RM0008
Reference manual
STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced Arm®-based 32-bit MCUs

Руководство

Пересказал А. Косенко
aikos55@gmail.com

1

[1] Кому не лень, проставьте ссылки оглавления сами.

1. Введение

Таблица 1. Где что описано.

	Low-density STM32F101xx	Medium-density STM32F101xx	High and XL-density STM32F101x	Low-density STM32F102xx	Medium-density STM32F102xx	Low-density STM32F103xx	Medium-density STM32F103xx	High and XL-density STM32F103xx	STM32F105xx	STM32F107xx
Section 2: Documentation conventions	•	•	•	•	•	•	•	•	•	•
Section 3: Memory and bus architecture	•	•	•	•	•	•	•	•	•	•
Section 4: CRC calculation unit	•	•	•	•	•	•	•	•	•	•
Section 5: Power control (PWR)	•	•	•	•	•	•	•	•	•	•
Section 6: Backup registers (BKP)	•	•	•	•	•	•	•	•	•	•
Section 7: Low-, medium-, high- and XL-density reset and clock control (RCC)	•	•	•	•	•	•	•	•		
Section 8: Connectivity line devices: reset and clock control (RCC)									•	•
Section 9: General-purpose and alternate-function I/Os (GPIOs and AFIOs)	•	•	•	•	•	•	•	•	•	•
Section 10: Interrupts and events	•	•	•	•	•	•	•	•	•	•
Section 13: Direct memory access controller (DMA)	•	•	•	•	•	•	•	•	•	•
Section 11: Analog-to-digital converter (ADC)	•	•	•	•	•	•	•	•	•	•
Section 12: Digital-to-analog converter (DAC)				•				•	•	•
Section 14: Advanced-control timers (TIM1 and TIM8)			•			•	•	•	•	•
Section 15: General-purpose timers (TIM2 to TIM5)	•	•	•	•	•	•	•	•	•	•
Section 16: General-purpose timers (TIM9 to TIM14)			•(1)					•(1)		
Section 17: Basic timers (TIM6 and TIM7)			•					•	•	•
Section 18: Real-time clock (RTC)	•	•	•	•	•	•	•	•	•	•
Section 19: Independent watchdog (IWDG)	•	•	•	•	•	•	•	•	•	•
Section 20: Window watchdog (WWDG)	•	•	•	•	•	•	•	•	•	•
Section 21: Flexible static memory controller (FSMC)			•					•		
Section 22: Secure digital input/output interface (SDIO)			•					•		
Section 23: Universal serial bus full-speed device interface (USB)				•	•	•	•	•		
Section 24: Controller area network (bxCAN)						•	•	•	•	•
Section 25: Serial peripheral interface (SPI)	•	•	•	•	•	•	•	•	•	•
Section 26: Inter-integrated circuit (I2C) interface	•	•	•	•	•	•	•	•	•	•
Section 27: Universal synchronous asynchronous receiver transmitter (USART)	•	•	•	•	•	•	•	•	•	•
Section 28: USB on-the-go full-speed (OTG_FS)									•	•
Section 29: Ethernet (ETH): media access control (MAC) with DMA controller										•
Section 30: Device electronic signature	•	•	•	•	•	•	•	•	•	•
Section 31: Debug support (DBG)	•	•	•	•	•	•	•	•	•	•

(1) Только в устройствах XL-плотности.

Таблица 2. Описания периферии.

	Backup registers (BKP)	General-purpose I/Os (GPIOs)	Analog-to-digital converter (ADC)	Digital-to-analog converter (DAC)	Advanced-control timers (TIM1&TIM8)	General-purpose timers (TIM2 to TIM5)	General-purpose timers (TIM9 to	Basic timers (TIM6&TIM7)	Real-time clock (RTC)	Independent watchdog (IWDG)	Window watchdog (WWDG)	Flexible static memory controller	Secure digital input/output interface	USB full-speed device (USB)	Controller area network (bxCAN)	Serial peripheral interface (SPI)	Inter-integrated circuit (I2C) interface	USART	USB on-the-go full-speed (OTG_FS)	Ethernet (ETH)
Section 2: Documentation conventions	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Section 3: Memory and bus architecture	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Section 4: CRC calculation unit																				
Section 5: Power control (PWR)	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Section 6: Backup registers (BKP)	●								◊											
Section 7: Low-, medium-, high- and XL-density reset and clock control (RCC)	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Section 8: Connectivity line devices: reset and clock control (RCC)	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Section 9: General-purpose and alternate-function I/Os (GPIOs and AFIOs)	◊	●	●	●	●	●	●		●	●	●	●	●	●	●	●	●	●	●	●
Section 10: Interrupts and events		◊	◊	◊	◊	◊	◊	◊	◊		◊	◊	◊	◊	◊	◊	◊	◊	◊	◊
Section 13: Direct memory access controller (DMA)			◊	◊	◊	◊	◊	◊				◊	◊			◊	◊	◊		
Section 11: Analog-to-digital converter (ADC)			●																	
Section 12: Digital-to-analog converter (DAC)				●																
Section 14: Advanced-control timers (TIM1 and TIM8)			◊	◊	●															
Section 15: General-purpose timers (TIM2 to TIM5)			◊	◊		●														
Section 16: General-purpose timers (TIM9 to TIM14)							●													
Section 17: Basic timers (TIM6 and TIM7)				◊				●												
Section 18: Real-time clock (RTC)	●								●											
Section 19: Independent watchdog (IWDG)										●										
Section 20: Window watchdog (WWDG)											●									
Section 21: Flexible static memory controller (FSMC)												●								
Section 22: Secure digital input/output interface (SDIO)													●							

	Backup registers (BKP)	General-purpose I/Os (GPIOs)	Analog-to-digital converter (ADC)	Digital-to-analog converter (DAC)	Advanced-control timers (TIM1&TIM8)	General-purpose timers (TIM2 to TIM5)	General-purpose timers (TIM9 to	Basic timers (TIM6&TIM7)	Real-time clock (RTC)	Independent watchdog (IWDG)	Window watchdog (WWDG)	Flexible static memory controller	Secure digital input/output interface	USB full-speed device (USB)	Controller area network (bxCAN)	Serial peripheral interface (SPI)	Inter-integrated circuit (I2C) interface	USART	USB on-the-go full-speed (OTG_FS)	Ethernet (ETH)
Section 23: Universal serial bus full-speed device interface (USB)														•						
Section 24: Controller area network (bxCAN)															•					
Section 25: Serial peripheral interface (SPI)																•				
Section 26: Inter-integrated circuit (I2C) interface																	•			
Section 27: Universal synchronous asynchronous receiver transmitter (USART)																		•		
Section 28: USB on-the-go full-speed (OTG_FS)																			•	
Section 29: Ethernet (ETH): media access control (MAC) with DMA controller																				•
Section 30: Device electronic signature																				
Section 31: Debug support (DBG)	◊	◊	◊	◊	◊	◊	◊	◊	◊	◊	◊	◊	◊	◊	◊	◊	◊	◊	◊	◊

Нужные секции отмечены "●"

Необязательные секции отмечены "◊"

2. Типографские соглашения

2.1. Сокращения для регистров

read/write (rw)	Чтение/запись
read-only (r)	Только чтение
write-only (w)	Только запись
read/clear (rc_w1)	Чтение/очистка записью 1, запись 0 не влияет.
read/clear (rc_w0)	Чтение/очистка записью 0, запись 1 не влияет.
read/clear by read (rc_r)	Чтение/очистка. Чтение сбрасывает в 0. Запись 0 не влияет.
read/set (rs)	Чтение/установка записью 1. Запись 0 не влияет.
read-only write trigger (rt_w)	Чтение. Запись 0 или 1 включает событие, но бита не изменяет.
toggle (t)	Software can only toggle this bit by writing '1'. Writing '0' has no effect.
Reserved (Res.)	Reserved bit, must be kept at reset value.

2.2. Словарь

- Устройства низкой плотности (**Low-density devices**) это STM32F101xx, STM32F102xx и STM32F103xx с флэш-памятью между 16 и 32 КБ.
- Устройства средней плотности (**Medium-density devices**) это STM32F101xx, STM32F102xx и STM32F103xx Устройства средней плотности 64 и 128 КБ.
- Устройства средней плотности (**High-density devices**) это STM32F101xx и STM32F103xx с флэш-памятью между 256 и 512 КБ.
- Устройства XL плотности (**XL-density devices**) это STM32F101xx и STM32F103xx с флэш-памятью между 768 КБ and 1 МБ.
- Сетевые устройства (**Connectivity line devices**) это STM32F105xx и STM32F107xx.
- Слово (**Word**): 32-битное данное.
- Полуслово (**Half-word**): 16-битное данное.
- Байт (**Byte**): 8-битное данное.

2.3. Доступность периферии

Это написано в описаниях устройств.

