**CH32V003 Reference Manual**



V1.7 [https://www.wch-ic.com](https://www.wch-ic.com/)

# 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:

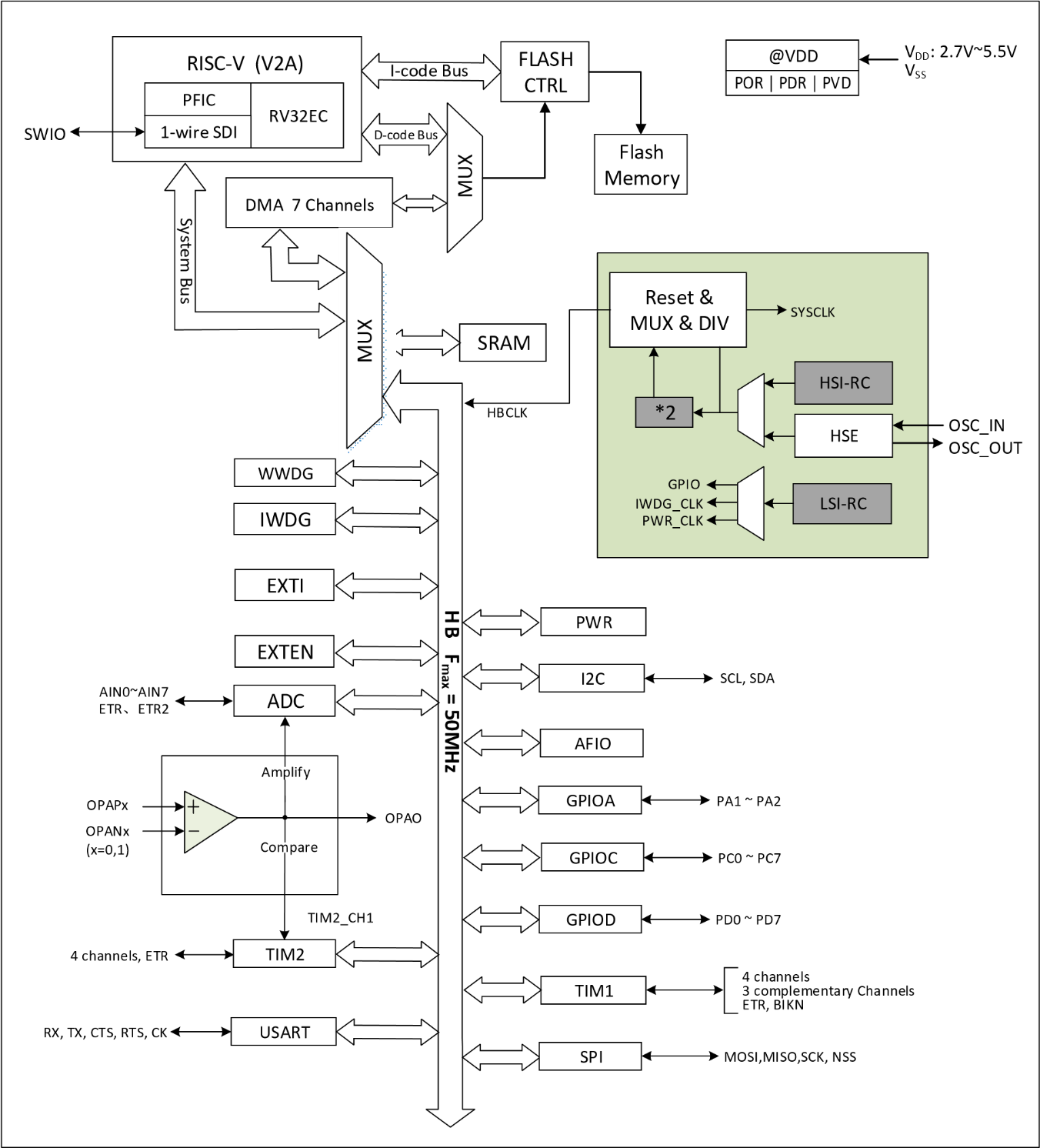
|  |  |
| --- | --- |
| Аббревиатура | Описание свойств |
| RF | Свойство только для чтения, которое считывает фиксированное значение. |
| RO | Атрибут, доступный только для чтения, изменяемый аппаратным обеспечением. |
| RZ | Свойство только для чтения, автоматическое удаление бита 0 после операции 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.

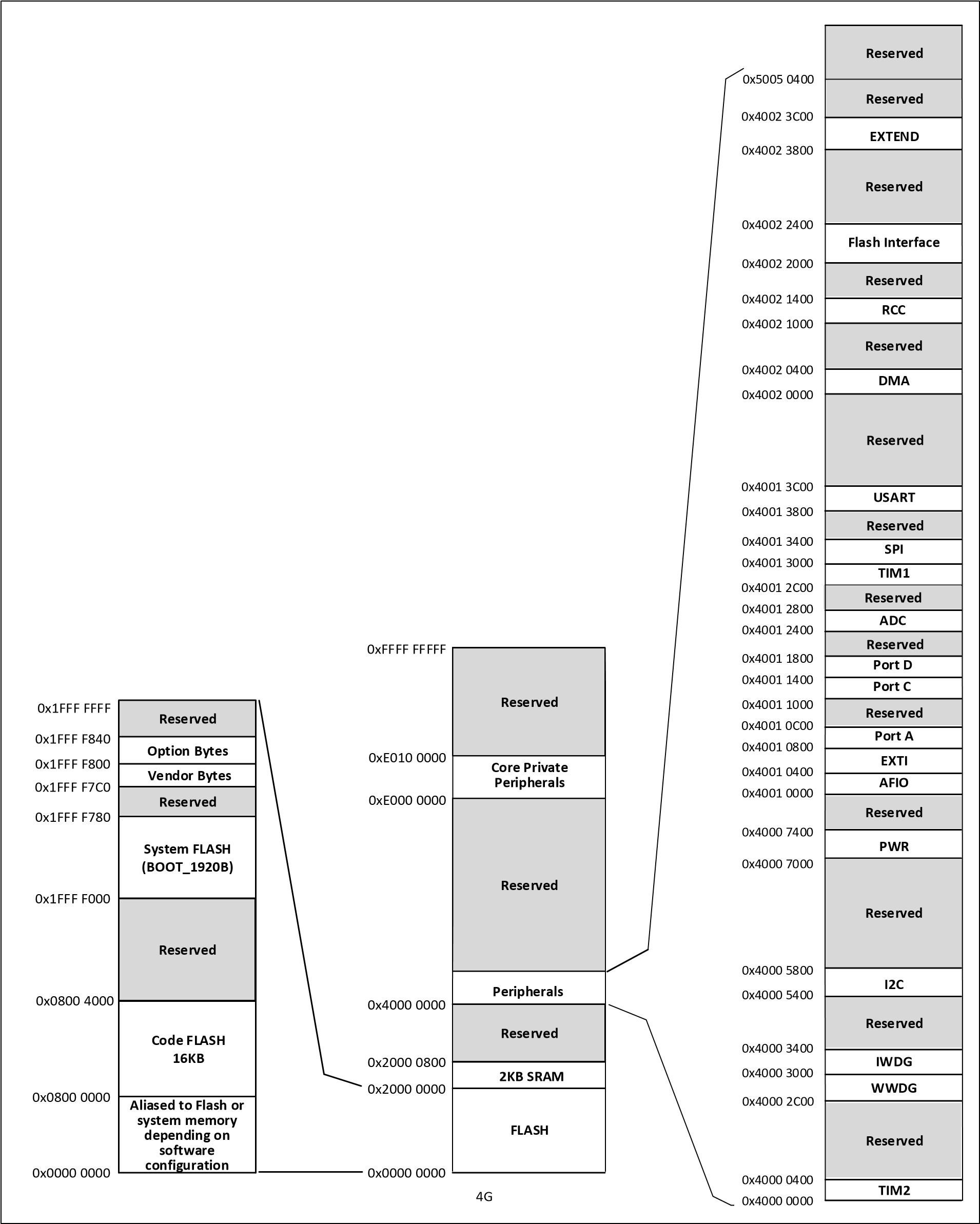
* Шина команд (I-Code) соединяет ядро с интерфейсом команд FLASH, и предварительная выборка выполняется по этой шине.
* Шина данных (D-Code) соединяет ядро с интерфейсом передачи данных FLASH для постоянной загрузки и отладки.
* Системная шина соединяет ядро с матрицей шин и используется для координации 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).

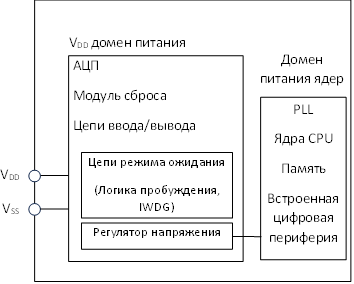
Встроенное пространство объемом 64 Б для хранения слов конфигурации поставщика, отлаженное на заводе-изготовителе и не изменяемое пользователями. Встроенное пространство объемом 64 Б для хранения пользовательских байт.

# 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

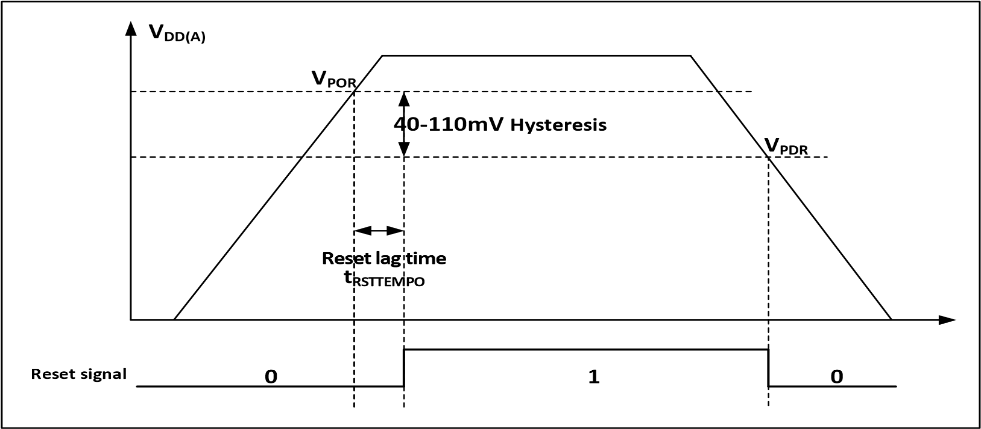


## 2.2 Управление питанием

### 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



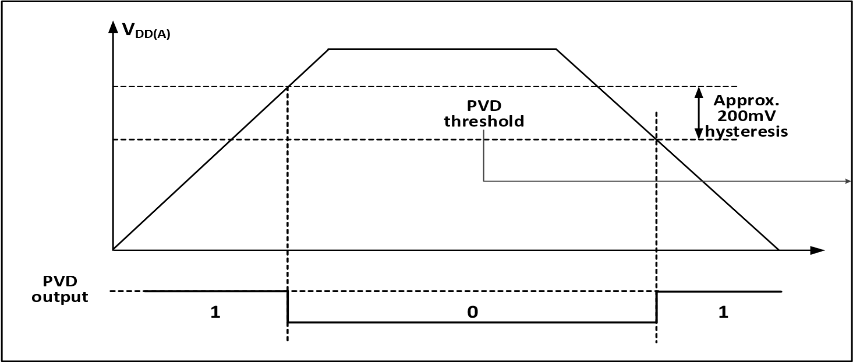
### 2.2.2 Программируемый датчик напряжения

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

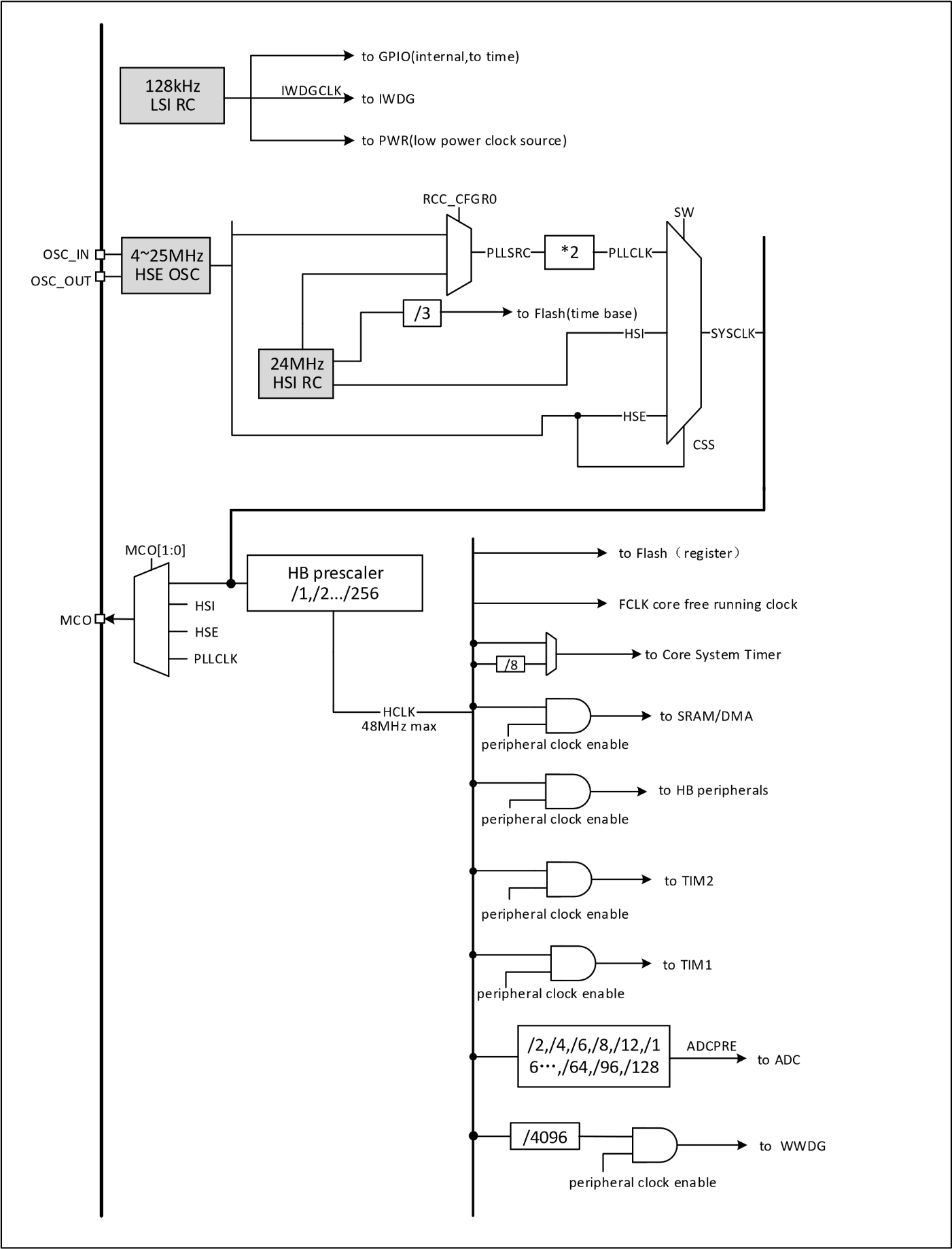
DDA

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

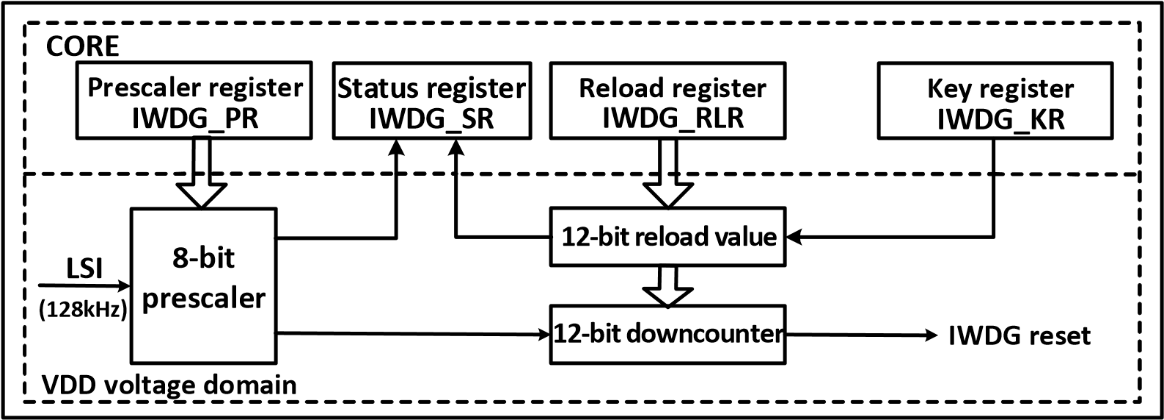
## 4.2 Описание функции

## 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:

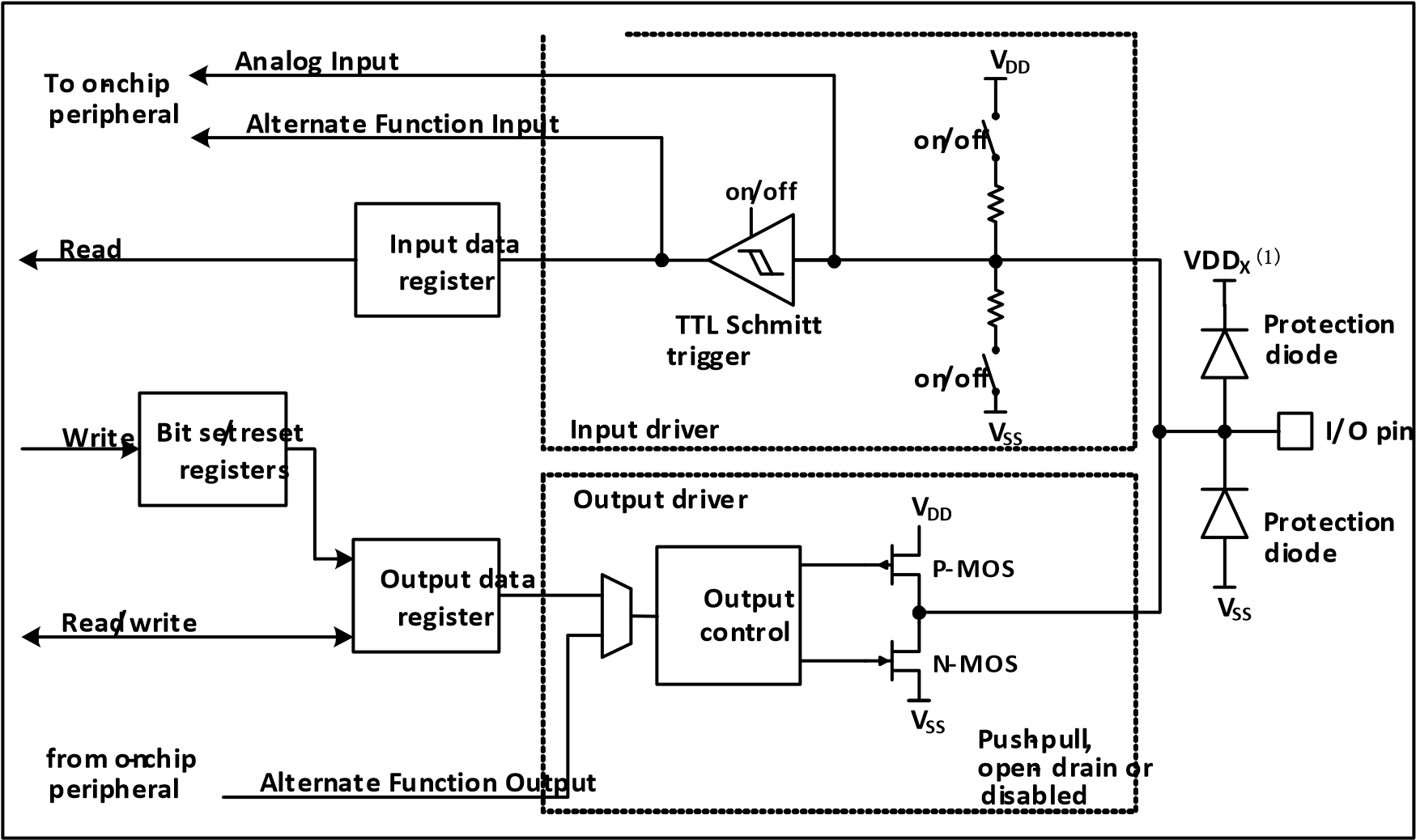
* Floating input (вход с плавающим потенциалом)
* Open drain output (выход с открытым стоком)
* Pull-up input (вход с подтяжкой вверх)
* Push-pull output (выход типа push-pull)
* Dropdown input (вход с подтягиванием вниз)
* Multiplexing the inputs and outputs of functions (мультиплексирование входов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

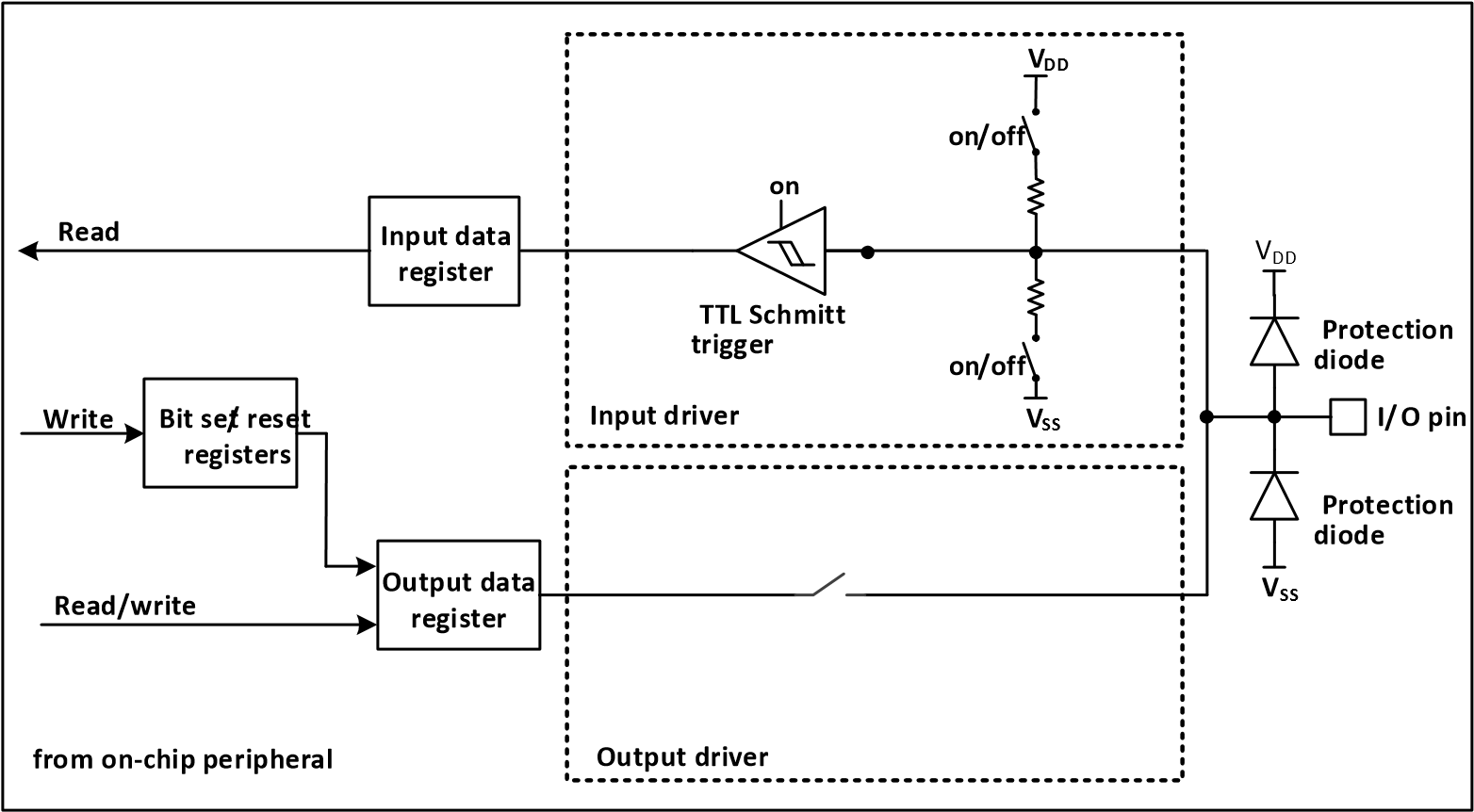
### 7.2.5 Механизм блокировки (

### 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

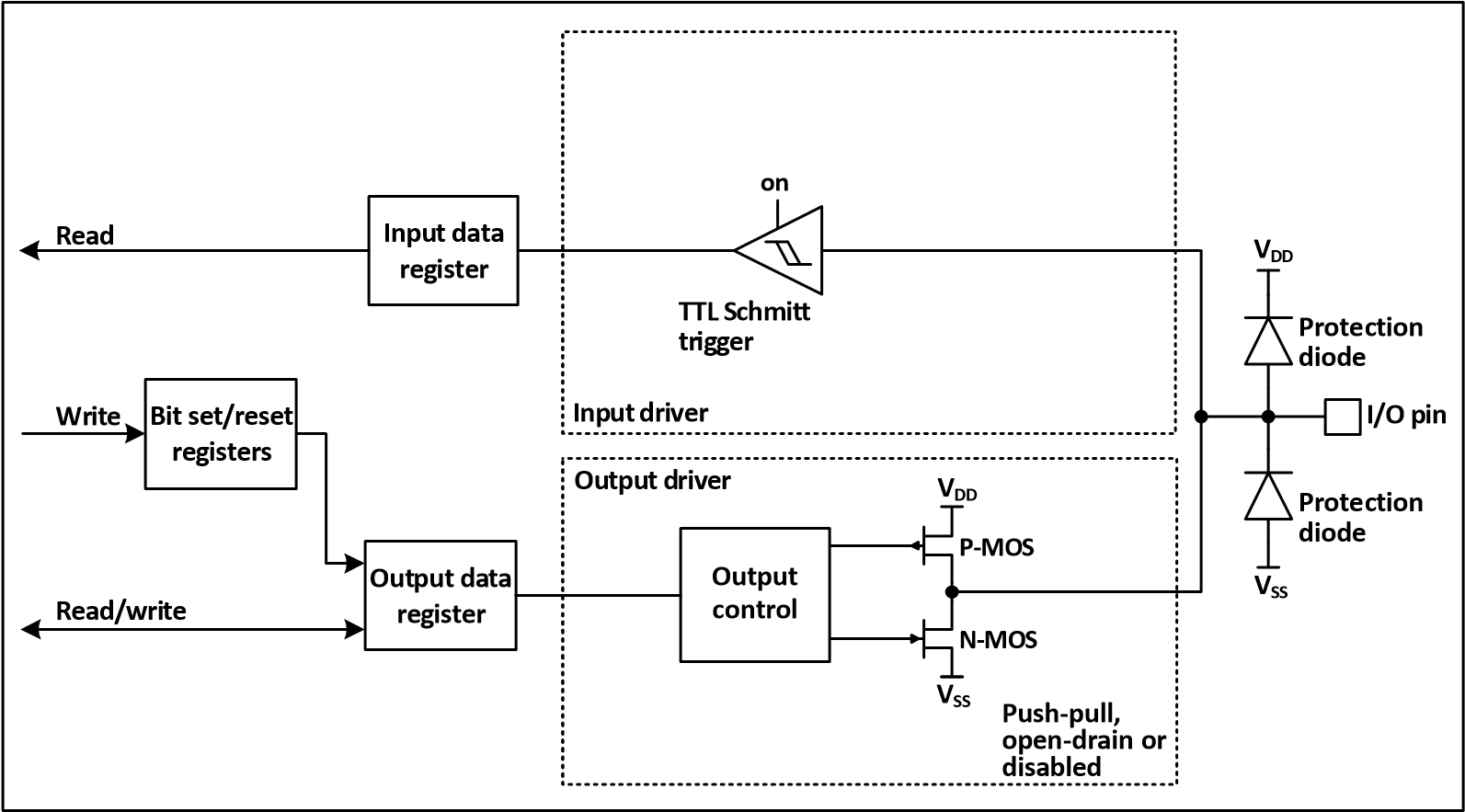
Рисунок 7-2 Блок-схема структуры конфигурации ввода модуля GPIO



