bit and the

data byte, and is deasserted during the stop bit (weak pull up), so that the receive can drive the line in case of a parity error. If TX_EN is not used, TX is driven at high level during the stop bit: Thus the receiver can drive the line as long as TX is configured in open-drain.
При подключении к **Smartcard**, выход **TX** должен быть двунаправленным, так как **Smartcard** также управляет им. Чтобы сделать это, **SW_RX** должен быть подключен к выводу **TX**. Разрешение вывода передатчика, как выхода **TX_EN**, ставится на время передачи старт-бита и байта данных, и отпускается на стоп-бите (*weak pull up*), так что приемник может управлять линией в случае ошибки паритета. Если **TX_EN** не используется, то **TX** ставится на высокий уровень к моменту передачи стоп-бита: Таким образом, приемник может управлять линией до тех пор, пока **TX** сконфигурирован как выход с открытым коллектором.

Smartcard is a single wire half duplex communication protocol.
Smartcard работает по однопроводному полу-дуплексному протоколу обмена.

- Transmission of data from the transmit shift register is guaranteed to be delayed by aminimum of 1/2 baud clock. In normal operation a full transmit shift register will start shifting on the next baud lock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
Передача данных из сдвигового регистра передачи гарантирована, если задержка не менее 1/2 **baud clock**. При нормальной работе, заполнение сдвигового регистра передачи вызовет старт сдвига по следующему перепаду тактового сигнала. В режиме **Smartcard** это передача будет отсрочена не менее, чем на 1/2 **baud clock**.

- If a parity error is detected during reception of a frame programmed with a 0.5 or 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the Smartcard that the data transmitted to USART has not been correctly received. This NACK signal (pulling transmit line low for 1 baud clock) will cause a framing error on the transmitter side (configured with 1.5 stop bits). The application can handle re-sending of data according to the protocol. A parity error is 'NACK'ed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted.
Если обнаружена ошибка паритета в процессе приема кадра при задании периода стоп бита как 0.5 или 1.5, то линия передачи будет притянута на низкий уровень на один период **baud clock** после завершения приема кадра. Это покажет **Smartcard**, что данные, переданные по **USART**, не были корректно приняты. Этот сигнал **NACK** (притяжка линии передачи к низу на 1 период **baud clock**) вызовет ошибку кадра на стороне передатчика (сконфигурированного на 1.5 стоп-бита). Приложение может управлять повторной посылкой данных в соответствии со своим протоколом. Ошибка паритета вызывает сигнал **NACK** от приемника, если поставлен бит управления **NACK**, в противном случае **NACK** не передается.

- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the guard time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the guard time counter reaches the programmed value TC is asserted high.
Выставление флага **TC** можно задержать, если запрограммировать **Guard Time** регистр. При нормальной работе, **TC** выставляется, когда сдвиговый регистр передачи опустошается и нет более запроса на передачу. В режиме **Smartcard** опустошение сдвигового регистра передачи запускает счетчик **Guard time**, чтобы отсчитать то время, что запрограммировано в регистре. На это время **TC** переводится в низкое состояние. Когда счетчик **Guard time** достигнет заданного значения, **TC** переводится в высокое состояние.

- The de-assertion of TC flag is unaffected by Smartcard mode.
Снятие флага **TC** не влияет на режим **Smartcard**.

- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the

NACK will not be detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
Если обнаружена ошибка кадра в конце передачи (так как получен **NACK** от приемника), то этот **NACK** не будет опознаваться как старт-бит, приемником на стороне передатчика. В соответствии с **ISO** протоколом, длительность принятого **NACK** может быть 1 или 2 периода **baud clock**.

- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver will not detect the NACK as a start bit.
На стороне приема, если обнаружена ошибка паритета и послан **NACK**, то он не будет опознаваться как старт-бит.

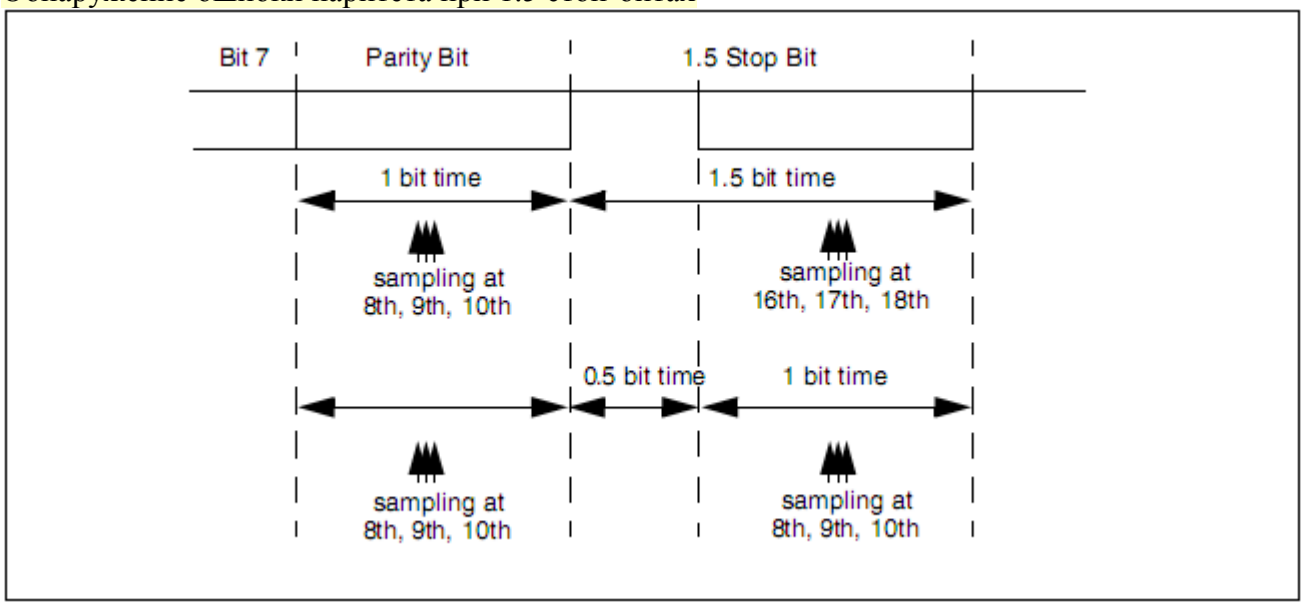
***Note**:*

1. *A break character is not significant in Smartcard mode. A 0x00 data with a framing error will be treated as data and not as a break.*
*Символ **Break** не опознается в режиме **Smartcard**. Данные со значением **0x00** и при наличии ошибки кадра рассматриваются именно как данные, а не как символ **Break**.*

2. *No IDLE frame is transmitted when toggling the TE bit. The IDLE frame (as defined for the other configurations) is not defined by the ISO protocol.*
*Никакого **IDLE** кадра не передается, когда переключается **TE** бит. Кадр **IDLE** не определен **ISO** протоколом.*

Figure 253 details how the NACK signal is sampled by the USART. In this example the USART is transmitting a data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.
Рис. 253 детализирует, как сигнал **NACK** сэмплируется **USART**. В этом примере **USART** передает данные и сконфигурирован на 1.5 стоп-бита. Принимающая часть **USART** разрешена, чтобы проверять целостность данных и сигнал **NACK**.

Figure 253. Parity error detection using the 1.5 stop bits
Обнаружение ошибки паритета при 1.5 стоп-битах



The USART can provide a clock to the smartcard through the SCLK output. In smartcard mode, SCLK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the prescaler register USART_GTPR. SCLK frequency can be programmed from $f_{CK}/2$ to $f_{CK}/62$, where f_{CK} is the peripheral input clock.
USART может обеспечить тактовый сигнал для **Smartcard** на выводе **SCLK**. В режиме **Smartcard**, **SCLK** не является частью интерфейса, он просто управляется от внутреннего

тактового сигнала через 5-ти битный делитель. Коэффициент деления конфигурируется в регистре делителя **USART_GTPR**. Частота **SCLK** может быть запрограммирована от **fCK/2** до **fCK/62**, где **fCK** - это входной тактовый периферии.

25.3.12 IrDA SIR ENDEC block (Блок SIR кодера/декодера для IrDA)

IrDA low-power mode (**IrDA** в режиме низкого потребления)

The IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:
Режим **IrDA** выбирается установкой бита **IREN** в регистре **USART_CR3**. В этом режиме следующие биты должны удерживаться очищенными:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
Биты **LINEN**, **STOP** и **CLKEN** в регистре **USART_CR2**,
- SCEN and HDSEL bits in the USART_CR3 register.
Биты **SCEN** и **HDSEL** в регистре **USART_CR3**.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 254*).
Физический слой спецификации **IrDA SIR** использует схему кодирования **RZI** (*Return to Zero, Inverted*), которая представляет логический '0', как импульс инфракрасного света (см. рис. 254).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2Kbps for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.
Кодер **SIR** передатчика преобразует выходной поток битов с **USART**, который работает по схеме **NRZ** (*Non Return to Zero*), в схему **RZI**. Выходной поток импульсов передается на внешний драйвер и инфракрасный светодиод. **USART** поддерживает скорости передачи информации только до 115.2 Kbps для **SIR ENDEC**. В нормальном режиме ширина передаваемых импульсов определяется как 3/16 периода бита.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to USART. The decoder input is normally HIGH (marking state) in the idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.
Декодер **SIR** приемника преобразует поток двоичных сигналов от инфракрасного датчика, который приходит в схеме **RZI**, обратно в схему **NRZ** и передает его на **USART**. Обычно, в неактивном состоянии, на входе декодера присутствует высокий уровень. На выходе кодера передатчика сигнал инверсный, относительно декодера приемника. Стартовый бит детектируется, когда на входе декодера присутствует низкий уровень.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (i.e. the USART is sending data to the IrDA encoder), any data on the IrDA receive line will be ignored by the IrDA decoder and if the Receiver is busy (USART is receiving decoded data from the USART), data on the TX from the USART to IrDA will not be encoded by IrDA. While receiving data, transmission should be avoided as the data to be transmitted could be corrupted.
IrDA - это полу-дуплексный протокол обмена. Если передатчик занят (то есть **USART** в процессе передачи данных на кодер **IrDA**), то любые данные на входной линии **IrDA** будут проигнорированы **IrDA** декодером, и, если приемник занят (**USART** в процессе приема декодированных данных), то данные от **USART** на выходе **TX** не будут закодированы **IrDA** кодером. Пока идет прием данных, нужно избегать передачи, поскольку данные, которые будут переданы, могут быть разрушены.

- A '0' is transmitted as a high pulse and a '1' is transmitted as a '0'. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see Figure 255).
Ноль передается как высокий импульс, а единица передается как '0'. Ширина импульса определена как 3/16 выбранного периода бита в нормальном режиме

- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
SIR декодер преобразует **IrDA** совместимый входной сигнал в битовый поток для **USART**.

- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
SIR приемник логически интерпретирует высокое состояние как логическую единицу и импульсы низкого уровня как логические нули.

- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when idle.
На выходе кодера передатчика сигнал инверсный, относительно декодера приемника. На выходе **SIR** находится низкий уровень при простое.

- The IrDA specification requires the acceptance of pulses greater than 1.41 us. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the IrDA low-power Baud Register, USART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than 2 periods will be accepted as a pulse. The IrDA encoder/decoder doesn't work when PSC=0.
Спецификация **IrDA** требует, чтобы ширина импульсов была не менее 1.41 нс. Приемлимая ширина импульса программируется. Логика обнаружения сбоя на стороне приемника отфильтровывает импульсы шириной менее 2-х **PSC** периодов (**PSC** - это значение делителя частоты, запрограммированное в регистре скорости обмена **IrDA** с низким потреблением, **USART_GTPR**). Импульсы шириной менее 1 **PSC** периода всегда вырезаются, те импульсы, что имеют ширину более одного и менее двух периодов, могут быть как приняты, так и потеряны, а когда ширина более двух периодов - всегда будут приняты как импульс. Кодер/декодер **IrDA** не работает, когда **PSC=0**.

- The receiver can communicate with a low-power transmitter.
Приемник может общаться с передатчиком, находящемся в режиме низкого потребления.

- In IrDA mode, the STOP bits in the USART_CR2 register must be configured to "1 stop bit".
В режиме **IrDA** биты **STOP** в регистре **USART_CR2** должны быть конфигурированы как "**1 stop bit**".

IrDA low-power mode (IrDA в режиме низкого потребления)

Transmitter: (Передатчик:)

In low-power mode the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally this value is 1.8432 MHz (1.42 MHz < PSC < 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.
В режиме низкого потребления ширина импульса не соответствует 3/16 периода бита. Вместо этого ширина импульса определяется как 3-х кратное значение скорости обмена в бодах для режима низкого потребления, минимальное значение которой 1.42 МГц. Как правило это значение составляет 1.8432 МГц (1.42 МГц < PSC < 2.12 МГц). Чтобы достигнуть этого значения, делитель частоты для режима низкого потребления использует системный тактовый сигнал.

Receiver: (Приемник:)

Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in USART_GTPR).
Приемник в режиме низкого потребления работает также, как и в нормальном режиме. В целях шумоподавления **USART** должен отфильтровывать импульсы, длительность которых менее 1/PSC. А корректно низкий импульс принимается только тогда, когда его продолжительность более 2-х периодов скорости обмена для режима **IrDA** с низким потреблением (это значение **PSC** в **USART_GTPR**).

***Note**:*

1. *A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.*
   *Импульсы шириной менее двух, но более одного периода **PSC** могут быть как приняты, так и потеряны.*

2. *The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).*
   *Время импульса, оптимальное для приемника, устанавливается программно. Спецификация физического уровеня **IrDA** определяет задержку не менее 10 ms между передачей и приемом (**IrDA** имеет полу-дуплексный протокол).*

Figure 254. IrDA SIR ENDEC- block diagram (Блок схема **SIR** кодера/декодера **IrDA**)

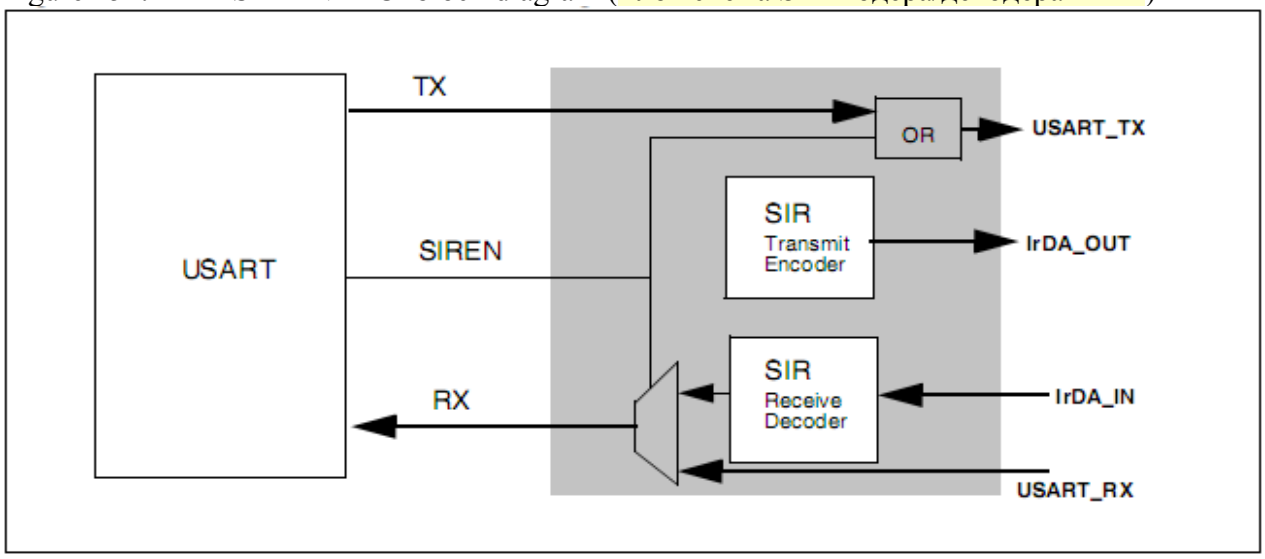
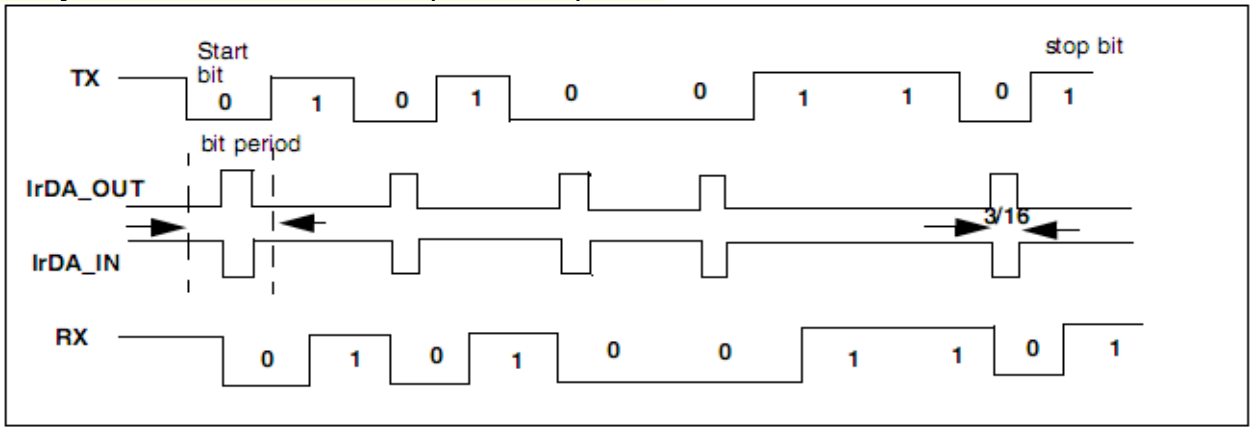


Figure 255. IrDA data modulation (3/16) -normal mode
Модуляция данных в **IrDA** - нормальный режим



25.3.13 Continuous communication using DMA
Непрерывный обмен с использованием DMA

Transmission using DMA (Передача с использованием **DMA**)
Reception using DMA (Прием с использованием **DMA**)
Error flagging and interrupt generation in multibuffer communication
Флаги ошибок и генерация прерываний при многобуферном обмене

The USART is capable to continue communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.
USART может осуществлять непрерывный обмен с использованием **DMA**. **DMA** требует независимые буфера для **Rx** и для **Tx**.

***Note**: You should refer to product specs for availability of the DMA controller. If DMA is not available in the product, you should use the USART as explained in Section 25.3.2 or 25.3.3. In the USART_SR register, you can clear the TXE/ RXNE flags to achieve continuous communication.*
*Вы должны обратиться к специфичной информации производителя по возможностям **DMA** контроллера. Если **DMA** не доступен в изделии, то вы должны использовать **USART** так, как поясняется в разделах 25.3.2 или 25.3.3. В регистре **USART_SR** вы можете очистить флаги **TXE/RXNE**, чтобы добиться непрерывного обмена.*

Transmission using DMA (Передача с использованием DMA)

DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data is loaded from a SRAM area configured using the DMA peripheral (refer to the DMA specification) to the USART_DR register whenever the TXE bit is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):
DMA режим можно разрешить для передачи установкой бита **DMAT** в регистре **USART_CR3**. Данные загружаются из области **SRAM**, сконфигурированной на использование **DMA** периферии (проверьте спецификацию **DMA**), в регистр **USART_DR**, когда установлен бит **TXE**. Чтобы отразить **DMA** канал на **USART** передатчик, используйте следующую процедуру (**x** - означает номер канала):

1. Write the USART_DR register address in the DMA control register to configure it as the destination of the transfer. The data will be moved to this address from memory after each TXE event.
Запишите адрес **USART_DR** регистра в **DMA** управляющий регистр, чтобы сконфигурировать его, как приемник для обмена. Данные будут перемещаться по этому адресу из памяти после каждого события **TXE**.