3. Архитектура памяти и шины

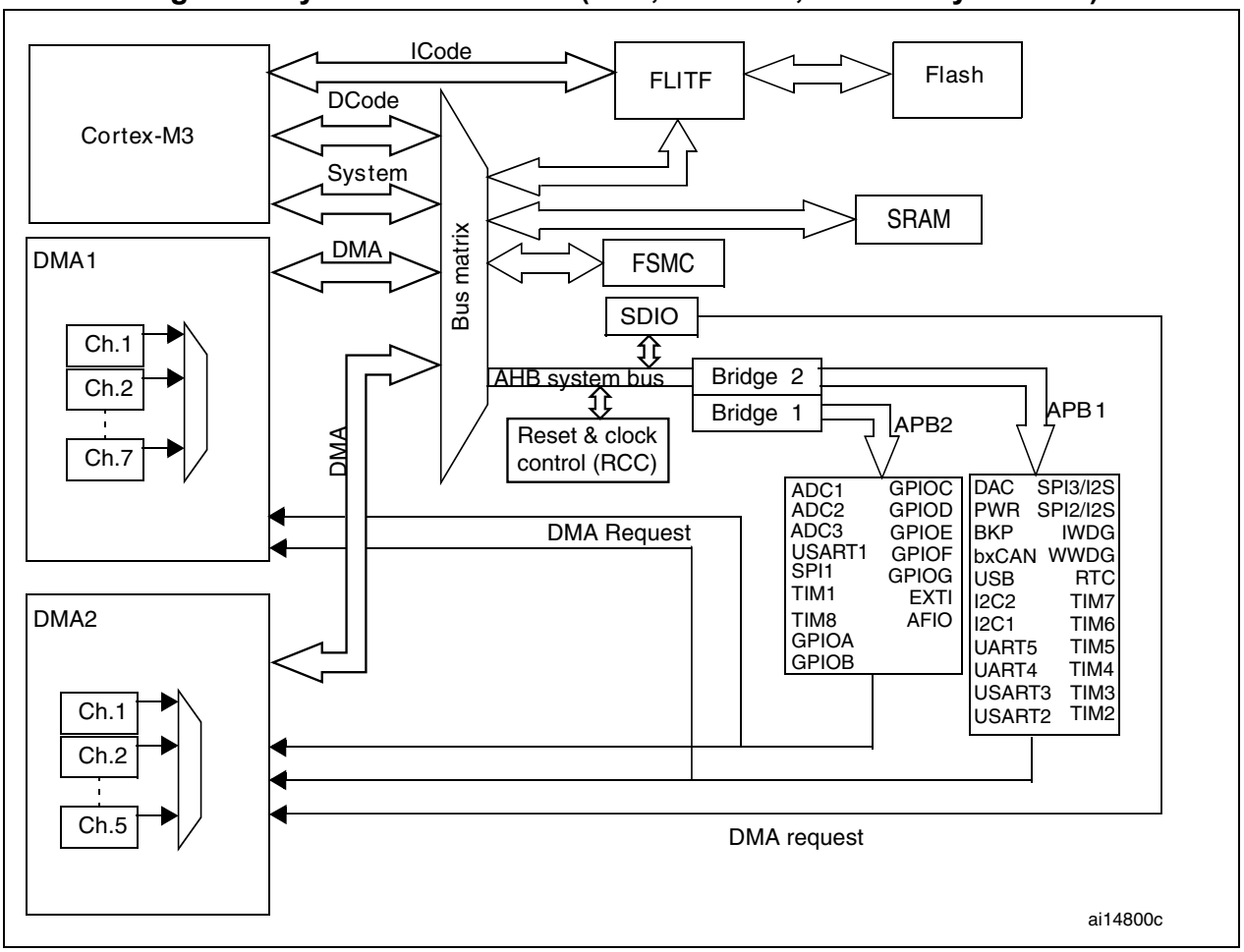
3.1. Архитектура системы

В устройствах разной плотности главная система состоит из:

- Четырёх ведущих:
  - Шин ядра Cortex®-M3 DCode bus (D-bus) и System bus (S-bus)
  - GP-DMA1 & 2 (DMA общего назначения)
- Четырёх ведомых:
  - Внутренней SRAM
  - Внутренней Флэш памяти
  - FSMC
  - AHB к APBx (APB1 или APB2), соединяющей периферию APB

Они соединены через многослойную шинную архитектуру AHB.

Figure 1. System architecture (low-, medium-, XL-density devices)

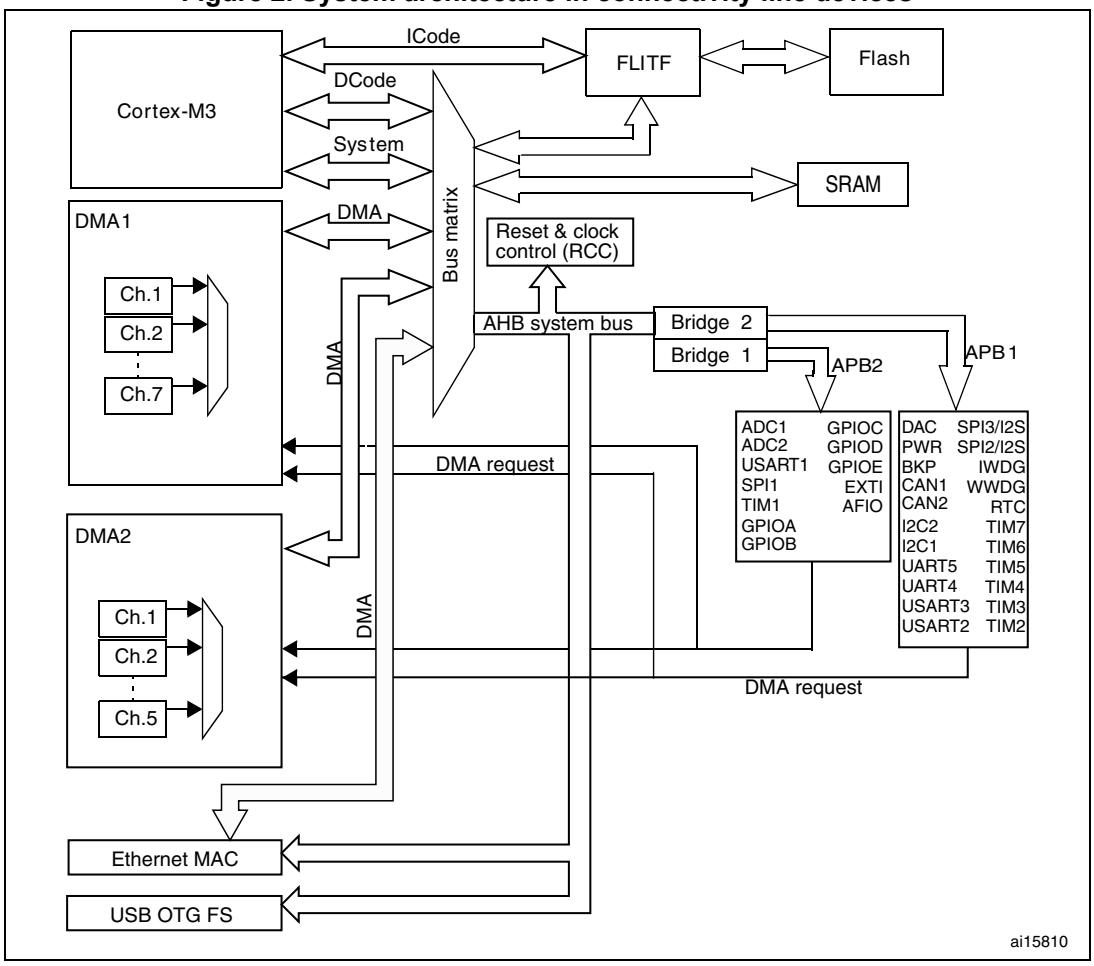


В сетевых устройствах главная система состоит из::

- Пяти ведущих:
  - Шин ядра Cortex®-M3 DCode bus (D-bus) и System bus (S-bus)
  - GP-DMA1 & 2 (DMA общего назначения)
  - Ethernet DMA
- Трёх ведомых:
  - Внутренней SRAM
  - Внутренней Флэш памяти
  - AHB к APBx (APB1 или APB2), соединяющей периферию APB

Они соединены через многослойную шинную архитектуру AHB. См. ниже.

Figure 2. System architecture in connectivity line devices



Шина ICode

По ней ядро Cortex®-M3 читает команды из флэш памяти.

Шина DCode

Обеспечивает доступ к данным (чтение и отладка) ядра Cortex®-M3 к флэш памяти.

Системная шина

Подключает периферийную шину ядра Cortex®-M3 к Шинной матрице (арбитру ядра и DMA).

Шина DMA

Соединяет AHB (ведущая шина DMA) к Шинной матрице (арбитру DCode ядра и DMA при доступе к SRAM, флэш памяти и периферии).

