**CH32V003 Reference Manual**



V1.7

# Overview

Серия CH32V003 - это промышленные микроконтроллеры общего назначения, разработанные на основе 32-разрядного набора команд и архитектуры RISC-V. В них используется ядро QingKe V2A, набор команд RV32EC и поддерживается 2 уровня вложенности прерываний. Устройства этой серии оснащены широкими периферийными интерфейсами и функциональными модулями. Их внутренняя организационная структура соответствует сценариям встраиваемых приложений с низкой стоимостью и низким энергопотреблением.

Данное руководство содержит подробную информацию об использовании серии CH32V003 для разработки пользовательских приложений и применимо к продуктам данной серии с различным объемом памяти, функциональными ресурсами и пакетами; любые различия будут специально объяснены в соответствующих функциональных главах.

Пожалуйста, ознакомьтесь с техническими характеристиками данного устройства в спецификации *CH32V003DS0*. Для получения информации о ядре, пожалуйста, обратитесь к руководству *QingKeV2\_Processor\_Manual*.

## RISC-V обзор версии ядра

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Набор инструкций | кол-во аппаратных стеков | Уровни вложенности прерываний | Быстрые каналы прерывания | Линий потока (Flow Line) | Модель таблицы векторов (Vector table model) | Поддержка инструкций по подключению расширений | Интерфейс отладки |
| QingKe V2A | RV32EC | 2 | 2 | 2 | 2 | Адреса или команды | Поддерживаеся | 1-wire |

Abbreviated description of the bit attribute in the register:

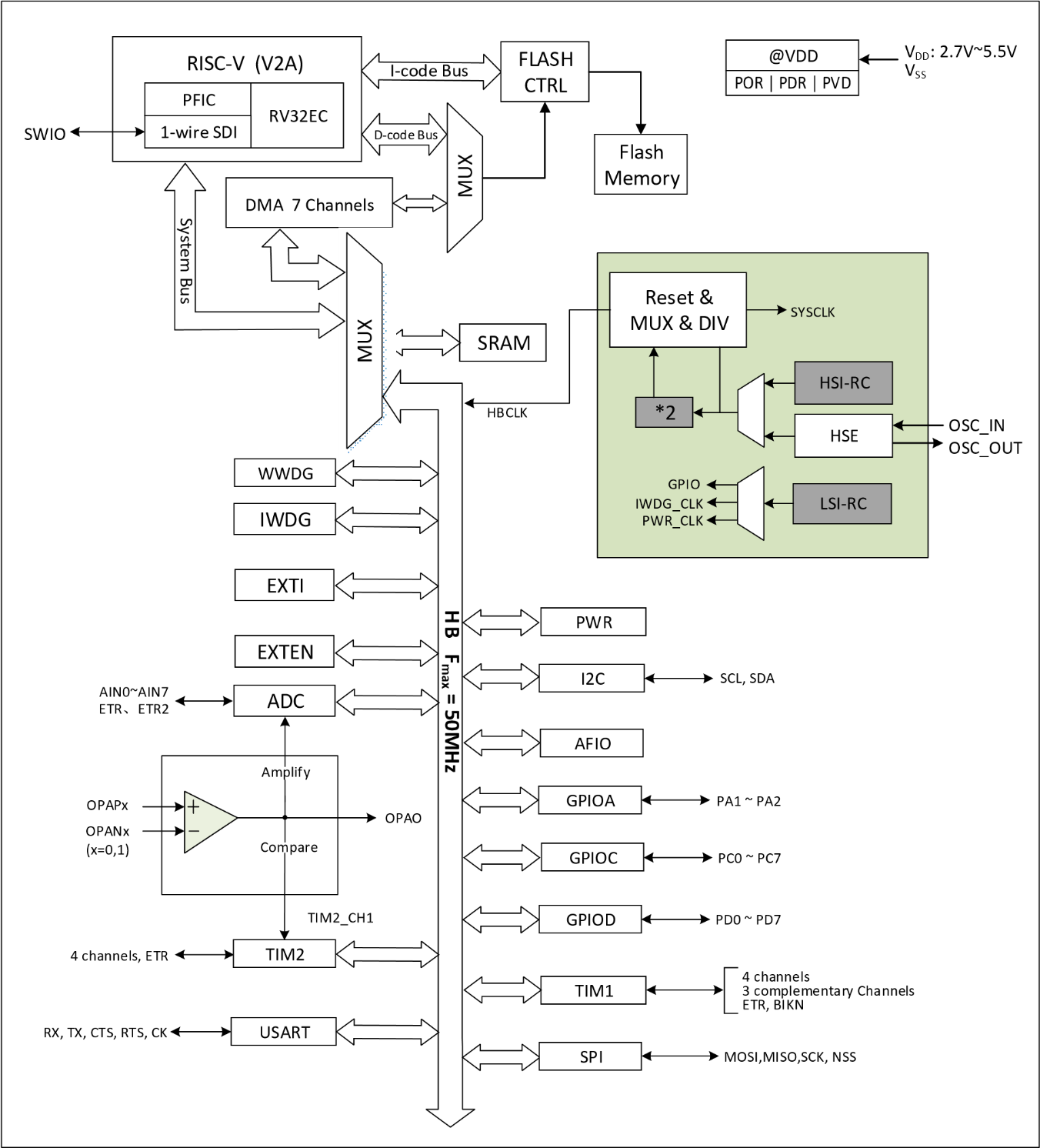
|  |  |
| --- | --- |
| Аббревиатура | Описание свойств |
| RF | Свойство только для чтения, которое считывает фиксированное значение. |
| RO | Атрибут, доступный только для чтения, изменяемый аппаратным обеспечением. |
| RZ | Свойство только для чтения, автоматическое удаление бита 0 после операции чтения. |
| WO | Атрибут только для записи (не читается, значение для чтения неопределенно) |
| WA | Атрибут, доступный только для записи в безопасном режиме. |
| WZ | Атрибут только для записи, автоматический сброс бита 0 после операции записи. |
| RW | Читаемый и доступный для записи. |
| RWA | Читаемый и доступный для записи в безопасном режиме. |
| RW1 | Читаемый, запись 1 допустима, запись 0 недопустима. |
| RW0 | Читаемый, запись 0 допустима, запись 1 недопустима. |
| RW1T | Читаемый, запись 0 недопустима, запись 1 перевернута. |

# Глава 1. Архитектура памяти и шины

## 1.1 Архитектура шины

Серия CH32V003 разработана на основе набора команд RISC-V, и ее архитектура обеспечивает взаимодействие ядра, модуля арбитража, модуля DMA, хранилища SRAM и других компонентов через несколько шин. В конструкцию встроен универсальный контроллер DMA для снижения нагрузки на процессор и повышения эффективности доступа, а также механизмы защиты данных, механизмы автоматического переключения тактовых импульсов и другие меры для повышения стабильности системы. Структурная схема системы показана на рисунке 1.1.

Рисунок 1-1 Структурная схема системы CH32V003



Система оснащена: универсальным контроллером DMA для снижения нагрузки на процессор и повышения эффективности; управлением древовидной иерархией тактирования для снижения общего энергопотребления периферийных устройств, а также механизмами защиты данных, механизмами защиты сторожевыми таймерами и другими мерами для повышения стабильности системы.

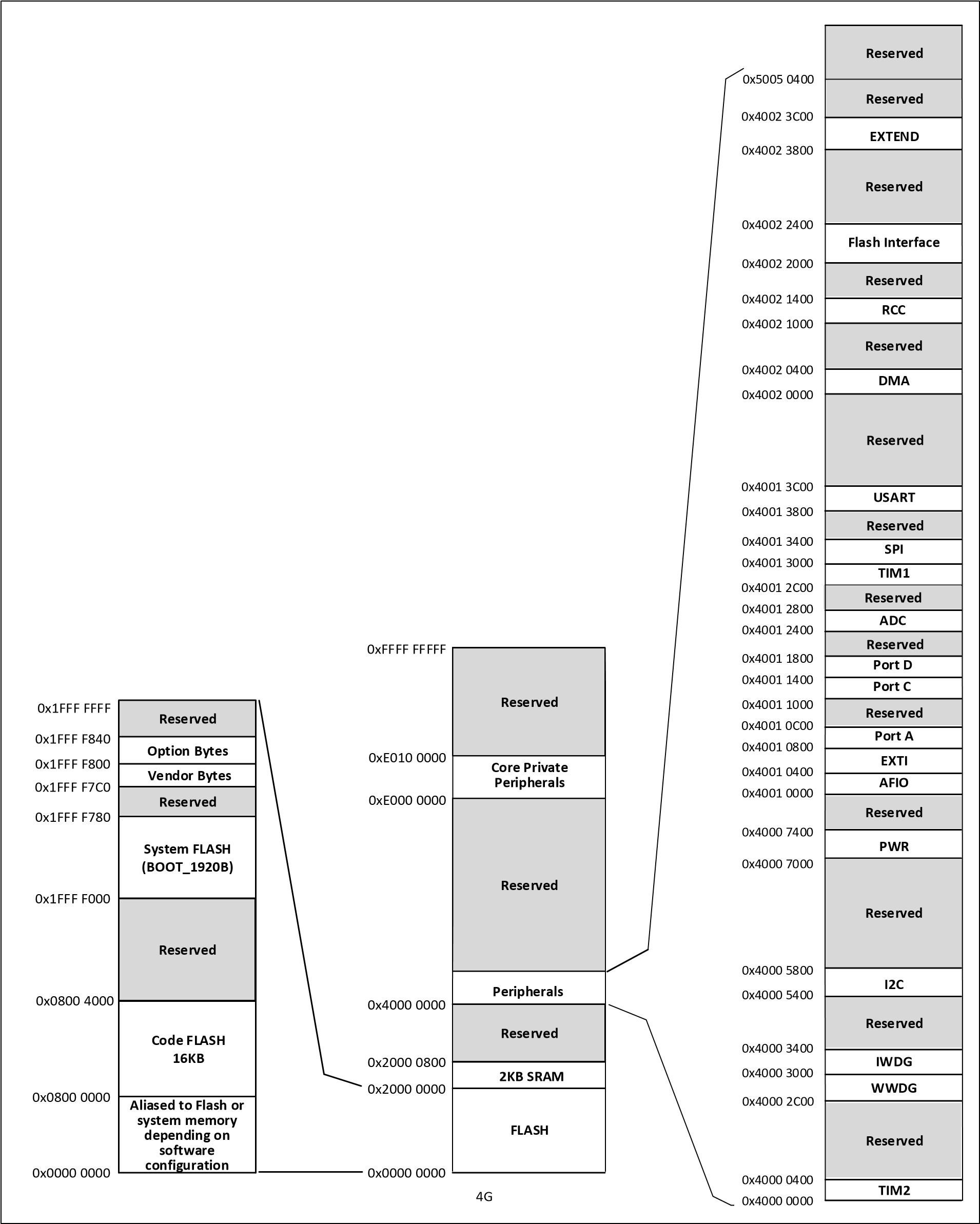
* Шина команд (I-Code) соединяет ядро с интерфейсом команд FLASH, и предварительная выборка выполняется по этой шине.
* Шина данных (D-Code) соединяет ядро с интерфейсом передачи данных FLASH для постоянной загрузки и отладки.
* Системная шина соединяет ядро с матрицей шин и используется для координации доступа к ядру, DMA, SRAM и периферийным устройствам.
* Шина DMA отвечает за DMA главного интерфейса HB, подключенного к матрице шины, доступ к которой осуществляется с помощью флэш-памяти, SRAM и периферийных устройств.
* Матрица шины отвечает за координацию доступа между системной шиной, шиной данных, шиной DMA, SRAM и мостом HB.

## 1.2 Образ памяти

Семейство CH32V003 содержит программную память, память данных, основные регистры, периферийные регистры и многое другое, все они расположены в линейном пространстве объемом 4 ГБ.

Системное хранилище хранит данные в сокращенном формате, т.е. младшие байты хранятся по младшему адресу, а старшие байты хранятся по старшему адресу.

Рисунок 1-2 Образ хранилища



### 1.2.1 Распределение памяти

Встроенная память SRAM емкостью 2 КБАЙТ, начальный адрес 0x20000000, поддерживает доступ к байтам, полусловам (2 байта) и полному слову (4 байта).

Встроенная программная флэш-память объемом 16 КБАЙТ (Code Flash) для хранения пользовательских приложений.

Встроенная системная память 1920B (bootloader) для хранения системного загрузчика (заводской загрузчик).

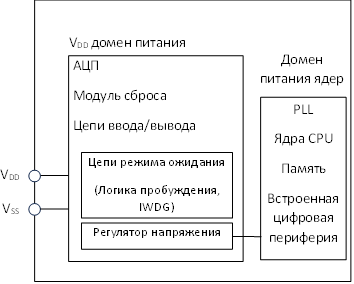
Встроенное пространство объемом 64 Б для хранения слов конфигурации поставщика, отлаженное на заводе-изготовителе и не изменяемое пользователями. Встроенное пространство объемом 64 Б для хранения пользовательских байт.

# Глава 2 Управление питанием (PWR)

## 2.1 Обзор

Рабочее напряжение системы VDD колеблется от 2.7 to 5.5V, а встроенный регулятор напряжения обеспечивает рабочее питание, необходимое ядру.

Рисунок 2-1 Структурная схема структуры источника питания

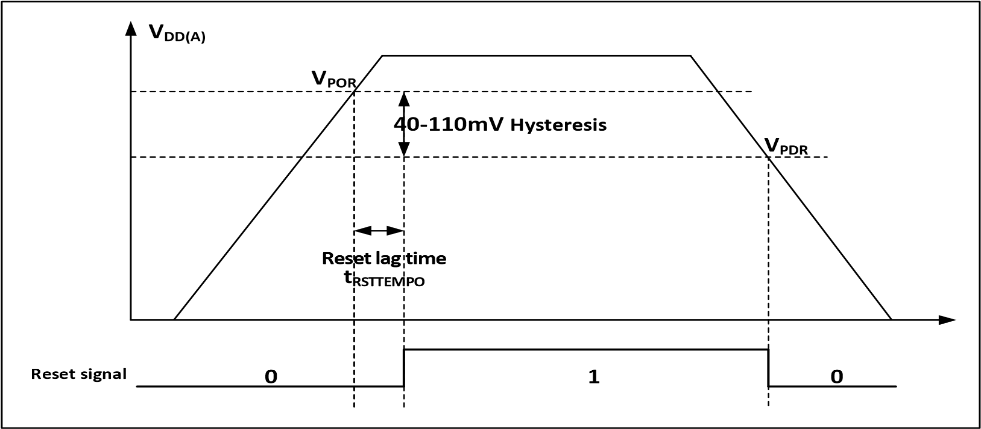


## 2.2 Управление питанием

### 2.2.1 Сброс при включении питания и Сброс при выключении питания

Система имеет внутреннюю схему сброса при включении питания и схему PDR сброса при выключении питания. Когда напряжение питания микросхемы VDD если напряжение падает ниже соответствующего порогового значения, система сбрасывается с помощью соответствующей схемы, и дополнительная внешняя схема сброса не требуется. Параметры порогового напряжения при включении приведены в соответствующем техническом описании VPOR и пороговое напряжение отключения питания VPDR.

Рисунок 2-2 Принципиальная схема работы POR и PDR



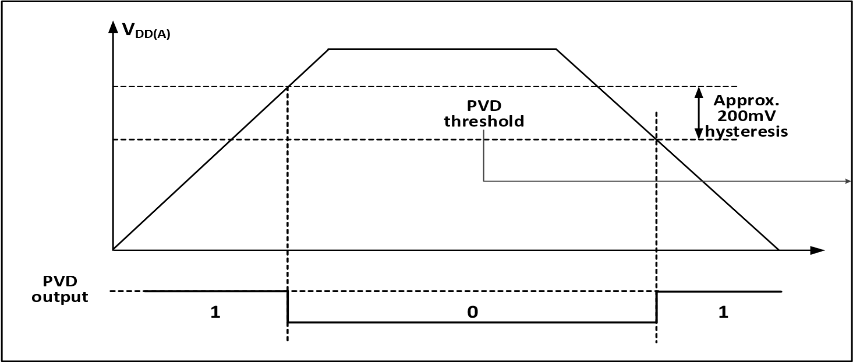
### 2.2.2 Программируемый датчик напряжения

Программируемый монитор напряжения, **PVD**, в основном используется для отслеживания изменения основного источника питания системы и сравнения его с пороговым напряжением, установленным **PLS**[2:0] в регистре управления питанием **PWR\_CTLR**, а при настройке регистра внешних прерываний (**EXTI**) он может генерировать соответствующие прерывания для своевременного уведомления системы о выполнении операций предварительного отключения питания, таких как сохранение данных.

Конкретная конфигурация заключается в следующем.

1. Установите в поле **PLS**[2:0] регистра **PWR\_CTLR** пороговое значение напряжения, которое будет контролироваться.
2. Дополнительная обработка прерываний. функция **PVD** внутренне подключается к настройке запуска нарастающего/спадающего фронта в строке 8 модуля **EXTI**, включает это прерывание (настраивает **EXTI**) и генерирует прерывание **PVD**, когда **VDD** падает ниже порогового значения **PVD** или поднимается выше порогового значения **PVD**.
3. Установите бит **PVDE** в регистре **PWR\_CTLR**, чтобы включить функцию **PVD**.
4. Считайте бит **PVD0** регистра состояния **PWR\_CSR**, чтобы получить текущее значение основного питания системы и соотношение пороговых значений настройки **PLS**[2:0], и выполните соответствующую мягкую обработку. Когда напряжение VDD превышает пороговое значение, установленное **PLS**[2:0], значение **PVD0** равно 0; когда напряжение VDD ниже порогового значения, установленного **PLS**[2:0], значение **PVD0** равно 1.

Figure 2-3 Schematic diagram of PVD operation



## 2.3 Режимы с низким энергопотреблением

После перезагрузки системы микроконтроллер переходит в нормальное рабочее состояние (режим запуска), в котором можно сэкономить энергопотребление системы, снизив основную частоту системы, отключив неиспользуемые периферийные часы или уменьшив рабочие периферийные часы. Если система не должна работать, вы можете перевести ее в режим пониженного энергопотребления и позволить системе выйти из этого состояния с помощью определенных событий.

В настоящее время микроконтроллеры предлагают 2 режима работы с низким энергопотреблением, разделенных с точки зрения различий в работе процессоров, периферийных устройств, регуляторов напряжения и т.д.

* Спящий режим: Ядро перестает работать, а все периферийные устройства (включая основные частные периферийные устройства) продолжают работать.
* Режим ожидания: Остановите все часы, проснитесь и переведите часы в режим HSI.

Таблица 2-1 Список режимов низкого энергопотребления

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Режим | Установки | Источник события пробуждения | Воздействие на тактирование | Регул. напряжения |
| Sleep | **WFI** | Любое прерывание | Часы ядра выключены, не влияет на другие системы тактирования | Вкл |
| **WFE** | Событие Wake-Up |
| Standby | Установить **SLEEPDEEP** в 1 Установить **PDDS** в 1 **WFI** or **WFE** | AWU событие *Note: Любое событие может вывести систему из сна, но после этого вызова Reset нет.* | HSE, HIS, PLL и тактирование переферии выключено | Откл |

*Примечание: Бит* ***SLEEPDEEP*** *относится к приватным битам управления периферийными устройствами ядра, ссылка на регистр* ***PFIC\_SCTLR*** *контроллера CH32V003.*

### 2.3.1 Варианты конфигурации с низким энергопотреблением

 WFI и WFE

WFI: Микроконтроллер включается источником прерывания с ответом контроллера прерывания, и функция обслуживания прерывания будет выполнена первой после пробуждения системы (за исключением сброса микроконтроллера).

WFE: Событие пробуждения запускает микроконтроллер для выхода из режима пониженного энергопотребления. К событиям пробуждения относятся:

1. Сконфигурируйте внешнюю или внутреннюю линию EXTI в режиме события, когда нет необходимости настраивать контроллер прерываний.
2. Или настройте источник прерываний, эквивалентный пробуждению WFI, при котором система определяет приоритет выполнения функции обслуживания прерываний.
3. Или настройте бит **SLEEPONPEN** так, чтобы он включал включение периферийных прерываний, но не включал прерывания в контроллере прерываний, и бит ожидания прерывания должен быть сброшен после пробуждения системы.

* **SLEEPONEXIT**

*Включен*: После выполнения команды WFI или WFE микроконтроллер гарантирует, что все ожидающие прерывания службы завершены, а затем переходит в режим пониженного энергопотребления.

*Отключен*: Микроконтроллер переходит в режим пониженного энергопотребления сразу после выполнения команды WFI или WFE.

* **SEVONPEND**

*Включен*: Все прерывания или события пробуждения могут привести к снижению энергопотребления, введенного при выполнении WFE.

*Отключен*: Только прерывания или события пробуждения, включенные в контроллере прерываний, могут активировать режим низкого энергопотребления, введенный при выполнении WFE.

### 2.3.2 Режим сна (SLEEP)

В этом режиме все контакты ввода-вывода находятся в рабочем состоянии, а все периферийные устройства работают нормально, поэтому попробуйте отключить бесполезные периферийные устройства перед переходом в спящий режим, чтобы снизить энергопотребление. В этом режиме пробуждение занимает меньше всего времени..

*Вход в режим*: Сконфигурируйте бит управления базовым регистром **SLEEPDEEP**=0, регистр управления питанием **PDDS**=0, выполните WFI или WFE, опционально **SEVONPEND** и **SLEEPONEXIT**.

*Выход из режима*: Произвольное прерывание или событие пробуждения.

### 2.3.3 Режим ожидания (STANDBY)

Режим ожидания представляет собой комбинацию периферийных механизмов управления тактовой частотой, основанных на режиме глубокого сна ядра (SLEEP DEEP), и позволяет регулятору напряжения работать с гораздо меньшим энергопотреблением. В этом режиме домен высокочастотных тактовых импульсов (HSE/HSI/PL) отключен, содержимое SRAM и регистра сохранено, а состояние pin-кода ввода-вывода сохранено. Система может продолжать работать после выхода из этого режима, и HSI называется системными часами по умолчанию.

Если выполняется программирование флэш-памяти, система не перейдет в режим ожидания до тех пор, пока не будет завершен доступ к памяти.

В режиме ожидания могут работать модули: независимый сторожевой таймер (IWDG), низкочастотные тактовые сигналы (LSI).

*Вход в режим*: Настройте управляющий бит основного регистра **SLEEPDEEP**=1, **ODDS**=1 в регистре управления питанием и выполните WFI или WFE, необязательно **SEVONPEND** и **SLEEPONEXIT**.

*Выход из режима*:

1. Любое прерывание/событие (заданное во внешнем регистре прерываний).
2. Событие AWU, часы переключаются на тактирование от HSI после этого пробуждения, система не сбрасывается.

### 2.3.4 Auto-пробуждение (AWU)

Может быть реализован автоматический режим пробуждения без внешних прерываний. Можно запрограммировать базу времени на периодический выход из режима ожидания.

Дополнительный внутренний низкочастотный генератор тактовых импульсов LSI частотой 128 кГц используется в качестве основы для автоматического подсчета времени пробуждения.

При включении функции прерывания AWU вам необходимо установить триггер нарастания/спада фронтов 9-й линии, внутренне подключенной к модулю EXTI, и включить это прерывание (конфигурация EXTI).

## 2.4 Описание регистров

Таблица 2-2 Список регистров, связанных с PWR

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Адрес** | **Описание** | **Значение сброса** |
| R32\_PWR\_CTLR | 0x40007000 | Регистр управления питанием | 0x00000000 |
| R32\_PWR\_CSR | 0x40007004 | Регулятор питания/регистр состояния | 0x00000000 |
| R32\_PWR\_AWUCSR | 0x40007008 | Управление автоматическим пробуждением /регистрация состояния | 0x00000000 |
| R32\_PWR\_AWUWR | 0x4000700С | Регистр значений для сравнения в окне автоматического пробуждения | 0x0000003F |
| R32\_PWR\_AWUPSC | 0x40007010 | Регистр коэффициента пересечения с автоматическим включением | 0x00000000 |

### 2.4.1 Регистр управления питанием (PWR\_CTLR)

Смещение адреса: 0x00

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Бит | Название | Доступ | Описание | | Знач. сброса |
| [31:8] | Res | RO | Резерв | | 0 |
| [7:5] | PLS[2:0] | RW | Настройка порога контроля напряжения PVD. Подробные инструкции приведены в разделе "Электрические характеристики" спецификации | | 0 |
| 000: | 2.85V восходящий фронт/2.7V нисходящий фронт |
| 001: | 3.05V восходящий фронт/2.9V нисходящий фронт |
| 010: | 3.3V восходящий фронт/3.15V нисходящий фронт |
| 011: | 3.5V восходящий фронт/3.3V нисходящий фронт |
| 100: | 3.7V восходящий фронт/3.5V нисходящий фронт |
| 101: | 3.9V восходящий фронт/3.7V нисходящий фронт |
| 110: | 4.1V восходящий фронт/3.9V нисходящий фронт |
| 111: | 4.4V восходящий фронт/4.2V нисходящий фронт |
| [4] | PVDE | RW | Бит флага включения функции контроля напряжения | | 0 |
| 1: | Включить функцию контроля напряжения |  |
| 0: | Отключить функцию контроля напряжения |  |
| [3:2] | Res | RO | Резерв | | 0 |
| [1] | PDDS | RW | Бит флага включения функции контроля напряжения | | 0 |
| 1: | Вход в режим ожидания |  |
| 0: | Вход в режим сна |  |
| [0] | Res | RO | Резерв | | 0 |

### 2.4.2 Power Control/Status Register (PWR\_CSR)

Смещение адреса: 0x04

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Резерв | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Резерв | | | | | | | | | | | | | PVD0 | Резерв | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Бит | Название | Доступ | Описание | | Значение сброса |
| [31:3] | Res | RO | Резерв | | 0 |
| [2] | PVD0 | RO | Бит флага состояния вывода PVD. Этот бит действителен, когда значение PVDE=1 в регистре PWR\_CTLR | | 0 |
| 1: | Значения VDD и VDDA ниже порогового значения PVD, установленного PLS[2:0] |
| 0: | Значения VDD и VDDA превышают пороговое значение PVD, установленное PLS[2:0] |
| [1:0] | Res | RO | Резерв | | 0 |

### 2.4.3 Auto-wakeup Control/Status Register (PWR\_AWUCSR)

Смещение адреса: 0x08

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Резерв | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Резерв | | | | | | | | | | | | | | AWUEN | Резерв |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Бит | Название | Доступ | Описание | | Значение сброса |
| [31:2] | Res | RO | Резерв | | 0 |
| [1] | AWUEN | RW | Включить автоматическое пробуждение | | 0 |
| 1: | Включение авто пробуждения |
| 0: | недостоверный |
| [0] | Res | RO | Резерв | | 0 |

### 2.4.4 Auto-wakeup Window Comparison Value Register (PWR\_AWUWR)

Смещение адреса: 0x0C

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Бит | Название | Доступ | Описание | Значение сброса |
| [31:6] | Res | RO | Резерв | 0 |
| [5:0] | AWUEN | RW | Значение окна AWU: Значение AWU окна равно входному значению AWU + 1; Значение окна AWARF используется для сравнения со значением счетчика вверх. Когда значение счетчика равно значению окна, генерируется сигнал пробуждения | 0 |

### 2.4.5 Auto-wakeup Crossover Factor Register (PWR\_ AWUPSC)

Смещение адреса: 0x10

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Бит | Название | Доступ | Описание | | Значение сброса |
| [31:4] | Res | RO | Резерв | | 0 |
| [3:0] | AWUPSC [3:0] | RW | База счета времени | | 0 |
| 0000: | Прескаллер выключен |
| 0001: | Прескаллер выключен |
| 0010: | Делитель 2 |
| 0011: | Делитель 4 |
| 0100: | Делитель 8 |
| 0101: | Делитель 16 |
| 0110: | Делитель 32 |
| 0111: | Делитель 64 |
| 1000: | Делитель 128 |
| 1001: | Делитель 256 |
| 1010: | Делитель 512 |
| 1011: | Делитель 1024 |
| 1100: | Делитель 2048 |
| 1101: | Делитель 4096 |
| 1110: | Делитель 10240 |
| 1111: | Делитель 61440 |  |

# Глава 3 Сброс и управление тактовой частотой (RCC)

Контроллер предоставляет различные формы сбросов и настраиваемые древовидные структуры синхронизации, основанные на разделении областей электропитания и особенностях управления питанием периферийных устройств в приложениях. В этом разделе описывается область применения каждой системы тактирования.

## 3.1 Основные характеристики

* Несколько форм сброса
* Несколько источников тактовой частоты, управление тактовой частотой шины
* Встроенная система мониторинга и защиты внешнего источника частоты
* Независимое управление всеми периферийными часами: сброс, включение, выключение
* Поддержка вывода внутреннего тактирования

## 3.2 Сброс

Контроллер обеспечивает 2 вида сброса: сброс питания и сброс системы.

### 3.2.1 Сброс питания

Когда происходит сброс питания, все регистры сбрасываются.

Сброс питания генерируется при возникновении следующего события:

 Сброс настроек включения/выключения питания (**POR**/**PDR**)

### 3.2.2 Сброс системы

Когда происходит сброс системы, она сбрасывает флаг сброса в дополнение к регистру управления/состояния **RCC\_RSTSCKR** и всем регистрам. Источник события сброса определяется путем просмотра бита флага сброса состояния в регистре **RCC\_RSTSCKR**.

Сброс системы происходит при возникновении одного из следующих событий:

* Низкий уровень сигнала на выводе **NRST** (внешний сброс)
* Окончен счет оконного сторожевого таймера (**WWDG** сброс) Окончен счет независимого сторожевого таймера (**IWDG** сброс)
* Программный сброс (**SW** сброс)
* Сброс от контроллера пониженного напряжения питания

Сброс оконного/независимого сторожевого таймера: генерируется триггером переполнения цикла подсчета периферийного таймера оконного/независимого сторожевого таймера, подробное описание которого приведено в соответствующем разделе.

Сброс программного обеспечения: Устройство CH32V003 выполняет сброс системы с помощью позиции 1 регистра **RSTSYS** конфигурации прерываний PFIC\_**CFGR** в программируемом контроллере прерываний **PFIC** или позиции **SYSRST** 1 регистра конфигурации **PFIC\_SCTLR** для сброса системного блока, подробности приведены в соответствующей главе.Low Power Management Сброс: Сброс режима ожидания будет активирован путем установки значения **STANDBY\_RST** в положение 1 в выбранном пользователем байте. При этом будет выполнен сброс системы вместо перехода в режим ожидания после завершения процесса перехода в режим ожидания.

Figure 3-1 System reset structure

System

Reset

Power Reset

Software Reset

WWDG Reset

IWDG Reset

Low-power Management Reset

R

PU

V

DD

/V

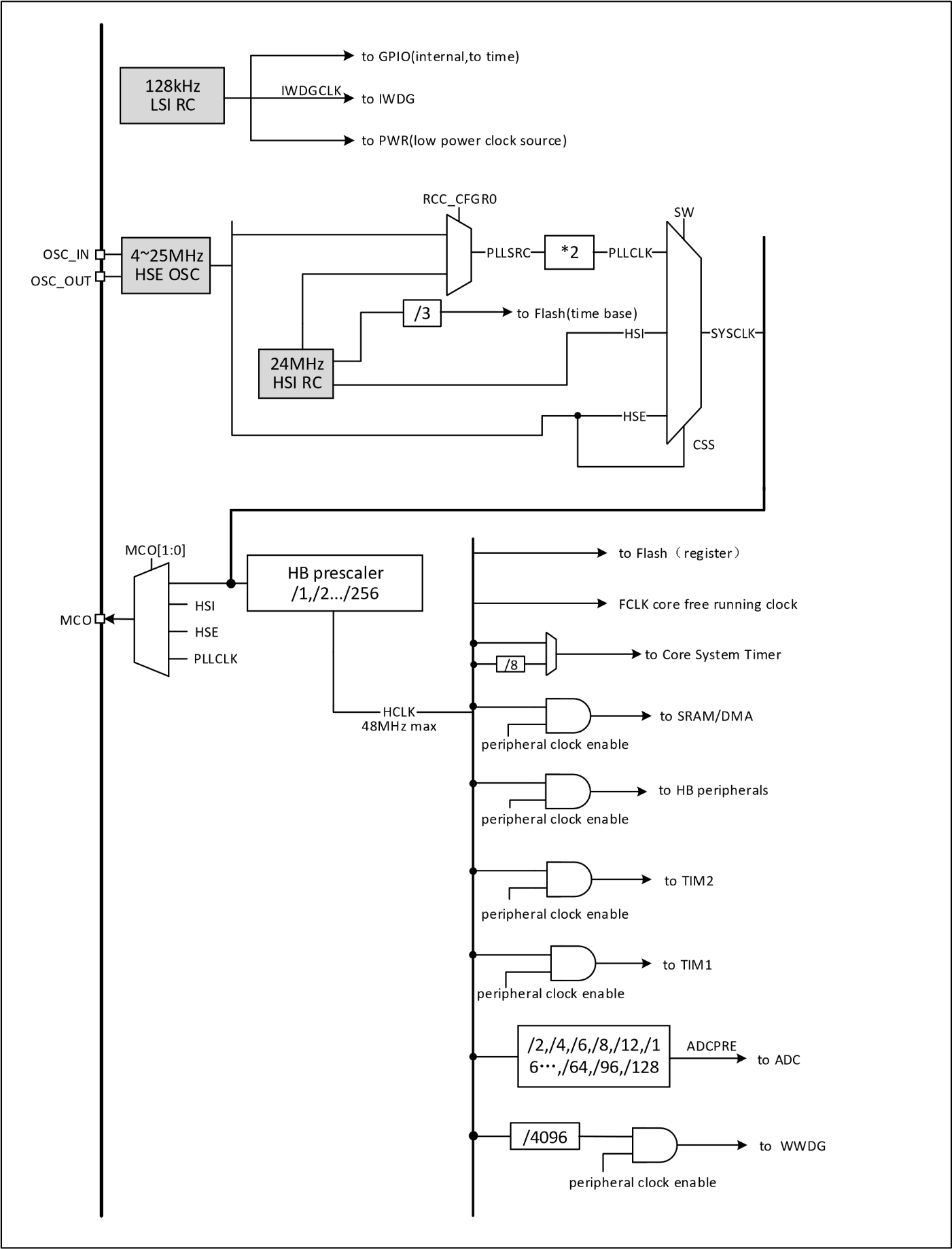
DDA

NRST

## 3.3 Тактирование

### 3.3.1 Структура системы тактирования

Таблица 3-2 CH32V003 диаграмма дерева тактирования



### 3.3.2 High-speed Clock (HSI/HSE)

HSI - это высокоскоростной тактовый сигнал, генерируемый внутренним RC-генератором системы с частотой 24 МГц. RC-генератор HSI может обеспечивать синхронизацию системы без каких-либо внешних устройств. Он имеет короткое время запуска. ЭТО включается и отключается установкой бита HSI в регистре **RCC\_CTLR**, а бит **DIRTY** указывает, является ли генератор RC HSI стабильным или нет. По умолчанию в системе для значений **HSION** и **HSIRDY** установлено значение 1 (рекомендуется не отключать их). Если установлен бит **HSIRDYIE** в регистре **RCC\_INTR**, будет сгенерировано соответствующее прерывание.

* Заводская калибровка: Различия в производственном процессе приводят к разной частоте RC-колебаний для каждой микросхемы, поэтому калибровка HSI выполняется для каждой микросхемы перед отправкой. После сброса системы заводское значение калибровки загружается в HSICAL[7:0] регистра RCC\_CTLR.
* Пользовательская настройка: В зависимости от различных напряжений или температур окружающей среды приложение может регулировать частоту HSI, используя биты **HSITRIM**[4:0] в регистре **RCC\_CTLR**.

*Примечание: Если кварцевый генератор HSE выходит из строя, часы HSI используются в качестве резервного источника тактового сигнала (система защиты часов).*

HSE - это внешний высокоскоростной тактовый сигнал, включающий генерацию внешнего кристаллического/керамического резонатора или внешнюю высокоскоростную подачу тактового сигнала.

* Внешний кристаллический/керамический резонатор (HSE Crystal): Внешний генератор с частотой 4-25 МГц обеспечивает более точный источник тактовых импульсов для системы. Дополнительную информацию можно найти в разделе "Электрические характеристики" спецификации. Кристалл HSE можно включать и выключать, установив бит **HSEON** в регистре **RCC\_CTLR**. Бит **HSERDY** указывает, стабильны ли колебания кристалла HSE или нет, и аппаратное обеспечение подает тактовый сигнал в систему только после установки **HSERDY** в положение 1. Если установлен бит **HSERDYIE** регистра **RCC\_INTR**, то будет сгенерировано соответствующее прерывание.

Рисунок 3-3 Схема подключения высокоскоростного внешнего резонатора

OSC\_IN

OSC\_OUT

C

L1

C

L2

4

～

25

MHz

Load

Capacitance

*Примечание: Нагрузочный конденсатор должен располагаться как можно ближе к контакту генератора, а значение емкости должно быть выбрано в соответствии с параметрами кристалла.*

* Внешний высокоскоростной источник тактовой частоты (HSE Bypass): В этом режиме источник тактовой частоты подается непосредственно с внешнего устройства на вывод **OSC\_IN**, при этом вывод **OSC\_OUT** не подключен. Максимальная поддерживаемая частота составляет 25 МГц. Приложению необходимо установить бит **HSEBYP** для включения функции обхода HSE с битом **HSEON** равным 0, а затем снова установить бит **HSEON**.

Рисунок 3-3 Схема подключения высокоскоростного внешнего генератора

OSC\_IN

**f**

**HSE\_ext**

External clock

source

OSC\_OUT

(

Suspended

)

### 3.3.3 Низкоскоростные тактовые импульсы (LSI)

LSI - это низкочастотный тактовый сигнал, генерируемый внутренним RC-генератором системы с частотой около 128 кГц. Он может поддерживаться в рабочем состоянии в режиме выключения и ожидания и обеспечивает синхронизацию для часов RTC, независимого сторожевого таймера и блока пробуждения. Более подробную информацию можно найти в разделе "Электрические характеристики" спецификации. LSI можно включать и отключать, устанавливая бит **LSION** в регистре **RCC\_RSTSCKR**, а затем определяя, стабильны ли RC-колебания LSI, запрашивая бит **LSIRDY**, и аппаратное обеспечение подает тактовый сигнал только после позиции **LSIRDY** 1. Если установлен бит **LSIRDYIE** в регистре **RCC\_INTR**, то будет сгенерировано соответствующее прерывание.

### 3.3.4 PLL тактирование

Настроив регистр **RCC\_CFGR0** и расширенный регистр **EXTEND\_CTR**, внутренние часы PLL могут выбирать 2 источника синхронизации, эти настройки необходимо выполнить до включения PLL, после запуска PLL эти параметры изменить нельзя. Установите бит **PLLON** в регистре **RCC\_CTLR** включенным и выключенным, бит **PLLRDY** - для указания стабильности синхронизации PLL, а аппаратное обеспечение - для подачи синхронизации в систему только после позиции 1 PLL. Если установлен бит **PLLRDYIE** регистра **RCC\_INTR**, то будет сгенерировано соответствующее прерывание.

PLL источник тактирования:

* HSI тактирование
* HSE тактирование

### 3.3.5 Тактирование шины/периферии

#### 3.3.5.1 Системное тактирование (SYSCLK)

Настройте источник системных тактовых импульсов, настроив регистр **RCC\_CFGR0** биты **SW**[1:0], **СЕК**[1:0] указывает текущий источник системных тактовых импульсов.

* HSI как источник тактов для системы
* HSE как источник тактов для системы
* PLL как источник тактов для системы

После перезагрузки контроллера в качестве системного источника тактовых импульсов выбираются часы HSI по умолчанию. Переключение между источниками тактовых импульсов должно происходить только тогда, когда целевой источник тактовых импульсов готов.

#### 3.3.5.2 HB тактирование шины периферии (HCLK)

Тактовые частоты шины HB можно настроить, настроив биты **PRE**[3:0] регистра **RCC\_CFGR0**. Тактовые частоты шины определяют опорную частоту доступа к периферийному интерфейсу, которая устанавливается под ними. Приложения могут настраивать различные значения, чтобы снизить энергопотребление при работе некоторых периферийных устройств. Различные разряды в регистрах **RCC\_APB1RSTR** и **RCC\_APB2PRSTR** могут возвращать различные периферийные модули в исходное состояние.

Каждый бит в регистрах **RCC\_AHBLPENR**, **RCC\_APB1PCENR** и **RCC\_APB2PCENR** может использоваться для индивидуального включения или выключения интерфейса синхронизации связи для различных периферийных модулей. При использовании периферийного устройства вам сначала необходимо включить его бит включения тактовой частоты, чтобы получить доступ к его регистрам.

#### 3.3.5.3 Тактирование независимого сторожевого таймера

Если независимый сторожевой таймер был установлен с помощью аппаратной конфигурации или запущен с помощью программного обеспечения, генератор LSI будет включен принудительно и не может быть выключен. После стабилизации генератора LSI тактовый сигнал подается на IWDG.

#### 3.3.5.4 Выход тактирования микроконтроллера (MCO)

Микроконтроллер позволяет выводить тактовые сигналы на контакты MCO. Следующие 4 тактовых сигнала могут быть выбраны в качестве выходных сигналов синхронизации MCO путем настройки режима мультиплексированного двухтактного вывода в соответствующих регистрах порта GPIO путем настройки битов **MCO**[2:0] регистра **RCC\_CFGR0**.

* Системный (SYSCLK) вывод тактов
* HSI вывод тактов
* HSE вывод тактов
* PLL вывод тактов

### 3.3.6 Защита системы тактирования

Система clock safety system - это механизм оперативной защиты контроллера, который переключается на внутреннее тактирование HSI в случае сбоя внешнего тактирования HSE и генерирует прерывании, позволяющее прикладному программному обеспечению выполнить обработку этого события.

Система clock security system активируется установкой CSSON в положение 1 регистра RCC\_CTLR. После этого мониторинг часов может быть включен после задержки запуска генератора HSE (HSERDY=1) и будет выключен после отключения часов HSE. Как только часы HSE выйдут из строя во время работы системы, генератор HSE будет выключен, событие сбоя часов будет отправлено на TIM1\_BKIN вход таймера расширенного управления (TIM1), и будет сгенерировано защитное прерывание часов с позицией CSSF 1, и приложение перейдет в немаскируемое прерывание NMI. Установив бит CSSC, можно снять флаг бита CSSF и отменить бит ожидания прерывания NMI.

Если в качестве системного тактирования используется текущее значение HSE или если текущее значение HSE используется в качестве входных тактов для PLL, а PLL используется в качестве системного тактирования, система безопасности тактирования автоматически переключит системную на генератор HSI и отключит генератор HSE и PLL в случае сбоя HSE.