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

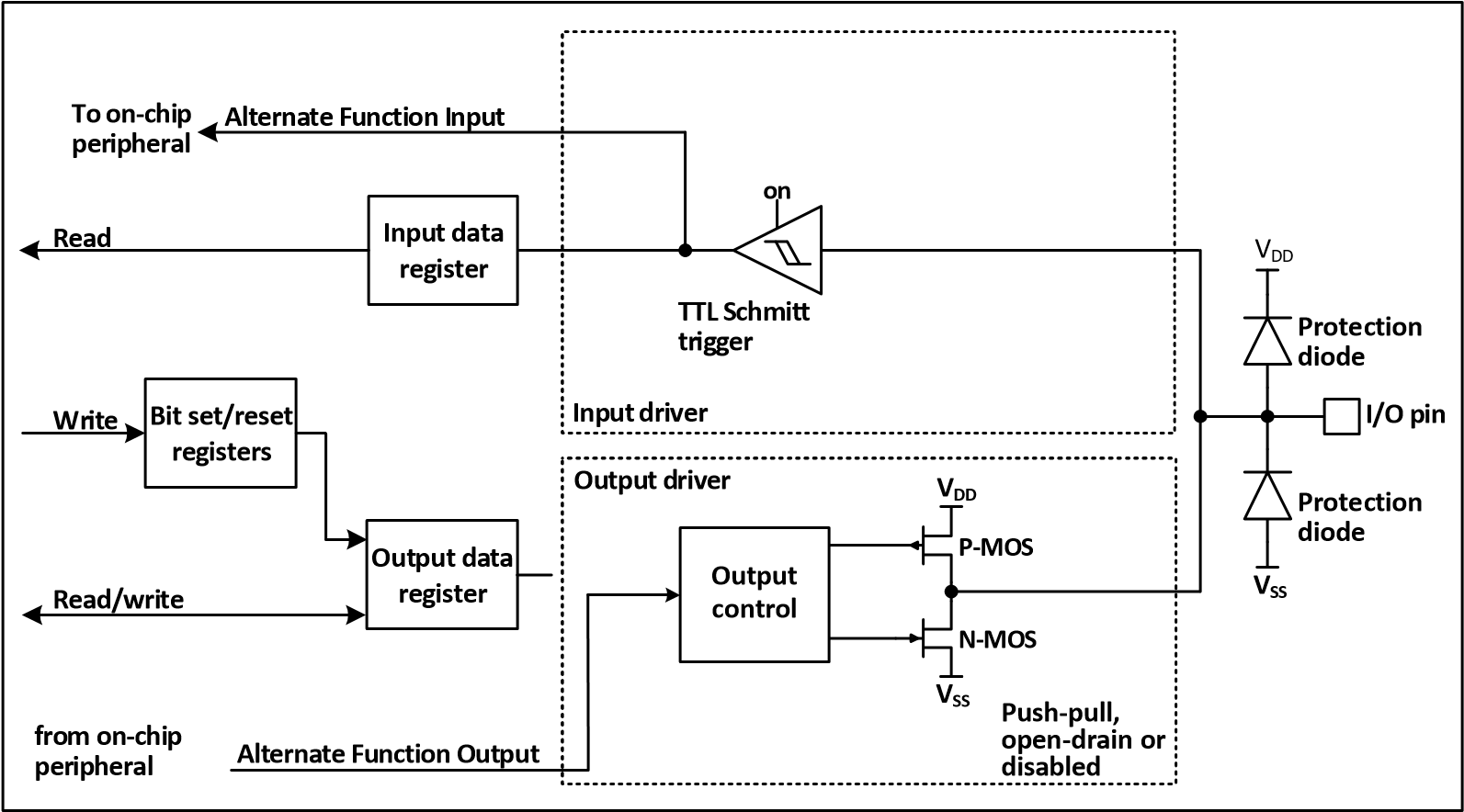
Рисунок 7-3 Блок-схема структуры конфигурации вывода модуля GPIO



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

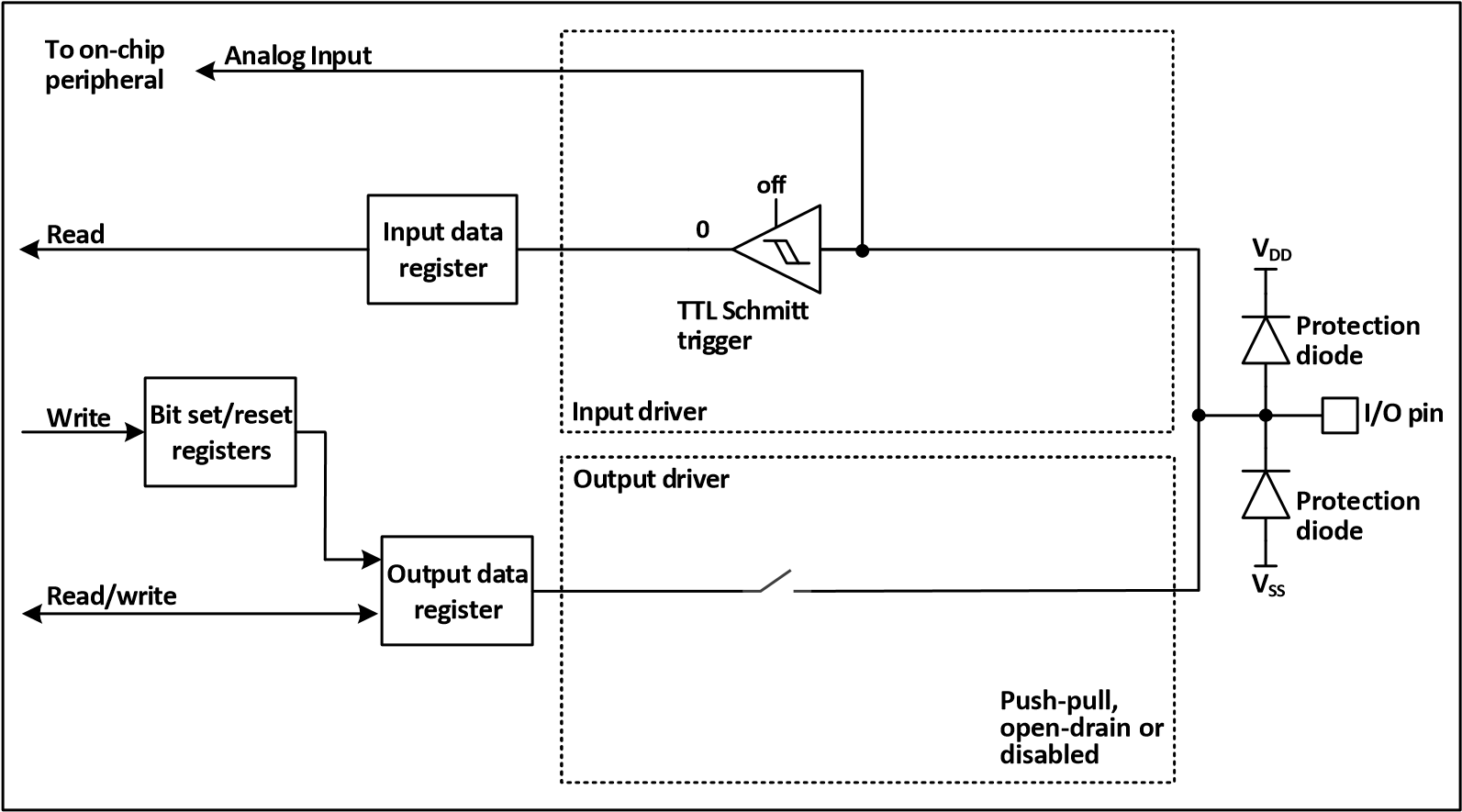
Рисунок 7-4 Структура модуля GPIO при его мультиплексировании другой периферией



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

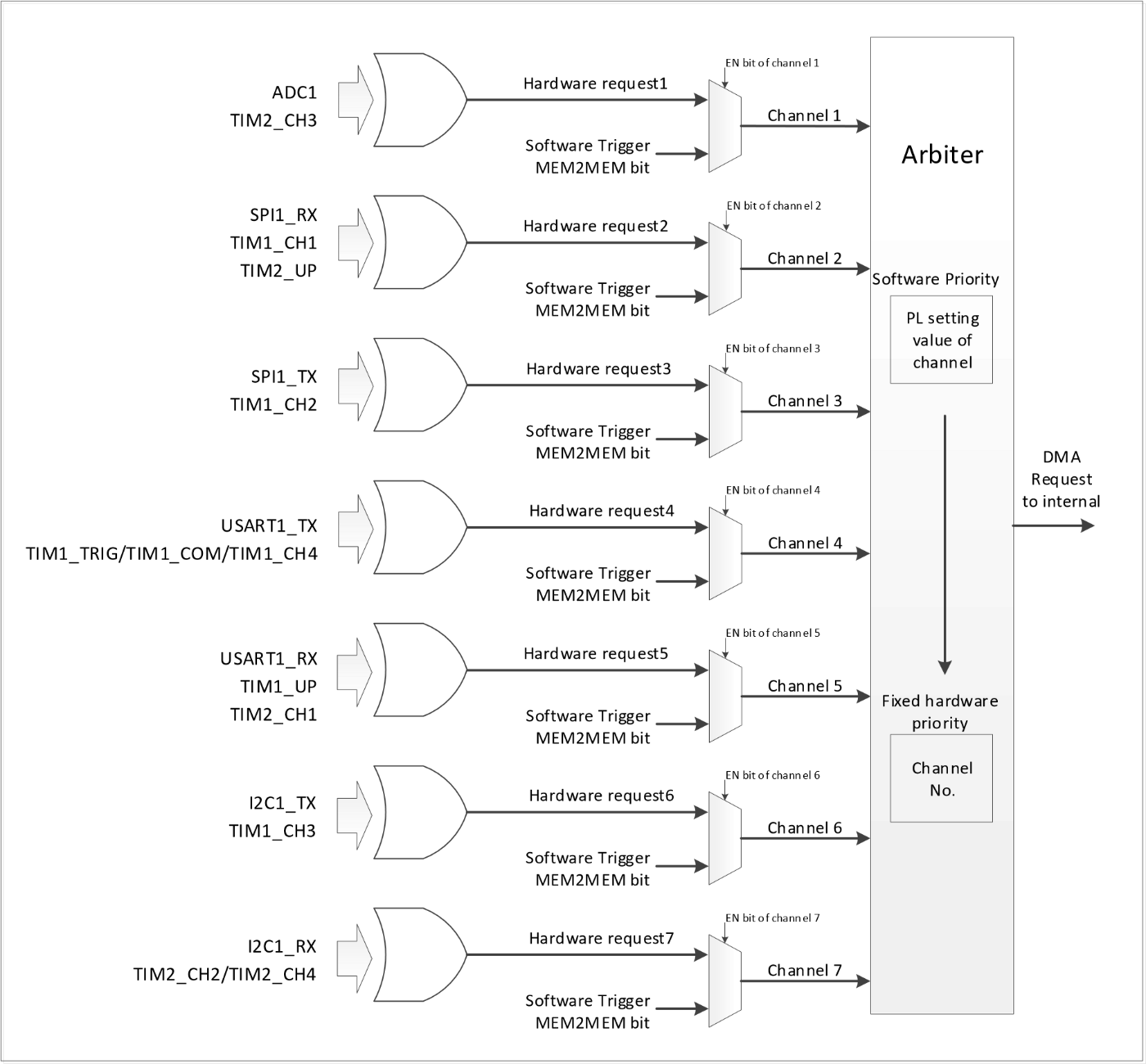


Таблица 8-2 Таблица сопоставления периферийных устройств DMA1 для каждого канала

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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.

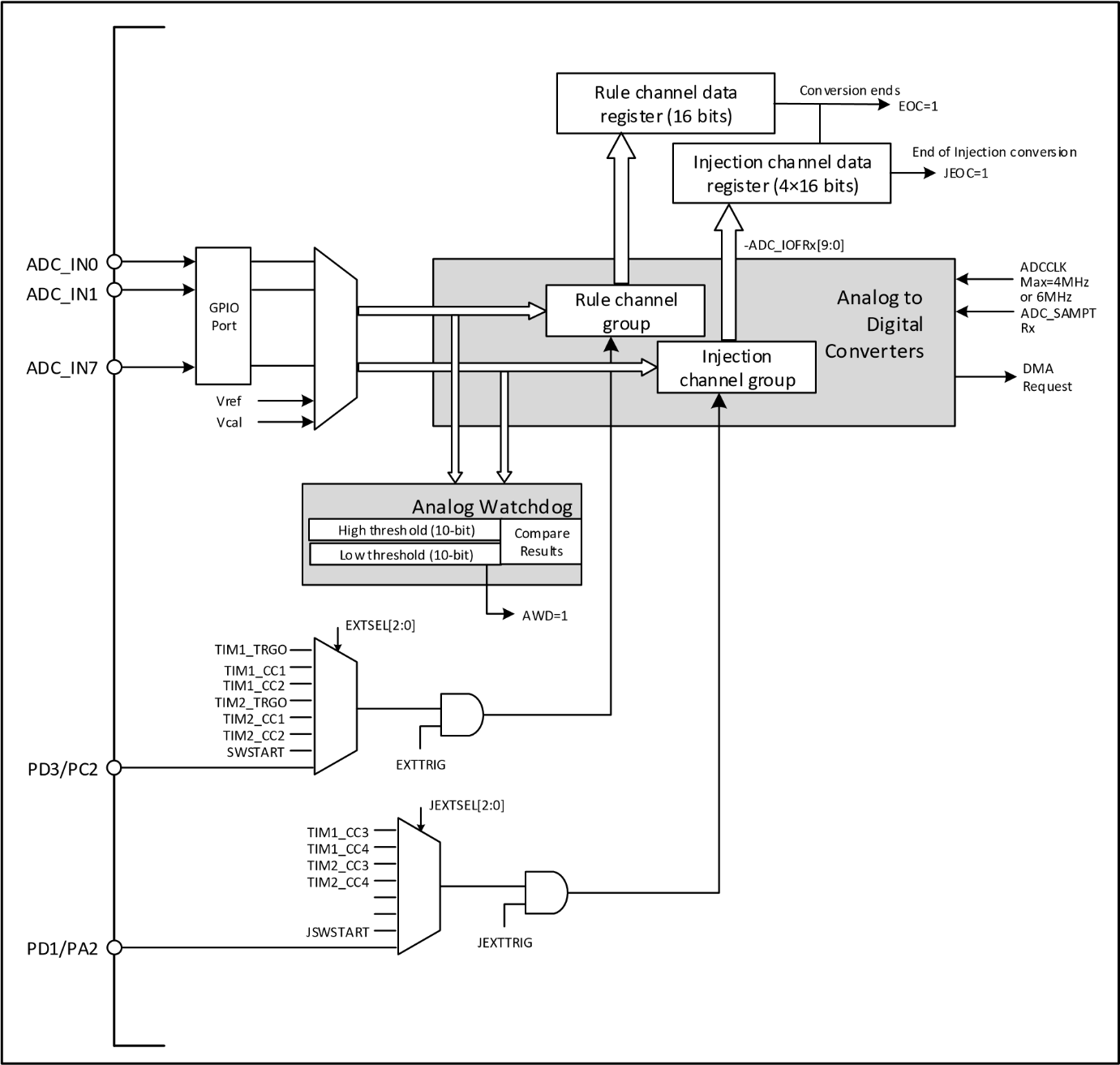
## 9.1 Основные возможности

* 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

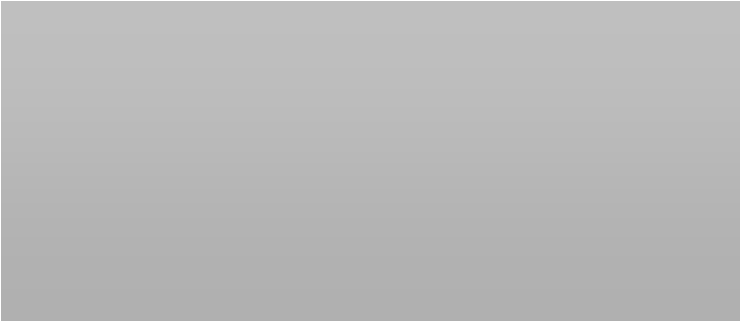
В режиме непрерывного преобразования новое преобразование начинается сразу после завершения предыдущего преобразования. Преобразование не останавливается на последнем канале выбранной группы, а продолжается с первого канала выбранной группы. Запускающими событиями в этом режиме являются внешние триггерные 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.

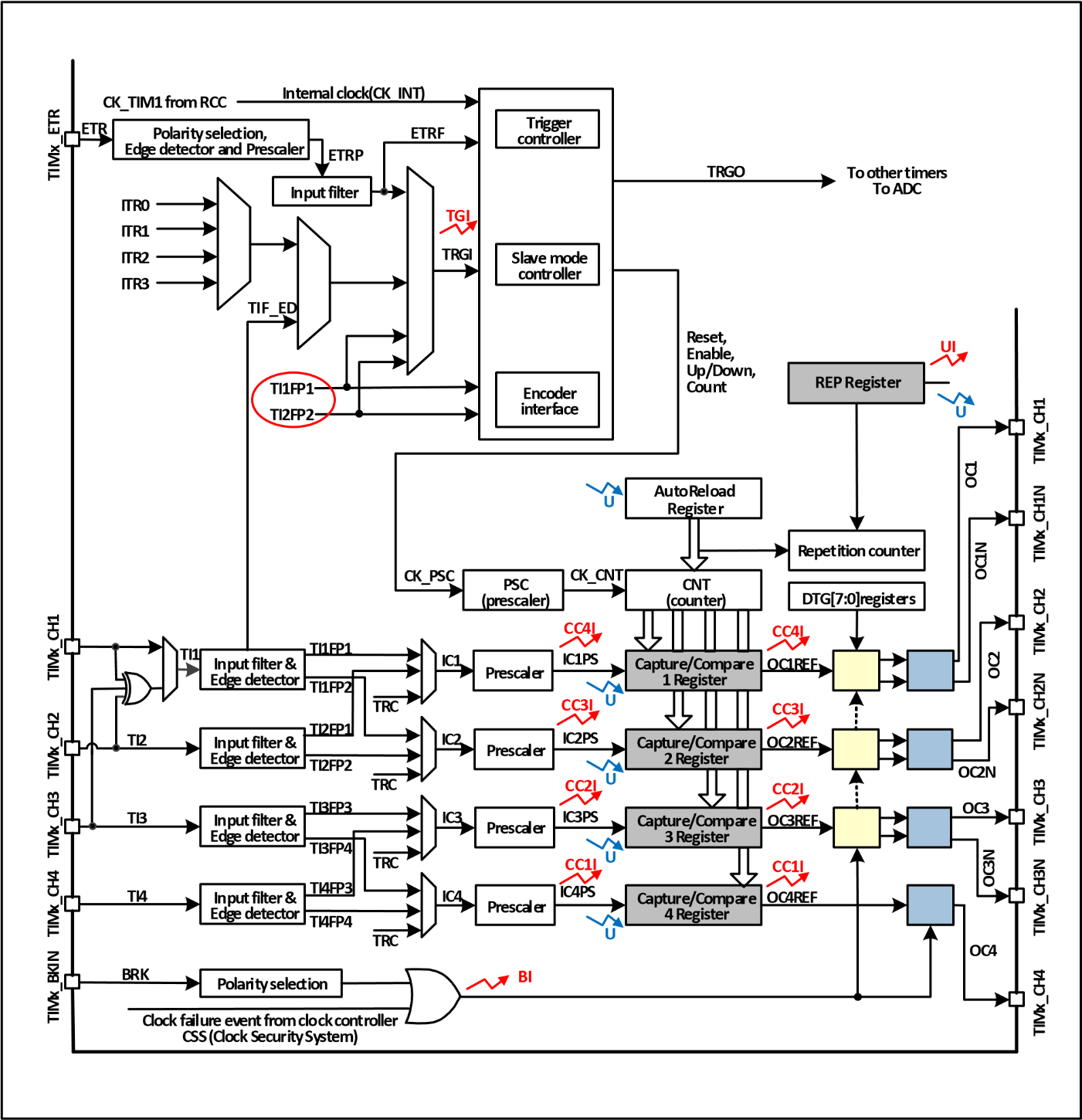
Таймер управления с расширенными возможностями может получать тактовый сигнал от шины HB (**CK\_INT**), внешнего входа тактового сигнала (**TIMx\_ETR**), других таймеров с выходом тактового сигнала (**ITRx**) или входа канала сравнения/захвата (**TIMx\_CHx**). Эти сигналы после различных операций фильтрации 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