Шинная Матрица

Это арбитр доступа системной шины ядра и ведущей шиной DMA. Используется кольцевой алгоритм Round Robin. В сетевых устройствах она составлена из пяти ведущих (шины CPU DCode, System, Ethernet DMA, DMA1 и DMA2) и трёх ведомых (FLITF, SRAM и мосты AHB2APB). В других устройствах Шинная матрица составлена из четырёх ведущих (шины CPU DCode, System, DMA1 и DMA2) и четырёх ведомых (FLITF, SRAM, FSMC и мосты AHB2APB).

Периферия AHB подключена к системной шине через Шинную матрицу для доступа через DMA.

Мосты AHB/APB (APB)

Два моста AHB/APB дают полностью синхронное соединение между AHB и 2 шинами APB. APB1 ограничена 36 MHz, APB2 работает на полной скорости (до 72 MHz, в зависимости от устройства).

Карта адресов периферии приведена в таблице ниже.

После сброса каждого устройства их тактирование выключено (кроме SRAM и FLITF) и перед использованием его нужно включить а регистрах `RCC_AHBENR`, `RCC_APB2ENR` или `RCC_APB1ENR`.

NB. При 16- или 8-бит доступе к регистрам APB мост преобразует 16- или 8-бит данные в 32-бит вектор.

3.2. Организация памяти

Память программ, данных, регистры и порты В/Выв организованы в единое линейное 4ГБ адресное пространство.

Всё кодируется в формате Little Endian. Байт слова с меньшим номером содержит младшие биты, байт с большим номером - старшие биты.

Адресное пространство разделено на 8 основных блоков по 512МБ.

Области памяти, не реализованные в кристалле или периферии называются "резервными".

3.3. Карта памяти

Границы адресов регистров:

Границы адресов	Периферия	Шина
0xA000 0000 - 0xA000 0FFF	FSMC	AHB
0x5000 0000 - 0x5003 FFFF	USB OTG FS	
0x4003 0000 - 0x4FFF FFFF	Резерв	
0x4002 8000 - 0x4002 9FFF	Ethernet	
0x4002 3400 - 0x4002 7FFF	Резерв	
0x4002 3000 - 0x4002 33FF	CRC	
0x4002 2000 - 0x4002 23FF	Интерфейс флэш памяти	
0x4002 1400 - 0x4002 1FFF	Резерв	
0x4002 1000 - 0x4002 13FF	Сброс и тактирование (RCC)	
0x4002 0800 - 0x4002 0FFF	Резерв	
0x4002 0400 - 0x4002 07FF	DMA2	
0x4002 0000 - 0x4002 03FF	DMA1	
0x4001 8400 - 0x4001 FFFF	Резерв	
0x4001 8000 - 0x4001 83FF	SDIO	
0x4001 5800 - 0x4001 7FFF	Резерв	APB2
0x4001 5400 - 0x4001 57FF	TIM11 timer	
0x4001 5000 - 0x4001 53FF	TIM10 timer	
0x4001 4C00 - 0x4001 4FFF	TIM9 timer	
0x4001 4000 - 0x4001 4BFF	Резерв	
0x4001 3C00 - 0x4001 3FFF	ADC3	
0x4001 3800 - 0x4001 3BFF	USART1	
0x4001 3400 - 0x4001 37FF	TIM8 timer	
0x4001 3000 - 0x4001 33FF	SPI1	
0x4001 2C00 - 0x4001 2FFF	TIM1 timer	
0x4001 2800 - 0x4001 2BFF	ADC2	
0x4001 2400 - 0x4001 27FF	ADC1	
0x4001 2000 - 0x4001 23FF	GPIO Port G	

Границы адресов	Периферия	Шина
0x4001 1C00 - 0x4001 1FFF	GPIO Port F	
0x4001 1800 - 0x4001 1BFF	GPIO Port E	
0x4001 1400 - 0x4001 17FF	GPIO Port D	
0x4001 1000 - 0x4001 13FF	GPIO Port C	
0x4001 0C00 - 0x4001 0FFF	GPIO Port B	
0x4001 0800 - 0x4001 0BFF	EXTI	
0x4001 0000 - 0x4001 03FF	AFIO	
0x4000 7800 - 0x4000 FFFF	Резерв	APB1
0x4000 7400 - 0x4000 77FF	DAC	
0x4000 7000 - 0x4000 73FF	Управление питанием PWR	
0x4000 6C00 - 0x4000 6FFF	Backup registers (BKP)	
0x4000 6400 - 0x4000 67FF	bxCAN1	
0x4000 6800 - 0x4000 6BFF	bxCAN2	
0x4000 6000[(1)] - 0x4000 63FF	Общая USB/CAN SRAM 512 байт	
0x4000 5C00 - 0x4000 5FFF	FS регистры USB	
0x4000 5800 - 0x4000 5BFF	I2C2	
0x4000 5400 - 0x4000 57FF	I2C1	
0x4000 5000 - 0x4000 53FF	UART5	
0x4000 4C00 - 0x4000 4FFF	UART4	
0x4000 4800 - 0x4000 4BFF	USART3	
0x4000 4400 - 0x4000 47FF	USART2	
0x4000 4000 - 0x4000 43FF	Резерв	
0x4000 3C00 - 0x4000 3FFF	SPI3/I2S	
0x4000 3800 - 0x4000 3BFF	SPI2/I2S	
0x4000 3400 - 0x4000 37FF	Резерв	
0x4000 3000 - 0x4000 33FF	Независимый сторожевой таймер (IWDG)	
0x4000 2C00 - 0x4000 2FFF	Сторожевой таймер (WWDG)	
0x4000 2800 - 0x4000 2BFF	RTC	
0x4000 2400 - 0x4000 27FF	Резерв	
0x4000 2000 - 0x4000 23FF	TIM14 timer	
0x4000 1C00 - 0x4000 1FFF	TIM13 timer	
0x4000 1800 - 0x4000 1BFF	TIM12 timer	
0x4000 1400 - 0x4000 17FF	TIM7 timer	
0x4000 1000 - 0x4000 13FF	TIM6 timer	

Границы адресов	Периферия	Шина
0x4000 0C00 - 0x4000 0FFF	TIM5 timer	
0x4000 0800 - 0x4000 0BFF	TIM4 timer	
0x4000 0400 - 0x4000 07FF	TIM3 timer	
0x4000 0000 - 0x4000 03FF	TIM2 timer	

1. Эта общая SRAM доступна только в устройствах разной плотности, но не в сетевых.

3.3.1. Встроенная SRAM

Кристаллы STM32F10xxx имеют до 96 КБ статической SRAM. Она имеет байтовый, полусловный и словный доступ. Начинается с адреса 0x2000 0000.

3.3.2. Атомарный доступ

Память Cortex®-M3 включает два отображаемых (bit-band) региона. Здесь одно слово региона синонимов соответствует одному биту отображаемого региона. Запись в слово синоним равна операции чтения/модификации/записи в бит отображаемого региона.

В STM32F10xxx и регистры периферии и SRAM это отображаемые регионы, что позволяет выполнять словный доступ к отдельным битам. Это доступно только для ядра Cortex®-M3, но не для других ведущих устройств шин (т.е. DMA).

Отображение выполняется по формуле:

bit_word_addr = bit_band_base + (byte_offset x 32) + (bit_number × 4)

где:

bit_word_addr адрес слова в памяти синонимов

bit_band_base начальный адрес региона синонимов

byte_offset номер байта в отображаемом регионе с нужным битом

bit_number позиция нужного бита (0-7).

Пример:

Отображаем бит 2 байта в SRAM по адресу 0x20000300 в регион синонимов:

0x22006008 = 0x22000000 + (0x300*32) + (2*4).

Запись по адресу 0x22006008 эквивалентна операции чтения/модификации/записи в бит 2 байта в SRAM по адресу 0x20000300.

Чтение по адресу 0x22006008 возвращает значение (0x01 или 0x00) бита 2 байта в SRAM по адресу 0x20000300.

Совсем точно описано в *Cortex®-M3 Technical Reference Manual*.