## 3.4 Описание регистров

Таблица 3-1 Список регистров, связанных с RCC

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Адрес** | **Описание** | **Значение сброса** |
| R32\_RCC\_CTLR | 0x40021000 | Регистр управления тактированием | 0x0000xx83 |
| R32\_RCC\_CFGR0 | 0x40021004 | Регистр конфигурации тактирования 0 | 0x00000020 |
| R32\_RCC\_INTR | 0x40021008 | Регистр прерываний тактирования | 0x00000000 |
| R32\_RCC\_APB2PRSTR | 0x4002100C | Регистр сброса периферийной шины PB2 | 0x00000000 |
| R32\_RCC\_APB1PRSTR | 0x40021010 | Регистр сброса периферийной шины PB1 | 0x00000000 |
| R32\_RCC\_AHBPCENR | 0x40021014 | Регистр включения тактирования периферийной шины HB | 0x00000004 |
| R32\_RCC\_APB2PCENR | 0x40021018 | Регистр включения тактирования периферийной шины PB2 | 0x00000000 |
| R32\_RCC\_APB1PCENR | 0x4002101C | Регистр включения тактирования периферийной шины PB1 | 0x00000000 |
| R32\_RCC\_RSTSCKR | 0x40021024 | Регистр статуса/управления | 0x0C000000 |

### 3.4.1 Регистр управления тактированием (RCC\_CTLR)

Смещение адреса: 0x00

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| Резерв | | | | | | PLLRDY | PLLON | Резерв | | | | CSSON | HSEBYP | HSERDY | HSEON | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| HSICAL[7:0] | | | | | | | | HSITRIM[4:0] | | | | | Рез. | HSIRDY | HSION |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Бит | Название | Доступ | Описание | | Знач. сброса |
| [31:26] | Res | RO | Резерв | | 0 |
| [25] | PLLRDY | RO | Бит готовности к блокировке тактовой частоты PLL | | 0 |
| 1: | Частота PLL заблокирована |
| 0: | Частота PLL не заблокирована |
| [24] | PLLON | RW | Управляющий бит включения такттирования PLL | | 0 |
| 1: | Включить такрирование PLL |
| 0: | Отключить такрирование PLL |
| [23:20] | Res | RO | Резерв | | 0 |
| [19] | CSSON | RW | Бит включения зищиты системы тактирования | | 0 |
| 1: | Включение защиты тактирования. Когда HSE готов (флаг HSERDY=1), аппаратное обеспечение активирует функцию мониторинга тактовой частоты HSE и устанавливает флаг CSSF и прерывание NMI при обнаружении аномалии в работе HSE; когда HSE не готов, аппаратное обеспечение отключает функцию мониторинга тактового сигнала HSE |
| 0: | Отключение защиты тактирования |
| [18] | HSEBYP | RW | Бит управления обходом внешнего высокоскоростного тактирования | | 0 |
| 1: | Байпас внешнего кварцевого генератора/резонатора |
| 0: | Байпас отключен |
| Примечание: этот бит будет записан при HSEON=0 | |
| [17] | HSERDY | RO | Бит готовности внешнего высокоскоростного тактового генератора (устанавливается аппаратно) | | 0 |
| 1: | Стабильное внешнее тактирование |
| 0: | Внешнее тактирование не стабильно |
| Примечание: этот бит будет записан при HSEON=0 через 6 тактов | |
| [16] | HSEON | RW | Бит управления включением внешнего высокоскоростного кристаллического генератора | | 0 |
| 1: | Включить такрирование HSE |
| 0: | Отключить такрирование HSE |
| Примечание: Этот бит автоматически сбрасывается в 0 аппаратным обеспечением после входа в режим Standby | |
| [15:8] | HSICAL | RO | Значение калибровки внутреннего высокоскоростного тактового генератора, которое автоматически инициализируется при запуске системы | | xxh |
| [7:3] | HSITRIM | RW | Внутреннее значение регулировки высокоскоростного тактового генератора. Пользователь может ввести корректирующее значение, которое будет наложено на значение HSICAL[7:0], чтобы настроить частоту внутреннего RC-генератора HSI с учетом изменений напряжения и температуры. По умолчанию установлено значение 16, что позволяет настроить HSI на 24 МГц ±1%. Изменение значения HSICAL регулируется примерно на 60 кГц за шаг | | 10000 |
| [2] | Res | RO | Резерв | | 0 |
| [1] | HSIRDY | RO | Бит готовности внутреннего тактового генератора (24MHz) (устанавливается аппаратно) | | 0 |
| 1: | Стабильное внутреннее тактирование |
| 0: | Внутреннее тактирование не стабильно |
| Примечание: этот бит будет записан при HSEON=0 через 6 тактов | |
| [0] | HSION | RW | Бит управления включением внутреннего тактового генератора | | 0 |
| 1: | Включить такрирование HSI |
| 0: | Отключить такрирование HIS |
| Примечание: Этот бит автоматически сбрасывается в 0 аппаратным обеспечением после входа в режим Standby | |

### 3.4.2 Регистр конфигурации тактирования 0 (RCC\_CFGR0)

Смещение адреса: 0x04

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Резерв | | | | | MCO[2:0] | | | Резерв | | | | | | | PLLSRC |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADCPRE[4:0] | | | | | Резерв | | | HPRE[3:0] | | | | SWS[1:0] | | SW[1:0] | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Бит | Название | Доступ | Описание | | Знач. сброса |
| [31:27] | Res | RO | Резерв | | 0 |
| [26:24] | MCO[2:0] | RW | Управление тактированием выхода микроконтрроллера на пин MCO | | 0 |
| 0xx: | Нет тактирования на выход |
| 100: | Выход тактируется от системных часов (SYSCLK) |
| 101: | Выход тактируется от HSI (24MHz) |
| 110: | Выход тактируется от внешнего резонатора HSE |
| 111: | Выход тактируется от PLL |
| [23:17] | Res | RO | Резерв | | 0 |
| [16] | PLLSRC | RW | Источник входного тактового сигнала для PLL (записывается только при выключенном PLL) | | 0 |
| 1: | HSE подается на PLL без деления частоты |
| 0: | HSI подается на PLL без деления частоты |
| [15:11] | ADCPRE [4:0] | RW | Управление предделителем источника тактовой частоты АЦП {13:11,15:14} | | 0 |
| 000xx: | HBCLK деленное на 2 как тактирование для АЦП |
| 010xx: | HBCLK деленное на 4 как тактирование для АЦП |
| 100xx: | HBCLK деленное на 6 как тактирование для АЦП |
| 110xx: | HBCLK деленное на 8 как тактирование для АЦП |
| 00100: | HBCLK деленное на 4 как тактирование для АЦП |
| 00101: | HBCLK деленное на 8 как тактирование для АЦП |
| 00110: | HBCLK деленное на 16 как тактирование для АЦП |
| 00111: | HBCLK деленное на 32 как тактирование для АЦП |
| 01100: | HBCLK деленное на 8 как тактирование для АЦП |
| 01101: | HBCLK деленное на 16 как тактирование для АЦП |
| 01110: | HBCLK деленное на 32 как тактирование для АЦП |
| 01111: | HBCLK деленное на 64 как тактирование для АЦП |
| 10100: | HBCLK деленное на 12 как тактирование для АЦП |
| 10101: | HBCLK деленное на 24 как тактирование для АЦП |
| 10110: | HBCLK деленное на 48 как тактирование для АЦП |
| 10111: | HBCLK деленное на 96 как тактирование для АЦП |
| 11100: | HBCLK деленное на 16 как тактирование для АЦП |
| 11101: | HBCLK деленное на 32 как тактирование для АЦП |
| 11110: | HBCLK деленное на 64 как тактирование для АЦП |
| 11111: | HBCLK деленное на 128 как тактирование для АЦП |
| Примечание: Тактовая частота АЦП не должна превышать максимальное значение 24 МГц | |
| [10:8] | Res | RO | Резерв | | 0 |
| [7:4] | HPRE[3:0] | RW | Управление предделителем источника тактовой частоты HB | | 0010 |
| 0000: | Предделитель отключен |
| 0001: | SYSCLK / 2 |
| 0010: | SYSCLK / 3 |
| 0011: | SYSCLK / 4 |
| 0100: | SYSCLK / 5 |
| 0101: | SYSCLK / 6 |
| 0110: | SYSCLK / 7 |
| 0111: | SYSCLK / 8 |
| 1000: | SYSCLK / 2 |
| 1001: | SYSCLK / 4 |
| 1010: | SYSCLK / 8 |
| 1011: | SYSCLK / 16 |
| 1100: | SYSCLK / 32 |
| 1101: | SYSCLK / 64 |
| 1110: | SYSCLK / 128 |
| 1111: | SYSCLK / 256 |
| Примечание: Если коэффициент предделителя источника тактовой частоты HB больше 1, необходимо включить буфер предварительной выборки. | |
| [3:2] | SWS[1:0] | RO | Статус системных часов (SYSCLK) (устанавливается аппаратно) | | 0 |
| 00: | Тактирование SYSCLK по тактированию из HSI |
| 01: | Тактирование SYSCLK по тактированию из HSE |
| 10: | Тактирование SYSCLK по тактированию из PLL |
| 11: | Недоступно |
| [1:0] | SW[1:0] | RW | Выбор источника тактирования для системных часов (SYSCLK) | | 0 |
| 00: | Тактирование SYSCLK по тактированию из HSI |
| 01: | Тактирование SYSCLK по тактированию из HSE |
| 10: | Тактирование SYSCLK по тактированию из PLL |
| 11: | Недоступно |
| Примечание: При включении функции защиты тактового сигнала (CSSON=1) в случае возврата из режимов Standby и Stop или при отказе внешнего генератора HSE, используемого в качестве системного тактового сигнала, внутренний генератор HSI принудительно выбирается аппаратно в качестве системного тактового сигнала | |

### 3.4.3 Регистр прерываний тактирования (RCC\_INTR)

Смещение адреса: 0x08

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | | 27 | | 26 | | 25 | | 24 | | 23 | | | 22 | 21 | | | 20 | | 19 | | 18 | | 17 | | | 16 |
| Резерв | | | | | | | | | | | | CSSC | | | Резерв | | | | PLLRDYC | | HSERDYC | | HSIRDYC | | Рез. | | LSIRDYC | | |
| 15 | 14 | 13 | 12 | | 11 | | 10 | | 9 | | 8 | | 7 | | | 6 | 5 | | | 4 | | 3 | | 2 | | 1 | | | 0 |
| Резерв | | PLLEDYIE | | HSERDYIE | | HSIRDYIE | | Рез. | | LSIRDYIE | | CSSF | | Резерв | | | | PLLRDYF | | | HSERDYF | | HSIRDYF | | Рез. | | | LSIRDYF | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Бит | Название | Доступ | Описание | | Знач. сброса |
| [31:24] | Res | RO | Резерв | | 0 |
| [23] | CSSC | WO | Сбросить флаг прерывания системы безопасности тактового генератора (CSSF) | | 0 |
| 1: | Сбросить флаг прерывания CSSF |
| 0: | Не сбрасывать |
| [22:21] | Res | RO | Резерв | | 0 |
| [20] | PLLRDYC | WO | Сбросить флаг готовности прерывания PLL | | 0 |
| 1: | Отчистить флаг прерывания PLLRDYF |
| 0: | Не сбрасывать |
| [19] | HSERDYC | WO | Сбросить флаг готовности прерывания HSE | | 0 |
| 1: | Отчистить флаг прерывания HSERDYF |
| 0: | Не сбрасывать |
| [18] | HSIRDYC | WO | Сбросить флаг готовности прерывания HSI | | 0 |
| 1: | Отчистить флаг прерывания HSIRDYF |
| 0: | Не сбрасывать |
| [17] | Res | RO | Резерв | | 0 |
| [16] | LSIRDYC | WO | Сбросить флаг готовности прерывания LSI | | 0 |
| 1: | Отчистить флаг прерывания LSIRDYF |
| 0: | Не сбрасывать |
| [15:13] | Res | RO | Резерв | | 0 |
| [12] | PLLRDYIE | RW | Бит разрешения прерывания по готовности PLL | | 0 |
| 1: | Включить прерывание |
| 0: | Отключить прерывание |
| [11] | HSERDYIE | RW | Бит разрешения прерывания по готовности HSE | | 0 |
| 1: | Включить прерывание |
| 0: | Отключить прерывание |
| [10] | HSIRDYIE | RW | Бит разрешения прерывания по готовности HSI | | 0 |
| 1: | Включить прерывание |
| 0: | Отключить прерывание |
| [9] | Res | RO | Резерв | | 0 |
| [8] | LSIRDYIE | RW | Бит разрешения прерывания по готовности LSI | | 0 |
| 1: | Включить прерывание |
| 0: | Отключить прерывание |
| [7] | CSSF | RO | Флаг прерывания по системе безопасности тактового генератора | | 0 |
| 1: | Отказ тактового сигнала HSE, который генерирует прерывание безопасности тактового сигнала CSSI |
| 0: | Нет прерывания от системы безопасности тактового генератора. Устанавливается аппаратно, программная запись бита CSSC со значением 1 очищает его |
| [6:5] | Res | RO | Резерв | | 0 |
| [4] | PLLRDYF | RO | Флаг блокировки прерывания готовности тактового сигнала PLL | | 0 |
| 1: | Блокировка тактового сигнала PLL, вызывающая прерывание |
| 0: | Нет блокировки прерывания тактового сигнала PLL |
| Устанавливается аппаратно, программная запись бита PLLRDYC=1 очищает его | |
| [3] | HSERDYF | RO | Флаг блокировки прерывания готовности тактового сигнала HSE | | 0 |
| 1: | Блокировка тактового сигнала HSE, вызывающая прерывание |
| 0: | Нет блокировки прерывания тактового сигнала HSE |
| Устанавливается аппаратно, программная запись бита HSERDYC=1 очищает его | |
| [2] | HSIRDYF | RO | Флаг блокировки прерывания готовности тактового сигнала HSI | | 0 |
| 1: | Блокировка тактового сигнала HSI, вызывающая прерывание |
| 0: | Нет блокировки прерывания тактового сигнала HSI |
| Устанавливается аппаратно, программная запись бита HSIRDYC=1 очищает его | |
| [1] | Res | RO | Резерв | | 0 |
| [0] | LSIRDYF | RO | Флаг блокировки прерывания готовности тактового сигнала LSI | | 0 |
| 1: | Блокировка тактового сигнала LSI, вызывающая прерывание |
| 0: | Нет блокировки прерывания тактового сигнала LSI |
| Устанавливается аппаратно, программная запись бита LSIRDYC=1 очищает его | |

### 3.4.4 Регистр сброса периферийной шины PB2 (RCC\_APB2PRSTR)

Смещение адреса: 0x0С

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Резерв | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Рез. | USART  1RST | Рез. | SPI1  RST | TIM1  RST | Рез. | ADC1  RST | Резерв | | | IOPD  RST | IOPC  RST | Рез. | IOPA  RST | Рез. | AFIO  RST |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Бит | Название | Доступ | Описание | | Знач. сброса |
| [31:15] | Res | RO | Резерв | | 0 |
| [14] | USART1 RST | RW | Управление сбросом интерфейса USART1 | | 0 |
| 1: | Сброс модуля |
| 0: | Без изменений |
| [15] | Res | RO | Резерв | | 0 |
| [12] | SPI1RST | RW | Управление сбросом интерфейса SPI1 | | 0 |
| 1: | Сброс модуля |
| 0: | Без изменений |
| [11] | TIM1RST | RW | Управление сбросом интерфейса TIM1 | | 0 |
| 1: | Сброс модуля |
| 0: | Без изменений |
| [10] | Res | RO | Резерв | | 0 |
| [9] | ADC1RST | RW | Управление сбросом интерфейса ADC1 | | 0 |
| 1: | Сброс модуля |
| 0: | Без изменений |
| [8:6] | Res | RO | Резерв | | 0 |
| [5] | IOPDRST | RW | Управление сбросом порта ввода-вывода D | | 0 |
| 1: | Сброс модуля |
| 0: | Без изменений |
| [4] | IOPCRST | RW | Управление сбросом порта ввода-вывода C | | 0 |
| 1: | Сброс модуля |
| 0: | Без изменений |
| [3] | Res | RO | Резерв | | 0 |
| [2] | IOPARST | RW | Управление сбросом порта ввода-вывода A | | 0 |
| 1: | Сброс модуля |
| 0: | Без изменений |
| [1] | Res | RO | Резерв | | 0 |
| [0] | AFIORST | RW | Управление сбросом интерфейса вспомогательных функций портов ввода-вывода | | 0 |
| 1: | Сброс модуля |
| 0: | Без изменений |

### 3.4.5 Регистр сброса периферийной шины PB1 (RCC\_APB1PRSTR)

Смещение адреса: 0x10

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Резерв | | | PWRRST | Резерв | | | | | |  | Резерв | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Резерв | | | | WW  DG  RST | Резерв | | | | | | | | | | TIM2  RST |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Бит | Название | Доступ | Описание | | Знач. сброса |
| [31:29] | Res | RO | Резерв | | 0 |
| [28] | PWRRST | RW | Управление сбросом интерфейса управления питанием | | 0 |
| 1: | Сброс модуля |
| 0: | Без изменений |
| [27:22] | Res | RO | Резерв | | 0 |
| [21] | I2C1RST | RW | Управление сбросом интерфейса шины I2C1 | | 0 |
| 1: | Сброс модуля |
| 0: | Без изменений |
| [20:12] | Res | RO | Резерв | | 0 |
| [11] | WWDGRST | RW | Управление сбросом интерфейса оконного сторожевого таймера | | 0 |
| 1: | Сброс модуля |
| 0: | Без изменений |
| [10:1] | Res | RO | Резерв | | 0 |
| [0] | TIM2RST | RW | Управление сбросом интерфейса таймера TIM2 | | 0 |
| 1: | Сброс модуля |
| 0: | Без изменений |

### 3.4.6 Регистр включения тактирования периферийной шины HB (RCC\_AHBPCENR)

Смещение адреса: 0x14

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Резерв | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Резерв | | | | | | | | | | | | | SRAM  EN | Рез. | DMA1  EN |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Бит | Название | Доступ | Описание | | Знач. сброса |
| [31:3] | Res | RO | Резерв | | 0 |
| [2] | SRAMEN | RW | Бит включения тактового сигнала модуля интерфейса SRAM | | 1 |
| 1: | Тактовый сигнал модуля интерфейса SRAM остается активным во время режима сна |
| 0: | Тактовый сигнал модуля интерфейса SRAM отключается в режиме сна |
| [1] | Res | RO | Резерв | | 0 |
| [0] | DMA1EN | RW | Бит включения тактового сигнала модуля DMA1 | | 0 |
| 1: | Тактовый сигнал модуля включен |
| 0: | Тактовый сигнал модуля отключен |

### 3.4.7 Регистр включения тактирования периферийной шины PB2 (RCC\_APB2PCENR)

Смещение адреса: 0x18

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | | 20 | | 19 | | 18 | 17 | 16 |
| Резерв | | | | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | | 4 | | 3 | | 2 | 1 | 0 |
| Рез. | USART1  EN | Рез. | SPI1  EN | TIM1  EN | Рез. | ADC1  EN | Резерв | | | | IOPD  EN | | IOPC  EN | | Рез. | IOPA  EN | Рез. | AFIO  EN |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Бит | Название | Доступ | Описание | | Знач. сброса |
| [31:15] | Res | RO | Резерв | | 0 |
| [14] | USART1EN | RW | Бит включения тактового сигнала интерфейса USART1 | | 0 |
| 1: | Тактовый сигнал включен |
| 0: | Тактовый сигнал отключен |
| [13] | Res | RO | Резерв | | 0 |
| [12] | SPI1EN | RW | Бит включения тактового сигнала интерфейса SPI1 | | 0 |
| 1: | Тактовый сигнал включен |
| 0: | Тактовый сигнал отключен |
| [11] | USART1EN | RW | Бит включения тактового сигнала интерфейса TIM1 | | 0 |
| 1: | Тактовый сигнал включен |
| 0: | Тактовый сигнал отключен |
| [10] | Res | RO | Резерв | | 0 |
| [9] | ADC1EN | RW | Бит включения тактового сигнала интерфейса ADC1 | | 0 |
| 1: | Тактовый сигнал включен |
| 0: | Тактовый сигнал отключен |
| [8:6] | Res | RO | Резерв | | 0 |
| [5] | IOPDEN | RW | Бит включения тактового сигнала порта I/O D | | 0 |
| 1: | Тактовый сигнал включен |
| 0: | Тактовый сигнал отключен |
| [4] | IOPCEN | RW | Бит включения тактового сигнала порта I/O C | | 0 |
| 1: | Тактовый сигнал включен |
| 0: | Тактовый сигнал отключен |
| [3] | Res | RO | Резерв | | 0 |
| [2] | IOPAEN | RW | Бит включения тактового сигнала порта I/O A | | 0 |
| 1: | Тактовый сигнал включен |
| 0: | Тактовый сигнал отключен |
| [1] | Res | RO | Резерв | | 0 |
| [0] | AFIOEN | RW | Бит включения тактового сигнала вспомогательных функций портов | | 0 |
| 1: | Тактовый сигнал включен |
| 0: | Тактовый сигнал отключен |

### 3.4.8 Регистр включения тактирования периферийной шины PB1 (RCC\_APB1PCENR)

Смещение адреса: 0x1С

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Резерв | | | PWREN | Резерв | | | | | | I2C1EN | Резерв | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Резерв | | | | WWDGEN | Резерв | | | | | | | | | | TIM2  EN |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Бит | Название | Доступ | Описание | | Знач. сброса |
| [31:29] | Res | RO | Резерв | | 0 |
| [28] | PWREN | RW | Бит включения тактового сигнала интерфейса управления питанием | | 0 |
| 1: | Тактовый сигнал включен |
| 0: | Тактовый сигнал отключен |
| [27:22] | Res | RO | Резерв | | 0 |
| [21] | I2C1EN | RW | Бит включения тактового сигнала интерфейса I2C1 | | 0 |
| 1: | Тактовый сигнал включен |
| 0: | Тактовый сигнал отключен |
| [20:12] | Res | RO | Резерв | | 0 |
| [11] | WWDGEN | RW | Бит включения тактового сигнала интерфейса WWGD | | 0 |
| 1: | Тактовый сигнал включен |
| 0: | Тактовый сигнал отключен |
| [10:1] | Res | RO | Резерв | | 0 |
| [0] | TIM2EN | RW | Бит включения тактового сигнала интерфейса TIM2 | | 0 |
| 1: | Тактовый сигнал включен |
| 0: | Тактовый сигнал отключен |

### 3.4.9 Control/Status Register (RCC\_RSTSCKR)

Смещение адреса: 0x24

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | | 16 | |
| LPWR  RSTF | WW  DG  RSTF | IW  DG  RSTF | SFT  RSTF | POR  RSTF | PIN  RSTF | Рез. | RM  VF | Резерв | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | 0 | |
| Резерв | | | | | | | | | | | | | | | LSI  RDY | | LSI  ON | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Бит | Название | Доступ | Описание | | Знач. сброса |
| [31] | LPWRRSTF | RO | Флаг сброса по низкому питанию | | 0 |
| 1: | Событие было |
| 0: | События не было |
| Устанавливается в 1 аппаратно при возникновении события; очищается программной записью в бит RMVF | |
| [30] | WWDG RSTF | RO | Флаг события сброса сторожевого таймера окна | | 0 |
| 1: | Событие было |
| 0: | События не было |
| Устанавливается в 1 аппаратно при возникновении события; очищается программной записью в бит RMVF | |
| [29] | IWDG RSTF | RO | Флаг события сброса сторожевого таймера | | 0 |
| 1: | Событие было |
| 0: | События не было |
| Устанавливается в 1 аппаратно при возникновении события; очищается программной записью в бит RMVF | |
| [28] | SFTRSTF | RO | Флаг события программного сброса | | 0 |
| 1: | Событие было |
| 0: | События не было |
| Устанавливается в 1 аппаратно при возникновении события; очищается программной записью в бит RMVF | |
| [27] | PORRSTF | RO | Флаг события сброса PowerUp/PowerDn | | 0 |
| 1: | Событие было |
| 0: | События не было |
| Устанавливается в 1 аппаратно при возникновении события; очищается программной записью в бит RMVF | |
| [26] | PINRSTF | RO | Флаг события сброса по внешнему сбросу (пин NRST) | | 0 |
| 1: | Событие было |
| 0: | События не было |
| Устанавливается в 1 аппаратно при возникновении события; очищается программной записью в бит RMVF | |
| [25] | Res | RO | Резерв | | 0 |
| [24] | RMVF | RW | Флаг управляющий сбросом флагов событий | | 0 |
| 1: | Сбросить флаги событий |
| 0: | Нет эффекта |
| [23:2] | Res | RO | Резерв | | 0 |
| [1] | LSIRDY | RO | Флаг готовности низкочастотного тактового генератора (LSI) устанавливается аппаратно | | 0 |
| 1: | Генератор LSI стабилен |
| 0: | Генератор LSI не стабилен |
| Примечание: После сброса LSION, требуется 3 LSI такта для сброса этого флага | |
| [0] | LSION | RW | Бит включения низкочастотного тактирования (LSI) | | 0 |
| 1: | Включить осциллятор LSI (128КГц) |
| 0: | Отключить осциллятор LSI (128КГц) |

*Примечание: Флаг очистки сброса может быть очищен за исключением бита BIT1, который очищается при включении питания.*

# Глава 4 Независимый сторожевой таймер (IWDG)

Система оснащена независимым сторожевым таймером (IWDG), который предназначен для обнаружения логических ошибок и сбоев программного обеспечения, вызванных внешними воздействиями окружающей среды. Источник тактового сигнала IWDG берется от низкочастотного генератора (LSI) и может работать независимо от основной программы, что делает его подходящим для приложений, где не требуется высокая точность.

## 4.1 Основные возможности

* **12-битный саморазрядуемый счетчик**
* **Источник тактовой частоты**: делитель LSI, может работать в режиме пониженного энергопотребления.
* **Условие сброса**: Значение счетчика уменьшается до нуля

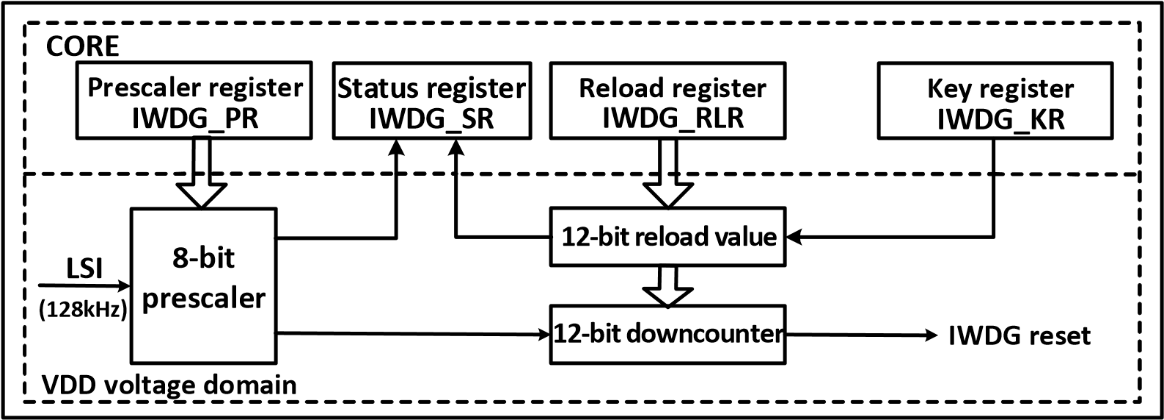
## 4.2 Описание функции

## Начало формы

## 4.2.1 Принцип работы и применение

Независимый сторожевой таймер использует в качестве источника тактового сигнала генератор LSI, и его функция продолжает работать даже в режимах отключения питания и ожидания. Когда счетчик сторожевого таймера самостоятельно уменьшается до нуля, генерируется системный сброс, поэтому время ожидания составляет (значение перезагрузки + 1) тактовый цикл.

Рисунок 4-1 Блок-схема структуры независимого сторожевого таймераНачало формы



* Включить независимый сторожевой таймер

После системного сброса сторожевой таймер выключен, и запись значения 0xCCCC в регистр **IWDG\_CTLR** включает его, после чего он не может быть снова отключён, если только не произойдёт новый сброс. Если бит включения аппаратного независимого сторожевого таймера (**IWDG\_SW**) установлен в байтах пользовательских опций, то IWDG будет автоматически включён после сброса микроконтроллера.

* Конфигурация сторожевого таймера

Сторожевой таймер представляет собой внутренний 12-битный счётчик, который работает по принципу уменьшения. Когда значение счётчика достигает нуля, происходит системный сброс. Чтобы включить функцию IWDG, необходимо выполнить следующие действия:

1. **Базовое время счёта:** Источником тактового сигнала IWDG является LSI; через регистр **IWDG\_PSCR** устанавливается значение деления LSI, которое используется в качестве базового времени счёта для IWDG. Метод работы заключается в том, чтобы сначала записать значение 0x5555 в регистр **IWDG\_CTLR**, а затем изменить значение деления в регистре **IWDG\_PSCR**. Бит **PVU** в регистре состояния **IWDG\_STATR** указывает статус обновления значения деления, и это значение можно изменять и считывать только при завершении обновления.
2. **Значение перезагрузки:** Используется для обновления текущего значения счётчика в автономном сторожевом таймере, причём счётчик уменьшается на это значение. Бит **RVU** в регистре статуса **IWDG\_STATR** показывает состояние обновления значения перезагрузки, и регистр **IWDG\_RLDR** может быть изменён и прочитан только при завершённом обновлении.
3. **Включение сторожевого таймера:** Запись значения 0xCCCC в регистр **IWDG\_CTLR** активирует функцию сторожевого таймера.
4. **«Кормление собаки»:** То есть обновление текущего значения счётчика перед тем, как счётчик сторожевого таймера уменьшится до нуля, чтобы предотвратить системный сброс. Для этого следует записать значение 0xAAAA в регистр **IWDG\_CTLR**, что позволит оборудованию обновить значение регистра **IWDG\_RLDR** в счётчике сторожевого таймера. Это действие должно выполняться регулярно после активации функции сторожевого таймера, иначе произойдёт сброс системы по сигналу сторожевого таймера.

### 4.2.2 Режим наладки

Когда система переходит в режим отладки, счётчик IWDG может быть настроен через регистр модуля отладки таким образом, чтобы продолжать работу или останавливаться.

## 4.3 Описание регистров

Таблица 4-1 Список регистров, связанных с IWDG

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Адрес** | **Описание** | **Значение сброса** |
| R16\_IWDG\_CTLR | 0x40003000 | Регистр управления | 0x0000 |
| R16\_IWDG\_PSCR | 0x40003004 | Регистр прескаллера | 0x0000 |
| R16\_IWDG\_RLDR | 0x40003008 | Регистр перезагрузки | 0x0FFF |
| R16\_IWDG\_STATR | 0x4000300C | Регистр статуса | 0x0000 |

### 4.3.1 Регистр управления (IWDG\_CTLR)

Адрес смещения: 0x00

### 4.3.2 Prescaler Register (IWDG\_PSCR)

Адрес смещения: 0x04

### 4.3.3 Reload Register (IWDG\_RLDR)

Адрес смещения: 0x08

### 4.3.4 Status Register (IWDG\_STATR)

Адрес смещения: 0x0C

*Примечание: После обновления значения предделителя или значения перезагрузки нет необходимости ждать сброса битов RVU или PVU, и выполнение следующего кода может продолжиться. (Эта операция записи будет выполнена до завершения даже в режиме пониженного энергопотребления.)*

# Глава 5 Оконный сторожевой таймер Начало формы

# (WWDG)

Оконный сторожевой таймер обычно используется для мониторинга работы системы на предмет программных сбоев, таких как внешние помехи, непредвиденные логические ошибки и другие условия. Он требует обновления счетчика (подачи сторожевого таймера) в пределах определенного временного окна (с верхним и нижним пределами); в противном случае раньше или позже этого временного окна схема сторожевого таймера сгенерирует системный сброс.

## 5.1 Основные характеристики

* Программируемый 7-битный счетчик с убыванием
* Двойное условие сброса: значение счетчика меньше 0x40 или значение счетчика перезагружается вне временного окна
* Функция раннего уведомления пробуждения (EWI) для своевременной подачи сторожевого таймера, предотвращающей системный сброс

## 5.2 Описание функций

### 5.2.1 Принцип работы и применение

Работа оконного сторожевого таймера основана на 7-битном счетчике с убыванием, который подключен к шине HB и считает частоту деления источника тактовой частоты WWDG\_CLK (HCLK/4096) с коэффициентом деления, установленным в поле WDGTB[1:0] регистра конфигурации WWDG\_CFGR. Счетчик находится в свободном состоянии выполнения, и счет продолжается независимо от того, включена функция сторожевого таймера или нет. На рисунке 5-1 показана блок-схема внутренней структуры оконного сторожевого таймера.

Рисунок 5-1 Блок-схема структуры оконного сторожевого таймера

**-**

**W6**

**W5**

**W4**

**W3**

**W2**

**W1**

**W0**

**WDGA**

**T6**

**T5**

**T4**

**T3**

**T2**

**T1**

**T0**

**/4096**

**WDGTB[1:0]**

**HCLK**

**Watchdog control register(WWDG\_CTLR)**

**RESET**

**Write WWDG\_CTLR[6:0]**

**Watchdog configuration register(WWDG\_CFGR)**

**T[6:0]**

＞

**W[6:0]**

**WWDG\_CLK**

**WWDG enable control, software on**

* Включение оконного сторожевого таймера

После системного сброса сторожевой таймер выключен. Установка бита **WDGA** в регистре **WWDG\_CTLR** включает сторожевой таймер, и после этого его нельзя отключить повторно, пока не произойдет сброс.

*Примечание: Функционирование сторожевого таймера можно косвенно остановить, установив регистр* ***RCC\_APB1PCENR*** *для отключения источника тактовой частоты WWDG и приостановки счета* ***WWDG\_CLK****, либо установив регистр* ***RCC\_APB1PRSTR*** *для сброса модуля WWDG, что эквивалентно роли сброса.*

* Настройка сторожевого таймера

Сторожевой таймер представляет собой внутренний 7-битный счётчик, который непрерывно уменьшает своё значение и поддерживает доступ на чтение и запись. Чтобы воспользоваться функцией сброса сторожевого таймера, необходимо выполнить следующие действия:

1. **База времени счёта:** Через битовые поля **WDGTB**[1:0] регистра **WWDG\_CFGR**, обратите внимание, что модуль WWDG блока RCC должен быть включен.
2. **Счётчик окна:** Установите битовые поля **W**[6:0] в регистре **WWDG\_CFGR**. Этот счётчик используется оборудованием для сравнения с текущим значением счётчика. Его значение задаётся программным обеспечением пользователя и не изменяется. Оно служит максимальным пределом для временного окна.
3. **Включение сторожевого таймера:** Установив бит WDGA в регистре **WWGD\_CTLR** в значение 1, включаем функцию сторожевого таймера. При этом возможно выполнение системного сброса.
4. **«Кормление собаки»:** То есть обновление текущего значения счётчика, для чего необходимо настроить битовые поля **T**[6:0] регистра **WWGD\_CTLR**. Эта операция должна выполняться в рамках заданного временного окна после включения функции сторожевого таймера. В противном случае произойдёт сброс системы по сигналу сторожевого таймера.

* Временное окно «кормления собаки»

Как показано на Рисунке 5-2, заштрихованная область — это зона мониторинга оконного сторожевого таймера. Верхнее время t2 соответствует моменту, когда текущее значение счётчика достигает значения окна W[6:0], а нижнее время t3 — моменту, когда текущее значение счётчика становится равным 0x3F. В течение этого интервала времени t2 < t < t3 можно провести операцию «кормления собаки» (записать T[6:0]), чтобы обновить текущее значение счётчика.

Рисунок 5-2 Режим счёта оконного сторожевого таймера

**RESET**

**T6 bit**

**Refresh not allowed**

**Refresh allowed**

**0**

**x3F**

**W[6:0]**

**Max=0x7F**

**Y[6:0]CNT Current value**

**Time**

**t1**

**t2**

**t3**

**Refresh will be**

**reset within**

**the disallowed**

**refresh time**

**i**

**nd**

**o**

**w**

**ar**

**ea**

**W**

**Timeout:T**

**HCLK1**

**(T[5:0]+1])**

**\***

**\*4096\*2**

**WDGTB**

**The counter will**

**reset when CNT**

**value<0x40**

* Сброс сторожевого таймера

1. Когда значение счётчика **T**[6:0] меняется с 0x40 на 0x3F из-за отсутствия своевременной операции «кормления собаки», произойдёт «сброс оконного сторожевого таймера», и будет сгенерирован системный сброс. То есть, оборудование обнаруживает, что бит **T6** равен нулю, и происходит системный сброс.

*Примечание: Приложение может установить бит* ***T6*** *в ноль программным способом, чтобы вызвать системный сброс, что аналогично функции программного сброса.*

1. Когда выполняется действие обновления счётчика в запрещённое для «кормления» время, т.е. операция записи битового поля **T**[6:0] производится в период времени t1 ≤ t ≤ t2, произойдёт «сброс оконного сторожевого таймера» и будет выполнен системный сброс.

* Предварительное пробуждение

Чтобы предотвратить системный сброс, вызванный несвоевременным обновлением счётчика, модуль сторожевого таймера предоставляет уведомление о раннем прерывании пробуждения (**EWI**). Когда счётчик самостоятельно уменьшается до 0x40, генерируется сигнал раннего пробуждения, и флаг **EWIF** устанавливается в 1. Если бит **EWI** установлен, одновременно будет вызвано прерывание оконного сторожевого таймера. В этот момент остаётся всего один такт счётчика (самоуменьшение до 0x3F) до аппаратного сброса, и приложение может немедленно выполнить операцию «кормления собаки» в течение этого времени.

### 5.2.2 Режим наладки

Когда система переходит в режим отладки, счётчик WWDG может быть настроен через регистр модуля отладки таким образом, чтобы продолжить работу или остановиться.

## 5.3 Описание регистров

Таблица 5-1 Список регистров, связанных с WWDG

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Адрес** | **Описание** | **Значение сброса** |
| R16\_WWDG\_CTLR | 0x40002C00 | Регистр управления | 0x007F |
| R16\_WWDG\_CFGR | 0x40002C04 | Регистр настройки | 0x007F |
| R16\_WWDG\_STATR | 0x40002C08 | Регистр статуса | 0x0000 |

### 5.3.1 Регистр управления (WWDG\_CTLR)

Адрес смещения: 0x00

### 5.3.2 Configuration Register (WWDG\_CFGR)

Адрес смещения: 0x04

### 5.3.3 Status Register (WWDG\_STATR)

Адрес смещения: 0x08

# 

# Глава 6 Прерывания и события

# (PFIC)

Серия CH32V003 имеет встроенный программируемый быстрый контроллер прерываний (PFIC), поддерживающий до 255 векторов прерываний. Текущая система управляет 23 периферийными каналами прерываний и 4 основными каналами прерываний, остальные зарезервированы.

## 6.1 Основные возможности

### 6.1.1 PFIC

* 23 периферийных прерывания, каждое прерывание имеет независимые биты триггера и маски, со специальными битами состояния
* Программируемое многоуровневое вложение прерываний, максимальная глубина вложения 2 уровня, глубина аппаратного стека 2 уровня
* Быстрая система входа и выхода из прерывания с автоматической аппаратной обработкой стека
* Механизм обработки прерываний Vector Table Free (VTF), прямое программирование доступа к адресам вектора прерывания двумя способами

## 6.2 Системный таймер

 Серия CH32V003

Ядро оснащено 32-битным суммирующим счётчиком (SysTick), который поддерживает HCLK или HCLK/8 в качестве временной базы с высоким приоритетом и может использоваться в качестве эталонного времени после калибровки.

## 6.3 Таблица векторов прерываний и исключений

Таблица 6-1 Таблица векторов серии CH32V003

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Приоритет | Тип | Имя | Описание | Адрес входа |
| 0 | - | - | - | - | 0x00000000 |
| 1 | - | - | - | - | 0x00000004 |
| 2 | -2 | fixed | NMI | Немаскируемые прерывания | 0x00000008 |
| 3 | -1 | fixed | HardFault | Аномальные прерывания | 0x0000000С |
| 4-11 | - | - |  | Зарезервировано | 0x00000010- 0x0000002С |
| 12 | 0 | prog | SysTick | Прерывание системного таймера | 0x00000030 |
| 13 | - | - |  | Зарезервировано | 0x00000034 |
| 14 | 1 | prog | SW | Програмные прерывания | 0x00000038 |
| 15 | - | - |  | Зарезервировано | 0x0000003С |
| 16 | 2 | prog | WWDG | Прерывание таймера оконного сторожевого таймера | 0x00000040 |
| 17 | 3 | prog | PVD | Прерывание детектора напряжения питания (EXTI) | 0x00000044 |
| 18 | 4 | prog | FLASH | Глобальное прерывание Flash | 0x00000048 |
| 19 | 5 | prog | RCC | Прерывания сброса и тактирования | 0x0000004С |
| 20 | 6 | prog | EXTI7\_0 | Прерывания EXTI линии 0-7 | 0x00000050 |
| 21 | 7 | prog | AWU | Прерывание пробуждения | 0x00000054 |
| 22 | 8 | prog | DMA\_CH1 | Глобальное прерывание DMA1 канал 1 | 0x00000058 |
| 23 | 9 | prog | DMA\_CH2 | Глобальное прерывание DMA1 канал 2 | 0x0000005С |
| 24 | 10 | prog | DMA\_CH3 | Глобальное прерывание DMA1 канал 3 | 0x00000060 |
| 25 | 11 | prog | DMA\_CH4 | Глобальное прерывание DMA1 канал 4 | 0x00000064 |
| 26 | 12 | prog | DMA\_CH5 | Глобальное прерывание DMA1 канал 5 | 0x00000068 |
| 27 | 13 | prog | DMA\_CH6 | Глобальное прерывание DMA1 канал 6 | 0x0000006С |
| 28 | 14 | prog | DMA\_CH7 | Глобальное прерывание DMA1 канал 7 | 0x00000070 |
| 29 | 15 | prog | ADC | Глобальное прерывание АЦП | 0x00000074 |
| 30 | 16 | prog | I2C1\_EV | Прерывание событий I2C1 | 0x00000078 |
| 31 | 17 | prog | I2C1\_ER | Прерывание ошибок I2C1 | 0x0000007C |
| 32 | 18 | prog | USART1 | Глобальное прерывание USART1 | 0x00000080 |
| 33 | 19 | prog | SPI1 | Глобальное прерывание SPI1 | 0x00000084 |
| 34 | 20 | prog | TIM1BRK | Прерывание останова TIM1 | 0x00000088 |
| 35 | 21 | prog | TIM1UP | Прерывание обновления TIM1 | 0x0000008С |
| 36 | 22 | prog | TIM1TRG | Прерывание вызываемое TIM1 | 0x00000090 |
| 37 | 23 | prog | TIM1CC | Прерывание TIM1 захвата и сравнения | 0x00000094 |
| 38 | 24 | prog | TIM2 | Глобальное прерывание TIM2 | 0x00000098 |

## 6.4 Контроллер внешних прерываний и событий (EXTI)

### 6.4.1 Обзор

Рисунок 6-1 Блок-схема интерфейса внешнего прерывания (EXTI)

**HBbus**

**Peripheral interface**

**INTFR**

**INTENR**

**SWIEVR**

**RTENR**

**FTENR**

**HCLK**

**10**

**10**

**10**

**10**

**10**

**Pulse**

**generator**

**EVENR**

**Edge detect**

**circuit**

**To PFIC interrupt**

**controller**

**10**

**10**

**10**

**10**

**10**

**10**

**10**

**10**

**Input**

**Line**

Как видно из рисунка 6-1, источником запуска внешнего прерывания может быть программное прерывание (SWIEVR) или реальный внешний канал прерывания. Сигнал внешнего канала прерывания сначала пройдет через схему детектора фронтов. Как только возникает одно из сигналов программного прерывания или внешнего прерывания, оно будет выведено на две схемы с вентилем ИЛИ, разрешения события и разрешения прерывания, через схему с вентиль ИЛИ на рисунке. Пока разрешено хотя бы одно прерывание или событие, будет сгенерировано прерывание или событие. Процессор получает доступ к шести регистрам EXTI через интерфейс HB.Начало формы

### 6.4.2 Событие пробуждения

Система может вывести микроконтроллер из режима сна, вызванного командой WFE, посредством события пробуждения. Событие пробуждения генерируется одним из двух следующих способов:

* **Разрешение прерывания в периферийном регистре**, но без разрешения этого прерывания в PFIC ядра, и одновременное разрешение бита **SEVONPEND** в ядре. Например, в EXTI это означает разрешить прерывание EXTI, но не разрешать прерывание EXTI в PFIC, и одновременно разрешить бит **SEVONPND**. Когда микроконтроллер просыпается от команды WFE, ему нужно сбросить бит флага прерывания EXTI и бит ожидания PFIC.
* **Конфигурация канала EXTI как канала события** устраняет необходимость для микроконтроллера сбрасывать бит флага прерывания и бит ожидающего состояния PFIC после пробуждения от команды WFE.

### 6.4.3 Описание

Использование внешнего прерывания требует настройки соответствующего канала внешнего прерывания, то есть выбора соответствующего фронта срабатывания и разрешения соответствующего прерывания. Когда установленный фронт срабатывания появляется на канале внешнего прерывания, генерируется запрос на прерывание, и соответствующий бит флага прерывания устанавливается. Флаг можно сбросить, записав 1 в бит флага.

Шаги использования внешних аппаратных прерываний.

1. Конфигурация операций GPIO.
2. Настройка бита разрешения прерывания (**EXTI\_INTENR**) для соответствующего канала внешнего прерывания.
3. Настройка фронта срабатывания (**EXTI\_RTENR** или **EXTI\_FTENR**) для выбора нарастающего фронта, спадающего фронта или двойного фронта.
4. Настройка прерываний EXTI в PFIC ядра, чтобы обеспечить их корректное реагирование.

Шаги для использования внешних аппаратных событий.

1. Конфигурация операций GPIO.
2. Настройте бит разрешения события (**EXTI\_EVENR**) для соответствующего канала внешнего прерывания.
3. Настройте фронт срабатывания (**EXTI\_RTENR** или **EXTI\_FTENR**) для выбора нарастающего фронта, спадающего фронта или двойного фронта.

Шаги по использованию программного прерывания/события.

1. Разрешите внешние прерывания (**EXTI\_INTENR**) или внешние события (**EXTI\_EVENR**).
2. Если используются функции обслуживания прерываний, прерывание EXTI необходимо настроить в PFIC ядра.
3. Установите запуск программного прерывания (**EXTI\_SWIEVR**), то есть будет сгенерировано прерывание.