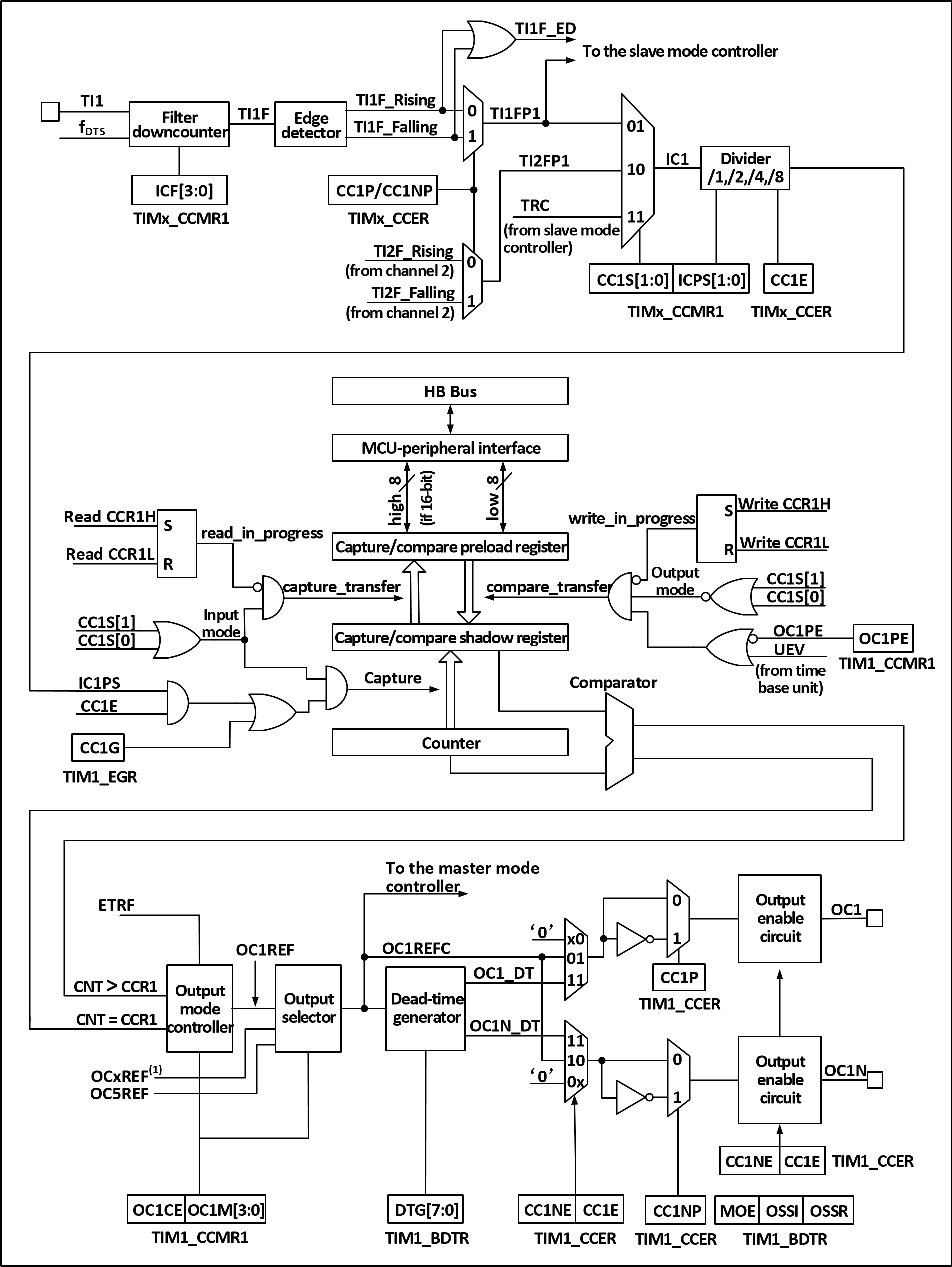
### 10.2.3 Счетчики и переферия

**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.

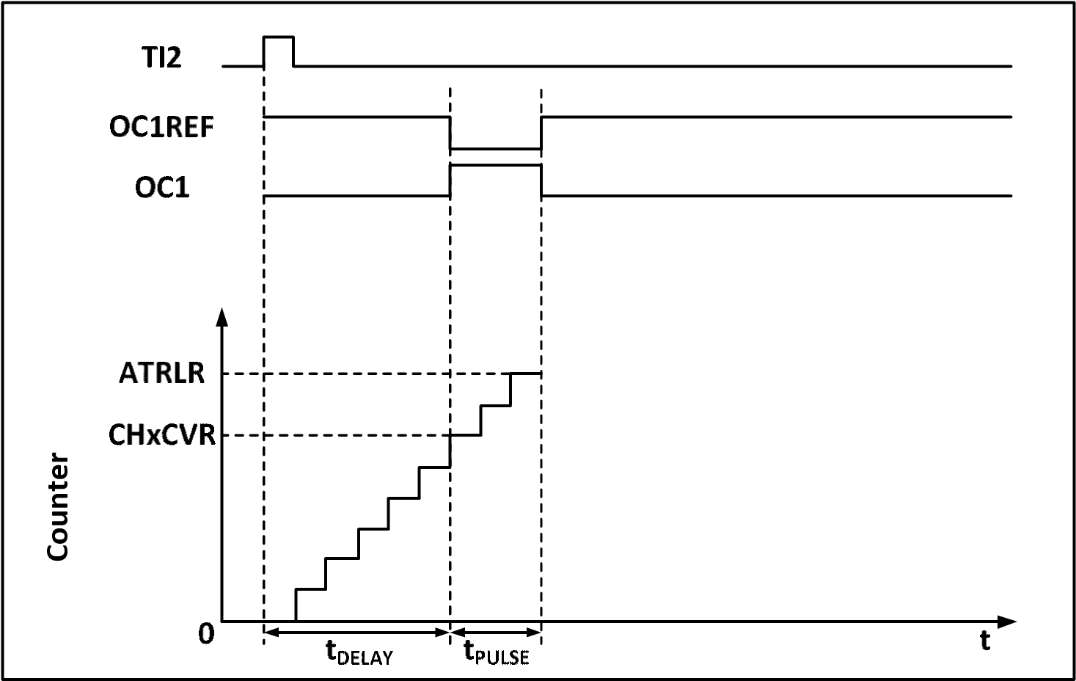
1. Биту **MOE** присваивается значение "0", и вывод устанавливается в неактивное, холостое или сбросовое состояние в зависимости от настройки бита **SOOI**.
2. После очистки бита **MOE** каждый выходной канал выводит уровень, определяемый битом **OISx**.
3. При использовании комплементарных выходов выходные 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

# Chapter 11 General-purpose Timer (GPTM)

The general-purpose timer module contains a 16-bit auto-reloadable timer, TIM2, for measuring pulse width or generating pulses of a specific frequency, PWM waves, etc. It can be used in automation control, power supply, etc.

## 11.1 Main Features

The main features of the general-purpose timer include.

* 16-bit auto-reload counter, supports incremental counting mode, decremental counting mode and incremental and decremental counting mode
* 16-bit prescaler with dynamically adjustable crossover factor from 1 to 65536
* Support four independent comparison capture channels
* Each comparison capture channel supports multiple operating modes, such as: input capture, output comparison, PWM generation, and single pulse output
* Support external signal control timer
* Support DMA in multiple modes
* Support incremental coding, cascading and synchronization between timers

## 11.2 Principle and Structure

Figure 11-1 Block diagram of the structure of the general-purpose timer

**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 Overview

As shown in Figure 11-1, the structure of the general-purpose timer can be roughly divided into three parts, namely the input clock part, the core counter part and the compare capture channel part.

The clock for the general-purpose timer can come from the HB bus clock (CK\_INT), from the external clock input pin (TIMx\_ETR), from other timers with clock output (ITRx), and from the input of the compare capture channel (TIMx\_CHx). These input clock signals become CK\_PSC clocks after various set filtering and dividing operations, etc., and are output to the core counter section. In addition, these complex clock sources can also be output as TRGO to other peripherals such as timers and ADCs.

The core of the general-purpose timer is a 16-bit counter (CNT). cK\_PSC is divided by a prescaler (PSC) to become cK\_CNT and then finally fed to the CNT, which supports incremental counting mode, decremental counting mode, and incremental and decremental counting mode, and has an auto-reload register (ATRLR) to reload the initialization value for the CNT at the end of each counting cycle.

The general-purpose timer has four sets of compare capture channels, each of which can input pulses from exclusive pins or output waveforms to pins, i.e., the compare capture channels support both input and output modes. The input of each channel of the compare capture register supports filtering, dividing, edge detection, and other operations, and supports mutual triggering between channels, and can also provide clock for the core counter CNT. Each comparison capture channel has a set of comparison capture registers (CHxCVR) that support comparison with the main counter (CNT) to output pulses.

### 11.2.2 Difference between General-purpose Timer and Advanced-control Timer

Compared to advanced-control timers, general purpose timers lack the following features.

1. The general-purpose timer lacks a repeat count register for counting the count cycles of the core counter.
2. The comparison capture channel of the general-purpose timer lacks deadband generation and has no complementary output.
3. The general-purpose timer does not have a brake signal mechanism.

### 11.2.3 Clock Input

This section discusses the source of CK\_PSC. The clock source portion of the overall block diagram of the general-purpose timer is captured here.

Figure 11-2 General-Purpose Timer CK\_PSC Source Block Diagram

**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]**

The optional input clocks can be divided into 4 categories.

1. Route of the external clock pin (ETR) input: ETR → ETRP → ETRF.
2. Internal HB clock input route: CK\_INT.
3. Route from the comparison capture channel pin (TIMx\_CHx): TIMx\_CHx → TIx → TIxFPx, this route is also used in encoder mode.
4. Input from other internal timers: ITRx.

The actual operation can be divided into 3 categories by determining the choice of input pulse for the SMS of the CK\_PSC source.

1. Selection of the internal clock source (CK\_INT).
2. External clock source mode 1.
3. External clock source mode 2.
4. Encoder mode.

All 4 clock source sources mentioned above can be selected by these 4 operations.

#### 11.2.3.1 Internal Clock Source (CK\_INT)

If the general-purpose timer is started when the SMS field is held at 000b, then it is the internal clock source (CK\_INT) that is selected as the clock. At this point CK\_INT is CK\_PSC.

#### 11.2.3.2 External Clock Source Mode 1

When the SMS domain is set to 111b, external clock source mode 1 is enabled. When external clock source 1 is enabled, TRGI is selected as the source for CK\_PSC. it is worth noting that the user also needs to select the source for TRGI by configuring the TS domain. the TS domain can select the following types of pulses as clock sources.

1. Internal trigger (ITRx, x is 0,1,2,3).
2. Comparison of the signal after capturing channel 1 through the edge detector (TI1F\_ED).
3. Comparison of signals TI1FP1, TI2FP2 of the capture channel.
4. The signal ETRF from the external clock pin input.

#### 11.2.3.3 External Clock Source Mode 2

Use external trigger mode 2 to count on every rising or falling edge of the external clock pin input. When the ECE position is set, the external clock source mode 2 is used. when using the external clock source mode 2, ETRF is selected as CK\_PSC. the ETR pin becomes ETRP after passing through the optional inverter (ETP), divider (ETPS), and then ETRF after passing through the filter (ETF).

With the ECE position bit and the SMS set to 111b, then it is equivalent to the TS selecting ETRF as the input.

#### 11.2.3.4 Encoder Mode

Setting the SMS to 001b, 010b, 011b will enable the encoder mode. Enabling encoder mode allows you to select a specific level in TI1FP1 and TI2FP2 to signal the output with another jump edge as the signal. This mode is used when an external encoder is used. Refer to Section 11.3.7 for specific functions.

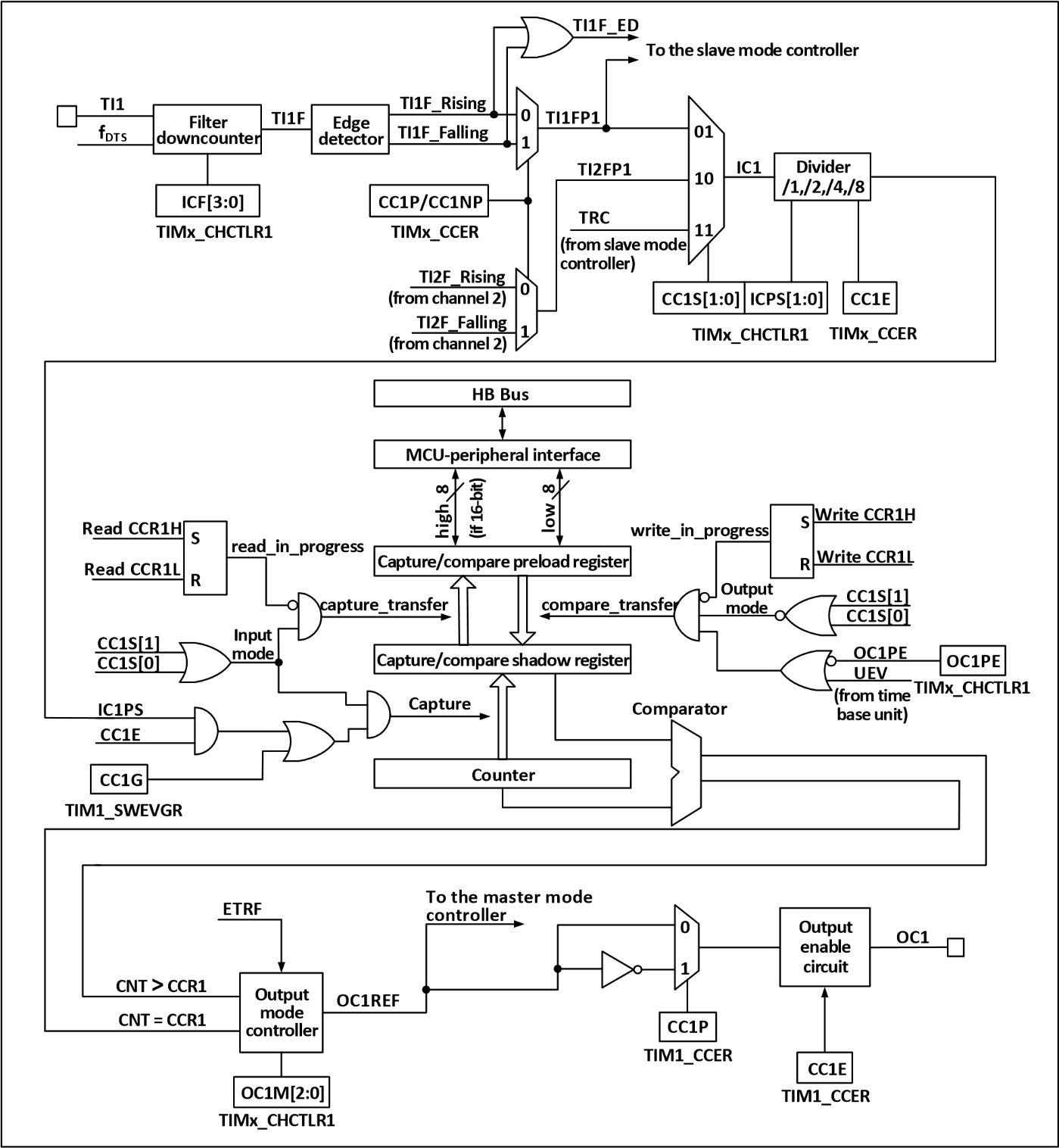
### 11.2.4 Counters and Peripherals

CK\_PSC is input to the prescaler (PSC) for dividing. the PSC is 16-bit and the actual dividing factor is equal to the value of R16\_TIMx\_PSC + 1. CK\_PSC goes through the PSC and becomes CK\_INT. changing the value of R16\_TIM1\_PSC does not take effect in real time, but is updated to the PSC after an update event. the update event includes a UG bit clear and reset.

### 11.2.5 Compare/capture Channels

The core of the compare/capture channel, which is the core of the timer to achieve complex functions, is the compare/capture register, supplemented by digital filtering, frequency division and inter-channel multiplexing in the peripheral input section, and comparator and output control in the output section. The structure block diagram of the compare/capture channel is shown in Figure 11-3.

Figure 11-3 Block diagram of the structure of the comparison capture channel



The signal is input from the channel x pin and optionally made as TIx (the source of TI1 can be more than CH1, see block diagram 10-1 of the timer), TI1 is passed through the filter (ICF[3:0]) to generate TI1F, and then divided into TI1F\_Rising and TI1F\_Falling through the edge detector, these two signals are selected (CC1P) to generate TI1FP1, TI1FP1 and TI2FP1 from channel 2 are sent together to CC1S to select to become IC1, which is sent to the comparison capture register after ICPS dividing.

The compare capture register consists of a preload register and a shadow register, and the read/write process operates only on the preload register. In capture mode, the capture occurs on the shadow register and is then copied to the preload register; in compare mode, the contents of the preload register are copied to the shadow register, and then the contents of the shadow register are compared to the core counter (CNT).

## 11.3 Functionality and Implementation

The complex functions of a general-purpose timer are implemented by manipulating the timer's compare capture channel, clock input circuitry, and counter and peripheral components. The clock input to the timer can be derived from multiple clock sources including the input to the compare capture channel. The operation of the compare capture host channel and clock source selection directly determines its function. The compare capture channel is bidirectional and can operate in both input and output modes.

### 11.3.1 Input Capture Mode

The input capture mode is one of the basic functions of the timer. The principle of input capture mode is that when a determined edge on the ICxPS signal is detected, a capture event is generated and the current value of the counter is latched into the compare capture register (R16\_TIMx\_CHCTLRx). The CCxIF (in R16\_TIMx\_INTFR) is set when a capture event occurs, and the corresponding interrupt or DMA is generated if enabled. If the CCxIF is already set when a capture event occurs, the CCxOF bit is set. the CCxIF can be cleared by software, or by hardware by reading the compare capture register. CCxOF is cleared by software.

An example of channel 1 to illustrate the steps to use the input capture mode is as follows.

1. Configure the CCxS domain to select the source of the ICx signal. For example, set it to 10b and select TI1FP1 as the source of IC1, not using the default setting, the CCxS domain defaults to making the comparison capture module the output channel.
2. Configure the ICxF domain to set the digital filter for the TI signal. The digital filter will sample the signal at a determined frequency, a determined number of times, and then output a hop. This sampling frequency and number of times is determined by ICxF.
3. Configure the CCxP bit to set the polarity of the TIxFPx. For example, keeping the CC1P bit low and selecting rising edge jumps.
4. Configure the ICxPS domain to set the ICx signal to be the crossover factor between ICxPS. For example, keeping ICxPS at 00b, without crossover.
5. Configure the CCxE bit to allow capturing the value of the core counter (CNT) into the compare capture register. Set the CC1E bit.
6. Configure the CCxIE and CCxDE bits as needed to determine whether to allow enable interrupts or DMA. This completes the comparison capture channel configuration.

When a captured pulse is input to TI1, the value of the core counter (CNT) is recorded in the compare capture register, CC1IF is set, and the CCIOF bit is set when CC1IF has been set before. If the CC1IE bit is set, then an interrupt is generated; if CC1DE is set, a DMA request is generated. An input capture event can be generated by software by way of writing the event generation register (R16\_TIMx\_SWEVGR).

### 11.3.2 Compare Output Mode

The compare output mode is one of the basic functions of the timer. The principle of the compare output mode is to output a specific change or waveform when the value of the core counter (CNT) agrees with the value of the compare capture register. the OCxM field (in R16\_TIMx\_CHCTLRx) and the CCxP bit (in R16\_TIMx\_CCER) determine whether the output is a definite high or low level or a level flip. The CCxIF bit is also set when a compare coherent event is generated. If the CCxIE bit is pre-set, an interrupt will be generated; if the CCxDE bit is pre-set, a DMA request will be generated.

To configure to compare output modes, proceed as follows.

1. Configuring the clock source and auto-reload value of the core counter (CNT).
2. Set the count value to be compared to the comparison capture register (R16\_TIMx\_CHxCVR). 3) Set the CCxIE bit if an interrupt needs to be generated.
3. Keep OCxPE at 0 to disable the preload register for the compare capture register.
4. Setting the output mode, setting the OCxM field and the CCxP bit.
5. Enable the output, setting the CCxE bit.
6. Setting the CEN bit to start the timer.

### 11.3.3 Forced Output Mode

The output pattern of the timer's compare capture channel can be forced by software to output a determined level without relying on comparison of the compare capture register's shadow register with the core counter. This is done by setting OCxM to 100b, which forces OCxREF to low, or by setting OCxM to 101b, which forces OCxREF to high.

Note that by forcing OCxM to 100b or 101b, the comparison process between the internal main counter and the compare capture register is still going on, the corresponding flags are still set, and interrupts and DMA requests are still being generated.

### 11.3.4 PWM Input Mode

The PWM input mode is used to measure the duty cycle and frequency of PWM and is a special case of the input capture mode. The operation is the same as input capture mode except for the following differences: PWM occupies two compare capture channels and the input polarity of the two channels is set to opposite, one of the signals is set as trigger input and SMS is set to reset mode.

For example, to measure the period and frequency of the PWM wave input from TI1, the following operations are required.

1. Set TI1 (TI1FP1) to be the input of IC1 signal. Set CC1S to 01b.
2. Set TI1FP1 to rising edge active. Holding CC1P at 0.
3. Set TI1 (TI1FP2) as the input of IC2 signal. Set CC2S to 10b.
4. Select TI1FP2 to set to falling edge active. Set CC2P to 1.
5. Select TI1FP1 as the source of the clock source. set TS to 101b.
6. Set the SMS to reset mode, i.e. 100b.
7. Enables input capture. cc1e and cc2e are set.

### 11.3.5 PWM Output Mode

PWM output mode is one of the basic functions of the timer. PWM output mode is most commonly used to determine the PWM frequency using the reload value and the duty cycle using the capture comparison register. Set 110b or 111b in the OCxM field to use PWM mode 1 or mode 2, set the OCxPE bit to enable the preload register, and finally set the ARPE bit to enable the automatic reload of the preload register. The value of the preload register can only be sent to the shadow register when an update event occurs, so the UG bit needs to be set to initialize all registers before the core counter starts counting. In PWM mode, the core counter and the compare capture register are always comparing, and depending on the CMS bit, the timer is able to output edge-aligned or center-aligned PWM signals.

 Edge alignment

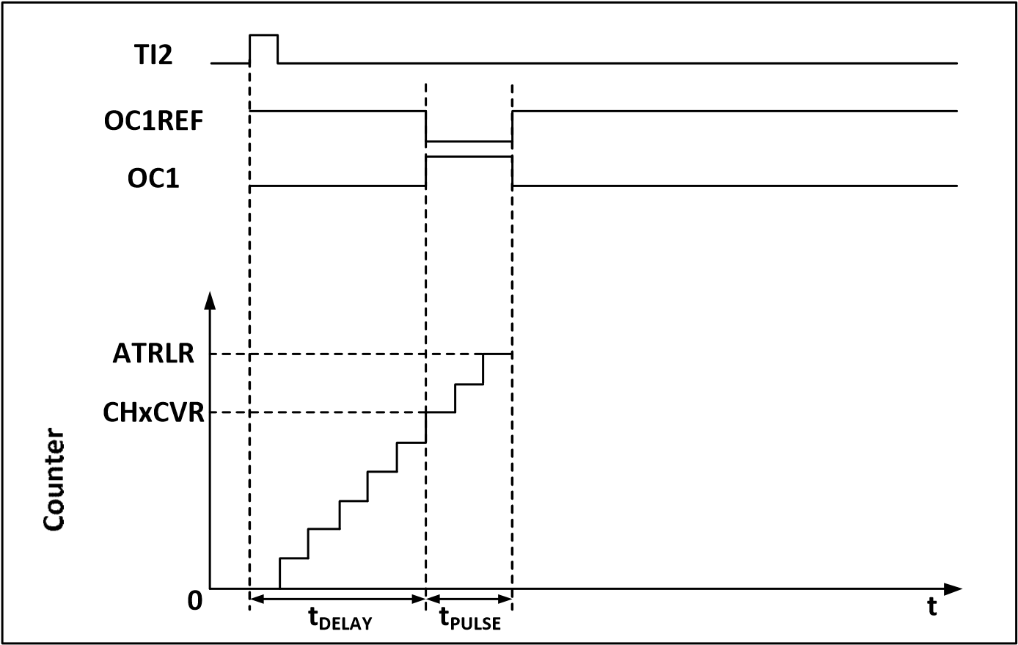
When using edge alignment, the core counter is incremented or decremented, and in the PWM mode 1 scenario, OCxREF rises to high when the core counter value is greater than the compare capture register; when the core counter value is less than the compare captureregister (for example, when the core counter grows to the value of R16\_TIMx\_ATRLR and reverts to full 0), OCxREF falls to low.  Central alignment

When using the central alignment modes, the core counter runs in alternating incremental and decremental count modes, and OCxREF performs rising and falling jumps when the values of the core counter and the compare capture register match. However, the comparison flags are set at different times in the three central alignment modes. When using the central alignment modes, it is best to generate a software update flag (set the UG bit) before starting the core counter.

### 11.3.6 Single Pulse Mode

The single pulse mode can respond to a specific event by generating a pulse after a delay, with programmable delay and pulse width. Setting the OPM bit stops the core counter when the next update event UEV is generated (counter flips to 0).

Figure 11-4 Event generation and impulse response



As shown in Figure 11-4, a positive pulse of length Tpulse needs to be generated on OC1 after a delay Tdelay at the beginning of a rising edge detected on the TI2 input pin.

1. Set TI2 to trigger. Setting the CC2S field to 01b to map TI2FP2 to TI2; setting the CC2P bit to 0b to set TI2FP2 as rising edge detection; setting the TS field to 110b to set TI2FP2 as trigger source; setting the SMS field to 110b to set TI2FP2 to be used to start the counter.
2. Tdelay is defined by the Compare Capture Register and Tpulse is determined by the value of the Auto Reload Value Register and the Compare Capture Register.

### 11.3.7 Encoder Mode

The encoder mode is a typical application of the timer and can be used to access the biphasic output of the encoder. The counting direction of the core counter is synchronized with the direction of the encoder's rotation axis, and each pulse output from the encoderwill add or subtract one from the core counter. To use the encoder, set the SMS field to 001b (count only on TI2 edge), 010b (count only on TI1 edge) or 011b (count on both TI1 and TI2 edges), connect the encoder to the input of the comparison capture channels 1 and 2, and set a reload value counter value, which can be set to a larger value. When in encoder mode, the internal compare capture register, prescaler, repeat count register, etc. of the timer are working normally. The following table shows the relationship between the counting direction and the encoder signal.

Table 11-1 Relationship between counting direction and encoder signal of timer encoder mode

### 11.3.8 Timer Synchronization Mode

Timers are capable of outputting clock pulses (TRGO) and also receiving inputs from other timers (ITRx). The source of ITRx (TRGO from other timers) is different for different timers. The timer internal trigger connections are shown in Table 11-2.

Table 11-2 GTPM internal trigger connection

From timer

ITR0(TS=000)

ITR1(TS=001)

ITR2(TS=010)

ITR3(TS=011)

TIM2

TIM1

TIM1

TIM2

### 11.3.9 Debug Mode

When the system enters the debug mode, the timer can be controlled to continue running or stop according to the setting of DBG module.

## 11.4 Register Description

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

The channel can be used in input (capture mode) or output (compare mode), and the direction of the channel is defined by the corresponding CCxS bit. The other bits of this register serve different purposes in input and output modes. OCxx describes the function of the channel in output mode and ICxx describes the function of the channel in input mode.