3.3.3. Встроенная флэш память

Свойства высокопроизводительного модуля флэш памяти:

- Устройства XL плотности: до 1 МБ в двух банках с возможностью чтения-при-записи (RWW):
  - банк 1: размером 512 КБ
  - банк 2: до 512 КБ
- Другие устройства: до 512 КБ

- Организация памяти: основной и информационный блоки:
  - Основной блок:
    до 128 КБ × 64 бит в 512 страницах по 2 КБ для XL-устройств (см. *Таблицу 8*)
    до 4 КБ × 64 бит в 32 страницах по 1 КБ для устройств низкой плотности (см. *Таблицу 4*)
    до 16 КБ × 64 бит в 128 страницах по 1 КБ для устройств средней плотности (см. *Таблицу 5*)

до 64 КБ × 64 бит в 256 страницах по 2 КБ для устройств высокой плотности (см. *Таблицу 6*)
до 32 Кб × 64 бит в 128 страницах по 2 КБ для сетевых устройств (см. *Таблицу 7*)

– Информационный блок:
770 × 64 бит для XL-устройств (см. *Таблицу 8*)
2360 × 64 бит для сетевых устройств (см. *Таблицу 7*)
258 × 64 бит других устройств (см. *Таблицу 4*, *Таблицу 5* и *Таблицу 6*)

Свойства интерфейса флэш памяти (FLITF):

- Интерфейс чтения с буфером предвыборки (2x64-бит слова)
- Необязательный байтовый Загрузчик
- Операция Программирования/Стирания
- Защита Чтения/Записи

Таблица 4. Устройства низкой плотности

Блок	Имя	Базовый адрес	Размер (байт)
Основной	Page 0	0x0800 0000 - 0x0800 03FF	1 Kbyte
	Page 1	0x0800 0400 - 0x0800 07FF	1 Kbyte
	Page 2	0x0800 0800 - 0x0800 0BFF	1 Kbyte
	Page 3	0x0800 0C00 - 0x0800 0FFF	1 Kbyte
	Page 4	0x0800 1000 - 0x0800 13FF	1 Kbyte
	. . .	. . .	. . .
	Page 31	0x0800 7C00 - 0x0800 7FFF	1 Kbyte
Информационный	Системная память	0x1FFF F000 - 0x1FFF F7FF	2 Kbytes
	Байты Опций	0x1FFF F800 - 0x1FFF F80F	16
Регистры интерфейса флэш памяти	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Резерв	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Таблица 5. Устройства средней плотности

Блок	Имя	Базовый адрес	Размер (байт)
Основной	Page 0	0x0800 0000 - 0x0800 03FF	1 Kbyte
	Page 1	0x0800 0400 - 0x0800 07FF	1 Kbyte
	Page 2	0x0800 0800 - 0x0800 0BFF	1 Kbyte
	Page 3	0x0800 0C00 - 0x0800 0FFF	1 Kbyte
	Page 4	0x0800 1000 - 0x0800 13FF	1 Kbyte
	. . .	. . .	. . .
	Page 127	0x0801 FC00 - 0x0801 FFFF	1 Kbyte
Информационный	Системная память	0x1FFF F000 - 0x1FFF F7FF	2 Kbytes
	Байты Опций	0x1FFF F800 - 0x1FFF F80F	16
Регистры интерфейса флэш памяти	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Резерв	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Таблица 6. Устройства высокой плотности

Блок	Имя	Базовый адрес	Размер (байт)
Основной	Page 0	0x0800 0000 - 0x0800 07FF	2 Kbytes
	Page 1	0x0800 0800 - 0x0800 0FFF	2 Kbytes
	Page 2	0x0800 1000 - 0x0800 17FF	2 Kbytes
	Page 3	0x0800 1800 - 0x0800 1FFF	2 Kbytes
	. . .	. . .	. . .
	Page 255	0x0800 1800 - 0x0800 1FFF	2 Kbytes
Информационный	Системная память	0x1FFF F000 - 0x1FFF F7FF	2 Kbytes
	Байты Опций	0x1FFF F800 - 0x1FFF F80F	16
Регистры интерфейса флэш памяти	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Резерв	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Таблица 7. Сетевые устройства

Блок	Имя	Базовый адрес	Размер (байт)
Основной	Page 0	0x0800 0000 - 0x0800 07FF	2 Kbytes
	Page 1	0x0800 0800 - 0x0800 0FFF	2 Kbytes
	Page 2	0x0800 1000 - 0x0800 17FF	2 Kbytes
	Page 3	0x0800 1800 - 0x0800 1FFF	2 Kbytes
	. . .	. . .	. . .
	Page 127	0x0803 F800 - 0x0803 FFFF	2 Kbytes
Информационный	Системная память	0x1FFF B000 - 0x1FFF F7FF	8 Kbytes
	Байты Опций	0x1FFF F800 - 0x1FFF F80F	16
Регистры интерфейса флэш памяти	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Резерв	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Таблица 8. Устройства XL плотности

Блок		Имя	Базовый адрес	Размер (байт)
Основной	Банк 1	Page 0	0x0800 0000 - 0x0800 07FF	2 Kbytes
		Page 1	0x0800 0800 - 0x0800 0FFF	2 Kbytes
		. . .	. . .	. . .
		Page 255	0x0807 F800 - 0x0807 FFFF	2 Kbytes
	Банк 2	Page 256	0x0808 0000 - 0x0808 07FF	2 Kbytes
		Page 257	0x0808 0800 - 0x0808 0FFF	2 Kbytes
		. . .	. . .	. . .
		Page 511	0x080F F800 - 0x080F FFFF	2 Kbytes
Информационный		Системная память	0x1FFF E000 - 0x1FFF F7FF	6 Kbytes
		Байты Опций	0x1FFF F800 - 0x1FFF F80F	16
Регистры интерфейса флэш памяти		FLASH_ACR	0x4002 2000 - 0x4002 2003	4
		FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
		FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
		FLASH_SR	0x4002 200C - 0x4002 200F	4
		FLASH_CR	0x4002 2010 - 0x4002 2013	4
		FLASH_AR	0x4002 2014 - 0x4002 2017	4
		Резерв	0x4002 2018 - 0x4002 201B	4
		FLASH_OBR	0x4002 201C - 0x4002 201F	4
		FLASH_WRPR	0x4002 2020 - 0x4002 2023	4
		Резерв	0x4002 2024 - 0x4002 2043	32
		FLASH_KEYR2	0x4002 2044 - 0x4002 2047	4
		Резерв	0x4002 2048 - 0x4002 204B	4
		FLASH_SR2	0x4002 204C - 0x4002 204F	4
		FLASH_CR2	0x4002 2050 - 0x4002 2053	4
		FLASH_AR2	0x4002 2054 - 0x4002 2057	4

NB. Основная и подробная информация о регистрах управления в документах: *“STM32F10xxx XL-density Flash programming manual”* (PM0068) для XL устройств и *“STM32F10xxx Flash programming manual”* (PM0075) для других устройств.

Чтение флэш памяти

Доступ к флэш памяти за командами и данными идёт по шине AHB. Блок предвыборки команд используется через шину ICode. Арбитраж выполняется в интерфейсе флэш памяти и приоритет даётся данным по шине DCode.

Опции конфигурации чтения:

- Задержка: число состояний ожидания операции чтения программируется "налету"
- Буфер предвыборки (2 x 64-бит блока): включён после сброса. Чтение из памяти выполняется целым блоком за раз (такова ширина памяти), это ускоряет работу процессора
- Половинный цикл: это оптимизация потребления.

NB. Эти опции должны соответствовать времени доступа в флэш памяти.

Состояния ожидания представляют отношение периода SYSCLK (system clock) и времени доступа к флэш памяти:

- без циклов ожидания, если 0 < SYSCLK ≤ 24 MHz
- один цикл ожидания, если 24 MHz < SYSCLK ≤ 48 MHz
- два цикла ожидания, если 48 MHz < SYSCLK ≤ 72 MHz

Конфигурация половинного цикла в сочетании с предделителем AHB невозможна. Системные такты (SYSCLK) должны быть равны тактам HCLK. Эту опцию можно использовать только с частотой тактов 8 MHz и меньше. Её можно получить из HSI или HSE, но не PLL (ФАПЧ).

Буфер предвыборки должен быть включён, если предделитель тактов AHB не равен 1.

Буфер предвыборки должен включаться/выключаться только если SYSCLK меньше 24 MHz и предделитель частоты AHB не используется (SYSCLK должен быть равен HCLK). Обычно буфер предвыборки включается/выключается в программе инициализации, когда микроконтроллер работает от внутреннего 8 MHz RC генератора (HSI).

Использование DMA: DMA обращается к флэш памяти по шине DCode и приоритетнее, чем доступ по ICode. DMA пропускает один свободный цикл после каждой передачи. Некоторые команда можно выполнять одновременно с передачами DMA.

Программирование и стирание флэш памяти

Программировать за раз можно 16 бит (полуслово).

При программировании/стирании флэш памяти внутренний RC генератор (HSI) должен быть ВКЛЮЧЁН.

Операцию стирания можно выполнять на уровне страницы или всей флэш области целиком (mass-erase). Такая операция не влияет на информационные блоки.

Во избежание пере-программирования блоки Контроллера программирования и стирания тактируются фиксированной частотой.

Конец операции записи/стирания может включать прерывание. можно использовать для выхода из режима WFI только если разрешены такты FLITF. В противном случае прерывание будет обработано только после выхода из режима WFI.

Для включения/выключения предвыборки, доступа половинного цикла и управления временем доступа к флэш памяти в соответствии с частотой CPU используется регистр `FLASH_ACR`.