### 6.4.4 Внешняя карта событий

Таблица 6-2 Соответствие прерываний EXTI

## 6.5 Описание регистров

### 6.5.1 Регистры EXTI

Таблица 6-3 Список регистров, связанных с EXTI

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Адрес** | **Описание** | **Значение сброса** |
| R32\_EXTI\_INTENR | 0x40010400 | Регистр включения прерываний | 0x00000000 |
| R32\_EXTI\_EVENR | 0x40010404 | Регистр включения событий | 0x00000000 |
| R32\_EXTI\_RTENR | 0x40010408 | Регистр включения триггеров восходящих фронтов | 0x00000000 |
| R32\_EXTI\_FTENR | 0x4001040C | Регистр включения триггеров нисходящих фронтов | 0x00000000 |
| R32\_EXTI\_SWIEVR | 0x40010410 | Регистр программного прерывания события | 0x00000000 |
| R32\_EXTI\_INTFR | 0x40010414 | Регистр флагов прерываний | 0x0000xxxx |

#### 6.5.1.1 Регистр включения прерываний (EXTI\_INTENR)

Адрес смещения: 0x00

#### 6.5.1.2 Event Enable Register (EXTI\_EVENR)

Адрес смещения: 0x00

#### 6.5.1.3 Rising Edge Trigger Enable Register (EXTI\_RTENR)

Адрес смещения: 0x00

#### 6.5.1.4 Falling Edge Trigger Enable Register (EXTI\_FTENR)

Адрес смещения: 0x00

#### 6.5.1.5 Software Interrupt Event Register (EXTI\_SWIEVR)

Адрес смещения: 0x00

#### 6.5.1.6 Interrupt Flag Register (EXTI\_INTFR)

Адрес смещения: 0x00

### 6.5.2 PFIC Registers

Таблица 6-4 Список регистров, связанных с PFIC

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Адрес** | **Описание** | **Значение сброса** |
| R32\_PFIC\_ISR1 | 0xE000E000 | Регистр состояния разрешения прерывания PFIC 1 | 0x0000000C |
| R32\_PFIC\_ISR2 | 0xE000E004 | Регистр состояния разрешения прерывания PFIC 2 | 0x00000000 |
| R32\_PFIC\_IPR1 | 0xE000E020 | Регистр состояния ожидания прерывания PFIC 1 | 0x00000000 |
| R32\_PFIC\_IPR2 | 0xE000E024 | Регистр состояния ожидания прерывания PFIC 2 | 0x00000000 |
| R32\_PFIC\_  ITHRESDR | 0xE000E040 | Регистр конфигурации порога приоритета прерывания PFIC | 0x00000000 |
| R32\_PFIC\_CFGR | 0xE000E048 | Регистр конфигурации прерывания PFIC | 0x00000000 |
| R32\_PFIC\_GISR | 0xE000E04C | Регистр глобального состояния прерывания PFIC | 0x00000000 |
| R32\_PFIC\_VTFIDR | 0xE000E050 | Регистр конфигурации идентификатора VTF прерывания PFIC | 0x00000000 |
| R32\_PFIC\_  VTFADDRR0 | 0xE000E060 | Регистр смещения адреса прерывания VTF PFIC 0 | 0x00000000 |
| R32\_PFIC\_  VTFADDRR1 | 0xE000E064 | Регистр смещения адреса прерывания VTF PFIC 1 | 0x00000000 |
| R32\_PFIC\_IENR1 | 0xE000E100 | Регистр установки разрешения прерывания PFIC 1 | 0x00000000 |
| R32\_PFIC\_IENR2 | 0xE000E104 | Регистр установки разрешения прерывания PFIC 2  Начало формы  Конец формы | 0x00000000 |
| R32\_PFIC\_IRER1 | 0xE000E180 | Регистр очистки разрешения прерывания PFIC 1 | 0x00000000  Начало формы  Конец формы |
| R32\_PFIC\_IRER2 | 0xE000E184 | Регистр очистки разрешения прерывания PFIC 2 | 0x00000000 |
| R32\_PFIC\_IPSR1 | 0xE000E200 | Регистр установки ожидания прерывания PFIC 1 | 0x00000000 |
| R32\_PFIC\_IPSR2 | 0xE000E204 | Регистр установки ожидания прерывания PFIC 2 | 0x00000000 |
| R32\_PFIC\_IPRR1 | 0xE000E280 | Регистр очистки ожидания прерывания PFIC 1 | 0x00000000 |
| R32\_PFIC\_IPRR2 | 0xE000E284 | Регистр очистки ожидания прерывания PFIC 2 | 0x00000000 |
| R32\_PFIC\_IACTR1 | 0xE000E300 | Регистр состояния активации прерывания PFIC 1 | 0x00000000 |
| R32\_PFIC\_IACTR2 | 0xE000E304 | Регистр состояния активации прерывания PFIC 2 | 0x00000000 |
| R32\_PFIC\_IPRIORx | 0xE000E400 | Регистр конфигурации приоритета прерывания PFIC | 0x00000000 |
| R32\_PFIC\_SCTLR | 0xE000ED10 | Регистр управления системой PFIC | 0x00000000 |

*Примечание:*

1. *Значение по умолчанию регистра PFIC\_ISR1 равно 0xC, то есть NMI и исключение всегда включены по умолчанию.*
2. *NMI и EXC поддерживают операции очистки и установки ожидания прерывания, но не поддерживают операции очистки и установки разрешения прерывания.*

#### 6.5.2.1 PFIC Interrupt Enable Status Register 1 (PFIC\_ISR1)

Адрес смещения: 0x00

#### 6.5.2.2 PFIC Interrupt Enable Status Register 2 (PFIC\_ISR2)

Адрес смещения: 0x00

#### 6.5.2.3 PFIC Interrupt Pending Status Register 1 (PFIC\_IPR1)

Адрес смещения: 0x00

#### 6.5.2.4 PFIC Interrupt Pending Status Register 2 (PFIC\_IPR2)

Адрес смещения: 0x00

#### 6.5.2.5 PFIC Interrupt Priority Threshold Configuration Register (PFIC\_ITHRESDR)

Адрес смещения: 0x00

#### 6.5.2.6 PFIC Interrupt Configuration Register (PFIC\_CFGR)

Адрес смещения: 0x00

#### 6.5.2.7 PFIC Interrupt Global Status Register (PFIC\_GISR)

Адрес смещения: 0x00

#### 6.5.2.8 PFIC VTF Interrupt ID Configuration Register (PFIC\_VTFIDR)

Адрес смещения: 0x00

#### 6.5.2.9 PFIC VTF Interrupt 0 Address Register (PFIC\_VTFADDRR0)

Адрес смещения: 0x00

#### 6.5.2.10 PFIC VTF Interrupt 1 Address Register (PFIC\_VTFADDRR1)

Адрес смещения: 0x00

#### 6.5.2.11 PFIC Interrupt Enable Setting Register 1 (PFIC\_IENR1)

Адрес смещения: 0x00

**6.5.2.12 PFIC Interrupt Enable Setting Register 2 (PFIC\_IENR2)**

Адрес смещения: 0x00

#### 6.5.2.13 PFIC Interrupt Enable Clear Register 1 (PFIC\_IRER1)

Адрес смещения: 0x00

#### 6.5.2.14 PFIC Interrupt Enable Clear Register 2 (PFIC\_IRER2)

Адрес смещения: 0x00

#### 6.5.2.15 PFIC Interrupt Pending Setup Register 1 (PFIC\_IPSR1)

Адрес смещения: 0x00

#### 6.5.2.16 PFIC Interrupt Pending Setup Register 2 (PFIC\_IPSR2)

Адрес смещения: 0x00

#### 6.5.2.17 PFIC Interrupt Pending Clear Register 1 (PFIC\_IPRR1)

Адрес смещения: 0x00

#### 6.5.2.18 PFIC Interrupt Pending Clear Register 2 (PFIC\_IPRR2)

Адрес смещения: 0x00

#### 6.5.2.19 PFIC Interrupt Activation Status Register 1 (PFIC\_IACTR1)

Адрес смещения: 0x00

#### 6.5.2.20 PFIC Interrupt Activation Status Register 2 (PFIC\_IACTR2)

Адрес смещения: 0x00

#### 6.5.2.21 PFIC Interrupt Priority Configuration Register (PFIC\_IPRIORx) (x=0-63)

Адрес смещения: 0x00

Контроллер поддерживает 256 прерываний (от 0 до 255), каждое из которых использует 8 бит для установки управляющего приоритета.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 31 | 24 | 23 | 16 | 15 | 8 | 7 | 0 |
| IPRIOR63 | PRIO\_255 | | PRIO\_254 | | PRIO\_253 | | PRIO\_252 | |
| … | … | | … | | … | | … | |
| IPRIORx | PRIO\_(4x+3) | | PRIO\_(4x+2) | | PRIO\_(4x+1) | | PRIO\_(4x+0) | |
| … | … | | … | | … | | … | |
| IPRIOR0 | PRIO\_3 | | PRIO\_2 | | PRIO\_1 | | PRIO\_0 | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Бит** | **Имя** | **Доступ** | **Описание** | **Значение сброса** |
| [2047:2040] | IP\_255 | RW | Такое же что и у IP\_0 | 0 |
| … | … | … | … | … |
| [31:24] | IP\_3 | RW | Такое же что и у IP\_0 | 0 |
| [23:16] | IP\_2 | RW | Такое же что и у IP\_0 | 0 |
| [15:8] | IP\_1 | RW | Такое же что и у IP\_0 | 0 |
| [7:0] | IP\_0 | RW | Конфигурация приоритета прерывания ноль. [7:6:4]: биты управления приоритетом. Если вложенность не настроена, отсутствуют биты предварительной выборки. Бит 7 является предварительным выбором, если настроены два уровня вложенности. [5:0]: зарезервировано, установлено на 0, запись недействительна. | 0 |

#### 6.5.2.22 Регистр управления системой PFIC (PFIC\_SCTLR)

### 6.5.3 Специальные регистры CSR

Архитектура RISC-V определяет ряд регистров управления и состояния (CSR), предназначенных для настройки, идентификации или регистрации рабочего состояния. Регистры CSR являются внутренними для ядра и используют выделенное 12-битное адресное пространство; микросхема CH32V003 дополнительно содержит несколько регистров, определенных производителем, помимо стандартных регистров, описанных в документации привилегированной архитектуры RISC-V, к которым требуется доступ с использованием инструкции csr.

*Примечание: Эти регистры помечены как "MRW, MRO, MRW1" и требуют, чтобы система находилась в машинном режиме для доступа к ним.*

#### 6.5.3.1 Interrupt System Control Register (INTSYSCR)

#### 6.5.3.2 Exception Entry Base Address Register (MTVEC)

### 6.5.4 STK Register Description

Table 6-5 STK-related registers list

#### 6.5.4.1 System Count Control Register (STK\_CTLR)

#### 6.5.4.2 System Count Status Register (STK\_SR)

#### 6.5.4.3 System Counter Register (STK\_CNTL)

#### 6.5.4.4 Counting Comparison Register (STK\_CMPLR)

# Глава 7 Общие входы-выходы (GPIO) и их альтернативные функции (GPIO/AFIO)

Порт GPIO может быть сконфигурирован для различных режимов ввода или вывода, оснащен встроенными подтягивающими или тянущими резисторами, которые могут быть отключены, и может быть настроен для выполнения функций типа push-pull или open-drain. Порт GPIO также может мультиплексироваться для других функций.

## 7.1 Основные характеристики

Каждый контакт порта может быть настроен на один из нескольких режимов:

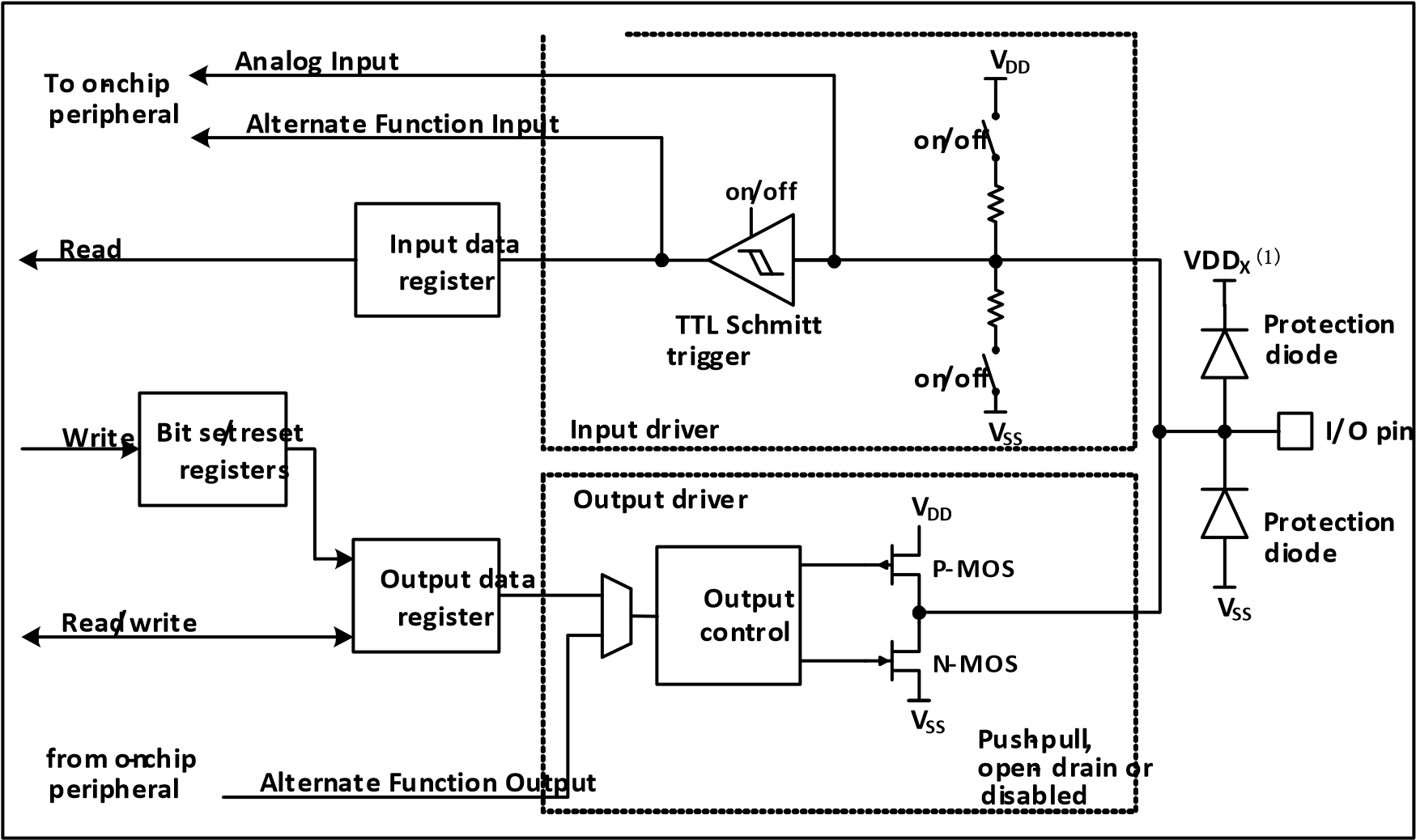
* Floating input (вход с плавающим потенциалом)
* Open drain output (выход с открытым стоком)
* Pull-up input (вход с подтяжкой вверх)
* Push-pull output (выход типа push-pull)
* Dropdown input (вход с подтягиванием вниз)
* Multiplexing the inputs and outputs of functions (мультиплексирование входов и выходов функций)

Многие контакты имеют возможность мультиплексирования, и многие другие периферийные устройства отображают свои выходные и входные каналы на эти контакты. Конкретное использование этих мультиплексированных контактов должно определяться индивидуально для каждого периферийного устройства, а содержание того, мультиплексируются ли эти контакты и переназначаются, объясняется в данной главе.

## 7.2 Описание функций

### 7.2.1 Обзор

Рисунок 7-1 Блок-схема базовой структуры модуля GPIO



*Примечание: (1) VDDx — это VDD, когда GPIO работает в нормальном режиме ввода-вывода, и VDDx — это VDD\_FT, когда GPIO используется в режиме FT.*

Как показано на рисунке 7-1 структура порта ввода-вывода, каждый контакт имеет две защитные диоды внутри чипа, и порт ввода-вывода может быть разделён на модули драйвера ввода и вывода. Из них драйвер ввода имеет опциональные слабые подтягивающие и тянущие резисторы, которые могут быть подключены к АЦП и другим аналоговым периферийным устройствам; если ввод осуществляется на цифровое периферийное устройство, то он должен пройти через триггер Шмитта TTL, а затем подключиться к регистрам ввода GPIO или другим мультиплексированным периферийным устройствам. Драйвер вывода имеет пару МОП-транзисторов, и порт ввода-вывода можно настроить на открытый сток или выходной сигнал типа push-pull путём настройки того, включены ли верхний и нижний МОП-транзисторы; драйвер вывода также может быть настроен внутри для управления выводом через GPIO или другие мультиплексированные периферийные устройства.

### 7.2.2 Функция инициализации GPIO

Сразу после сброса порты GPIO работают в начальном состоянии, при котором большинство портов ввода-вывода находятся в состоянии плавающего ввода, но существуют также связанные с периферией выводы, такие как HSE, работающие на функции мультиплексирования периферии. За конкретной информацией об инициализации обратитесь к главе, связанной с описанием выводов.

### 7.2.3 Внешние прерывания

Все порты GPIO могут быть настроены с внешними каналами ввода прерываний, но один внешний канал ввода прерываний может быть сопоставлен максимум с одним контактом GPIO, и номер последовательного порта внешнего прерывания должен совпадать с номером бита порта GPIO. Например, PA1 (или PC1, PD1 и т.д.) может быть сопоставлено только с EXTI1, и EXTI1 может принимать только один из PA1, PC1 или PD1 и так далее. Отображение обеих сторон является взаимно однозначным.

### 7.2.4 Функции мультиплексирования

Важно отметить следующее при использовании функции мультиплексирования:

* Для использования функции мультиплексирования в направлении ввода порт должен быть настроен в мультиплексированный режим ввода, а настройка подтягивания может быть установлена в соответствии с фактическими потребностями.
* Для использования функции мультиплексирования в направлении вывода порт должен быть настроен в мультиплексированный режим вывода, и можно выбрать тип push-pull или open-drain в зависимости от реальной ситуации.
* В случае двунаправленного мультиплексирования порт должен быть настроен в режим мультиплексированного вывода, а драйвер — в режим плавающего ввода.

Один и тот же порт ввода-вывода может иметь несколько периферийных устройств, мультиплексируемых на этот контакт, поэтому для максимального увеличения пространства для каждой периферии мультиплексированные контакты периферийных устройств могут быть переназначены на другие контакты в дополнение к контактам по умолчанию, избегая занятых контактов.

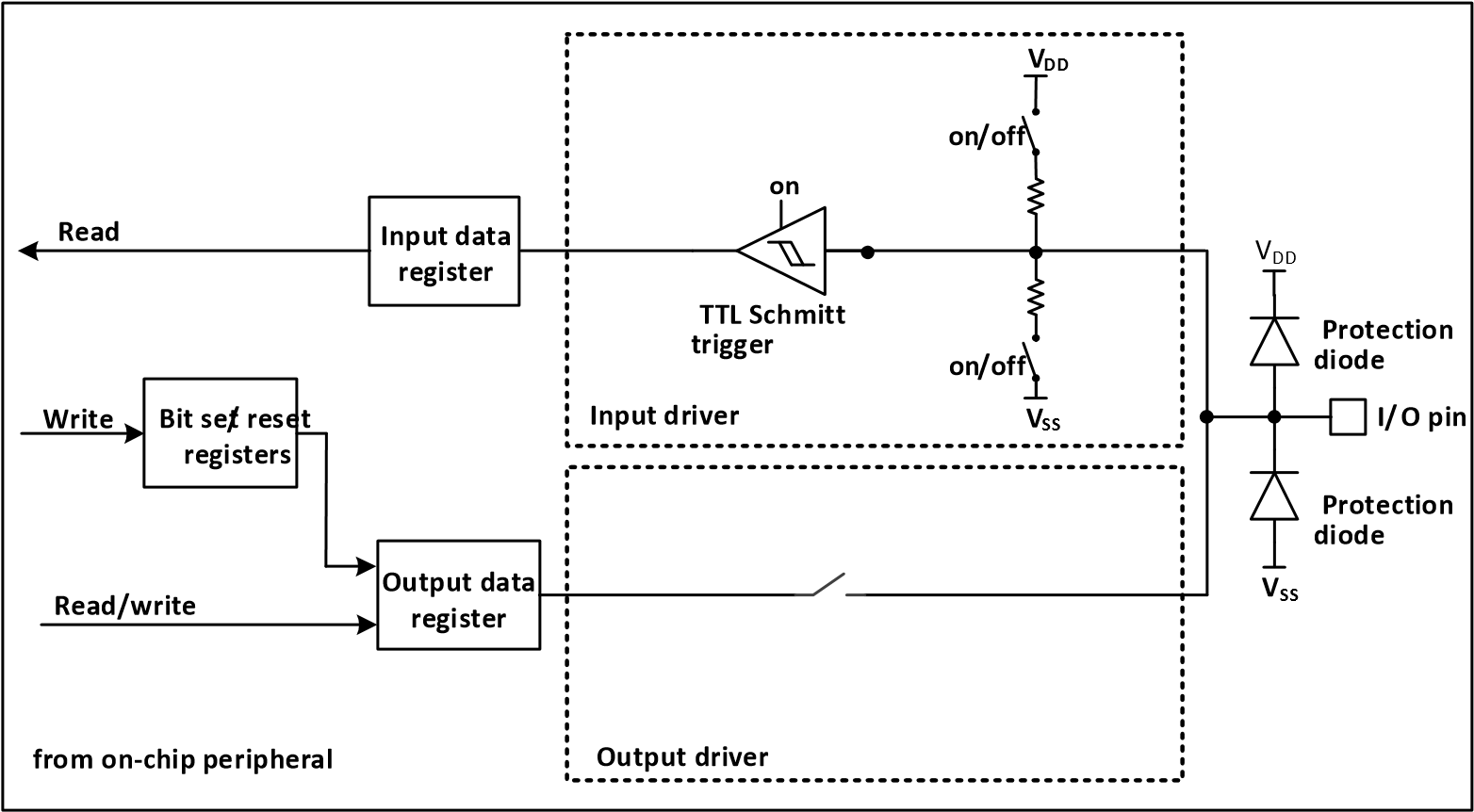
### 7.2.5 Механизм блокировки (

### Locking Mechanism)

Механизм блокировки фиксирует конфигурацию порта ввода-вывода. После определенной последовательности записи конфигурация выбранного контакта ввода-вывода будет заблокирована и не сможет изменяться до следующего сброса.

### 7.2.6 Настройка входа

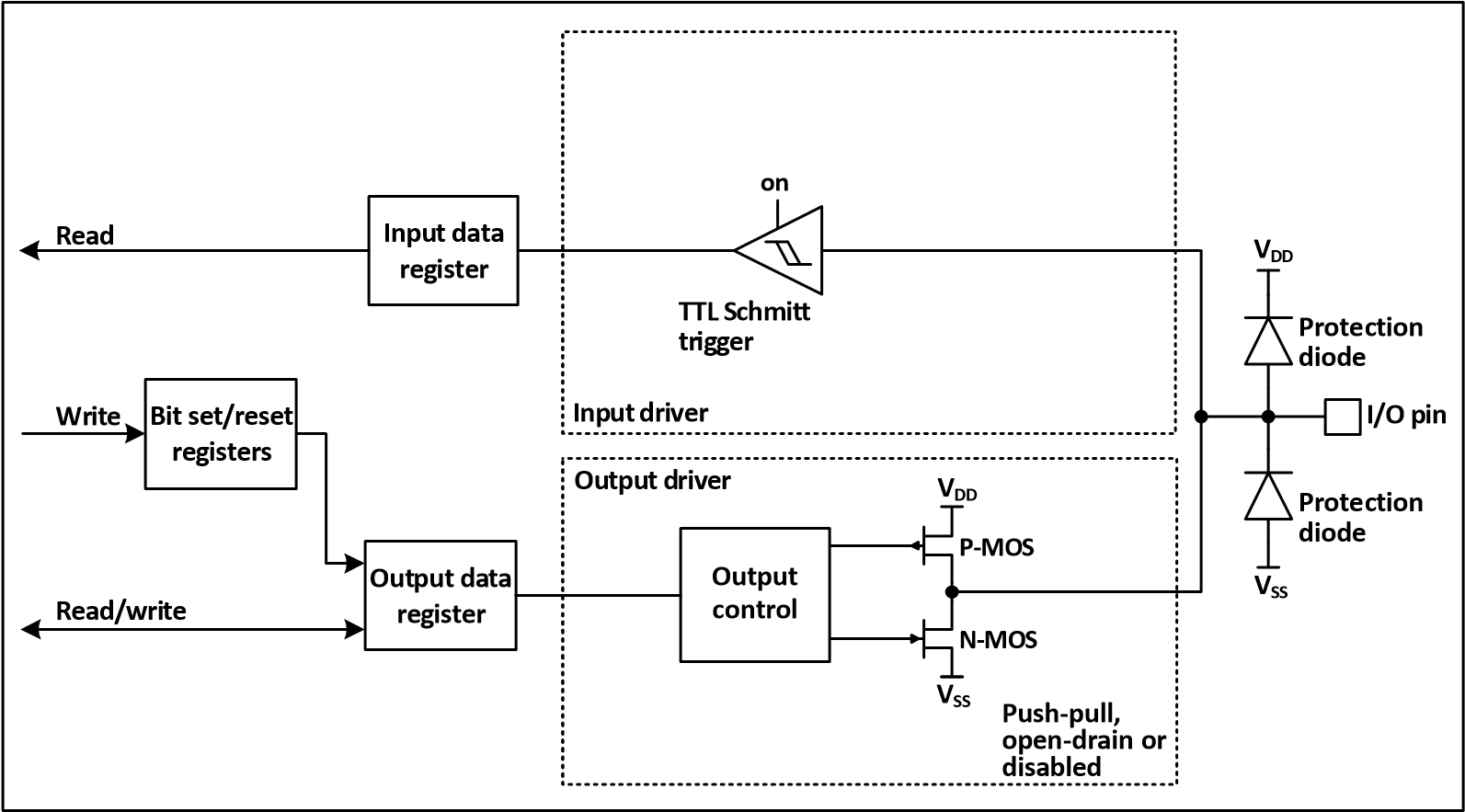
Рисунок 7-2 Блок-схема структуры конфигурации ввода модуля GPIO



Когда порт ввода-вывода настроен в режим ввода, выходной драйвер отключается, подтягивание и опускание ввода выбираются, и никакие мультиплексированные функции или аналоговые входы не подключаются. Данные на каждом порту ввода-вывода собираются в регистр данных ввода на каждом такте HB, и уровень сигнала соответствующего контакта определяется чтением соответствующего бита регистра данных ввода.

### 7.2.7 Конфигурация выхода

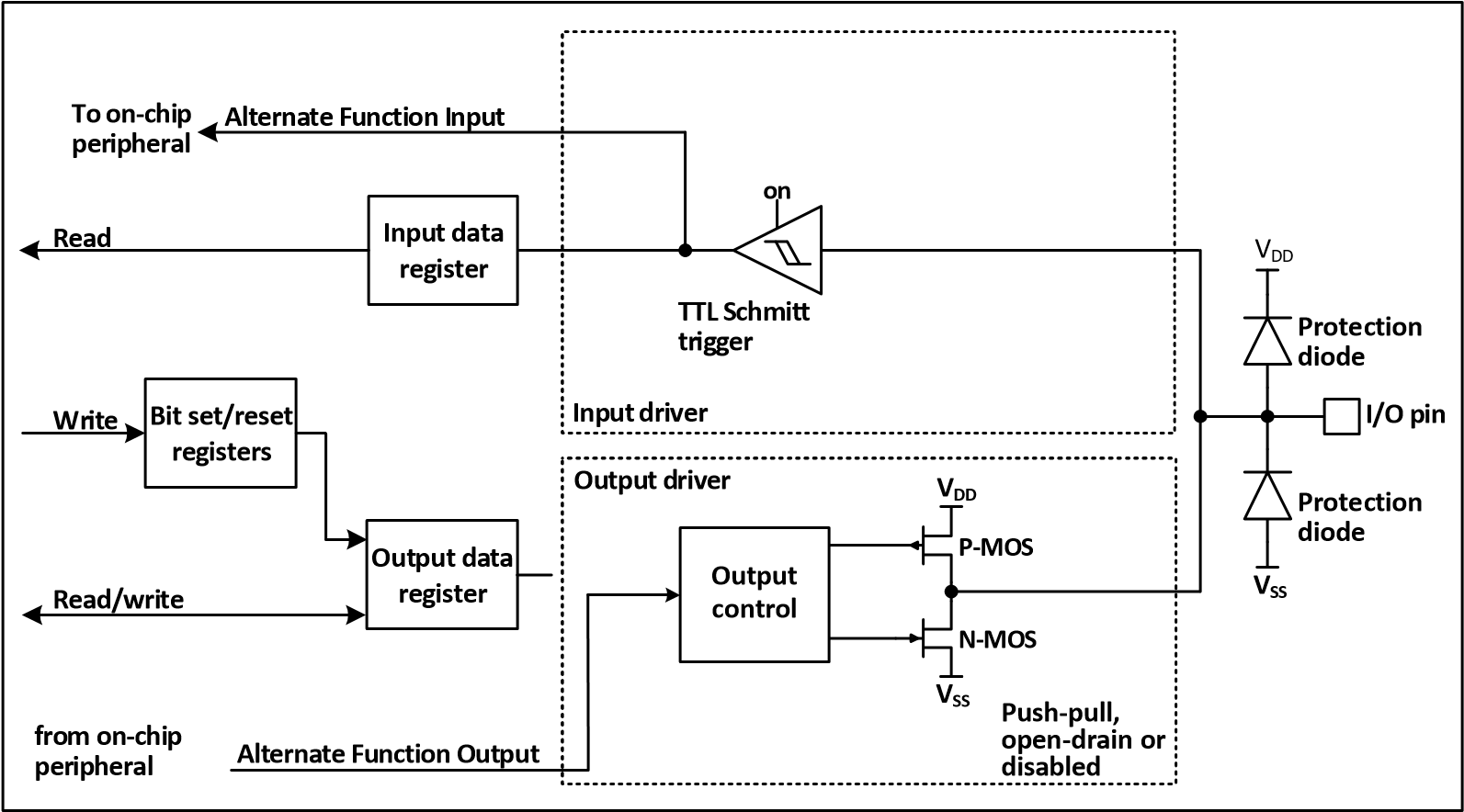
Рисунок 7-3 Блок-схема структуры конфигурации вывода модуля GPIO



Когда порт ввода-вывода настроен на режим вывода, пара МОП-транзисторов в выходном драйвере может быть настроена на режим push-pull или open-drain в зависимости от необходимости, без использования функции мультиплексирования. Подтягивающие и тянущие резисторы драйвера ввода отключаются, активируется триггер Шмитта TTL, и уровни, появляющиеся на контактах ввода-вывода, будут собираться в регистры данных ввода на каждом такте HB. Таким образом, чтение регистров данных ввода даст состояние ввода-вывода, а в режиме вывода push-pull доступ к регистрам данных вывода даст последнее записанное значение.

### 7.2.8 Конфигурация функции мультиплексирования

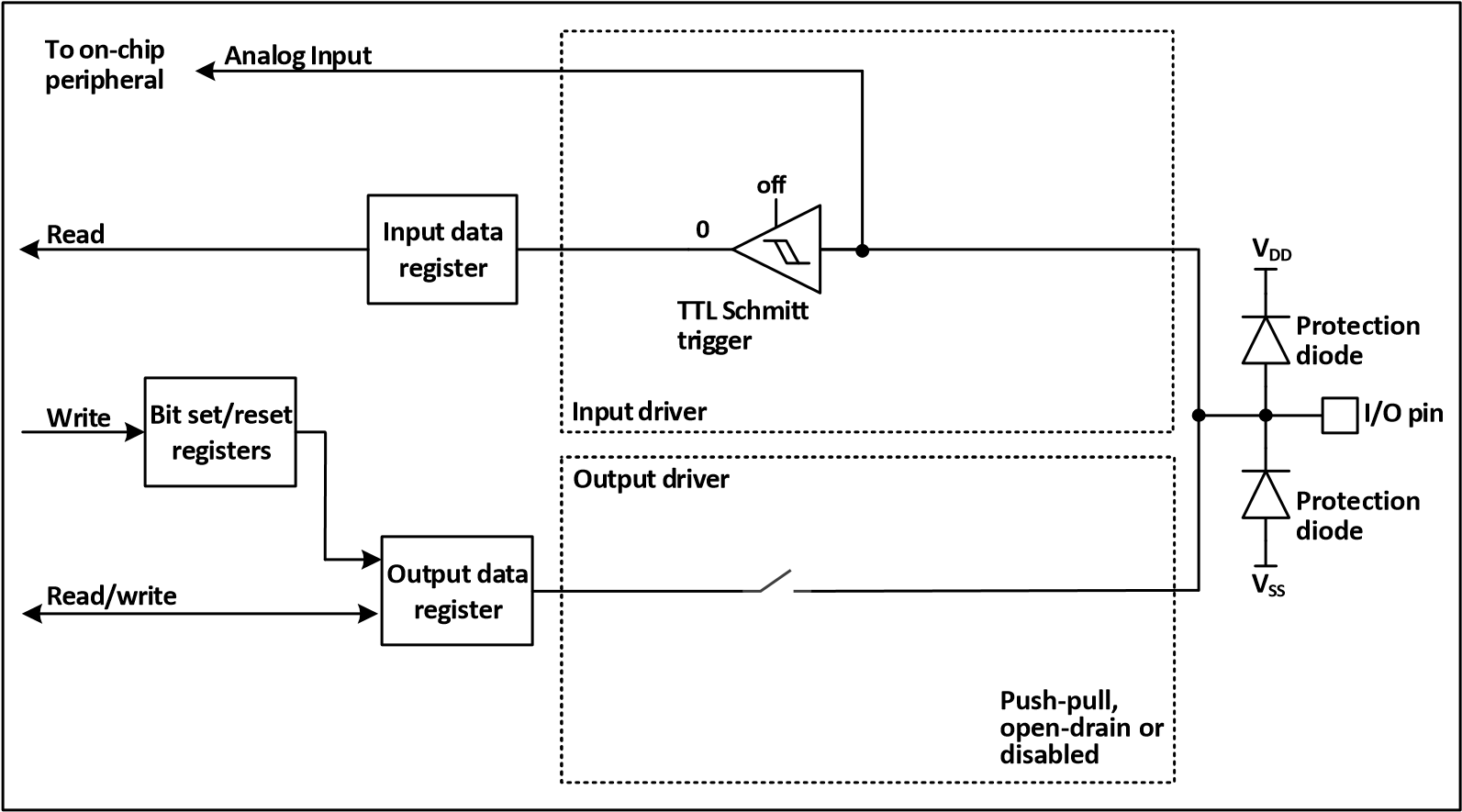
Рисунок 7-4 Структура модуля GPIO при его мультиплексировании другой периферией



При включении мультиплексирования включается выходной драйвер, который может быть настроен на режим открытого стока или push-pull в зависимости от требований, включается триггер Шмитта, соединяются линии ввода и вывода функции мультиплексирования, но отключаются регистры выходных данных, и уровни, появляющиеся на контактах ввода-вывода, будут собираться в регистры данных ввода на каждом такте HB. В режиме открытого стока чтение регистра данных ввода даст текущее состояние порта ввода-вывода; в режиме push-pull чтение регистра выходных данных даст последнее записанное значение.

### 7.2.9 Конфигурация аналогового входа

Рисунок 7-5 Структура конфигурации модуля GPIO в качестве аналогового входа



Когда включен аналоговый вход, буфер вывода отключается, вход триггера Шмитта в драйвере ввода отключается для предотвращения генерации потребления на порту ввода-вывода, подтягивающие и тянущие резисторы отключаются, и регистр чтения входных данных всегда будет содержать 0.

### 7.2.10 Настройки GPIO для периферийных устройств

В следующей таблице рекомендуются соответствующие конфигурации порта GPIO для каждого контакта периферийного устройства.

Начало формы

Table 7-1 Advanced-control timer (TIM1)

### 7.2.11 Alternate Function Remapping GPIO Configuration 7.2.11.1 Timer Alternate Function Remapping

*Note: For the mapping function of TIM1\_CH1 in the table, the condition is TIM1\_1\_RM=0. When TIM1\_1\_RM=1, TIM1\_CHI is mapped to LSI (for LSI calibration).*

Table 7-9 TIM2 alternate function remapping

#### 7.2.11.2 USART Alternate Function Remapping

Table 7-10 USART1 alternate function remapping

#### 7.2.11.3 SPI Alternate Function Remapping

Table 7-11 SPI alternate function remapping

#### 7.2.11.4 I2C Alternate Function Remapping

Table 7-12 I2C alternate function remapping

#### 7.2.11.4 ADC Alternate Function Remapping

Table 7-13 ADC external trigger injection conversion alternate function remapping

Table 7-13 ADC external trigger rule conversion alternate function remapping

## 7.3 Register Description

### 7.3.1 GPIO Register Description

Unless otherwise specified, the registers of the GPIO must be operated as words (operate these registers with 32 bits).

Table 7-8 GPIO-related registers list

#### 7.3.1.1 Port Configuration Register Low (GPIOx\_CFGLR) (x=A/C/D) Offset address: 0x00

#### 7.3.1.2 Port Input Register (GPIOx\_INDR) (x=A/C/D)

#### 7.3.1.3 Port Output Register (GPIOx\_OUTDR) (x=A/C/D)

#### 7.3.1.4 Port Reset/Set Register (GPIOx\_BSHR) (x=A/C/D)

#### 7.3.1.5 Port Reset Register (GPIOx\_BCR) (x=A/C/D)

#### 7.3.1.6 Port Configuration Lock Register (GPIOx\_LCKR) (x=A/C/D) Offset address: 0x18

### 7.3.2 AFIO Register Description

Unless otherwise specified, AFIO registers must be operated as words (operate these registers with 32 bits).

Table 7-9 List of AFIO-related registers

**7.3.2.1 Remap Register 1 (AFIO\_PCFR1)**

Offset address: 0x04

**7.3.2.2 External Interrupt Configuration Register 1 (AFIO\_EXTICR)**

# Глава 8 Управление прямым доступом к памяти (DMA)

Контроллер прямого доступа к памяти (DMA) обеспечивает высокоскоростной метод передачи данных между периферийными устройствами и памятью или между памятью и памятью без вмешательства процессора, а данные могут быть быстро перемещены с помощью DMA для экономии ресурсов процессора для других операций. Каждый канал контроллера DMA предназначен для управления запросами на доступ к памяти от одного или нескольких периферийных устройств. Также существует арбитр для координации приоритетов между каналами.

## 8.1 Основные характеристики

* Несколько независимых настраиваемых каналов
* Каждый канал напрямую подключен к выделенному аппаратному запросу DMA и поддерживает программный триггер
* Управление буфером с поддержкой циклов
* Приоритет запросов между несколькими каналами может быть установлен программированием (очень высокий, высокий, средний и низкий), а при равенстве приоритет определяется номером канала (чем ниже номер канала, тем выше приоритет)
* Поддерживает передачу данных "периферия-память", "память-периферия" и "память-память"
* В качестве источников и целей доступа можно использовать флеш-память, SRAM, периферийный SRAM и периферии HB
* Программируемое количество байт для передачи данных: до 65 535

## 8.2 Описание функций

### 8.2.1 Обработка каналов DMA

**1) Приоритеты арбитража**

Запросы DMA, поступающие от нескольких независимых каналов, поступают в контроллер DMA через логическую схему типа "ИЛИ", и в любой момент времени обслуживается только один из них. Арбитр внутри модуля выбирает, какой доступ к периферии или памяти следует инициировать, исходя из приоритета поступившего запроса.

С помощью программного управления приложение может настроить уровень приоритета для каждого канала индивидуально, установив соответствующие значения бит **PL**[1:0] в регистре конфигурации DMA (**DMA\_CFGRx**). Доступны четыре уровня приоритета: самый высокий, высокий, средний и низкий. Если уровни приоритета, установленные программно, совпадают у нескольких каналов, то выбор между ними производится на основании фиксированного аппаратного приоритета: чем меньше номер канала, тем выше его приоритет по сравнению с другими каналами.

**2) Конфигурация DMA**

Когда контроллер DMA получает сигнал запроса, он обращается к запрашиваемой периферии или памяти и устанавливает передачу данных между периферией или памятью и памятью. Он состоит из следующих трех основных этапов работы.

#### Начало формы

1. Извлечение данных из адреса памяти, указанного регистром данных периферии или текущим регистром адреса периферии/памяти. Начальный адрес для первой передачи – это базовый адрес периферии или адрес памяти, указанный в регистрах **DMA\_PADDRx** или **DMA\_MADDRx**.
2. Сохранение данных по адресу памяти, указанному регистром данных периферии или текущим регистром адреса периферии/памяти. Начальным адресом для первой передачи является базовый адрес периферии или адрес памяти, указанные в регистрах **DMA\_PADDRx** или **DMA\_MADDRx**.
3. Выполнение операции декремента значения в регистре **DMA\_CNTRx**, которое указывает количество текущих ожидающих выполнения передач.

Каждый канал включает в себя 3 типа методов передачи данных DMA.

* Передача данных с периферии в память (**MEM2MEM** = 0, **DIR** = 0)
* Передача данных из памяти на периферию (**MEM2MEM** = 0, **DIR** = 1)
* Передача данных между областями памяти (**MEM2MEM** = 1)

Начало формы

*Примечание: Режим передачи данных между областями памяти не требует сигнала запроса от периферии. После настройки этого режима (****MEM2MEM*** *= 1) включается канал (****EN*** *= 1), чтобы начать передачу данных. Этот режим не поддерживает циклический режим.*

Процесс конфигурации выглядит следующим образом:Начало формы

1. Установите первый адрес регистра периферии или адрес данных памяти в режиме передачи данных между областями памяти (**MEM2MEM** = 1) в регистре **DMA\_PADDRx**. Этот адрес станет исходным или целевым адресом для передачи данных при возникновении запроса DMA.
2. Установите адрес данных памяти в регистре **DMA\_MADDRx**. При возникновении запроса DMA переданные данные будут считаны из этого адреса или записаны в него.
3. Установите объем данных, которые должны быть переданы, в регистре **DMA\_CNTRx**. Это значение уменьшается после каждой передачи данных.
4. Установите приоритет канала в битах **PL**[1:0] регистра **DMA\_CFGRx**.
5. Установите направление передачи данных, циклический режим, инкрементный режим для периферии и памяти, ширину данных для периферии и памяти, прерывание на половину передачи, завершение передачи и ошибку передачи в регистре **DMA\_CFGRx**.
6. Установите бит **ENABLE** в регистре **DMA\_CCRx** для запуска канала x.

*Примечание: Регистры* ***DMA\_PADDRx****/****DMA\_MADDRx****/****DMA\_CNTRx*** *и биты управления направлением передачи данных (DIR), циклическим режимом (локацией) и инкрементным режимом для периферии и памяти (MINC/PINC) в регистре* ***DMA\_CFGRx*** *могут быть настроены только тогда, когда канал DMA выключен.*

#### 3) Циклический режим

Установка бита **CIRC** в 1 в регистре **DMA\_CFGRx** активирует функцию циклического режима для передачи данных по каналу. В циклическом режиме, когда количество передач данных становится равным нулю, содержимое регистра **DMA\_CNTRx** автоматически перезагружается до своего первоначального значения, а внутренние регистры адреса периферии и памяти перезагружаются до начальных значений адресов, установленных регистрами **DMA\_PADDRx** и **DMA\_MADDRx**. Операция DMA будет продолжаться до тех пор, пока канал не будет отключен или режим DMA не будет деактивирован.

#### 4) Статус обработки DMA

#### Начало формы

* **Передача половины данных**: Соответствует установке бита **HTIFx** в регистре **DMA\_INTFR**. Флаг, указывающий на передачу половины байтов DMA, будет сгенерирован, когда количество передач DMA уменьшится менее чем наполовину от начального установленного значения, и если установлен бит **HTIE** в регистре **DMA\_CCRx**, произойдет прерывание. Аппаратное обеспечение использует этот флаг, чтобы предупредить приложение о том, что оно может подготовиться к новому циклу передачи данных.
* **Завершение передачи**: Соответствует настройке бита **TCIFx** в регистре **DMA\_INTFR**. Когда количество передаваемых байтов DMA уменьшается до нуля, генерируется флаг завершения передачи DMA, и, если установлен бит **TCIE** в регистре **DMA\_CCRx**, происходит прерывание.
* **Ошибка передачи**: Соответствует установленной аппаратуре биту **TEIFx** в регистре **DMA\_INTFR**. Чтение или запись в зарезервированную область адреса приведет к возникновению ошибки передачи DMA. Одновременно аппаратное обеспечение модуля автоматически сбросит бит **EN** в регистре **DMA\_CCRx** соответствующего канала, в котором произошла ошибка, и канал будет отключен. Если установлен бит **TEIE** в регистре **DMA\_CCRx**, также произойдет прерывание.

Когда приложение запрашивает статус канала DMA, оно сначала может получить доступ к биту **GIFx** регистра DMA\_INTFR, чтобы определить, какой канал в настоящее время испытывает событие DMA, а затем обработать конкретное содержание события DMA для этого канала.

### 8.2.2 Программируемый общий размер передаваемых данных/ширина битов данных/выравнивание

Общий размер данных, передаваемых за один цикл работы канала DMA, программируется до 65535 раз, и количество оставшихся байтов для передачи указывается в регистре **DMA\_CNTRx**. При **EN**=0 записывается установленное значение, а при **EN**=1, когда канал передачи DMA включен, этот регистр становится атрибутом только для чтения со значением, уменьшающимся после каждой передачи.

Значение данных, извлекаемое из периферии и памяти, поддерживает функцию автоматического увеличения указателя адреса с программируемым увеличением указателя. Первый адрес передаваемых данных, к которому они обращаются, хранится в регистрах **DMA\_PADDRx** и **DMA\_MADDRx**. Путём установки бита **PINC** или положения **MINC** равным 1 в регистре **DMA\_CFGRx** соответственно можно включить режим автоувеличения адреса периферии или автоувеличения адреса памяти. **PSIZE**[1:0] задаёт размер данных и размер автоувеличения для извлечения адреса периферии. **MSIZE**[1:0] задает размер данных для извлечения из памяти и размер её автоувеличения, включая три варианта: 8-битные, 16-битные и 32-битные. Конкретные методы передачи данных перечислены в следующей таблице.

Таблица 8-1 Передача DMA с различной шириной битов данных (**PINC**=**MINC**=1)

### 8.2.3 Сопоставление запросов DMA (DMA Request Mapping)

### Контроллер DMA предоставляет семь каналов, каждый из которых соответствует нескольким запросам периферийных устройств. Включать или отключать функцию DMA для каждого периферийного устройства можно независимо, устанавливая соответствующие биты управления DMA в соответствующих регистрах периферийного оборудования. Вот конкретные соответствия.