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

Offset address: 0x1C

The channel can be used in input (capture mode) or output (compare mode), and the direction of the channel is defined by the corresponding CCxS bit. The other bits of this register serve different purposes in input and output modes. OCxx describes the function of the channel in output mode and ICxx describes the function of the channel in input mode.

### 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)

# Chapter 12 Universal Synchronous Asynchronous Receiver Transmitter (USART)

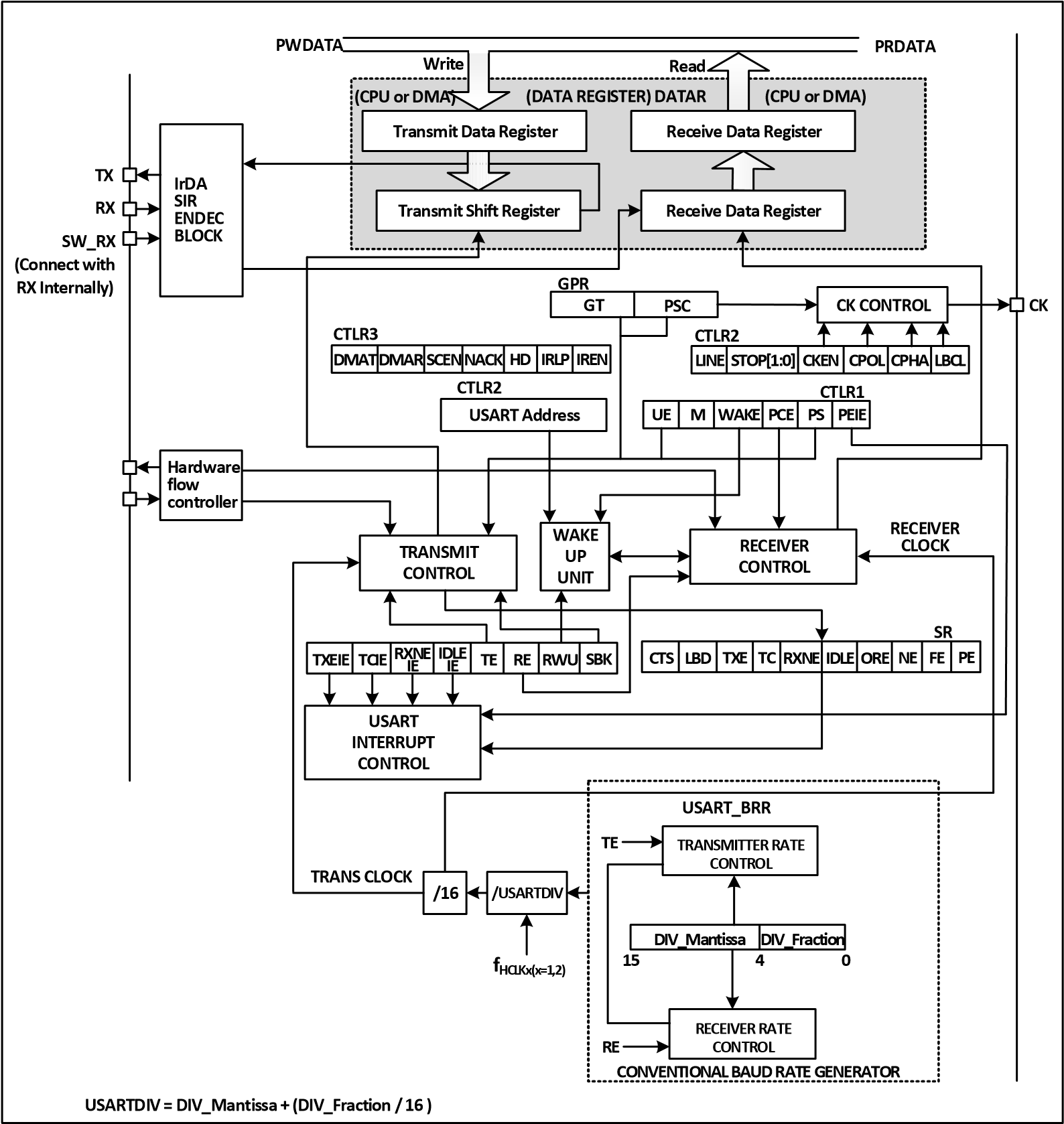
The module contains one Universal Synchronous Asynchronous Transceiver USART1.

## 12.1 Main Features

* Full-duplex or half-duplex synchronous or asynchronous communication  NRZ data format
* Fractional baud rate generator, up to 3Mbps
* Programmable data length
* Configurable stop bits
* Support LIN, IrDA encoders, smart cards
* DMA support
* Multiple interrupt sources

## 12.2 Overview

Figure 12-1 Block diagram of a general-purpose synchronous/asynchronous transceiver



When TE (transmit enable bit) is set, the data in the transmit shift register is output on the TX pin and the clock is output on the CK pin. When transmitting, the first bit shifted out is the least significant bit and each data frame starts with a low start bit, then the transmitter sends an 8- or 9-bit data word depending on the setting on the M (word length) bit, and finally a configurable number of stop bits. If equipped with a parity check bit, the last bit of the data word is the check bit. After the TE is set an idle frame is sent, which is 10 or 11 bits high and contains the stop bit. The disconnect frame is 10 or 11 bits low followed by the stop bit.

## 12.3 Baud Rate Generator

The baud rate of the transceiver = HCLK/(16\*USARTDIV), HCLK is the clock of HB. The value of USARTDIV is determined by the two fields DIV\_M and DIV\_F in USART\_BRR, which is calculated by the formula The formula is as follows.

USARTDIV = DIV\_M+(DIV\_F/16)

It is important to note that the bit rate generated by the baud rate generator may not always generate exactly the baud rate required by the user, and there may be deviations. In addition to taking as close a value as possible, a way to reduce the deviation is to increase the HB clock. For example, if you set the baud rate to 115200bps, the value of USARTDIV is set to 39.0625, which will give you a baud rate of exactly 115200bps at the highest frequency, but if you need a baud rate of 921600bps, the calculated USARTDIV is 4.88, but the closest value filled in USART\_BRR is actually only 4.875. 4.875, the actual baud rate is 923076bps, which is 0.16% error. When the serial waveform sent by the sender is transmitted to the receiver, the baud rate of the receiver and the sender is subject to some error. The error mainly comes from three aspects: the actual baud rate of the receiver and the sender is not the same; the receiver and the sender's clock has errors; the waveform in the line generated by the change. Peripheral module receiver is a certain receiving tolerance, when the sum of the above three aspects of the total deviation is less than the module's tolerance limit, the total deviation does not affect the transmission and reception. The tolerance limitof the module is affected by whether to use fractional baud rate and M-bit (data field word length), using fractional baud rate and using 9-bit data field length will reduce the tolerance limit, but not less than 3%.

## 12.4 Synchronous Mode

Synchronous mode allows the system to output a clock signal when using the USART module. When synchronous mode is enabled to send data externally, the CK pin will output the clock externally at the same time.

The way to turn on the synchronous mode is to the CLKEN position bit in control register 2 (R16\_USARTx\_CTLR2), but also need to turn off the LIN mode, smart card mode, infrared mode and half duplex mode, i.e. ensure that the SCEN, HDSEL and IREN bits are in reset, these three in control register 3 (R16\_USARTx\_CTLR3).

The key point of using synchronous mode is the clock output control. There are several points to note.

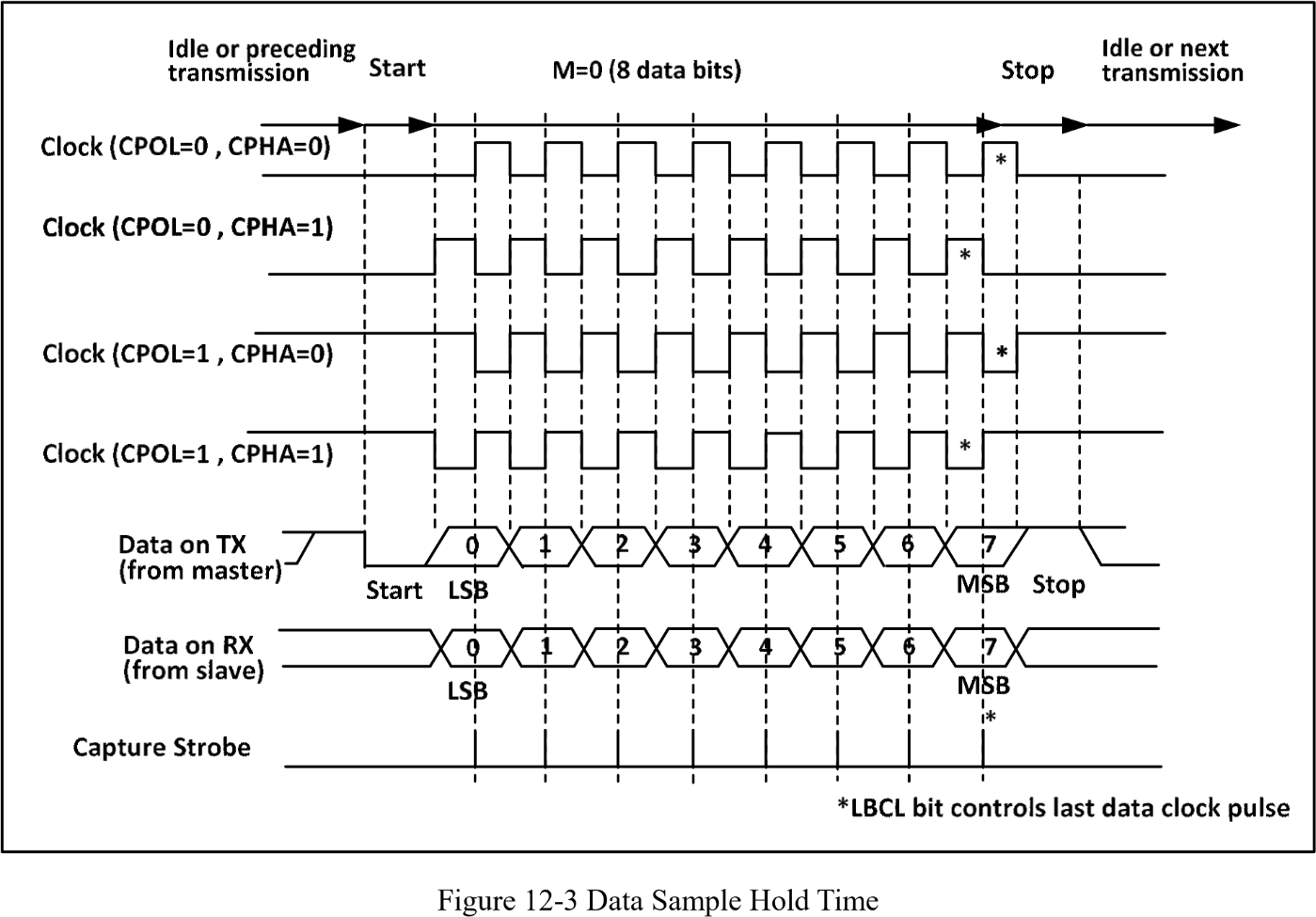
a) The USART module synchronization mode works only in the main mode, i.e. the CK pin outputs only the clock and does not receive inputs.

Outputs a clock signal only when data is output on the TX pin.

The LBCL bit determines whether the clock is output when the last data bit is sent, the CPOL bit determines the polarity of the clock, and the CPHA determines the phase of the clock. These three bits are in control register 2 (R16\_USARTx\_CTLR2), which needs to be set when TE and RE are not enabled, see Figure 12-2 for the differences.

The receiver will only sample at the output clock in synchronous mode, requiring a certain amount of signal build time and hold time from the device, as shown in Figure 12-3.

Figure 12-2 USART clock timing example (M=0)



**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 1-wire Half-duplex Mode

Half-duplex mode supports the use of a single pin (TX pin only) for receive and transmit, with the TX and RX pins connected internally on the chip.

The way to turn on the half-duplex mode is to set the HDSEL position bit in control register 3 (R16\_USARTx\_CTLR3), but it is also necessary to turn off the LIN mode, smart card mode, IR mode and synchronous mode, i.e. to ensure that the SCEN, CLKEN and IREN bits are in reset, which are in control registers 2 and 3 (R16\_USARTx\_CTLR2 and R16\_USARTx\_CTLR3).

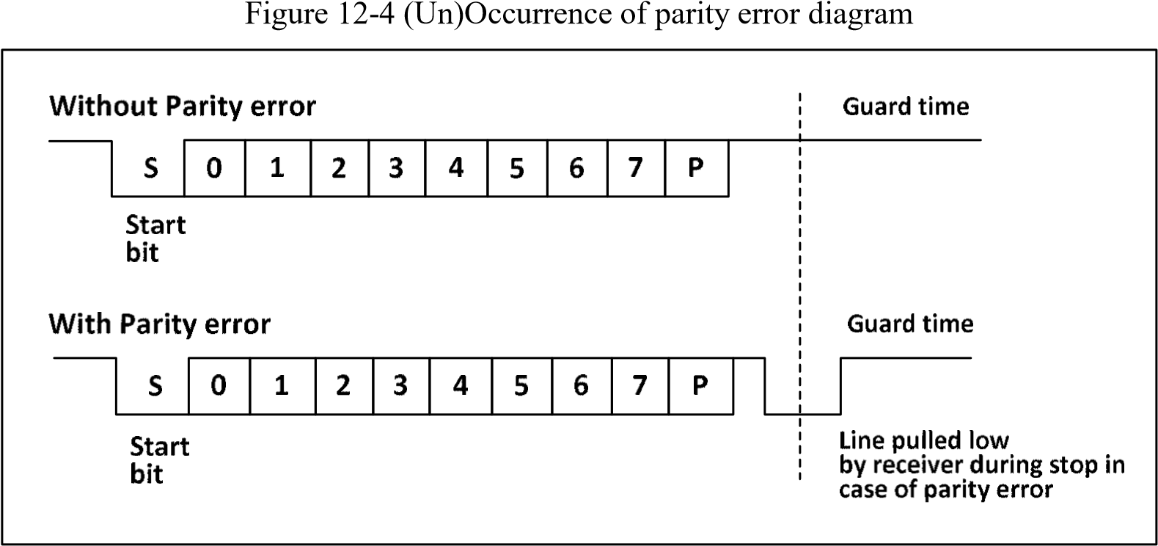
After setting to half duplex mode, you need to set the IO port of TX to open-drain output high mode. With TE set, the data will be sent out as soon as it is written to the data register. Special attention should be paid to the fact that the half-duplex mode may cause bus conflicts when multiple devices use a single bus to send and receive, which needs to be avoided by the user with software itself.

## 12.6 Smart Card

Smart card mode supports ISO7816-3 protocol access to smart card controllers.

The smart card mode is turned on by setting the SCEN position bit in control register 3 (R16\_USARTx\_CTLR3), but it is also necessary to turn off LIN mode, half duplex mode and IR mode, i.e. to ensure that the LINEN, HDSEL and IREN bits are in reset, but CLKEN can be turned on to output the clock, these bits are in control registers 2 and 3 (R16\_USARTx\_CTLR2 and R16\_USARTx\_CTLR3).

To support smart card mode, USART should be set to 8 bits of data plus 1 bit of parity, and its stop bit is recommended to be configured to 1.5 bits for both transmit and receive. Smart card mode is a 1-wire halfduplex protocol that uses the TX line for data communication and should be configured as an open-drain output plus a pull. When the receiver receives a frame of data and detects a parity error, it sends a NACK signal, i.e., it actively pulls the TX down by one cycle during the stop bit, and the sender detects the NACK signal, which generates a frame error whereby the application can retransmit. Figure 17-4 shows the waveforms on the TX pin in the correct case and in the case of a parity error. the TC flag (transmit complete flag) of the USART can delay the GT (protection time) generation by one clock, and the receiver will not recognize the NACK signal it sets as the start bit.



In smart card mode, the waveform output from the CK pin when enabled has nothing to do with communication; it simply clocks the smart card with the value of the AHB clock followed by a five-bit settable clock division (twice the value of the PSC, up to 62 divisions).

## 12.7 IrDA

The USART module supports control of IrDA infrared transceivers for physical layer communication. The LINEN, STOP, CLKEN, SCEN and HDSEL bits must be cleared to use IrDA. NRZ (non-return to zero) coding is used between the USART module and the SIR physical layer (infrared transceiver) and is supported up to 115200 bps rates.

IrDA is a half-duplex protocol, if UASRT is sending data to SIR physical layer, then IrDA decoder will ignore the newly sent IR signal, if USART is receiving data from SIR, then SIR will not accept the signal from USART. the level logic of USART to SIR and SIRto USART is different. In SIR receive logic, the high level is 1 and the low level is 0, but in SIR send logic, the high level is 0 and the low level is 1.

## 12.8 DMA

The USART module supports DMA function, which can be used to achieve fast and continuous sending and receiving. When DMA is enabled, the DMA writes data from the set memory space to the transmit buffer when TXE is set. When using DMA to receive, each time RXNE is set, DMA transfers the data in the receive buffer to a specific memory space.

## 12.9 Interruptions

The USART module supports a variety of interrupt sources, including transmit data register empty (TXE), CTS, transmit complete (TC), receive data ready (RXNE), dataoverflow (ORE), line idle (IDLE), parity error

(PE), disconnect flag (LBD), noise (NE), overflow for multi-buffered communication (ORT), and frame error (FE), among others.

Table 12-1 Relationship between interrupts and corresponding enable bits

Interrupt source Enable bit

Transmit data register empty (TXE) TXEIE

Allowed to send (CTS) CTSIE

Transmission complete（TC） TCIE

Received data ready to be read

（RXNE） RXNEIE

Overrun error detected（ORE）

Idle line detected（IDLE） IDLEIE

Parity error（PE） PEIE

Break flag（LBD） LBDIE

Noise flag（NE）

Overflow of multi-buffered

communication (ORE) EIE

Frame error for multibuffered communication (FE)

## 12.10 Register Description

Table 12-2 USART-related registers list

Name Offset address Description Reset value

R32\_USART\_STATR 0x40013800 UASRT status register 0x000000C0

R32\_USART\_DATAR 0x40013804 UASRT data register 0x000000XX

R32\_USART\_BRR 0x40013808 UASRT baud rate register 0x00000000

R32\_USART\_CTLR1 0x4001380C UASRT control register 1 0x00000000

R32\_USART\_CTLR2 0x40013810 UASRT control register 2 0x00000000

R32\_USART\_CTLR3 0x40013814 UASRT control register 3 0x00000000

R32\_USART\_GPR 0x40013818 UASRT protection time and prescaler register 0x00000000

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

Offset address: 0x00

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

Reserved

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

Reserved CTS LBD TXE TC RXNE IDLE ORE NE FE PE

Bit Name Access Description Reset value

[31:10] Reserved RO Reserved 0

CTS state change flag. If the CTSE bit is set, this bit will be set high by hardware when the nCTS output state changes. It is cleared to zero by

9 CTS RW0 software. If the CTSIE bit is already set, an 0

interrupt will be generated.

1: the presence of changes on the nCTS state line. 0: No change on the nCTS state line.

LIN Break detection flag. This bit is set by

8 LBD RW0 hardware when a LIN Break is detected. It is 0

cleared by software.

If LBDIE is already set, an interrupt will be generated.

1: LIN Break detected.

0: No detection of pending LIN Break.

Send data register empty flag. This bit is set by hardware when the data in the TDR register is transferred to the shift register by hardware. If

TXEIE is already set, an interrupt will be

7 TXE RO generated to perform a write operation to the data 1

register and this bit will be reset.

1: the data has been transferred to the shift register. 0: The data has not been transferred to the shift register.

Send completion flag. When a frame containing data is sent and TXE is set, the hardware will set this bit, and if TCIE is set, an interrupt will be

generated, and the software will clear this bit by

6 TC RW0 1

reading it and then writing to the data register. It is also possible to write 0 directly to clear this bit.

1: Sending completed.

0: Sending is not yet complete.

Read data register non-empty flag, this bit is set by hardware when data in the shift register is transferred to the data register. If RXNEIE is already set, a corresponding interrupt is also

5 RXNE RW0 generated. A read operation of the data register 0

clears this bit. It is also possible to clear the bit by writing a 0 directly.

1: Data received and able to be read out.

0: The data has not been received.

Bus idle flag. When the bus is idle, this bit will be set by hardware. If IDLEIE is already set, the corresponding interrupt will be generated. The operation of reading the status register and then

4 IDLE RO reading the data register will clear this bit. 0

1: The bus is idle.

0: No bus idle is detected.

*Note: This bit will not be set again until RXNE is set.*

Overload error flag. This bit will be set when there is data in the receive shift register that needs to be transferred to the data register, but there is still data in the receive field of the data register that has not been read out. If RXNEIE is set, the corresponding interrupt will also be generated.

3 ORE RO 1: Occurrence of an overload error. 0

0: No overload error.

*Note: In case of an overload error, the value of the data register is not lost, but the value of the shift register is overwritten. If the EIEable bit is set, the ORE flag position bit generates an interrupt in multi-buffer communication mode.*

Noise error flag. It is set by hardware when the noise error flag is detected. The operation of reading the status register and then reading the data register resets this bit.

2 NE RO 0

1: Noise detected.

0: No noise is detected.

*Note: This bit does not generate an*

*interrupt. If the EIE bit is set, the FE flag position*

*bit generates an interrupt in multi-buffer communication mode.*

Frame error flag. This bit will be set by hardware when a synchronization error, excessive noise or disconnect character is detected. Reading this bit and then reading the data register operation will reset this bit.

1 FE RO 0

1: Frame error detected.

0: No frame error detected.

*Note: This bit will not generate an interrupt. If the EIE bit is set, the FE flag position bit will generate an interrupt in multi-buffer communication mode.* Checksum error flag. In receive mode, hardware sets this bit if a parity check error is generated. A read of this bit and then a read of the data register operation resets this bit. Before clearing this bit,

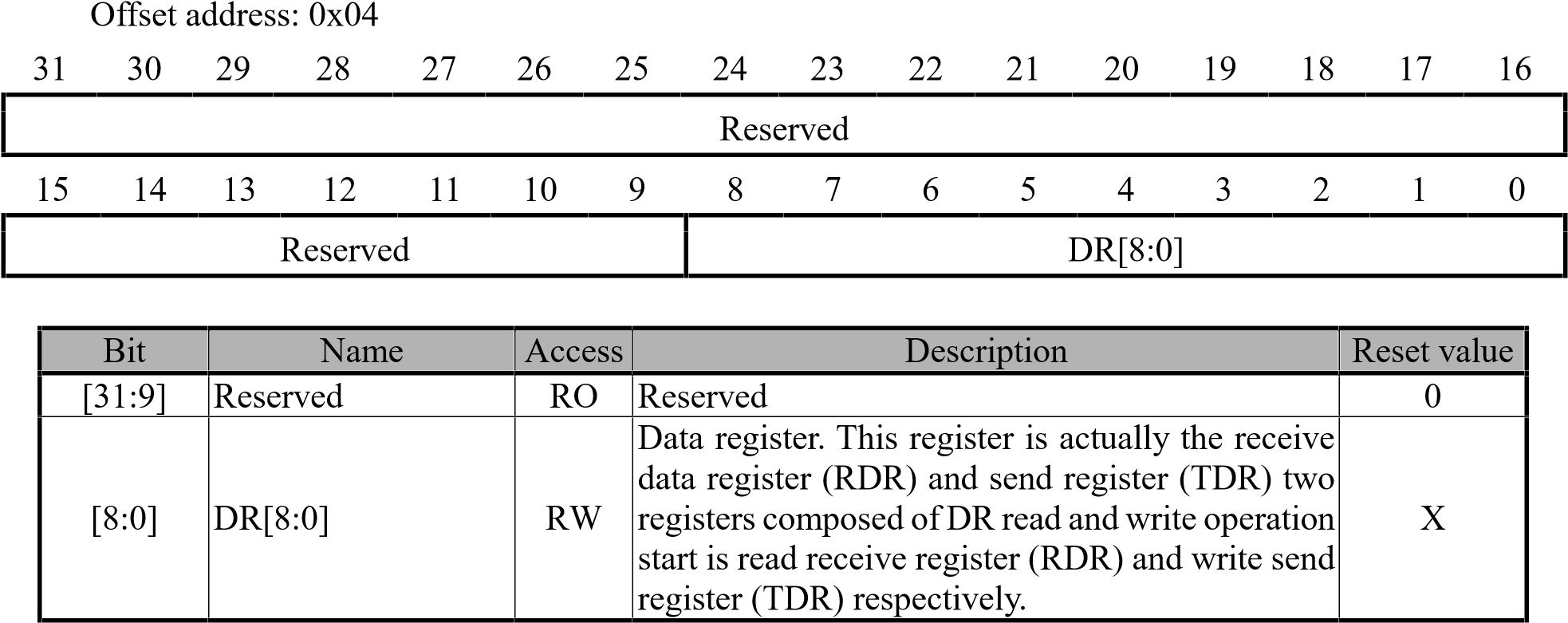
0 PE RO software must wait for the RXNE flag bit to be set. 0

If the PEIE has been set previously, then this bit being set generates a corresponding interrupt.