Полная информация о работе флэш памяти приведена в *STM32F10xxx Flash programming manual* (PM0075) или для XL в *STM32F10xxx Flash programming manual* (PM0068).

Регистр управления доступом `FLASH_ACR`

Смещение: 0x00

По сбросу: 0x0000 0030

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										PRFTBS	PRFTBE	HLFCYA	LATENCY		
										r	rw	rw	rw	rw	rw

— Биты 31:6 Резерв, изменять нельзя.

— Бит 5 **PRFTBS**: Состояние буфера предвыборки

0: Выключен.

1: Включён.

— Бит 4 **PRFTBE**: Разрешение буфера предвыборки

0: Выключен.

1: Включён.

— Бит 3 **HLFCYA**: Разрешение половинного цикла доступа

0: Выключен.

1: Включён.

— Биты 2:0 **LATENCY**: Отношение периода SYSCLK и времени доступа к флэш памяти.
000 Без циклов ожидания, если 0 < SYSCLK≤ 24 MHz
001 Один цикл ожидания, если 24 MHz < SYSCLK ≤ 48 MHz
010 Два цикла ожидания, если 48 MHz < SYSCLK ≤72 MHz

3.4. Конфигурация загрузки

В STM32F10xxx есть 3 режима загрузки, определяемых ножками BOOT[1:0].

BOOT[1]	BOOT[0]	Откуда грузить
x	0	Основная флэш память
0	1	Системная память
1	1	Встроенная SRAM

Значения на ножках BOOT запоминаются на 4-м переднем фронте SYSCLK после сброса. Для выбора режима загрузки классно устанавливать ножки BOOT1 и BOOT0 после Сброса.

Также ножки BOOT опрашиваются при выходе из режима Standby. Следовательно, в этом режиме они должны быть в нужном состоянии. После истечения этой задержки CPU извлекает указатель стека по адресу 0x0000 0000 и начинает выполнение с адреса, хранящегося в векторе 0x0000 0004.

Из-за фиксированной карты памяти область команд начинается с адреса 0x0000 0000 (доступ через шины ICode/DCode), тогда как область данных (SRAM) начинается с адреса 0x2000 0000 (доступ через системную шину). Cortex®-M3 CPU всегда извлекает вектор сброса через шину ICode, что подразумевает область загрузки только в пространстве команд (обычно, флэш память). Микроконтроллеры STM32F10xxx имеют механизм загрузки из SRAM, а не только из основной флэш памяти или системной памяти.

В зависимости от выбранного режима загрузки флэш память, системная память или SRAM доступны так:

- Флэш память: основная флэш память картируется на адреса загрузки (0x0000 0000), но одновременно доступна и по её родному адресу (0x800 0000).
- Системная память: системная память картируется на адреса загрузки (0x0000 0000), но одновременно доступна и по её родному адресу (0x1FFF B000 в сетевых устройствах, 0x1FFF F000 в других устройствах).
- Встроенная SRAM: SRAM доступна только по адресу 0x2000 0000.

NB. При загрузке из SRAM вам надо в коде инициализации программы с помощью регистра смещения таблицы исключений NVIC переместить таблицу векторов в SRAM.

В XL устройствах есть возможность загрузиться из любого банка основной флэш памяти. По умолчанию выбран Банк 1. Банк 2 выбирается очисткой бита `BFB2` в байтах опций пользователя. Если он чист и на ножках загрузки выбрана основная флэш память, то устройство загружается из системной памяти и загрузчик переходит на выполнение программы в Банке 2. За деталями идите в AN2606.

NB. При загрузке из Банка 2 вам надо в коде инициализации программы с помощью регистра смещения таблицы исключений NVIC переместить таблицу векторов в его базовый адрес (0x0808 0000).

Встроенный загрузчик

Встроенный загрузчик размещается в Системной памяти при производстве кристалла. Он используется для программирования флэш памяти через один из доступных последовательных интерфейсов:

- В устройствах низкой, средней и высокой плотности загрузчик активируется через интерфейс USART1.
- В XL устройствах загрузчик активируется через интерфейсы USART1 или USART2 (переключается).

- В сетевых устройствах загрузчик активируется через интерфейсы: USART1, USART2 (переключается), CAN2 (переключается) или USB OTG FS в режиме Устройства (DFU: device firmware upgrade).

Периферия USART работает от внутреннего генератора 8 MHz (HSI). CAN и USB OTG FS, работают при наличии внешнего, 8 MHz, 14.7456 MHz или 25 MHz сигнала (HSE).

За деталями снова идите в AN2606.

4. Блок вычисления CRC

4.1. Введение в CRC

Как и другие алгоритмы стандарта EN/IEC 60335-1, техника CRC используется для проверки целостности передачи и хранения данных. Блок вычисления CRC помогает получить сигнатуру программы при выполнении и сравнить её с полученной при компоновке и хранящейся по определённому адресу.

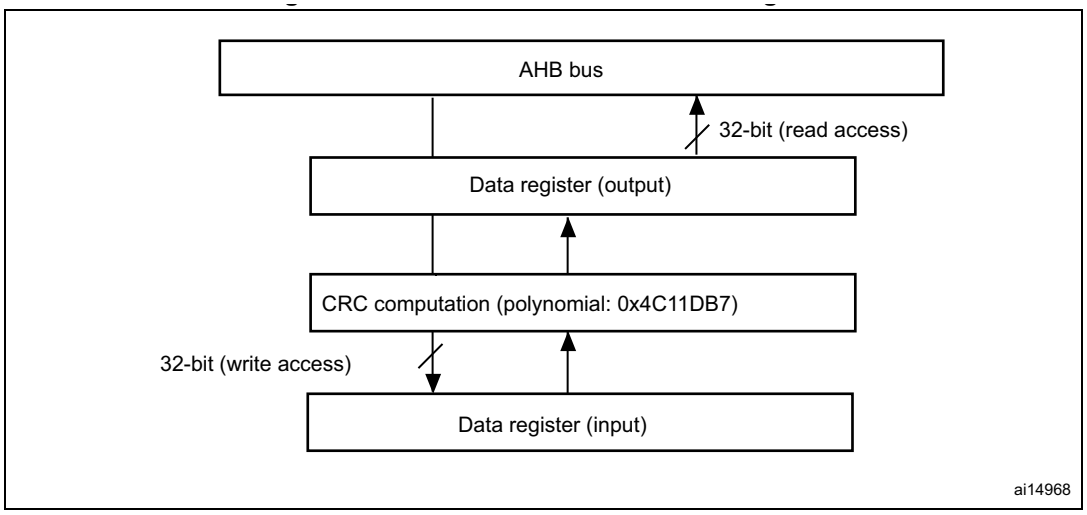
4.2. Основные свойства CRC

- Использует CRC-32 (Ethernet) полином: 0x4C11DB7

– $X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$

- Один 2-бит регистр ввода/вывода данных
- CRC вычисляется за 4 такта AHB (HCLK)
- Доступный 8-бит регистр общего назначения

Блок-схема.



4.3. Функциональное описание CRC

Блок вычисления CRC в основном состоит из одного 32-бит регистра данных, который:

- при записи получает новое данное для калькулятора CRC
- при чтении содержит результат предыдущего вычисления

Каждая запись в регистр данных создаёт новый результат с предыдущим CRC (CRC вычисляется 32-бит словами данных целиком, а не байт за байтом).

Операция записи задерживается до конца вычисления CRC, что позволяет и непрерывные операции записи и чередовать запись и чтение.

Калькулятор CRC сбрасывается в 0xFFFF FFFF битом `RESET` в регистре `CRC_CR`. Эта операция не влияет на содержимое регистра `CRC_IDR`.

4.4. Регистры CRC

Блок вычисления CRC содержит два регистра данных и регистр управления, доступных как 32-бит слова.

4.4.1. Регистр данных (CRC_DR)

При записи получает новое данное, при чтении содержит результат предыдущей операции.

Смещение адреса: 0x00

По сбросу: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR [31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR [15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

4.4.2. Независимый регистр данных (CRC_IDR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000

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

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

Биты 31:8 Резерв, изменять нельзя.

Биты 7:0 8-бит регистр общего назначения. Сброс CRC битом RESET в регистре CRC_CR на него не влияет.

4.4.3. Регистр управления (CRC_CR)

Смещение адреса: 0x08

По сбросу: 0x0000 0000

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

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

— Биты 31:8 Резерв, изменять нельзя.

— Бит 7:0 **RESET** Запись 1 сбрасывает CRC и пишет в CRC_DR число 0xFFFF FFFF. Очищается автоматически.