Рисунок 8-1 DMA1 марта запросов

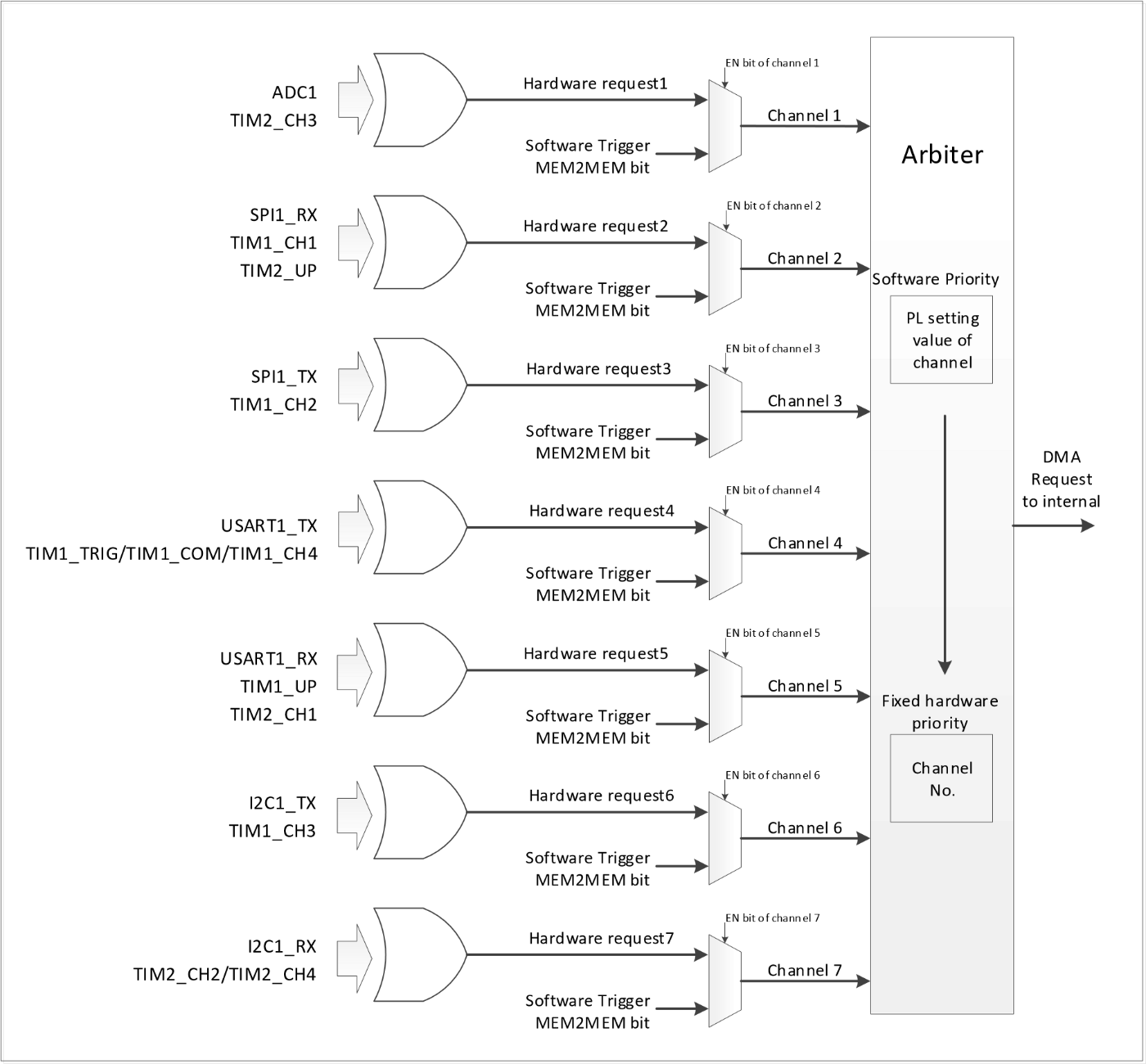


Таблица 8-2 Таблица сопоставления периферийных устройств DMA1 для каждого канала

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Переферия | Канал 1 | Канал 2 | Канал 3 | Канал 4 | Канал 5 | Канал 6 | Канал 7 |
| ADC1 | ADC1 |  |  |  |  |  |  |
| SPI1 |  | SPI1\_RX | SPI1\_TX |  |  |  |  |
| USART1 |  |  |  | USART1\_TX | USART1\_RX |  |  |
| I2C1 |  |  |  |  |  | I2C1\_TX | I2C1\_RX |
| TIM1 |  | TIM1\_CH1 | TIM1\_CH2 | TIM1\_CH4 TIM1\_TRIG TIM1\_COM | TIM1\_UP | TIM1\_CH3 |  |
| TiIM2 | TIM2\_CH3 | TIM2\_UP |  |  | TIM2\_CH1 |  | TIM2\_CH2 TIM2\_CH4 |

## 8.3 Описание регистров

Таблица 8-3 Список регистров, связанных с DMA

### 8.3.1 Регистр состояния прерывания DMA (DMA\_INTFR)

### 8.3.2 DMA Interrupt Flag Clear Register (DMA\_INTFCR)

### 8.3.3 DMA Channel x Configuration Register (DMA\_CFGRx) (x=1/2/3/4/5/6/7)

### 8.3.4 DMA Channel x Number of Data Register (DMA\_CNTRx) (x=1/2/3/4/5/6/7)

### 8.3.5 DMA Channel x Peripheral Address Register (DMA\_PADDRx) (x=1/2/3/4/5/6/7)

*Note: This register can only be changed when EN=0 and cannot be written when EN=1.*

### 8.3.6 DMA Channel x Memory Address Register (DMA\_MADDRx) (x=1/2/3/4/5/6/7)

*Note: This register can only be changed when EN=0 and cannot be written when EN=1.*

# Глава 9 Аналого-цифровой преобразователь (АЦП)

# Начало формы

Модуль АЦП содержит 10-битный аналого-цифровой преобразователь последовательных приближений с входной тактовой частотой до 24 МГц. Он поддерживает выборку до 8 внешних каналов и 2 внутренних источников сигналов. Можно выполнять одиночное преобразование и непрерывное преобразование каналов, автоматическое сканирование между каналами, прерывистый режим, режим внешнего запуска, двойную выборку, задержку запуска и многое другое. С помощью функции аналогового сторожевого таймера можно контролировать напряжение на канале, чтобы убедиться, что оно находится в пределах порогового диапазона.

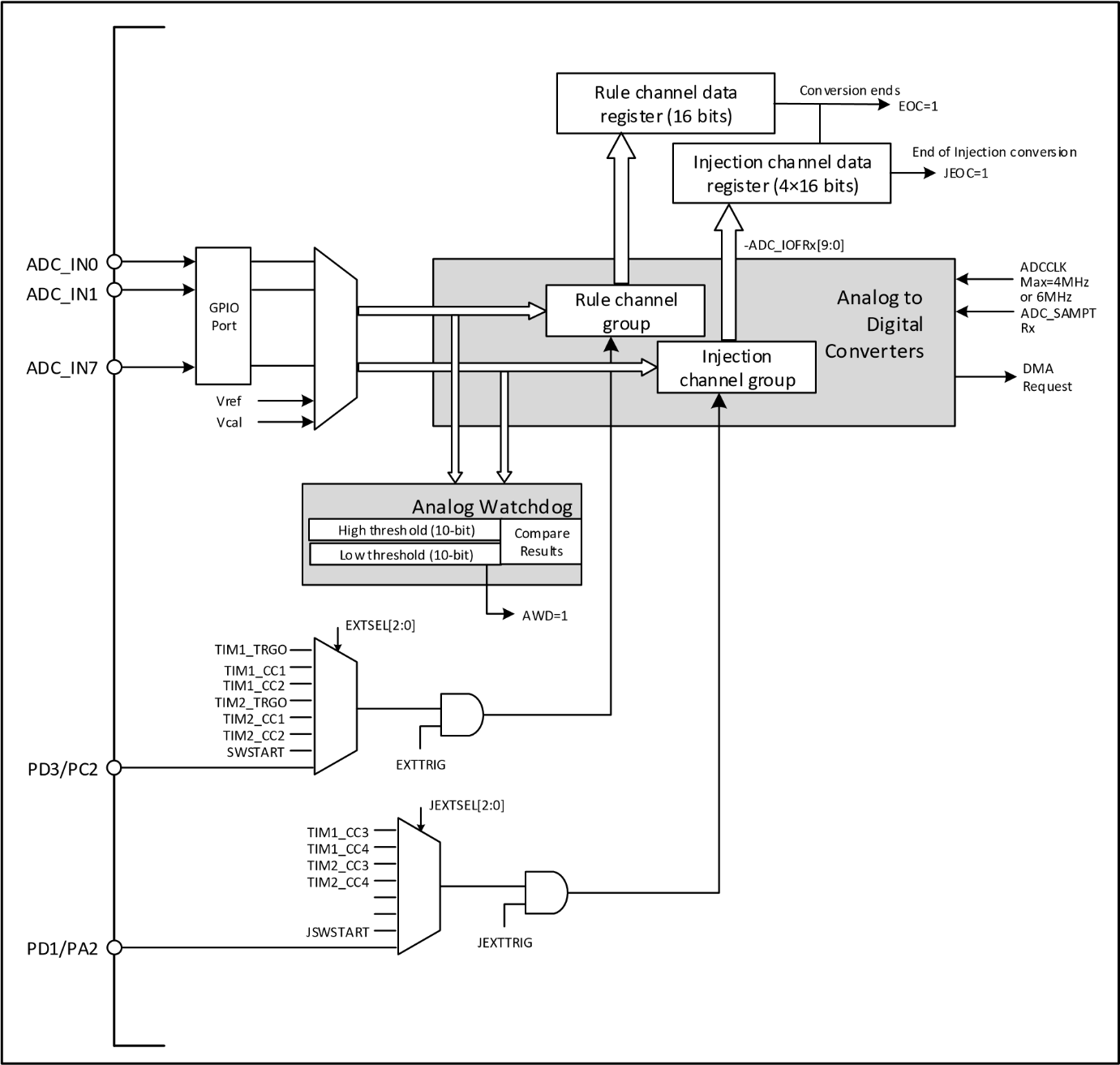
## 9.1 Основные возможности

* Разрешение 10 бит
* Поддержка 8 внешних каналов и 2 внутренних источников сигналов для выборки
* Многочисленные методы преобразования выборок для нескольких каналов: однократная, непрерывная, сканирующая, запускаемая, прерывистая и т.д.
* Режимы выравнивания данных: выровненные слева, выровненные справа
* Время выборки может быть запрограммировано отдельно для каждого канала
* Как обычное, так и инъекционное преобразование поддерживают внешний запуск
* Аналоговый сторожевой таймер для мониторинга напряжения на канале, функция самокалибровки
* Диапазон входного сигнала канала АЦП: 0 ≤ VIN ≤ VDDA
* Задержка запуска

## 9.2 Описание функциональности

### 9.2.1 Структура модуля

Рисунок 9-1 Блок-схема модуля АЦП



### 9.2.2 Конфигурирование ADC

#### 1) Включение модуля

Бит ADON, установленный в 1 в регистре **ADC\_CTLR2**, указывает, что модуль АЦП включен. Когда модуль АЦП переходит в состояние включения (**ADON**=1) из режима пониженного энергопотребления (**ADON**=0), требуется период задержки tSTAB для стабилизации модуля. После этого бит **ADON** снова устанавливается в 1 и используется в качестве сигнала начала для старта преобразования АЦП программным обеспечением. Сбросив бит **ADON**, можно завершить текущее преобразование и перевести модуль АЦП в режим пониженного энергопотребления, в котором АЦП практически не потребляет энергию.

#### 2) Тактирование

Работа регистра модуля основана на тактовой частоте **HBCLK** (шина HB), а опорная частота его блока преобразования, **ADCCLK**, конфигурируется полем **ADCPRE** регистра **RCC\_CFGR0** для деления частоты. За подробной информацией обратитесь к техническому описанию CH32V003DS0.

#### 3) Конфигурирование каналов

Модуль АЦП предоставляет 10 источников выборки каналов, включая 8 внешних каналов и 2 внутренних канала. Они могут быть сконфигурированы в две группы преобразования: обычные группы и группы инъекции. Это позволяет достичь группового преобразования, состоящего из серии преобразований в любом порядке на любом количестве каналов.

* Обычная группа: состоит максимум из 16 преобразований. Каналы обычной группы и порядок их преобразования устанавливаются в регистре **ADC\_RSQRx**. Общее количество преобразований в обычной группе должно быть записано в **L**[3:0] регистра **ADC\_RSQR1**.
* Группа инъекции: состоит максимально из 4 преобразований. Каналы группы инъекции и порядок их преобразований устанавливаются в регистре **ADC\_ISQR**. Общее число преобразований в группе инъекции должно быть указано в **JL**[1:0] регистра **ADC\_ISQR**.

*Примечание: Если во время преобразования изменяются регистры* ***ADC\_RSQRx*** *или* ***ADC\_ISQR****, текущее преобразование прекращается, и новый стартовый сигнал отправляется в АЦП для преобразования вновь выбранной группы.*

2 внутренних канала.

* **Внутреннее опорное напряжение Vref**: подключено к каналу ADC\_IN8.
* **Напряжение внутренней калибровки Vcal**: подключено к каналу ADC\_IN9, выбираемое двумя шагами.
* Начало формы

#### 4) Калибровка

АЦП имеет встроенный режим самотестирования. Сеанс калибровки значительно снижает погрешность точности, вызванную изменениями в банках внутренних конденсаторов. Во время калибровки код коррекции ошибок рассчитывается для каждого конденсатора, который используется для устранения ошибок, возникающих на каждом конденсаторе в последующих преобразованиях. Инициализируйте регистр калибровки, установив положение **RSTCAL** в регистре **ADC\_CTLR2** в 1, и дождитесь, пока аппаратное обеспечение **RSTCAL** очистит 0, что указывает на завершение инициализации. Установите бит **CAL**, чтобы запустить функцию калибровки. После завершения калибровки аппаратное обеспечение автоматически очистит бит **CAL** и сохранит код калибровки в **ADC\_RDATAR**. После этого можно начинать нормальную функцию преобразования. Рекомендуется проводить калибровку АЦП при включении модуля АЦП.

*Примечание: Прежде чем приступить к калибровке, необходимо убедиться, что модуль АЦП находится во включенном состоянии (****ADON****=1) не менее двух полных циклов тактового сигнала АЦП.*

#### 5) Программируемое время выборки

ЦП использует несколько циклов **ADCCLK** для выборки входного напряжения. Количество циклов выборки для канала можно изменить, используя биты **SMPx**[2:0] в регистрах **ADC\_SAMPTR1** и **ADC\_SAMPTR2**.

Каждому каналу можно присвоить отдельное время выборки. Общее время преобразования вычисляется следующим образом.Начало формы

TCONV = время выборки + 11TADCCLK

Преобразование каналов по правилам АЦП поддерживает функцию DMA. Значение преобразования канала по правилам сохраняется в регистре данных **ADC\_RDATAR**. Чтобы предотвратить несвоевременное получение данных из регистра **ADC\_RDATAR** при последовательном преобразовании нескольких каналов по правилу, можно включить функцию DMA АЦП. Аппаратное обеспечение будет генерировать запрос DMA в конце преобразования канала по правилу (установлен **EOC**) и передавать преобразованные данные из регистра **ADC\_RDATAR** по заданному пользователем адресу назначения. После завершения настройки канала модуля контроллера DMA запишите позицию DMA в регистре **ADC\_CTLR2** в 1, чтобы включить функцию DMA для АЦП.

*Примечание: Преобразование группы инъекции не поддерживает функцию DMA.*

#### 6) Выравнивание данныхНачало формы

Бит **ALIGN** в регистре **ADC\_CTLR2** выбирает способ выравнивания хранения данных, преобразованных АЦП. Для 10-битных данных поддерживаются режимы выравнивания по левому и правому краям.

Регистр данных **ADC\_RDATAR** канала обычной группы хранит фактическое преобразованное 10-битное цифровое значение; в то время как регистр данных **ADC\_IDATARx** канала группы инъекции представляет собой фактически преобразованные данные минус значение, записанное после смещения, определенного в регистре **ADC\_IOFRx**. Могут возникать положительные и отрицательные случаи, поэтому имеются биты знака (**SIGNB**).

Таблица 9-1 Выравнивание данных по левому краю

Начало формы

Регистр данных обычной группы

D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 0 0 0 0 0 0

Регистр данных группы инъекции

Начало формы

SIGNB D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 0 0 0 0 0

Таблица 9-2 Выравнивание данных по правому краю

Регистр данных обычной группы

0 0 0 0 0 0 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0

Регистр данных группы инъекции

SIGNB SIGNB SIGNB SIGNB SIGNB SIGNB D9 D8 D7 D6 D5 D4 D3 D2 D1 D0

### 9.2.3 Источник внешнего запуска

Начало преобразования АЦП может быть инициировано внешним событием. Если установлены биты **EXTTRIG** или **JEXTTRIG** в регистре **ADC\_CTLR2**, преобразование канала обычной группы или группы инъекции может быть запущено внешним событием соответственно. В этом случае конфигурация битов **EXTSEL**[2:0] и **JEXTSEL**[2:0] определяет источник внешнего события для обычной группы и группы инъекции.

*Примечание: Когда для преобразования АЦП по правилу или инъекции выбран внешний сигнал запуска, только его передний фронт может начать преобразование.*

Таблица 9-3 Источники внешнего запуска для каналов обычной группы

### 9.2.4 Режим преобразования

Таблица 9-5 Комбинации режимов преобразования

Начало формы

*Примечание: Внешние события запуска для обычных групп и групп инъекции различаются, и бит '****ACON****' может инициировать только преобразование канала обычной группы, поэтому события инициации для преобразования каналов обычной группы и группы инъекции являются независимыми.*

#### 1) Единичный одноканальный режим преобразования

В этом режиме выполняется только одно преобразование для текущего канала. Этот режим выполняет преобразование канала, который отсортирован первым в обычной группе или группе инъекции, где он инициируется установкой бита **ADON** в положении 1 регистра **ADC\_CTLR2** (только для каналов обычной группы) или может быть инициирован внешним запуском (для каналов обычной группы или каналов инъекции). После завершения преобразования выбранного канала оно завершится. Если преобразование выполняется для канала обычной группы, данные преобразования сохраняются в 16-битном регистре **ADC\_RDATAR**, устанавливается флаг **EOC**, и вызывается прерывание АЦП, если установлен бит **EOCIE**. Если преобразование выполняется для канала группы инъекции, данные преобразования сохраняются в 16-битном регистре **ADC\_IDATAR1**, устанавливаются флаги **EOC** и **JEOC**, и возникает прерывание АЦП, если установлены биты **JEOCIE** или **EOCIE**.

#### 2) Преобразование в режиме однократного сканирования

Режим сканирования АЦП включается путем установки бита **SCAN** в регистре **ADC\_CTLR1** в 1. Этот режим используется для сканирования группы аналоговых каналов и выполнения однократного преобразования для всех каналов, выбранных регистром **ADC\_RSQRx** (для обычных каналов) или **ADC\_ISQR** (для каналов инъекции), по одному, и следующий канал в той же группе преобразуется автоматически, когда завершается преобразование текущего канала. В режиме сканирования существует разделение на режимы запуска инъекции и автоматической инъекции в зависимости от состояния бита **JAUTO**.

* **Запуск инъекции**

Бит **JAUTO** равен 0. Когда во время сканирования каналов обычной группы происходит событие запуска преобразования канала группы инъекции, текущее преобразование сбрасывается, и последовательность каналов инъекции выполняется в одном сканировании, и прерванное преобразование последнего канала обычной группы возобновляется после завершения сканирующего преобразования всех выбранных каналов группы инъекции.

Если во время текущего сканирования последовательности каналов группы инъекции происходит событие начала преобразования канала обычной группы, преобразование группы инъекции не прерывается, но последовательность преобразований правила выполняется повторно после завершения последовательности преобразований инъекции.

*Примечание: При использовании преобразований с запуском инъекции необходимо обеспечить, чтобы интервал между событиями запуска был больше, чем сама последовательность инъекции. Например, если общее время, необходимое для завершения преобразования последовательности инъекции, составляет 28* ***ADCCLK****, то минимальное значение интервала событий для запуска канала инъекции равно 29* ***ADCCLK****.*

* **Автоматическая инъекция**

Бит **JAUTO** установлен в 1, и преобразование выбранного канала группы инъекции выполняется автоматически после сканирования всех каналов, выбранных для преобразования группой правил. Этот метод позволяет преобразовать до 20 последовательностей преобразований в регистрах **ADC\_RSQRx** и **ADC\_ISQR**. В этом режиме внешняя активация канала инъекции должна быть отключена (**IEXTTRIG JEXTTRIG**=0).

*Примечание: Для коэффициента предварительного масштабирования тактовой частоты АЦП (****ADCPRE****[1:0]), равного 4–8, автоматически вставляется интервал 1* ***ADCCLK*** *при переключении с обычного преобразования на последовательность инъекции или с преобразования инъекции на последовательность правил; когда коэффициент предварительного масштабирования тактовой частоты составляет 2, вводится задержка в два интервала* ***ADCCLK****.*

#### 3) Преобразование в режиме единичного прерывистого режима

Режимом прерывистой работы для обычной группы или группы инъекции управляет установка бита **RDISCEN** или **IDISCEN** в регистре **ADC\_CTLR1** в 1. Этот режим отличается от сканирования полного набора каналов в режиме сканирования, но делит набор каналов на несколько коротких последовательностей, и каждое внешнее событие запуска выполнит короткую последовательность переходов сканирования. Длина короткой последовательности n (n <= 8) определяется в **DISCNUM**[2:0] регистра **ADC\_CTLR1**, когда **RDISCEN** равно 1, это означает прерывающийся режим работы для группы правил, а общая длина для преобразования определяется в **L**[3:0]. в регистре **ADC\_RSQR1**; когда **IDISCEN** установлено на 1, это прерывающаяся работа для группы инъекции, и общая длина для конвертации определяется в **JL**[1:0] в регистре **ADC\_ISQR**. Невозможно установить оба режима для обычной и группы инъекций в прерывистом режиме.

Пример прерывистого режима работы для обычной группы.

**RDISCEN**=1, **DISCNUM**[2:0]=3, **L**[3:0]=8, каналы для преобразования = 1, 3, 2, 5, 8, 2, 7

Первый внешний триггер: последовательность преобразований: 1, 3, 2

Второй внешний триггер: последовательность преобразований: 5, 8, 4

Третий внешний триггер: последовательность преобразований: 10, 6, при генерации событий **EOC**

Четвертый внешний триггер: последовательность преобразований: 1, 3, 2

**IDISCEN**=1, **DISCNUM**[2:0]=1, **JL**[1:0]=3, каналы для преобразования: 1, 3, 2

Первый внешний триггер: последовательность преобразований: 1

Второй внешний триггер: последовательность преобразований: 3

Третий внешний триггер: преобразуются каналы: 2, генерируются события **EOC** и **JEOC**

Четвертый внешний триггер: начинается новая последовательность преобразований: 1

*Примечание:*

1. *При преобразовании обычной группы или группы инъекции в прерывистом режиме последовательность преобразований не начинает автоматически начинаться с самого начала после завершения. Когда все подгруппы были преобразованы, следующее событие триггера запускает преобразование первой подгруппы.*
2. *Вы не можете одновременно использовать автоинъекцию (****JAUTO****=1) и прерывистый режим.*
3. *Вы не сможете настроить прерываемый режим как для обычных групп, так и для групп инъекции, и вы можете использовать прерывистую работу только для одной группы преобразований.*

#### 4) Непрерывное преобразование

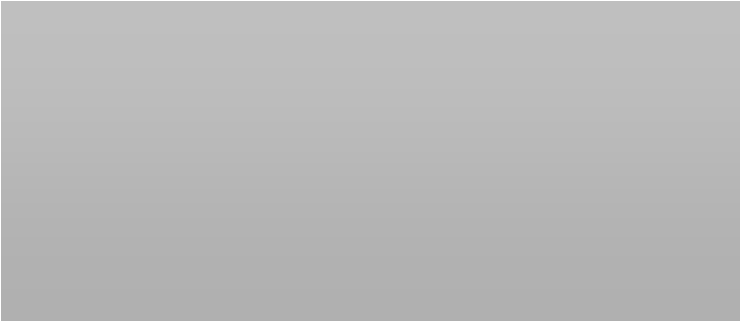
В режиме непрерывного преобразования новое преобразование начинается сразу после завершения предыдущего преобразования. Преобразование не останавливается на последнем канале выбранной группы, а продолжается с первого канала выбранной группы. Запускающими событиями в этом режиме являются внешние триггерные события, а бит **ADON** устанавливается в 1. После настройки запуска бит **CONT** должен быть установлен в 1.

Если производится преобразование обычного канала, данные преобразования хранятся в регистре **ADC\_RDATAR**, устанавливается флаг окончания преобразования **EOC**, и если установлен бит **EOCIE**, генерируется прерывание. При преобразовании канала инжекции данные преобразования сохраняются в регистре **ADC\_IDATARx**, устанавливается флажок завершения преобразования **JEOC**, а если установлен флаг **JEOCIE**, создается прерывание.

### 9.2.5 Сторожевой таймер АЦП

Флаг состояния аналогового сторожа (AWD) устанавливается, если аналоговое напряжение, которое преобразуется АЦП, ниже низкого порога или выше высокого порога. Настройки порогов находятся в десяти младших значащих разрядах регистров ADC\_WDHTR и ADC\_WDLTR. Бит AWDIE в регистре ADC\_CTLR1 устанавливается для генерации соответствующего прерывания.

Рисунок 9-4 Область порога сторожевого таймера АЦП



ADC\_WDHTR

ADC\_WDLTR

Нижний предупредительный порог

Верхний предупредительный порог

Зона предупреждения

Значения преобразования аналогового напряжения

Настройте биты **AWDSGL**, **AWDEN**, **JAWDEN** и **AWDCH**[4:0] в регистре **ADC\_CTLR1** для выбора канала для сигнализации аналогового сторожа, как показано в следующей таблице.

Таблица 9-6 Выбор канала аналогового сторожевого таймера

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имитация канала оповещения сторожа | Биты управления регистром ADC\_CTLR1 | | | |
| AWDSGL | AWDEN | JAWDEN | AWDCH[4:0] |
| Отсутствие реакции | Ignore | 0 | 0 | Ignore |
| Все инжектированные каналы | 0 | 0 | 1 | Ignore |
| Все обычные каналы | 0 | 1 | 0 | Ignore |
| Все обычные и инжектированные каналы | 0 | 1 | 1 | Ignore |
| Один инжектированый канал | 1 | 0 | 1 | Зависит от номера канала |
| Один обычный канал | 1 | 1 | 0 | Зависит от номера канала |
| Один инжектированный и обычный канал | 1 | 1 | 1 | Зависит от номера канала |

## 9.3 Описание регистров

Таблица 9-7 Список регистров, связанных с АЦП

### 9.3.1 ADC Status Register (ADC\_STATR)

### 9.3.2 ADC Control Register 1 (ADC\_CTLR1)

### 9.3.3 ADC Control Register 2 (ADC\_CTLR2)

*Note: A conversion is initiated when only ADON is changed in the register, and no new conversion is initiated*

### 9.3.4 ADC Sample Time Configuration Register 1 (ADC\_SAMPTR1)

### 9.3.5 ADC Sample Time Configuration Register 2 (ADC\_SAMPTR2)

### 9.3.6 ADC Injected Channel Data Offset Register x (ADC\_IOFRx) (x=1/2/3/4)

### 9.3.7 ADC Watchdog High Threshold Register (ADC\_WDHTR)

*Note: You can change the values of WDHTR and WDLTR during the conversion process, but they will take effect at the next conversion.*

### 9.3.8 ADC Watchdog Low Threshold Register (ADC\_WDLTR)

*Note: You can change the values of WDHTR and WDLTR during the conversion process, but they will take effect at the next conversion.*

### 9.3.9 ADC Regular Sequence Register 1(ADC\_RSQR1)

### 9.3.10 ADC Regular Sequence Register 2(ADC\_RSQR2)

### 9.3.11 ADC Regular Sequence Register 3(ADC\_RSQR3)

### 9.3.12 ADC Injected Sequence Register (ADC\_ISQR)

*Примечание: В отличие от обычной последовательности преобразований, если длина* ***JL****[1:0] меньше 4, порядок преобразования начинается с (4 -* ***JL****).*

*Например, когда* ***JL****[1:0]=3 (4 инжектированных перехода в секвенсере), АЦП будет преобразовывать каналы в следующем порядке:* ***JSQ1****[4:0],* ***JSQ2****[4:0],* ***JSQ3****[4:0] и* ***JSQ4****[4:0].*

*Когда* ***JL****[1:0]=2 (3 инжектированных перехода в секвенсере), АЦП будет преобразовывать каналы в следующем порядке:* ***JSQ2****[4:0],* ***JSQ3****[4:0] и* ***JSQ4****[4:0]; Когда* ***JL****[1:0]==1 (два инжектированных перехода в секвенсер), АЦП сначала преобразует* ***JSQ3****[4:0], а затем* ***JSQ4****[4:0].*

*Когда* ***JL****[1:0] == 0 (одно инжектированное преобразование ввода в секвенсере), АЦП будет преобразовывать только каналы* ***JSQ4****[4:0]. Если* ***ADCx\_ISQR****[21:0]=10 00111 00011 00111 00010, АЦП преобразует каналы в следующем порядке:* ***JSQ2****[4:0],* ***JSQ3****[4:0], и* ***JSQ4****[4:0], указывая, что сканирующие преобразования выполняются в следующем порядке каналов: 7, 3, 7.*

### 9.3.13 ADC Injected Data Register (ADC\_IDATARx) (x=1/2/3/4)

### 9.3.14 ADC Regular Data Register (ADC\_RDATAR)

### 9.3.15 ADC Delayed Data Register (ADC\_DLYR)

# Глава 10. Таймер управления с расширенными функциями (ADTM)

Модуль таймера управления с расширенными функциями содержит мощный 16-битный таймер с авто перезагрузкой, TIM1, который может использоваться для измерения ширины импульса или генерации импульсов, ШИМ-сигналов и т.д. Он применяется в управлении двигателями, источниках питания и других приложениях.

## 10.1 Основные возможности

Основные особенности таймера управления с расширенными функциями TIM1 включают:.

* 16-битный счетчик с автоперезагрузкой, поддерживающий режимы инкрементального счета, декрементального счёта и комбинированного инкрементально-декрементального режима.
* 16-битное предварительное деление с динамически настраиваемыми коэффициентами деления от 1 до 65536.
* Поддержка четырех независимых каналов сравнения/захвата.
* Каждый канал сравнения/захвата поддерживает несколько режимов работы, таких как захват входных сигналов, сравнение выходных сигналов, генерация ШИМ-сигнала и однократная генерация импульса.
* Комплементарные выходы с возможностью программирования мертвого времени.
* Возможность управления таймером внешними сигналами.
* Обновление состояния таймера через заданный интервал времени с использованием счетчика повторений.
* Перезапуск таймера или установка его в состояние "OK" с помощью сигнала торможения.
* Поддержка использования DMA в нескольких режимах.
* Работа с инкрементальными энкодерами.
* Каскадирование и синхронизация между несколькими таймерами.

## 10.2 Принципы и структура

Этот раздел посвящен внутреннему устройству таймеров управления с расширенными функциями..

### 10.2.1 Обзор

Как показано на Рисунке 10-1, структура таймера управления с расширенными функциями может быть условно разделена на три части: блок входного тактового сигнала, ядро счетчика и часть канала сравнения/захвата.

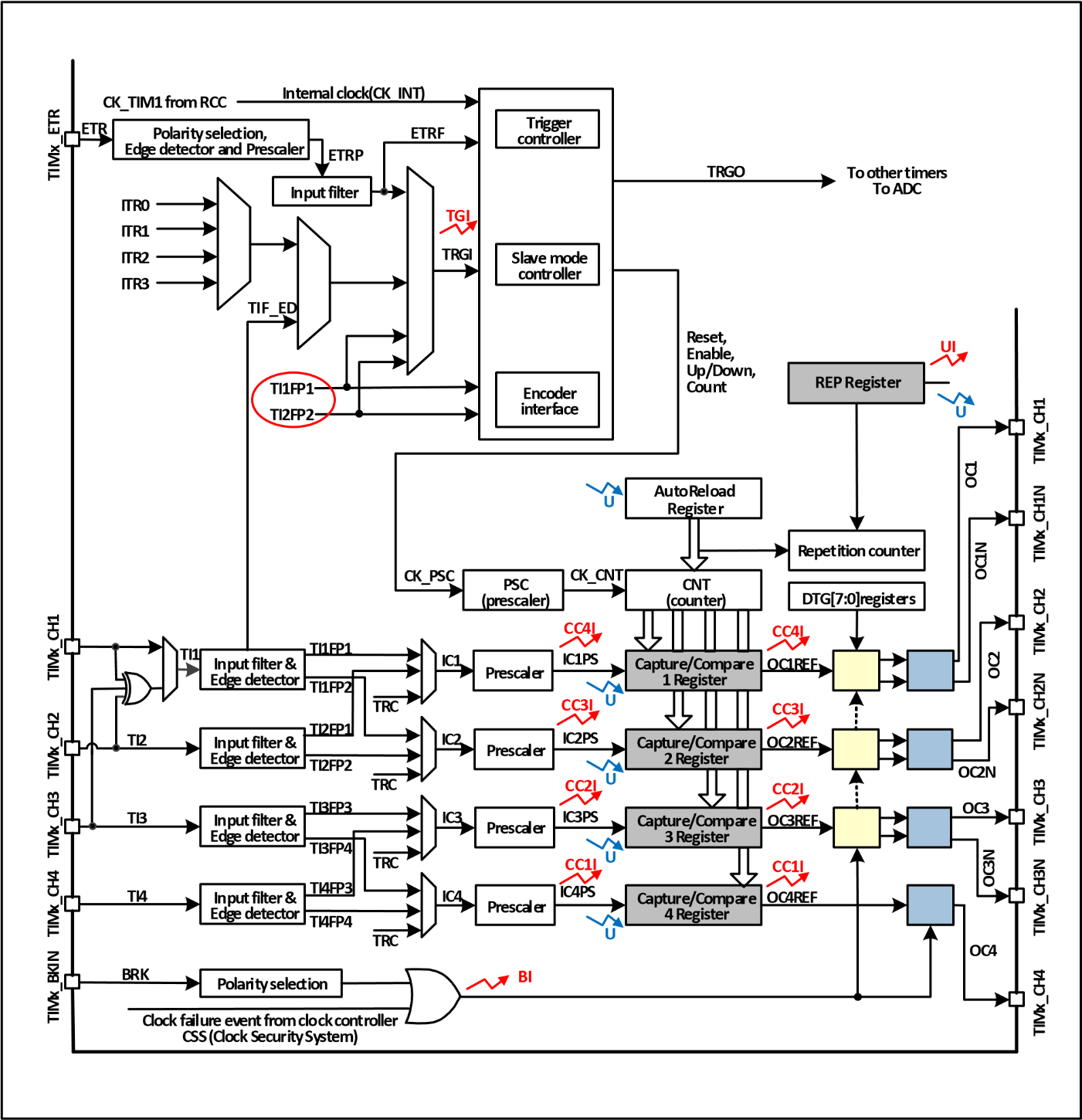
Таймер управления с расширенными возможностями может получать тактовый сигнал от шины HB (**CK\_INT**), внешнего входа тактового сигнала (**TIMx\_ETR**), других таймеров с выходом тактового сигнала (**ITRx**) или входа канала сравнения/захвата (**TIMx\_CHx**). Эти сигналы после различных операций фильтрации и деления преобразуются в сигнал **CK\_PSC**, который подается на секцию ядра счетчика. Кроме того, эти сложные источники тактовых сигналов могут также выводиться как **TRGO** для других периферийных устройств, таких как таймеры и АЦП.

Центральным элементом таймера является 16-битный счетчик (**CNT**), на который поступает сигнал **CK\_CNT**, полученный путем деления **CK\_PSC** с помощью предделителя (**PSC**). Вспомогательный счетчик подсчитывает количество раз, когда регистр **ATRLR** загружает начальное значение в **CNT**, и генерирует определенное событие, когда достигнуто число, установленное в регистре повтора (**RPTCR**).

У таймера имеется четыре набора каналов сравнения/захвата, каждый из которых может принимать импульсы с отдельных выводов или выводить сигналы на выводы, то есть каналы сравнения/захвата поддерживают как входной, так и выходной режимы. Вход каждого канала регистра захвата поддерживает операции фильтрации, деления и обнаружения фронтов, поддерживает взаимную активацию между каналами, а также обеспечивает тактовый сигнал для основного счетчика **CNT**. Каждый канал сравнения/захвата имеет набор регистров сравнения/захвата (**CHxCVR**), которые сравнивают значения с главным счетчиком (**CNT**) для формирования импульсных сигналов.

.

Рисунок 10-1. Блок-схема структуры таймера управления с расширенными функциями.



### 10.2.2 Clock Input

Рисунок 10-2. Блок-схема источника сигнала **CK\_PSC** для таймера управления с расширенными функциями.

**Encoder**

**mode**

**External clock**

**mode 1**

**External clock**

**mode 2**

**Internal clock**

**mode**

**CK\_PSC**

**TI2F**

**TI1F**

**or**

**or**

**TRGI**

**ETRF**

**CK\_INT**

**or**

**)**

**(**

**internal clock**

**TS[2:0]**

**TIMx\_SMCR**

**xx**

**0**

**100**

**101**

**110**

**111**

**ITRx**

**TI1\_ED**

**TI1FP1**

**TI2FP2**

**ETRF**

**0**

**1**

**CC2P**

**TIMx\_CCER**

**Edge**

**detector**

**TI2F\_Rising**

**TI2F\_Falling**

**Filter**

**ICF[3:0]**

**TIMx\_CCMR1**

**TI2**

**ETR pin**

**ETP**

**TIMx\_SMCR**

**0**

**1**

**ETR**

**Divider**

**/1,/2,/4,/8**

**ETPS[1:0]**

**TIMx\_SMCR**

**ETF[3:0]**

**TIMx\_SMCR**

**ETRP**

**f**

**DTS**

**Filter**

**downcounter**

**ECE**

**TIMx\_SMCR**

**SMS[2:0]**

Для тактового сигнала **CK\_PSC** таймера управления с расширенными функциями существует множество источников, которые можно разделить на 4 категории:

1. Маршрут ввода тактового сигнала от внешнего вывода (**ETR**): **ETR** → **ETRP** → **ETRF**;
2. Маршрут ввода внутреннего тактового сигнала APB: **CK\_INT**;
3. Маршрут от вывода канала сравнения/захвата (**TIMx\_CHx**): **TIMx\_CHx** → **TIx** → **TIxFPx**; этот маршрут также используется в режиме энкодера;
4. Вход от других внутренних таймеров: **ITRx**;

Реальная операция может быть разделена на 4 категории путем определения выбора входного импульса для источника **CK\_PSC**:

1. Выбор внутреннего тактового источника (**CK\_INT**);
2. Внешний тактовый источник, режим 1;
3. Внешний тактовый источник, режим 2;
4. Режим энкодера.

Эти 4 операции позволяют выбирать любой из перечисленных выше 4 источников тактового сигнала.

#### 10.2.2.1 Внутренний источник тактирования (CK\_INT)

Если поле **SMS** удерживается в состоянии 000b для запуска таймера управления с расширенными функциями, то выбирается внутренний тактовый источник (**CK\_INT**) в качестве тактового сигнала. В этом случае **CK\_INT** становится **CK\_PSC**.

#### 10.2.2.2 Внешний источник тактирования Mode 1

Когда область **SMS** установлена в 111b, включается внешний тактовый источник Mode 1. При активации внешнего тактового источника 1, в качестве источника **CK\_PSC** выбирается **TRGI**. Важно отметить, что источник **TRGI** также должен быть выбран путем конфигурации области TS. Область TS может выбрать следующие типы импульсов в качестве источников тактового сигнала:

1. Внутренние триггеры (**ITRx**, где x = 0, 1, 2, 3).
2. Сигнал после прохождения через детектор фронта канала захвата 1 (**TI1F\_ED**).
3. Сигналы **TI1FP1**, **TI2FP2** канала захвата.
4. Сигнал **ETRF** от входа внешнего тактового вывода

#### 10.2.2.3 Внешний источник тактирования Mode 2

Используйте внешний режим триггера 2 для подсчета на каждом восходящем или нисходящем фронте входного сигнала внешнего тактового вывода. Когда позиция **ECE** установлена, используется внешний тактовый источник режима 2. При использовании внешнего тактового источника режима 2, в качестве **CK\_PSC** выбирается **ETRF**. Пин **ETR** превращается в **ETRP** после прохождения через необязательный инвертор (**ETP**), делитель (**ETPS**), а затем в **ETRF** после прохождения через фильтр (**ETF**).

При установке битов позиции **ECE** и **SMS** на 111b, это эквивалентно выбору **ETRF** в качестве входа областью **TS**

#### 10.2.2.4 Режим энкодера

Установка **SMS** на 001b, 010b или 011b включит режим энкодера. Включение режима энкодера позволяет выбрать определенный уровень в **TI1FP1** и **TI2FP2** для передачи сигнала с другим перепадом фронта в качестве сигнала. Этот режим используется при работе с внешним энкодером. За подробностями о конкретных функциях обратитесь к разделу 10.3.9

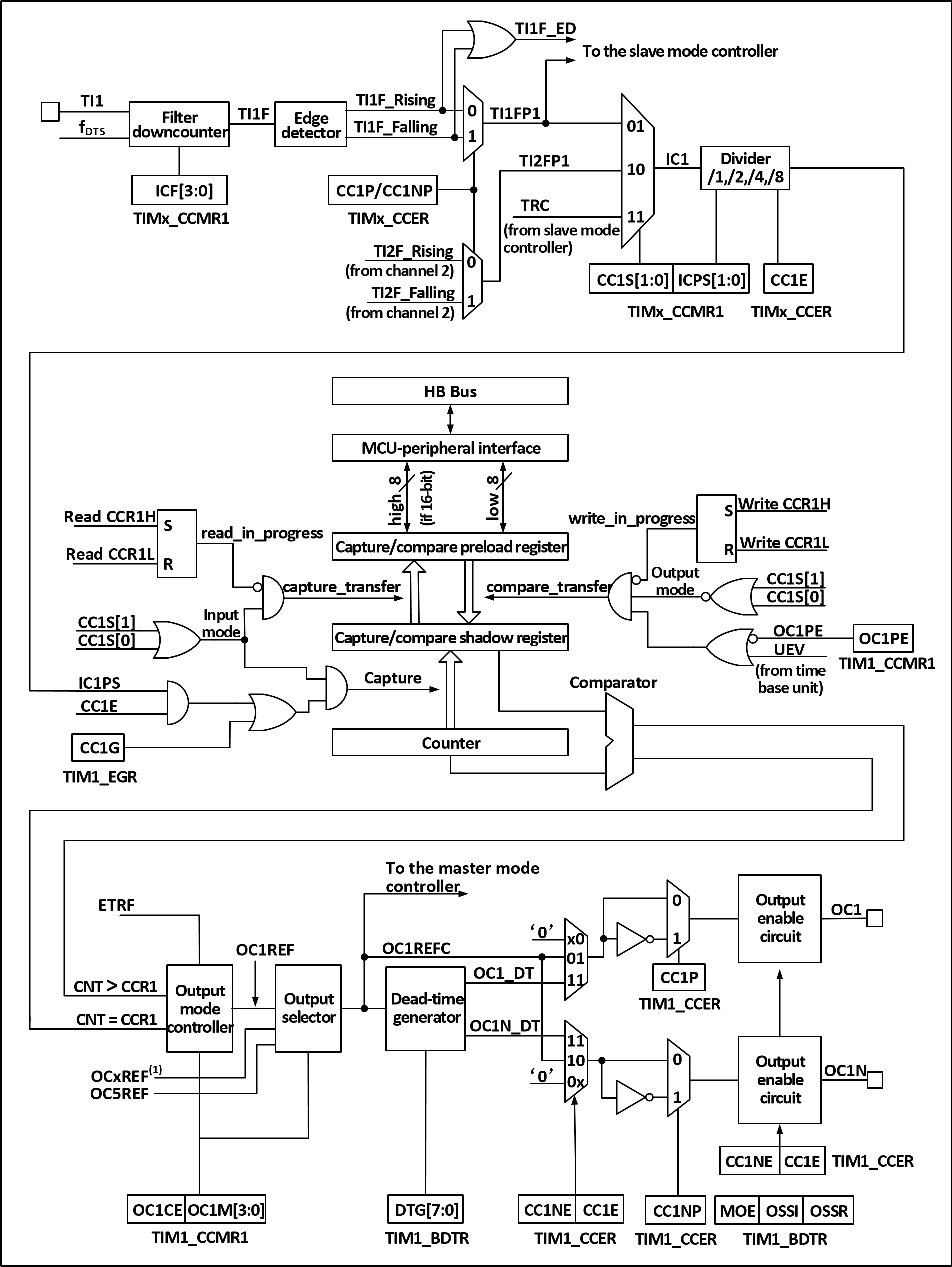
### 10.2.3 Счетчики и переферия

**CK\_PSC** поступает на предделитель (**PSC**) для деления. **PSC** является 16-битным, и реальный коэффициент деления равен значению **R16\_TIMx\_PSC** + 1. **CK\_PSC** проходит через **PSC** и становится **CK\_INT**. Изменение значения **R16\_TIM1\_PSC** не вступает в силу немедленно, но обновляется в **PSC** после события обновления. Событием обновления является очистка и сброс бита **UG**. Ядро таймера представляет собой 16-битный счётчик (**CNT**). В конечном итоге **CK\_CNT** подаётся на **CNT**, который поддерживает режимы инкрементного счёта, декрементного счёта и комбинированного инкрементно-декрементного счёта. Также имеется регистр автоматической перезагрузки (**ATRLR**), который перезагружает начальное значение для **CNT** в конце каждого цикла счёта. Есть также вспомогательный счётчик, который отслеживает количество раз, когда **ATRLR** перезагружал начальное значение для **CNT**, и может генерировать определённое событие, когда достигается количество раз, установленное в регистре счётчика повторений (**RPTCR**).