2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data will be loaded into the USART_DR register from this memory area after each TXE event.
Запишите адрес памяти в **DMA** управляющий регистр, чтобы сконфигурировать его, как источник для обмена. Данные будут перемещаться в регистр **USART_DR** из этой памяти после каждого события **TXE**.

3. Configure the total number of bytes to be transferred to the DMA control register.
Задайте общее число байт, которое будет передано, в управляющий регистр **DMA**.

4. Configure the channel priority in the DMA register.
Задайте приоритет канала в управляющем регистре **DMA**.

5. Configure DMA interrupt generation after half/ full transfer as required by the application.
Задайте генерацию **DMA** прерывания после **half/full** обмена, как того требует приложение.

6. Activate the channel in the DMA register. (Активируйте канал в **DMA** регистре.)

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector. The DMAT bit should be cleared by software in the USART_CR3 register during the interrupt subroutine.
Когда достигнуто число байт для обмена, запрограммированное в **DMA** контроллере, то он генерирует прерывание в векторе **DMA** канала. Бит **DMAT** должен быть очищен программно в регистре **USART_CR3** в обработчике прерывания.

***Note**: If DMA is used for transmission, do not enable the TXEIE bit.*
*Если **DMA** используется для передачи, не ставьте бит **TXEIE**.*

Reception using DMA (Прием с использованием DMA)

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data is loaded from the USART_DR register to a SRAM area configured using the DMA peripheral (refer to the DMA specification) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:
DMA режим можно разрешить для приема установкой бита **DMAR** в регистре **USART_CR3**. Данные загружаются из регистра **USART_DR** в область **SRAM**, сконфигурированной на использование **DMA** периферии (проверьте спецификацию **DMA**), как только получен байт данных. Чтобы отразить **DMA** канал на **USART** приемник, используйте следующую процедуру:

1. Write the USART_DR register address in the DMA control register to configure it as the source of the transfer. The data will be moved from this address to the memory after each RXNE event.
Запишите адрес **USART_DR** регистра в **DMA** управляющий регистр, чтобы сконфигурировать его, как источник для обмена. Данные будут перемещаться из этого адреса в память после каждого события **RXNE**.

2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data will be loaded from USART_DR to this memory area after each RXNE event.
Запишите адрес памяти в **DMA** управляющий регистр, чтобы сконфигурировать его, как приемник для обмена. Данные будут загружены из регистра **USART_DR** в эту память после каждого события **RXNE**.

3. Configure the total number of bytes to be transferred in the DMA control register.
Задайте общее число байт, которое будет принято, в управляющем регистре **DMA**.

4. Configure the channel priority in the DMA control register.
Задайте приоритет канала в управляющем регистре **DMA**.

5. Configure interrupt generation after half/ full transfer as required by the application.
Задайте генерацию **DMA** прерывания после **half/full** обмена, как того требует приложение.

6. Activate the channel in the DMA control register. (Активируйте канал в **DMA** регистре.)

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector. The DMAR bit should be cleared by software in the USART_CR3 register during the interrupt subroutine.
Когда достигнуто число байт для обмена, запрограммированное в **DMA** контроллере, то он генерирует прерывание в векторе **DMA** канала. Бит **DMAR** должен быть очищен программно в регистре **USART_CR3** в обработчике прерывания.

***Note**: If DMA is used for reception, do not enable the RXNEIE bit.*
*Если **DMA** используется для приема, не ставьте бит **RXNEIE**.*

Error flagging and interrupt generation in multibuffer communication
Флаги ошибок и генерация прерываний при многобуферном обмене

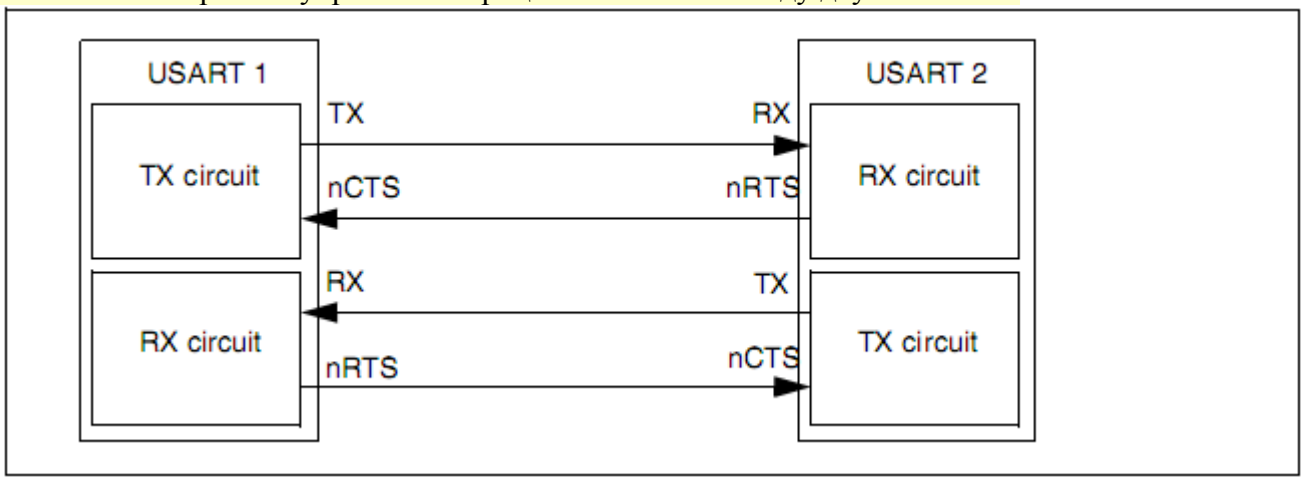
In case of multibuffer communication if any error occurs during the transaction the error flag will be asserted after the current byte. An interrupt will be generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE in case of single byte reception, there will be separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which if set will issue an interrupt after the current byte with either of these errors.
В случае многобуферного обмена, когда случается любая ошибка в процессе обмена, то после текущего байта будет ставиться флаг ошибки. Если стоит флаг разрешения прерывания, то оно будет генерироваться. Для флагов ошибок **framing**, **overrun** и **noise**, которые ставятся одновременно с **RXNE**, в случае однобайтного обмена, существуют отдельный флаг разрешения прерывания (**EIE** бит в регистре **USART_CR3**), который, если выставлен, генерирует прерывание после текущего байта, который вызвал эту ошибку.

25.3.14 Hardware flow control
Аппаратное управление процессом обмена

RTS flow control (Обмен с использованием линии **RTS**)
CTS flow control (Обмен с использованием линии **CTS**)

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The *Figure 256* shows how to connect 2 devices in this mode: Возможно управлять процессом обмена последовательных данных между двумя устройствами, используя вход **nCTS** и выход **nRTS**. Рис. 256 показывает, как соединить два устройства в этом режиме:

Figure 256. Hardware flow control between 2 USART
Рис. 256. Аппаратное управление процессом обмена между двумя **USART**



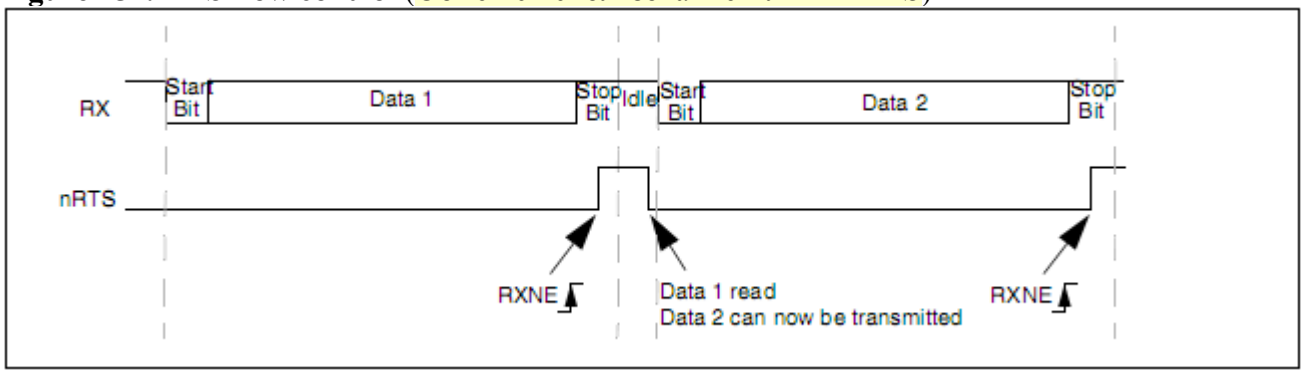
RTS and CTS flow control can be enabled independently by writing respectively RTSE and CTSE bits to 1 (in the USART_CR3 register).
Управляющие сигналы **RTS** и **CTS** могут быть разрешены независимо, записав '1' в соответствующий бит **RTSE** и **CTSE** (в регистре **USART_CR3**).

RTS flow control (Обмен с использованием линии RTS)

If the RTS flow control is enabled (RTSE=1), then nRTS is asserted (tied low) as long as the USART receiver is ready to receive new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 257* shows an example of communication with RTS flow control enabled.
Если разрешен управляющий сигнал **RTS** (**RTSE=1**), то выставляется сигнал **nRTS** (низкий

уровень) как только приемник **USART** готов принимать новые данные. Когда регистр приема полон, сигнал **nRTS** отпускается, показывая, что обмен ожидает остановки в конце текущего кадра. Рис. 257 показывает пример обмена с использованием линии **RTS**.

Figure 257. RTS flow control (Обмен с использованием линии **RTS**)

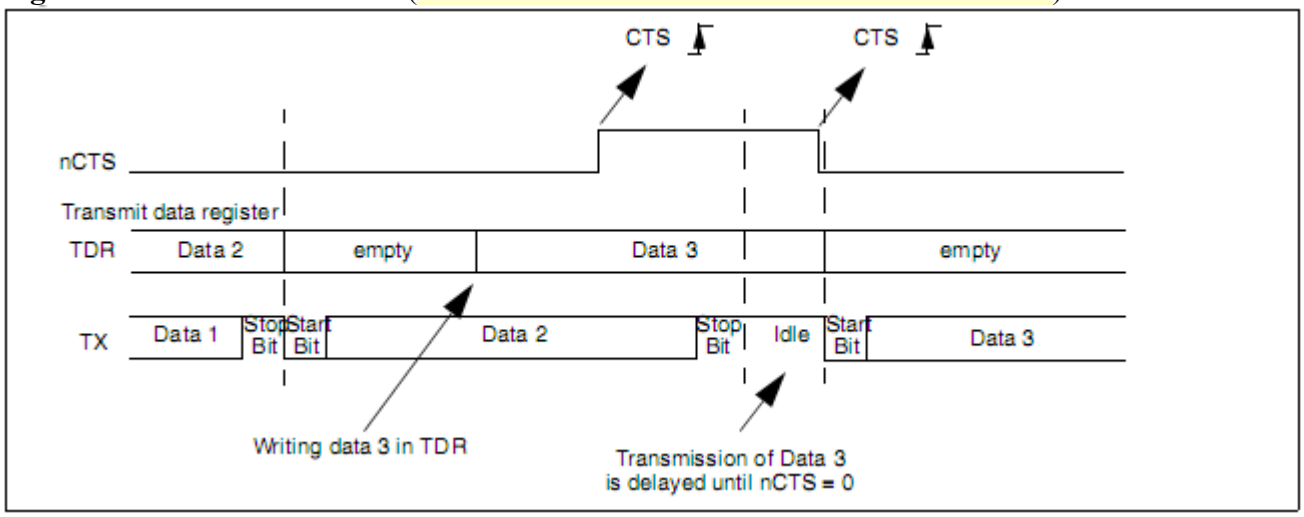


CTS flow control (Обмен с использованием линии CTS)

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that a data is to be transmitted, in other words, if TXE=0), else the transmission does not occur. When nCTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.
Если разрешен управляющий сигнал **CTS** (**CTSE=1**), то передатчик проверяет вход сигнала **nCTS**, перед передачей следующего кадра. Если сигнал **nCTS** выставлен (низкий уровень), то передаются следующие данные (конечно, если предыдущие данные были уже переданы, то есть **TXE=0**), иначе передача не идет. Когда сигнал **nCTS** отпускается в процессе передачи, то текущий обмен завершается и только потом передача останавливается.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. The figure below shows an example of communication with CTS flow control enabled.
Когда **CTSE=1**, то статусный бит **CTSIF** ставится автоматически аппаратно, как только защелкнут **nCTS** вход. Это показывает, что приемник готов (или не готов) к приему. Если установлен бит **CTSIE** в регистре **USART_CR3**, то генерируется прерывание. Рисунок ниже показывает пример обмена с использованием линии **CTS**.

Figure 258. CTS flow control (Рис. 258. Обмен с использованием линии **CTS**)



25.4 USART interrupts (Прерывания от USART)

Table 178. USART interrupt requests (Запросы **USART** на прерывание)

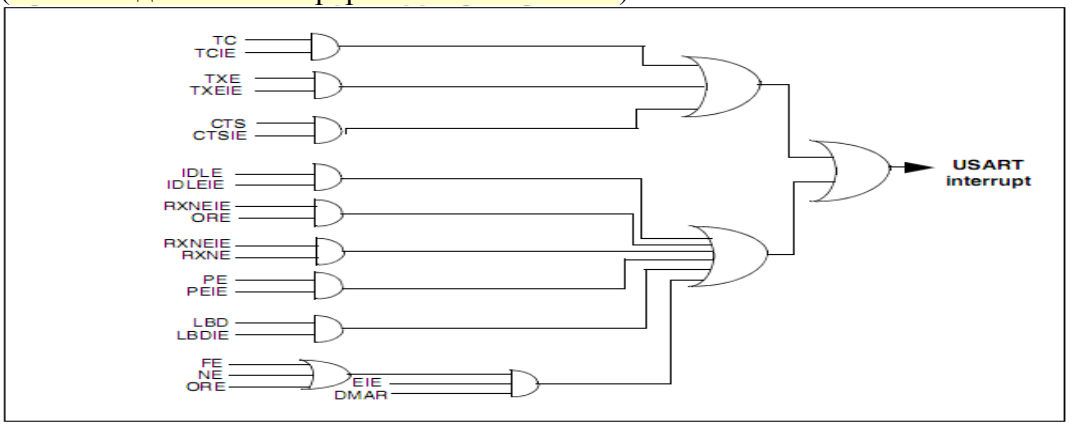
Interrupt event **Событие**	**Event flag** **Флаг**	**Enable Control bit (Бит разрешения)**
Transmit Data Register Empty (Регистр передачи пуст)	TXE	TXEIE
CTS flag (Флаг наличия сигнала **CTS**)	CTS	CTSIE
Transmission Complete (Передача завершена)	TC	TCIE
Received Data Ready to be Read (Принимаемые данные готов для чтения)	RXNE	RXNEIE
Overrun Error Detected (Есть ошибка Переполнения)	ORE	
Idle Line Detected (Обнаружен кадр **Idle**)	IDLE	IDLEIE
Parity Error (Есть ошибка паритета)	PE	PEIE
Break Flag (флаг символа **Break**)	LBD	LBDIE
Noise Flag, Overrun error and Framing Error in multibuffer communication Флаг наличия Шума, ошибка Переполнения и ошибка Кадра при многобуферном обмене	NE or ORE or FE	EIE

The USART interrupt events are connected to the same interrupt vector (see *Figure 259*).
К одному вектору прерывания подключены следующие события от **USART** (см. рис. 259).

- During transmission: Transmission Complete, Clear to Send or Transmit Data Register empty interrupt. В процессе передачи: "Передача завершена" (**TC**), "Сигнал готовности к приему" (**CTS**) и "Регистр передачи пуст" (**TXE**).
- While receiving: Idle Line detection, Overrun error, Receive Data register not empty, Parity error, LIN break detection, Noise Flag (only in multi buffer communication) and Framing Error (only in multi buffer communication).
В процессе приема: "Обнаружен кадр **Idle**" (**IDLE**), "Ошибка Переполнения" (**ORE**), "Регистр приема не пустой" (**RXNE**), "Ошибка Паритета" (**PE**), "обнаружен кадр **LIN Break**" (**LBD**), "Флаг наличия Шума" (**NE**) (только в для многобуферного обмена) и "Ошибка Кадра" (**FE**) (только в для многобуферного обмена).

These events generate an interrupt if the corresponding Enable Control Bit is set.
Эти события генерируют прерывание, если стоит соответствующий бит разрешения.

Figure 259. USART interrupt mapping diagram
(Схема подключений прерываний от **USART**)



25.5 USART mode configuration (Конфигурация режимов USART)

Table 179. USART mode configuration(1) (Наличие режимов **USART**)

USART modes	USART1	USART2	USART3	UART4	UART5
Asynchronous mode	+	+	+	+	+
Hardware Flow Control	+	+	+	-	-
Multibuffer Communication (DMA)	+	+	+	+	-
Multiprocessor Communication	+	+	+	+	+
Synchronous	+	+	+	-	-
Smartcard	+	+	+	-	-
Half-Duplex (Single-Wire mode)	+	+	+	+	+
IrDA	+	+	+	+	+
LIN	+	+	+	+	+

1. X = supported; NA = not applicable.

25.6 USART registers (USART регистры)

25.6.1 Status register (USART_SR) (Регистр статуса)
25.6.2 Data register (USART_DR) (Регистр данных)
25.6.3 Baud rate register (USART_BRR) (Регистр скорости обмена)
25.6.4 Control register 1 (USART_CR1) (Управляющий регистр 1)
25.6.5 Control register 2 (USART_CR2) (Управляющий регистр 2)
25.6.6 Control register 3 (USART_CR3) (Управляющий регистр 3)
25.6.7 Guard time and prescaler register (USART_GTPR)
Регистр делителя и задержки (*Guard time*) (**USART_GTPR**)
25.6.8 USART register map (Карта регистров **USART**)

Refer to *Section 1.1 on page 37* for a list of abbreviations used in register descriptions.
См. раздел 1.1, где приведен перечень сокращений, используемых в описании регистров.

25.6.1 Status register (USART_SR) (Регистр статуса)

Address offset: 0x00
Reset value: 0x00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
Res.						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

Bits 31:10 Reserved:
forced by hardware to 0. (Аппаратно ставится в 0.)

Bit 9 CTS: CTS flag Флаг сигнала на линии CTS

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software (by writing it to 0). An interrupt is generated if CTSIE=1 in the USART_CR3 register.
Этот бит ставится аппаратно, когда переключается вход **nCTS**, если стоит бит **CTSE**. И он очищается программно (записью в него нуля). Если в **USART_CR3** бит **CTSIE=1**, то генерируется прерывание.

0: No change occurred on the nCTS status line (Никаких изменений на линии **nCTS** нет)
1: A change occurred on the nCTS status line (Есть изменение на линии **nCTS**)

Note: This bit is not available for UART4 & UART5.
Этот бит недоступен для **UART4** и **UART5**.