4.4.4. Карта регистров CRC

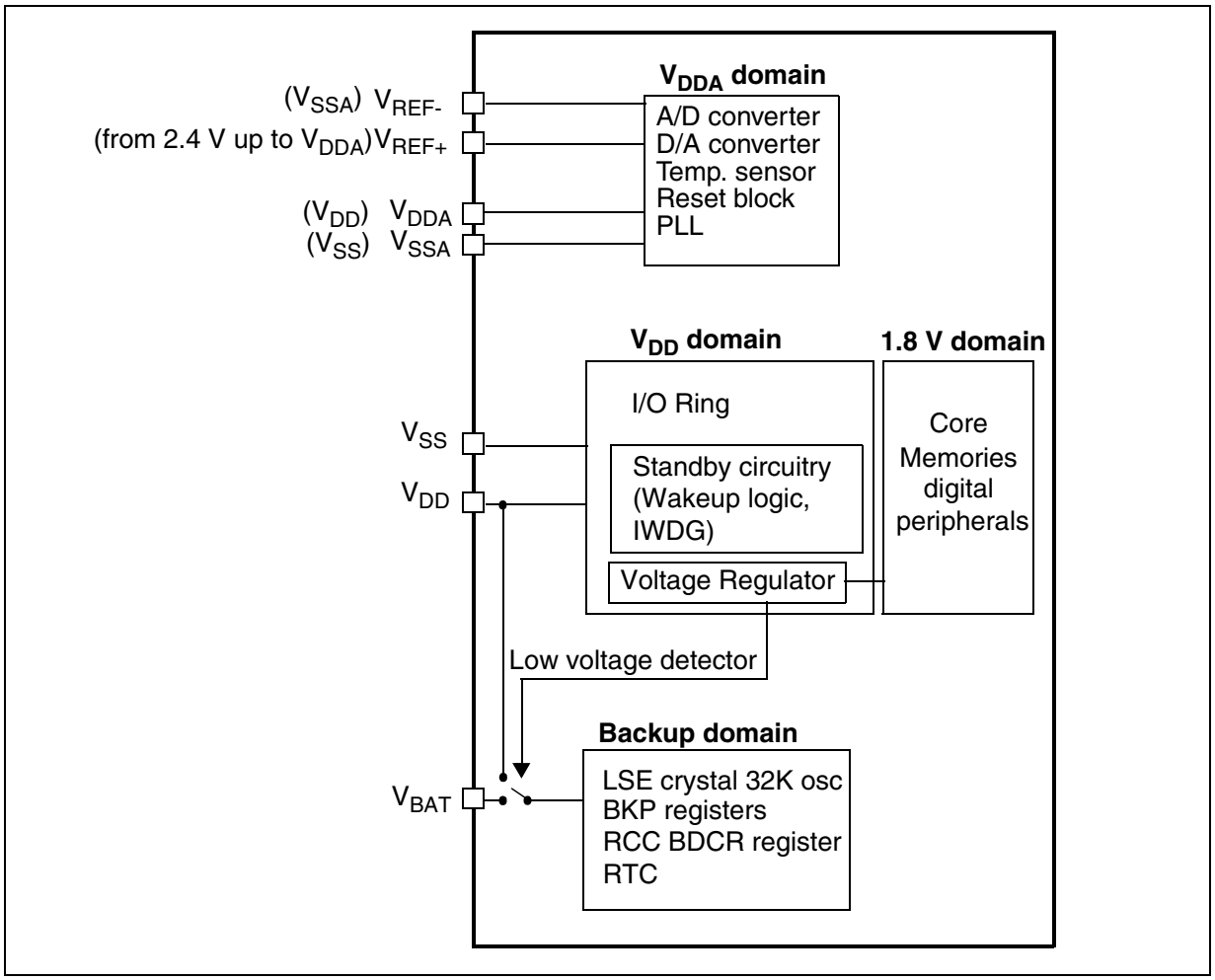
Offset	Register	31-24	23-16	15-8	7	6	5	4	3	2	1	0
0x00	**CRC_DR**	Data register										
	Reset value	0xFFFF FFFF										
0x04	**CRC_IDR**	Reserved			Independent data register							
	Reset value				0x00							
0x08	**CRC_CR**	Reserved										RESET
	Reset value											0

5. Управление питанием PWR

5.1. Источники питания

Устройство требует напряжения от 2.0 до 3.6 V (V_{DD}). Встроенный регулятор даёт 1.8 V питания.

Часы реального времени (RTC) и регистры хранения могут питаться от V_{BAT} при выключенном V_{DD}.



1. V_{DDA} and V_{SSA} must be connected to V_{DD} and V_{SS}, respectively.

5.1.1. Независимое питание ADC и DAC и опорное напряжение

Для повышения точности преобразования, ADC и DAC имеют независимое питание, которое можно отдельно фильтровать и защитить от шума PCB.

- Питание ADC и DAC подаётся на отдельную ножку V_{DDA}.
- Земля изолированного источника подключается к ножке V_{SSA}.

Если позволяет корпус, то V_{REF-} должно подключать к V_{SSA}.

Корпуса с 100 и 144 ножками

Для лучшей точности низковольтных входов и выходов можно подключать внешний источник опорного напряжения, подключённый к V_{REF+}. V_{REF+} это наибольшее напряжение, представленное полным значением ADC и DAC. Напряжение V_{REF+} имеет значение от 2.4 V до V_{DDA}.

Корпуса с 64 ножками и меньше

Ножки V_{REF+} и V_{REF-} внутренне подключены к питанию ADC (V_{DDA}) и земле (V_{SSA}).

5.1.2. Резервное батарейное питание

Для сохранения содержимого регистров хранения и работы RTC при выключенном V_{DD} ножку V_{BAT} можно подключать к необязательному источнику питания от батареи или иному источнику.

Ножка V_{BAT} питает блок RTC, генератор LSE контакты от PC13 до PC15 IO, позволяя RTC работать при выключенном V_{DD}. Включение питания V_{BAT} управляется Power Down Reset в блоке Сброса.

Внимание:

Во время $t_{RSTTEMPO}$ (ожидание при запуске а V_{DD}) или после обнаружения PDR переключатель питания между V_{BAT} и V_{DD} остаётся подключённым к V_{BAT}.

Во время фазы запуска, если V_{DD} меньше чем $t_{RSTTEMPO}$ и $V_{DD} > V_{BAT} + 0.6$ V, ток в V_{BAT} может подаваться через внутренний диод между V_{DD} и переключателем питания (V_{BAT}).

Если подключённый к V_{BAT} источник/батарея не может обеспечить подачу этого тока, то весьма рекомендуется между источником и V_{BAT} подключить внешний диод с низким падением напряжения.

Если внешней батареи нет, то рекомендуется подключать V_{BAT} к V_{DD} через внешний керамический 100 nF конденсатор развязки (см. документ AN2586).

При питании домена резервного хранения от V_{DD} (аналоговый ключ подключён к V_{DD}) доступны следующие функции:

- PC14 и PC15 можно использовать как GPIO или LSE
- PC13 можно использовать как GPIO, контакт TAMPER, RTC Calibration Clock, RTC Alarm или Выход секунд. См. *Главу 6*.)

Из-за того, что ключ пропускает только 3 mA тока, использование GPIO от PC13 до PC15 на вывод ограничено: скорость надо ограничить до 2 MHz с максимальной нагрузкой 30 pF и эти IO нельзя использовать как источник тока (например, управление светодиодами).

При питании домена резервного хранения от V_{BAT} (V_{DD} нету) доступны следующие функции:

- PC14 и PC15 можно использовать только как контакты LSE
- PC13 можно использовать как TAMPER, RTC Alarm или Выход секунд. См. *Подраздел 6.4.2*).

5.1.3. Регулятор напряжения

После Сброса регулятор напряжения всегда включён и может работать в трёх режимах.