### 10.2.4 Каналы сравнения/захвата и периферии

Ядром таймера являются регистры сравнения/захвата, которые дополняются цифровой фильтрацией, делением частоты и мультиплексированием между каналами в секции входных периферийных компонентов, компаратором и управлением выхода в выходной секции.

Рисунок 10-3. Блок-схема структуры канала сравнения/захвата



Блок-схема структуры канала сравнения/захвата показана на рисунке 10-3. Сигнал поступает с вывода канала x и опционально формируется как **TIx** (источником **TI1** может быть не только **CH1**, см. блок-схему структуры таймера 10-1), **TI1** проходит через фильтр (**ICF**[3:0]), формируя **TI1F**, затем делится на **TI1F\_Rising** и **TI1FP\_Falling** через детектор фронта, эти два сигнала выбираются (**CC1P**) для создания **TI1FP1**, **TI1FP1** и **TI2FP1** из второго канала отправляются вместе в **CC1S** для выбора и превращаются в **IC1**, который отправляется в регистр сравнения/захвата после деления **ICPS**.

Регистр сравнения/захвата состоит из регистра предварительной загрузки и теневого регистра, а процесс чтения/записи выполняется только над регистром предварительной загрузки. В режиме захвата захват происходит в теневом регистре, а затем копируется в регистр предварительной загрузки; в режиме сравнения содержимое регистра предварительной загрузки копируется в теневой регистр, а затем содержимое теневого регистра сравнивается с основным счетчиком (**CNT**)

## 10.3 Функциональность и реализация

Реализация сложных функций таймера управления с расширенными возможностями осуществляется посредством работы каналов сравнения/захвата таймера, схемы ввода тактового сигнала, счетчика и периферийных компонентов. Тактовый сигнал, подаваемый на таймер, может поступать из множества источников, включая ввод от канала сравнения/захвата. Работа канала сравнения/захвата и выбор источника тактового сигнала непосредственно определяют его функциональность. Канал сравнения/захвата является двунаправленным и может работать как в режиме ввода, так и в режиме вывода.

### 10.3.1 Режим захвата входа (Input Capture)

Режим захвата входного сигнала является одной из базовых функций таймера. Принцип работы режима захвата входного сигнала заключается в том, что событие захвата происходит, когда обнаруживается определенный фронт сигнала **ICxPS**, и текущее значение счетчика фиксируется в регистре сравнения/захвата (**R16\_TIMx\_CHCTLRx**). Когда происходит событие захвата, устанавливается флаг **CCxIF** (в **R16\_TIMx\_INTFR**), и если разрешено прерывание или DMA, генерируется соответствующее прерывание или запрос DMA. Если флаг **CCxIF** уже установлен, когда происходит событие захвата, то устанавливается бит **CCxOF**. Флаг **CCxIF** может быть сброшен программно или аппаратно при чтении регистра сравнения/захвата. Бит **CCxOF** сбрасывается программно. Пример использования режима захвата на входе для канала 1 приведен ниже.

1. Настройте область **CCxS** для выбора источника сигнала **ICx**. Например, установите значение 10b и выберите **TI1FP1** в качестве источника **IC1** вместо использования значений по умолчанию, где область **CCxS** по умолчанию делает модуль сравнения/захвата каналом вывода.
2. Настройте область **ICxF** для установки цифрового фильтра для сигнала **TI**. Цифровой фильтр будет выборочно считывать сигнал с определенной частотой указанное количество раз, а затем выдавать скачок. Эта частота выборки и количество выборок определяется полем **ICxF**.
3. Настройте бит **CCxP** для установки полярности сигнала **TIxFPx**. Например, оставьте бит **CC1P** низким и выберите переходы по нарастающему фронту.
4. Настройте область **ICxPS** для установки коэффициента пересчета сигнала **ICx**. Например, оставляйте **ICxPS** равным 00b без пересчета.
5. Настройте бит **CCxE** для разрешения записи значения основного счетчика (**CNT**) в регистр сравнения/захвата. Установите бит **CC1E**.
6. Настройте биты **CCxIE** и **CCxDE** по мере необходимости, чтобы определить, следует ли разрешить прерывания или DMA. На этом завершается конфигурация канала сравнения/захвата.

Когда захваченный импульс подается на **TI1**, значение основного счетчика (**CNT**) записывается в регистр сравнения/захвата, устанавливается бит **CC1IF**, и бит **CCIOF** устанавливается, если ранее был установлен бит **CC1IF**. Если установлен бит **CC1IE**, то генерируется прерывание; если установлен бит **CC1DE**, генерируется запрос DMA. Событие захвата на входе может быть сгенерировано программно путем записи в регистр генерации событий (**TIMx\_SWEVGR**)

.

### 10.3.2 Режимы выходного сравнения (Compare Output)

Режим выходного сравнения является одной из основных функций таймера. Принцип работы этого режима заключается в том, что при совпадении значения основного счетчика (**CNT**) со значением регистра сравнения/захвата на выходе формируется конкретное изменение или волна. Поле **OCxM** (в **R16\_TIMx\_CHCTLRx**) и бит **CCxP** (в **R16\_TIMx\_CCER**) определяют, будет ли на выходе зафиксирован высокий или низкий уровень, или же произойдет смена уровня. Бит **CCxIF** также устанавливается, когда генерируется событие совпадения сравнения. Если заранее установлен бит **CCxIE**, генерируется прерывание; если заранее установлен бит **CCxDE**, генерируется запрос DMA.

Для настройки режима выходного сравнения выполните следующие действия:

1. Настройте источник тактового сигнала и значение авто перезагрузки основного счетчика (**CNT**).
2. Установите значение для сравнения в регистр сравнения/захвата (**R16\_TIMx\_CHxCVR**).
3. Если требуется сгенерировать прерывание, установите бит **CCxIE**.
4. Оставьте **OCxPE** равным 0, чтобы отключить регистр предварительной загрузки регистра сравнения.
5. Настройте режим вывода, установив поле **OCxM** и бит **CCxP**.
6. Включите вывод, установив бит **CCxE**.
7. Установите бит **CEN** для запуска таймера.

### 10.3.3 Режим принудительного вывода (Forced Output)

Шаблон вывода канала сравнения/захвата таймера может быть принудительно изменен программным обеспечением для вывода определенного уровня без необходимости сравнивать теневой регистр регистра сравнения/захвата с основным счетчиком. Для этого нужно установить **OCxM** в значение 100b, что приведет к принудительному снижению уровня **OCxREF**, или установить **OCxM** в 101b, чтобы принудительно повысить уровень **OCxREF**.

Обратите внимание, что при принудительном установлении **OCxM** на 100b или 101b процесс сравнения внутренних основных счетчиков и регистров сравнения/захвата все равно продолжается, соответствующие флаги устанавливаются, и продолжают генерироваться прерывания и запросы DMA.

### 10.3.4 Режим входа ШИМ (PWM Input)

Режим ввода ШИМ используется для измерения коэффициента заполнения и частоты сигнала ШИМ и является специальным случаем режима захвата входного сигнала. Операция аналогична режиму захвата входного сигнала, за исключением следующих отличий: ШИМ занимает два канала сравнения/захвата, и полярность ввода этих двух каналов установлена противоположной, один из сигналов настроен как вход триггера, а **SMS** установлен в режим сброса.

Например, чтобы измерить период и частоту входящего сигнала ШИМ от **TI1**, необходимы следующие операции.

1. Установите **TI1** (**TI1FP1**) в качестве входа сигнала **IC1**. Установите **CC1S** на 01b.
2. Установите **TI1FP1** на активный передний фронт. Держите **CC1P** на нуле.
3. Установите **TI1** (**TI1FP2**) в качестве входа для сигнала **IC2**. Установите **CC2S** на 10b.
4. Выберите **TI1FP2** для активного заднего фронта. Установите **CC2P** на 1.
5. Выберите **TI1FP1** в качестве источника тактового сигнала. Установите **TS** на 101b.
6. Установите **SMS** в режим сброса, то есть 100b.
7. Разрешите захват входного сигнала. Устанавливайте **cc1e** и **cc2e**.

Таким образом, значение регистра сравнения/захвата 1 является периодом ШИМ, а значение регистра сравнения/захвата 2 – его коэффициентом заполнения.

### 10.3.5 Режим выхода ШИМ (PWM Output)

Режим вывода ШИМ является одной из базовых функций таймера. Режим вывода ШИМ чаще всего используется для определения частоты ШИМ с помощью значения перезагрузки и коэффициента заполнения с использованием регистра сравнения/захвата. Чтобы использовать режим ШИМ 1 или 2, установите 110b или 111b в поле **OCxM**, установите бит **OCxPE** для включения регистра предварительной загрузки и, наконец, установите бит **ARPE** для разрешения автоматической перезагрузки регистра предварительной загрузки. Поскольку значение регистра предварительной загрузки может быть отправлено в теневой регистр только при возникновении события обновления, необходимо установить бит **UG** для инициализации всех регистров перед началом счета основным счетчиком. В режиме ШИМ основной счетчик и регистр сравнения/захвата всегда находятся в процессе сравнения, и в зависимости от бита **CMS** таймер способен выводить сигналы ШИМ, выровненные по краю или центру.

**Выравнивание по краю**

При использовании выравнивания по краю основной счетчик увеличивается или уменьшается, и в сценарии режима ШИМ 1 **OCxREF** находится в высоком состоянии, когда значение основного счетчика больше, чем у регистра сравнения/захвата, и переходит в низкое состояние, когда оно меньше (например, когда основной счетчик достигает значения **R16\_TIMx\_ATRLR** и возвращается к нулевым значениям).

**Центровка**

При использовании центрированных режимов основной счетчик работает в чередующихся режимах инкрементного и декрементного счета, и **OCxREF** совершает подъемы и спады, когда значения основного счетчика и регистра сравнения/захвата совпадают. Однако флаги сравнения устанавливаются в разное время в трех центральных режимах выравнивания. При использовании центрально выровненных режимов рекомендуется сгенерировать флаг программного обновления (установить бит **UG**) перед запуском основного счетчика.

### 10.3.6 Комплиментарные выходы и мертвые зоны (Complementary Outputs and Dead Zones)

Канал сравнения/захвата обычно имеет два выходных контакта (у четвертого канала сравнения/захвата есть только один выходной контакт), и он может выводить два комплементарных сигнала (**OCx** и **OCxN**). Для **OCx** и **OCxN** можно независимо настроить полярность с помощью битов **CCxP** и **CCxNP**, включить независимый вывод с помощью **CCxE** и **CCxNE**, а также независимо управлять выводом с помощью бит **MOE**, **OIS**, **OISN**, **OSSI** и **OSSR** для контроля мертвого времени и других параметров. Одновременное включение выходов **OCx** и **OCxN** вставляет зону задержки, и для каждого канала предусмотрен генератор зоны задержки на 10 бит. **OCx** и **OCxN** формируются на основе ассоциации с **OCxREF**. Если оба сигнала **OCx** и **OCxN** активны на высоком уровне, тогда **OCx** совпадает с **OCxREF**, за исключением того, что передний фронт **OCx** соответствует задержке относительно **OCxREF**, а **OCxN** является обратной величиной **OCxREF** таким образом, что его передний фронт будет иметь задержку относительно заднего фронта опорного сигнала. Если задержка превышает эффективную ширину выходного сигнала, соответствующий импульс не будет сгенерирован. На рис. 10-4 изображены отношения между **OCx**, **OCxN** и **OCxREF** с учетом зоны задержки.Начало формы

Рисунок 10-4. Комплементарные выходы и зона задержки.Начало формы



### 10.3.7 Сигнал тормоза (Brake Signal)

Когда генерируется сигнал тормоза, сигнал разрешения вывода и недействительный уровень модифицируются в соответствии с битами **MOE**, **OIS**, **OISN**, **OSSI** и **OSSR**. Однако ни в какой момент времени **OCx** и **OCxN** не будут находиться на активном уровне. Источник события тормоза может исходить от вывода тормозного контакта или это может быть событие отказа тактового сигнала, которое генерируется системой безопасности тактового сигнала (**CSS**). После сброса системы функция тормоза отключается по умолчанию (бит **MOE** установлен на низкий уровень), а установка бита **BKE** включает функцию тормоза. Полярность входного тормозного сигнала можно задать, установив **BKP**, а сигналы **BKE** и **BKP** могут быть записаны одновременно, и перед реальной записью есть задержка на один такт HB, поэтому вам нужно подождать один цикл HB, чтобы правильно считать записанное значение. При наличии выбранного уровня на контакте тормоза система выполнит следующие действия.

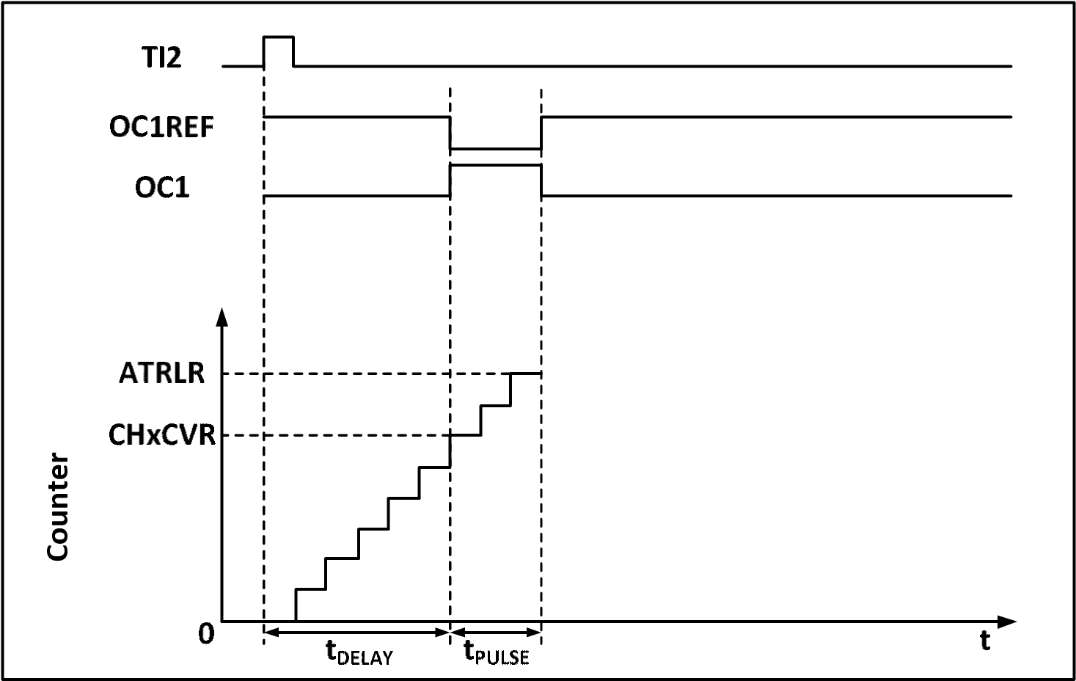
1. Биту **MOE** присваивается значение "0", и вывод устанавливается в неактивное, холостое или сбросовое состояние в зависимости от настройки бита **SOOI**.
2. После очистки бита **MOE** каждый выходной канал выводит уровень, определяемый битом **OISx**.
3. При использовании комплементарных выходов выходные данные переходят в нулевое состояние в соответствии с полярностью.
4. Если установлен бит **BIE**, генерируется прерывание, когда установлен бит **BIF**; если установлен бит BDE, создается запрос DMA.
5. Если установлено значение **AOE**, бит **MOE** автоматически устанавливается при следующем событии обновления **UEV**.

### 10.3.8 Режим одного импульса (Single Pulse)

Режим одиночного импульса позволяет микроконтроллеру реагировать на определенное событие, генерируя импульс после задержки, при этом длина и ширина импульса программируемые. Установка бита **OPM** позволяет основному счетчику останавливаться, когда генерируется следующее событие обновления **UEV** (счетчик сбрасывается до нуля).

Как показано на Рисунке 10-5, на выходе OC1 необходимо сгенерировать положительный импульс длительностью Tpulse после задержки Tdelay при обнаружении переднего фронта на входе TI2.

Рисунок 10-5. Генерация одиночного импульса.



1. Настройте **TI2** как триггер. Установите поле **CC2S** на 01b, чтобы сопоставить **TI2FP2** с **TI2**; установите бит **CC2P** на 0b, чтобы установить **TI2FP2** на детектирование переднего фронта; установите поле **TS** на 110b, чтобы установить **TI2FP2** в качестве источника триггера; установите поле **SMS** на 110b для использования **TI2FP2** для запуска счетчика.
2. Задержка Tdelay определяется значением регистра захвата сравнения, а длительность Tpulse определяется значениями регистра автоматического перезагрузки и регистра захвата/сравнения.

### 10.3.9 Режим энкодера (Encoder)

Режим энкодера является типичным применением таймера и может использоваться для доступа к выходному сигналу энкодера с двумя фазами. Направление счета основного счетчика синхронизируется с направлением вращения оси энкодера, и каждый импульс, поступающий от энкодера, вызывает увеличение или уменьшение значения основного счетчика на единицу. Чтобы использовать энкодер, установите поле **SMS** на 001b (подсчёт только по переднему фронту **TI2**), 010b (счёт только по переднему фронту **TI1**) или 011b (учет обоих фронтов **TI1** и **TI2**). Подключите энкодер к входу каналов сравнения/захвата 1 и 2 и установите значение для регистра перезагрузки, которое может быть увеличено. В режиме энкодера внутренние регистры захвата сравнения, предельное деление, регистр повторного счёта и другие компоненты таймера работают нормально. В следующей таблице показаны взаимосвязь направления счёта с сигналом энкодера.

Таблица 10-1. Взаимосвязь между направлением счёта таймера и сигналом энкодера в этом режиме.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Сонфигурация эфективного фронта | Уровень подчиненного сигнала | Фронт сигнала TI1FP | | Фронт сигнала TI2FP2 | |
| Позитивный фронт | Негативный фронт | Позитивный фронт | Негативный фронт |
| Счет только на фронте TI1 | Высокий | Счет вниз | Счет вверх | Нет счета | |
| Низкий | Счет вверх | Счет вниз |
| Счет только на фронте TI2 | Высокий | Нет счета | | Счет вверх | Счет вниз |
| Низкий | Счет вниз | Счет вверх |
| Счет по обеим фронтам TI1 и TI2 | Высокий | Счет вниз | Счет вверх | Счет вверх | Счет вниз |
| Низкий | Счет вверх | Счет вниз | Счет вниз | Счет вверх |

### 10.3.10 Режим синхронизации таймеров

Таймеры способны генерировать тактовые импульсы (**TRGO**) и принимать входы от других таймеров (**ITRx**). Источники **ITRx** (**TRGO** от других таймеров) различаются для разных таймеров. Соединения внутренних триггеров таймера приведены в Таблице 10-2.

Table 10-2 TIMx подключение внутренних триггеров

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Подчиненный таймер | ITR0 (TS=000) | ITR0 (TS=001) | ITR0 (TS=010) | ITR0 (TS=011) |
| TIM1 |  | TIM2 |  |  |
| TIM2 | TIM1 |  |  |  |

### 10.3.11 Debug Mode

Когда система входит в режим отладки, таймер продолжает работу или останавливается в зависимости от настроек модуля DBG.

## 10.4 Описание регистров

Table 10-3 TIM1-related registers list

### 10.4.1 Control Register 1 (TIM1\_CTLR1)

Offset address: 0x00

### 10.4.2 Control Register 2 (TIM1\_CTLR2)

Offset address: 0x04

### 10.4.3 Slave Mode Control Register (TIM1\_SMCFGR)

Offset address: 0x08

### 10.4.4 DMA/Interrupt Enable Register (TIM1\_DMAINTENR)

Offset address: 0x0C

### 10.4.5 Interrupt Status Register (TIM1\_INTFR)

Offset address: 0x10

### 10.4.6 Event Generation Register (TIM1\_SWEVGR)

### 10.4.7 Compare/Capture Control Register 1 (TIM1\_CHCTLR1)

Offset address: 0x18

### 10.4.8 Compare/Capture Control Register 2 (TIM1\_CHCTLR2)

Offset address: 0x1C

**10.4.9 Compare/Capture Enable Register 2 (TIM1\_CCER)**

Offset address: 0x20

### 10.4.10 Counter for Advanced-control Timer (TIM1\_CNT)

Offset address: 0x24

### 10.4.11 Counting Clock Prescaler (TIM1\_PSC)

Offset address: 0x28

### 10.4.12 Auto-reload Value Register (TIM1\_ATRLR)

Offset address: 0x2C

### 10.4.13 Repeat Count Value Register (TIM1\_RPTCR)

Offset address: 0x30

### 10.4.14 Compare/Capture Register 1 (TIM1\_CH1CVR)

### 10.4.15 Compare/Capture Register 2 (TIM1\_CH2CVR)

### 10.4.16 Compare/Capture Register 3 (TIM1\_CH3CVR)

Offset address: 0x3C

**10.4.17 Compare/Capture Register 4 (TIM1\_CH4CVR)**

### 10.4.18 Brake and Deadtime Register (TIM1\_BDTR)

Offset address: 0x44

### 10.4.19 DMA Control Register (TIM1\_DMACFGR)

### 10.4.20 DMA Address Register for Continuous Mode (TIM1\_DMAADR)

Offset address: 0x4C

# Глава 11: Универсальный таймер общего назначения (GPTM)

Модуль универсального таймера содержит 16-битный авто перезагружаемый таймер, TIM2, для измерения ширины импульса или генерации импульсов с определенной частотой, ШИМ-сигналов и т.д. Он может использоваться в системах автоматизации управления, источниках питания и других приложениях.

## Основные характеристики

Основные особенности универсального таймера включают:

* **16-битный авто перезагружаемый счетчик**, поддерживает режимы инкрементного счёта, декрементного счёта и комбинированного режима счёта (инкрементно-декрементный).
* **16-битный прескейлер** с динамически настраиваемым коэффициентом деления от 1 до 65536.
* **Поддерживает четыре независимых канала сравнения захвата**.
* Каждый канал сравнения захвата поддерживает несколько режимов работы, такие как: захват входных сигналов, сравнение выходов, генерацию ШИМ-сигнала и однократную генерацию импульса.
* Поддерживается управление таймером внешними сигналами.
* Поддержка DMA в нескольких режимах.
* Возможны инкрементация кодировки, каскадирование и синхронизация между несколькими таймерами.

## 11.2 Принцип работы и структура

Рисунок 11-1. Блок-схема структуры универсального таймера.

**CK\_TIM from RCC**

**Internal clock(CK\_INT)**

**TRGO**

**To other timers**

**To ADC**

**Reset,**

**Enable,**

**Up, Count**

**AutoReload**

**Register**

**U**

**CK\_PSC**

**PSC**

**(**

**prescaler**

**)**

**CK\_CNT**

**CNT**

**)**

**(**

**counter**

**Stop, clear or up**

**U**

**U**

**Trigger**

**controller**

### 11.2.1 Обзор

Как показано на рисунке 11-1, структуру универсального таймера можно условно разделить на три части: часть входного тактового сигнала, основную часть счетчика и часть каналов сравнения-захвата. Тактовый сигнал для универсального таймера может поступать от шины HB (**CK\_INT**), от внешнего входа тактового сигнала (**TIMx\_ETR**), от других таймеров с выходом тактового сигнала (**ITRx**) и от входа канала сравнения-захвата (**TIMx\_CHx**). Эти входные тактовые сигналы после различных операций фильтрации, деления и т.п. преобразуются в тактовые сигналы **CK\_PSC** и выводятся в секцию основного счетчика. Кроме того, эти сложные источники тактовых сигналов также могут быть выведены как **TRGO** для других периферийных устройств, таких как таймеры и АЦП. Ядром универсального таймера является 16-битный счетчик (**CNT**). Сигнал **CK\_PSC** делится с помощью преселектора (**PSC**), превращаясь в **CK\_CNT**, и затем подается на **CNT**. **CNT** поддерживает режимы инкрементного счета, декрементного счета и комбинированного инкрементно-декрементного режима, а также имеет регистр автоматической перезагрузки (**ATRLR**), который перезагружает начальное значение для **CNT** в конце каждого цикла счета. Универсальный таймер оснащен четырьмя наборами каналов сравнения-защиты, каждый из которых может принимать импульсы от эксклюзивных выводов или выводить волновые формы на выводы, то есть каналы сравнения-защиты поддерживают как входной, так и выходной режимы. Вход каждого канала регистра сравнения-зачата поддерживает фильтрацию, деление, обнаружение фронтов и другие операции, поддерживает взаимную активацию между каналами и может также обеспечивать тактовый сигнал для основного счетчика **CNT**. У каждого канала сравнения-зачета имеется набор регистров сравнения-зацета (**CHxCVR**), которые поддерживают сравнение с главным счетчиком (**CNT**) для выдачи импульсов.

### 11.2.2 Различия между универсальным таймером общего назначения и усовершенствованным таймером управления.

По сравнению с усовершенствованными таймерами управления, универсальные таймеры общего назначения не обладают следующими функциями:

1. Универсальный таймер не имеет регистра повторного счета для подсчета циклов счета основного счетчика.
2. Канал сравнения захвата универсального таймера не обладает функцией генерации мертвого времени и не имеет комплементарного выхода.
3. Универсальный таймер не располагает механизмом тормозного сигнала.

### 11.2.3 Вход тактирования

В этом разделе обсуждается источник сигнала CK\_PSC. Здесь представлена часть общей блок-схемы универсального таймера, относящаяся к источнику тактовых сигналов.

Рисунок 11-2. Блок-схема источника сигнала CK\_PSC универсального таймера.

**Encoder**

**mode**

**External clock**

**mode 1**

**External clock**

**mode 2**

**Internal clock**

**mode**

**CK\_PSC**

**TI2F**

**TI1F**

**or**

**or**

**TRGI**

**ETRF**

**CK\_INT**

**or**

**)**

**(**

**internal clock**

**TS[2:0]**

**TIMx\_SMCR**

**xx**

**0**

**100**

**101**

**110**

**111**

**ITRx**

**TI1\_ED**

**TI1FP1**

**TI2FP2**

**ETRF**

**0**

**1**

**CC2P**

**TIMx\_CCER**

**Edge**

**detector**

**TI2F\_Rising**

**TI2F\_Falling**

**Filter**

**ICF[3:0]**

**TIMx\_CCMR1**

**TI2**

**ETR pin**

**ETP**

**TIMx\_SMCR**

**0**

**1**

**ETR**

**Divider**

**/1,/2,/4,/8**

**ETPS[1:0]**

**TIMx\_SMCR**

**ETF[3:0]**

**TIMx\_SMCR**

**ETRP**

**f**

**DTS**

**Filter**

**downcounter**

**ECE**

**TIMx\_SMCR**

**SMS[2:0]**

Опциональные входные тактовые сигналы могут быть разделены на 4 категории.

1. Маршрут ввода внешнего тактового сигнала (**ETR**): **ETR** → **ETRP** → **ETRF**.
2. Маршрут внутреннего тактового сигнала HB: **CK\_INT**.
3. Маршрут от контакта канала сравнения захвата (**TIMx\_CHx**): **TIMx\_CHx** → **TIx** → **TIxFPx**; этот маршрут также используется в режиме энкодера.
4. Вход от других внутренних таймеров: **ITRx**. Фактическая операция может быть разделена на 3 категории путем определения выбора входного импульса для **SMS** источника **CK\_PSC**.
5. Выбор внутреннего источника тактового сигнала (**CK\_INT**).
6. Режим внешнего тактового источника 1.
7. Режим внешнего тактового источника 2.
8. Режим энкодера.

Все 4 упомянутых выше источника тактовых сигналов могут быть выбраны этими 4 операциями.

#### 11.2.3.1 Внутренний источник тактовых сигналов (CK\_INT)

Если универсальный таймер запускается при удержании поля SMS в состоянии 000b, то выбирается внутренний источник тактовых сигналов (CK\_INT) в качестве тактового сигнала. В этот момент CK\_INT становится CK\_PSC.

#### 11.2.3.2 Режим 1 (Mode 1) внешнего тактового источника

Когда область **SMS** установлена на 111b, включается первый режим внешнего тактового источника. При включении первого внешнего тактового источника в качестве источника для **CK\_PSC** выбирается **TRGI**. Следует отметить, что пользователь также должен выбрать источник для **TRGI**, настроив область **TS**. Область **TS** может выбирать следующие типы импульсов в качестве источников тактовых сигналов:

1. Внутренний триггер (**ITRx**, где x равно 0,1,2,3).
2. Сравнение сигнала после захвата канала 1 через детектор фронта (**TI1F\_ED**).
3. Сравнение сигналов **TI1FP1**, **TI2FP2** канала захвата.
4. Сигнал **ETRF** от входа внешнего тактового сигнала.

#### 11.2.3.3 Режим 2 (Mode 2) внешнего тактового источника

Используйте второй режим внешнего триггера для подсчета на каждом восходящем или нисходящем фронте входного сигнала внешнего тактового вывода. Когда позиция **ECE** установлена, используется второй режим внешнего тактового источника. При использовании второго режима внешнего тактового источника в качестве **CK\_PSC** выбирается **ETRF**. Контакт **ETR** становится **ETRP** после прохождения через необязательный инвертор (**ETP**), делитель (**ETPS**), а затем **ETRF** после прохождения через фильтр (**ETF**). При установке бита позиции **ECE** и **SMS** на 111b это эквивалентно выбору **ETRF** в качестве ввода с помощью **TS**.

#### 11.2.3.4 Режим энкодера

Установка SMS на 001b, 010b или 011b включит режим энкодера. Включение режима энкодера позволяет выбрать определенный уровень в **TI1FP1** и **TI2FP2** для передачи сигнала с другим перепадом уровня в качестве сигнала. Этот режим используется при применении внешнего энкодера. За подробностями о конкретных функциях обратитесь к разделу 11.3.7.

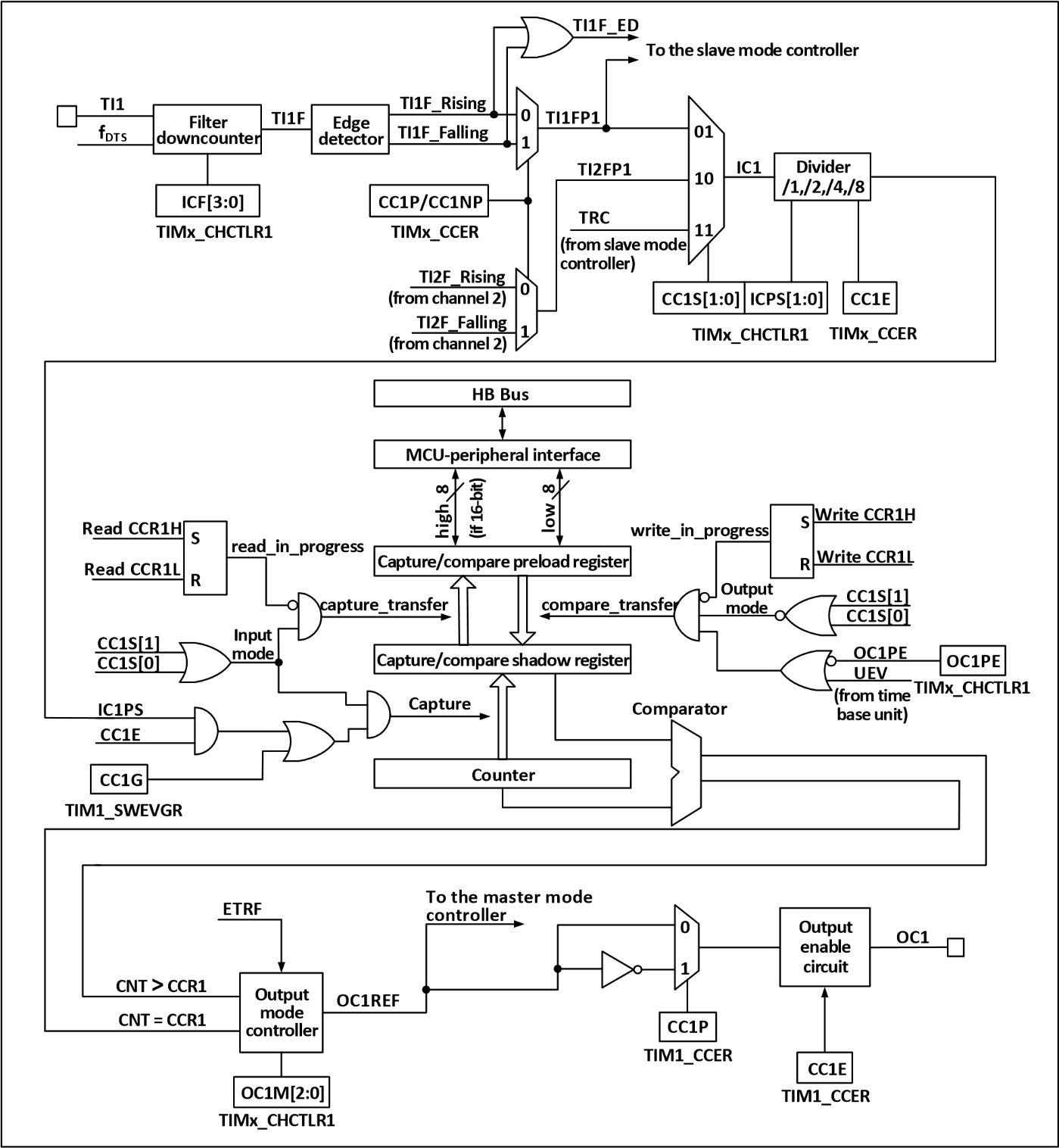
### 11.2.4 Счетчики и переферия

**CK\_PSC** поступает на предварительный делитель (**PSC**) для деления. **PSC** является 16-битным, и реальный коэффициент деления равен значению **R16\_TIMx\_PSC** + 1. **CK\_PSC** проходит через **PSC** и становится **CK\_INT**. Изменение значения **R16\_TIM1\_PSC** не вступает в силу в реальном времени, но обновляется в **PSC** после события обновления. Событие обновления включает очистку и сброс бита **UG**.

### 11.2.5 Каналы захвата и сравнения

Ядром канала сравнения/захвата, который является основой таймера для реализации сложных функций, является регистр сравнения/захвата. Он дополняется цифровой фильтрацией, делением частоты и межканальным мультиплексированием в секции периферийного ввода, а также компаратором и управлением вывода в секции вывода. Блок-схема структуры канала сравнения/захвата показана на Рисунке 11-3.

Рисунок 11-3. Блок-схема структуры канала сравнения захвата.



Сигнал вводится с контакта канала x и опционально формируется как **TIx** (источником **TI1** может быть больше чем CH1, см. блок-схему 10-1 таймера), **TI1** проходит через фильтр (**ICF**[3:0]), чтобы сформировать **TI1F**, а затем делится на **TI1F\_Rising** и **TI1F\_Falling** через детектор фронта, эти два сигнала выбираются (**CC1P**) для генерации **TI1FP1**, **TI1FP1** и **TI2FP1** из канала 2 отправляются вместе на **CC1S** для выбора и становятся **IC1**, который отправляется в регистр сравнения захвата после деления **ICPS**. Регистр сравнения захвата состоит из регистра предварительной загрузки и теневого регистра, а процесс чтения/записи выполняется только на регистре предварительной загрузки. В режиме захвата захват происходит на теневом регистре, а затем копируется в регистр предварительной загрузки; в режиме сравнения содержимое регистра предварительной загрузки копируется в теневой регистр, а затем содержимое теневого регистра сравнивается с основным счетчиком (**CNT**).

## 11.3 Функциональные возможности и реализация

Сложные функции универсального таймера реализуются путем манипуляции каналом сравнения захвата таймера, схемой ввода тактового сигнала и компонентами счетчика и периферии. Тактовый сигнал, подаваемый на таймер, может быть получен из множества источников тактовых сигналов, включая вход канала сравнения захвата. Работа основного канала сравнения захвата и выбор источника тактовых сигналов непосредственно определяют его функцию. Канал сравнения захвата является двунаправленным и может работать как в режиме ввода, так и в режиме вывода.

### 11.3.1 Режим захвата входа (Input Capture)

Режим захвата входного сигнала является одной из базовых функций таймера. Принцип работы режима захвата входного сигнала заключается в том, что при обнаружении определенного фронта сигнала **ICxPS** генерируется событие захвата, и текущее значение счетчика фиксируется в регистре сравнения захвата (**R16\_TIMx\_CHCTLRx**). Бит **CCxIF** (в **R16\_TIMx\_INTFR**) устанавливается при возникновении события захвата, и соответствующий прерывание или DMA генерируются, если они включены. Если бит **CCxIF** уже установлен при возникновении события захвата, то устанавливается бит **CCxOF**. Бит **CCxIF** может быть очищен программно или аппаратно путем чтения регистра сравнения захвата. Бит **CCxOF** очищается программно. Пример использования режима захвата на входе для канала 1 приведен ниже.

1. Настройте область **CCxS** для выбора источника сигнала **ICx**. Например, установите ее на 10b и выберите **TI1FP1** в качестве источника **IC1**, не используя настройку по умолчанию. По умолчанию область **CCxS** настраивает модуль сравнения захвата как выходной канал.
2. Настройте область **ICxF** для установки цифрового фильтра для сигнала **TI**. Цифровой фильтр будет выборочно считывать сигнал с определенной частотой определенное количество раз, а затем выдавать скачок. Эта частота выборки и количество раз определяются областью **ICxF**.
3. Настройте бит **CCxP** для установки полярности сигнала **TIxFPx**. Например, оставьте бит **CC1P** низким и выберите переходы на восходящий фронт.
4. Настройте область **ICxPS** для установки коэффициента перехода сигнала **ICx** между **ICxPS**. Например, оставляйте **ICxPS** на уровне 00b без перехода.
5. Настройте бит **CCxE**, чтобы разрешить захват значения основного счетчика (**CNT**) в регистр сравнения захвата. Установите бит **CC1E**.
6. Настройте биты **CCxIE** и **CCxDE** по мере необходимости, чтобы определить, следует ли разрешать использование прерываний или DMA. На этом завершается настройка канала сравнения захвата.

Когда захваченный импульс подается на **TI1**, значение основного счетчика (**CNT**) записывается в регистр сравнения захвата, устанавливается бит **CC1IF**, а бит **CCIOF** устанавливается, когда бит **CC1IF** был установлен ранее. Если установлен бит **CC1IE**, то генерируется прерывание; если установлен бит **CC1DE**, генерируется запрос DMA. Событие захвата на входе может быть сгенерировано программным способом посредством записи в регистр генерации событий (**R16\_TIMx\_SWEVGR**).

### 11.3.2 Режим сравнения (Compare Mode)

Режим сравнения выходов является одной из базовых функций таймера. Принцип работы режима сравнения выходов заключается в том, чтобы выводить определенное изменение или форму волны, когда значение основного счетчика (**CNT**) совпадает со значением регистра сравнения захвата. Поле **OCxM** (в **R16\_TIMx\_CHCTLRx**) и бит **CCxP** (в **R16\_TIMx\_CCER**) определяют, является ли выходной сигнал определенным высоким или низким уровнем или переключением уровня. Бит **CCxIF** также устанавливается, когда генерируется согласованное событие сравнения. Если предварительно установлен бит **CCxIE**, будет сгенерировано прерывание; если предварительно установлен бит **CCxDE**, будет сформирован запрос DMA. Чтобы настроить режим сравнения выходов, выполните следующие шаги:

1. Настройка источника тактового сигнала и значения автоперезагрузки основного счетчика (CNT).
2. Установка значения для сравнения с регистром сравнения захвата (R16\_TIMx\_CHxCVR).
3. Установка бита CCxIE, если необходимо сгенерировать прерывание.
4. Оставьте OCxPE равным 0, чтобы отключить регистр предварительной загрузки для регистра сравнения захвата.
5. Установка режима вывода, установка поля OCxM и бита CCxP.
6. Включите вывод, установив бит CCxE.
7. Установка бита CEN для запуска таймера.

### 11.3.3 Режим принудительного вывода (Forced Output Mode)

Выходной шаблон канала сравнения захвата таймера может быть принудительно изменен программным обеспечением для вывода определенного уровня без необходимости сравнения теневого регистра регистра сравнения захвата с основным счетчиком. Это делается путем установки OCxM на 100b, что вынуждает OCxREF опускаться до низкого уровня, или путем установки OCxM на 101b, что заставляет OCxREF подниматься до высокого уровня. Обратите внимание, что принудительная установка OCxM в положение 100b или 101b все еще продолжается процесс сравнения между внутренним основным счетчиком и регистром сравнения захвата, соответствующие флаги все еще устанавливаются, и все еще генерируются прерывания и запросы DMA.

### 11.3.4 Режим измерения ШИМ (PWM Input Mode)

Режим ввода ШИМ используется для измерения коэффициента заполнения и частоты ШИМ и является особым случаем режима захвата на входе. Операции аналогичны режиму захвата на входе, за исключением следующих отличий: ШИМ занимает два канала сравнения захвата, и полярность ввода двух каналов установлена противоположная, один из сигналов установлен как вход триггера, а **SMS** установлен в режим сброса. Например, чтобы измерить период и частоту входящего сигнала ШИМ от **TI1**, требуются следующие операции.

1. Установите TI1 (TI1FP1) в качестве входа сигнала IC1. Установите CC1S на 01b.
2. Установите TI1FP1 на активный фронт нарастания. Держите CC1P на нуле.
3. Установите TI1 (TI1FP2) в качестве входного сигнала IC2. Установите CC2S на 10b.
4. Выберите TI1FP2 для активации спадающего фронта. Установите CC2P на 1.
5. Выберите TI1FP1 в качестве источника тактового сигнала. Установите TS на 101b.
6. Выберите режим сброса для SMS, то есть 100b.
7. Включите захват на входе. Устанавливаются cc1e и cc2e.

### 11.3.5 Режим генерации ШИМ (PWM Output Mode)

ежим вывода ШИМ является одной из основных функций таймера. Режим вывода ШИМ чаще всего используется для определения частоты ШИМ с использованием значения перезагрузки и коэффициента заполнения с использованием регистра сравнения захвата. Установите 110b или 111b в поле **OCxM** для использования режима ШИМ 1 или режима 2, установите бит **OCxPE** для включения регистра предварительной загрузки и, наконец, установите бит ARPE для разрешения автоматического перезагрузки регистра предварительной загрузки. Значение регистра предварительной загрузки может быть отправлено в теневой регистр только тогда, когда происходит событие обновления, поэтому бит **UG** необходимо установить для инициализации всех регистров перед началом счета основного счетчика. В режиме ШИМ основной счетчик и регистр сравнения захвата всегда находятся в сравнении, и в зависимости от бита **CMS** таймер способен выводить сигналы ШИМ, выровненные по краю или центру.

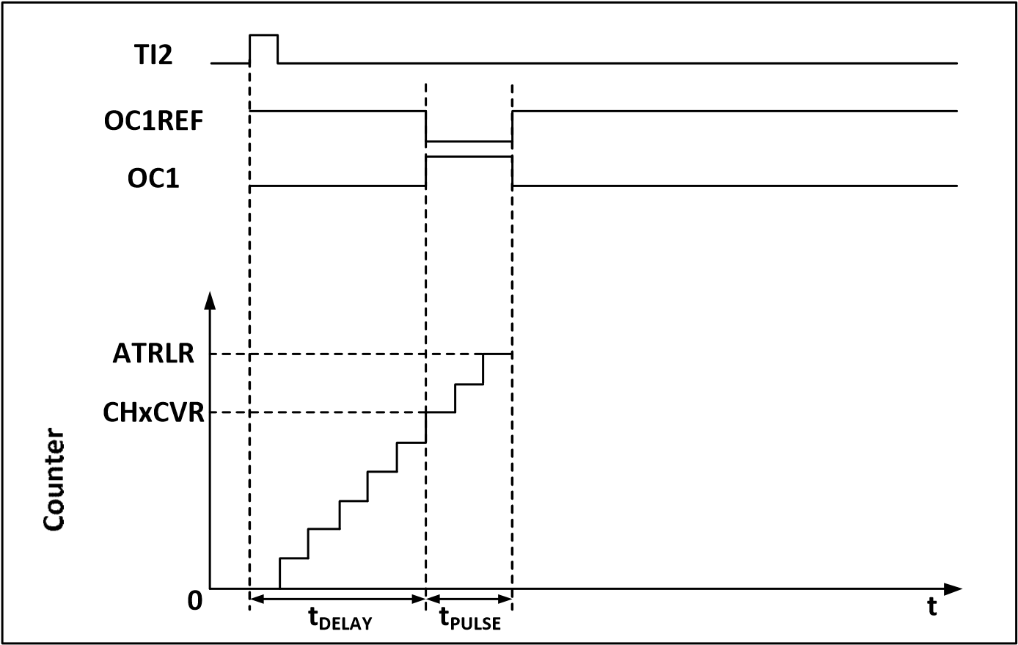
* Выравнивание по краю

При выравнивании по краю основной счетчик увеличивается или уменьшается, и в сценарии режима ШИМ 1, когда значение основного счетчика превышает значение регистра сравнения захвата, **OCxREF** поднимается до высокого уровня; когда значение главного счетчика меньше значения регистра сравнения (например, когда главный счетчик растет до значения **R16\_TIMx\_ATRLR** и возвращается к полному нулю), **OCxREF** падает до низкого уровня. Выравнивание по центру При использовании режима центрированного выравнивания основной счетчик работает в чередовании режимов инкрементного и декрементального счета, и **OCxREF** выполняет подъемы и падения, когда значения основного счетчика и регистра сравнения совпадают. Тем не менее, флажки сравнения устанавливаются в разное время в трех режимах центрального выравнивания. При использовании режимов центрального выравнивания рекомендуется сгенерировать флаг программного обновления (установить бит **UG**) перед запуском основного счетчика.