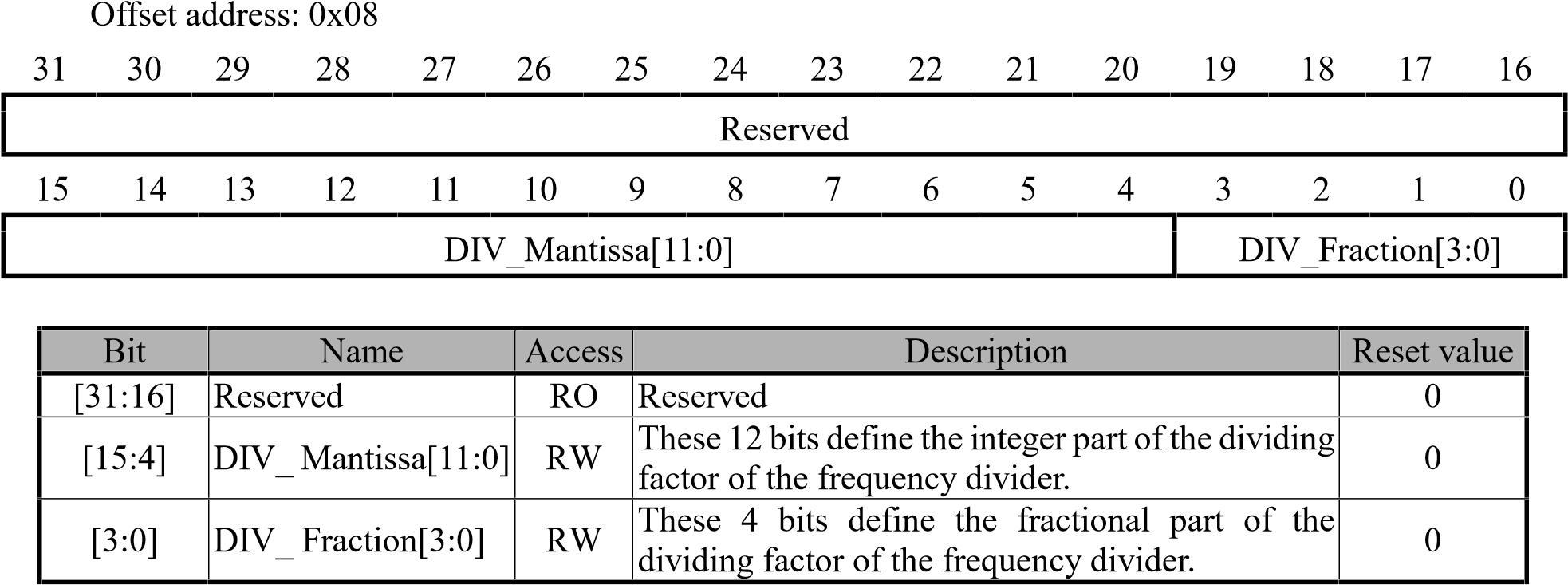
1: A parity error.

0: No inspection error.

### 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)

Offset address: 0x0C

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

Reserved

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

Reserved UE M WAKE PCE PS PEIE TXEIE TCIE RXNEIE IDLEIE TE RE RWU SBK

Bit Name Access Description Reset value

[31:14] Reserved RO Reserved 0

USART enable bit. When this bit is set, both the

13 UE RW USART divider and the output stop working after 0

the current byte transfer is completed. Word long bit.

12 M RW 0

1: 9 data bits; 0: 8 data bits.

Wake-up bit. This bit determines the method of

11 WAKE RW waking up the USART. 0

1: Address marker; 0: Bus idle.

The parity bit is enabled. For the receiver, it is the parity check of the data; for the sender, it is the

10 PCE RW insertion of the parity bit. Once this bit is set, the 0

parity bit enable will take effect only after the current byte transmission is completed.

Parity selection. 0 means even parity, 1 means odd

parity. When this bit is set, the parity bit enable

9 PS RW 0

will take effect only after the current byte transmission is completed.

Parity check interrupt enable bit. This bit indicates

8 PEIE RW 0

that parity check error interrupts are allowed.

TXE interrupt enable. This bit indicates that a

7 TXEIE RW 0

TXE interrupt is allowed to be generated.

Transmit completion interrupt enable. This bit

6 TCIE RW indicates that the transmit completion interrupt is 0

allowed to be generated.

RXNE interrupt enable. This bit indicates that a

5 RXNEIE RW 0

RXNE interrupt is allowed to be generated.

IDLE interrupt enable. This bit allows IDLE

4 IDLEIE RW 0

interrupt to be generated.

Transmitter enable. Setting this bit will enable the

3 TE RW 0

transmitter.

Receiver enable. Setting this bit enables the

2 RE RW receiver, which starts detecting the start bit on the 0

RX pin.

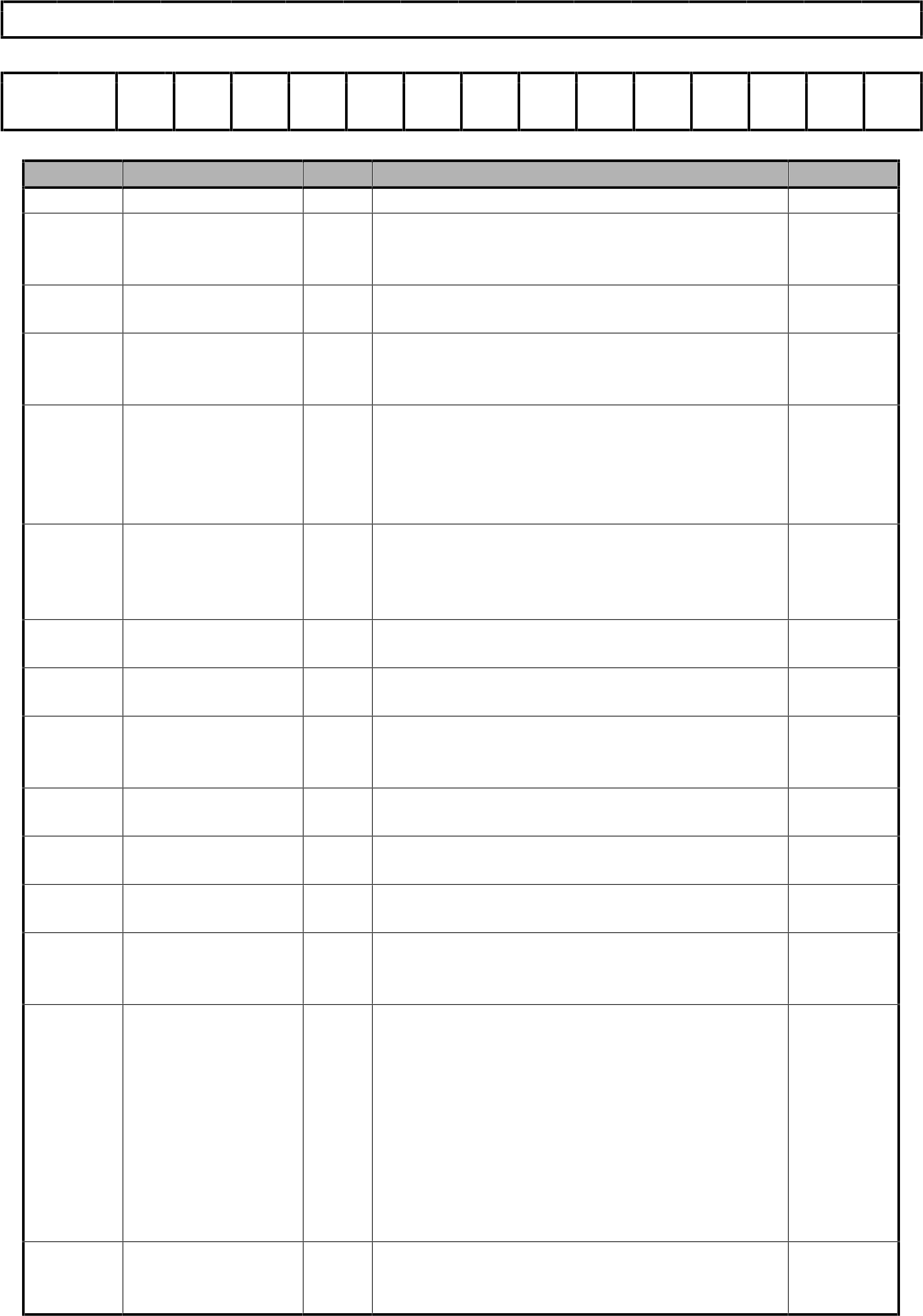
Receiver wakeup. This bit determines whether to place the USART in silent mode.

1: The receiver is in silent mode.

0: The receiver is in normal operation mode.

*Note 1: Before setting the RWU bit, the USART*

1 RWU RW 0

*needs to receive a data byte first, otherwise it cannot be woken up by bus idle in silent mode. Note 2: When configured as address mark wakeup, the RWU bit cannot be modified by software when RXNE is set.*

Send break bit.

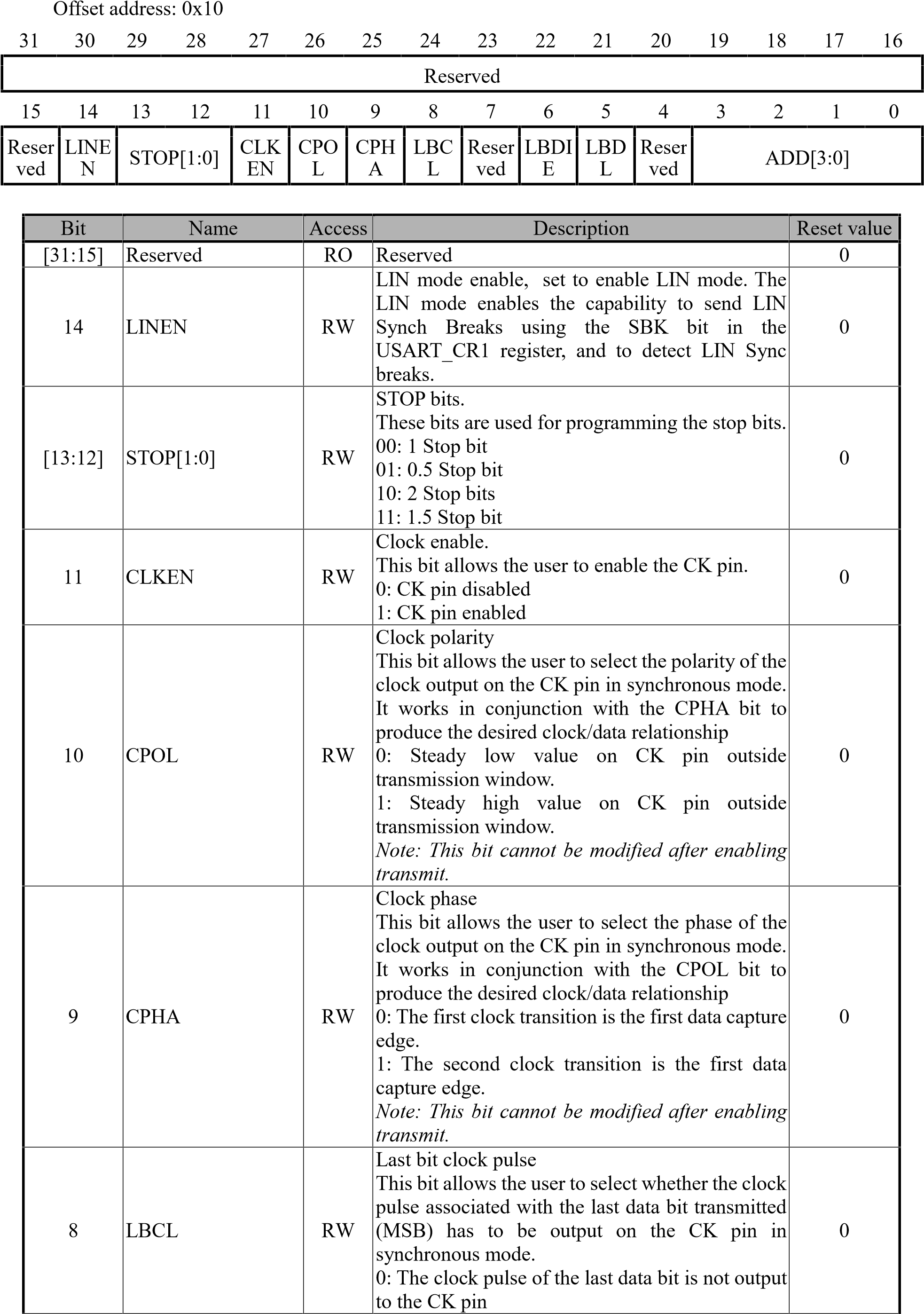
0 SBK RW Set this bit to send break character. It is reset by 0

hardware on the stop bit of the break frame.

1:

Send; 0: Do not send.

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



1: The clock pulse of the last data bit is output to the CK pin

*Note: This bit cannot be modified after enabling transmit.*

7 Reserved RW Reserved 0

LIN Break detection interrupt enable, this

6 LBDIE RW 0

position bit enables interrupts caused by LBD. LIN Break detection length, this bit is used to select whether the Break character detection is 11

5 LBDL RW bits or 10 bits. 0

1: 11-bit Break character detection.

0: 10-bit Break character detection.

4 Reserved RW Reserved 0

Address of the USART node, this bit-field gives

the address of the USART node. This is used in

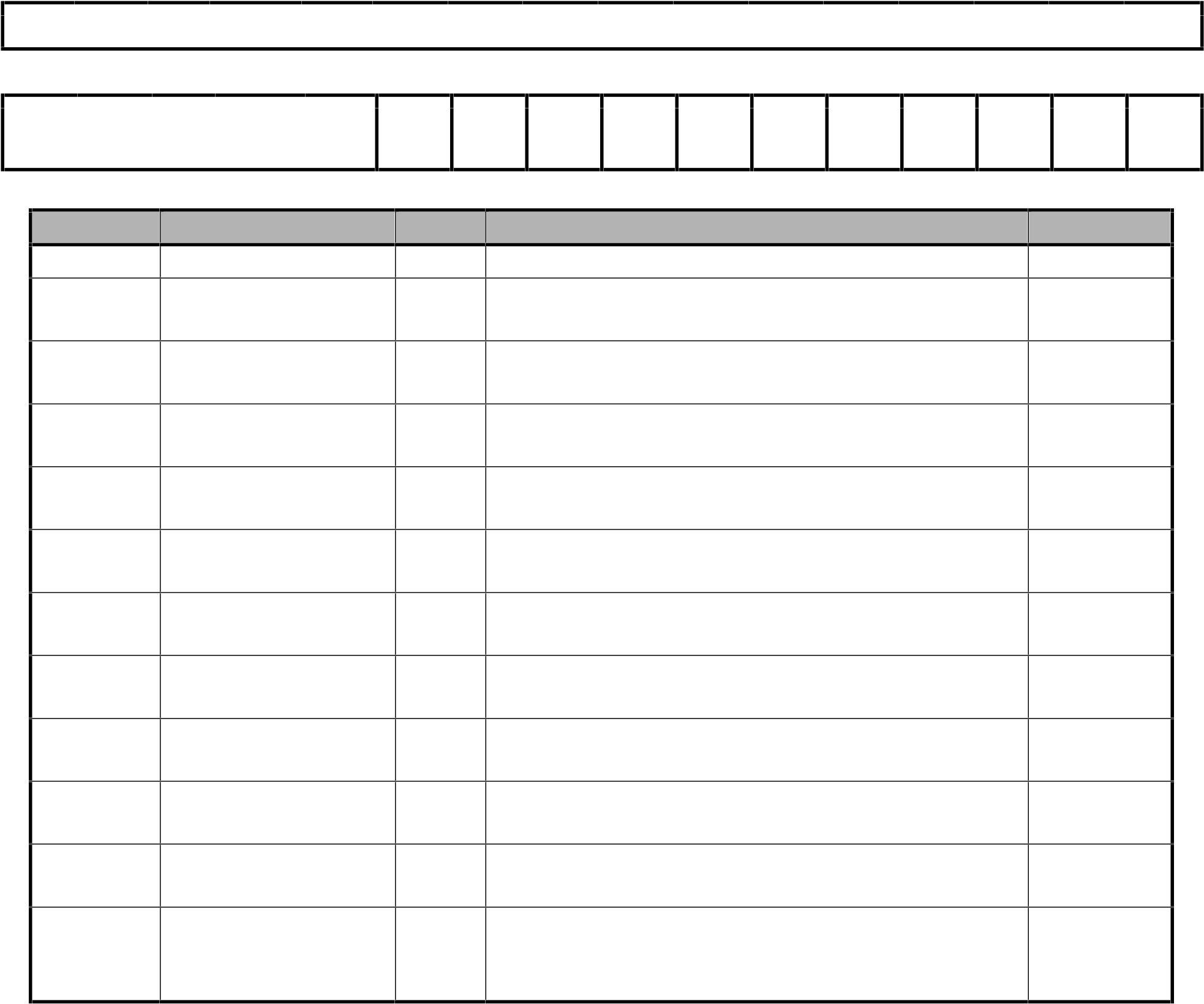
[3:0] ADD[3:0] RW 0

multiprocessor communication during mute mode, for wake up with address mark detection.

|  |  |  |
| --- | --- | --- |
| **12.10.6 USART Control Register 3 (USART\_CTLR3)**  Offset address: 0x14 |  |  |
| 31 30 29 28 27 26 25 24 23 22 | 21 20 19 18 17 | 16 |

Reserved

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

 Reserved CTSIE CTSE RTSE DMAT DMAR SCEN NACK HDSEL IRLP IREN EIE

Bit Name Access Description Reset value

[31:11] Reserved RO Reserved 0

CTS interrupt enable bit, when this bit is set, an

10 CTSIE RW 0

interrupt will be generated when CTS is set.

CTS enable bit, setting this bit will enable CTS

9 CTSE RW 0

flow control.

RTS enable bit, setting this bit will enable RTS

8 RTSE RW 0

flow control.

DMA transmit enable bit. This bit 1 uses DMA

7 DMAT RW 0

when transmitting.

DMA receive enable bit. This position 1 uses

6 DMAR RW 0

DMA on receive.

Smartcard mode enable bit, set to 1 to enable

5 SCEN RW 0

smart card mode.

Smartcard NACK enable bit, set this bit to send

4 NACK RW 0

NACK in case of check error.

Half-duplex selection bit, set this bit to select half-

3 HDSEL RW 0

duplex mode.

IrDA low-power bit, set this bit to enable low-

2 IRLP RW 0

power mode when IrDA is selected.

IrDA enable bit, set this bit to enable infrared

1 IREN RW 0

mode.

Error interrupt enable bit, when set, generates an

0 EIE RW interrupt if FE, ORE or NE is set provided that 0

DMAR is set.

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

Offset address: 0x18

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

Reserved

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

GT[7:0] PSC[7:0]

Bit Name Access Description Reset value

[31:16] Reserved RO Reserved 0

Guard time value. This bit-field gives the Guard

time value in terms of number of baud clocks. This

[15:8] GT[7:0] RW 0

is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.

Prescaler value field.

In IrDA Low-power mode, the source clock is divided by this value (all 8 bits valid), with a value of 0 indicating retention.

[7:0] PSC[7:0] RW In normal IrDA mode, this bit can only be set to 1. 0

In smartcard mode, the source clock is divided by twice this value (valid in the lower 5 bits) to clock the smart card, with a value of 0 indicating retention.

# Chapter 13 Inter-integrated Circuit (I2C) interface

The Internal Integrated Circuit Bus (I2C) is widely used for communication between microcontrollers and sensors and other off-chip modules, it supports multi-master and multi-slave modes, and can communicate at 100KHz (standard) and 400KHz (fast) using only two lines (SDA and SCL). Timing and DMA, with CRC checksum function.

## 13.1 Main Features

* Support master and slave modes
* Support 7-bit or 10-bit addresses
* Slave devices support dual 7-bit addresses
* Support two speed modes: 100KHz and 400KHz
* Multiple status modes, multiple error flags
* Support extended clock function
* 2 interrupt vectors
* DMA support
* Support PEC
* SMBus compatible

## 13.2 Overview

I2C is a half-duplex bus that can only operate in one of the following four modes at the same time: master device transmit mode, master device receive mode, slave device transmit mode and slave device receive mode. the I2C module works in slave mode by default and automatically switches to master mode when a start condition is generated and to slave mode when arbitration is lost or a stop signal is generated. the I2C module supports multi-master functionality. When working in master mode, the I2C module actively emits data and addresses. Both data and address are transmitted in 8-bit units, with the high bit before and the low bit after. After the start event is a one-byte (in 7-bit address mode) or two-byte (in 10-bit address mode) address, and for every 8-bit data or address sent by the host, the slave needs to reply with an answer ACK, which pulls the SDA bus low, as shown in Figure 13-1.

Figure 13-1 I2C Timing Diagram



In order to work properly the I2C must be fed with the correct clock, which is a minimum of 2MHz in standard mode and 4MHz in fast mode.

Figure 13-2 shows the functional block diagram of the I2C module.

**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 Master Mode

In master mode, the I2C module dominates the data transfer and outputs the clock signal, and the data transfer starts with a start event and ends with an end event. The steps to use master mode communication are. Setting the correct clock in control register 2 (R16\_I2Cx\_CTLR2) and clock control register (R16\_I2Cx\_CKCFGR).

Setting the appropriate rising edge in the rising edge register (R16\_I2Cx\_RTR).

Setting the PE bit in the control register (R16\_I2Cx\_CTLR1) to start the peripheral.

Set the START bit in the control register (R16\_I2Cx\_CTLR1) to generate the start event.

After setting the START bit, the I2C module will automatically switch to the main mode, the MSL bit will be set and the start event will be generated. After the start event is generated, the SB bit will be set and if the ITEVTEN bit (in R16\_I2Cx\_CTLR2) is set, an interrupt will be generated. The status register 1 (R16\_I2Cx\_STAR1) should be read at this time and the SB bit will be cleared automatically after writing from the address to the data register.

If the 10-bit address mode is used, then the write data register sends the header sequence (the header sequence is 11110xx0b, where the xx bits are the top two bits of the 10-bit address). After sending the header sequence, the ADD10 bit of the status register will be set, and if the ITEVTEN bit has been set, an interrupt will be generated, at this time the R16\_I2Cx\_STAR1 register should be read and the ADD10 bit cleared after writing the second address byte to the data register.

Then write the data register to send the second address byte, after sending the second address byte, the ADDR bit of the status register will be set, if the ITEVTEN bit is already set, an interrupt will be generated, at this time the R16\_I2Cx\_STAR1 register should be read and then read the R16\_I2Cx\_STAR2 register once to clear the ADDR bit;

If the 7-bit address mode, then write data register to send address byte, after sending address byte, ADDR bit of status register will be set, if ITEVTEN bit has been set, then interrupt will be generated, at this time, R16\_I2Cx\_STAR1 register should be read and then R16\_I2Cx\_STAR2 register should be read once to clear ADDR bit;

In 7-bit address mode, the first byte sent is the address byte, the first 7 bits represent the address of the target slave device, the 8th bit determines the direction of the subsequent message, 0 means the master device writes data to the slave device, 1 means the master device reads information to the slave device.

In 10-bit address mode, as shown in Figure 13-3, in the send address phase, the first byte is 11110xx0, xx is the highest 2 bits of the 10-bit address, and the second byte is the lower 8 bits of the 10-bit address. If subsequently enter the master device transmit mode, continue to send data; if subsequently ready to enter the master device receive mode, you need to re-send a start condition, follow to send a byte as 11110xx1, and then enter the master device receive mode.

Figure 13-3 Schematic diagram of master sending and receiving data at 10-bit address

**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**

Master transmit mode:

The master device's internal shift register sends data from the data register to the SDA line. When the master device receives an ACK, TxE in status register 1 (R16\_I2Cx\_STAR1) is set, and an interrupt is also generated if ITEVTEN and ITBUFEN are set. Writing data to the data register will clear the TxE bit.

If the TxE bit is set and no new data was written to the data register before the last data was sent, then the BTF bit will be set and SCL will remain low until it is cleared, and writing data to the data register after reading R16\_I2Cx\_STAR1 will clear the BTF bit.

Figure 13-4 Master transmitter transmission sequence diagram

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7-bit master send   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | S |  | Address | A |  | | Data1 | A | Data2 | | A | …… | | DataN | A |  | P | |  | EVT5 |  | | EVT6 | EVT8\_1 | EVT8 |  | EVT8 |  | | EVT8 |  | | | EVT8\_2 |  |   10-bit master send   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | S |  | Frame header | A |  | Address | A |  | | Data1 | A | …… | | DataN | A |  | P | |  | EVT5 |  | | EVT9 |  | | EVT6 | EVT8\_1 | EVT8 |  | EVT8 |  | | | EVT8\_2 |  |   Description: S=Start (start condition), Sr=repeated start condition, P=Stop (stop condition), A=response, NA=nonresponse, EVTx=event (interrupt generated when ITEVFEN=1)  EVT5: SB=1, reading SR1 and then writing the address to the DR register will clear the event.  EVT6;ADDR=1,reading SR1 then reading SR2 will clear the event.  EVT8\_1: TxE=1, shift register empty, data register empty, write DR register.  EVT8: TxE=1,shift register is not empty,data register is empty,writing DR register will clear the event.  EVT8\_2: TxE=1, BTF=1, request to set the stop bit. the TxE and BTF bits are cleared by hardware when the stop condition is generated.  EVT9: ADDR10=1, reading SR1 and then writing to DR register will clear the event.  *Note: 1: EVT5, EVT6, EVT9, EVT8\_1 and EVT8\_2 events elongate the SCL low until the end of the corresponding software sequence.*  *2: The software sequence of EVT8 must be completed before the end of the current byte transfer.* |

Master receive mode:

The I2C module will receive data from the SDA line and write it into the data register via a shift register. After each byte, if the ACK bit is set, then the I2C module will send an answer low, and the RxNE bit will be set, and an interrupt will be generated if ITEVTEN and ITBUFEN are set. If RxNE is set and the original data is not read before the new data is received, the BTF bit will be set and SCL will remain low until the BTF is cleared, and reading R16\_I2Cx\_STAR1 and then reading the data register will clear the BTF bit. 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

When the master device ends sending data, it will actively send an end event, i.e. set the STOP bit, and the I2C will switch to slave mode. In receive mode, the master device needs to NAK at the answer position of the last data bit, and after receiving NACK, the slave device releases control of the SCL and SDA lines; the master device can then send a stop/restart condition. Note that the I2C module will automatically switch to slave mode after the stop condition is generated.