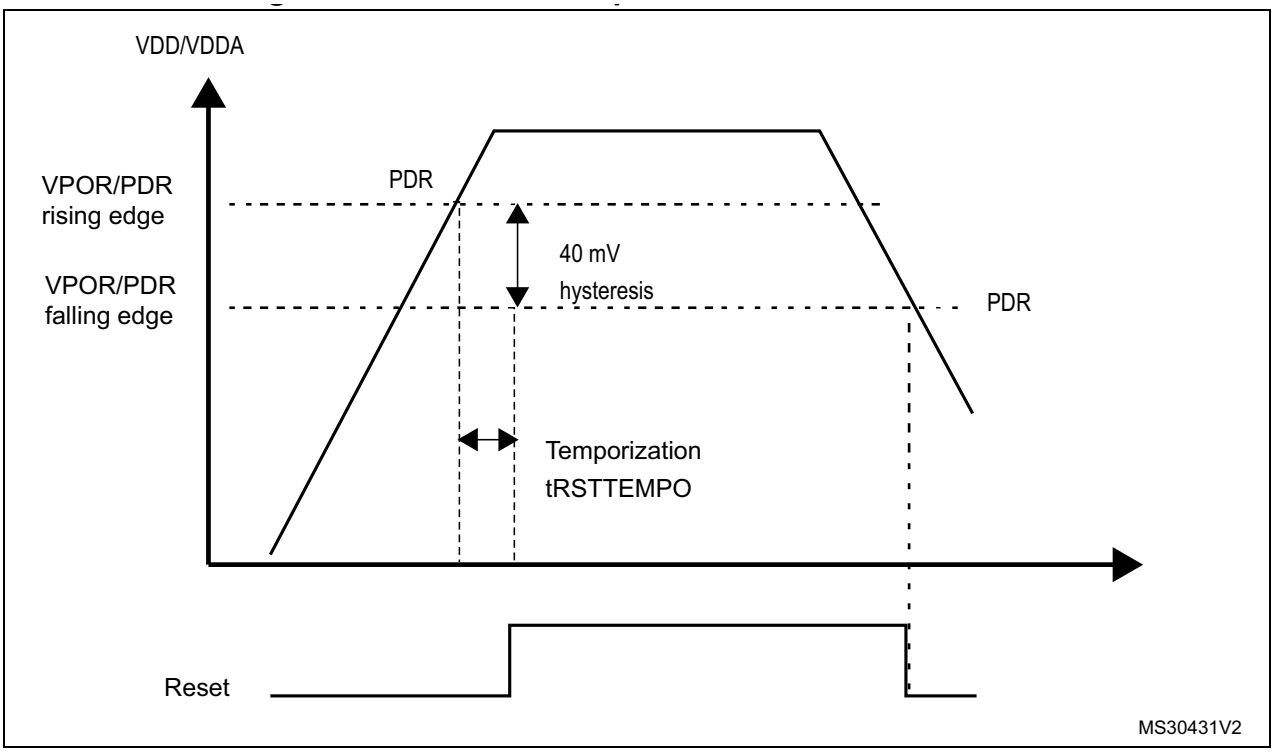
- Режим Run: подаётся полное питание на 1.8 V домен (ядро, память и цифровая периферия).
- Режим Stop: подаётся пониженное питание на 1.8 V домен, сохраняя регистры и SRAM
- Режим Standby: регулятор выключен. Содержимое регистров и SRAM теряется кроме схем Standby и домена резервного хранения.

5.2. Супервизор питания

5.2.1. Сброс по включению (POR)/сброс по выключению (PDR)

Встроенная схема POR/PDR обеспечивает правильную работу с/до 2 V.

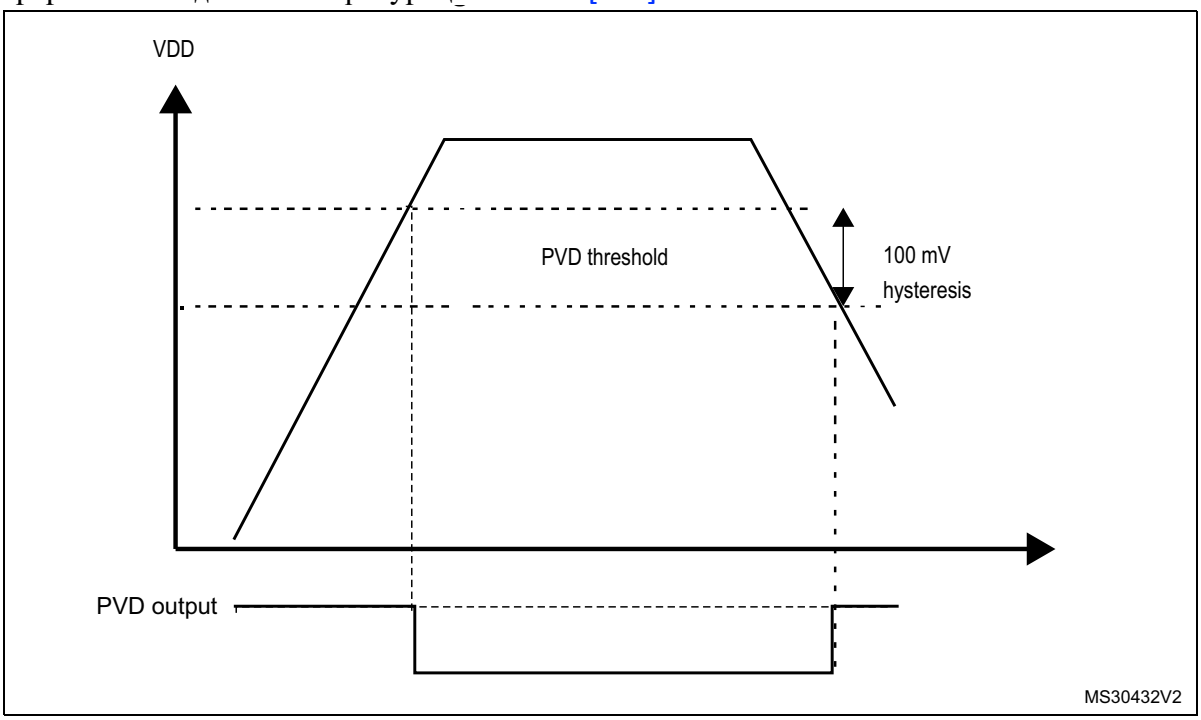
Устройство остаётся в режиме Сброса при V_{DD}/V_{DDA} ниже установленного порога, $V_{POR/PDR}$, без нужды во внешней схеме сброса. Подробности в описании устройств.



5.2.2. Программируемый детектор напряжения (PVD)

PVD отслеживает уровень питания V_{DD}/V_{DDA} сравнивая его с порогом, указанным в битах `PLS[2:0]` регистра управления питанием (`PWR_CR`). Включается PVD установкой бита `PVDE`.

Флаг `PVDO` в регистре управления/состояния питания (`PWR_CSR`) показывает, что V_{DD}/V_{DDA} выше или ниже порога PVD. Это событие внутренне подключено к и может вызвать прерывание при его разрешении в регистрах `EXTI`. Направление перехода V_{DD}/V_{DDA} через порог PVD для генерации прерывания задаётся конфигурацией `EXTI[16]`.



5.3. Режимы пониженного потребления

По умолчанию после Сброса системы или питания микроконтроллер работает в режиме Run. Однако есть в наличии и режимы пониженного потребления, когда CPU совсем делать нечего. Это круто выбрать нужный компромисс между низким потреблением, быстрым запуском и ресурсами пробуждения.

В STM32F10xxx есть три экономичных режима:

- Сон (Sleep) — (тактов CPU нет, вся периферия, включая like NVIC, SysTick и т.д., работает)

- Останов (Stop) (все тики остановлены)
- Дежурный (Standby) (1.8V домен выключен)

Кроме того, потребление можно понизить и в режиме Run (Забег):

- Замедляя системные тики
- Закрывая тики для ненужной периферии APB и AHB.

Режимы низкого потребления

Режим	Вкл.	Выкл.	Такты 1.8V домена	Такты V_{DD} домена	Регулятор напряжения
Sleep (Sleep now or Sleep-on -exit)	WFI	Прерывание	Выкл. только CPU	Не влияет	ON
	WFE	Событие			
Stop	Биты PDDS и LPDS + бит SLEEPDEEP + WFI или WFE	Любая линия EXTI (выбор регистрами EXTI)	Все такты 1.8V домена выключены	Генераторы HSI и HSE выключены	ON или низкое потребление (см. регистр PWR_CR)
Standby	Бит PDDS + бит SLEEPDEEP + WFI или WFE	Передний фронт WKUP, RTC alarm, внешний сброс на NRST, сброс IWDG			OFF

5.3.1. Замедление системных тактов

В режиме Run частоту системных тиков (SYSCLK, HCLK, PCLK1, PCLK2) можно снизить регистрами предделителей. Ими также можно замедлить периферию перед входом в режим Sleep mode. Точнее см. в *Секции 7.3.2*.

5.3.2. Выключение тактов периферии

В режиме Run такты HCLK и PCLKx для периферии и памяти можно выключать, а для режима Sleep их можно выключить перед командами `WFI` или `WFE`.

Тактирование периферии AHB управляется регистром (`RCC_AHBENR`), APB1 - регистром (`RCC_APB1ENR`) и APB2 - регистром (`RCC_APB2ENR`).

5.3.3. Режим сна

Вход

Режим Sleep включается командами `WFI` (Wait For Interrupt) или `WFE` (Wait for Event). Две опции режима выбираются битом `SLEEPONEXIT` в регистре System Control:

- Sleep-now: если бит `SLEEPONEXIT` чист, то MCU включает режим сразу после выполнения команд `WFI` или `WFE`.
- Sleep-on-exit: если бит `SLEEPONEXIT` стоит, то MCU включает режим сразу после выхода из ISR с наименьшим приоритетом.

В режиме Sleep все ножки I/O сохраняют то же состояние, что и в режиме Run. В Таблицах 12 и 13 приведены детали входа/выхода режима сна.