### 11.3.6 Режим одного импульса (Single Pulse Mode)

Режим одиночного импульса может реагировать на конкретное событие, генерируя импульс после задержки с программируемыми задержкой и шириной импульса. Установка бита **OPM** останавливает основной счетчик, когда генерируется следующее событие обновления **UEV** (счетчик переходит в ноль).

Рисунок 11-4. Генерация событий и реакция на импульс



Как показано на Рисунке 11-4, на выводе OC1 необходимо сгенерировать положительный импульс длиной Tpulse после задержки Tdelay в начале растущего фронта, обнаруженного на выводе **TI2**. Для этого требуется выполнить следующие шаги:

1. Настройки для генерации триггера. В поле **CC2S** установлено значение 01b, которое отображает **TI2FP2** как **TI2**; бит **CC2P** установлен на 0b, что устанавливает **TI2FP2** для обнаружения растущего края; поле **TS** установлено на 110b, устанавливающее **TI2FP2** в качестве триггерного источника; поле **SMS** установлено на значение 110b для установки **TI2FP2**, используемого для запуска счетчика.
2. Задержка Tdelay определяется значением регистра сравнения захвата, а ширина импульса Tpulse определяется значениями регистра автоперезагрузки и регистра сравнения захвата.

### 11.3.7 Режим энкодера (Encoder Mode)

Режим энкодера является типичным применением таймера и может использоваться для получения фазового выхода энкодера. Направление счёта основного счётчика синхронизируется с направлением вращения оси энкодера, и каждое импульс, исходящий от энкодера, увеличит или уменьшит значение основного счётчика на единицу. Чтобы использовать энкодер, установите поле **SMS** на 001b (подсчёт только по краю **TI2**), 010b (счёт только на краю **TI1**) или 011b (Счёт на краях **TI1** и **TI2**). Подключите энкодер к входам каналов сравнения захвата 1 и 2, и установите значение счётчика перегрузки, которое может быть установлено на большее значение. В режиме энкодера внутренние регистры сравнения захвата, пресет-счётчик, счётчик повторяющихся чисел и т.д. таймера работают нормально. В следующей таблице показаны отношения между направлением счёта и сигналом энкодера.

Таблица 11-1 Связь между направлением счёта и сигналом энкодера в режиме энкодера таймера

### 11.3.8 Режим синхронизации таймера (Timer Synchronization Mode)

Таймеры способны генерировать тактовые импульсы (TRGO) и получать входы от других таймеров (ITRx). Источник ITRx (TRGO от других таймеров) различается для разных таймеров. Соединения внутренней связи триггеров таймера показаны в Таблице 11-2.

Таблица 11-2 Подключение внутренних GTPM триггеров таймера

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| От таймера | ITR0(TS=000) | ITR0(TS=001) | ITR0(TS=010) | ITR0(TS=011) |
| TIM2 | TIM1 |  |  |  |
| TIM1 |  | TIM2 |  |  |

### 11.3.9 Режим наладки

Когда система входит в режим отладки, таймер может продолжать выполнение команд или остановиться в зависимости от настроек модуля DBG.

## 11.4 Описание регистров

Table 11-3 TIM2-related registers list

### 11.4.1 Control Register 1 (TIM2\_CTLR1)

### 11.4.2 Control Register 2 (TIM2\_CTLR2)

Offset address: 0x04

### 11.4.3 Slave Mode Control Register (TIM2\_SMCFGR)

Offset address: 0x08

**11.4.4 TIM2 DMA/Interrupt Enable Register (TIM2\_DMAINTENR)**

### 11.4.5 Interrupt Status Register (TIM2\_INTFR)

Offset address: 0x10

### 11.4.6 TIM2 Event Generation Register (TIM2\_SWEVGR)

Offset address: 0x14

### 11.4.7 Compare/Capture Control Register 1 (TIM2\_CHCTLR1)

Offset address: 0x18

Канал может использоваться в режиме ввода (режим захвата) или вывода (режим сравнения), а направление канала задаётся соответствующим битом **CCxS**. Другие биты этого регистра выполняют разные функции в режимах ввода и вывода. Биты **OCxx** описывают функциональность канала в режиме вывода, а биты **ICxx** описывают функциональность канала в режиме ввода.

### 11.4.8 Compare/Capture Control Register 2 (TIM2\_CHCTLR2)

Offset address: 0x1C

Канал может быть использован в режиме ввода (режим захвата) или вывода (режим сравнения), а направление канала определяется соответствующим битом **CCxS**. Остальные биты этого регистра служат различными целями в режимах ввода и вывода. Биты **OCxx** описывают функцию канала в режиме вывода, а биты **ICxx** описывают функцию канала в режиме ввода.

### 11.4.9 Compare/Capture Enable Register (TIM2\_CCER)

Offset address: 0x20

### 11.4.10 Counter for General-purpose Timer (TIM2\_CNT)

### 11.4.11 Counting Clock Prescaler (TIM2\_PSC)

### 11.4.12 Auto-reload Value Register (TIM2\_ATRLR)

### 11.4.13 Compare/capture Register 1 (TIM2\_CH1CVR)

Offset address: 0x34

### 11.4.14 Compare/capture Register 2 (TIM2\_CH2CVR)

### 11.4.15 Compare/capture Register 3 (TIM2\_CH3CVR)

### 11.4.16 Compare/capture Register 4 (TIM2\_CH4CVR)

### 11.4.17 DMA Control Register (TIM2\_DMACFGR)

Offset address: 0x48

### 11.4.18 DMA Address Register for Continuous Mode (TIM2\_DMAADR)

# Глава 12: Универсальный синхронно-асинхронный приёмопередатчик (USART)

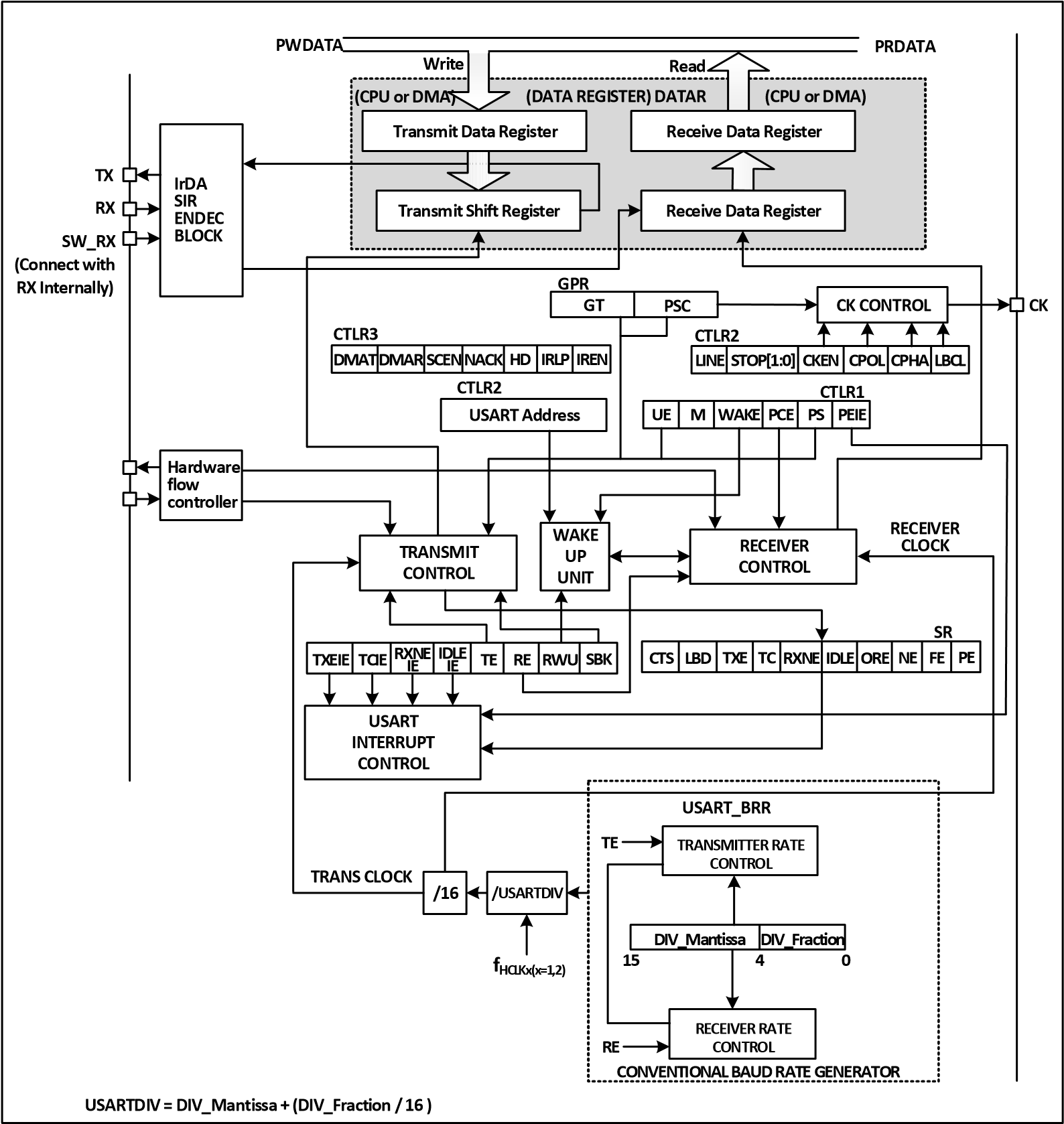
Модуль включает в себя универсальный синхронно-асинхронный приемопередатчик USART1.

## 12.1 Основные характеристики

* Полнодуплексная или полудуплексная синхронная или асинхронная связь
* Формат данных NRZ
* Генератор дробной скорости до 3 Mbps
* Программируемая длина данных
* Конфигурация стоповых битов
* Поддержка протоколов LIN, кодировщиков IrDA, смарт-карт
* Поддержка DMA
* Множество источников пререрываний

## 12.2 Обзор

Рисунок 12-1. Блок-диаграмма универсального синхронно-асинхронного приемопередатчика



Когда бит **TE** (бит разрешения передачи) установлен, данные из регистра сдвига передачи выводятся на пин TX, а тактовый сигнал передается на пина CK. При передаче данных таймер выводит самый незначимый бит (**LSB**) из сдвигового регистра на пину TX, и каждая передача данных начинается с низкоуровневого стартового бита. Передача данных осуществляется передачей 8-битового или 9-битового слова данных в зависимости от настройки бита **M** (длины слова). После завершения передачи данных и установки бита **TE**, передатчик передает кадр начала, содержащий низкий стартовый бит, а затем 8 или 9 бит данных, в зависимости от установки битов **M**. В случае наличия бита проверки на четность, последний бит данных служит битами проверки. После отправки кадра, при установке бита **TE**, посылается кадр «idle», состоящий из 10 или 11 высоких битов, за которыми следует бит остановки. Кадр отключения содержит 10 или 10 бит низких данных, за которым следует бит останова.

## 12.3 Тактирование генератора битрейта передачи (Baud Rate Generator)

Скорость передачи данных передатчика рассчитывается по формуле BaudRate = Fbaud, где Fbaud – это частота передачи данных, а **HCLK** – это тактовая частота шины HB. Значение **USARTDIV** определяется двумя полями, **DIV\_M** и **DIV\_F** в регистре **USART\_BRR**. Формула расчета выглядит следующим образом:

Где **HCLK** — это частота тактового генератора шины HB, а **USARTDIV** — это значение, определяемое полями **DIV\_M** and **DIV\_F**. Отсюда:

Важно отметить, что скорость передачи битов, генерируемая генератором скорости передачи символов (baud rate), может не всегда точно соответствовать требуемой пользователем скорости передачи данных, и возможны отклонения. Помимо выбора максимально близкого значения, одним из способов уменьшения отклонений является увеличение частоты тактового генератора HB. Например, если вы установите скорость передачи данных 115200 бит/с, значение **USARTDIV** будет установлено равным 39,0625, что даст вам точную скорость передачи 115200 бит/с при максимальной частоте, но если вам нужна скорость передачи данных 921600 бит/с, рассчитанное значение **USARTDIV** составляет 4,88, однако ближайшее значение, которое можно установить в **USART\_BRR**, фактически равно только 4,875. При значении 4,875 реальная скорость передачи составит 923076 бит/с с ошибкой 0,16%. Когда последовательный сигнал, отправленный передатчиком, передается приемнику, между скоростью передачи приемника и передатчика могут возникать некоторые ошибки. Ошибка возникает по трем основным причинам: действительная скорость передачи приемника отличается от таковой у передатчика; часы приемника и передатчика имеют погрешности; изменения, возникающие в сигнале линии. Приемник периферийного модуля имеет определенную допустимую погрешность приема, когда сумма всех трех вышеуказанных отклонений меньше предела допуска модуля, суммарное отклонение не влияет на передачу и прием. Предел допуска модуля зависит от использования дробной скорости передачи и длины слова данных (Mбит), использование дробной скорости передачи данных и длина поля данных в 9 бит уменьшают предел допуска, но он не должен быть менее 3%.Начало формы

## 12.4 Синхронный режим (Synchronous Mode)

## Режим синхронизации позволяет системе выдавать сигнал синхронизации при использовании модуля USART. Когда включается синхронный режим для передачи данных во внешнюю среду, контакт CK одновременно начнет выдавать внешний тактовый сигнал.

## Чтобы включить синхронный режим, необходимо установить бит CLKEN в контрольном регистре 2 (R16\_USARTx\_CTLR2). Также требуется отключить режимы LIN, смарт-карты, инфракрасный и полудуплексный режимы, то есть убедиться, что биты SCEN, HDSEL и IREN сброшены. Эти три бита находятся в контрольном регистре 3 (R16\_USARTx\_CTLR3).Начало формы

## Ключевой момент при использовании синхронного режима – это управление выходом тактового сигнала. Следует учесть несколько важных аспектов:

## Режим синхронизации модуля USART работает только в основном режиме, то есть контакт CK выдает только тактовый сигнал и не принимает входных данных.

## Тактовый сигнал выдается только тогда, когда данные отправляются через контакт TX.

## Бит LBCL определяет, будет ли выдаваться тактовый сигнал после отправки последнего бита данных, бит CPOL задает полярность тактового сигнала, а бит CPHA — его фазу. Эти три бита расположены в контрольном регистре 2 (R16\_USARTx\_CTLR2) и должны быть настроены до активации режимов TE и RE. Различия между этими настройками показаны на Рисунке 12-2.

## Приемник будет производить выборку данных только по выходному тактовому сигналу в синхронном режиме, требуя определенного времени нарастания и удержания сигнала от устройства, как показано на Рисунке 12-3.

Рисунок 12-2 Пример временных характеристик тактового сигнала USART (M=0)

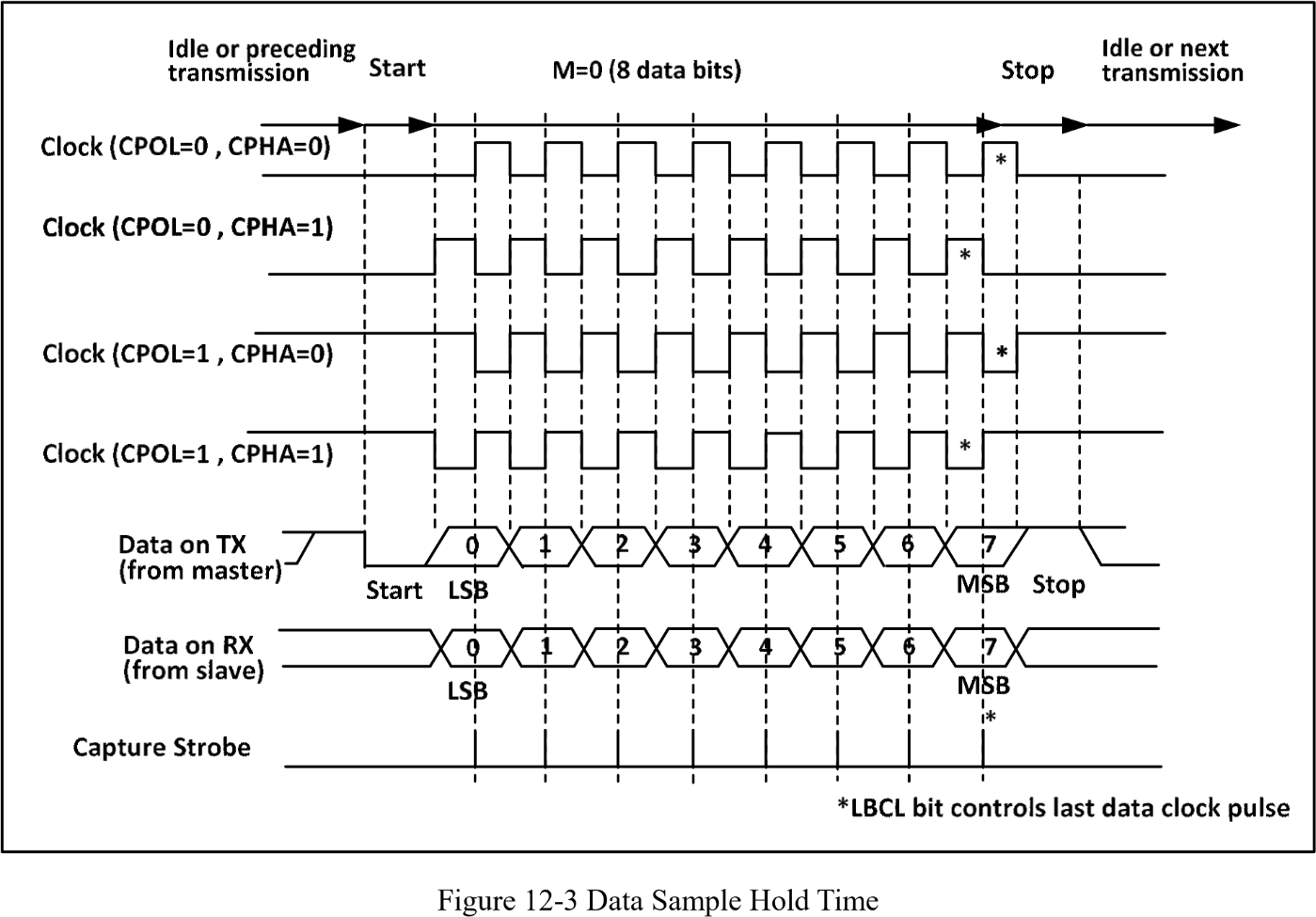


Рисунок 12-3 Время удержания выборки данных

**CK (capture strobe on CK**

**rising edge in this example)**

**Data on RX**

**from slave**

**)**

**(**

**valid DATA bit**

**t**

**SETUP**

**t**

**HOLD**

**t**

**SETUP**

**=**

**t**

**HOLD**

**1**

**/16 bit time**

## 12.5 Однопроводный полудуплексный режим (1-Wire Half-duplex Mode)

Полудуплексный режим поддерживает использование одного контакта (только TX) для приёма и передачи данных, при этом контакты TX и RX соединяются внутри чипа.

Для включения полудуплексного режима необходимо установить бит **HDSEL** в позиции регистра управления 3 (**R16\_USARTx\_CTLR3**), но также необходимо отключить режим LIN, режим смарт-карты, ИК-режим и синхронный режим, то есть обеспечить сброс битов **SCEN**, **CLKEN** и **IREN**, находящихся в регистрах управления 2 и 3 (**R16\_USARTx\_CTLR2** и **R16\_USARTx\_CTLR3**).

После установки в полудуплексный режим необходимо настроить порт ввода-вывода TX на работу в режиме открытого стока с высоким уровнем выхода. При установке **TE** данные будут отправляться сразу же после записи их в регистр данных. Особое внимание следует уделить тому факту, что полудуплексный режим может привести к конфликтам шины при использовании нескольких устройств одной шины для отправки и получения данных, чего пользователь должен избегать программными средствами.

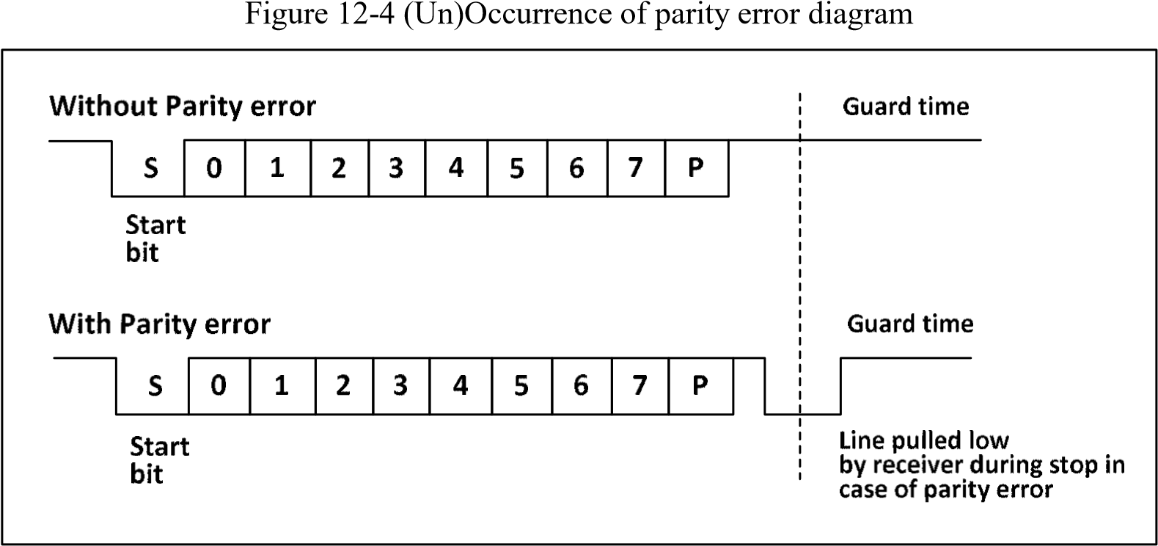
## 12.6 Режим смарткарт (Smart Card)

Режим смарт-карты поддерживает доступ к контроллерам смарт-карт по протоколу ISO7816-3.

Для включения режима смарт-карта необходимо установить бит **SCEN** в позиции регистра управления 3 (**R16\_USARTx\_CTLR3**), но также необходимо отключить режим LIN, полудуплексный режим и ИК-режим, то есть обеспечить сброс битов **LINEN**, **HDSEL** и **IREN**, но **CLKEN** может быть включён для вывода тактового сигнала, эти биты находятся в регистрах управления 2 и 3 (**R16\_USARTx\_CTLR2** и **R16\_USARTx\_CTLR3**).

Для поддержки режима смарт-карточек USART должен быть установлен на 8 бит данных плюс 1 бит чётности, а стоп-бит рекомендуется настраивать на 1,5 бита как для передачи, так и для приёма. Режим смарткарты представляет собой полудуплексный протокол с использованием одной линии связи, где используется линия TX для обмена данными и должна быть сконфигурирована как выходной каскад с открытым стоком плюс подтягивающий резистор. Когда приёмник получает кадр данных и обнаруживает ошибку чётности, он посылает сигнал NACK, то есть активно притягивает линию TX вниз на один цикл во время стоп-бита, а отправитель обнаруживает сигнал NACK и генерирует ошибку кадра, благодаря чему приложение может повторно передать данные. На рис. 17-4 показаны формы сигналов на контакте TX в нормальном случае и в случае ошибки чётности. Флаг **TC** (флаг завершения передачи) USART может задержать генерацию **GT** (время защиты) на один такт, и приёмник не распознаёт установленный им сигнал NACK как стартовый бит.

Рисунок 12-4 Диаграмма (не)возникновения ошибки чётности



В режиме смарт-карты форма сигнала, выводимого контактом CK при включении, никак не связана с коммуникацией; она просто подаёт тактовый сигнал на смарт-карту со значением тактового сигнала AHB, разделённого на пять бит (в два раза больше значения **PSC**, вплоть до 62 делений).

## 12.7 IrDA

Модуль USART поддерживает управление инфракрасными трансиверами IrDA для физической коммуникации. Для использования IrDA необходимо сбросить биты **LINEN**, **STOP**, **CLKEN**, **SCEN** и **HDSEL**. Между модулем USART и физическим уровнем SIR (инфракрасный трансивер) используется кодирование **NRZ** (без возврата к нулю), поддерживаемое до скоростей 115200 бод.

IrDA является полудуплексным протоколом. Если USART передает данные физическому уровню **SIR**, декодер IrDA игнорирует вновь поступающий инфракрасный сигнал. Если USART принимает данные от **SIR**, то **SIR** не примет сигнал от USART. Логика уровней USART для **SIR** и **SIR** для USART различается. В логике приема **SIR** высокий уровень соответствует 1, а низкий — 0, но в логике передачи SIR высокий уровень равен 0, а низкий — 1.

## 12.8 Прямой доступ к памяти (DMA)

Модуль USART поддерживает функцию DMA, которая может использоваться для достижения быстрой и непрерывной передачи и приема данных. Когда DMA активирован, DMA записывает данные из заданного пространства памяти в буфер передачи, когда установлен флаг **TXE**. При использовании DMA для приема данных, каждый раз, когда устанавливается флаг **RXNE**, DMA переносит данные из буфера приема в определенное пространство памяти.

## 12.9 Прерывания

Модуль USART поддерживает различные источники прерываний, включая пустоту регистра передачи данных (**TXE**), CTS, завершение передачи (**TC**), готовность принимаемых данных (**RXNE**), переполнение данных (**ORE**), неактивную линию (IDLE), ошибку четности (**PE**), флаг отключения (**LBD**), шум (**NE**), переполнение для многобуферной связи (**ORT**) и ошибку кадра (**FE**), среди прочих.

Таблица 12-1 Соотношение между прерываниями и соответствующими битами разрешения

## 12.10 Описание регистров

Таблица 12-2 Список регистров, связанных с USART

Начало формы

### 12.10.1 USART Status Register (USART\_STATR)

### 12.10.2 USART Data Register (USART\_DATAR)

### 12.10.3 USART Baud Rate Register (USART\_BRR)

### 12.10.4 USART Control Register 1 (USART\_CTLR1)

### 12.10.5 USART Control Register 2 (USART\_CTLR2)

### 12.10.6 USART Control Register 3 (USART\_CTLR3)

### 12.10.7 USART Guard Time and Prescaler Register (USART\_GPR)

# Глава 13 Интерфейс межмикросхемной связи (I2C)

Шина внутренней микросхемы (I2C) широко применяется для связи микроконтроллеров с датчиками и другими внекристальными модулями. Она поддерживает режимы с несколькими ведущими и несколькими ведомыми устройствами и может обеспечивать связь на скоростях 100 КГц (стандартная) и 400 КГц (быстрая) всего лишь двумя линиями (SDA и SCL). Поддерживаются функции тайминга и DMA, а также функция проверки контрольной суммы CRC.

## 13.1 Основные характеристики

* Поддержка режимов мастера и слейва
* Поддержка 7-битных и 10-битных адресов
* Устройства-слейвы поддерживают двойные 7-битные адреса
* Поддержка двух скоростных режимов: 100 КГц и 400 КГц
* Множество состояний и множество флагов ошибок
* Поддержка расширенных функций часов
* Два вектора прерываний
* Поддержка DMA
* Поддержка PEC
* Совместимость с SMBus

## 13.2 Обзор

I2C представляет собой полудуплексную шину, которая может работать только в одном из следующих четырех режимов одновременно: режим передачи ведущего устройства, режим приема ведущего устройства, режим передачи ведомого устройства и режим приема ведомого устройства. Модуль I2C по умолчанию работает в режиме слейва и автоматически переключается в режим мастера при генерации условия старта и в режим слейва при потере арбитража или генерации сигнала остановки. Модуль I2C поддерживает функциональность с несколькими мастерами. Работая в режиме мастера, модуль I2C активно излучает данные и адреса. Как данные, так и адрес передаются блоками по 8 бит, причем старший бит идет первым, а младший — последним. После события старта следует однобайтовая (в режиме 7-битного адреса) или двухбайтовая (в режиме 10-битного адреса) информация об адресе, и для каждых 8-битных данных или адреса, отправленных хостом, слейву необходимо ответить подтверждением ACK, притянув шину SDA к низкому уровню, как показано на Рисунке 13-1.

Рисунок 13-1 Временная диаграмма I2C



Для корректной работы I2C необходимо подавать правильный тактовый сигнал, который должен быть минимум 2 МГц в стандартном режиме и 4 МГц в быстром режиме.

На рисунке 13-2 представлена функциональная блок-схема модуля I2C.

**Noise**

**filter**

**Data**

**control**

**Data shift register**

**Data register**

**Comparator**

**PEC calculation**

**Own address register**

**Dual address register**

**PEC register**

**Noise**

**filter**

**Clock**

**control**

**Clock control**

**Register (CKCFGR)**

**Control registers**

**CTLR1&CTLR**

**2)**

**(**

**Status registers**

**(**

**2)**

**STAR1&STAR**

**Control**

**logic**

**SDA**

**SCL**

**SMBA**

**Interrupts**

**DMA requests & ACK**

## 13.3 Режим мастера (Master Mode)

В режиме мастера модуль I2C управляет передачей данных и выдает сигнал синхронизации, а передача данных начинается с события начала и заканчивается событием окончания. Шаги для использования общения в режиме мастера следующие. Установка правильного тактового сигнала в регистре управления 2 (R16\_I2Cx\_CTLR2) и регистре контроля тактового сигнала (R16\_I2Cx\_CKCFGR). Установка соответствующего фронта в регистре фронтов (R16\_I2Cx\_RTR). Установка бита PE в регистре управления (R16\_I2Cx\_CTLR1) для запуска периферии. Установка бита START в регистре управления (R16\_I2Cx\_CTLR1) для генерации события начала.После установки бита START модуль I2C автоматически перейдет в основной режим, бит MSL будет установлен и событие начала будет сгенерировано. После генерации события начала будет установлен бит SB, и если установлен бит ITEVTEN (в R16\_I2Cx\_CTLR2), произойдет прерывание. В этот момент следует прочитать регистр статуса 1 (R16\_I2Cx\_STAR1), а бит SB будет автоматически очищен после записи адреса в регистр данных.

Если используется режим 10-битного адреса, то регистр записи данных отправляет заголовочную последовательность (заголовочная последовательность имеет вид 11110xx0b, где биты xx являются старшими двумя битами 10-битного адреса). После отправки заголовочной последовательности в регистре состояния будет установлен бит ADD10, и если был установлен бит ITEVTEN, будет сгенерировано прерывание. В этот момент следует прочитать регистр R16\_I2Cx\_STAR1 и очистить бит ADD10 после записи второго байта адреса в регистр данных.

Затем запишите второй байт адреса в регистр данных для его отправки. После отправки второго байта адреса в регистре состояния будет установлен бит ADDR. Если уже установлен бит ITEVTEN, будет сгенерировано прерывание. В этот момент следует прочитать регистр R16\_I2Cx\_STAR1, а затем однократно прочитать регистр R16\_I2Cx\_STAR2, чтобы очистить бит ADDR;

Если используется режим 7-битного адреса, то запишите байт адреса в регистр данных. После отправки байта адреса в регистре состояния установится бит ADDR. Если установлен бит ITEVTEN, то будет сгенерировано прерывание. В этом случае следует прочитать регистр R16\_I2Cx\_STAR1 и затем однократно прочитать регистр R16\_I2Cx\_STAR2 для очистки бита ADDR.

В режиме 7-битного адреса первый отправляемый байт — это байт адреса, первые 7 бит представляют адрес целевого ведомого устройства, 8-й бит определяет направление последующего сообщения: 0 означает, что ведущее устройство записывает данные в ведомое устройство, 1 означает, что ведущее устройство считывает информацию из ведомого устройства.

В режиме 10-битного адреса, как показано на Рисунке 13-3, на этапе отправки адреса первый байт имеет формат 11110xx0, где xx — это две старших цифры 10-битного адреса, а второй байт содержит младшие 8 бит 10-битного адреса. Если впоследствии предполагается перейти в режим передачи данных мастером, продолжайте отправлять данные; если планируется переход в режим приема данных мастером, необходимо повторно отправить условие начала, затем отправить байт в формате 11110xx1 и после этого перейти в режим приема данных мастером.

Рисунок 13-3 Диаграмма отправки и получения данных мастером при 10-битном адресе

**S**

**1**

**1 1 1 0 X X**

**0**

**A**

**Address 7- 0**

**A**

**DATA**

**A**

**DATA**

**A**

**P**

**(**

**The upper 2 bits**

**of the address)**

**)**

**(**

**Write**

**The lower 8 bits of**

**the address**

**Transmitter**

**S**

**1**

**1 1 1 0 X X**

**0**

**A**

**Address 7- 0**

**A**

**DATA**

**A**

**DATA**

**A**

**P**

**The upper 2 bits**

**(**

**of the address)**

**(**

**Write**

**)**

**The lower 8 bits of**

**the address**

**S**

**1**

**1 1 1 0 X X**

**1**

**A**

**(**

**Read**

**)**

**Receiver**

Режим передачи данных мастером

:

Внутренний сдвиговый регистр ведущего устройства отправляет данные из регистра данных на линию SDA. Когда ведущее устройство получает подтверждение (ACK), в регистре состояния 1 (R16\_I2Cx\_STAR1) устанавливается бит TxE, и также генерируется прерывание, если установлены биты ITEVTEN и ITBUFEN. Запись данных в регистр данных очистит бит TxE.

Если бит TxE установлен и новые данные не были записаны в регистр данных до отправки последних данных, то будет установлен бит BTF, и SCL останется низким до тех пор, пока он не будет очищен. Запись данных в регистр данных после чтения R16\_I2Cx\_STAR1 очистит бит BTF.

Рисунок 13-4 Диаграмма последовательности передачи данных передатчиком мастера

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | S |  | Address | A |  | | Data1 | A | Data2 | | A | …… | | DataN | A |  | P | |  | EVT5 |  | | EVT6 | EVT8\_1 | EVT8 |  | EVT8 |  | | EVT8 |  | | | EVT8\_2 |  |   7-bit master send  10-bit master send   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | S |  | Frame header | A |  | Address | A |  | | Data1 | A | …… | | DataN | A |  | P | |  | EVT5 |  | | EVT9 |  | | EVT6 | EVT8\_1 | EVT8 |  | EVT8 |  | | | EVT8\_2 |  | |

Описание:

S=Старт (условие старта), Sr=повторный старт, P=Стоп (условие остановки), A=ответ, NA=отсутствие ответа, **EVTx**=событие (прерывание генерируется, когда **ITEVFEN**=1)

EVT5: **SB**=1, чтение **SR1**, а затем запись адреса в регистр **DR** очищает событие.

EVT6; **ADDR**=1, сначала читаем **SR1**, затем читаем **SR2**, чтобы очистить событие.

EVT81: **TxE**=1, сдвиговый регистр пуст, регистр данных пуст, запись в регистр **DR**.

EVT8: **TxE**=1, сдвиговый регистр не пустой, регистр данных пустой, запись в **DR** регистр очистит событие.

EVT82: **TxE**=1, **BTF**=1, запрос на установку бита остановки. Биты **TxE** и **BTF** аппаратно очищаются при генерации условия остановки.

EVT9: **ADDR10**=1, чтение **SR1** и последующая запись в **DR** очистят событие.

*Примечание:*

*1: События EVT5, EVT6, EVT9, EVT8\_1 и EVT8\_2 удлиняют низкий уровень SCL до конца соответствующей программной последовательности.*

*2: Программная последовательность EVT8 должна быть завершена до окончания текущей передачи байта.*

Режим приема мастером:

Модуль I2C будет принимать данные с линии SDA и записывать их в регистр данных через сдвиговый регистр. После каждого байта, если установлен бит **ACK**, модуль I2C отправит ответ с низким уровнем, и будет установлен бит **RxNE**, и, если установлены биты **ITEVTEN** и **ITBUFEN**, будет сгенерировано прерывание. Если установлен бит **RxNE** и исходные данные не прочитаны до получения новых данных, будет установлен бит **BTF**, и SCL останется низким до тех пор, пока **BTF** не будет очищен. Чтение **R16\_I2Cx\_STAR1**, а затем чтение регистра данных очистит бит **BTF**.

Figure 13-5 Receiver transmission sequence diagram

S

Address

A

EVT5

EVT6

EVT6\_1

Data1

A

（

1

）

Data2

A

EVT7

EVT7

DataN

NA

P

EVT7

S

Frame

header

A

Address

A

EVT6

Sr

Frame

header

A

EVT6

EVT6\_1

EVT7

Data1

A

（

1

）

EVT7

DataN

NA

P

EVT7

7-

bit master reception

Description: S=Start (start condition), Sr=repeated start condition, P=Stop (stop condition), A=response, NA=non-response,

EVTx=event (interrupt generated when ITEVFEN=1)

EVT5: SB=1, reading SR1 and then writing the address to DR register will clear the event.

EVT6: ADDR=1,reading SR1 and then reading SR2 will erase this event. In 10-bit master receive mode, START=1 of CR2 should be

set after this event.

EVT6\_1: There is no corresponding event flag and it is only suitable for receiving 1 byte. Exactly after EVT6 (i.e. after ADDR

is cleared), the response and stop condition generation bits should be cleared.

EVT7: RxNE=1, read DR register to clear the event.

EVT7\_1: RxNE=1, read the DR register to clear this event. Set ACK=0 and STOP request.

EVT9: ADDR10=1, reading SR1 and then writing to DR register will clear this event.

bit master reception

10-

……

……

EVT7\_1

EVT5

EVT9

EVT5

Data2

A

EVT7\_1

Описание:

S=Старт (условие старта), Sr=повторный старт, P=Стоп (условие остановки), A=ответ, NA=отсутствие ответа, **EVTx**=событие (прерывание генерируется, когда **ITEVFEN**=1)

EVT5: **SB**=1, чтение **SR1**, а затем запись адреса в регистр **DR** очистит событие.

EVT6: **ADDR**=1, сначала читаем **SR1**, затем читаем **SR2**, чтобы стереть это событие. В режиме приема мастером с 10-битным адресом после этого события следует установить START=1 в **CR2**.

EVT61: Нет соответствующего флага события, и оно подходит только для приема 1 байта. Сразу после EVT6 (то есть после очистки **ADDR**), биты генерации условий ответа и остановки должны быть очищены.

EVT7: **RxNE**=1, читайте регистр **DR**, чтобы очистить событие.

EVT71: **RxNE**=1, прочтите регистр **DR** для очистки этого события. Установить **ACK**=0 и запрос на остановку.

EVT9: **ADDR10**=1, чтение **SR1** и последующая запись в регистр **DR** удаляют это событие.

Когда ведущее устройство заканчивает отправку данных, оно активно отправляет событие окончания, то есть устанавливает бит STOP, и I2C переключается в режим слейва. В режиме приема ведущему устройству необходимо отправить NAK на позицию ответа последнего бита данных, а после получения NACK ведомое устройство освобождает контроль над линиями SCL и SDA; ведущее устройство затем может отправить условие остановки/повторного старта. Обратите внимание, что модуль I2C автоматически переключается в режим ведомого устройства после генерации условия остановки.

## 13.4 Режим слейва (Slave Mode)

Когда модуль I2C находится в режиме слейва, он распознает свой собственный адрес и широковещательный вызов. Программное обеспечение может контролировать, включена или отключена функция распознавания широковещательного вызова. Как только обнаружено событие начала, модуль I2C сравнивает данные SDA через сдвиговый регистр со своим собственным адресом (количество битов зависит от ENDUAL и ADDMODE) или широковещательным адресом (при установленной ENGC). Если обнаружено несоответствие, оно будет проигнорировано до появления нового события начала. Если совпадает с заголовком, генерируется сигнал ACK, и ожидается адрес второго байта; если адрес второго байта также совпадает или совпадает полный сегментный адрес в случае 7-битного адреса, то: сначала генерируется ответ ACK; устанавливается бит ADDR, и если бит ITEVTEN уже установлен, генерируются соответствующие пререрывание; если используется режим двойного адреса (установлен бит ENDUAL), необходимо также прочитать бит DUALF, чтобы определить, какой адрес вызывает хоста. По умолчанию используется режим слейвом приеме. Если последний бит полученной заголовочной последовательности равен 1, или последний бит 7-битового адреса равен 1 (в зависимости от того, была получена заголовочная последовательность впервые или обычный 7-битный адрес), модуль I2C перейдёт в режим передачи, а бит TRA указывает, находится ли он сейчас в режиме приема или передачи.

Режим передачи слейвом:

После очистки бита ADDR модуль I2C отправляет байты из регистра данных на линию SDA через сдвиговый регистр. После получения подтверждения ACK, устанавливается бит TxE, и генерируется прерывание, если установлены биты ITEVTEN и ITBUFEN. Если установлен бит TxE и новые данные не были записаны в регистр данных до окончания отправки следующих данных, будет установлен бит BTF. SCL остаётся низким до тех пор, пока BTF не будет очищен. Чтение регистра состояния 1 (R16\_I2Cx\_STAR1) и последующая запись данных в регистр данных очистит бит BTF.

Рисунок 13-6 Диаграмма последовательности передачи данных передатчика слейва

S

Address

A

EVT1

EVT3\_1

EVT3

Data1

A

Data2

A

EVT3

EVT3

DataN

NA

P

EVT3\_2

S

Frame

header

A

Address

A

EVT1

Sr

Frame

header

A

EVT1

EVT3\_1

EVT3

Data1

A

EVT3

DataN

NA

P

EVT3\_2

bit slave send

7-

bit slave send

10-

……

……

Описание:

S=Старт (условие старта), Sr=повторный старт, P=Остановка (условие остановки), A=Ответ, NA=Отказ, **EVTx**=Событие (прерывание генерируется, когда **ITEVFEN**=1)

EVT1; **ADDR**=1, чтение **SR1**, а затем чтение **SR2** устранит событие.

EVT31: **TxE**=1, сдвиговый регистр пуст, регистр данных пуст, напишите **DR**.

EVT3: **TxE**=1, регистр сдвига не пуст, регистр данных пусто, запись **DR** очистит событие.

EVT32: **AF**=1, записать '0' в бит **AF** регистра **SR1** для очистки бита **AF**.

*Примечание:*

*1: События EVT1 и EVT3\_1 продлевают состояние низкого уровня SCL до завершения соответствующей программной последовательности.*

*2: Программная последовательность EVT3 должна быть завершена до завершения текущего байта передачи.*

Режим приема слейвом:

После очистки бита **ADDR** модуль I2C сохраняет данные с SDA в регистр данных через сдвиговый регистр. После получения каждого байта модуль I2C устанавливает бит **ACK** и устанавливает бит **RxNE**, и генерирует прерывание, если установлены биты **ITEVTEN** и **ITBUFEN**. Если установлен бит **RxNE** и старые данные не были прочитаны до получения новых данных, то устанавливается бит **BTF**. SCL остается на низком уровне до тех пор, пока бит **BTF** не очищен. Чтение регистра состояния 1 (**R16\_I2Cx\_STAR1**) и чтение данных в регистре данных очистит бит **BTF**.

Рисунок 13-7 Диаграмма последовательности передачи данных приемника

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7-bit slave reception   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | S | Address | A |  | Data1 | |  | A | Data2 | | A |  | | 10-bit slave rece | | | EVT1 |  | |  | | EVT2 |  |  | EVT2 | | ption |  |  | | S | Frame header | A | Address | | A | Data1 | | A | |  | | | | | | EVT1 | |  | |  | EVT2 | | DataN  A  P  EVT2  DataN  A  P  EVT4  ……  EVT2  EVT4  …… |   Description: S=Start (start condition), Sr=repeated start condition, P=Stop (stop condition), A=response, NA=nonresponse, EVTx=event (interrupt generated when ITEVFEN=1)  EVT1;ADDR=1,reading SR1 then SR2 will erase the event.  EVT2: RxNE=1, reading DR will clear the event.  EVT4: STOPF=1,reading SR1 and then writing CR1 register will clear the event.  *Note: 1: EVT1 event elongates SCL low until the end of the corresponding software sequence.*  *2: The software sequence of EVT2 must be completed before the end of the current byte transfer.* |

Описание:

S=Старт (условие старта), Sr=Повторный старт, P=Стоп (условие остановки), A=Ответ, NA=Отказ, EVTx=Событие (прерывание генерируется, когда ITEVFEN=1)

EVT1; ADDR=1, чтение SR1, а затем чтение SR2 удаляет событие.

EVT2: RxNE=1, чтении DR очищает событие.

EVT4: STOPF=1, чтению SR1 и последующей записи в регистр CR1 очистит событие.

*Примечание:*

*1: Событие EVT1 увеличивает длительность низкого уровня SCL до завершения соответствующей программной последовательности.*

*2: Программная последовательность EVT2 должна быть завершена до завершения текущего байта передачи.*

Ведущее устройство сгенерирует условие остановки после передачи последнего байта данных. Когда модуль I2C обнаружит событие остановки, он установит бит STOPF, и если бит ITEVFEN установлен, он также сгенерирует прерывание. Пользователю необходимо прочитать регистр состояния (например, R16\_I2Cx\_STAR1) и затем записать в регистр управления (например, сбросить слово управления SWRST) для его очистки. (См. EVT4 на приведенном выше рисунке).