Bit 8 LBD: LIN break detection flag Флаг обнаружения символа LIN Break

This bit is set by hardware when the LIN break is detected. It is cleared by software (by writing it to 0). An interrupt is generated if LBDIE = 1 in the USART_CR2 register. Этот бит ставится аппаратно, когда обнаруживается символ **LIN Break**. И он очищается программно (записью в него нуля). Если в **USART_CR2** бит **LBDIE=1**, то генерируется прерывание.

0: LIN Break not detected (Символ **LIN Break** не обнаружен)
1: LIN break detected (Символ **LIN break** обнаружен)

Note: An interrupt is generated when LBD=1 if LBDIE=1
Прерывание генерируется, когда **LBD=1** (если конечно **LBDIE=1**)

Bit 7 TXE: Transmit data register empty (Регистр данных передачи пуст)

This bit is set by hardware when the content of the TDR register has been transferred into the shift register. An interrupt is generated if the TXEIE bit =1 in the USART_CR1 register. It is cleared by a write to the USART_DR register.
Этот бит ставится аппаратно, когда содержимое регистра **TDR** было передано в сдвиговый регистр. Если в **USART_CR1** бит **TXEIE=1**, то генерируется прерывание. Этот бит очищается программно, записью в регистр **USART_DR**.

0: Data is not transferred to the shift register
Данные еще не переданы в сдвиговый регистр
1: Data is transferred to the shift register
Данные уже переданы в сдвиговый регистр

Note: This bit is used during single buffer transmission.
Этот бит используется в процессе однобуферного обмена.

Bit 6 TC: Transmission complete (Передача завершена)

This bit is set by hardware if the transmission of a frame containing data is complete and if TXE is set. An interrupt is generated if TCIE=1 in the USART_CR1 register. It is cleared by a software sequence (a read from the USART_SR register followed by a write to the USART_DR register). The TC bit can also be cleared by writing a '0' to it. This clearing sequence is recommended only for multibuffer communication.
Этот бит ставится аппаратно, когда передача кадра, содержащего данные, завершена и если установлен бит **TXE**. Если в **USART_CR1** бит **TCIE=1**, то генерируется прерывание. Этот бит очищается программно, последовательностью операций чтения регистра **USART_SR**, с последующей записью в регистр **USART_DR**. Бит **TC** можно также очистить записью в него '0'. Но это рекомендуется только при многобуферном обмене.

0: Transmission is not complete (Передача не завершена)
1: Transmission is complete (Передача завершена)

Bit 5 RXNE: Read data register not empty (В регистре данных приема есть данные)

This bit is set by hardware when the content of the RDR shift register has been transferred to the USART_DR register. An interrupt is generated if RXNEIE=1 in the USART_CR1 register. It is cleared by a read to the USART_DR register. The RXNE flag can also be cleared by writing a zero to it. This clearing sequence is recommended only for multibuffer communication.
Этот бит ставится аппаратно, когда содержимое сдвигового регистра **RDR** уже передано в **USART_DR** регистр. Если в **USART_CR1** бит **RXNEIE=1**, то генерируется прерывание. Этот бит очищается чтением регистра **USART_DR**. Флаг **RXNE** можно также очистить записью в него '0'. Но это рекомендуется только при многобуферном обмене.

- **0:** Data is not received (Данные еще не приняты)
- **1:** Received data is ready to be read. (Принимаемые данные готовы для чтения.)

Bit 4 IDLE: IDLE line detected (Обнаружено состояние простоя на линии)

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if the IDLEIE=1 in the USART_CR1 register. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).
Этот бит ставится аппаратно, когда обнаружено состояние **Idle**. Если в **USART_CR1** бит **IDLEIE=1**, то генерируется прерывание. Этот бит очищается программно, последовательностью операций чтения регистра **USART_SR**, с последующим чтением регистра **USART_DR**.

- **0:** No Idle Line is detected (Нет состояния состояния простоя на линии)
- **1:** Idle Line is detected (Обнаружено состояние простоя на линии)

Note: The IDLE bit will not be set again until the RXNE bit has been set itself (i.e. a new idle line occurs). Бит **IDLE** не будет ставиться снова до тех пор, пока не появится бит **RXNE** (то есть пока не случится новое **Idle** состояние).

Bit 3 ORE: Overrun error (Ошибка Переполнения)

This bit is set by hardware when the word currently being received in the shift register is ready to be transferred into the RDR register while RXNE=1. An interrupt is generated if RXNEIE=1 in the USART_CR1 register. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).
Этот бит ставится аппаратно, когда байт, полученный в сдвиговом регистре, готов к перемещению в регистр **RDR**, и появился бит **RXNE=1**. Если в **USART_CR1** бит **RXNEIE=1**, то генерируется прерывание. Этот бит очищается программно, последовательностью операций чтения регистра **USART_SR**, с последующим чтением регистра **USART_DR**.

- **0:** No Overrun error (Нет ошибки Переполнения)
- **1:** Overrun error is detected (Обнаружена ошибка Переполнения)

Note: When this bit is set, the RDR register content will not be lost but the shift register will be overwritten. An interrupt is generated on ORE flag in case of Multi Buffer communication if the EIE bit is set.
Note: Когда ставится этот бит, содержимое регистра **RDR** не теряется, но содержимое сдвигового регистра будет переписано. Если есть флаг **ORE** и стоит бит **EIE**, то, в случае многобуферного обмена, генерируется прерывание.

Bit 2 NE: Noise error flag (Флаг наличия Шума)

This bit is set by hardware when noise is detected on a received frame. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).
Этот бит ставится аппаратно, когда обнаружен шум в принимаемом кадре. Этот бит очищается программно, последовательностью операций чтения регистра **USART_SR**, с последующим чтением регистра **USART_DR**.

- **0:** No noise is detected (Нет Шума)
- **1:** Noise is detected (Обнаружен Шум)

Note: This bit does not generate interrupt as it appears at the same time as the RXNE bit which itself generates an interrupting interrupt is generated on NE flag in case of Multi Buffer communication if the EIE bit is set.
Note: Этот бит не генерирует прерывание, так как он появляется в тоже время, что и бит **RXNE**, который сам по себе генерирует прерывание. Перерывание от флага **NE** будет генерироваться только в случае многобуферного обмена, когда стоит бит **EIE**.

Bit 1 FE: Framing error (Ошибка Кадра)

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).
Этот бит ставится аппаратно, при расссинхронизации, интенсивном шуме или обнаружении символа **Break**. Этот бит очищается программно, последовательностью операций чтения регистра **USART_SR**, с последующим чтением регистра **USART_DR**.

- **0:** No Framing error is detected (Нет ошибки Кадра)
- **1:** Framing error or break character is detected (Обнаружена ошибка Кадра)

Note: This bit does not generate interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. If the word currently being transferred causes both frame error and overrun error, it will be transferred and only the ORE bit will be set.
An interrupt is generated on FE flag in case of Multi Buffer communication if the EIE bit is set.
Note: Этот бит не генерирует прерывание, так как он появляется в тоже время, что и бит **RXNE**, который сам генерирует прерывание. Если текущий принимаемый байт вызывает обе ошибки, Кадра и Переполнения, то он будет принят, но будет послављен флаг **ORE**.
Флаг **FE** будет генерировать прерывание в случае многобуферного обмена, если поставлен бит **EIE**.

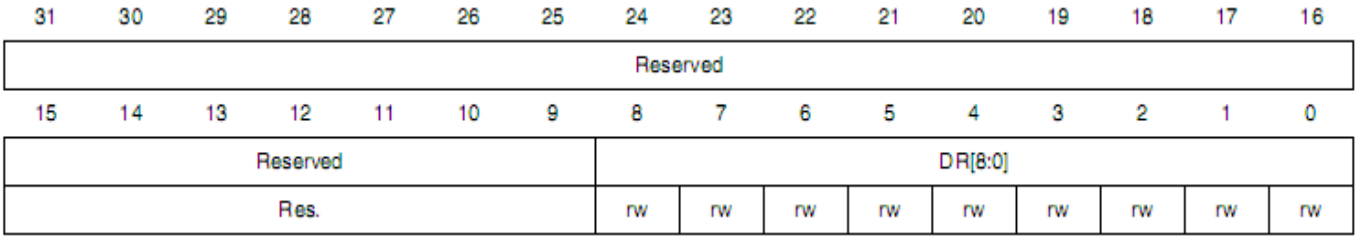
Bit 0 PE: Parity error (Ошибка Паритета)

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by a software sequence (a read to the status register followed by a read to the USART_DR data register). The software must wait for the RXNE flag to be set before clearing the PE bit. An interrupt is generated if PEIE = 1 in the USART_CR1 register.
Этот бит ставится аппаратно, когда обнаружена ошибка паритета при приеме байта. Этот бит очищается программно, последовательностью операций чтения регистра **USART_SR**, с последующим чтением регистра **USART_DR**. Программа должна дождаться установки **RXNE** флага, перед тем как очистить **PE** бит. Если в **USART_CR1** бит **PEIE=1**, то генерируется прерывание.

- **0:** No parity error (Нет ошибки Паритета)
- **1:** Parity error (Есть ошибка Паритета)

25.6.2 Data register (USART_DR) (Регистр данных)

Address offset: 0x04
Reset value: Undefined



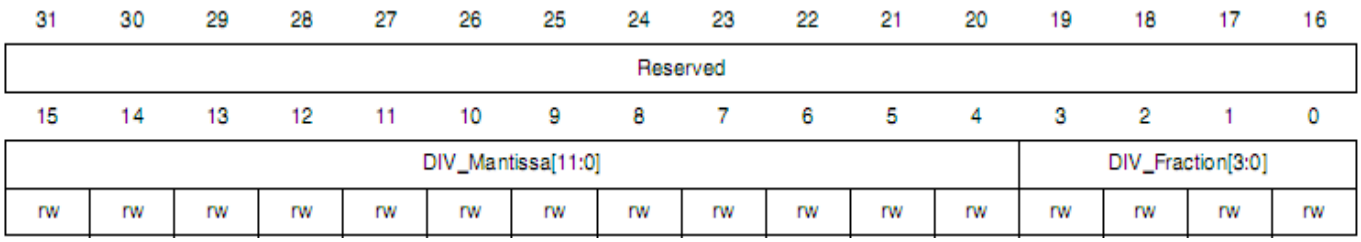
Bits 31:9 Reserved:
forced by hardware to 0. (Аппаратно ставится в 0.)

Bits 8:0 DR[8:0]: Data value (Значение данных)
Contains the Received or Transmitted data character, depending on whether it is read from or written to. The Data register performs a double function (read and write) since it is composed of two registers, one for transmission (TDR) and one for reception (RDR). The TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 1). The RDR register provides the parallel interface between the input shift register and the internal bus. When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity. When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.
Содержит байт принятых или передаваемых данных, в зависимости от текущей операции чтения или записи. Регистр данных выполняет двойную функцию (чтение и запись), так как в реальности он состоит из двух регистров, один для передачи (**TDR**), а другой для приема (**RDR**). Регистр **TDR** предоставляет параллельный интерфейс между внутренней шиной и выходным сдвиговым регистром (см. рис. 1). Регистр **RDR** предоставляет параллельный интерфейс между входным сдвиговым и внутренней шиной. Когда разрешен обмен с паритетом, (установлен бит **PCE** в регистре **USART_CR1**), то старший бит записанного значения (бит 7 или 8, в зависимости от длины данных) не имеет значения, так как он переписывается битом паритета. Когда разрешен прием с паритетом, то старший бит принятого значения - это бит паритета.

25.6.3 Baud rate register (USART_BRR) (Регистр скорости обмена)

Note: The baud counters stop counting if the TE or RE bits are disabled respectively.
Счетчик бодов останавливает счет, если бит **TE** (или **RE**, для примника) отключен.

Address offset: 0x08
Reset value: 0x0000



Bits 31:16 Reserved:
forced by hardware to 0. (Аппаратно ставится в 0.)

Bits 15:4 DIV_Mantissa[11:0]: mantissa of USARTDIV (целая часть значения USARTDIV)

These 12 bits define the mantissa of the USART Divider (USARTDIV)
Эти 12 бит определяют мантиссу делителя **USART** (**USARTDIV**)

Bits 3:0 DIV_Fraction[3:0]: fraction of USARTDIV (дробная часть USARTDIV)

These 4 bits define the fraction of the USART Divider (USARTDIV)
Эти 4 бита определяют дробная часть делителя **USART** (**USARTDIV**)

25.6.4 Control register 1 (USART_CR1) (Управляющий регистр 1)

Address offset: 0x0C
Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
Res.		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved:

forced by hardware to 0. (Аппаратно ставится в 0.)

Bit 13 UE: USART enable (разрешение USART)

When this bit is cleared the USART prescalers and outputs are stopped and the end of the current byte transfer in order to reduce power consumption. This bit is set and cleared by software.
Когда этот бит очищен, то делитель **USART** и его выход останавливаются, и заканчивается передача текущего байта, в целях сокращения потребления. Этот бит ставится и очищается программно.

`0:` USART prescaler and outputs disabled (делитель **USART** и выход на вывод отключены)
`1:` USART enabled (**USART** разрешен)

Bit 12 M: Word length (Длина слова)

This bit determines the word length. It is set or cleared by software.
Этот бит определяет длину слова. Этот бит ставится и очищается программно.

`0:` 1 Start bit, 8 Data bits, n Stop bit (1 старт-бит, 8 бит данных, **n** стоп-бит)
`1:` 1 Start bit, 9 Data bits, n Stop bit (1 старт-бит, 9 бит данных, **n** стоп-бит)

Note: The M bit must not be modified during a data transfer (both transmission and reception)
Бит **M** не должен модифицироваться в процессе обмена данных (как передачи, так и приема)

Bit 11 WAKE: Wakeup method (Способ пробуждения)

This bit determines the USART wakeup method, it is set or cleared by software.
Этот бит определяет способ пробуждения **USART**. Этот бит ставится и очищается программно.

`0:` Idle Line (Пробуждение от кадра **Idle**)
`1:` Address Mark (Пробуждение от корректного адреса)

Bit 10 PCE: Parity control enable (Разрешение контроля паритета)

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

Этот бит выбирает аппаратный контроль за паритетом (генерация и проверка). Когда контроль разрешен, то расчитанный бит паритета вставляется на место старшего бита (9-й бит при **M=1**; 8-й бит при **M=0**) и производится проверка паритета при приеме данных. Этот бит ставится и очищается программно. Когда ставят бит **PCE**, то он активируется после текущего байта (принимаемого или передаваемого).

0: Parity control disabled (Контроль паритета отключен)

1: Parity control enabled (Контроль паритета разрешен)

Bit 9 PS: Parity selection (Выбор типа паритета)

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity will be selected after the current byte.

Этот бит выбирает нечетный или четный паритет при его генерации/проверке (если есть бит **PCE**). Этот бит ставится и очищается программно. Новый тип паритета вступит в силу после текущего байта.

0: Even parity (Четный паритет)

1: Odd parity (Нечетный паритет)

Bit 8 PEIE: PE interrupt enable (Разрешение прерывания от PE (ошибка паритета))

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: Interrupt is inhibited (Прерывание отключено)

1: An USART interrupt is generated whenever PE=1 in the USART_SR register

Генерируется **USART** прерывание, когда есть бит **PE** в регистре **USART_SR**

Bit 7 TXEIE: TXE interrupt enable

Разрешение прерывания от TXE (регистр передачи пуст)

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: Interrupt is inhibited (Прерывание отключено)

1: An USART interrupt is generated whenever TXE=1 in the USART_SR register

Генерируется **USART** прерывание, когда есть бит **TXE** в регистре **USART_SR**

Bit 6 TCIE: Transmission complete interrupt enable

Разрешение прерывания от TC (передача завершена)

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: Interrupt is inhibited (Прерывание отключено)

1: An USART interrupt is generated whenever TC=1 in the USART_SR register

Генерируется **USART** прерывание, когда есть бит **TC** в регистре **USART_SR**

Bit 5 RXNEIE: RXNE interrupt enable

Разрешение прерывания от RXNE (регистр приема не пуст)

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: Interrupt is inhibited (Прерывание отключено)

1: An USART interrupt is generated whenever ORE=1 or RXNE=1 in the USART_SR register

Генерируется **USART** прерывание, когда есть бит **ORE** в регистре **USART_SR**

Bit 4 IDLEIE: IDLE interrupt enable

Разрешение прерывания от состояния простоя на линии

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: Interrupt is inhibited (Прерывание отключено)

1: An USART interrupt is generated whenever IDLE=1 in the USART_SR register

Генерируется **USART** прерывание, когда есть бит **IDLE** в регистре **USART_SR**

Bit 3 TE: Transmitter enable (Разрешение передатчика)

This bit enables the transmitter. It is set and cleared by software.

Этот бит разрешает передачу. Он ставится и очищается программно.

`0:` Transmitter is disabled (Передатчик отключен)

`1:` Transmitter is enabled (Передатчик разрешен)

Note:

1. During transmission, a “0” pulse on the TE bit (“0” followed by “1”) sends a preamble (idle line) after the current word, except in smartcard mode.
   В процессе передачи, переход **TE** бита с "0" в "1", посылает преамбулу (**Idle** кадр) после текущего символа, кроме режима **Smartcard**.
2. When TE is set there is a 1 bit-time delay before the transmission starts.
   Когда установлен **TE**, есть задержка в 1 **bit-time** перед передачей старт-бита.

Bit 2 RE: Receiver enable (Разрешение приемника)

This bit enables the receiver. It is set and cleared by software.

Этот бит разрешает прием. Он ставится и очищается программно.

`0:` Receiver is disabled (Приемник отключен)

`1:` Receiver is enabled and begins searching for a start bit

Приемник разрешен и начинает поиск старт-бита

Bit 1 RWU: Receiver wakeup (Пробуждение приемника)

This bit determines if the USART is in mute mode or not. It is set and cleared by software and can be cleared by hardware when a wakeup sequence is recognized.

Этот бит определяет, будет ли **USART** находится в безмолвном режиме (**mute**). Этот бит можно ставить и очищаеть программно, и его можно очистить аппаратно, когда распознается пробуждающая последовательность.

`0:` Receiver in active mode (Приемник в активном режиме)

`1:` Receiver in mute mode (Приемник в безмолвном режиме.)

Note:

1. Before selecting Mute mode (by setting the RWU bit) the USART must first receive a data byte, otherwise it cannot function in Mute mode with wakeup by Idle line detection.
   Перед выбором режима **Mute** (установкой бита **RWU**) **USART** должен сначала принять байт данных, в противном случае он не будет функционировать в режиме **Mute** с пробуждением от состояния **Idle** на линии.
2. In Address Mark Detection wakeup configuration (WAKE bit=1) the RWU bit cannot be modified by software while the RXNE bit is set.
   В конфигурации пробуждения при обнаружении адресного маркера (**WAKE=1**), бит **RWU** не может быть модифицирован программно до тех пор, пока стоит бит **RXNE**.

Bit 0 SBK: Send break (Посылка символа Break)

This bit set is used to send break characters. It can be set and cleared by software. It should be set by software, and will be reset by hardware during the stop bit of break.

Этот бит используется для посылки символа **Break**. Его можно ставить и очищаеть программно. Его нужно ставить программно и он будет очищаться аппаратно в процессе распознавания стоп-бита кадра **Break**.

`0:` No break character is transmitted (Символ **Break** не посылается)

`1:` Break character will be transmitted (Будет послан символ **Break**)

25.6.5 Control register 2 (USART_CR2) (Управляющий регистр 2)

Address offset: 0x10
Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LINEN	STOP[1:0]		CLK EN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	Res.	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved:
forced by hardware to 0. (Аппаратно ставится в 0.)