## 13.4 Slave Mode

When in slave mode, the I2C module recognizes its own address and the broadcast call address. The software can control whether the recognition of the broadcast call address is enabled or disabled. Once a start event is detected, the I2C module compares the SDA data through the shift register with its own address (number of bits depends on ENDUAL and ADDMODE) or the broadcast address (when ENGC is set), if there is a mismatch it will be ignored until a new start event is generated. If it matches the header sequence, an ACK signal is generated and the address of the second byte is waited for; if the address of the second byte also matches or the full segment address matches in the case of a 7- bit address, then:

first an ACK answer is generated;

the ADDR bit is set, and if the ITEVTEN bit is already set, then a corresponding interrupt is also generated; if the dual address mode is used (ENDUAL bit is set), the DUALF bit also needs to be read to determine which address the host is evoking.

The slave mode is receive mode by default. In case the last bit of the received header sequence is 1, or the last bit of the 7-bit address is 1 (depending on whether the header sequence is received for the first time or a normal 7-bit address), the I2C module will go to transmitter mode and the TRA bit will indicate whether it is currently receiver or transmitter mode.

Slave transmit mode:

After clearing the ADDR bit, the I2C module sends bytes from the data register to the SDA line via a shift register. After an answer ACK is received, the TxE bit is set and an interrupt is generated if ITEVTEN and ITBUFEN are set. If TxE is set but no new data is written to the data register before the end of the next data send, the BTF bit will be set. SCL will remain low until the BTF is cleared. Reading status register 1 (R16\_I2Cx\_STAR1) and then writing data to the data register will clear the BTF bit.

Figure 13-6 Slave transmitter transmission sequence diagram

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-

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

response, EVTx=event (interrupt is generated when ITEVFEN=1)

EVT1;ADDR=1,read SR1 then read SR2 will eliminate the event.

EVT3\_1: TxE=1, shift register empty, data register empty, write DR.

EVT3: TxE=1,shift register is not empty,data register is empty,writing DR will clear the event.

EVT3\_2: AF=1, write '0' in AF bit of SR1 register to clear AF bit.

*Note: 1: EVT1 and EVT3\_1 events elongate SCL low until the end of the corresponding software sequence.*

*2: The software sequence of EVT3 must be completed before the end of the current byte transfer.*

bit slave send

10-

……

……

Slave receive mode:

After ADDR is cleared, the I2C module stores the data on SDA into the data register via the shift register. After each byte is received, the I2C module sets an ACK bit and sets the RxNE bit, and generates an interrupt if ITEVTEN and ITBUFEN are set. If RxNE is set and the old data is not read before the new data is received, then BTF is set. SCL will remain low until the BTF bit is cleared. Reading status register 1 (R16\_I2Cx\_STAR1) and reading the data in the data register will clear the BTF bit.

Figure 13-7 Receiver transmission sequence diagram

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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.* |

The master device will generate a stop condition after the last data byte is transferred. When the I2C module detects a stop event, it will set the STOPF bit, and if the ITEVFEN bit is set, it will also generate an interrupt. The user needs to read the status register (R16\_I2Cx\_STAR1) and then write the control register (e.g. reset control word SWRST) to clear it. (See EVT4 in the figure above).

## 13.5 Error Conditions

### 13.5.1 Bus Error (BERR)

A bus error will be generated when the I2C module detects an external start or stop event during address or data transfer. When a bus error is generated, the BERR bit is set and an interrupt is generated if ITERREN is set. In slave mode, the data is discarded and the hardware releases the bus. If it is a start signal, the hardware assumes it is a restart signal and starts waiting for an address or stop signal; if it is a stop signal, it operates ahead of normal stop conditions. In master mode, the hardware does not release the bus while not affecting the current transfer, and it is up to the user code to decide whether to abort the transfer.

### 13.5.2 Acknowledge Failure (AF)

An answer error will be generated when the I2C module detects a byte and then no answer. When an answer error is generated: AF will be set and an interrupt will be generated if ITERREN is set; when an AF error is encountered, the hardware must release the bus if the I2C module is working in slave mode and the software must generate a stop event if it is in master mode.

### 13.5.3 Arbitration Lost (ARLO)

An arbitration lost error is generated when the I2C module detects an arbitration lost. When an arbitration loss error is generated: the ARLO bit is set and an interrupt is generated if ITERREN is set; the I2C module switches to slave mode and no longer responds to transfers initiated against its slave address unless a new start event is initiated by the host; the hardware releases the bus.

### 13.5.4 Overrun/underrun Error (OVR)  Overrun error

In Slave mode, if the clock extension is disabled and the I2C module is receiving data, an overrun error will occur if a byte of data has been received but the last received data has not been read out. When an overrun error occurs, the last received byte will be discarded and the sender should retransmit the last sent byte.

###  Underrun error

In Slave mode, if the clock is forbidden to extend and the I2C module is sending data, an underrun error will occur if new data has not been written to the data register before the next byte of the clock comes. In case of an underrun error, the data in the previous data register will be sent twice, and if an underrun error occurs, then the receiver should discard the data received repeatedly. In order not to generate an underrun error, the I2C module should write the data to the data register before the first rising edge of the next byte.

## 13.6 Clock Extension

If clock extension is disabled, then there is a possibility of overrun/underrun errors. However, if clock extension is enabled:

* In transmit mode, if TxE is set and BTF is set, SCL will always be low, always waiting for the user to read the status register and write the data to be sent to the data register.
* In receive mode, if RxNE is set and BTF is set, SCL will remain low after data is received until the user reads the status register and reads the data register.

It can be seen that enabling clock extension can avoid overrun/underrun errors.

## 13.7 SMBus

SMBus is also a 2-wire interface, which is generally used between system and power management. SMBus and I2C have many similarities, for example, SMBus uses the same 7-bit address mode as I2C, and the following are common to SMBus and I2C.

1. Master-slave communication mode, where the host provides the clock and supports multiple masters and slaves.
2. 2-wire communication architecture, with an optional warning line for SMBus. 3) Both support 7-bit address format.

There are also differences between SMBus and I2C.

1. I2C supports speeds up to 400 KHz, while SMBus supports up to 100 KHz, and SMBus has a minimum speed limit of 10 KHz.
2. A timeout will be reported when the SMBus clock is low for more than 35mS, but there is no such limit for I2C.
3. SMBus has a fixed logic level, while I2C does not, depending on VDD.
4. SMBus has a bus protocol, while I2C does not.

SMBus also includes device identification, address resolution protocols, unique device identifiers, SMBus reminders and various bus protocols as described in the SMBus specification version 2.0. When using SMBus, only the SMBus bit of the control register needs to be set, and the SMBTYPE bit and ENAARP bit need to be configured as needed.

## 13.8 Interruptions

Each I2C module has two interrupt vectors, event interrupts and error interrupts. Both interrupts support the interrupt sources in Figure 13-4.

Figure 13-4 I2C Interrupt Request



## 13.9 DMA

DMA can be used to send and receive bulk data. The ITBUFEN bit of the control register cannot be set when using DMA.

 Transmission using DMA

DMA mode can be activated by setting the DMAEN bit of the CTLR2 register. As long as the TxE bit is set, data will be loaded by DMA from the set memory into the data register of the I2C. The following settings are required to allocate channels for I2C.

1. Set the I2Cx\_DATAR register address to the DMA\_PADDRx register and the memory address in the DMA\_MADDRx register so that after each TxE event, data will be sent from memory to the I2Cx\_DATAR register.
2. Set the required number of bytes to be transferred in the DMA\_CNTRx register. This value will be decremented after each TxE event.
3. Configure the channel priority using the PL[0:1] bits in the DMA\_CFGRx register.
4. Set the DIR bit in the DMA\_CFGRx register and depending on the application requirements can be configured to issue an interrupt request when the entire transfer is half or fully completed.
5. Activate the channel by setting the EN bit on the DMA\_CFGRx register.

When the number of data transfer bytes set in the DMA controller has been completed, the DMA controller sends an end of transfer EOT/ EOT\_1 signal to the I2C interface. A DMA interrupt will be generated if the interrupt is allowed.

 Reception using DMA

DMA receive mode can be performed after setting DMAEN in the CTLR2 register. When using DMA receive, DMA transfers the data in the data register to the preset memory area. The following steps are required to allocate channels for I2C.

1. Set the I2Cx\_DATAR register address to the DMA\_PADDRx register and the memory address in the DMA\_MADDRx register so that after each RxNE event, data will be written to memory from the I2Cx\_DATAR register.
2. Set the required number of bytes to be transferred in the DMA\_CNTRx register. This value will be decremented after each RxNE event.
3. Configure the channel priority with PL[0:1] in the DMA\_CFGRx register.
4. The DIR bit in the DMA\_CFGRx register is cleared, and depending on the application requirements, an interrupt request can be set to be issued when the data transfer is half or fully completed.
5. Set the EN bit in the DMA\_CFGRx register to activate the channel.

When the number of data transfers set in the DMA controller has been completed, the DMA controller sends an end of transfer EOT/EOT\_1 signal to the I2C interface. A DMA interrupt will be generated if the interrupt is allowed.

## 13.10 Packet Error Checking

Packet Error Checksum (PEC) is an additional CRC8 checksum step to provide transmission reliability, calculated for each bit of serial data using the following polynomial.

C=X8 +X2 +X+1

The PEC calculation is activated by the ENPEC bit in the control register and is performed on all information bytes, including address and read/write bits. In transmitting, enabling PEC adds a byte of CRC8 calculation result after the last byte of data; while in receiving mode, in the last byte is considered as CRC8 check result, and if it does not match with the internal calculation result, it will reply a NAK, and in case of the main receiver, regardless of the correct check result.

## 13.11 Register Description

Table 13-1 I2C-related registers list

Name Offset address Description Reset value

R16\_I2C\_CTLR1 0x40005400 I2C control register 1 0x0000

R16\_I2C\_CTLR2 0x40005404 I2C control register 2 0x0000

R16\_I2C\_OADDR1 0x40005408 I2C address register 1 0x0000

R16\_I2C\_OADDR2 0x4000540C I2C address register 2 0x0000

R16\_I2C\_DATAR 0x40005410 I2C data register 0x0000

R16\_I2C\_STAR1 0x40005414 I2C status register 1 0x0000

R16\_I2C\_STAR2 0x40005418 I2C status register 2 0x0000

R16\_I2C\_CKCFGR 0x4000541C I2C clock register 0x0000

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

Offset address: 0x00

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

SWRST Reserved PEC POS ACK STOP START TRETNOSCH ENGC ENPEC Reserved PE

Bit Name Access Description Reset value

Software reset, setting this bit by user code will reset the I2C peripheral. Make sure the pins of the

I2C bus are released and the bus is idle before the

15 SWRST RW reset. 0

*Note: This bit resets the I2C module when no stop condition is detected on the bus but the busy bit is 1.*

[14:13] Reserved RO Reserved 0

Packet error checking bit, set this bit to enable packet error detection. The user code can set or clear this bit; the hardware clears this bit when the

PEC is transmitted, when a start or end signal is

12 PEC RW generated, or when the PE bit is cleared to 0. 0

1: With PEC.

0: Without PEC.

*Note: The PEC is invalidated when arbitration is lost.*

ACK and PEC position setting bits, which can be set or cleared by user code and can be cleared by hardware after the PE has been cleared.

1: ACK bit controls the ACK or NAK of the next byte received in the shift register. The next byte received in the PEC shift register is the PEC.

0: The ACK bit controls the ACK or NAK of the byte currently being accepted in the shift register.

11 POS RW the PEC bit indicates that the byte in the shift 0

register before the current bit is PEC.

*Note: The POS bit is used in 2-byte data reception as follows: it must be configured before reception. In order to NACK the 2nd byte, the ACK bit must be cleared immediately after clearing the ADDR bit; in order to detect the PEC of the second byte, the PEC bit must be set after the ADDR event and after configuring the POS bit.*

Acknowledge enable, This bit is set and cleared by software and cleared by hardware when PE=0.

10 ACK RW 1：Acknowledge returned after a byte is received. 0

0：No acknowledge returned.

Stop generation bit. This bit is set and cleared by software, cleared by hardware when a Stop condition is detected, set by hardware when a timeout error is detected. In Master mode:

9 STOP RW 1：Stop generation after the current byte transfer 0

or after the current Start condition is sent.

0：No Stop generation.

In Slave mode:

1：Release the SCL and SDA lines after the current byte transfer.

0：No Stop generation.

Start generation. This bit is set and cleared by software and cleared by hardware when start is sent or PE=0. In Master mode:

8 START RW 1：Repeated start generation 0

0：No Start generation In Slave mode:

1：Start generation when the bus is free

0：No Start generation

Clock stretching disable bit. This bit is used to disable clock stretching in slave mode when ADDR or BTF flag is set, until it is reset by

7 NOSTRETCH RW software. 0

1：Clock stretching disabled.

0：Clock stretching enabled.

General call enable bit. Set this bit to enable

6 ENGC RW 0

broadcast call and answer broadcast address 00h.

PEC enable bit, set this bit to enable PEC

5 ENPEC RW 0 calculation.

[4:1] Reserved RO Reserved 0

I2C peripheral enable bit.

0 PE RW 1: Enable the I2C module. 0

0: Disable the I2C module.

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

Offset address: 0x04

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

Reserved LAST DMAEN ITBUFEN ITEVTEN ITERREN Reserved FREQ[5:0]

Bit Name Access Description Reset value

[15:13] Reserved RO Reserved 0

DMA last transfer bit.

1：Next DMA EOT is the last transfer.

12 LAST RW 0：Next DMA EOT is not the last transfer. 0

*Note: This bit is used in master receiver mode to permit the generation of a NACK on the last received data.*

DMA requests enable bit. Set this bit to allow

11 DMAEN RW 0

DMA request when TxE or RxEN is set.

Buffer interrupt enable bit.

1：When TxE or RxEN is set, event interrupt is

10 ITBUFEN RW generated. 0

0：When TxE or RxEN is set, no interrupt is generated.

Event interrupt enable bit. Set this bit to enable event interrupt. This interrupt will be generated under the following conditions.

SB=1 (Master mode).

ADDR=1 (Master-slave mode).

9 ITEVTEN RW 0

ADDR10 = 1 (Master mode).

STOPF=1 (Slave mode).

BTF = 1, but no TxE or RxEN events.

TxE event to 1 if ITBUFEN = 1.

RxNE event to 1if ITBUFEN = 1.

Error interrupt enable bit. Set to allow error interrupts.

The interrupt will be generated under the

8 ITERREN RW 0

following conditions.

BERR=1; ARLO=1; AF=1; OVR=1; PECERR=1.

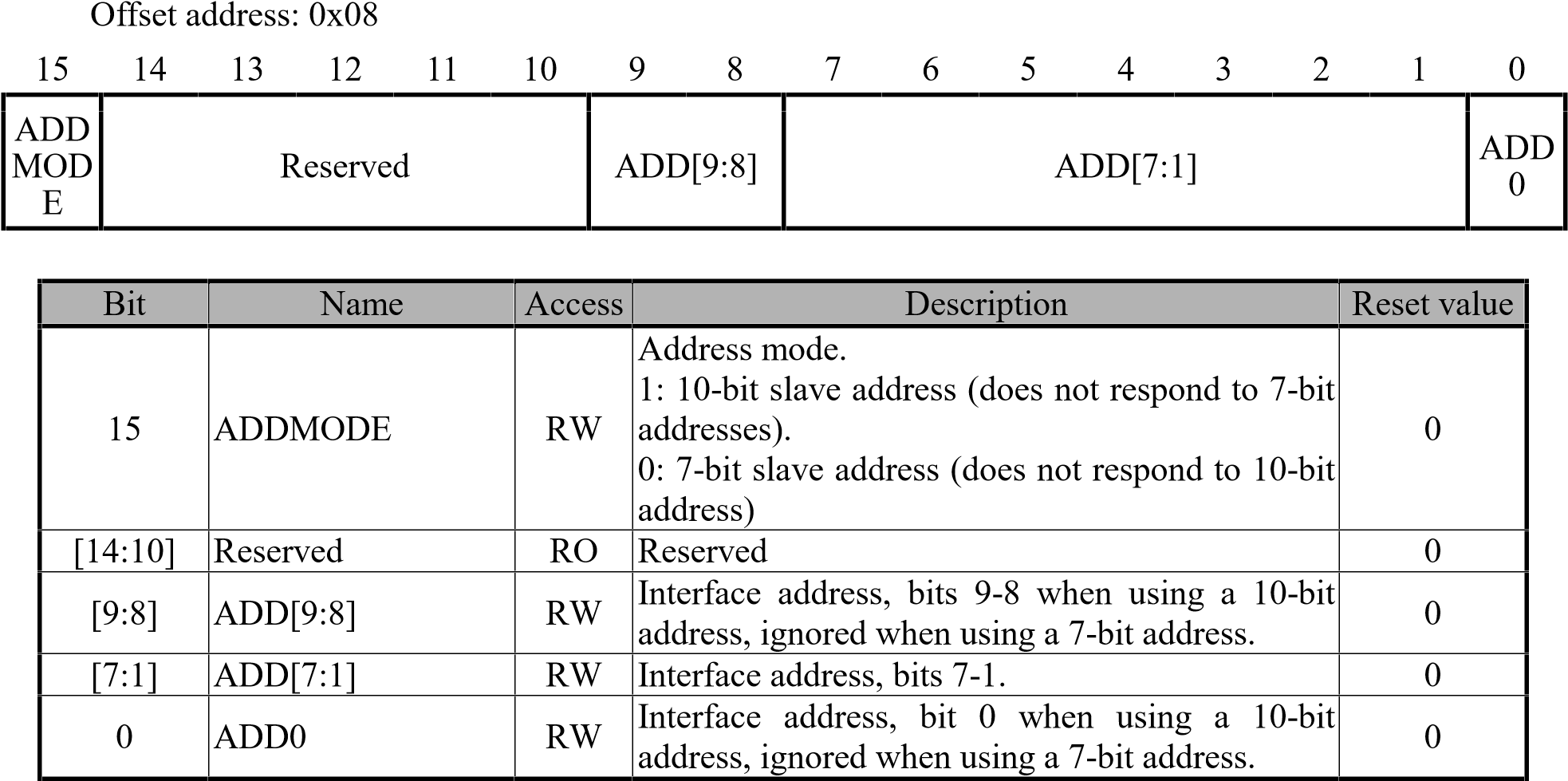
TIMEOUT=1; SMBAlert=1.

[7:6] Reserved RO Reserved 0

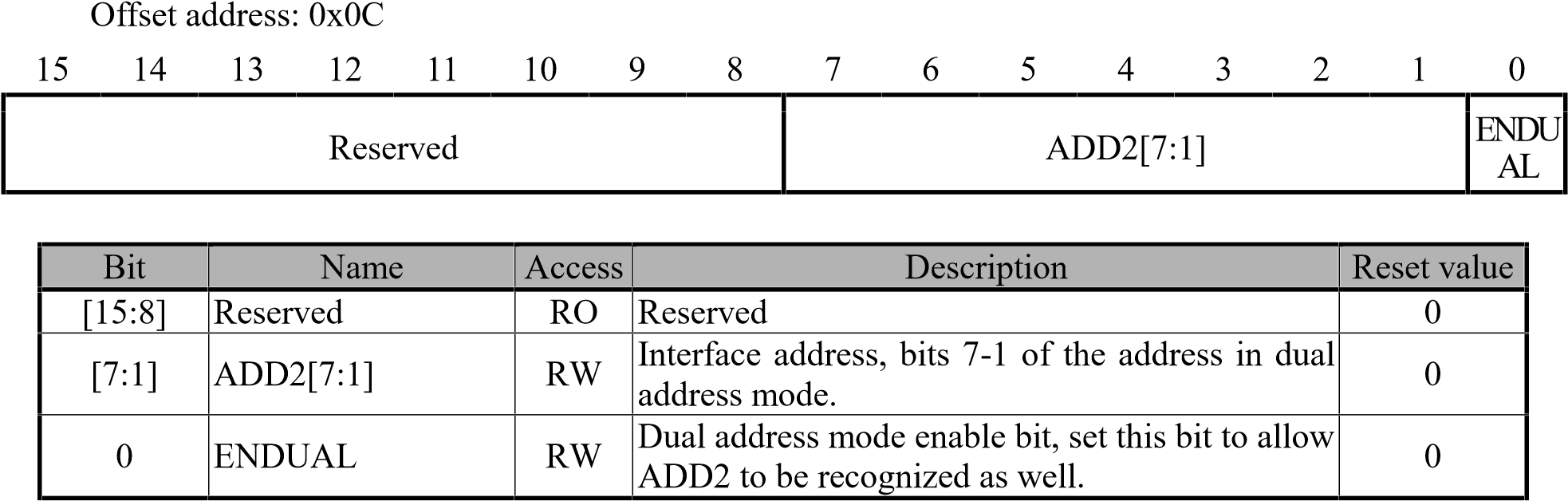
The I2C module clock frequency field, which must be entered at the correct clock frequency to

[5:0] FREQ[5:0] RW produce the correct timing, allows a range 0 between 8-48MHz. It must be set between 001000b and 110000b in MHz.

### 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)

Offset address: 0x10

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

Reserved

DR[7:0]

Bit Name Access Description Reset value

15:8 Reserved RO Reserved 0

Data register, this field is used to store the received

7:0 DR[7:0] RW 0 data or to store the data used to send to the bus.

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

Offset address: 0x14

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

Reserved PECERR OVR AF ARLO BERR TxE RxNE Reserved STOPF ADD10 BTF ADDR SB

Bit Name Access Description Reset value

[15:13] Reserved RO Reserved 0

The PEC error flag bit occurs on reception, and this bit can be reset by a user write of 0 or by hardware when PE goes low.

12 PECERR RW0 0

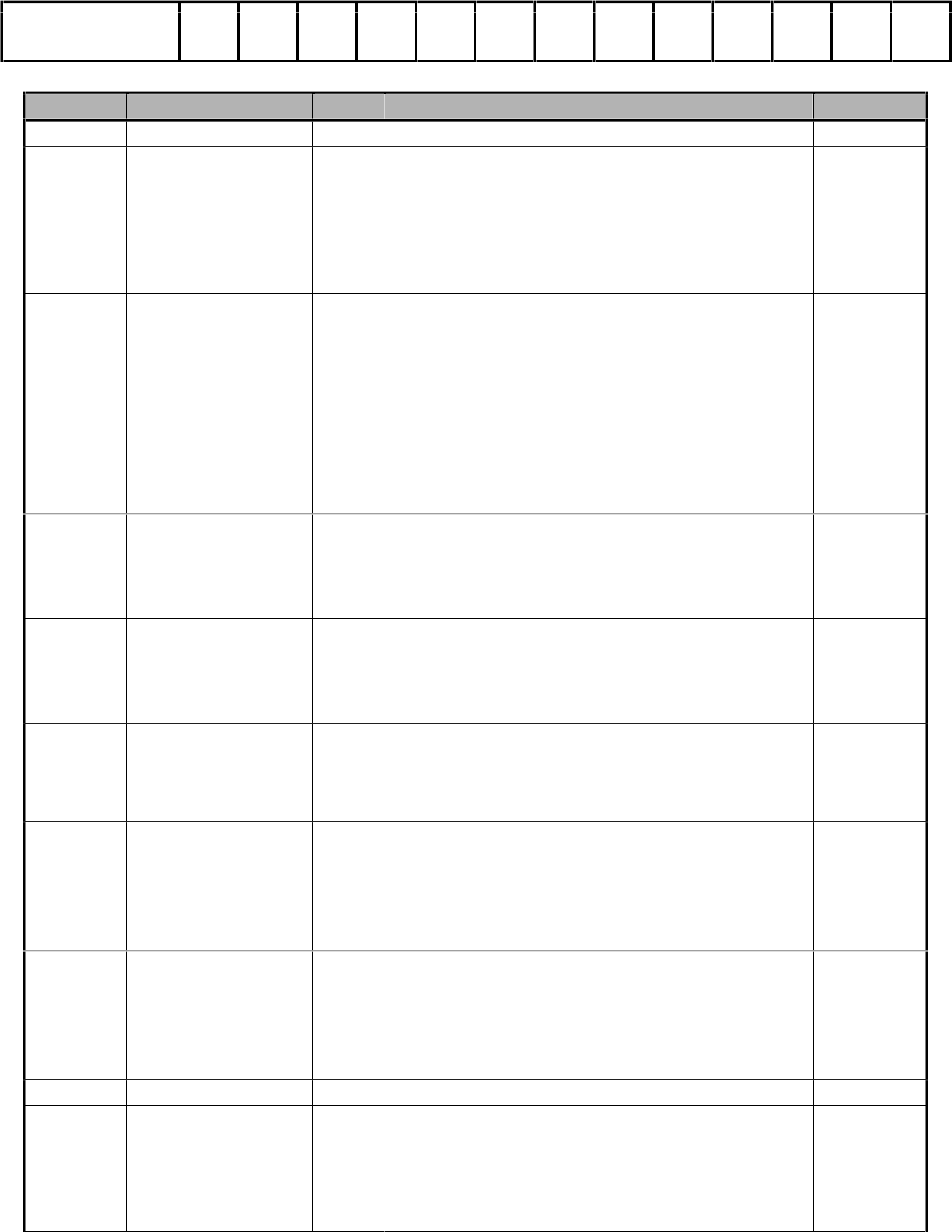
1: There is a PEC error and the PEC is received and NAK is returned.