## 13.5 Условия возникновения ошибок

### 13.5.1 Ошибка шины (BERR)

Ошибка шины возникает, когда модуль I2C обнаруживает внешние события начала или остановки во время передачи адреса или данных. Когда возникает ошибка шины, устанавливается бит **BERR**, и генерируется прерывание, если установлен бит **ITERREN**. В режиме слейва данные отбрасываются, и оборудование освобождает шину. Если это сигнал старта, оборудование считает его сигналом перезапуска и ожидает сигнала адреса или остановки. Если сигнал остановки, оборудование продолжает функционировать как обычно, ожидая нормального сигнала остановки. В мастерском режиме, оборудование не отпускает шину, и пользовательский код решает, продолжить передачу или прервать её.

### 13.5.2 Ошибка подтверждения (AF)

Ошибка ответа возникает, когда модуль I2C обнаруживает байт данных, но не получает подтверждения. Когда возникает ошибка ответа:

* Устанавливается бит AF.
* Генерируется прерывание, если установлен бит ITERREN.
* При возникновении ошибки AF, оборудование должно освободить шину, если модуль I2C работает в режиме слейва, и программное обеспечение должно сгенерировать событие остановки, если оно работает в мастер-режиме.

### 13.5.3 Потеря арбитража (ARLO)

Ошибка потери арбитража возникает, когда модуль I2C обнаруживает потерю арбитража. Когда возникает ошибка потери арбитража:

* Устанавливается бит ARLO.
* Генерируется прерывание, если установлен бит ITERREN.
* Модуль I2C переходит в режим слейва и перестает отвечать на запросы, направленные на его адрес, если хозяин не инициировал новое событие старта.

### 13.5.4 Ошибка переполнения/недостаточного заполнения (OVR)

### Ошибка переполнения

В режиме слейва, если расширение тактовой частоты отключено и модуль I2C получает данные, возникает ошибка переполнения, если байт данных был принят, но последние полученные данные еще не были прочитаны. При возникновении ошибки переполнения последнее принятый байт отбрасывается, и отправляющий должен повторить передачу последнего байта.

### Ошибка недонаполнения

В режиме слейва, если запрещено удлинение тактовой частоты и модуль I2C отправляет данные, возникает ошибка недонаполнения, если новые данные не были записаны в регистр данных до наступления следующего байтa clock edge. В случае ошибки недонаполненности, данные из предыдущего регистра данных отправляются дважды, и если возникает ошибка недоучёта, получатель должен отбрасывать повторные данные. Чтобы избежать ошибки недонаполнения, модуль I2C должен записать данные в регистр данных перед появлением первого растянутого края следующего байта.

## 13.6 Расширение тактовых импульсов

Если расширение тактовых импульсов отключено, возникают риски ошибок переполнения/недостаточного заполнения. Однако, если расширение тактовой частоты включено:

* В режиме передачи, если установлен бит **TxE** и бит **BTF**, сигнал SCL будет оставаться на низким уровне, ожидаемый пользователем для чтения регистра статуса и записи данных в регистр данных.
* В режиме приема, если установлены биты **RxNE** и **BTF**, SCL останется на низким уровне после получения данных, ожидающегося чтения регистра статусов и данных пользователем. Активация расширения тактовых импульсов помогает предотвратить ошибки переполнения/недостаточно заполненного состояния.

Видно, что включение расширения тактовых импульсов помогает избежать ошибок переполнения/недостаточного заполнения.

## 13.7 Шина SMBus

## SMBus также является двухпроводным интерфейсом, который обычно используется между системой и управлением питанием. У SMBus и I2C много общего, например, SMBus использует тот же режим адресации с 7 битами, что и I2C, а следующие функции являются общими для SMBus и I2C:

## Режим связи "ведущий-ведомый", при котором хост предоставляет тактовый сигнал и поддерживает несколько ведущих устройств и подчиненных.

## Двухпроводная архитектура связи, с опциональной линией предупреждения для SMBus.

## Оба поддерживают формат адреса длиной 7 битов.

Начало формы

Также существуют различия между SMBus и I2C:

1. I2C поддерживает скорости до 400 кГц, тогда как SMBus поддерживает до 100 кГц; кроме того, у SMBus есть ограничение минимальной скорости в 10 кГц.
2. При низком уровне тактового сигнала SMBus более чем на 35 мс будет сообщаться о тайм-ауте, однако такого ограничения нет у I2C.
3. U SMBus фиксированный логический уровень, тогда как у I2C он зависит от напряжения питания (VDD).
4. U SMBus имеется протокол шины, которого нет у I2C.

Кроме того, SMBus включает идентификацию устройств, протоколы разрешения адресов, уникальные идентификаторы устройств, напоминания SMBus и различные шинойные протоколы, описанные в спецификации версии 2.0 SMBus. При использовании SMBus необходимо установить только бит SMBus в регистре управления, а биты SMBTYPE и ENAARP должны быть настроены по мере необходимости.

## 13.8 Прерывания

Каждый модуль I2C имеет два вектора прерываний: прерывания событий и прерывания ошибок. Оба типа прерываний поддерживают источники прерываний, указанные на Рисунке 13-4.

Рисунок 13-4 Запрос прерывания I2C



## 13.9 Прямой доступ к памяти (DMA)

DMA может использоваться для отправки и получения больших объемов данных. Бит **ITBUFEN** в регистре управления не должен быть установлен при использовании DMA.

Начало формы

* Передача с использованием DMA

РНачало формы

ежим DMA активируется путем установки бита **DMAEN** в регистре **CTLR2**. Как только бит **TxE** установлен, данные будут загружены через DMA из заданной памяти в регистр данных I2C. Для выделения каналов для I2C требуются следующие настройки:

1. Установите адрес регистра **I2Cx\_DATAR** в регистр **DMA\_PADDRx**, а адрес памяти – в регистр **DMA\_MADDRx** таким образом, чтобы после каждого события TxE данные отправлялись из памяти в регистр **I2Cx\_DATAR**.
2. Задайте необходимое количество байтов для передачи в регистре **DMA\_CNTRx**. Это значение будет уменьшаться после каждого события TxE.
3. Настройте приоритет канала с помощью битов **PL**[0:1] в регистре **DMA\_CFGRx**.
4. Установите бит **DIR** в регистре **DMA\_CFGRx** и, в зависимости от требований приложения, настройте его для выдачи запроса на прерывание, когда вся передача будет выполнена наполовину или полностью.
5. Активируйте канал, установив бит **EN** в регистре **DMA\_CFGRx**.

Начало формы

Когда количество байтов передачи данных, установленных в контроллере DMA, завершено, контроллер DMA отправляет сигнал окончания передачи (EOT/EOT\_1) на интерфейс I2C. Если разрешено прерывание, будет сгенерировано прерывание DMA.

* Прием с использованием DMA

Режим приема DMA может быть выполнен после установки DMAEN в регистре CTLR2. При использовании режима приема DMA, DMA передает данные из регистра данных в предварительно заданную область памяти. Для выделения каналов для I2C необходимы следующие шаги:

1. Установите адрес регистра I2Cx\_DATAR в регистр DMA\_PADDRx, а адрес памяти – в регистр DMA\_MADDRx таким образом, чтобы после каждого события RxNE данные записывались в память из регистра I2Cx\_DATAR.
2. Задайте необходимое количество байтов для передачи в регистре DMA\_CNTRx. Это значение будет уменьшаться после каждого события RxNE.
3. Настройте приоритет канала с помощью битов PL[0:1] в регистре DMA\_CFGRx.
4. Сбросьте бит DIR в регистре DMA\_CFGRx, и в зависимости от требований приложения можно настроить выдачу запроса на прерывание, когда передача данных будет выполнена наполовину или полностью.
5. Установите бит EN в регистре DMA\_CFGRx для активации канала.

Начало формы

Когда количество передач данных, установленное в контроллере DMA, завершено, контроллер DMA отправляет сигнал окончания передачи (EOT/EOT\_1) на интерфейс I2C. Если прерывание разрешено, будет сгенерировано прерывание DMA.

## 13.10 Проверка ошибок пакета

Контрольная сумма пакета (PEC) — это дополнительный шаг проверки с использованием CRC8 для повышения надежности передачи, рассчитываемый для каждого бита последовательных данных с использованием следующего полинома.

Расчёт PEC активируется битом ENPEC в регистре управления и выполняется для всех информационных байтов, включая адрес и биты чтения/записи. При передаче включение PEC добавляет байт результата расчёта CRC8 после последнего байта данных; в режиме приёма последний байт рассматривается как результат проверки CRC8, и если он не совпадает с результатом внутреннего расчёта, будет отправлен ответ NAK, а в случае основного приёмника, независимо от правильности результата проверки.

## 13.11 Описание регистров

Таблица 13-1 Список регистров, связанных с I2C

### 13.11.1 I2C Control Register 1(I2C1\_CTLR1)

### 13.11.2 I2C Control Register 2(I2C1\_CTLR2)

### 13.11.3 I2C Own Address Register 1(I2C1\_OAR1)

### 13.11.4 I2C Own Address Register 2(I2C1\_OAR2)

### 13.11.5 I2C Data Register (I2C\_DATAR)

### 13.11.6 I2C Status Register 1(I2C\_STAR1)

### 13.11.7 I2C Status Register 2(I2C\_STAR2)

### 13.11.8 I2C Clock Register (I2C1\_CKCFGR)

# Chapter 14 Serial Peripheral Interface (SPI)

SPI supports data interaction in a 3-wire synchronous serial mode, plus a chip selector line to support hardware switching between Master and Slave modes, and supports communication on a single data line.

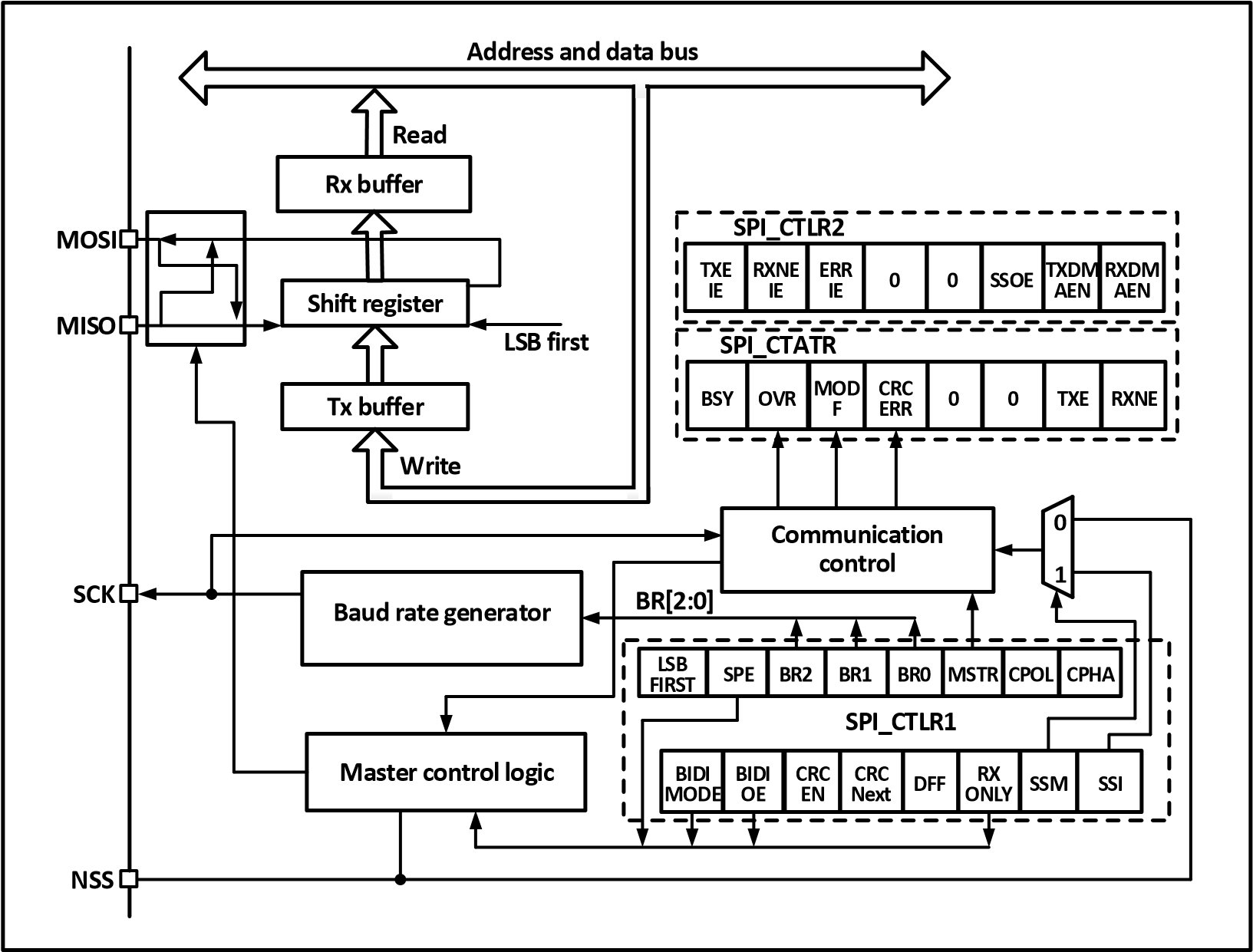
## 14.1 Main Features

* Support full-duplex synchronous serial mode
* Support single-line half-duplex mode
* Support Master mode and Slave mode, Multi-slave mode
* Support 8-bit or 16-bit data structures
* Maximum clock frequency supports up to half of FHCLK
* Data order supports MSB or LSB first
* Support hardware or software control of NSS pins
* Hardware CRC checksum support for sending and receiving
* Transceiver buffers support DMA transfers
* Support modification of clock phase and polarity

## 14.2 Function Description

### 14.2.1 Overview

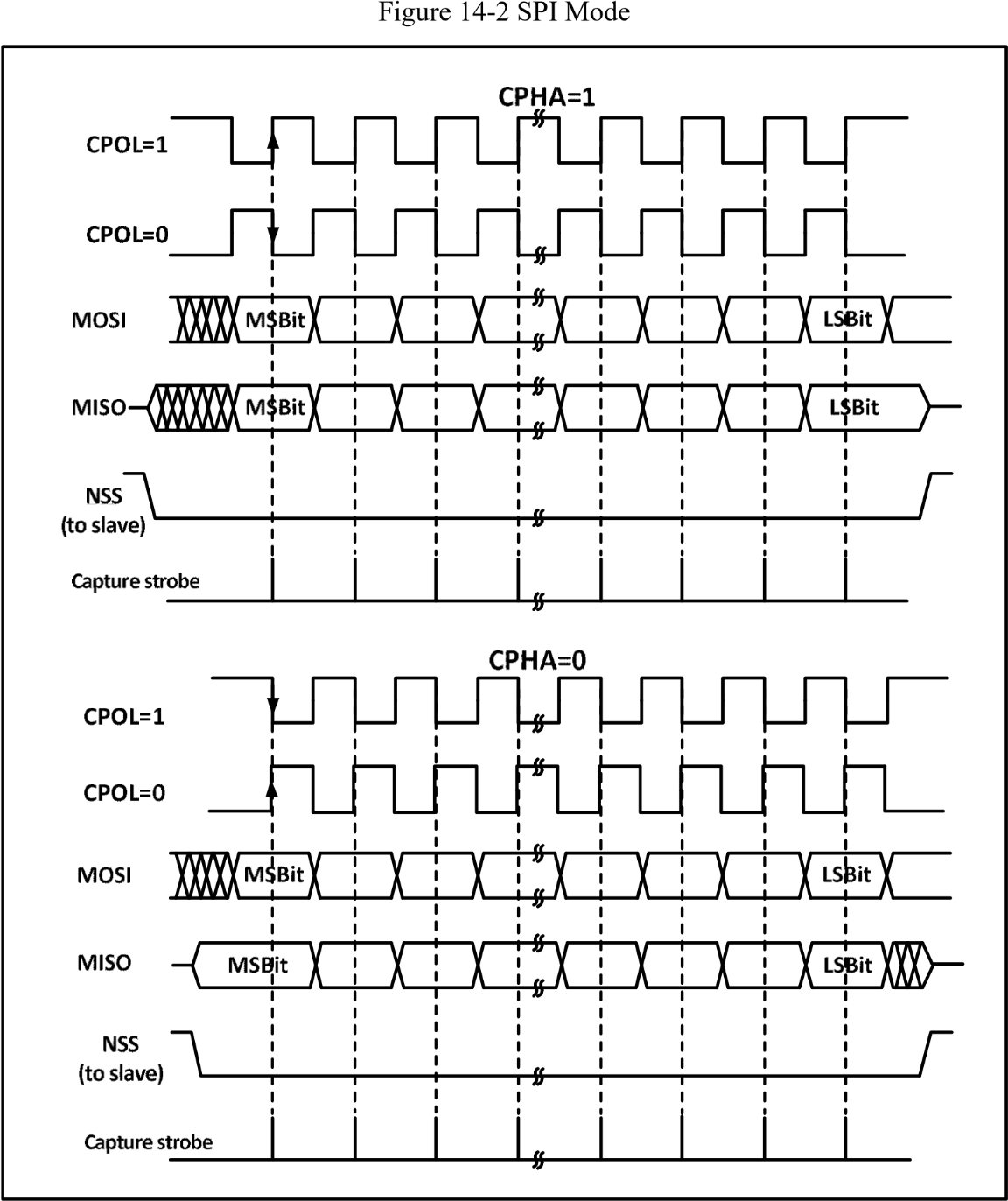
Figure 14-1 SPI structure block diagram



As can be seen from Figure 14-1, the four main SPI-related pins are MISO, M0SI, SCK and NSS. The MISO pin is the data input pin when the SPI module is operating in Master mode and the data output pin when it is operating in Slave mode. the MOSI pin is the data output pin when it is operating in Master mode and the data input pin when it is operating in Slave mode. the SCK is the clock pin, the clock signal is always output by the host and the slave receives the clock signal and synchronizes the data sending and receiving. the NSS pin is the chip select pin with the following usage.

1. NSS controlled by software: When SSM is set and the internal NSS signal is output high or low as determined by SSI, this case is generally used in SPI Master mode.
2. NSS is controlled by hardware: When the NSS output is enabled, i.e., when SSOE is set, the NSS pin will be actively pulled down when the SPI host sends outputs outward, and if it does not succeed in pulling down the NSS pin, which indicates that there is another master device on the main line that is communicating, a hardware error will be generated; SSOE is not set, it can be used in multi-master mode, and if it is pulled down it will be forced to enter the slave mode, and the MSTR bit will be cleared automatically.

CPHA is set to indicate that the module samples data on the second edge of the clock and the data is latched, while CPHA is not set to indicate that the SPI module samples data on the first edge of the clock and the data is latched, and CPOL indicates whether the clock is held high or low when there is no data. See Figure 14-2 below for details.



The host and device need to be set to the same SPI mode, and the SPE bit needs to be cleared before configuring the SPI mode. the DEF bit determines whether the individual data length of the SP is 8 bits or 16 bits. LSBFIRST controls whether a single data word is preceded by the high bit or the low bit.

### 14.2.2 Master Mode

The serial clock is generated by SCK when the SPI module is operating in master mode. The following steps are performed to configure into master mode.

Configure the BR[2:0] field of the control register to determine the clock.

Configure the CPOL and CPHA bits to determine the SPI mode.

Configure DEF to determine the data word length.

Configure LSBFIRST to determine the frame format;

Configure the NSS pin, for example by setting the SSOE bit and letting the hardware set the NSS. it is also possible to set the SSM bit and set the SSI bit high.

To set the MSTR bit and the SPE bit, you need to make sure that the NSS is already high at this time.

When you need to transmit data you just need to write the data to be transmitted to the data register. SPI will transmit the data from the transmit buffer to the shift register in parallel and then transmit the data from the shift register according to the setting of LSBFIRST. When the data is already in the shift register, the TXE flag will be set. If the TXEIE is already set, then an interrupt will be generated. If the TXE flag position bit needs to be filled with data into the data register to maintain the complete data flow.

When the receiver receives data, when the last sample clock edge of the data word comes, the data is transferred from the shift register to the receive buffer in parallel, the RXNE bit is set, and an interrupt is generated if the RXNEIE bit was previously set. At this time, the data register should be read as soon as possible to take away the data.

### 14.2.3 Slave Mode

When the SPI module is operating in slave mode, SCK is used to receive the clock from the host and its own baud rate setting is invalid. To configure into slave mode, proceed as follows.

Configure the DEF bit to set the data bit length.

Configure the CPOL and CPHA bits to match the host mode. Configure LSBFIRST to match the host data frame format;

The NSS pin needs to be held low in hardware management mode, if NSS is set to software management (SSM set), then keep SSI unset.

Clear the MSTR bit and set the SPE bit to enable SPI mode. In transmitting, when the first slave receive sample edge appears in SCK, the slave starts to transmit. The process of sending is to move the data in the transmit buffer t the transmit shift register. When the data in the transmit buffer is moved to the shift register, the TXE flag will be set, and if the TXEIE bit was set before, then an interrupt will be generated.

During reception, after the last clock sample edge, the RXNE bit is set, the bytes received by the shift register are transferred to the receive buffer, and the read operation of the read data register can obtain the data in the receive buffer. If RXNEIE is set before RXNE is set, then an interrupt is generated.

### 14.2.4 Simplex Mode

The SPI interface can operate in half-duplex mode, where the master device uses the MOSI pin and the slave device uses the MISO pin for communication. When using half-duplex communication, you need to set BIDIMODE and use BIDIOE to control the transmission direction.

Setting the RXONLY bit in normal full-duplex mode sets the SPI module to receive-only simplex mode, releasing a data pin after RXONLY is set. The SPI can also be set to transmit only mode by ignoring the received data.

### 14.2.5 CRC

The SPI module uses CRC checksum to ensure the reliability of full-duplex communication, and separate CRC calculators are used for data sending and receiving. the polynomial for CRC calculation is determined by the polynomial register, and different calculations are used for 8-bit data width and 16-bit data width, respectively. Setting the CRCEN bit will enable CRC checksum and at the same time will reset the CRC calculator. After the last data byte is sent, setting the CRCNEXT bit will send the TXCRCR calculator calculation after the current byte is sent, while the CRCERR bit will be set if the last received receive shift register value does not match the locally calculated RXCRCR calculation. Using the CRC checksum requires setting the polynomial calculator and setting the CRCEN bit when configuring the SPI operating mode, and setting the CRCNEXT bit on the last word or half-word to send the CRC and perform the receive CRC checksum. Note that the polynomial for the CRC calculation should be unified for both sending and receiving.

### 14.2.6 DMA

The SPI module supports the use of DMA to speed up data communication, either by using DMA to fill the transmit buffer or by using DMA to pick up data from the receive buffer in a timely manner. DMA will pick up or send data in a timely manner using RXNE and TXE as signals. DMA can also operate in simplex or CRC mode. **14.2.7 Errors**

* Master mode fault (MODF)

When the SPI is operating in NSS pin hardware management mode, an external pull-down of the NSS pin occurs; or in NSS pin software management mode, the SSI bit is cleared; or the SPE bit is cleared, causing the SPI to be shut down; or the MSTR bit is cleared and the SPI enters slave mode. If the ERRIE bit is already set, an interrupt is also generated. Steps to clear the MODF bit: First perform a read or write operation to R16\_SPI1\_STATR, and then write R16\_SPI1\_CTLR1.

* Overrun condition

If the host sends data and there is unread data in the receive buffer of the slave device, an overflow error occurs, the OVR bit is set, and an interrupt is also generated if ERRIE is set. Sending an overflow error should restart the current transmission. Reading the data register and then reading the status register will eliminate this bit.

* CRC error

When the received CRC word and the value of RXCRCR do not match, a CRC error will be generated and the CRCERR bit will be set.

### 14.2.8 Interrupts

The SPI module supports five interrupt sources, among which the TXE and RXNE events are set when the TXEIE and RXNEIE bits are set respectively. In addition to the above three errors will also generate interrupts, namely MODF, OVR and CRCERR, after enabling the ERRIE bit, these three errors will also generate error interrupts.

## 14.3 Register Description

Table 14-1 SPI-related registers list

Name Access address Description Reset value

R16\_SPI\_CTLR1 0x40013000 SPI Control register1 0x0000

R16\_SPI\_CTLR2 0x40013004 SPI Control register2 0x0000

R16\_SPI\_STATR 0x40013008 SPI Status register 0x0002

R16\_SPI\_DATAR 0x4001300C SPI Data register 0x0000

R16\_SPI\_CRCR 0x40013010 SPI Polynomial register 0x0007

R16\_SPI\_RCRCR 0x40013014 SPI Receive CRC register 0x0000

R16\_SPI\_TCRCR 0x40013018 SPI Transmit CRC register 0x0000

R16\_SPI\_HSCR 0x40013024 SPI High-speed control register 0x0000

### 14.3.1 SPI Control Register 1 (SPI\_CTLR1)

Offset address: 0x00



1: Selection of 1-line bidirectional mode.

0: Selection of 2-line bi-directional mode. Output enable in bidirectional mode bit, used in conjunction with BIDImode.

14 BIDIOE RW 0

1: Enable output, transmit only.

0: Disable output, receive only.

Hardware CRC checksum enable bit, this bit can only be written when SPE is 0. This bit can only

13 CRCEN RW be used in full-duplex mode. 0

1: Initiate CRC calculation.

0: CRC calculation is disabled.

After the next data transfer, send the value of the

CRC register. This should be set immediately after

12 CRCNEXT RW the last data is written to the data register. 0

1: Sending CRC checksum results.

0: Continue to send data from the data register. Data frame format bit, this bit can only be written when SPE is 0.

11 DFF RW 0

1: Sending and receiving using 16-bit data length.

0: Use 8-bit data length for sending and receiving. The receive-only bit in 2-wire mode is used in conjunction with BIDIMODE. Setting this bit 10 RXONLY RW allows the device to receive only and not transmit. 0

1: Receive only, simplex mode.

0: Full-duplex mode.

Software slave management bit, this bit determines whether the level of the NSS pin is

9 SSM RW controlled by hardware or software. 0

1: Software control of the NSS pins.

0: Hardware control NSS pins.

Internal slave select bit, with SSM set, this bit determines the level of the NSS pin.

8 SSI RW 0

1: NSS is high.

0: NSS is low.

Frame format control bit. It is not possible to modify this bit during communication.

7 LSBFIRST RW 1: LSB is transmitted first; 0

0: MSB is transmitted first.

Note: LSB is only supported by SPI as host.

SPI enable bit.

6 SPE RW 1: Enable SPI. 0

0: Disable SPI.

Baud rate setting field, this field cannot be modified during communication.

000: F

[5:3] BR[2:0] RW HCLK /2; 001: FHCLK /4. 0

010: FHCLK /8; 011: FHCLK /16.

100: FHCLK /32; 101: FHCLK /64.

110: FHCLK /128; 111: FHCLK /256.

Master-slave setting bit, this bit cannot be modified during communication.

2 MSTR RW 0b

1: Configured as a master device.

0: Configured as a slave device.

Clock polarity selection bit, this bit cannot be modified during communication.

1 CPOL RW 0

1: SCK is held high in idle state.

0: SCK is held low in idle state.

Clock phase setting bit, this bit cannot be modified

0 CPHA RW during communication. 0

1: Data sampling starts from the second clock

edge.

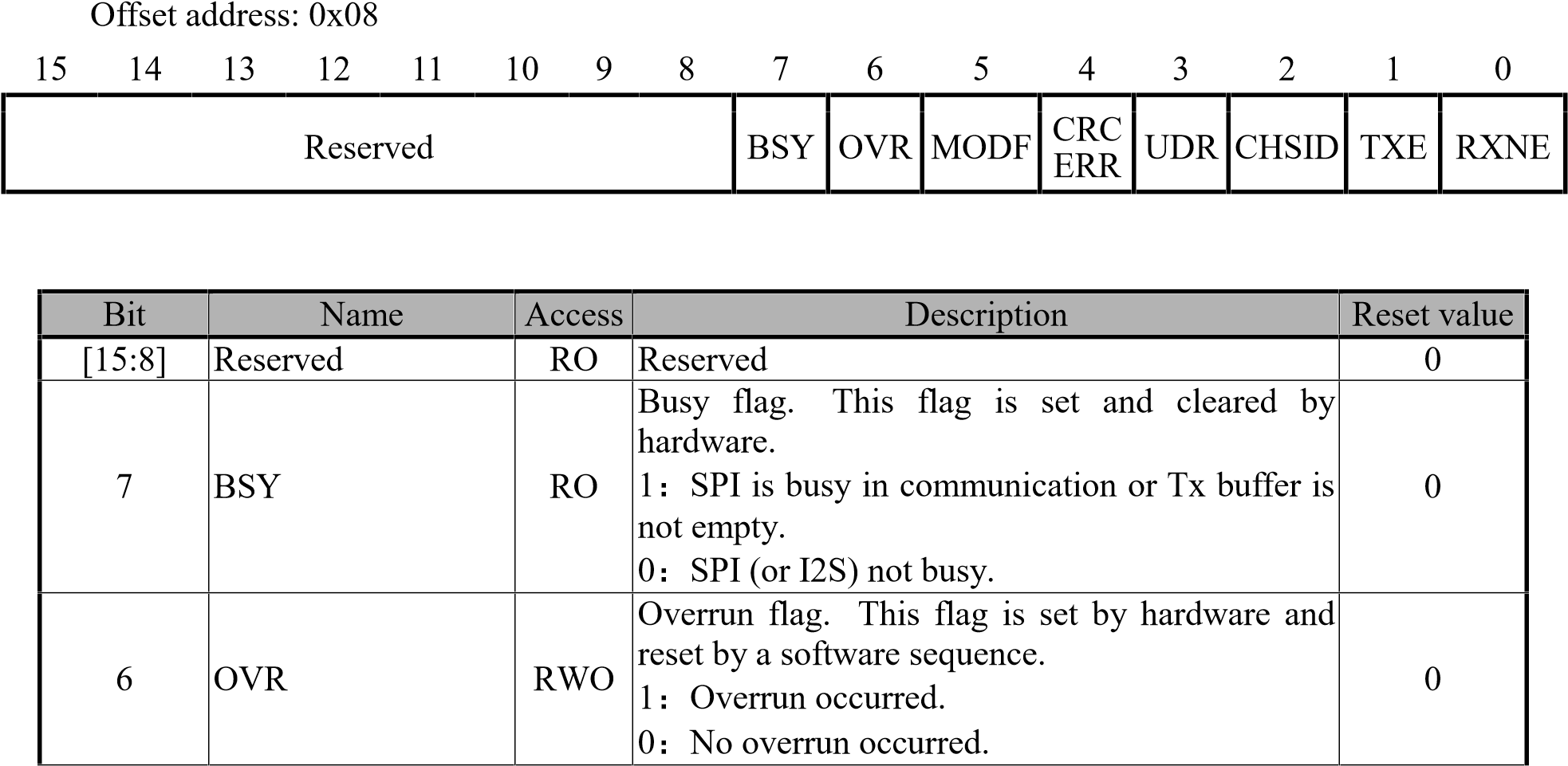
0:

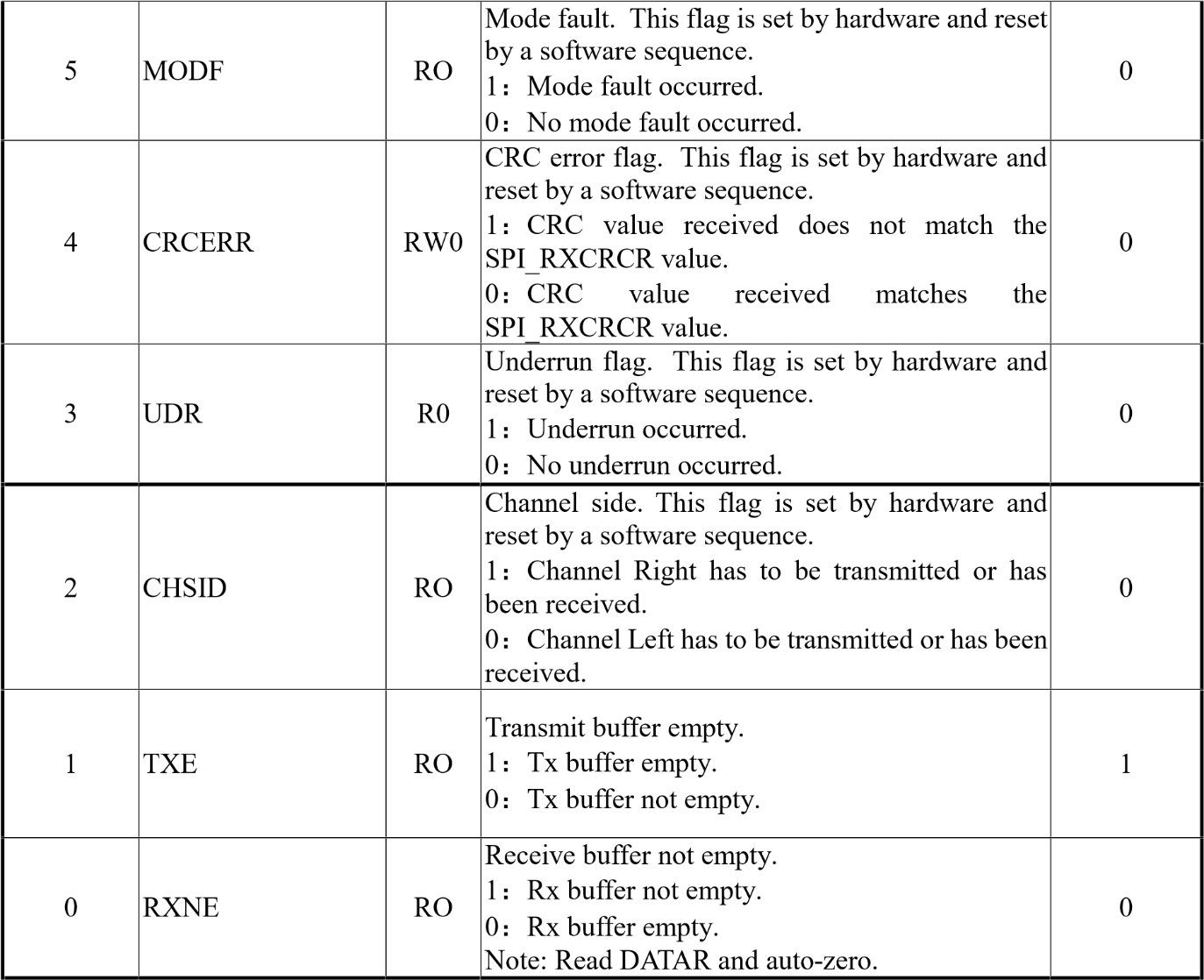
Data sampling starts from the first clock edge.

### 14.3.2 SPI Control Register 2 (SPI\_CTLR2)



### 14.3.3 SPI Status Register (SPI\_STATR)





### 14.3.4 SPI Data Register (SPI\_DATAR)

Offset address: 0x0C

15 14 13 12 11 10 9 8 6 5 4 3 2 1 0

DR[15:0]

Bit Name Access Description Reset value

Data register. The data registers are used to store the received data or pre-store the data to be sent out, so the reading and writing of the data registers actually correspond to the operation of different areas, where the read pairs use the receive buffer and the write pairs correspond to the send buffer.

[15:0] DR[15:0] RW Data can be received and sent in 8 or 16 bits, and 0

it is necessary to determine how many bits of data to use before transmission. When using 8 bits for data transmission, only the lower 8 bits of the data registers are used, and the higher 8 bits are forced to 0 for reception. using a 16-bit data structure causes all 16 bits of the data registers to be used.

7

### 14.3.5 SPI1 Polynomial Register (SPI\_CRCR)

Offset address: 0x10

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

CRCPOLY[15:0]

Bit Name Access Description Reset value

CRC polynomial. This register contains the

[15:0] CRCPOLY[15:0] RW 7h

polynomial for the CRC calculation.

### 14.3.6 SPI1 Receive CRC Register (SPI\_RCRCR)

Offset address: 0x14

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

RXCRC[15:0]

Bit Name Access Description Reset value

Rx CRC. Store the result of the calculated CRC checksum of the received byte. Setting CRCEN resets this register. The calculation method uses

the polynomial used in CRCPOLY. 8-bit mode

[15:0] RXCRC[15:0] RO 0

only the lower 8 bits are involved in the calculation, 16-bit mode all 16 bits are involved in the calculation. It is necessary to read this register when BSY is 0.

### 14.3.7 SPI1 Transmit CRC Register (SPI\_TCRCR)

Offset address: 0x18

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

TXCRC[15:0]

Bit Name Access Description Reset value

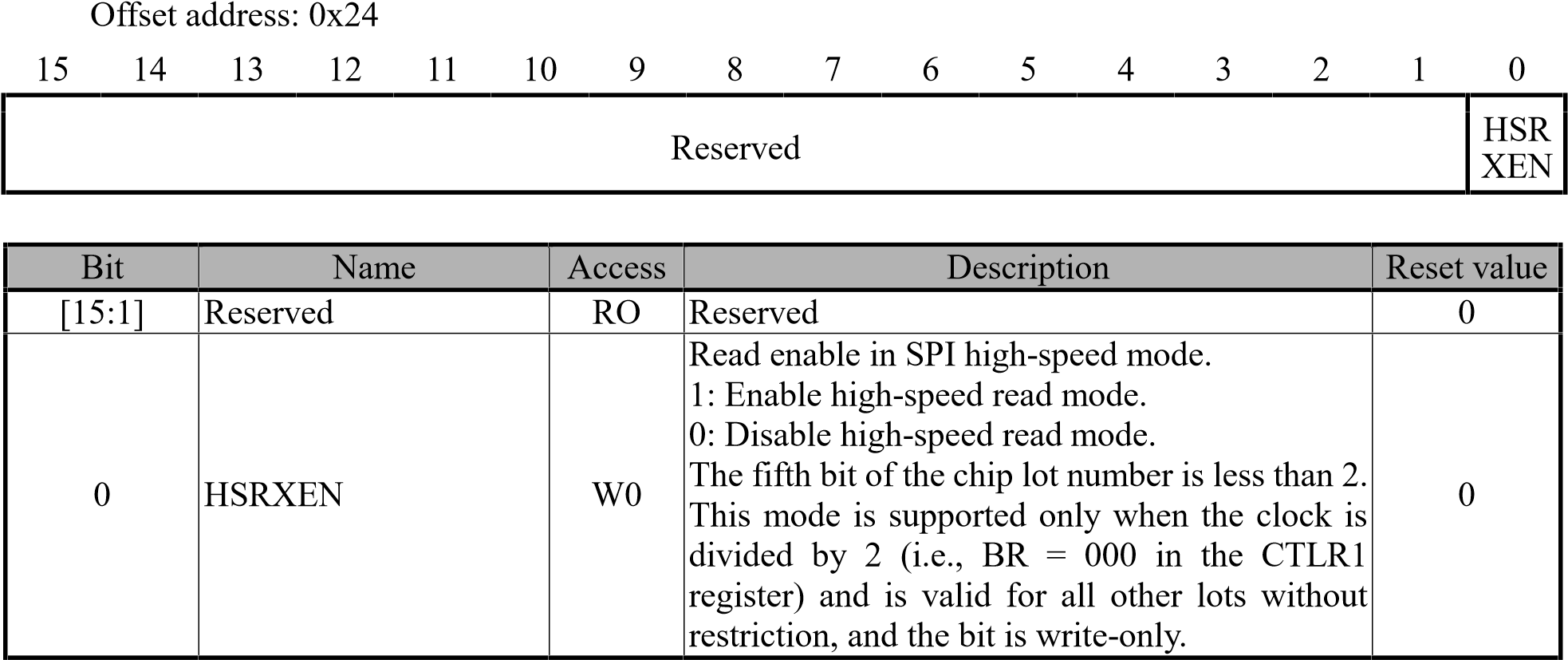
Tx CRC. Store the result of the calculated CRC checksum of the bytes that have been sent out. Setting CRCEN resets this register. The

calculation method uses the polynomial used in

[15:0] TXCRC[15:0] RO 0

CRCPOLY. 8-bit mode only the lower 8 bits are involved in the calculation, while in 16-bit mode all 16 bits are involved. It is necessary to read this register when BSY is 0.

### 14.3.8 SPI High-speed Control Register (SPI\_HSCR)



# Chapter 15 Electronic Signature (ESIG)

The electronic signature contains the chip identification information: the flash memory area capacity and a unique identifier. It is burned into the system storage area of the memory module by the manufacturer at the factory and can be read by SWD (SDI) or application code.

## 15.1 Functional Description

Flash capacity: Indicates the current size of the chip that can be used by user applications.

Unique identification: 96-bit binary code, unique to any microcontroller, the user can only read access cannot be modified. This unique identification information can be used as a microcontroller (product) security password, encryption and decryption keys, product serial numbers, etc., to improve system security mechanisms or to indicate the identity information.

All the above can be read accessed by 8/16/32 bit by the user.

## 15.2 Register Description

Table 15-1 ESIG-related registers list

Name Access Address Description Reset value

R16\_ESIG\_FLACAP 0x1FFFF7E0 Flash capacity register 0xXXXX

R32\_ESIG\_UNIID1 0x1FFFF7E8 UID register 1 0xXXXXXXXX

R32\_ESIG\_UNIID2 0x1FFFF7EC UID register 2 0xXXXXXXXX

R32\_ESIG\_UNIID3 0x1FFFF7F0 UID register 3 0xXXXXXXXX

### 15.2.1 Flash capacity register (ESIG\_FLACAP)

15

14

13

12

11

10

9

8

7

6

5

4

3

2

1

0

F

\_SIZE

[15:0]

Bit

Name

Access

Description

Reset value

[15:0]

F\_SIZE[15:0]

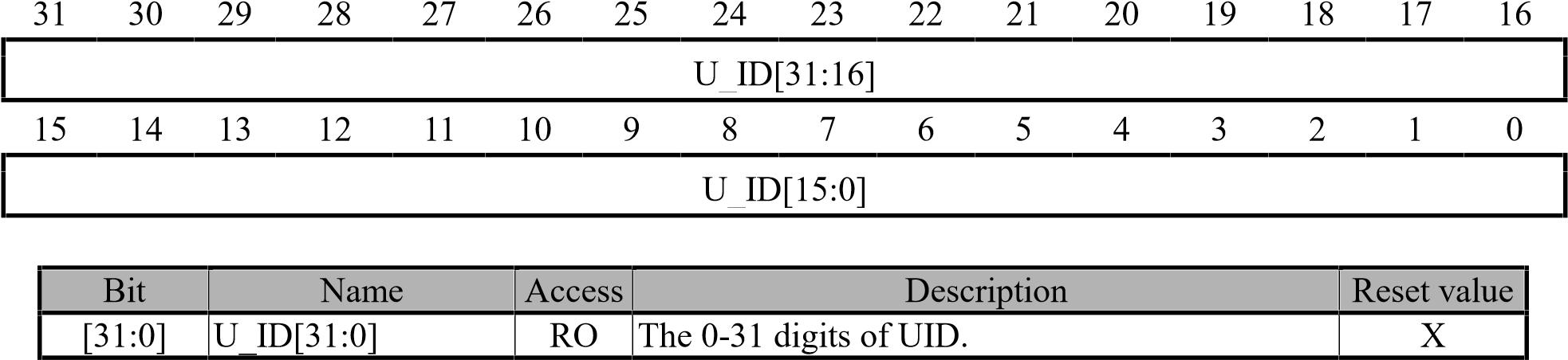
RO

Flash capacity in Kbyte.

Example: 0x0080 = 128 K bytes

X

### 15.2.2 UID Register (ESIG\_UNIID1)



### 15.2.3 UID Register (ESIG\_UNIID2)

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16

U\_ID[63:48]

15

14

13

12

11

10

9

8

7

6

5

4

3

2

1

0

U\_ID[47:32]

Bit Name Access Description Reset value

[31:0] U\_ID[63:32] RO The 32-63 digits of UID. X

### 15.2.4 UID Register (ESIG\_UNIID3)



# Chapter 16 Flash Memory and User Option Bytes

## 16.1 Flash Memory Organization

The internal flash memory of the chip is organized as follows.

Table 16-1 Flash Memory Organization

Block Name Address Range Size(byte)

Page 0 0x0800 0000 – 0x0800 003F 64

Page 1 0x0800 0040 – 0x0800 007F 64

Main Page 2 0x0800 0080 – 0x0800 00BF 64 memory Page 3 0x0800 00C0 – 0x0800 00FF 64

… … …

Page 256 0x0800 3FC0 – 0x0800 3FFF 64

Information Launcher code 0x1FFF F000 – 0x1FFF F77F 2K-128 block User option bytes 0x1FFF F800 – 0x1FFF F83F 64

*Notes: The above main memory area is used for user's application storage and is write-protected in 1K byte (16 pages) units; except for the "vendor configuration word" area which is factory locked and inaccessible to the user, the other areas are user-operable under certain conditions.*

## 16.2 Flash Memory Programming and Security

### 16.2.1 Two Programming/Erasing Methods

* Standard programming: This mode is the default programming mode (compatible mode). In this mode, the CPU performs programming in single 2-byte mode and performs erase and whole erase operation in single 1K byte.
* Fast programming: This method uses page operation (recommended). After a specific sequence of unlocking, it performs a single 64-byte programming and 64-byte erasing, 1K-byte erasing (standard 1K whole chip erasing is also applicable to fast programming).

### 16.2.2 Security - Prevent Illegal Access (read, write, erase)

* Page write protection
* Read protection

When the chip is in the read-protected state.