Bit 14 LINEN: LIN mode enable (Разрешение LIN режима)
This bit is set and cleared by software. (Этот бит ставится и очищается программно.)
0: LIN mode disabled (Режим **LIN** отключен)
1: LIN mode enabled (Режим **LIN** разрешен)
The LIN mode enables the capability to send LIN Synch Breaks (13 low bits) using the SBK bit in the USART_CR1 register, and to detect LIN Sync breaks.
Режим **LIN** дает возможность посылать синхронизирующие символы **LIN Break** (13 "нулей") с помощью бита **SBK** в регистре **USART_CR1**, и детектировать этот символ.

Bits 13:12 STOP: STOP bits (Формат стоп-битов)
These bits are used for programming the stop bits.
Эти биты используются для задания формата стоп-битов.
00: 1 Stop bit
01: 0.5 Stop bit
10: 2 Stop bits
11: 1.5 Stop bit

Note: The 0.5 Stop bit and 1.5 Stop bit are not available for UART4 & UART5.
Форматы 0.5 стоп-бита и 1.5 стоп-бита недоступны для **UART4** и **UART5**.

Bit 11 CLKEN: Clock enable (Разрешение тактового сигнала)
This bit allows the user to enable the SCLK pin.
Этот бит позволяет разрешить вывод тактового сигнала на вывод **SCLK**.
0: SCLK pin disabled (Вывод **SCLK** отключен)
1: SCLK pin enabled (Вывод **SCLK** разрешен)

Note: This bit is not available for UART4 & UART5.
Этот бит недоступен для **UART4** и **UART5**.

Bit 10 CPOL: Clock polarity (Полярность тактового сигнала)

This bit allows the user to select the polarity of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship
Этот бит позволяет выбрать полярность тактового сигнала на выводе **SCLK** в синхронном режиме. Он работает в связке с битом **CPHA**, чтобы получить желаемое соотношение тактового сигнала к данным.

0: Steady low value on SCLK pin outside transmission window.
Устойчивое низкое состояние на выводе **SCLK** вне окна обмена данными.

1: Steady high value on SCLK pin outside transmission window.
Устойчивое высокое состояние на выводе **SCLK** вне окна обмена данными.

Note: This bit is not available for UART4 & UART5.
Этот бит недоступен для **UART4** и **UART5**.

Bit 9 CPHA: Clock phase (Фаза тактового сигнала)

This bit allows the user to select the phase of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see figures *249* to *250*)
Этот бит позволяет выбрать фазу тактового сигнала на выводе **SCLK** в синхронном режиме. Он работает в связке с битом **CPOL**, чтобы получить желаемое соотношение тактового сигнала к данным. (см. рис. 249 и 250)

0: The first clock transition is the first data capture edge.
Первый перепад тактового импульса соответствует первому перепаду бита данных

1: The second clock transition is the first data capture edge.
Второй перепад тактового импульса соответствует первому перепаду бита данных

Note: This bit is not available for UART4 & UART5.
Этот бит недоступен для **UART4** и **UART5**.

Bit 8 LBCL: Last bit clock pulse (Тактовый импульс для последнего бита)

This bit allows the user to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the SCLK pin in synchronous mode.
Этот бит позволяет выбрать, будет ли тактовый импульс ассоциирован с последним (старшим) битом данных и выведен на **SCLK** в синхронном режиме.

0: The clock pulse of the last data bit is not output to the SCLK pin
Тактовый импульс не выводится для последнего бита на вывод **SCLK**)

1: The clock pulse of the last data bit is output to the SCLK pin
Выводится тактовый импульс для последнего бита на вывод **SCLK**

Note:

1. The last bit is the 8th or 9th data bit transmitted depending on the 8 or 9 bit format selected by the M bit in the USART_CR1 register. Последний бит 8-ми или 9-ти битного обмена зависит от бита **M** в регистре **USART_CR1**.
2. This bit is not available for UART4 & UART5.
   Этот бит недоступен для **UART4** и **UART5**.

Bit 7 Reserved:

forced by hardware to 0. (Аппаратно ставится в 0.)

Bit 6 LBDIE: LIN break detection interrupt enable
Разрешение прерывания при обнаружении LIN Break символа

Break interrupt mask (break detection using break delimiter).
Маска прерывания **Break** (обнаружение **Break** с использованием разделителя).

0: Interrupt is inhibited (Прерывание отключено)

1: An interrupt is generated whenever LBD=1 in the USART_SR register
Генерируется прерывание, если бит **LBD=1** в регистре **USART_SR**

Bit 5 LBDL: LIN break detection length (Длина LIN Break кадра)

This bit is for selection between 11 bit or 10 bit break detection.
Этот бит позволяет выбрать детектирование **Break** по 11 или по 10 битам.

`0:` 10 bit break detection (Детектирование **Break** по 10 битам)
`1:` 11 bit break detection (Детектирование **Break** по 11 битам)

Bit 4 Reserved:

forced by hardware to 0. (Аппаратно ставится в 0.)

Bits 3:0 ADD[3:0]: Address of the USART node (Адрес узла USART)

This bit-field gives the address of the USART node. This is used in multiprocessor communication during mute mode, for wake up with address mark detection.
Это битовое поле предоставляет адрес узла **USART**. Это используется при многопроцессорном обмене для выхода из режима **mute** по адресному маркеру.

Note: These 3 bits (CPOL, CPHA, LBCL) should not be written while the transmitter is enabled.
Эти три бита (**CPOL**, **CPHA**, **LBCL**) не должны записываться, пока разрешена передача.

25.6.6 Control register 3 (USART_CR3) (Управляющий регистр 3)

Address offset: 0x14
Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
Res.					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved:

forced by hardware to 0. (Аппаратно ставится в 0.)

Bit 10 CTSIE: CTS interrupt enable (Разрешение прерывания от CTS)

`0:` Interrupt is inhibited (Прерывание отключено)
`1:` An interrupt is generated whenever CTS=1 in the USART_SR register
Генерируется прерывание, если бит **CTS=1** в регистре **USART_SR**

Note: This bit is not available for UART4 & UART5.
Этот бит недоступен для **UART4** и **UART5**.

Bit 9 CTSE: CTS enable (Разрешение работы по CTS)

`0:` CTS hardware flow control disabled (Аппаратный контроль по сигналу **CTS** отключен)
`1:` CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0).
If the nCTS input is deasserted while a data is being transmitted, then the transmission is completed before stopping. If a data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.
CTS режим разрешен, данные передаются только тогда, когда выставлен сигнал на входе **nCTS** (низкий уровень). Если вход **nCTS** отпускается в процессе передачи данных, то текущая передача завершается а последующая останавливается. Если данные записываются в регистр данных, пока нет сигнала **nCTS**, то их передача откладывается до появления такого сигнала.

Note: This bit is not available for UART4 & UART5. Этот бит недоступен для **UART 4** и **5**.

Bit 8 RTSE: RTS enable (Разрешение работы по RTS)

`0:` RTS hardware flow control disabled (Работа по сигналу **RTS** отключена)

`1:` RTS interrupt enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (tied to 0) when a data can be received.
Разрешено прерывание от **RTS**, данные запрашиваются только тогда, когда есть место в буфере приема. Передача нового байта данных ожидается в случае, когда передача текущего байта завершена. Сигнал на выходе **nRTS** ставится (низкий уровень), когда данные можно принимать.

Note: This bit is not available for UART4 & UART5.
Этот бит недоступен для **UART4** и **UART5**.

Bit 7 DMAT: DMA enable transmitter (Разрешение передачи через DMA)

This bit is set/reset by software (Этот бит ставится и очищается программно.)

`0:` DMA mode is enabled for transmission (Режим **DMA** отключен для передатчика)

`1:` DMA mode is disabled for transmission (Режим **DMA** разрешен для передатчика)

Note: This bit is not available for UART5. (Этот бит недоступен для **UART5**.)

Bit 6 DMAR: DMA enable receiver (Разрешение приема через DMA)

This bit is set/reset by software (Этот бит ставится и очищается программно.)

`0:` DMA mode is enabled for reception (Режим **DMA** отключен для приемника)

`1:` DMA mode is disabled for reception (Режим **DMA** разрешен для приемника)

Note: This bit is not available for UART5. (Этот бит недоступен для **UART5**.)

Bit 5 SCEN: Smartcard mode enable (Разрешение режима Smartcard)

This bit is used for enabling Smartcard mode.
Этот бит используется для разрешения режима **Smartcard**.

`0:` Smartcard Mode disabled (Режим **Smartcard** отключен)

`1:` Smartcard Mode enabled (Режим **Smartcard** разрешен)

Note: This bit is not available for UART4 & UART5.
Этот бит недоступен для **UART4** и **UART5**.

Bit 4 NACK: Smartcard NACK enable (Разрешение NACK в режиме Smartcard)

`0:` NACK transmission in case of parity error is disabled
Запрещена передача **NACK** при обнаружении ошибки паритета

`0:` NACK transmission during parity error is enabled
Разрешена передача **NACK** при обнаружении ошибки паритета

Note: This bit is not available for UART4 & UART5.
Этот бит недоступен для **UART4** и **UART5**.

Bit 3 HDSEL: Half-duplex selection (Выбор режима Half-duplex)

Selection of Single-wire Half-duplex mode
Выбор однопроводного полу-дуплексного режима

`0:` Half duplex mode is not selected (Режим **Half duplex** отключен)

`1:` Half duplex mode is selected (Выбран режим **Half duplex**)

Bit 2 IRLP: IrDA low-power (Режим низкого потребления для IrDA)

This bit is used for selecting between normal and low-power IrDA modes
Этот бит используется для выбора между нормальным режимом **IrDA** и режимом с низким потреблением.

`0:` Normal mode (Нормальный режим)

`1:` Low-power mode (Режим низкого потребления)

Bit 1 IREN: IrDA mode enable (Разрешение режима IrDA)

This bit is set and cleared by software. (Этот бит ставится и очищается программно.)

0: IrDA disabled (Режим **IrDA** отключен)

1: IrDA enabled (Режим **IrDA** разрешен)

Bit 0 EIE: Error interrupt enable (Разрешение прерывания от ошибки)

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise error (FE=1 or ORE=1 or NE=1 in the USART_SR register) in case of Multi Buffer Communication (DMAR=1 in the USART_CR3 register).

Этот бит требуется для разрешения генерации прерывания в случае ошибки Кадра, ошибки Переполнения или ошибки Шума (**FE=1** или **ORE=1** или **NE=1** в регистре **USART_SR**), при многобуферном обмене (**DMAR=1** в **USART_CR3**).

0: Interrupt is inhibited (Прерывание отключено)

1: An interrupt is generated whenever DMAR=1 in the USART_CR3 register and FE=1 or ORE=1 or NE=1 in the USART_SR register.

Генерируется прерывание, если бит **DMAR=1** в регистре **USART_CR3** и есть бит **FE** или бит **ORE**, или бит **NE=1** в регистре **USART_SR**.

25.6.7 Guard time and prescaler register (USART_GTPR)

Регистр делителя и задержки

Address offset: 0x18
Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved:

forced by hardware to 0. (Аппаратно ставится в 0.)

Bits 15:8 GT[7:0]: Guard time value (Значение времени Guard time)

This bit-field gives the Guard time value in terms of number of baud clocks. This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.

Это битовое поле задает значение **Guard time** в тактах **baud clocks**. Оно используется в режиме **Smartcard**. Флаг завершения передачи ставится после этого времени.

Note: This bit is not available for UART4 & UART5.

Этот бит недоступен для **UART4** и **UART5**.

Bits 7:0 PSC[7:0]: Prescaler value (Значение делителя)

- **In IrDA Low-power mode**: (**В режиме IrDA с низким потреблением:**)
  PSC[7:0] = IrDA Low-Power Baud Rate
  Used for programming the prescaler for dividing the system clock to achieve the low-power frequency:
  The source clock is divided by the value given in the register (8 significant bits):
  00000000: Reserved - do not program this value
  00000001: divides the source clock by 1
  00000010: divides the source clock by 2
  ...
  PSC[7:0]: Скорость обмена для **IrDA** в режиме низкого потребления.
  Используется для программирования делителя на такое деление тактового сигнала системы, чтобы получить скорость обмена для режима **Low-Power**:
  Тактовый сигнал делится на значение, данное в регистре (8 значащих бит):
  00000000: **Reserved** - не программируйте это значение
  00000001: делит источник тактового сигнала на 1
  00000010: делит источник тактового сигнала на 2
  ...

- **In normal IrDA mode**: (В нормальном **IrDA** режиме:)
  PSC must be set to 00000001. (**PSC** должен быть установлен в **00000001**.

- **In smartcard mode**: (**В режиме Smartcard:**)
  PSC[4:0]: Prescaler value
  Used for programming the prescaler for dividing the system clock to provide the smartcard clock.
  The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:
  00000: Reserved - do not program this value
  00001: divides the source clock by 2
  00010: divides the source clock by 4
  00011: divides the source clock by 6
  ...
  PSC[4:0]: Значение делителя
  Используется для программирования делителя на такое деление тактового сигнала системы, чтобы получить скорость обмена для режима **Smartcard**:
  Значение, заданное в регистре (5 значащих бит) умножается на 2, что дает следующие значения делителя источник тактового сигнала:
  00000: **Reserved** - не программируйте это значение
  00001: делит источник тактового сигнала на 2
  00010: делит источник тактового сигнала на 4
  00011: делит источник тактового сигнала на 6
  ...

Note:

1. Bits [7:5] have no effect if Smartcard mode is used.
   Биты **[7:5]** не имеют эфекта в режиме **Smartcard**.
2. This bit is not available for UART4 & UART5.
   Эти биты недоступны для **UART4** и **UART5**.

25.6.8 USART register map (Карта регистров USART)

The table below gives the USART register map and reset values.
В таблице ниже дана карта регистров **USART** и их значение после сброса.

Table 180. USART register map and reset values
Карта регистров **USART** и их значение после сброса

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_SR	Reserved																						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value																							0	0	1	1	0	0	0	0	0	0
0x04	USART_DR	Reserved																							DR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x08	USART_BRR	Reserved																DIV_Mantissa[15:4]												DIV_Fraction [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USART_CR1	Reserved																		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_CR2	Reserved																	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Reserved	LBDIE	LBDL	Reserved	ADD[3:0]			
	Reset value																		0	0	0	0	0	0	0		0	0		0	0	0	0
0x14	USART_CR3	Reserved																					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x18	USART_GTPR	Reserved																GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Table 1 on page 41* for the register boundary addresses.
См. Табл. 1, где даны относительные адреса регистров.

STM32F105xx and STM32F107xx Errata sheet
Список ошибок семейства CL

STM32F105xx and STM32F107xx revision Z connectivity line device limitations
Ограничения на устройства **STM32F105xx** и **STM32F107xx** семейства **CL** ревизии **Z**

Silicon identification (Идентификация кристалла)
1 ARM™ 32-bit Cortex®-M3 limitations
(Ограничения со стороны ядра **ARM 32-bit Cortex-M3**)
2 STM32F10xxx silicon limitations (Ограничения со стороны кристалла **STM32F10xxx**)
Appendix A Revision and date codes on device marking
Приложение A. "Ревизии и коды на маркировке устройства"
Revision history (История изменений)

Silicon identification (Идентификация кристалла)

This errata sheet applies to the revision Z of the STMicroelectronics STM32F105xx and STM32F107xx connectivity line products. This family features an ARM 32-bit Cortex-M3 core, for which an errata notice is also available (see Section 1 for details).
Этот документ применим к ревизии **Z** контроллеров семейства **Connectivity Line** от **STMicroelectronics**: **STM32F105xx** и **STM32F107xx**. Особенностью этого семейства является ядро **ARM 32-bit Cortex-M3**, для которого также доступен свой документ об ошибках (*errata notice*) (подробности см. в разделе 1).

The full list of part numbers is shown in Table 2. The products are identifiable as shown in Table 1:
Полный список устройств семейства приведен в таблице 2. Устройства можно идентифицировать, как показано в таблице 1:

- by the revision code marked below the sales type on the device package
по коду ревизии, маркированному ниже типа устройства на его корпусе
- by the last three digits of the internal sales type printed on the box label
по последним трем цифрам внутреннего **sales type**, напечатанного на ярлыке упаковки

Table 1. Device Identification[1] (Идентификация устройств)

Sales type	Revision code[2] marked on device (Код ревизии на корпусе)
STM32F105xx	"Z"
STM32F107xx	"Z"

1. The REV_ID bits in the DBGMCU_IDCODE register show the revision code of the device (see the STM32F10xxx reference manual for details on how to find the revision code).
Поле **REV_ID** в регистре **DBGMCU_IDCODE** показывает код ревизии устройства (подробности, как найти этот код, см. в руководстве по STM32F10xxx).

2. Refer to Appendix A: Revision and date codes on device marking for details on how to identify the revision code on the different packages.
См. Приложение A: "Ревизии и коды на маркировке устройства", чтобы определиться с

местоположнием кодов на различных корпусах.

Table 2. Device summary (Полный список устройств)

Reference	Part number
STM32F105xx	STM32F105R8, STM32F105V8 STM32F105RB, STM32F105VB STM32F105RC, STM32F105VC
STM32F107xx	STM32F107RB, STM32F107VB STM32F107RC, STM32F107VC

Здесь, в оригинальном тексте, находится полное содержание документа, но, как уже упоминалось в предисловии, оно было переделано в систему иерархических содержаний.

1 ARM™ 32-bit Cortex®-M3 limitations
Ограничения со стороны ядра ARM 32-bit Cortex-M3

1.1 Cortex-M3 limitation description for the STM32F105xx / STM32F107xx connectivity line devices Описание ограничений от **Cortex-M3** на устройства **STM32F105xx - STM32F107xx**

An errata notice of the STM32F10xxx core is available from the following web address: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.eat0420a/.
The direct link to the errata notice pdf is:
http://infocenter.arm.com/help/topic/com.arm.doc.eat0420a/Cortex-M3-Errata-r1p1-v0.2.pdf.
Документ "**Errata Notice**" на ядро для **STM32F10xxx** доступно по следующим **web** адресам:
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.eat0420a/.
Прямая ссылка на **PDF** документ "**Errata Notice**":
http://infocenter.arm.com/help/topic/com.arm.doc.eat0420a/Cortex-M3-Errata-r1p1-v0.2.pdf.

All the described limitations are minor and related to the revision r1p1-01rel0 of the Cortex-M3 core. Table 3 summarizes these limitations and their implications on the behavior of the STM32F105xx / STM32F107xx connectivity line devices.
Все описанные ограничения незначительны и относятся к ревизиям **r1p1-01rel0** ядра **Cortex-M3**. Таблица 3 суммирует эти ограничения и их влияние на поведение устройств семейства **Connectivity Line**: **STM32F105xx - STM32F107xx**.