0: No PEC error.

Overrun and underrun flag bits.

1: There are overrun and underrun events occurring: when NOSTRETCH=1, when a new byte is received in receive mode, the content in the

11 OVR RW0 data register has not been read out, then the newly 0

received byte will be lost; when in send mode, no new data is written to the data register, and the same byte will be sent twice. 0: No overrun or underrun events.

Acknowledge failure bit. Cleared by software writing 0, or by hardware when PE=0.

10 AF RW0 1：Acknowledge failure. 0

0：No acknowledge failure.

Arbitration lost bit. Cleared by software writing

0, or by hardware when PE=0.

9 ARLO RW0 1：Arbitration Lost detected. 0

0：No Arbitration Lost detected.

The bus error flag bit, which can be reset by a user

write of 0, or by hardware when PE goes low.

8 BERR RW0 0

1: Error in start or stop condition; 0: Normal.

Data register empty bit. Cleared by software writing to the DR register or by hardware after a

7 TxE RO start or a stop condition or when PE=0. 0

1：Data register empty.

0：Data register not empty.

Data register not empty bit. Cleared by software reading or writing the DR register or by hardware

6 RxNE RO when PE=0. 0

1：Data register not empty.

0：Data register empty.

5 Reserved RO Reserved 0

Stop detection bit. Cleared by software reading the SR1 register followed by a write in the CR1

4 STOPF RO register, or by hardware when PE=0 0

1：Set by hardware when a Stop condition is detected on the bus by the slave after an acknowledge (if ACK=1). 0：No Stop condition detected.

10-bit header sent bit. Cleared by software reading the SR1 register followed by a write in the DR register of the second address byte, or by

3 ADD10 RO hardware when PE=0. 0

1：Master has sent first address byte.

0：No ADD10 event occurred.

Byte transfer finished bit. Cleared by software reading SR1 followed by either a read or write in the DR register or by hardware after a start or a stop condition in transmission or when PE=0.

1：Data byte transfer succeeded. When

2 BTF RO NOSTRETCH=0: when sending, when a new data 0

is sent and the data register has not yet been written with new data; when receiving, when a new byte is received but the data register has not yet been read.

0：Data byte transfer not done.

Address sent /matched bit. This bit is cleared by software reading SR1 register followed reading SR2, or by hardware when PE=0. In Master mode:

1：End of address transmission. For 10-bit addressing, the bit is set after the ACK of the 2nd

1 ADDR RW0 byte. For 7-bit addressing, the bit is set after the 0

ACK of the byte.

0：No end of address transmission.

In Slave mode:

1：Received address matched.

0：Address mismatched or not received.

Start bit. Cleared by software by reading the SR1 register followed by writing the DR register, or by

0 SB RO hardware when PE=0 0

1：Start condition generated.

0：No Start condition.

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

Offset address: 0x18

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

PEC[7:0] DUALF Reserved GENCALL Reserved TRA BUSY MSL

Bit Name Access Description Reset value

Packet error checking bit. When PEC is enabled

[15:8] PEC[7:0] RO 0

(ENPEC is set), this field holds the value of PEC. Dual flag. Cleared by hardware after a Stop condition or repeated Start condition, or when

7 DUALF RO PE=0. 0

1：Received address matched with OAR2.

0：Received address matched with OAR1.

[6:5] Reserved RO Reserved 0

General call address bit. Cleared by hardware

4 GENCALL RO after a Stop condition or repeated Start condition, 0 or when PE=0.

1：General Call Address received when ENGC=1.

0：No General Call.

3 Reserved RO Reserved 0

Transmitter/receiver bit. It is cleared by hardware after detection of Stop condition (STOPF=1), repeated Start condition, loss of bus arbitration (ARLO=1), or when PE=0.

2 TRA RO 1：Data bytes transmitted. 0

0：Data bytes received.

This bit is set depending on the R/W bit of the address byte.

Bus busy bit. Cleared by hardware on detection of a Stop condition. This information is still updated when the interface is disabled (PE=0).

1 BUSY RO 1：Communication ongoing on the bus: low level 0

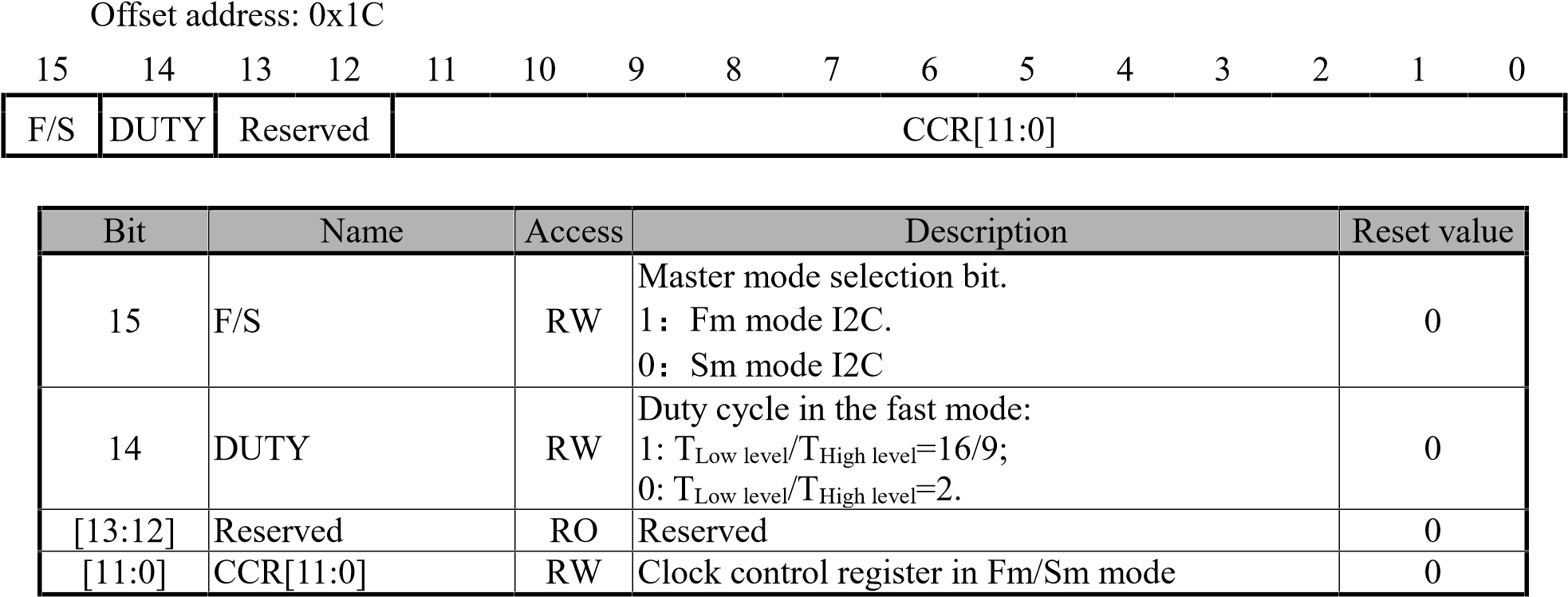
present in SDA or SCL. 0：No communication on the bus.

Master/slave bit. Set by hardware as soon as the interface is in Master mode (SB=1). Cleared by

0 MSL RO hardware after detecting a Stop condition on the 0

bus or a loss of arbitration (ARLO=1), or by hardware when PE=0.

### 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.

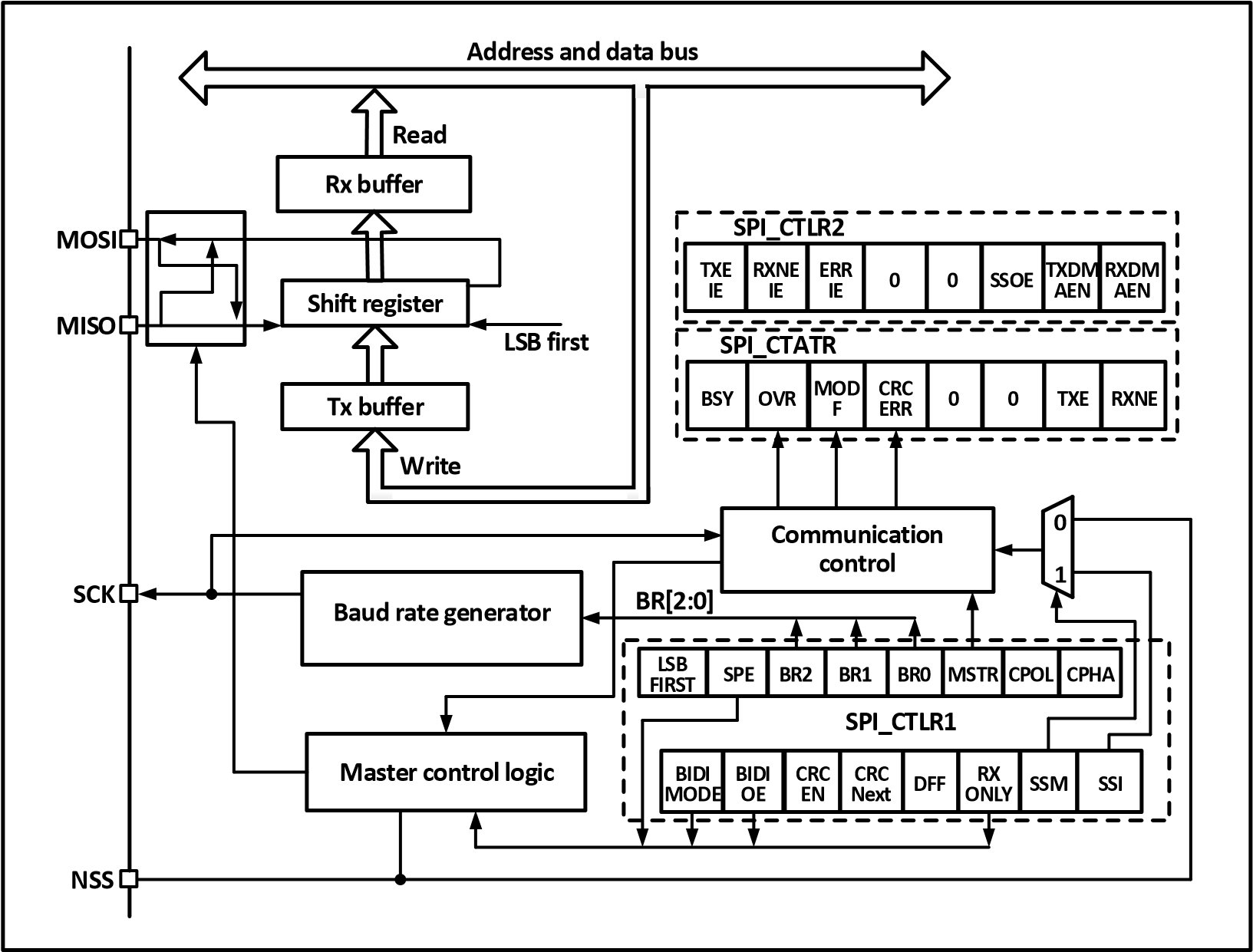
## 14.1 Main Features

* 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

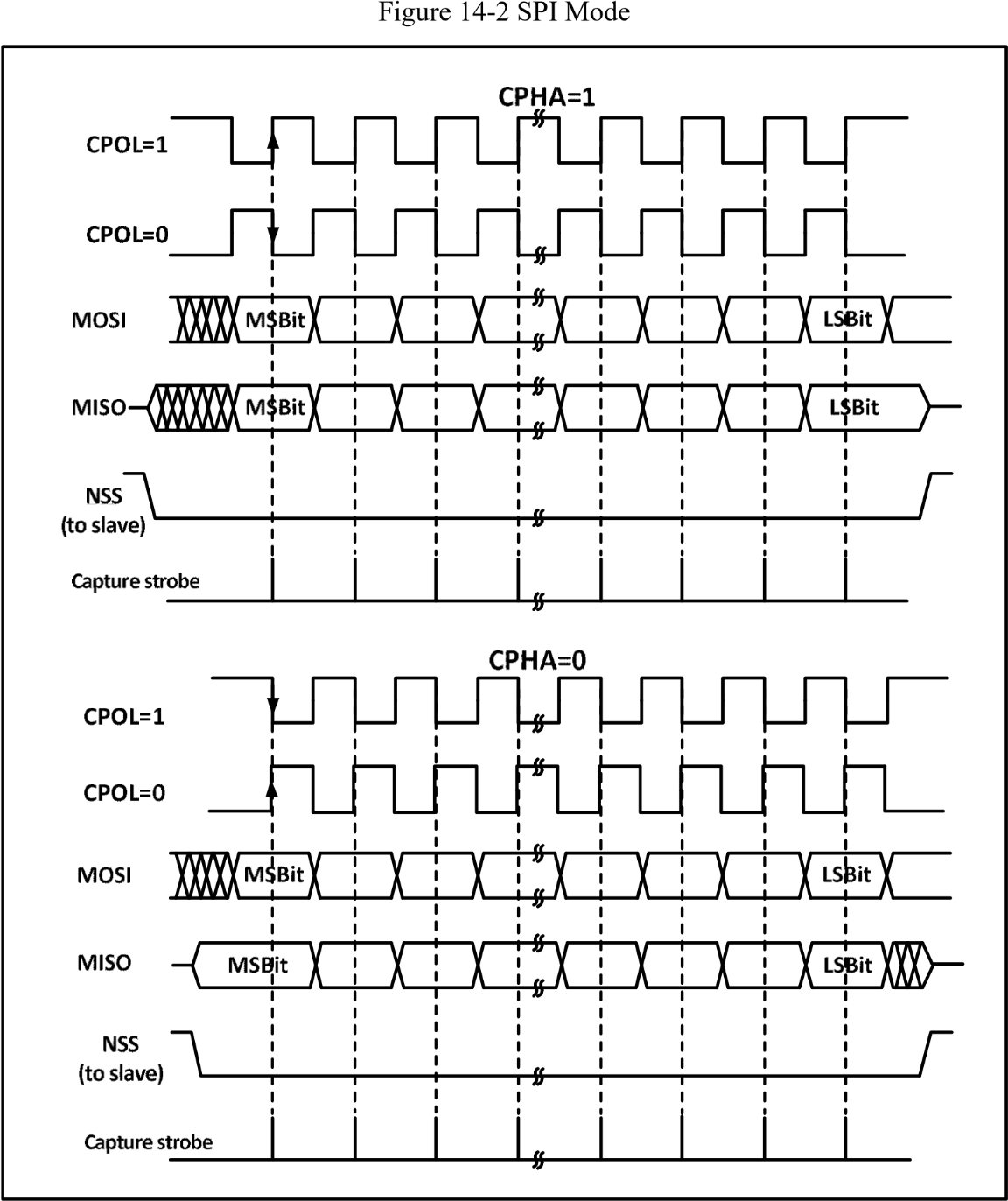
Figure 14-1 SPI structure block diagram



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.

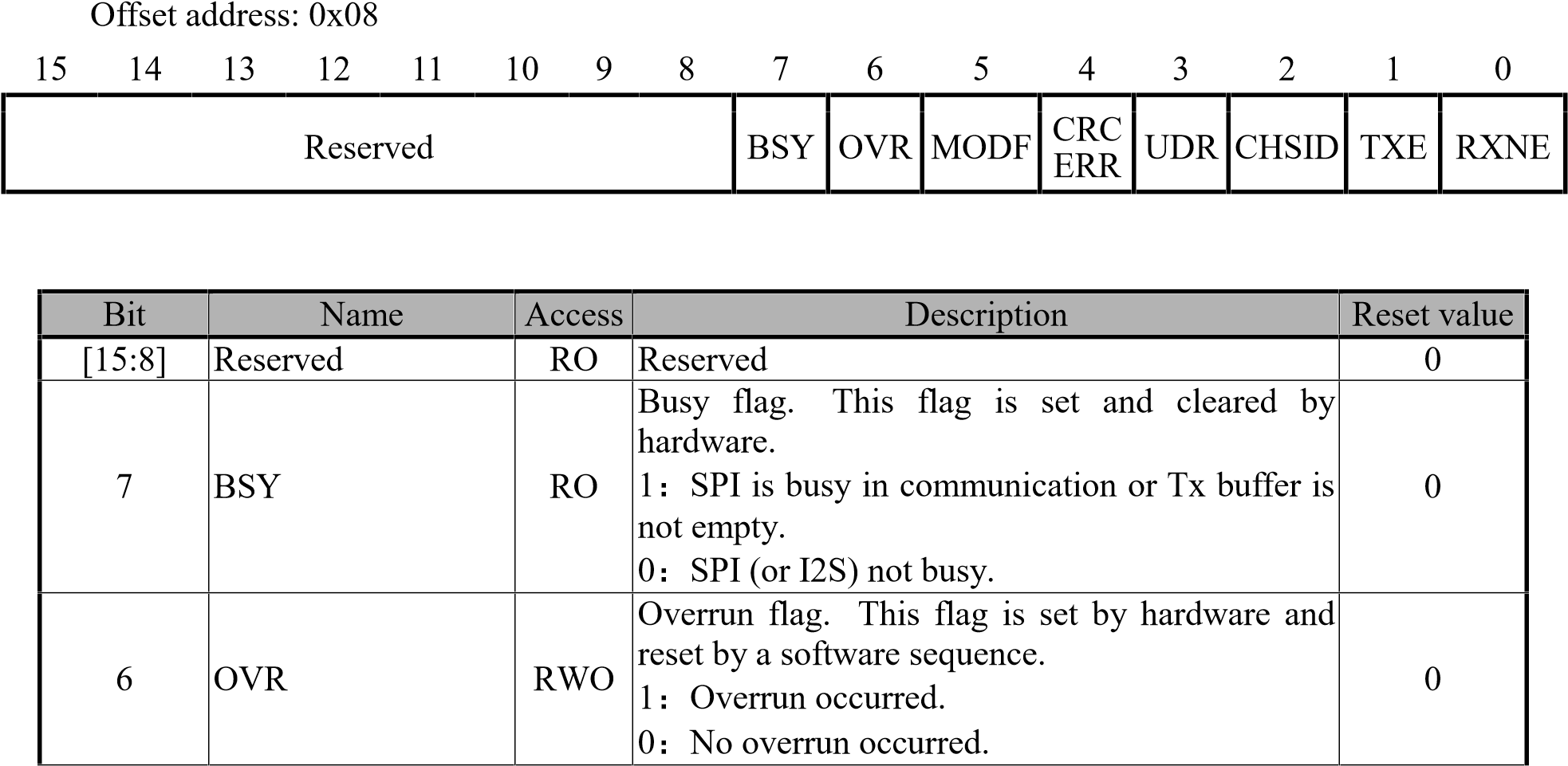
0:

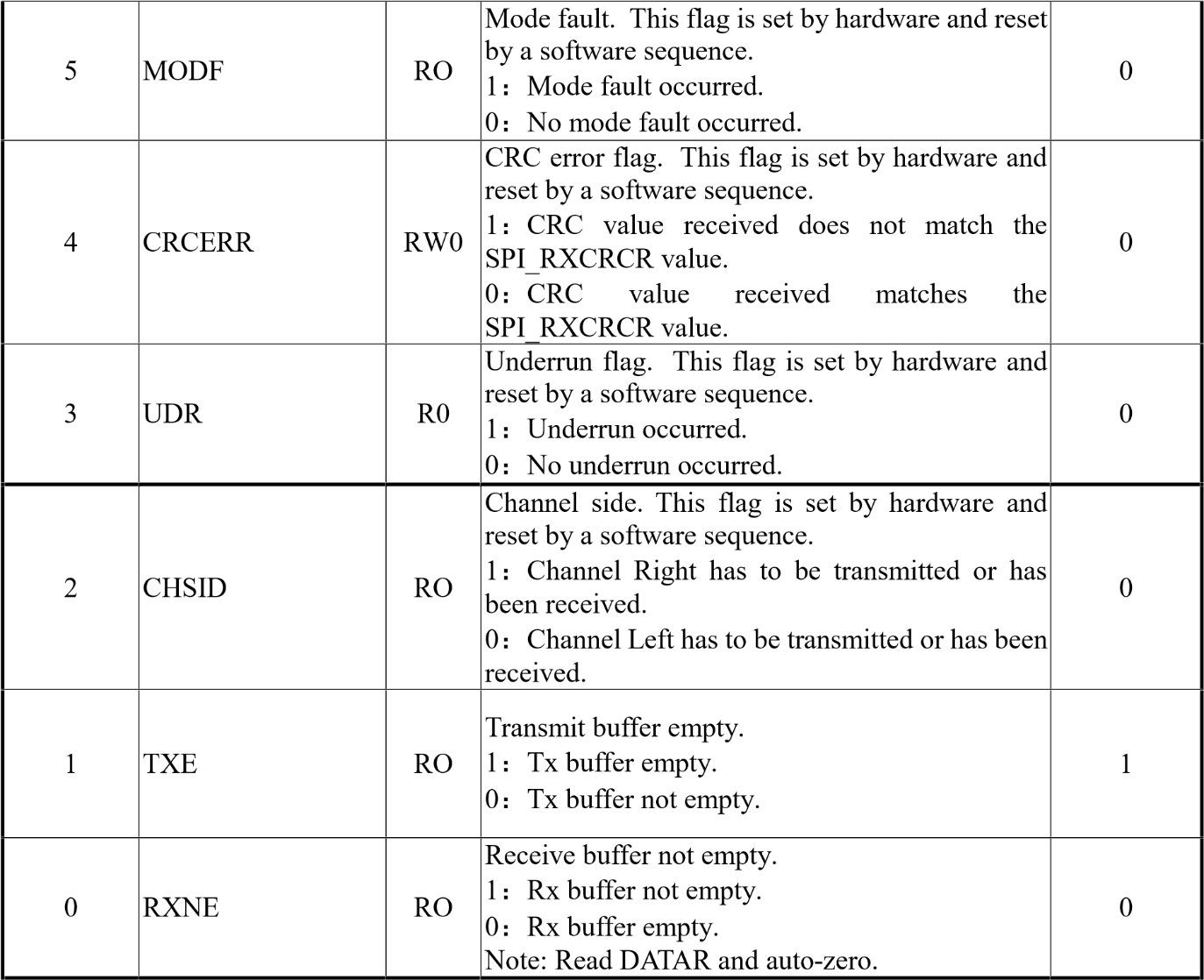
Data sampling starts from the first clock edge.