1. Main memory pages 0-32 (2K bytes) are automatically write-protected state, not controlled by FLASH\_WPR register; unread-protected state, all main memory pages are controlled by FLASH\_WPR register.
2. The system boot code area, SDI mode, and RAM area are not erasable or programmable for main memory, except for whole chip erasure. User-option bytes areas can be erased or programmed. If an attempt is made to unprotect the read (program the user word), the chip will automatically erase the entire user area. *Note: The internal RC oscillator (HSI) must be turned on when performing a program/erase operation of the flash memory.*

## 16.3 Register Description

Table 16-2 FLASH-related registers list

Name

Access

address

Description

Reset value

R32\_FLASH\_ACTLR

0x40022000

Control register

0

x00000000

R32\_FLASH\_KEYR 0x40022004 FPEC key register X

R32\_FLASH\_OBKEYR 0x40022008 OBKEY register X

R32\_FLASH\_STATR 0x4002200C Status register 0x00008000

R32\_FLASH\_CTLR 0x40022010 Configuration register 0x00008080

R32\_FLASH\_ADDR 0x40022014 Address register X

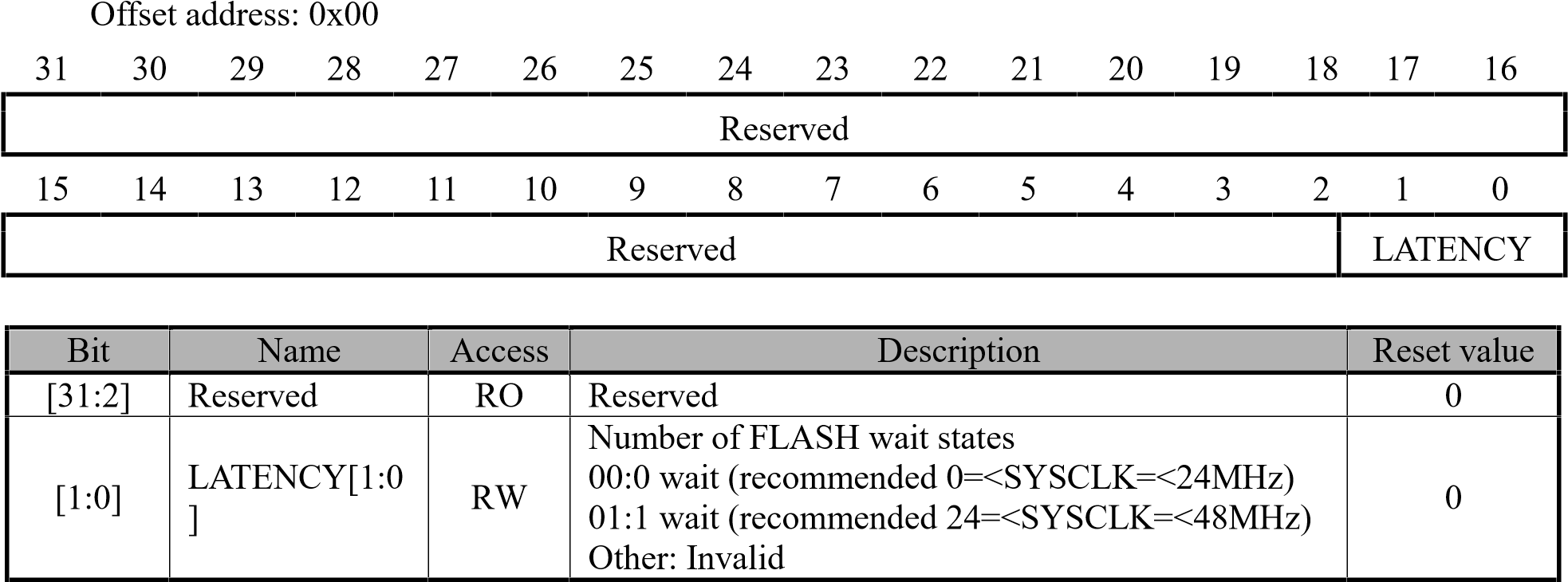
R32\_FLASH\_OBR 0x4002201C Option byte register 0x0XXXXXXX

R32\_FLASH\_WPR 0x40022020 Write protection register 0xFFFFFFF

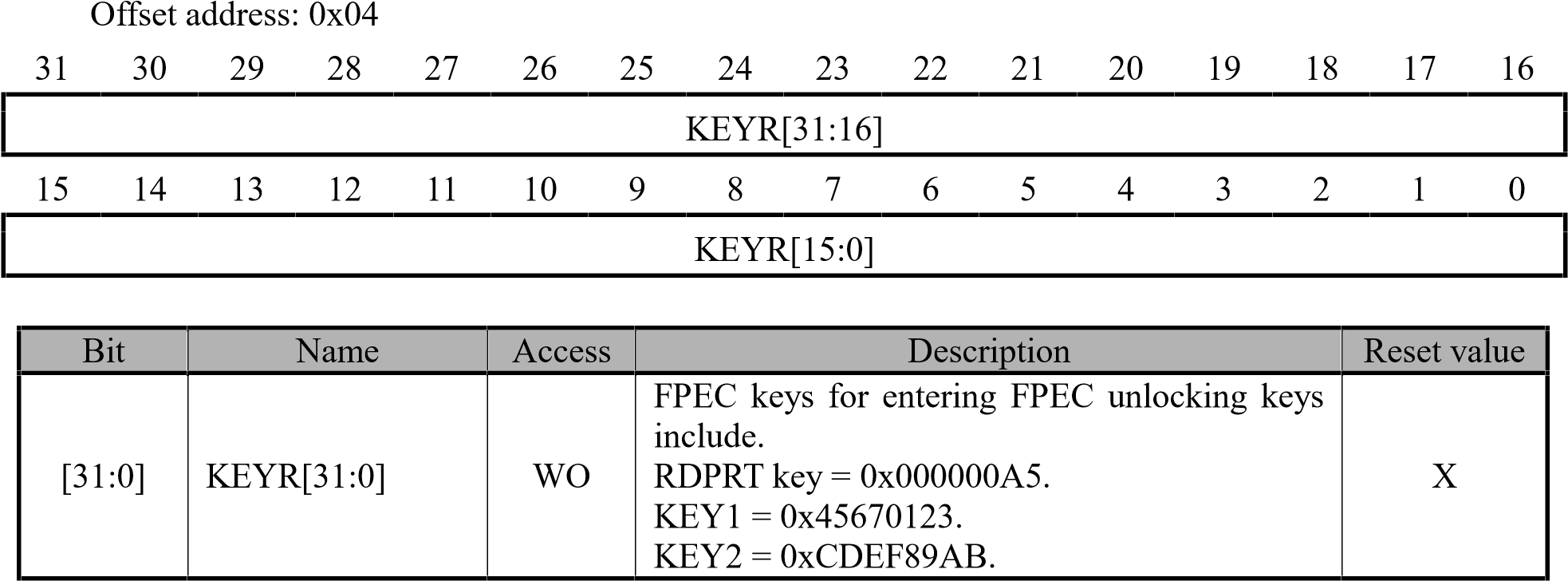
R32\_FLASH\_MODEKEYR 0x40022024 Extended key register X

R32\_FLASH\_BOOT\_MODEKEYR 0x40022028 Unlock BOOT key register X

### 16.3.1 Control Register (FLASH\_ACTLR)



### 16.3.2 FPEC Key Register (FLASH\_KEYR)



### 16.3.3 OBKEY Register (FLASH\_OBKEYR)

Offset address: 0x08

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16

OBKEYR[31:16]

15

14

13

12

11

10

9

8

7

6

5

4

3

2

1

0

OBKEYR[15:0]

Reset

Bit Name Access Description

value

Option bytes key for entering the option bytes

[31:0] OBKEYR[31:0] WO X

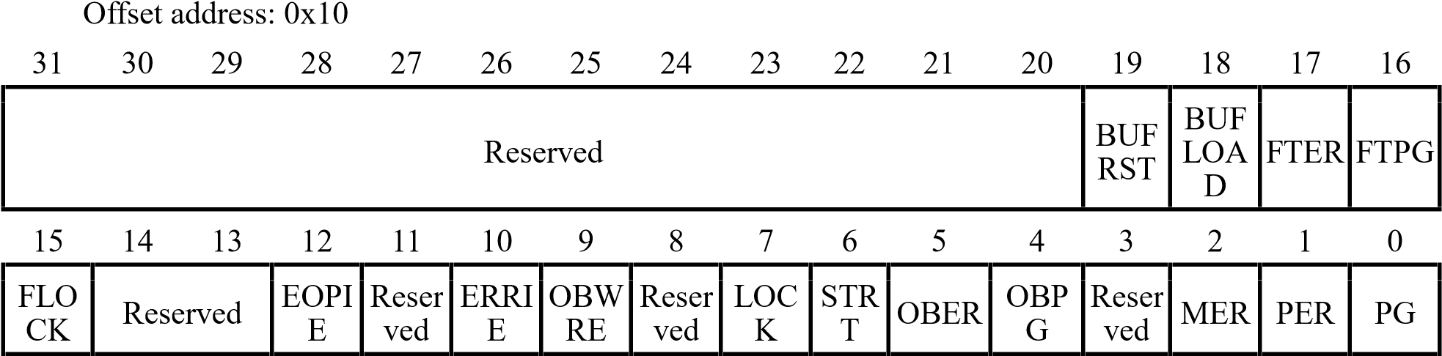
key to release OPTWRE.

### 16.3.4 Status Register (FLASH\_STATR)



*Note: When performing the programming operation, you need to make sure the STRT bit of FLASH\_CTLR register is 0.*

### 16.3.5 Configuration Register (FLASH\_ CTLR)



Reset

Bit Name Access Description

value

[31:20] Reserved RO Reserved 0

19 BUFRST RW BUF reset operation 0 18 BUFLOAD RW Cache data into BUF 0

17 FTER RW Performs a fast page (64Byte) erase operation. 0

16 FTPG RW Performs quick page programming operations. 0

Fast programming lock. Write '1' only. When this bit is '1' it indicates that fast programming/erase

15 FLOCK RW1 mode is not available. Hardware clears this bit to 1

'0' after the correct unlock sequence is detected.

The software is set to 1 and re-locked.

[14:13] Reserved RO Reserved 0

Operation completion interrupt control (EOP set in FLASH\_STATR register).

12 EOPIE RW 0

1: Allow generation of interrupts.

0: Interrupt generation is disabled.

11 Reserved RO Reserved 0

Error status interrupt control

(PGERR/WRPRTERR set in FLASH\_STATR

10 ERRIE RW register). 0

1: Allow generation of interrupts.

0: Interrupt generation is disabled.

User selects word lock, software clears 0.

1: Indicates that the user option bytes can be programmed for operation. It needs to be set by

9 OBWRE RW0 hardware after writing the correct sequence in 0

FLASH\_OBKEYR register.

0: Re-lock the user selection word after the software is cleared.

8 Reserved RO Reserved 0

Lock. Only '1' can be written. When this bit is '1' it means that FPEC and FLASH\_CTLR are locked and unwritable. Hardware clears this bit to '0' after

7 LOCK RW1 1

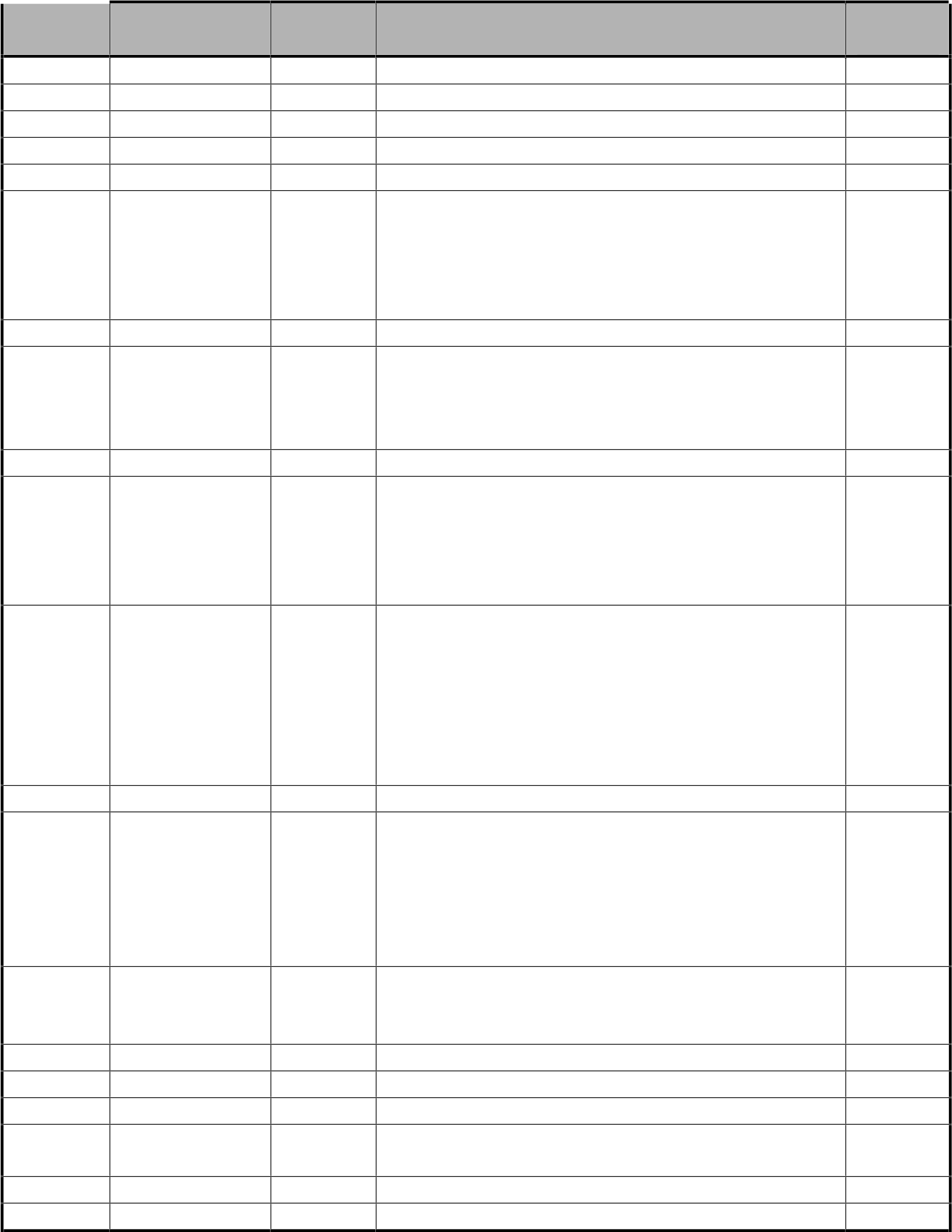
the correct unlock sequence is detected.

After an unsuccessful unlock operation, the bit will not be changed again until the next system reset.

Start. Set 1 to start an erase action and the

6 STRT RW1 hardware automatically clears 0 (BSY becomes 0

'0').

5 OBER RW Perform user-option bytes erasure 0 4 OBPG RW Perform user-option bytes programming 0

3 Reserved RO Reserved 0

Performs a full-erase operation (erases the entire

2 MER RW 0 user area).

1 PER RW Perform sector erase (1K) 0

0 PG RW Performs standard programming operations. 0

|  |  |  |
| --- | --- | --- |
| **16.3.6 Address Register (FLASH\_ ADDR)**  Offset address: 0x14 |  |  |
| 31 30 29 28 27 26 25 24 | 23 22 21 20 19 18 17 | 16 |

FAR

[31:16]

15

14

13

12

11

10

9

8

7

6

5

4

3

2

1

0

FAR

[15:0]

Bit Name Access Description Reset value

The flash memory address, when programming, is the programmed address, and when erasing, is the

[31:0] FAR[31:0] WO start address of the erase. 0

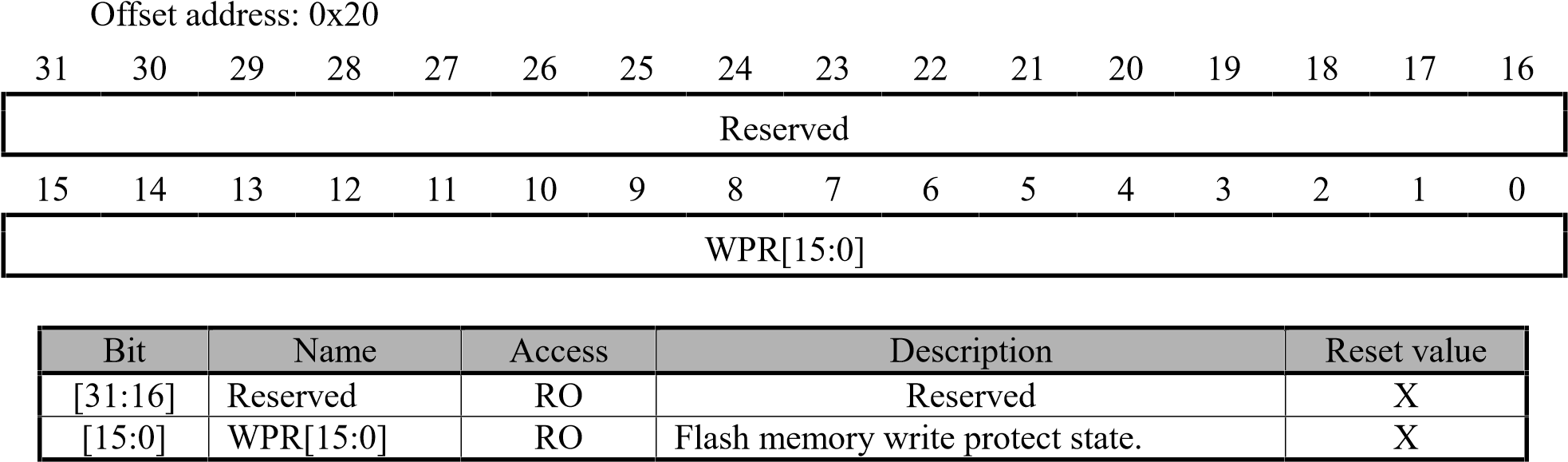
When the BSY bit in FLASH\_SR register is '1', this register cannot be written.

### 16.3.7 Option Byte Register (FLASH\_OBR)



*Note: USER and RDPRT are loaded from the user-option bytes area after a system reset.*

### 16.3.8 Write Protect Register (FLASH\_WPR)



1:

Write protection failure.

0:

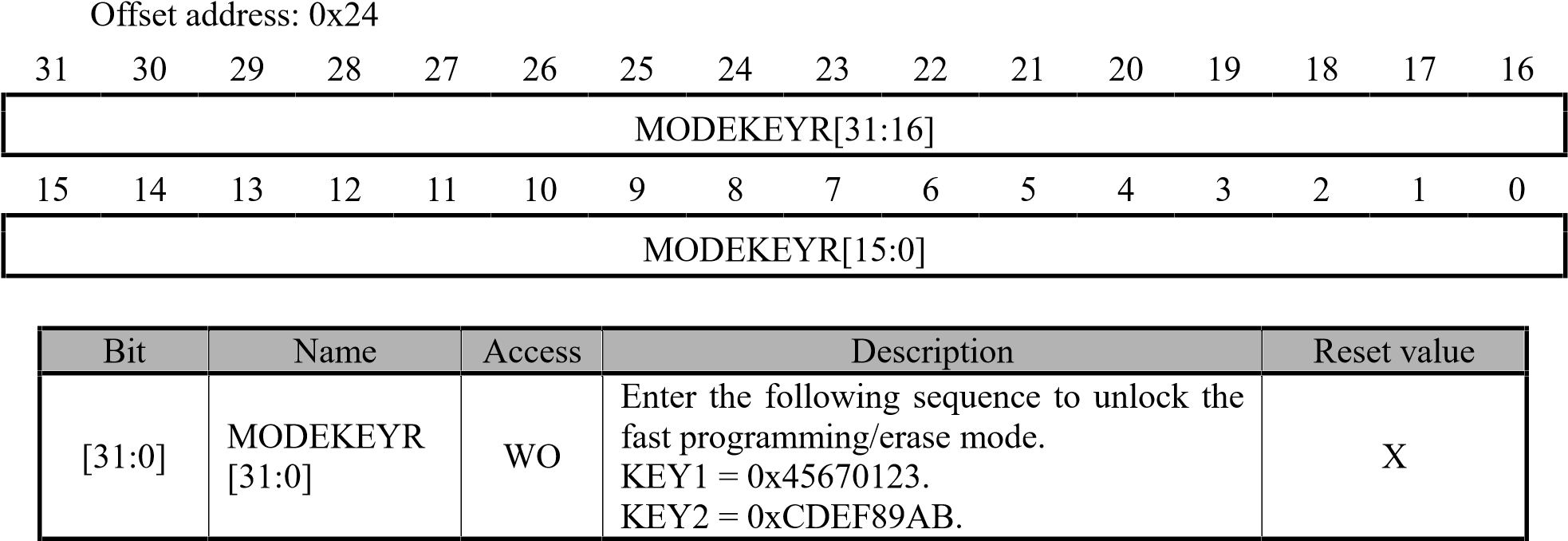
Write protection is valid.

Each bit represents 1K bytes (16 pages) of

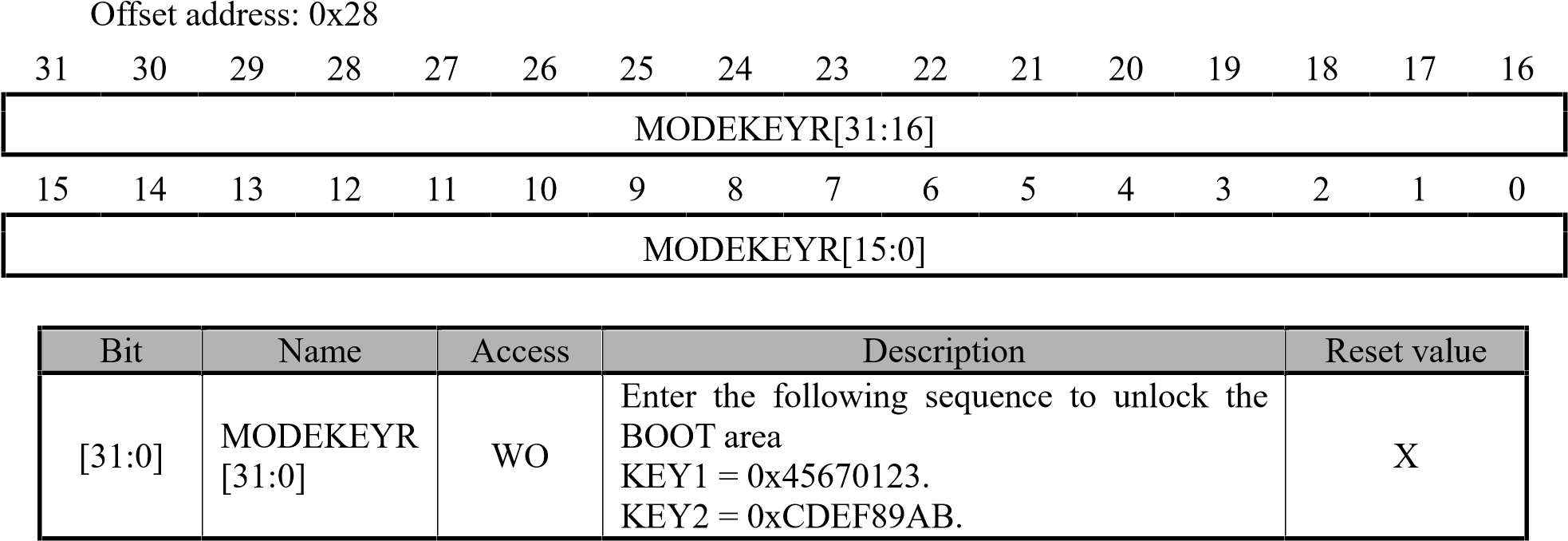
storage write protection status.

*Note: WPR is loaded from the user-option bytes area after a system reset.*

### 16.3.9 Extended Key Register (FLASH\_MODEKEYR)



### 16.3.10 BOOT Key Register (FLASH\_BOOT\_MODEKEYP)



## 16.4 Flash Memory Operation Flow

### 16.4.1 Read Operations

With direct addressing in the general address space, any read operation of 8/16/32-bit data can access the contents of the flash module and get the corresponding data.

### 16.4.2 Unlocking the Flash Memory

After a system reset, the flash controller (FPEC) and FLASH\_CTLR registers are locked and inaccessible. The flash controller module can be unlocked by writing a sequence to the FLASH\_KEYR register.

Unlock sequence.

1. Write KEY1 = 0x45670123 to the FLASH\_KEYR register (step 1 must be KEY1).
2. Write KEY2 = 0xCDEF89AB to FLASH\_KEYR register (step 2 must be KEY2).

The above operations must be executed sequentially and consecutively, otherwise they are error operations and will lock the FPEC module and FLASH\_CTLR registers and generate bus errors until the next system reset.

The flash memory controller (FPEC) and FLASH\_CTLR registers can be locked again by setting the "LOCK" bit of the FLASH\_CTLR register to 1.

### 16.4.3 Main Memory Standard Programming

Standard programming can be written 2 bytes at a time. When the PG bit of FLASH\_CTLR register is '1', each half-word (2 bytes) written to the flash address will initiate programming once, and writing any non-half-word data will cause the FPEC to generate a bus error. During programming, the BSY bit is '1', and at the end of programming, the BSY bit is '0' and the EOP bit is '1'.

*Note: When the BSY bit is '1', it will prohibit to perform write operation to any register.*

Figure 16-1 FLASH Programming

Read the LOCK bit of

FLASH\_CTRL

BSY bit=1

？

Perform "Unlock Flash

Memory" operation

Set FLASH\_CTLR PG bit=1

Specified address write half

word (2 bytes)

Read EOP/WRPRTERR to judge the

programming result Read the programmed

address to check the written data

Continue

programming?

Over

，

PG bit=0

LOCK bit=1?

YES

NO

NO

YES

YES

NO

1. Check the FLASH\_CTLR register LOCK, if it is 1, you need to execute the "Unlock Flash" operation.
2. Set the PG bit of FLASH\_CTLR register to '1' to enable the standard programming mode.
3. Write the half word to be programmed to the specified flash address (even address).
4. Wait for the BSY bit to become '0' or the EOP bit of FLASH\_STATR register to be '1' to indicate the end of programming, and clear the EOP bit to 0.
5. Query the FLASH\_STATR register to see if there is an error or read the programmed address data checksum.
6. Continue programming you can repeat steps 3-5 and end programming to clear the PG bit to 0.

### 16.4.4 Main Memory Standard Erase

Flash memory can be erased by standard page (1K bytes) or by whole chip.

Figure 16-2 FLASH Page Erase

Read the LOCK bit of

FLASH\_CTRL

BSY bit=1

？

Perform "Unlock Flash

Memory" operation

Set FLASH\_CTLR

的

PER bit=1

Set FLASH\_CTLR STRT bit=1

Read out erase page data

verification

Continue

erasing?

Over

，

PEG bit=0

LOCK bit=1?

YES

NO

NO

YES

YES

NO

Write the erased page header

address in FLASH\_ADDR register

(

)

erase 8 pages at a time

1. Check the LOCK bit of FLASH\_CTLR register, if it is 1, you need to execute the "Unlock Flash" operation.
2. Set the PER bit of FLASH\_CTLR register to '1' to enable the standard page erase mode.
3. Write the page header address of the selected erase to FLASH\_ADDR register.
4. Set the STRT bit of FLASH\_CTLR register to '1' to initiate an erase action.
5. Wait for the BSY bit to become '0' or the EOP bit of FLASH\_STATR register to be '1' to indicate the end of erase, and clear the EOP bit to 0.
6. Read the data of the erased page for verification.
7. Continue the standard page erase can repeat steps 3-5 and end the erase to clear the PEG bit to 0.

Figure 16-3 FLASH whole chip erase

Read the LOCK bit of

FLASH\_CTRL

BSY bit=1

？

Perform "Unlock Flash

Memory" operation

Set FLASH\_CTLR MER bit=1

Set FLASH\_CTLR STRT bit=1

Read out all pages of data to

verify

Over

，

MEG

bit=0

LOCK bit=1?

YES

NO

YES

NO

1. Check the LOCK bit of FLASH\_CTLR register, if it is 1, you need to execute the "Unlock Flash" operation.
2. Set the MER bit of FLASH\_CTLR register to '1' to enable the whole chip erase mode.
3. Set the STRT bit of FLASH\_CTLR register to '1' to start the erase action.
4. Wait for the BSY bit to become '0' or the EOP bit of FLASH\_STATR register to be '1' to indicate the end of erase, and clear the EOP bit to 0.
5. Read the data of the erased page for verification.
6. Clear the MER bit to 0.

### 16.4.5 Fast Programming Mode Unlocking

Fast programming mode operation can be unlocked by writing a sequence to the FLASH\_MODEKEYR register. After unlocking, the FLOCK bit of FLASH\_CTLR register will be cleared to 0, indicating that fast erase and programming operations can be performed. The FLASH\_CTLR register is locked again by software setting the "FLOCK" bit to 1.

Unlock sequence.

1. Write KEY1 = 0x45670123 to the FLASH\_MODEKEYR register.
2. Write KEY2 = 0xCDEF89AB to FLASH\_MODEKEYR register.

The above operations must be performed sequentially and consecutively, otherwise they are wrong operations will be locked and cannot be unlocked again until the next system reset.

*Note: Quick programming operation requires unlocking the "LOCK" and "FLOCK" layers.*

### 16.4.6 Main Memory Fast Programming

Fast programming by page (64 bytes).

1. Check the LOCK bit of FLASH\_CTLR register, if it is 1, you need to execute the "Unlock Flash" operation.
2. Check the BSY bit of the FLASH\_STATR register to confirm that there are no other programming operations in progress.
3. Check the FLASH\_CTLR register FLOCK bit, if it is 1, you need to execute the "fast programming mode unlock" operation.
4. Set the FTPG bit of FLASH\_CTLR register to enable the fast programming mode function.
5. Set the BUFRST bit of FLASH\_CTLR register to perform the operation of clearing the internal 64-byte buffer.
6. Wait for the BSY bit to become '0' or the EOP bit of FLASH\_STATR register to be '1' to indicate the end of clearing, and clear the EOP bit to 0.
7. Start writing 4 bytes of data to the specified address (4 bytes/operation), then set the BUFLOAD bit of FLASH\_CTLR register and execute loading to the buffer.
8. Wait for the BYS bit to become '0' or the EOP bit of FLASH\_STATR register to be '1' to indicate the end of loading, and clear the EOP bit to 0.
9. Repeat steps 7-8 a total of 16 times to load all 64 bytes of data into the buffer (16 rounds of operation addresses should be consecutive).
10. Write the first address of the fast programming page to the FLASH\_ADDR register.
11. Set the STRT bit of FLASH\_CTLR register to '1' to start a fast page programming action.
12. Wait for the BSY bit to become '0' or the EOP bit of FLASH\_STATR register to be '1' to indicate the end of programming, and clear the EOP bit to 0.
13. Query FLASH\_STATR register to see if there is an error, or read the programmed address data checksum.
14. Continue the Quick Page programming can repeat steps 5-13 and end the programming to clear the FTPG bit to 0.

### 16.4.7 Main Memory Fast Erase

Fast Erase erases by page (64 bytes).

1. Check the LOCK bit of FLASH\_CTLR register, if it is 1, you need to execute the "Unlock Flash" operation.
2. Check the FLASH\_CTLR register FLOCK bit, if it is 1, you need to execute the "fast programming mode unlock" operation.
3. Check the BSY bit of the FLASH\_STATR register to confirm that there are no other programming operations in progress.
4. Set the FTER bit of FLASH\_CTLR register to '1' to enable the fast page erase (64 bytes) mode function.
5. Write the first address of the fast erase page to the FLASH\_ADDR register.
6. Set the STRT bit of FLASH\_CTLR register to '1' to initiate a fast page erase (64 bytes) action.
7. Wait for the BSY bit to become '0' or the EOP bit of FLASH\_STATR register to be '1' to indicate the end of erase, and clear the EOP bit to 0.
8. Query FLASH\_STATR register to see if there is an error, or read the erase page address data checksum.
9. Continue fast page erase can repeat steps 5-8, end erase will FTER bit clear 0.

## 16.5 User Option Bytes

The user-option bytes is solidified in FLASH and will be reloaded into the corresponding register after system reset, and can be erased and programmed by the user at will. The user option bytes information block has a total of 8 bytes (4 bytes for write protection, 1 byte for read protection, 1 byte for configuration options, and 2 bytes for storing user data), and each bit has its inverse code bit for checksum during loading. The following describes the structure and meaning of the option bytes information.

Table 16-3 32-bit option byte format division

[31:24] [23:16] [15:8] [7:0]

Option bytes byte 1 Option bytes byte 0

Option bytes byte 1 Option bytes byte 0 inverse code inverse code

Table 16-4 User option byte information structure

Address

[31:24] [23:16] [15:8] [7:0]

Bit

0x1FFFF800 nUSER USER nRDPR RDPR

0x1FFFF804 nData1 Data1 nData0 Data0

0x1FFFF808 nWRPR1 WRPR1 nWRPR0 WRPR0

0x1FFFF80C Reserved Reserved Reserved Reserved

Name/Byte Description Reset value

Read protection control bit to configure whether the code in the flash memory can be read out.

0xA5: if this byte is 0xA5 (nRDP must be 0x5A), it

means that the current code is in a non-read protected

RDPR 0x01

state and can be read out.

Other values: indicates code read protection status, not readable, pages 0-31 (4K) will be automatically write protected and not controlled by WRPR0.

[7:6] Reserved Reserved (must be 1) 11b

Power-on startup mode

1: Boot from BOOT area

START\_M

5 0: Boot from user area 1

ODE

*Note: This function is not applicable to products with 0 in the penultimate digit of the lot number.*

PD7 multiplexed as external pin reset.

USER 00: Ignoring pin states within 128us after turning on the multiplexing function.

RST\_MOD 01: Ignoring pin states within 1ms after turning on the

[4:3] 10b

E multiplexing function.

10: Ignoring pin states within 12ms after turning on the multiplexing function.

11: Multiplexing function off, PD7 for I/O function.

STANDYR System reset control in Standby mode:

2 1

ST 1: Not enabled, does not reset when entering Standby

mode system;

0: Enabled, generates a system reset when entering Standby mode.

1 Reserved Reserved 1

Independent Watchdog (IWDG) hardware enable configuration.

1: IWDG is enabled by software and disabled from being enabled by hardware.

0 IWDGSW 0: IWDG is turned on by hardware itself (since the clock 1 for IWDG is provided by LSI, it is automatically turned on by LSI).

*Note: The core stops in debug mode and the* *watchdog hardware enable will be disabled.*

Data0–Data1 Store 2 bytes of user data. FFFFh

Write-protect control bits. Each bit is used to control the write-protect status of 1 sector (1K bytes/sector) in main memory.

1: Disable write protection.

0: Enable Write protection.

WRPR0 - WRPR3 2 bytes are used to protect a total of 16K bytes of main FFFFh memory.

WRP0: Sector 0-7 storage write protection control.

WRP1: Sector 8-15 storage write protection control.

WRP2: Reserved.

WRP3: Reserved.

### 16.5.1 User Option Bytes Unlocking

The user option bytes operation can be unlocked by writing a sequence to the FLASH\_OBKEYR register. After unlocking, the OBWRE bit of FLASH\_CTLR register will be set to 1, indicating that the user option bytes can be erased and programmed. It can be locked again by clearing the "OBWRE" bit of FLASH\_CTLR register to 0 by software.

Unlock sequence.

1. Write KEY1 = 0x45670123 to FLASH\_OBKEYR register.
2. Write KEY2 = 0xCDEF89AB to FLASH\_OBKEYR register.

*Note: User-option bytes operation requires unlocking the "LOCK" and "OBWRE" layers.*

### 16.5.2 User Option Bytes Programming

Only the standard programming method is supported, writing half-words (2 bytes) at a time. In practice, when programming the user-option bytes, FPEC uses only the low byte in the half-word and automatically calculates the high byte (the high byte is the inverse of the low byte) and then starts the programming operation, which will ensure that the byte in the user option bytes and its inverse code are always correct.

1. Check the LOCK bit of FLASH\_CTLR register, if it is 1, you need to execute the "Unlock Flash" operation.
2. Check the BSY bit of the FLASH\_STATR register to confirm that there are no other programming operations in progress.
3. Set the OBPG bit of FLASH\_CTLR register to '1', after that set the STAT bit of FLASH\_CTLR register to '1' to turn on the user option bytes programming.
4. Set the OBPG bit of FLASH\_CTLR register to '1'.
5. Write the half word (2 bytes) to be programmed to the specified address.
6. Wait for the BSY bit to become '0' or the EOP bit of FLASH\_STATR register to be '1' to indicate the end of programming, and clear the EOP bit to 0.
7. Read the programmed address data checksum.
8. Continue programming you can repeat steps 5-7 and end programming to clear the OBPG bit to 0.

*Note: When "Read Protected" in the modified selection word becomes "Unprotected", a whole-slice erase of the main memory will be performed automatically. If the selection other than "read protected" is modified, the whole erase operation will not occur.*

### 16.5.3 User Option Bytes Erasure

Directly erase the entire 64-byte user option bytes area.

1. Check the LOCK bit of FLASH\_CTLR register, if it is 1, you need to execute the "Unlock Flash" operation.
2. Check the BSY bit of the FLASH\_STATR register to confirm that there is no programming operation in progress.
3. Check the OBWRE bit of FLASH\_CTLR register, if it is 0, it is necessary to execute the operation of "user option bytes unlock".
4. Set the OBER bit of FLASH\_CTLR register to '1', after that set the STAT bit of FLASH\_CTLR register to '1' to enable the user option bytes erase.
5. Wait for the BSY bit to become '0' or the EOP bit of FLASH\_STATR register to be '1' to indicate the end of erase, and clear the EOP bit to 0 6) Read and erase the address data checksum.

7) End to clear the OBER bit to 0.

### 16.5.4 Unprotecting Reads

Whether the flash memory is read protected or not is determined by the user option bytes. Read the FLASH\_OBR register, when the RDPRT bit is '1' indicates that the flash memory is currently in the readprotected state, and the flash memory is operationally protected by a series of security guards for the readprotected state. The process of unprotecting the read protection is as follows.

1. Erase the entire user-option bytes area, at which point the read protection field RDPR, at which point the read protection remains in effect.
2. User-option bytes programming and writes the correct RDPR code 0xA5 to unprotect the flash memory from reads. (This step will first cause the system to automatically perform an entire erase operation on the flash memory)
3. Perform a power-on reset to reload the selection byte (including the new RDPR code), at which point the read protection is removed.

# Chapter 17 Extended configuration

## 17.1 Extended Configuration

The system provides the EXTEND extended configuration unit (EXTEND\_CTR register). This unit uses the HB clock and performs a reset action only at system reset. It mainly includes the following extended control bit functions.

1. Adjusting the built-in voltage: The LDOTRIM field selects the default value, which can be modified when adjusting performance and power consumption.
2. Lock-up function monitoring: The LKUPEN field is enabled, which will open the Lock-up situation monitoring of the system. Once the Lock-up situation occurs, the system will perform a software reset and set the LKUPRESET field to 1. After reading, you can write 1 to clear this flag.
3. Configure the op-amp: set OPA\_EN to enable the OPA, configure OPA\_PSEL to select the positive input pin of the OPA, and configure OPA\_NSEL to select the negative input pin of the OPA.

## 17.2 Register Description

Table 17-1 EXTEND-related registers list

Name Access address Description Reset value

R32\_EXTEN\_CTR 0x40023800 Configure extended control registers 0x00000400

### 17.2.1 Configure Extended Control Register (EXTEND\_CTR)



6

LKUPEN

RW

LOCKUP monitoring function.

1:

Enabled, performs a reset and sets

LOCKUP\_RESET when a lock-up occurs on

the system.

Not enabled.

0:

1

[5:0]

Reserved

RO

Reserved

0

# Chapter 18 Debug Support (DBG)

## 18.1 Main Features

This register allows the MCU to be configured in the debug state. It includes:

* Independent Watchdog (IWDG) enabled counters
* Window Watchdog (WWDG) enabled counters
* Timer1 enabled counters  Timer2 enabled counters

## 18.2 Register Description

### 18.2.1 Debug MCU Configuration Register (DBGMCU\_CR) Address: 0x7C0(CSR)

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

TIM2 TIM1 WW IWD

STAN Reser SLEE

Reserved \_STO \_STO Reserved DG\_S G\_ST Reserved

DBY ved P

P P TOP OP

Reset

Bit Name Access Description

value

[31:14] Reserved RW Reserved 0

Timer 2 debug stop bit. The counter stops when the core enters the debug state.

13 TIM2\_STOP RW 0

1: Timer 2's counter stops working.

0: Timer 2's counter is still working normally. Timer 2 debug stop bit. The counter stops when the core enters the debug state.

12 TIM1\_STOP RW 0

1: Timer 2's counter stops working.

0: Timer 2's counter is still working normally.

[10:11] Reserved RW Reserved 0

WWDG debug stop bit. The debug WWDG stops working when the core enters the debug state.

9 WWDG\_STOP RW 0

1: WWDG counter stops working.

0: WWDG counter is still working normally. IWDG debug stop bit. The debug IWDG stops working when the core enters the debug state.

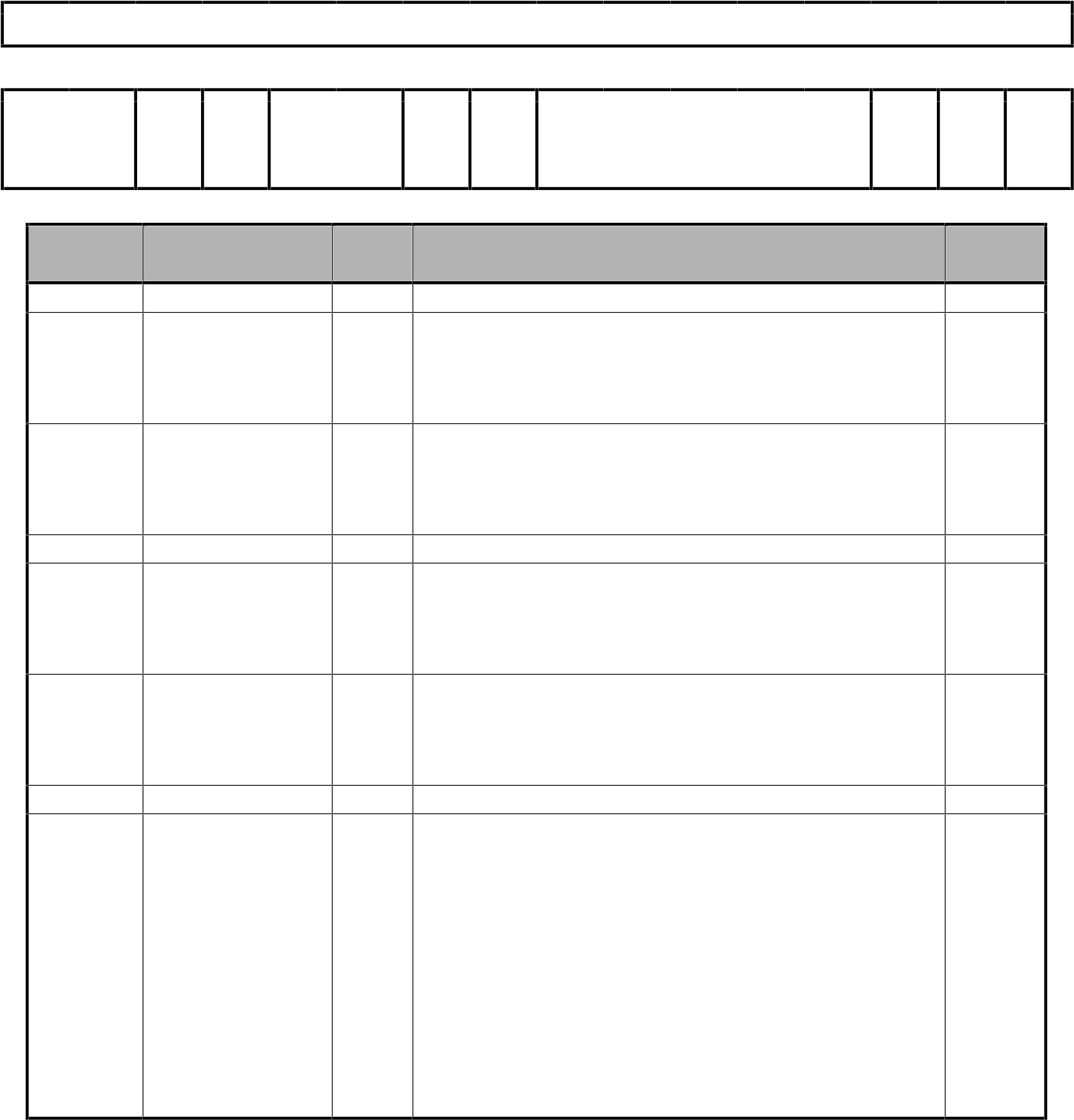
8 IWDG\_STOP RW 0

1: IWDG counter stops working.

0: IWDG counter is still working normally.

[7:3] Reserved RW Reserved 0

Debug the standby mode bits.

1: (FCLK on, HCLK on) The digital circuitry section is not powered down, and the FCLK and HCLK clocks are clocked by the internal RL oscillator. Alternatively, the microcontroller exits STANDBY mode and reset by

2 STANDBY RW generating a system reset is the same. 0

0: (FCLK off, HCLK off) The entire digital circuitry section is powered down.

From the software point of view, exiting STANDBY mode is the same as a reset (except that some status bits indicate that the microcontroller has just exited from STANDBY state).

1 Reserved RO Reserved 0

Debug sleep mode bits.

1: (FCLK on, HCLK on) In Sleep mode, both FCLK and HCLK clocks are provided by the originally configured system clock.

0: (FCLK on, HCLK off) In Sleep mode, FCLK is

0 SLEEP RO 0

provided by the originally configured system clock, and HCLK is off. Since Sleep mode does not reset the configured clock system, the software does not need to reconfigure the clock system when exiting from sleep mode.