Table 3. Cortex-M3 core limitations and impact on microcontroller behavior
Ограничения ядра **Cortex-M3** и их влияние на поведение микроконтроллера

ID	ARM category	ARM summary of errata	Impact on STM32F105xx / STM32F107xx connectivity line devices
602117	Cat 2	LDRD with base in list may result in incorrect base register when interrupted or faulted Инструкция **LDRD** с базой в списке аргументов может привести к некорректному значению **SP** при наличии прерывания или ошибки шины	Minor
563915	Cat 2	Event register is not set by interrupts and debug Биты регистра событий не ставятся от прерываний и отладчиком	Minor
531064	impl	SWJ-DP missing POR reset sync	No
511864	Cat 3	Cortex-M3 may fetch instructions using incorrect privilege on return from an exception При возврате из исключения **Cortex-M3** может делать выборку инструкций, используя неправильную привилегию	No
532314	Cat 3	DWT CPI counter increments during sleep В режиме **sleep** счетчик **DWT CPI** продолжает считать	No
538714	Cat 3	Cortex-M3 TPIU clock domain crossing	No
548721	Cat 3	Internal write buffer could be active whilst asleep	No
463763	Cat 3	BKPT in debug monitor mode can cause DFSR mismatch Инструкция **BKPT** в режиме **debug monitor** может привести к	Minor

		несоответствию **DFSR**	
463764	Cat 3	Core may freeze for SLEEPONEXIT single instruction ISR Ядро **Cortex-M3** может быть "замороженным" при единственной инструкции обработчика **SLEEPONEXIT**	Minor
463769	Cat 3	Unaligned MPU fault during a write may cause the wrong data to be written to a successful first access Ошибка невыровненного **MPU** во время записи может привести к тому, что при успешном первом доступе могут быть записаны неправильные данные	No

1.1 Cortex-M3 limitation description for the STM32F105xx / STM32F107xx connectivity line devices
Описание ограничений от Cortex-M3 на устройства STM32F105xx - STM32F107xx

1.1.1 Cortex-M3 LDRD with base in list may result in incorrect base register when interrupted or faulted (**Cortex-M3 LDRD** с базой в списке аргументов может привести к некорректному значению **SP** при наличии прерывания или ошибки шины)
1.1.2 Cortex-M3 event register is not set by interrupts and debug
Биты регистра событий **Cortex-M3** не ставятся от прерываний и отладчиком
1.1.3 Cortex-M3 BKPT in debug monitor mode can cause DFSR mismatch (Инструкция **Cortex-M3 BKPT** в режиме **debug monitor** может привести к несоответствию **DFSR)**
1.1.4 Cortex-M3 may freeze for SLEEPONEXIT single instruction ISR (Ядро **Cortex-M3** может быть "замороженным" при единственной инструкции обработчика **SLEEPONEXIT**)

Only the limitations described below have an impact, even though minor, on the implementation of STM32F105xx / STM32F107xx connectivity line devices.
Только те ограничения, что описаны ниже, имеют влияние, пусть и незначительное, на работу устройств **STM32F105xx - STM32F107xx** линии **CL**.

All the other limitations described in the ARM errata notice (and summarized in Table 3 above) have no impact and are not related to the implementation of the STM32F105xx / STM32F107xx connectivity line devices (Cortex-M3 r1p1-01rel0).
Все другие ограничения, описанные в "**ARM errata notice**" (и сумированные в Табл. 3 выше) не имеют влияния на работу устройств **STM32F105xx - STM32F107xx** линии **CL** (**Cortex-M3 r1p1-01rel0**).

1.1.1 Cortex-M3 LDRD with base in list may result in incorrect base register when interrupted or faulted
Cortex-M3 LDRD с базой в списке аргументов может привести к некорректному значению SP при наличии прерывания или ошибки шины

Description (Описание)

The Cortex-M3 Core has a limitation when executing an LDRD instruction from the system- bus area, with the base register in a list of the form LDRD Ra, Rb, [Ra, #imm]. The execution may not complete after loading the first destination register due to an interrupt before the second loading completes or due to the second loading getting a bus fault.
Ядро **Cortex-M3** имеет ограничение, когда выполняет инструкцию **LDRD** в области **system-bus**, с базовым регистром в списке аргументов `LDRD Ra, Rb, [Ra, #imm]`. Ее выполнение может быть незавершенным после загрузки первого целевого регистра, вследствие прерывания перед загрузкой второго регистра, или вследствие того, что вторая загрузка вызвала ошибку шины.

Workarounds (Как обойти)

1. This limitation does not impact the STM32F10xxx code execution when executing from the embedded Flash memory, which is the standard use of the microcontroller.
Это ограничение не влияет на выполнение кода в **STM32F10xxx** при его исполнении из встроенной **Flash** памяти, что является стандартным при использовании микроконтроллера.

2. Use the latest compiler releases. As of today, they no longer generate this particular sequence. Moreover, a scanning tool is provided to detect this sequence on previous releases (refer to your preferred compiler provider).

Используйте последние версии компилятора. Сейчас они не генерируют подобное использование инструкции. Более того, предоставляются сканирующий инструмент, который определяет наличие такой инструкции в коде предыдущих версий компиляторов (справьтесь у провайдера вашего любимого компилятора).

1.1.2 Cortex-M3 event register is not set by interrupts and debug
Биты регистра событий Cortex-M3 не ставятся от прерываний и отладчиком

Description (Описание)

When interrupts related to a WFE occur before the WFE is executed, the event register used for WFE wakeup events is not set and the event is missed. Therefore, when the WFE is executed, the core does not wake up from WFE if no other event or interrupt occur.
Когда прерывание, связанное с **WFE** случается до того, как устройство вошло в режим **WFE** (пониженного потребления), то биты в регистре событий (*event*), используемый для пробуждения из **WFE**, не ставится и событие теряется. Следовательно, когда выполняется переход в **WFE**, то ядро не будет пробуждаться от **WFE**, если нет других событий или прерываний.

Workarounds (Как обойти)

Use STM32F10xxx external events instead of interrupts to wake up the core from WFE by configuring an external or internal EXTI line in event mode.
Используйте внешнее событие **STM32F10xxx**, вместо прерываний для пробуждения ядра из режима **WFE**, сконфигурировав внешнюю или внутреннюю линию **EXTI** в режим **event**.

1.1.3 Cortex-M3 BKPT in debug monitor mode can cause DFSR mismatch
Инструкция Cortex-M3 BKPT в режиме debug monitor может привести к несоответсвию DFSR

Description (Описание)

A BKPT may be executed in debug monitor mode. This causes the debug monitor handler to be run. However, the bit 1 in the Debug fault status register (DFSR) at address 0xE000ED30 is not set to indicate that it was originated by a BKPT instruction. This only occurs if an interrupt other than the debug monitor is already being processed just before the BKPT is executed.
Инструкция **BKPT** может быть выполнена в режиме **debug monitor**. Это приведет к тому, что будет запущен хендлер монитора отладчика. Однако, бит 1 в регистре **DFSR** (*Debug fault status register*) по адресу **0xE000ED30** не ставится, чтобы показать, что это была оригинальная инструкция **BKPT**. Это случается только тогда, когда уже началось обрабатываться другое прерывание, отличное от **debug monitor**, точно перед выполнением **BKPT**.

Workarounds (Как обойти)

If the DFSR register does not have any bit set when the debug monitor is entered, this means that we must be in this “corner case” and so, that a BKPT instruction was executed in debug monitor mode.
Если регистр **DFSR** не имеет каких либо установленных бит, при входе в **debug monitor**, то это означает, что попали в этот "крайний случай", и поэтому, эта инструкция **BKPT** была выполнена в режиме **debug monitor**.

1.1.4 Cortex-M3 may freeze for SLEEPONEXIT single instruction ISR

Ядро Cortex-M3 модет быть "замороженным" при единственной инструкции обработчика SLEEPONEXIT

Description (Описание)

If the Cortex-M3 SLEEPONEXIT functionality is used and the concerned interrupt service routine (ISR) contains only a single instruction, the core becomes frozen. This freezing may occur if only one interrupt is active and it is preempted by an interrupt whose handler only contains a single instruction.
Если используется функция **SLEEPONEXIT** ядра **Cortex-M3** и соответсвующий обработчик прерывания (**ISR**) содержит единственную инструкцию, то ядро становится "замороженным". Такое "замораживание" может случиться, если активно только одно прерывание и оно прервано (*preemption*) другим прерыванием, которое содержит единственную инструкцию.

However, any new interrupt that causes a preemption would cause the core to become unfrozen and behave correctly again. Однако, любое новое прерывание, которое прерывает текущее, заставил бы ядро разморозиться и поведение снова станет корректным.

Workarounds (Как обойти)

This scenario does not happen in real application systems since all enabled ISRs should at least contain one instruction. Therefore, if an empty ISR is used, then insert an NOP or any other instruction before the exit instruction (BX or BLX).
Такой сценарий не может произойти в реальном приложении, так как все разрешенные прерывания должны иметь по крайней мере одну инструкцию. Следовательно, если используется пустой обработчик, то вставьте **NOP** или любую другую инструкцию перед инструкцией выхода (**BX** или **BLX**).

2 STM32F10xxx silicon limitations
Ограничения со стороны кристалла STM32F10xxx

*Здесь, в оригинальном тексте, находится **Table 4. Summary of silicon limitations in revision Z devices**, но поскольку она функционально дублирует иерархические содержания, то она была опущена.*

2.1 Voltage glitch on ADC input 0 (Шум на входе 0 **ADC**)
2.2 Flash memory read after WFI/WFE instruction
Неверное чтение из **Flash** после инструкции **WFI/WFE**
2.3 Alternate function (Альтернативные функции)
2.4 Boundary scan TAP: wrong pattern sent out after the “capture IR” state (Интерфейс "**Boundary scan TAP**": посылается неправильный паттерн после состояния "**capture IR**")
2.5 Flash memory BSY bit delay versus STRT bit setting
Задержка бита **BSY Flash** памяти относительно установки бита **STRT**
2.6 I2C peripheral (**I2C** интерфейс)
2.7 SPI peripheral (**SPI** интерфейс)
2.8 General-purpose timers (Таймеры общего назначения)
2.9 LSI clock stabilization time (Время стабилизации тактового **LSI**)
2.10 PLL not locking when sourced by HSI/2 after reset
if it was previously sourced by HSE with predivider >1 or PLL2 (**PLL** не защелкивается)
2.11 OTG_FS
2.12 Ethernet MAC
2.13 Boot loader unavailability on STM32F105xx and STM32F107xx devices with a date code below 937 (Системный загрузчик недоступен для устройств **STM32F105xx** и **STM32F107xx** с кодом менее 937)

2.1 Voltage glitch on ADC input 0 (Шум на входе 0 ADC)

Description (Описание)

A low-amplitude voltage glitch may be generated (on ADC input 0) on the PA0 pin, when the ADC is converting with injection trigger. It is generated by internal coupling and synchronized to the beginning and the end of the injection sequence, whatever the channel(s) to be converted.
Могут генерироваться шумы малой амплитуды на входе **PA0**, когда **ADC** ведет преобразование с **injection trigger**. Этот шум генерируется внутренним соединением (утечкой) и синхронизирован с началом и окончанием **injection** процедуры, независимо от того, в каком канале происходит преобразование.

The glitch amplitude is less than 150 mV with a typical duration of 10 ns (measured with the I/O configured as high-impedance input and left unconnected). If PA0 is used as a digital output, this has no influence on the signal. If PA0 is used has a digital input, it will not be detected as a spurious transition, providing that PA0 is driven with an impedance lower than 5 kΩ. This glitch does not have any influence on the remaining port A pin or on the ADC conversion injection results, in single ADC configuration.
Амплитуды шума менее 150 mV с типичной длительностью около 10 ns (измерено при конфигурации **I/O** как плавающего входа, который остается неподключенным). Если **PA0** использовать как цифровой выход, то это не оказывает воздействия на сигнал. Если **PA0** использовать как цифровой вход, это не будет детектироваться, как ложное переключение, если обеспечен импеданс **PA0** менее 5 kΩ. Этот шум не оказывает какого либо влияния на остальные выводы порта **A** или на результат преобразования **ADC**, в конфигурации с одним **ADC**.

When using the ADC in dual mode with injection trigger, and in order to avoid any side effect, it is advised to distribute the analog channels so that Channel 0 is configured as an injected channel.
При использовании **ADC** в режиме **dual** с **injection trigger**, чтобы избежать каких либо побочных

эффектов, советуем распределить аналоговые каналы так, чтобы канал 0 был сконфигурирован как **injected** канал.

Workarounds (Как обойти)

None. (Никак.)

2.2 Flash memory read after WFI/WFE instruction
Неверное чтение из Flash после инструкции WFI/WFE

Conditions (Условия)

- Flash prefetch on (Включена предварительная выборка **Flash**)
- Flash memory timing set to 2 wait states

Тайминг для **Flash** памяти установлен в значение **2 wait states**

- FLITF clock stopped in Sleep mode (Тактовый **FLITF** остановлен в режиме **Sleep**)

Description (Описание)

If a WFI/WFE instruction is executed during a Flash memory access and the Sleep duration is very short (less than 2 clock cycles), the instruction fetch from the Flash memory may be corrupted on the next wakeup event.
Если инструкция **WFI/WFE** выполняется в процессе доступа к **Flash** памяти и длительность **Sleep** режима очень короткая (менее 2 циклов тактового), то выборка инструкции из **Flash** памяти может быть повреждена последующим событием пробуждения.

Workarounds (Как обойти)

When using the Flash memory with two wait states and prefetch on, the FLITF clock must not be stopped during the Sleep mode – the FLITFEN bit in the RCC_AHBENR register must be set (keep the reset value).
Когда **Flash** память используется с таймингом **2 wait states** и включена предварительная выборка, то тактовый **FLITF** не должен быть остановлен в течении режима **Sleep** - должен быть установлен бит **FLITFEN** регистра **RCC_AHBENR** (удерживать значение по умолчанию).

2.3 Alternate function (Альтернативные функции)

In some specific cases, some potential weakness may exist between alternate functions mapped onto the same pin. В некоторых особых случаях, могут существовать некоторые ограничения на использование альтернативных функций, дважды назначенных на один вывод.

2.3.1 SPI1 in slave mode and USART2 in synchronous mode
SPI1 в режиме **slave** и **USART2** в синхронном режиме
2.3.2 SPI1 in master mode and USART2 in synchronous mode
SPI1 в режиме **master** и **USART2** в синхронном режиме
2.3.3 SPI2 in slave mode and USART3 in synchronous mode
SPI2 в режиме **slave** и **USART3** в синхронном режиме
2.3.4 SPI2 in master mode and USART3 in synchronous mode
SPI2 в режиме **master** и **USART3** в синхронном режиме
2.3.5 I2S2 in master/slave mode and Ethernet/USART3 in synchronous mode
I2S2 в режиме **master/slave** и **Ethernet/USART3** в синхронном режиме
2.3.6 USARTx_TX pin usage (Использование вывода **USARTx_TX**)

2.3.1 SPI1 in slave mode and USART2 in synchronous mode
SPI1 в режиме slave и USART2 в синхронном режиме

Conditions (Условия)

- SPI1 and USART2 are clocked (Подан тактовый на **SPI1** и **USART2**)
- I/O port pin PA4 is configured as an alternate function output.

Вывод **PA4** сконфигурирован как альтернативный выход.

Description (Описание)

USART2 cannot be used in synchronous mode (USART2_CK signal), if SPI1 is used in slave mode.
USART2 нельзя использовать в синхронном режиме (нет сигнала **USART2_CK**), если **SPI1** используется в режиме **slave**.

Workarounds (Как обойти)

None. (Никак.)

2.3.2 SPI1 in master mode and USART2 in synchronous mode
SPI1 в режиме master и USART2 в синхронном режиме

Conditions (Условия)

- SPI1 and USART2 are clocked (Подан тактовый на **SPI1** и **USART2**)
- I/O port pin PA4 is configured as an alternate function output.

Вывод **PA4** сконфигурирован как альтернативный выход.

Description (Описание)

USART2 cannot be used in synchronous mode (USART2_CK signal) if SPI1 is used in master mode and SP1_NSS is configured in software mode. In this case USART2_CK is not output on the pin.
USART2 нельзя использовать в синхронном режиме (нет сигнала **USART2_CK**), если **SPI1** используется в режиме **master** и **SP1_NSS** сконфигурирован на программный режим. В этом случае не выводится сигнал на вывод **USART2_CK**.

Workarounds (Как обойти)

In order to output USART2_CK, the SSOE bit in the SPI1_CR2 register must be set to configure the pin in output mode.
Чтобы вывести сигнал на **USART2_CK**, надо установить бит **SSOE** в регистре **SPI1_CR2**, чтобы вывод был сконфигурирован как выход.

2.3.3 SPI2 in slave mode and USART3 in synchronous mode
SPI2 в режиме slave и USART3 в синхронном режиме

Conditions (Условия)

- SPI2 and USART3 are clocked (Подан тактовый на **SPI2** и **USART3**)
- I/O port pin PB12 is configured as an alternate function output.

Вывод **PB12** сконфигурирован как альтернативный выход.

Description (Описание)

USART3 cannot be used in synchronous mode (USART3_CK signal) if SPI2 is used in slave mode.
USART3 нельзя использовать в синхронном режиме (нет сигнала **USART3_CK**), если **SPI2** используется в режиме **slave**.

Workarounds (Как обойти)

None. (Никак.)

2.3.4 SPI2 in master mode and USART3 in synchronous mode
SPI2 в режиме master и USART3 в синхронном режиме

Conditions (Условия)

- SPI2 and USART3 are clocked (Подан тактовый на **SPI2** и **USART3**)
- I/O port pin PB12 is configured as an alternate function output

Вывод **PB12** сконфигурирован как альтернативный выход.

Description (Описание)

USART3 cannot be used in synchronous mode (USART3_CK signal) if SPI2 is used in master mode and SP2_NSS is configured in software mode. In this case USART3_CK is not output on the pin.
USART3 нельзя использовать в синхронном режиме (нет сигнала **USART3_CK**), если **SPI2** используется в режиме **master** и **SP2_NSS** сконфигурирован на программный режим. В этом случае не выводится сигнал на вывод **USART3_CK**.

Workarounds (Как обойти)

In order to output USART3_CK, the SSOE bit in the SPI2_CR2 register must be set to configure the pin in output mode.
Чтобы вывести сигнал на **USART3_CK**, надо установить бит **SSOE** в регистре **SPI2_CR2**, чтобы вывод был сконфигурирован как выход.

2.3.5 I2S2 in master/slave mode and Ethernet/USART3 in synchronous mode
I2S2 в режиме master/slave и Ethernet/USART3 в синхронном режиме

Conditions (Условия)

- USART3 in synchronous mode or Ethernet is clocked

USART3 в синхронном режиме или подан тактовый на **Ethernet**

- I2S2 is not clocked (На **I2S2** нет тактового)
- I/O port pin PB12 is configured as an alternate function output

Вывод **PB12** сконфигурирован как альтернативный выход.

Description (Описание)

If I2S2 was used prior to operating USART3 in synchronous mode or the Ethernet, a conflict occurs between the I2S2_WS and the ETH_MII_TXD0 / USART3_CK signals even though the I2S2 clock was disabled.
Если перед работой с **USART3** в синхронном режиме был задействован **I2S2** или **Ethernet**, то возникает конфликт между сигналами **I2S2_WS** и **ETH_MII_TXD0 / USART3_CK**, хотя с **I2S2** уже был снят тактовый.

Workarounds (Как обойти)

To use USART3 in synchronous mode, first disable the I2S2 clock, then perform a software reset of SPI2(I2S2).
Чтобы использовать **USART3** в синхронном режиме, сначала снимите тактовый с **I2S2**, затем выполните программный сброс **SPI2(I2S2)**.

To use the Ethernet, first disable the I2S2 clock, then either perform a software reset of SPI2(I2S2) or switch off the I2S mode of SPI2.
Чтобы использовать **Ethernet**, сначала снимите тактовый с **I2S2**, затем либо выполните программный сброс **SPI2(I2S2)**, либо выключите **I2S** режим **SPI2**.