### 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

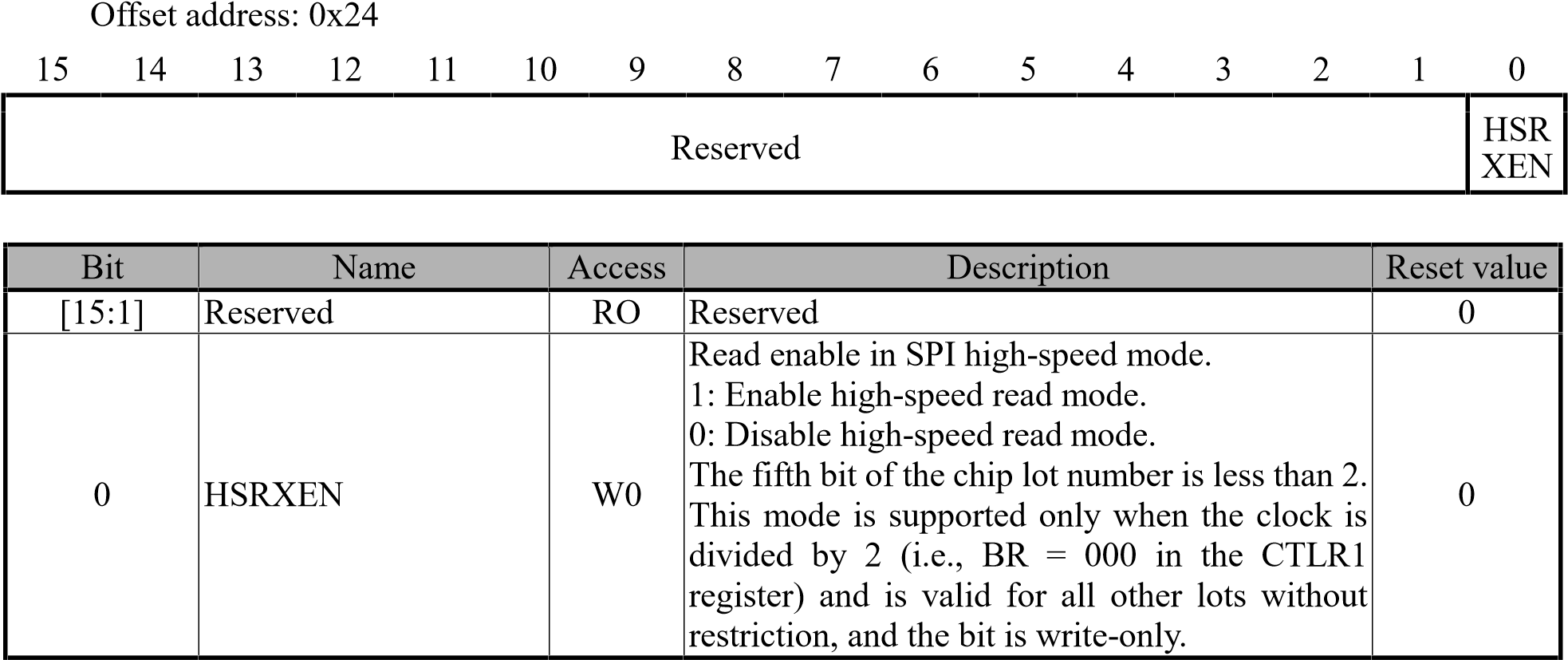
Tx CRC. Store the result of the calculated CRC checksum of the bytes that have been sent out. Setting CRCEN resets this register. The

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]

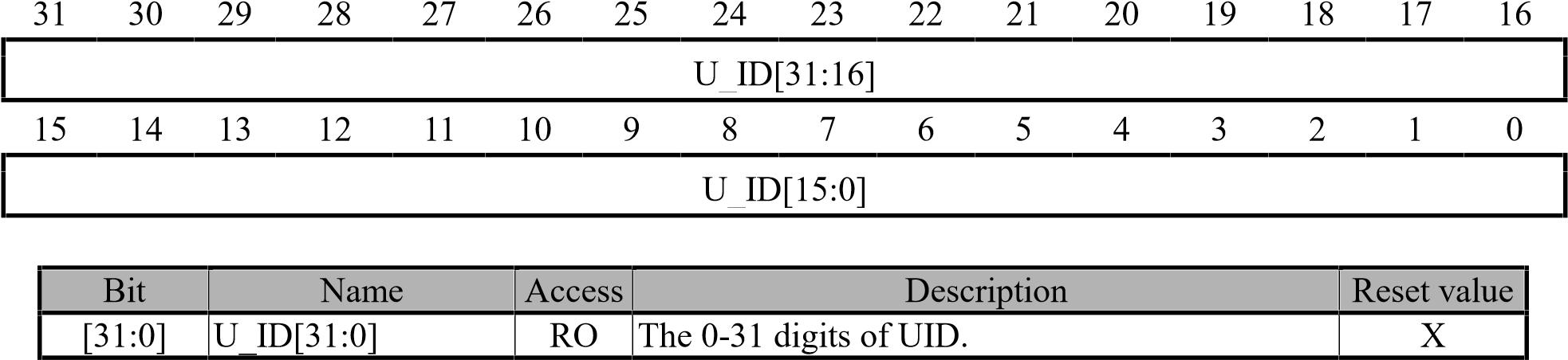
RO

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

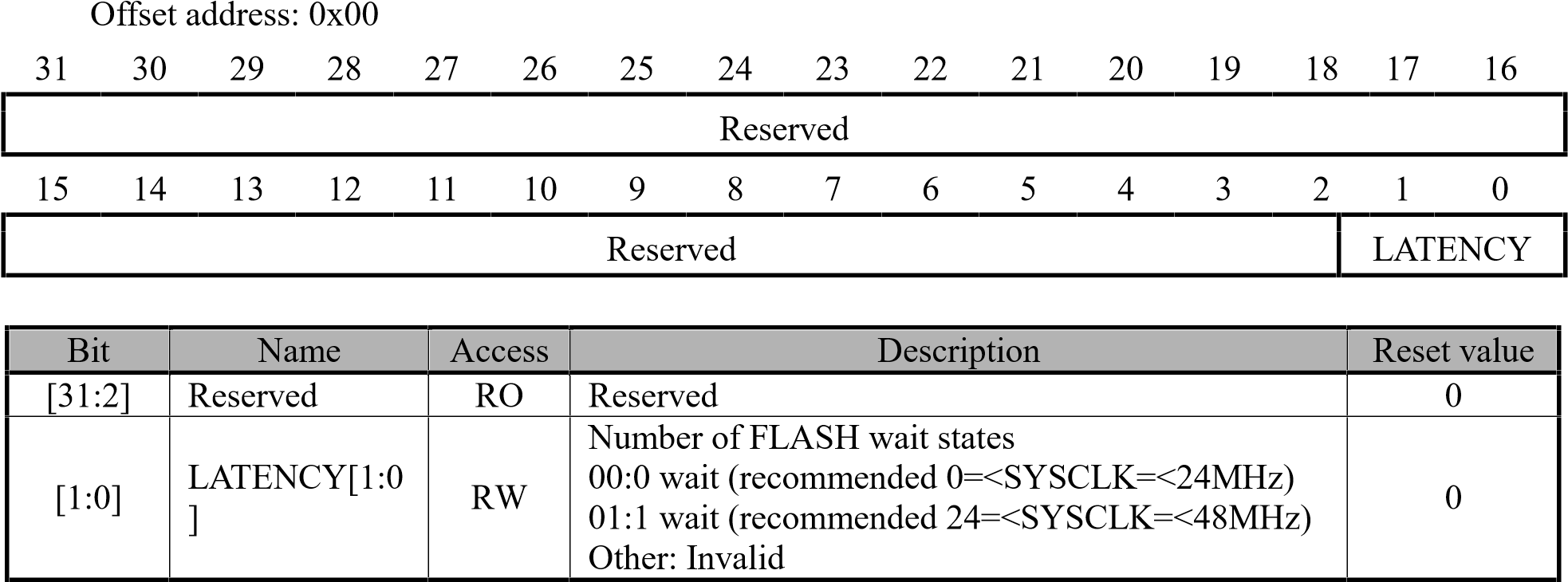
R32\_FLASH\_OBR 0x4002201C Option byte register 0x0XXXXXXX

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

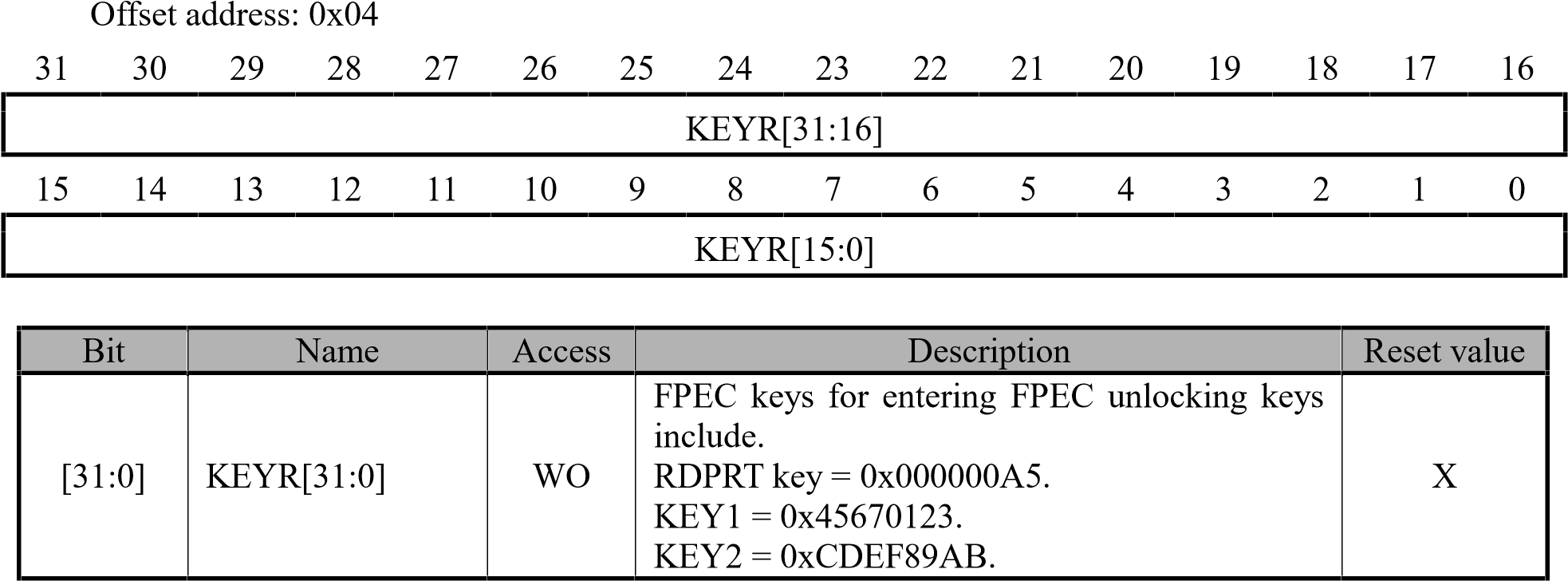
R32\_FLASH\_MODEKEYR 0x40022024 Extended key register X

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

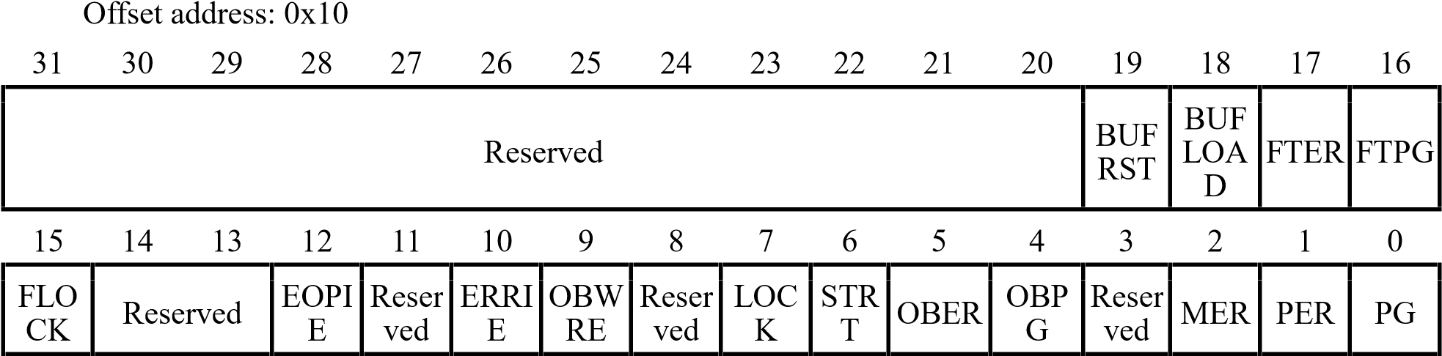
key to release OPTWRE.

### 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

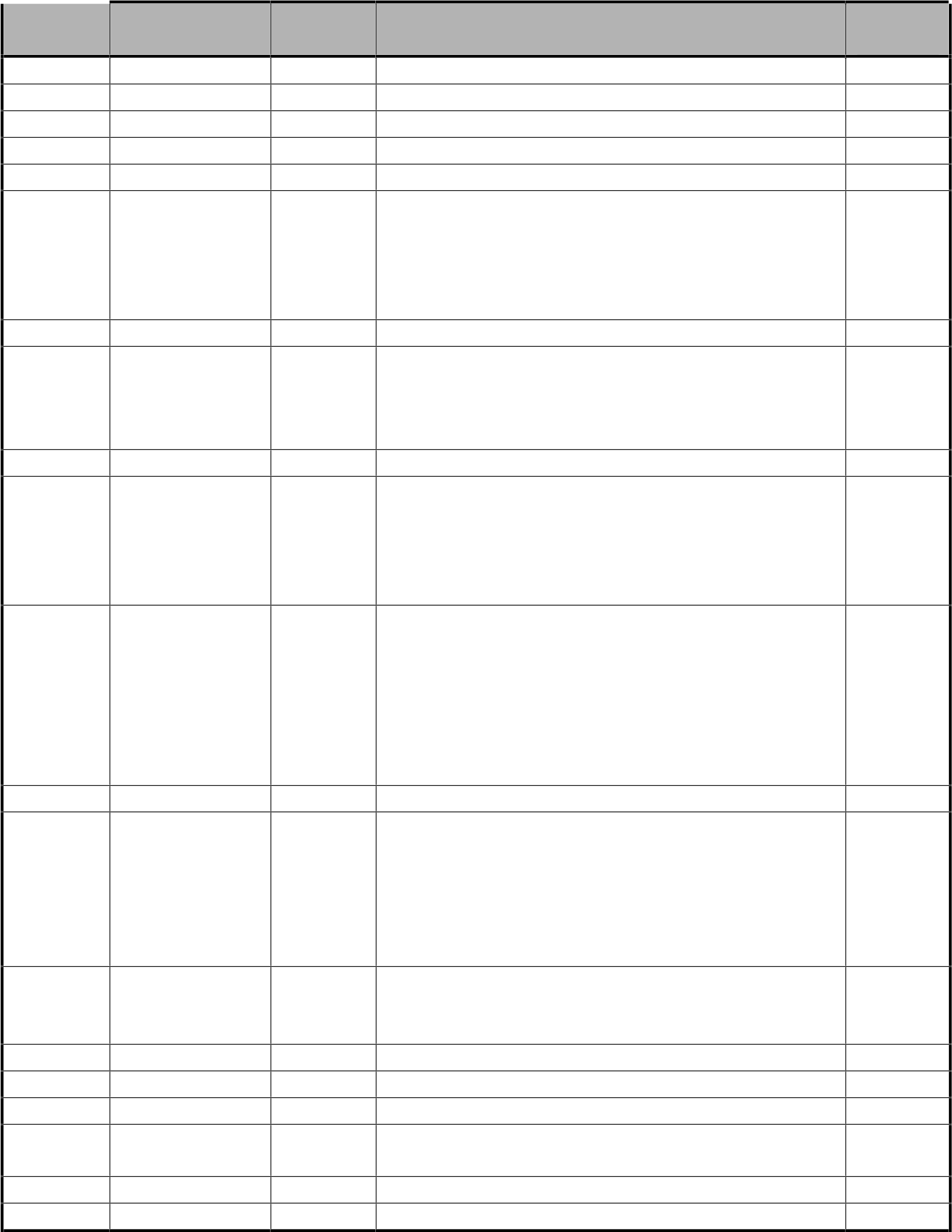
the correct unlock sequence is detected.

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

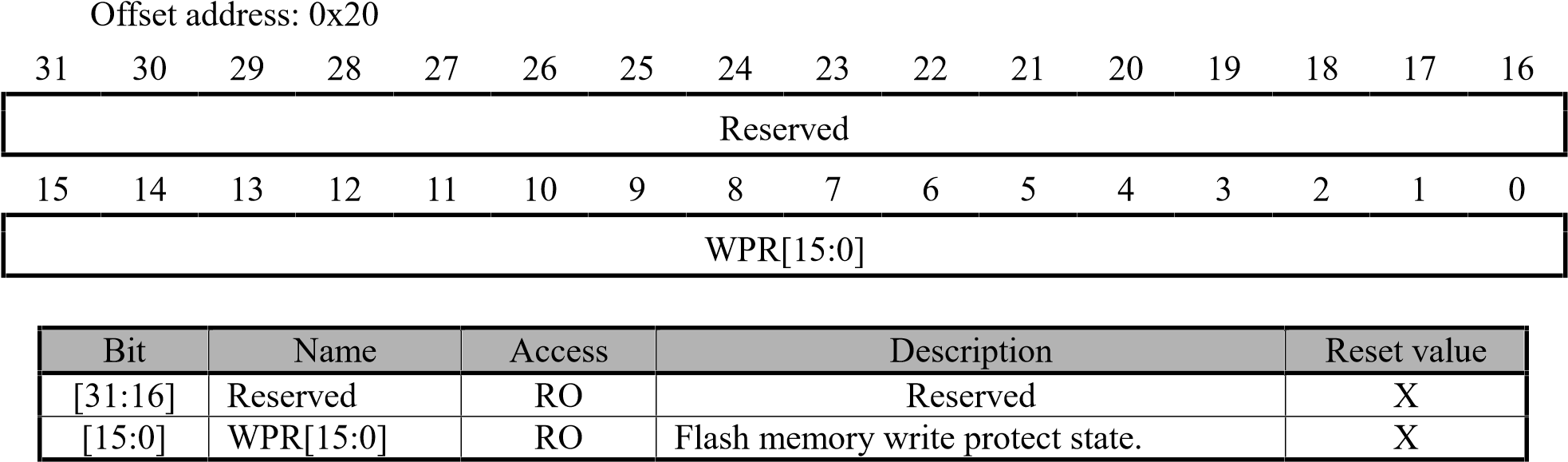
When the BSY bit in FLASH\_SR register is '1', this register cannot be written.

### 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:

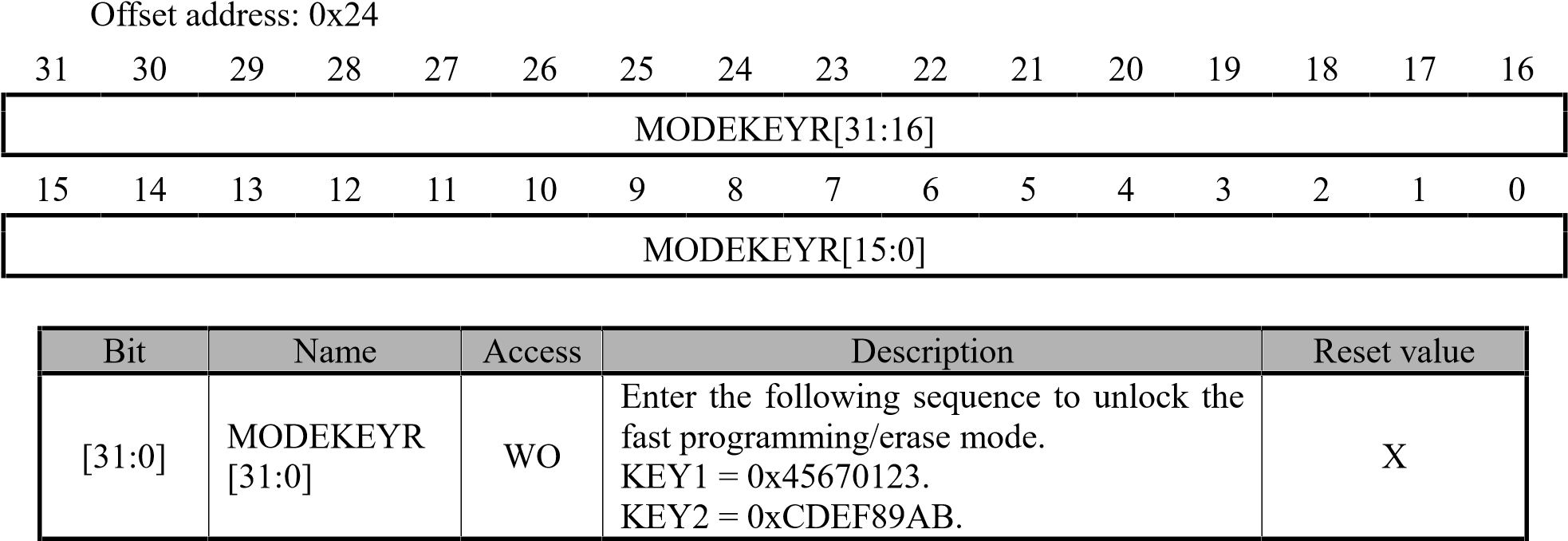
Write protection is valid.

Each bit represents 1K bytes (16 pages) of

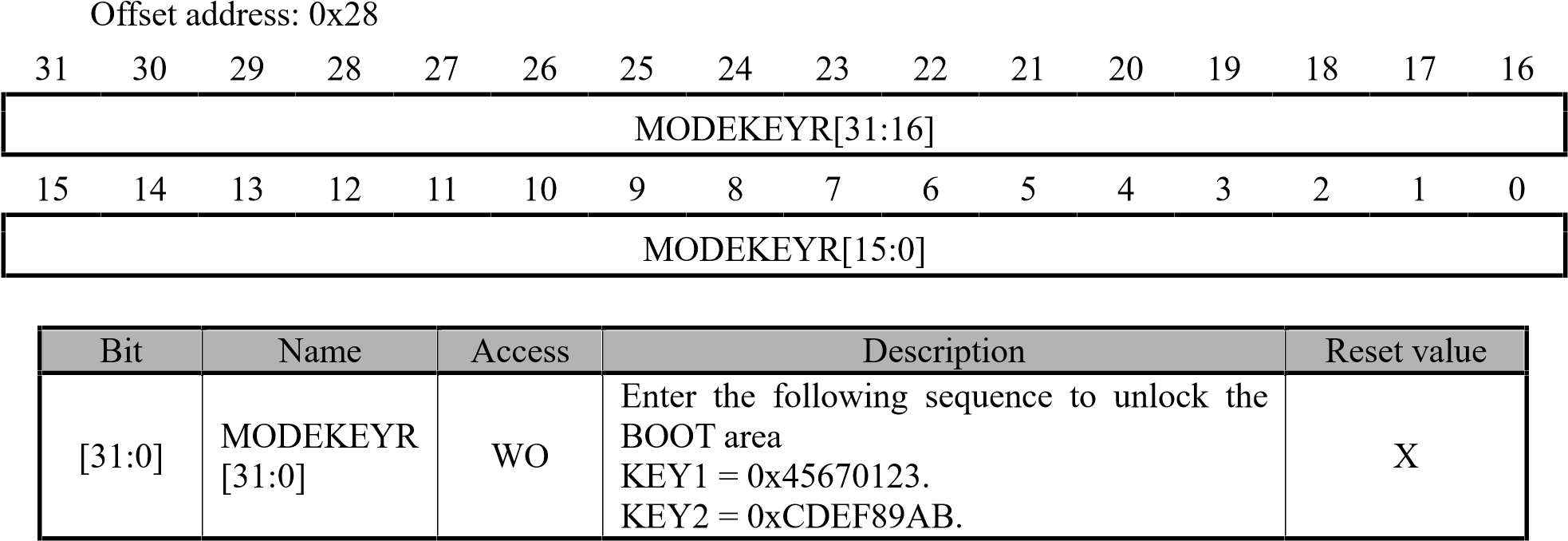
storage write protection status.

*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.

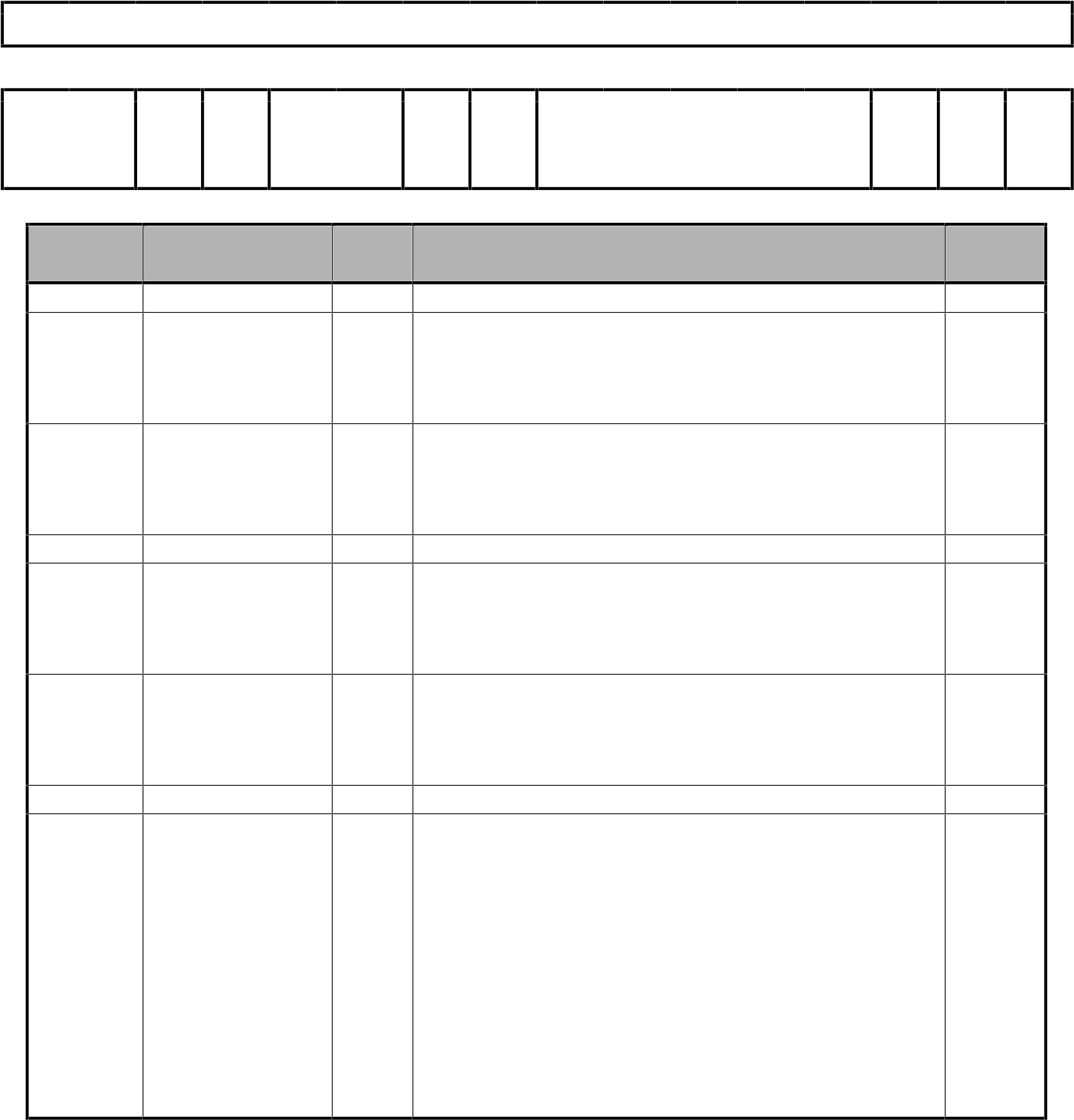
8 IWDG\_STOP RW 0

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.