2.3.6 USARTx_TX pin usage (Использование вывода USARTx_TX)

Description (Описание)

In USART receive-mode-only communication (TE = 0 in the USARTx_CR1 register), even when the USARTx_TX pin is not being used, the corresponding I/O port pin cannot be used to output another alternate function (in this mode the USARTx_TX output is set to 1 and thus no other alternate function output can be used).
При обмене с **USART** только в режиме приема (**TE=0** в регистре **USARTx_CR1**), даже когда вывод **USARTx_TX** не используется, вывод соответствующего **I/O** порта нельзя использовать в качестве другого альтернативного выхода (в этом режиме выход **USARTx_TX** ставится в '1' и, таким образом, никакая другая альтернативная функция не может быть использована как выход).

This limitation applies to all USARTx_TX pins that share another alternate function output.
Это ограниение применимо ко всем выводам **USARTx_TX**, которые разделяют его с другим альтернативным выходом.

Workarounds (Как обойти)

Do not use the corresponding I/O port of the USARTx_TX pin in alternate function output mode. Only the input mode can be used (TE bit in the USARTx_CR1 has to be cleared).
Не используйте соответствующий **I/O** порт вывода **USARTx_TX** как альтернативный выход. Его можно использовать только как вход (бит **TE** регистра **USARTx_CR1** должен быть очищен).

2.4 Boundary scan TAP: wrong pattern sent out after the “capture IR” state
Интерфейс "Boundary scan TAP": посылается неправильный паттерн после состояния "capture IR"

Description (Описание)

After the “capture IR” state of the boundary scan TAP, the two lower significant bits in the instruction register should be loaded with “01” for them to be shifted out whenever a next instruction is shifted in.
После состояния "**capture IR**" интерфейса "**boundary scan TAP**", два младших бита регистра инструкций должны быть загружены значением '01', которые затем будут выдвигаться всякий раз, когда следующая инструкция задвигает свои биты.

]However, the boundary scan TAP shifts out the latest value loaded into the instruction register, which could be “00”, “01”, “10” or “11”.
Однако, "**boundary scan TAP**" выдвигает последнее значение, загруженное в регистр инструкций, которое должно быть '00', '01', '10' или '11'.

Workarounds (Как обойти)

The data shifted out, after the capture IR state, in the boundary scan flow should therefore be ignored and the software should check not only the two least significant bits (XXX01) but all register bits (XXXXX).
Данные, выдвигаемые после состояния "**capture IR**", в потоке "**Boundary scan TAP**", должны быть, следовательно, игнорированы и программа должна проверять не только два последних младших бита (**XXX01**), но весь регистр (**XXXXX**).

2.5 Flash memory BSY bit delay versus STRT bit setting
Задержка бита BSY Flash памяти относительно установки бита STRT

Description (Описание)

When the STRT bit in the Flash memory control register is set (to launch an erase operation), the BSY bit in the Flash memory status register goes high one cycle later.
Когда ставится бит **STRT** в управляющем регистре **Flash** памяти (чтобы запустить операцию стирания), бит **BSY** в статусном регистре **Flash** памяти переходит в '1' на один такт позже.

Therefore, if the FLASH_SR register is read immediately after the FLASH_CR register is written (STRT bit set), the BSY bit is read as 0.
Следовательно, при чтении регистра **FLASH_SR** сразу после записи в регистр **FLASH_CR** (установка бита **STRT**), бит **BSY** читается как '0'.

Workarounds (Как обойти)

Read the BSY bit at least one cycle after setting the STRT bit.
Читать бит **BSY**, по крайней мере, через один такт после установки бита **STRT**.

2.6 I2C peripheral (I2C интерфейс)

2.6.1 Some software events must be managed before the current byte is being transferred
(Некоторые программные события должны управляться до того, как начнет передаваться текущий байт)
2.6.2 SMBus standard not fully supported (Не полностью поддерживается стандарт **SMBus**)
2.6.3 Start cannot be generated after a misplaced Stop
(**Start** не может генерироваться после неуместного **Stop**)
2.6.4 Mismatch on the “Setup time for a repeated Start condition” timing parameter
(Несоответствие параметра "Время установки для повторного старт-состояния")
2.6.5 Data valid time ($t_{VD;DAT}$) violated without the OVR flag being set
(Время валидности данных (**tVD;DAT**) нарушается установкой флага **OVR**)

2.6.1 Some software events must be managed before the current byte is being transferred
Некоторые программные события должны управляться до того, как начнет передаваться текущий байт

Description (Описание)

When the EV7, EV7_1, EV6_1, EV2, EV8, and EV3 events are not managed before the current byte is being transferred, problems may be encountered such as receiving an extra byte, reading the same data twice or missing data.
Когда не управляют событиями **EV7**, **EV7_1**, **EV6_1**, **EV2**, **EV8** и **EV3** прежде, чем начнет передаваться текущий байт, можно столкнуться с такими проблемами, как получение лишнего байта, чтение тех же самых данных дважды или потеря данных.

Workarounds (Как обойти)

When it is not possible to manage the EV7, EV7_1, EV6_1, EV2, EV8, and EV3 events before the

current byte transfer and before the acknowledge pulse when changing the ACK control bit, it is recommended to:
Когда не возможно управлять этими событиями перед текущей передачей байта и перед началом передачи текущего байта с последующим битом уведомления, когда изменение **ACK** управляет битом, то рекомендуется следующее:

1. use the I2C with DMA in general, except when the Master is receiving a single byte
как правило, используйте **I2C** совместно с **DMA**, кроме тех случаев, когда **Master** получает единственный байт
2. use I2C interrupts and boost their priorities to the highest one in the application to make them uninterruptible
используйте прерывания **I2C** и повысьте их приоритеты до максимума, чтобы сделать их непрерывными

2.6.2 SMBus standard not fully supported
Не полностью поддерживается стандарт SMBus

Description (Описание)

The I2C peripheral is not fully compliant with the SMBus v2.0 standard since It does not support the capability to NACK an invalid byte/command.
Интерфейс **I2C** не полностью соответствует стандарту **SMBus v2.0**, так как он не поддерживает способность отвечать **NACK** в ответ на неверный байт (команду).

Workarounds (Как обойти)

A higher-level mechanism should be used to verify that a write operation is being performed correctly at the target device, such as:
Должен использоваться высокоуровневый механизм, чтобы убедиться, что операция записи в целевое устройство выполняется корректно, это такие механизмы, как:

1. Using the SMBA pin if supported by the host
Использование вывода **SMBA**, если он поддерживается Хостом
2. the alert response address (ARA) protocol
Использование протокола ответа с помощью аварийного адреса (**ARA**)
3. the Host notify protocol
Использование протокола уведомления от Хоста

2.6.3 Start cannot be generated after a misplaced Stop
Start не может генерироваться после неуместного Stop

Description (Описание)

If a master generates a misplaced Stop on the bus (bus error), the peripheral cannot generate a Start anymore.
Если **Master** генерирует неуместное стоп-состояние на шине (ошибка шины), то интерфейс не может больше генерировать старт-состояние.

Workarounds (Как обойти)

In the I²C standard, it is allowed to send a Stop only at the end of the full byte (8 bits + acknowledge), so this scenario is not allowed. Other derived protocols like CBUS allow it, but they are not supported by the I²C peripheral.
В стандарте **I2C** допускается посылать стоп-состояние только в конце полного байта (8 бит + уведомление), таким образом этот сценарий не допускается. Другие, производные протоколы, такие как **CBUS**, допускают это, но они не поддерживаются **I2C** интерфейсом.

A software workaround consists in asserting the software reset using the SWRST bit in the I2C_CR1 control register.
Программный обход проблемы состоит в программном сбросе интерфейса, используя бит **SWRST** в управляющем регистре **I2C_CR1**.

2.6.4 Mismatch on the “Setup time for a repeated Start condition” timing parameter
Несоответсвие параметра "Время установки для повторного старт-состояния"

Description (Описание)

In case of a repeated Start, the “Setup time for a repeated Start condition” (named Tsu;sta in the I²C specification) can be slightly violated when the I²C operates in Master Standard mode at a frequency between 88 kHz and 100 kHz.
В случае повторного старт-состояния, “Время установки для повторного старт-состояния” (именованное в спецификации **I2C** как **Tsu;sta**), может немного не соответсвовать, когда **I2C** работает в стандартном режиме **Master** на частоте 88 - 100 кГц.

The issue can occur only in the following configuration:
Проблема может произойти только в следующей конфигурации:

- in Master mode (в режиме **Master**)
- in Standard mode at a frequency between 88 kHz and 100 kHz (no issue in Fast-mode)
в стандартном режиме на частоте 88 - 100 кГц. (не бывает в режиме **Fast**)
- SCL rise time: (Высокий уровень **SCL**:)
    - If the slave does not stretch the clock and the SCL rise time is more than 300 ns (if the SCL rise time is less than 300 ns the issue cannot occur)
    Если **slave** не затягивает тактовый импульс, и время высокого уровня **SCL** - больше чем 300 ns (если время высокого уровня **SCL** - меньше чем 300 ns, проблема не возникает)
    - If the slave stretches the clock (Если **slave** затягивает тактовый импульс)

The setup time can be violated independently of the APB peripheral frequency.
Время установки может быть нарушено независимо от частоты периферии **APB**.

Workarounds (Как обойти)

Reduce the frequency down to 88 kHz or use the I²C Fast-mode if supported by the slave.
Уменьшите частоту ниже 88 кГц или используйте **Fast** режим **I2C**, если он поддерживается **slave**-ом.

2.6.5 Data valid time ($t_{VD;DAT}$) violated without the OVR flag being set
Время валидности данных ($t_{VD;DAT}$) нарушается установкой флага OVR

Description (Описание)

The data valid time ($t_{VD;DAT}$, $t_{VD;ACK}$) described by the I²C standard can be violated (as well as the maximum data hold time of the current data ($t_{HD;DAT}$)) under the conditions described below. This violation cannot be detected because the OVR flag is not set (no transmit buffer underrun is detected).
Время валидности данных ($\mathbf{t_{VD;DAT}}$, $\mathbf{t_{VD;ACK}}$), описанное в стандарте **I2C**, может быть нарушено (так же как и максимальное время удержания текущих данных ($\mathbf{t_{HD;DAT}}$)) при условиях, описанных ниже. Это нарушение не может быть обнаружено, так как флаг **OVR** не ставится (никакой ошибки **underrun** буфера передачи не обнаруживается).

This issue can occur only under the following conditions:
Проблема может произойти только в следующей конфигурации:

- in Slave transmit mode (в режиме **Slave** передатчик)
- with clock stretching disabled (NOSTRETCH=1)
при отключенном затягивании тактового (**NOSTRETCH=1**)
- if the software is late to write the DR data register, but not late enough to set the OVR flag (the data register is written before)
если программа опаздывает с записью регистра данных **DR**, но еще не запоздала с установкой флага **OVR** (регистр данных был записан прежде)

Workarounds (Как обойти)

If the master device allows it, use the clock stretching mechanism by programming the bit NOSTRETCH=0 in the I2C_CR1 register.
Если **Master** позволяет это, то используйте механизм затягивания тактового (**NOSTRETCH=0** в регистре **I2C_CR1**).

If the master device does not allow it, ensure that the software is fast enough when polling the TXE or ADDR flag to immediately write to the DR data register. For instance, use an interrupt on the TXE or ADDR flag and boost its priority to the higher level.
Если **Master** не позволяет это, убедитесь, что программное обеспечение работает достаточно быстро, когда опрашивает флаги **TXE** или **ADDR**, чтобы немедленно записать в регистр данных **DR**. Например, используйте прерывание от флага **TXE** или **ADDR** и повысьте его приоритет до максимального.

2.7 SPI peripheral (SPI интерфейс)

2.7.1 CRC still sensitive to communication clock when SPI is in slave mode even with NSS high
Блок CRC чувствителен к тактовому, когда SPI находится в режиме slave и высокий уровень на NSS

Description (Описание)

When the SPI is configured in slave mode with the CRC feature enabled, the CRC is calculated even if the NSS pin deselects the SPI (high level applied on the NSS pin).
Когда **SPI** сконфигурирован в **slave** режим с включенным расчетом **CRC**, то **CRC** вычисляется даже тогда, когда **SPI** является не выбранным (высокий уровень на выводе **NSS**).

Workarounds (Как обойти)

The CRC has to be cleared on both Master and Slave sides between the slave deselection (high level on NSS) and the slave selection (low level on NSS), in order to resynchronize the Master and Slave for their respective CRC calculation.
CRC должен быть очищен как на стороне **Master**-а, так и на стороне и **Slave**-а, между событием отключения **Slave**-а (высокий уровень на **NSS**) и его выбора (низкий уровень на **NSS**), чтобы повторно синхронизировать **Master** со **Slave**-ом для корректного вычисления **CRC**.

To procedure to clear the CRC is the following:
Чтобы очистить **CRC**, выполните следующую процедуру:

1. disable the SPI (SPE = 0) (отключите **SPI** (**SPE=0**))
2. clear the CRCEN bit (очистите бит **CRCEN**)
3. set the CRCEN bit (поставьте бит **CRCEN**)
4. enable the SPI (SPE = 1) (включите **SPI** (**SPE=1**))

2.8 General-purpose timers (Таймеры общего назначения)

2.8.1 Missing capture flag (Потеря флага захвата)
2.8.2 Overcapture detected too early (Флаг **Overcapture** обнаруживается слишком рано)
2.8.3 General-purpose timer: regulation for 100% PWM
Поведение таймера при 100% заполнении **PWM**

2.8.1 Missing capture flag (Потеря флага захвата)

Description (Описание)

In capture mode, when a capture occurs while the CCRx register is being read, the capture flag (CCxIF) may be cleared without the overcapture flag (CCxOF) being set. The new data are actually captured in the capture register.
В режиме захвата, когда происходит захват в то же время, как читается регистр **CCRx**, флаг захвата **CCxIF** может быть очищен без установки флага сверхзахвата **CCxOF**. При этом новые данные реально захвачены в регистре захвата.

Workarounds (Как обойти)

An external interrupt can be enabled on the capture I/O just before reading the capture register (in the capture interrupt), and disabled just after reading the captured data. Possibly, a missed capture will be detected by the EXTI peripheral.
Можно разрешить внешнее прерывание на захват от **I/O** прямо перед чтением регистра захвата (в обработчике захвата), и запретить его сразу после чтения захваченных данных. Возможно, пропущенный захват будет обнаружен модулем **EXTI**.

2.8.2 Overcapture detected too early
Флаг Overcapture обнаруживается слишком рано

Description (Описание)

In capture mode, the overcapture flag (CCxOF) can be set even though no data have been lost.
В режиме захвата может быть установлен флаг сверхзахвата (**CCxOF**), хотя никакие данные не были утеряны.

Conditions (Условия)

If a capture occurs while the capture register is being read, an overcapture is detected even though the previously captured data are correctly read and the new data are correctly stored into the capture register.
Если захват происходит в то время, когда читается регистр захвата, детектируется сверхзахват даже тогда, когда ранее захваченные данные уже корректно прочитаны, а новые данные корректно сохраняются в регистре захвата.

The system is at the limit of an overcapture but no data are lost.
Система находится в состоянии сверхзахвата, но никакие данные не потеряны.

Workarounds (Как обойти)

None. (Никак.)

2.8.3 General-purpose timer: regulation for 100% PWM
Поведение таймера при 100% заполнении PWM

Description (Описание)

When the OCREF_CLR functionality is activated, the OCxREF signal becomes de-asserted (and consequently OCx is deasserted / OCxN is asserted) when a high level is applied on the OCREF_CLR signal. The PWM then restarts (output re-enabled) at the next counter overflow.
Когда активизирована функция **OCREF_CLR**, сигнал **OCxREF** снимается (и, следовательно, **OCx** - снимается / **OCxN** ставится), когда появляется высокий уровень на сигнале **OCREF_CLR**. Затем перезапускается **PWM** (выход повторно разрешается) при следующем переполнении счетчика.

But if the PWM is configured at 100% (CCxR > ARR), then it does not restart and OCxREF remains de-asserted.
Но, если **PWM** сконфигурирован на 100 % (**CCxR > ARR**), то перезапуска не происходит и **OCxREF** остается снятым.

Workarounds (Как обойти)

None. (Никак.)

2.9 LSI clock stabilization time
Время стабилизации тактового LSI

Description (Описание)

When the LSIRDY flag is set, the clock may still be out of the specified frequency range f_{LSI} parameter, see LSI oscillator characteristics in the product datasheet).
Когда установлен флаг **LSIRDY**, тактовый сигнал может быть все еще вне указанного частотного диапазона (параметр $\mathbf{f_{LSI}}$, см. особенности генератора **LSI** в **datasheet**-те на кристалл).

Workarounds (Как обойти)

To have a fully stabilized clock in the specified range, a software temporization of 100 µs should be added.
Чтобы иметь полностью устойчивый тактовый сигнал в указанном диапазоне, в программе должна быть добавлена задержка в 100 µs.

2.10 PLL not locking when sourced by HSI/2 after reset if it was previously sourced by HSE with predivider >1 or PLL2
PLL не защелкивается

Description (Описание)

The limitation occurs when the sequence below is followed:
Есть ограничение, когда происходит последовательность, приведенная ниже:

- PLL source: HSI/2, SYSCLK source: PLL
Источник **PLL**: **HSI/2**, источник **SYSCLK**: **PLL**
- PLL source: HSE with predivider >1 or PLL2, SYSCLK source: PLL
Источник **PLL**: **HSE** с пред-делителем более 1 или **PLL2**, источник **SYSCLK**: **PLL**
- system reset (сброс системы)
- PLL source: HSI/2, SYSCLK source: PLL
Источник **PLL**: **HSI/2**, источник **SYSCLK**: **PLL**

The PLL cannot be locked when sourced by HSI/2 after applying system reset if it was previously sourced by HSE with predivider >1 or by PLL2.
PLL не может сцепиться, когда, после сброса системы, тактовый на него подается от **HSI/2**, если ранее он питался от **HSE** с пред-делителем более 1, или от **PLL2**.

Workarounds (Как обойти)

Enable the HSE oscillator and let the PLL lock on it before switching the PLL source to HSI/2.
Разрешите **HSE** генератор и позвольте **PLL** соединиться, прежде чем переключить источник **PLL** на **HSI/2**.

2.11 OTG_FS

2.11.1 Data in RxFIFO are overwritten when all channels are disabled simultaneously
Данные в **RxFIFO** перезаписываются, когда все каналы отключаются одновременно
2.11.2 OTG host blocks the receive channel when receiving IN packets and no TxFIFO is configured (Хост **OTG** блокирует канал приема при приеме **IN** пакета, когда **TxFIFO** не сконфигурирован)
2.11.3 Host channel-halted interrupt not generated when the channel is disabled
Хост не генерирует прерывание от остановки канала
2.11.4 Error in software-read OTG_FS_DCFG register values
Ошибка чтения **OTG_FS_DCFG** регистра

2.11.1 Data in RxFIFO are overwritten when all channels are disabled simultaneously
Данные в RxFIFO перезаписываются, когда все каналы отключаются одновременно

Description (Описание)

If the available RxFIFO is just large enough to host 1 packet + its data status, and is currently occupied by the last received data + its status and, at the same time, the application requests that more IN channels be disabled, the OTG_FS peripheral does not first check for available space before inserting the disabled status of the IN channels. It just inserts them by overwriting the existing data payload.
Если доступный буфер **RxFIFO** является достаточно большим, чтобы разместить только 1 пакет + его статус, и, в настоящее время, он занят последними полученным пакетом и, в то же самое время, приложение просит, чтобы в дальнейшем **IN** каналы были отключены, то модуль **OTG_FS** не делает предварительной проверки на наличие доступного места, прежде чем выставить статус "отключен" для **IN** каналов. Это просто вставляет его, переписывая существующие полезные данные.

Workarounds (Как обойти)

Use one of the following recommendations:
Используйте одну из следующих рекомендаций:

1. Configure the RxFIFO to host a minimum of 2 × MPSIZ + 2 × data status entries.
Конфигурируйте **RxFIFO** так, чтобы размещать как минимумом 2 пакета размером **MPSIZ** с их статусом.

2. The application has to check the RXFLVL bit (RxFIFO non-empty) in the OTG_FS_GINTSTS register before disabling each IN channel. If this bit is not set, then the application can disable an IN channel at a time. Each time the application disables an IN channel, however, it first has to check that the RXFLVL bit = 0 condition is true.
Приложение должно проверить бит **RXFLVL** (непустой **RxFIFO**) в регистре **OTG_FS_GINTSTS** перед отключением каждого **IN** канала. Если этот бит не установлен, то приложение может отключить **IN** канал сразу. Однако, каждый раз, когда приложение отключает **IN** канал, оно должно сначала проверить, что выполняется условие **RXFLVL=0**.

2.11.2 OTG host blocks the receive channel when receiving IN packets and no TxFIFO is configured
Хост OTG блокирует канал приема при приеме IN пакета, когда TxFIFO не сконфигурирован

Description (Описание)

When receiving data, the OTG_FS core erroneously checks for available TxFIFO space when it should only check for RxFIFO space. If the OTG_FS core cannot see any space allocated for data transmission, it blocks the reception channel and no data are received.
Получая данные, ядро **OTG_FS** ошибочно проверяет **TxFIFO** на доступное место, хотя должно проверить только **RxFIFO**. Если ядро **OTG_FS** не видит какого либо места, выделенного для передачи данных, это блокирует канал приема, и никакие данные не принимаются.

Workarounds (Как обойти)

Set at least one TxFIFO equal to the maximum packet size. In this way, the host application, which intends to supports only IN traffic, also has to allocate some space for the TxFIFO.
Установите по крайней мере один **TxFIFO** буфер, равный максимальному размеру пакета. Таким образом, приложение на Хосте, которое предназначено для поддержки только входного трафика, также должно выделить некоторое место для **TxFIFO** буфера.

Since a USB host is expected to support any kind of connected endpoint, it is good practice to always configure enough TxFIFO space for OUT endpoints.
Поэтому, хорошей практикой будет, когда **USB** Хост поддерживает любой тип подключенных конечных точек, и всегда имеет достаточно **TxFIFO** пространства для **OUT** конечных точек.

2.11.3 Host channel-halted interrupt not generated when the channel is disabled
Хост не генерирует прерывание от остановки канала

Description (Описание)

When the application enables, then immediately disables the host channel before the OTG_FS host has had time to begin the transfer sequence, the OTG_FS core, as a host, does not generate a channel-halted interrupt. The OTG_FS core continues to operate normally.
Когда приложение разрешает, а затем немедленно запрещаает Хост канал прежде, чем у **OTG_FS** Хоста было время, чтобы начать передачу, то ядро **OTG_FS**, как хозяин, не генерирует прерывание от остановки канала. Ядро **OTG_FS** продолжает работать нормально.

Workarounds (Как обойти)

Do not disable the host channel immediately after enabling it.
Не отключайте Хост канал хозяина сразу после его разрешения.

2.11.4 Error in software-read OTG_FS_DCFG register values
Ошибка чтения OTG_FS_DCFG регистра

Description (Описание)

When the application writes to the DAD and PFIVL bitfields in the OTG_FS_DCFG register, and then reads the newly written bitfield values, the read values may not be correct.
Когда приложение делает запись в битовые поля **DAD** и **PFIVL** в регистре **OTG_FS_DCFG**, и затем читает эти недавно записанные битовые поля, прочитанные значения могут быть не верными.

The values written by the application, however, are correctly retained by the core, and the normal operation of the device is not affected.
Однако, значения, записанные приложением, правильно сохранены ядром, и нормальная работа устройства не затронута.

Workarounds (Как обойти)

Do not read from the OTG_FS_DCFG register's DAD and PFIVL bitfields just after programming them.
Не читайте битовые поля **DAD** и **PFIVL** в регистре **OTG_FS_DCFG** сразу после их программирования.

2.12 Ethernet MAC

2.12.1 Possible underflow when TxFIFO is configured in Store-and-Forward mode and a relatively large frame is aborted in Half-duplex mode
(Возможно **underflow**, когда **TxFIFO** сконфигурирован в режиме **Store-and-Forward** и относительно большой фрейм отвергнут в режиме **Half-duplex**)
2.12.2 Possible CRC error when TxFIFO is configured in Store-and-Forward mode and a relatively large frame is aborted in Half-duplex mode with transmit checksum offload enabled (Возможна ошибка **CRC**, когда **TxFIFO** сконфигурирован в режиме **Store-and-Forward** и относительно большой фрейм отвергнут в режиме **Half-duplex**, при разрешенном расчете **CRC**)
2.12.3 Erroneous automatic checksum insertion after TxFIFO is dynamically switched from Threshold to Store-and-Forward mode
Ошибка автоматически вставляемой **CRC** после динамического переключения **TxFIFO**
2.12.4 Erroneous automatic checksum insertion after a large frame (longer than the TxFIFO) transmission (Ошибка автоматически вставляемой **CRC** после передачи большого фрейма (больше чем **TxFIFO**))
2.12.5 In half-duplex mode, the MAC transmitter ignores collisions after it is disabled during a frame transmission (В режиме **Half-duplex** передатчик **MAC** игнорирует коллизии после того, как он отключается во время передачи фрейма)
2.12.6 Interrupt due to an RMON (MMC) counter may be set again after it is cleared
(Прерывание из-за того, что счетчик **RMON** (**MMC**) может быть установлен снова, после того, как он очищен)
2.12.7 Incorrect layer 3 (L3) checksum is inserted in transmitted IPv6 packets without TCP, UDP or ICMP payloads (Вставка некорректной контрольных суммы **L3** (слой 3) в передаваемый пакет **IPv6** без без полезной нагрузки **TCP**, **UDP** или **ICMP**)
2.12.8 The Ethernet MAC processes invalid extension headers in the received IPv6 frames
(**Ethernet MAC** обрабатывает недействительные расширенные заголовки в полученных структурах **IPv6**)
2.12.9 MAC stuck in the Idle state on receiving the TxFIFO flush command exactly 1 clock cycle after a transmission completes (**MAC** вставляет состояние и при получении команды сброса потока **TxFIFO**, точно через 1 такт после завершения передачи)
2.12.10 Transmit frame data corruption (Разрушение данных фрейма передачи)

2.12.1 Possible underflow when TxFIFO is configured in Store-and-Forward mode and a relatively large frame is aborted in Half-duplex mode
Возможно underflow, когда TxFIFO сконфигурирован в режиме Store-and-Forward и относительно большой фрейм отвергнут в режиме Half-duplex

Description (Описание)

In Store-and-Forward mode, the frame is transferred to the MAC only when the full frame is available in the TxFIFO. There is one exception to this rule, when the TxFIFO is almost full.
В режиме **Store-and-Forward** фрейм передается на **MAC** только тогда, когда в **TxFIFO** доступен весь фрейм. Из этого правила есть одно исключение, когда **TxFIFO** почти полон.

This problem may arise when a relatively large frame is loaded into the TxFIFO and a second frame is partially loaded (because of lack of space) into the free TxFIFO space, generating an almost full condition. In this case, if the first frame is aborted, due to a carrier loss, the absence of a carrier or a late collision, the second frame is sent to the MAC before it is completely loaded into the TxFIFO. If the DMA bandwidth does not allow the rest of the frame to be uploaded before the end of the frame transmission, an underflow may occur.
Эта проблема может возникнуть, когда относительно большой фрейм загружен в **TxFIFO**, и частично загружен второй фрейм (из-за нехватки места) в свободное место **TxFIFO**, что

генерирует условие - "почти полный". В этом случае, если первый фрейм прерывается (*aborted*), из-за потери "курьера", отсутствия "курьера" или последней коллизии, то второй фрейм посылается на **MAC** прежде, чем это будет полностью загружено в **TxFIFO**. Если полоса пропускания **DMA** не позволяет дозагрузить остальную часть фрейма до окончания его передачи, то может случиться недостача (*underflow*).

In this case, the software must detect the underflow condition and resend the frame.
В этом случае, программа должна обнаружить условие **underflow** и повторно послать фрейм.

Workarounds (Как обойти)

Resend the frame on underflow detection.
Повторно пошлите фрейм при обнаружении **underflow**.

2.12.2 Possible CRC error when TxFIFO is configured in Store-and-Forward mode and a relatively large frame is aborted in Half-duplex mode with transmit checksum offload enabled

Возможна ошибка CRC, когда TxFIFO сконфигурирован в режиме Store-and-Forward и относительно большой фрейм отвергнут в режиме Half-duplex, при разрешенном расчете CRC

Description (Описание)

In Store-and-Forward mode, the frame is transferred to the MAC only when the full frame is available in the TxFIFO. There is one exception to this rule, when the TxFIFO is almost full.
В режиме **Store-and-Forward** фрейм передается на **MAC** только тогда, когда в **TxFIFO** доступен весь фрейм. Из этого правила есть одно исключение, когда **TxFIFO** почти полон.

This problem may arise when a relatively large frame is loaded into the TxFIFO and a second frame is partially loaded (because of lack of space) into the free TxFIFO space, generating an almost full condition. In this case, if the first frame is aborted, due to a carrier loss, the absence of a carrier or a late collision, the second frame is sent to the MAC before it is completely loaded into the TxFIFO. Since for CRC computation, the entire frame must be available in the TxFIFO, the CRC of the subsequent frames will be erroneous.
Эта проблема может возникнуть, когда относительно большой фрейм загружен в **TxFIFO**, и частично загружен второй фрейм (из-за нехватки места) в свободное место **TxFIFO**, что генерирует условие - "почти полный". В этом случае, если первый фрейм прерывается (*aborted*), из-за потери "курьера", отсутствия "курьера" или последней коллизии, то второй фрейм посылается на **MAC** прежде, чем это будет полностью загружено в **TxFIFO**. Так как вычисляется **CRC**, то весь фрейм должен быть доступен в **TxFIFO**, и **CRC** последующих фреймов будет некорректным.

TCP/IP checksum errors may be detected at the remote end, causing data to be dropped.
Ошибки контрольной суммы **TCP/IP** могут быть обнаружены на другом конце, что вызовет потерю данных.

Workarounds (Как обойти)

Wait for the TxFIFO to become empty and then send the txFIFO flush command.
Ждите, пока **TxFIFO** не станет пустым, а затем пошлите команду сброса (*flush*) **txFIFO**.

2.12.3 Erroneous automatic checksum insertion after TxFIFO is dynamically switched from Threshold to Store-and-Forward mode
Ошибка автоматически вставляемой CRC после динамического переключения TxFIFO

Description (Описание)

Automatic checksum insertion in transmitted frames can be enabled (through the CIC bits in the TDES0 register) and used only when the TxFIFO is configured to operate in Store-and-Forward mode. When the TxFIFO is operated in Threshold mode, the CIC bits are ignored.
Может быть разрешена автоматическая вставка **CRC** в передаваемые фреймы (бит **CIC** в регистре **TDES0**) которая будет действовать только тогда, когда **TxFIFO** сконфигурирован на работу в режиме **Store-and-Forward**. Когда **TxFIFO** работает в режиме **Threshold**, бит **CIC** игнорируется.

When the TxFIFO dynamically switches from Threshold to Store-and-Forward mode, the CRC computation may be erroneous on the subsequent frames.
Когда **TxFIFO** динамически переключается из режима **Threshold** в **Store-and-Forward**, то вычисленный **CRC** может быть ошибочным для последующих фреймов.

This results in the detection of TCP/IP checksum errors at the remote end, causing data to be dropped.
Это приводит к детектированию ошибки **CRC TCP/IP** на другом конце, что вызовет потерю данных.

Workarounds (Как обойти)

Use the Store-and-Forward mode only. (Используйте только режим **Store-and-Forward**.)

2.12.4 Erroneous automatic checksum insertion after a large frame (longer than the TxFIFO) transmission
Ошибка автоматически вставляемой CRC после передачи большого фрейма (больше чем TxFIFO)

Description (Описание)

Automatic checksum insertion in transmitted frames can be enabled (through the CIC bits in TDES0) and used only for frames with a length lesser than the TxFIFO depth.
Может быть разрешена автоматическая вставка **CRC** в передаваемые фреймы (бит **CIC** в регистре **TDES0**) которая будет действовать только тогда, когда длина фрейма меньше глубины **TxFIFO**.

If a long frame is transmitted, with checksum insertion disabled (CIC=00), immediately followed by a short frame, with checksum insertion enabled, the computed checksum may be erroneous. This is due to the fact that the second frame may have be transmitted too early from the TxFIFO, due to a TxFIFO almost full condition.
Если передан длинный фрем, с отключенной вставкой **CRC** (**CRC**), после которого сразу передается короткий фрейм, с разрешенной вставкой **CRC**, то вычисленная **CRC** может быть ошибочной. Это происходит вследствие того, что второй фрейм может быть передан слишком рано из **TxFIFO**, так как этот **TxFIFO** соответствует условию - "почти полный".

This results in the detection of TCP/IP checksum errors at the remote end, causing data to be dropped. Это приводит к детектированию ошибки **CRC TCP/IP** на другом конце, что вызовет потерю данных.

Workarounds (Как обойти)

1. Avoid enabling automatic checksum insertion for any frame if your system transmits frames with a size larger than the depth of the TxFIFO.
Избегайте разрешения автоматической вставки **CRC** для любого фрейма, если ваша система может передавать фреймы, размер которых превышает глубину **TxFIFO**.

2. Wait for the long frame transmission completion before re-enabling the automatic checksum insertion.
Ждите завершения передачи длинного фрема, прежде чем повторно разрешить автоматическую вставку контрольной суммы.

2.12.5 In half-duplex mode, the MAC transmitter ignores collisions after it is disabled during a frame transmission
В режиме Half-duplex передатчик MAC игнорирует коллизии после того, как он отключается во время передачи фрейма

Description (Описание)

If the TE bit is cleared during a transmission, the transmission part of the MAC is effectively disabled after the complete transmission of the current frame. From the clearing of the TE bit until the end of the transmission, potential collisions are ignored.
Если, во время передачи, очистить бит **TE**, то часть передачи на **MAC** эффективно отключается после полной передачи текущего фрейма. В течении времени от очистки бита **TE** до конца передачи потенциальные коллизии игнорируются.

If a collision event occurs after the MAC is disabled, the transmitter does not recognize the event and continues to transmit the complete frame without any JAM pattern. It reports a successful frame transmission status (without any collision) even though the frame is corrupted at the remote receivers due to collision.
Если событие коллизии происходит после того, как **MAC** отключен, то передатчик не распознает это событие и продолжит работу с передачи полного фрейма без какого либо **JAM** паттерна. Это сообщает об успешном статусе передачи фрейма (без коллизий) даже при том, что фрейм был разушен на стороне приемника из-за наличия коллизии.

Since a JAM signal might not be sent during a collision, frames may be lost when a remote transmitter does not detect that a collision occurred.
Так как сигнал **JAM** не может быть послан во время коллизии, фреймы могут быть потеряны, когда удаленный передатчик не обнаруживает, что была коллизия.

Workarounds (Как обойти)

Disable the MAC transmitter only after the completion of the transmission of all scheduled frames in Half-duplex mode.
Отключайте **MAC** передатчик только после завершения передачи всех запланированных фреймов в режиме **Half-duplex**.

2.12.6 Interrupt due to an RMON (MMC) counter may be set again after it is cleared
Прерывание из-за того, что счетчик RMON (MMC) может быть установлен снова, после того, как он очищен

Description (Описание)

When enabled, an interrupt asserted due to an RMON (remote monitoring) counter becoming full, is cleared when the corresponding counter is read. The counter is also cleared by the read operation if bit 1 (counter stop rollover) in ETH_MMCCR is set. If this clear signal coincides with the counter update signal generated by the presence of a new frame, then the corresponding interrupt bit is set again. This results in the software getting a spurious interrupt from the MMC even though the corresponding counter value is very low.
Когда разрешено прерывание, то его флаг выставляется, когда счетчик **RMON** (*Remote Monitoring*) становится полным, и очищается, когда чтением соответствующего счетчика. Счетчик также очищается операцией чтения, если стоит бит ? (*counter stop rollover*) в регистре **ETH_MMCCR**. Если этот сигнал очистки совпадает с сигналом обновления счетчика, генерируемым наличием нового фрейма, то соответствующий бит прерывания ставится снова. Это приводит к тому, что программа получает ложное прерывание от **MMC**, хотя значение в соответствующем счетчике очень низкое.

Workarounds (Как обойти)

None. (Никак.)

2.12.7 Incorrect layer 3 (L3) checksum is inserted in transmitted IPv6 packets without TCP, UDP or ICMP payloads
Вставка некорректной контрольных суммы L3 (слой 3) в передаваемый пакет IPv6 без полезной нагрузки TCP, UDP или ICMP

Description (Описание)

The application provides the per-frame control to instruct the MAC to insert the L3 checksums for TCP, UDP and ICMP packets. When automatic checksum insertion is enabled and the input packet is an IPv6 packet without the TCP, UDP or ICMP payload, then the MAC may incorrectly insert a checksum into the packet. For IPv6 packets without a TCP, UDP or ICMP payload, the MAC core considers the next header (NH) field as the extension header and continues to parse the extension header. Sometimes, the payload data in such packets matches the NH field for TCP, UDP or ICMP and, as a result, the MAC core inserts a checksum.
Приложение обеспечивает предварительный управляющий фрейм, чтобы проинструктировать **MAC**, чтобы он вставлял контрольную сумму **L3** для пакетов **TCP**, **UDP** и **ICMP**. Когда разрешена автоматическая вставка контрольной суммы, и входным пакетом является **IPv6** без полезной нагрузки **TCP**, **UDP** или **ICMP**, тогда **MAC** может некорректно вставить контрольную сумму в пакет. Для пакетов **IPv6** без полезной нагрузки **TCP**, **UDP** или **ICMP**, ядро **MAC** будет рассматривать поле следующего заголовка (**NH**), как расширенный заголовок, и продолжит анализ уже расширенного заголовка. Иногда, полезные данные в таких пакетах соответствуют области **NH** для **TCP**, **UDP** или **ICMP**, и, в результате, ядро **MAC** вставляет контрольную сумму.

Workarounds (Как обойти)

When the IPv6 packets have a TCP, UDP or ICMP payload, enable checksum insertion for transmit frames, or bypass checksum insertion by using the CIC (checksum insertion control) bits in TDES0 (bits 23:22).

Когда пакеты **IPv6** имеют полезную нагрузку **TCP**, **UDP** или **ICMP**, разрешите вставку контрольной суммы в передаваемый фрейм, или обойдите вставку контрольной суммы с помощью бита **CIC** (*Checksum Insertion Control*) в регистре **TDES0** (биты 23:22).

2.12.8 The Ethernet MAC processes invalid extension headers in the received IPv6 frames
Ethernet MAC обрабатывает недействительные расширенные заголовки в полученных структурах IPv6

Description (Описание)

In IPv6 frames, there can be zero or some extension headers preceding the actual IP payload. The Ethernet MAC processes the following extension headers defined in the IPv6 protocol: Hop-by-Hop Options header, Routing header and Destination Options header. All extension headers except the Hop-by-Hop extension header can be present multiple times and in any order before the actual IP payload. The Hop-by-Hop extension header, if present, has to come immediately after the IPv6's main header.
Во фреймах **IPv6** могут быть расширенные заголовки, которые предшествуют фактическим полезным данным **IP**. **Ethernet MAC** обрабатывает следующие расширенные заголовки, определенные в протоколе **IPv6**: **Hop-by-Hop Options** заголовок, **Routing** заголовок и **Destination Options** заголовок. Все расширенные заголовки, кроме **Hop-by-Hop**, могут присутствовать многократно и в любом порядке перед фактическим полезными данными **IP**. Расширенный заголовок **Hop-by-Hop**, если есть, должен быть сразу после главного заголовка **IPv6**.

The Ethernet MAC processes all (valid or invalid) extension headers including the Hop-by-Hop extension headers that are present after the first extension header. For this reason, the GMAC core will accept IPv6 frames with invalid Hop-by-Hop extension headers. As a consequence, it will accept any IP payload as valid IPv6 frames with TCP, UDP or ICMP payload, and then incorrectly update the Receive status of the corresponding frame.
MAC Ethernet обрабатывает все (действительные или недействительные) расширенные заголовки, включая **Hop-by-Hop**, которые присутствуют после первого расширенного заголовка. Поэтому ядро **GMAC** примет фрейм **IPv6** с недействительными заголовком **Hop-by-Hop**. Как следствие, он примет любые полезные данные **IP**, как действительные фреймы **IPv6** с полезными данными **TCP**, **UDP** или и, а затем некорректно обновит статус приема соответствующего фрейма.

Workarounds (Как обойти)

None. (Никак.)

2.12.9 MAC stuck in the Idle state on receiving the TxFIFO flush command exactly 1 clock cycle after a transmission completes
MAC вставляет состояние Idle при получении команды сброса потока TxFIFO, точно через 1 такт после завершения передачи

Description (Описание)

When the software issues a TxFIFO flush command, the transfer of frame data stops (even in the middle of a frame transfer). The TxFIFO read controller goes into the Idle state (TFRS=00 in ETH_MACDBGR) and then resumes its normal operation.
Когда программа выполняет команду сброса потока (*flush*) **TxFIFO**, передача фрейма данных останавливается (даже в середине передачи фрейма). Контроллер, читающий **TxFIFO**, входит в состояние **Idle** (поле **TFRS=00** в **ETH_MACDBGR**) и затем возобновляет нормальную работу.

However, if the TxFIFO read controller receives the TxFIFO flush command exactly oneclock cycle after receiving the status from the MAC, the controller remains stuck in the Idle state and stops transmitting frames from the TxFIFO. The system can recover from this state only with a reset (e.g. a soft reset).
Однако, если этот контроллер получает команду сброса и точно через один такт после получения статуса от **MAC**, то он остается в состоянии **Idle** и прекращает передавать данные фрейма из **TxFIFO**. Система (модуль) может вернуться от этого состояния только с помощью сброса (например, программного сброса).

Workarounds (Как обойти)

Do not use the TxFIFO flush feature. (Не используйте функцию сброса потока **TxFIFO**.)

If TXFIFO flush is really needed, wait until the TxFIFO is empty prior to using the TxFIFO flush command.
Если сброс потока **TXFIFO** действительно необходим, подождите, пока **TxFIFO** не станет пуст перед использованием команды сброса потока **TxFIFO**.

2.12.10 Transmit frame data corruption
Разрушение данных фрейма передачи

Description (Описание)

Frame data corrupted when the TxFIFO is repeatedly transitioning from non-empty to empty and then back to non-empty.
Разрушение данных фрейма, когда **TxFIFO** неоднократно переходит из непустого состояния в пустое и обратно.

Frame data may get corrupted when the TxFIFO is repeatedly transitioning from non-empty to empty for a very short period, and then from empty to non-empty, without causing an underflow.
Данные фрейма могут быть разрушены, когда **TxFIFO** неоднократно переходит из непустого состояния в пустое и обратно в течение очень короткого периода, не вызывая ошибки **underflow**.

This transitioning from non-empty to empty and back to non-empty happens when the rate at which the data are being written to the TxFIFO is almost equal to or a little less than the rate at which the data are being read.
Этот переход от непустого состояния в пустое и обратно случается, когда время, за которое данные пишутся в **TxFIFO**, почти равно, или немного меньше, чем время, за которое данные читаются.

This corruption cannot be detected by the receiver when the CRC is inserted by the MAC, as the corrupted data are used for the CRC computation.
Эта разрушение не может быть обнаружено приемником, когда **CRC** вставляется **MAC**-ом, поскольку для вычисления **CRC** используются эти разрушенные данные.

Workarounds (Как обойти)

Use the Store-and-Forward mode: TSF=1 (bit 21 in ETH_DMAOMR). In this mode the data are transmitted only when the whole packet is available in the TxFIFO.
Используйте режим **Store-and-Forward**: **TSF=1** (бит 21 в **ETH_DMAOMR**). В этом режиме данные передаются только тогда, когда весь пакет доступен в **TxFIFO**.

2.13 Boot loader unavailability on STM32F105xx and STM32F107xx devices with a date code below 937

Системный загрузчик недоступен для устройств STM32F105xx и STM32F107xx с кодом менее 937

Description (Описание)

During the boot loader activation phase, if the USART1_RX (PA10), USART2_RX (PD6, remapped pin), CAN2_Rx (PB5, remapped pin), OTG_FS_DM (PA11) and/or OTG_FS_DP (PA12) pin(s) are connected to low level or left floating, the boot loader cannot be used. It is not possible to connect to the boot loader through either of CAN2 (remapped), DFU (OTG FS in Device mode), USART1 or USART2 (remapped).
Во время фазы активации загрузчика (*bootloader*), если вывод(ы) **USART1_RX** (**PA10**), **USART2_RX** (**PD6**, **remapped**), **CAN2_Rx** (и, **remapped**), **OTG_FS_DM** (**PA11**) и/или **OTG_FS_DP** (**PA12**) притянуты к низкому уровню или оставлены в воздухе, то загрузчик не сможет работать. Невозможно соединиться с зарузчиком через любой из следующих интерфейсов **CAN2** (**remapped**), **DFU** (**OTG FS** в режиме **Device**), **USART1** или **USART2** (**remapped**).

In 64-pin packages, the USART2_RX remapped pin PD6 is not available and is internally grounded. Therefore, the bootloader cannot be used at all.
В 64-х выводных корпусах **remapped** вывод **PD6** для **USART2_RX** недоступен и внутренне соединен с "землей". Поэтому, **bootloader** не может использоваться вообще.

Workarounds (Как обойти)

- For 64-pin packages: none. The boot loader cannot be used.
Для 64-х выводных корпусов: никак. **Bootloader** недоступен.

- For 100-pin packages: depending on the used peripheral, the pins for the unused peripherals have to be kept at a high level during the boot loader activation phase as described below:
Для 100 выводных корпусов: в зависимости от используемой периферии, выводы для неиспользуемой периферии должны быть притянуты к высокому уровню во время фазы активации загрузчика как описано ниже:

    - If USART1 is used to connect to the boot loader: PD6 and PB5 have to be kept at a high level
    Если используется **USART1** для соединения с загрузчиком: то **PD6** и и **PD5** должны быть притянуты к высокому уровню

    - If USART2 is used to connect to the boot loader: PA10, PB5, PA11 and PA12 have to be kept at a high level
    Если используется **USART2** для соединения с загрузчиком: то **PA10**, **PB5**, **PA11** и **PA12** должны быть притянуты к высокому уровню

    - If CAN2 is used to connect to the boot loader: PA10, PD6, PA11 and PA12 have to be kept at a high level
    Если используется **CAN2**, чтобы соединиться с загрузчиком: то **PA10**, и, **PA11** и **PA12** должны быть притянуты к высокому уровню

    - If DFU is used to connect to the boot loader: PA10, PB5 and PD6 have to be kept at a high level
    Если используется **DFU**, чтобы соединиться с загрузчиком: то **PA10**, **PB5** и **PD6** должны быть притянуты к высокому уровню

***Note**:*

*This limitation concerns only STM32F105xx and STM32F107xx devices with a date code below 937. (Это ограничение касается только устройств **STM32F105xx** и **STM32F107xx** с кодом ниже 937.)*

STM32F105xx and STM32F107xx devices with a date code of 937 and above are not impacted.
*(На устройства **STM32F105xx** и **STM32F107xx** с кодом 937 и выше не действует.)*

See Appendix A: Revision and date codes on device marking for where to find the date code on the device marking.
(См. Приложение А: "Ревизии и коды на маркировке устройства" для того, где найти этот код.)

Appendix A Revision and date codes on device marking
Приложение А. "Ревизии и коды на маркировке устройства"

Figure 1 and Figure 2 show the marking compositions for the LQFP100 and LQFP64 packages, respectively. The only fields shown are the Additional field containing the revision code and the Year and Week fields making up the date code.
Рисунок 1 и 2 показывает структуру маркировки для корпусов **LQFP100** и **LQFP64**, соответственно. Только поле, именованное как "**Additional field containing the revision code**", показывает то место, где содержится код ревизии, а поля "**Year**" и "**Week**" дают место под код даты.

Figure 1. LQFP100 top package view (Вид маркировки корпуса **LQFP100**)

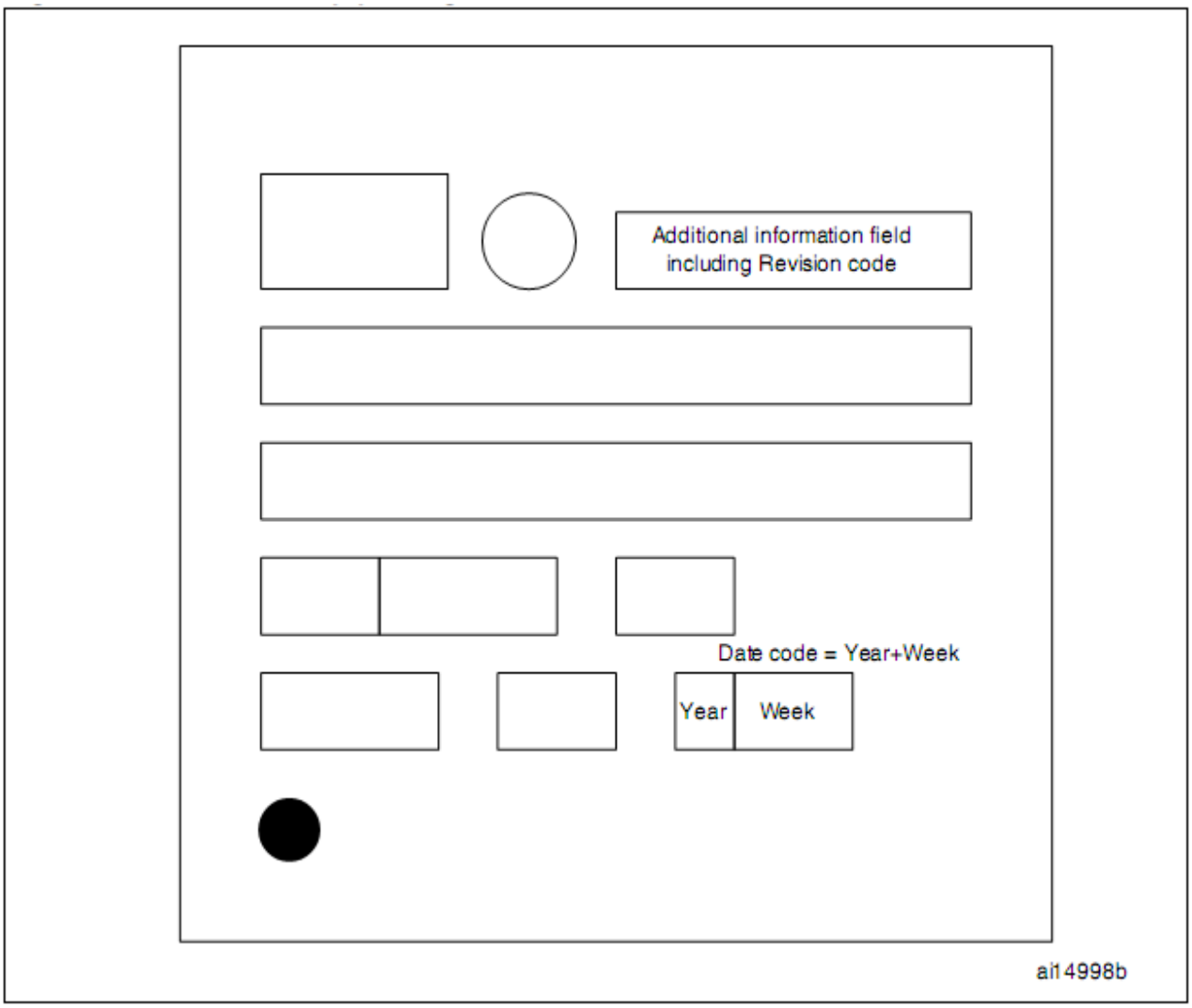
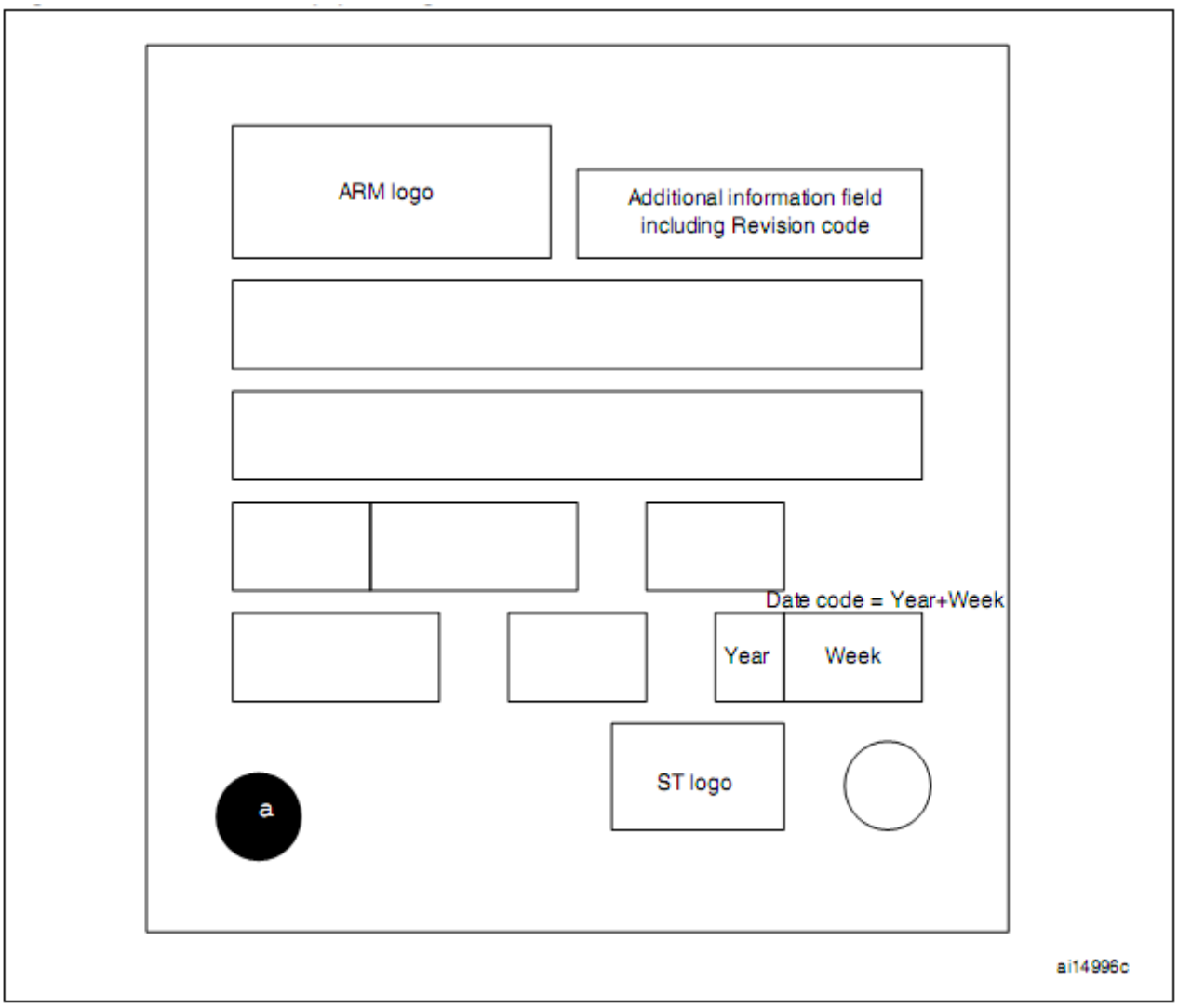


Figure 2. LQFP64 top package view (Вид маркировки корпуса **LQFP64**)



Revision history (История изменений)

Table 5. Document revision history (История изменений документа)

Date	Revision	Changes (Изменения)
16.06.2009	1	Initial release. (Первичный документ.)
16.10.2009	2	Section 2.3.5: I2S2 in master/slave mode and Ethernet/USART3 in synchronous mode added. Table 4: Summary of silicon limitations in revision Z devices modified. Section 2.11: OTG_FS added. Section 2.12: Ethernet MAC added. Section 2.13: Boot loader unavailability on STM32F105xx and STM32F107xx devices with a date code below 937 added. Figure 1: LQFP100 top package view and Figure 2: LQFP64 top package view updated to show the Date code. Добавлен раздел 2.3.5: "**I2S2** в режиме **master/slave** и **Ethernet/USART3** в синхронном режиме" Добавлена таблица 4: "Summary of silicon limitations in revision Z devices modified". (здесь опущена, так как дублирует содержание) Добавлен раздел 2.11: "**OTG_FS**". Добавлен раздел 2.12: "**Ethernet MAC**". Добавлен раздел 2.13: "Системный загрузчик недоступен для устройств **STM32F105xx** и **STM32F107xx** с кодом менее 937". Добавлен рис. 1: "Вид маркировки корпуса **LQFP100**" и рис. 2: "Вид маркировки корпуса **LQFP64**".
15.12.2009	3	Added limitations: – Section 2.3.6: USARTx_TX pin usage – Section 2.6.3: Start cannot be generated after a misplaced Stop – Section 2.6.4: Mismatch on the “Setup time for a repeated Start condition” timing parameter – Section 2.6.5: Data valid time (tVD;DAT) violated without the OVR flag being set – Section 2.7.1: CRC still sensitive to communication clock when SPI is in slave mode even with NSS high Добавлены ограничения: – Добавлен раздел 2.3.6: "Использование вывода **USARTx_TX**" – Добавлен раздел 2.6.3: "**Start** не может генерироваться после неуместного **Stop**" – Добавлен раздел 2.6.4: 'Несоответсвие параметра "Время установки для повторного старт-состояния"' – Добавлен раздел 2.6.5: "Время валидности данных ($t_{VD;DAT}$) нарушается установкой флага **OVR**" – Добавлен раздел 2.7.1: "Блок **CRC** чувствителен к тактовому, когда **SPI** находится в режиме **slave** и высокий уровень на **NSS**"