Выход

Если для входа в режим использовалась команда `WFI`, то для выхода из него может послужить любое прерывание от NVIC.

Если использовалась команда `WFE`, то MCU выходит из него при появлении события. Оно генерируется так:

- разрешением прерывания в регистрах периферии, но не в NVIC и разрешением бита SEVONPEND в регистре System Control. При выходе MCU из WFE бит удержания периферии и бит удержания канала IRQ в NVIC (в регистре очистки удержания NVIC) надо очищать.
- или конфигурацией внешней или внутренней линии EXTI в режим события. При выходе CPU из WFE очищать бит удержания периферии или канала IRQ в NVIC чистить не надо, в ситуации события они и не ставились.

В этом режиме время восстановления самое маленькое, без затрат на прерывание.

Таблица 12. Sleep-now

Sleep-now	Описание
Вход	WFI (Wait for Interrupt) или WFE (Wait for Event) когда: – SLEEPDEEP = 0 и – SLEEPONEXIT = 0.
Выход	При входе по WFI: Прерывание: См. Секцию 10.1.2: Векторы исключения и прерываний При входе по WFE Событие: См. Секцию 10.2.3: События пробуждения
Задержка пробуждения	Нету

Таблица 13. Sleep-on-exit

Sleep-on-exit	Описание
Вход	WFI (Wait for Interrupt) когда: – SLEEPDEEP = 0 и – SLEEPONEXIT = 1.
Выход	Прерывание: См. Секцию 10.1.2: Векторы исключения и прерываний
Задержка пробуждения	Нету

5.3.4. Режим остановки

Режим Stop основан на сочетании режима *deepsleep* с отключением тактирования периферии. Регулятор напряжения можно переключать в нормальный или экономный режим. В режиме Stop тактирование 1.8V останавливается, RC генераторы PLL, HSI и HSE выключаются. SRAM и регистры сохраняются.

В режиме Stop все контакты I/O сохраняют состояние режима Run.

Вход

Как входить написано Таблице 14. Ради экономии можно битом LPDS в регистре PWR_CR переключить в бережливый режим и регулятор напряжения.

Во время программирования флэш памяти переход в режим Stop откладывается до завершения доступа к памяти.

Во время работы домена APB переход в режим Stop откладывается до завершения доступа APB.

В режиме Stop отдельными битами можно включить:

- Независимый сторожевой таймер (IWDG): IWDG запускается записью в его регистр Ключа или аппаратно. Запущенный однажды он останавливается только Сбросом. См. *Секцию 19.3*.
- Real-time clock (RTC): битом RTCEN в регистре RCC_BDCR
- Внутренний RC генератор (LSI RC): битом LSION в регистре RCC_CSR.
- Внешний 32.768 kHz генератор (LSE OSC): битом LSEON в регистре RCC_BDCR.

В режиме Stop блоки ADC и DAC могут кушать электричество, даже если отключить их. Во избежание этого нужно записать 0 в бит ADON регистра ADC_CR2 и в ENx регистра DAC_CR.

NB. Если перед остановкой нужно выключить внешний тактовый генератор, то его надо отключить битом HSEON и переключиться на внутренний HSI. Иначе, если бит HSEON остаётся включённым и внешние такты при остановке устраняются, то должна быть включена система безопасности тактирования (CSS) для обнаружения сбоев внешнего тактирования и предупреждения недостойного поведения системы.

Выход

Как выходить написано Таблице 14.

При выходе из режима Stop по прерыванию или событию в качестве системных тактов выбирается RC генератор HSI.

При выходе из режима Stop при регуляторе напряжения в экономном режиме появляется дополнительная задержка, но включённый регулятор напряжения в режиме Stop увеличивает потребление тока.

Таблица 14. Режим Остановки.

Режим Stop	Описание
Вход	WFI (Wait for Interrupt) или WFE (Wait for Event): – Установить бит SLEEPDEEP в регистре System Control – Очистить бит PDDS в регистре Power Control (PWR_CR) – Выбрать режим регулятора напряжения битом LPDS в PWR_CR **NB:** При входе в Stop надо очистить все биты удержания EXTI Line, прерываний периферии и флаг RTC Alarm. Иначе, Stop не включается и выполнение продолжается.
Выход	При входе по WFI: Любая EXTI Line в режиме Interrupt (этот вектор EXTI NVIC нужно разрешать). См. Секцию 10.1.2. При входе по WFE: Любая EXTI Line в режиме события. См. Секцию 10.2.3.
Задержка	Пробуждение HSI RC + переключение регулятора напряжения

5.3.5. Дежурный режим

Самый экономный режим Standby это сочетание режима *deepsleep* с выключением регулятора напряжения. Выключаются: домен 1.8V, PLL, генераторы HSI и HSE. SRAM и регистры теряются, кроме домена резервного хранения и схематики Standby.

Вход

Как входить написано Таблице 15.

В режиме Standby отдельными битами можно включить:

- Независимый сторожевой таймер (IWDG): IWDG запускается записью в его регистр Ключа или аппаратно. Запущенный однажды он останавливается только Сбросом. См. *Секцию 19.3*.
- Real-time clock (RTC): битом `RTCEN` в регистре `RCC_BDCR`
- Внутренний RC генератор (LSI RC): битом `LSION` в регистре `RCC_CSR`.
- Внешний 32.768 kHz генератор (LSE OSC): битом `LSEON` в регистре `RCC_BDCR`.

Выход

MCU выходит из режима Standby по внешнему сбросу (ножка NRST), сбросу IWDG, переднему фронту на ножке WKUP или переднему фронту RTC alarm. Все регистры, кроме `PWR_CSR`, сбрасываются.

После выхода из Standby выполнение эквивалентно выходу из Сброса (опрос ножек загрузки, чтение вектора сброса и т.д.), но флаг `SBF` в регистре `PWR_CSR` указывает на выход из Standby.

Как выходить написано Таблице 15.

Таблица 15. Дежурный режим.

Режим Standby	Описание
Вход	WFI (Wait for Interrupt) или WFE (Wait for Event): – Установить бит SLEEPDEEP в регистре System Control – Установить бит PDDS в регистре Power Control (PWR_CR) – Очистить бит WUF в регистре PWR_CSR – Нет задержанных прерываний (для WFI) или событий (for WFI).
Выход	При входе по WFI: Любая EXTI Line в режиме Interrupt (этот вектор EXTI NVIC нужно разрешать). См. Секцию 10.1.2. При входе по WFE: Любая EXTI Line в режиме события. См. Секцию 10.2.3.
Задержка	Передний фронт на WKUP, события RTC alarm, внешний сброс на ножке NRST, IWDG Reset.

Состояние ножек I/O в режиме Standby

Все ножки находятся в третьем состоянии, кроме:

- Reset pad (доступно)
- Ножка TAMPER, если сконфигурирована для этого или выхода калибровки
- Ножка WKUP, если разрешена

Режим отладки

По умолчанию из-за остановки ядра Cortex®-M3 подключение отладки в режимах Stop и Standby теряется.

Но в установкой битов конфигурации в регистре DBGMCU_CR отлаживать можно и в экономных режимах. Детали в *Секции 31.16.1*.

5.3.6. Автопробуждение (AWU) из экономного режима

Для пробуждения MCU из экономного режима без внешнего сигнала можно использовать сигнал от RTC (Автопробуждение). Битами RTCSEL[1:0] в регистре RCC_BDCR можно выбрать два из трёх источников RTC:

- Внешний 32.768 kHz кварцевый генератор (LSE OSC). Это самый точный и экономный режим (менее 1µA добавки)
- Внутренний RC генератор (LSI RC). Экономия на стоимости кварцевого резонатора.

Для выхода из режима Stop по событию RTC alarm нужно:

- Сделать EXTI Line 17 чувствительной к переднему фронту
- Заставить RTC генерировать RTC alarm

Для выхода из режима Standby конфигурировать EXTI Line 17 не нужно.

5.4. Регистры управления питанием

Они доступны как полуслова и слова.

5.4.1. Регистр управления (PWR_CR)

Смещение адреса: 0x00

По сбросу: 0x0000 0000 (сброс выходом из Standby)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
							rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw

— Биты 31:9 Резерв, не трогать.

— Бит 8 **DBP**: Отключить защиту записи резервного домена.

В состоянии сброса RTC и резервные регистры защищены от паразитной записи, а потом её нужно разрешать.

0: Доступа нет

1: Запись разрешена.

NB: Если такты RTC это HSE/128, то этот бит должен оставаться в 1.

— Биты 7:5 **PLS[2:0]**: Выбор уровня детектора питания.

000: 2.2V
001: 2.3V
010: 2.4V
011: 2.5V
100: 2.6V
101: 2.7V
110: 2.8V
111: 2.9V

NB: См. описание устройства.

— Бит 4 **PVDE**: Включение детектора питания, изменяется программно.
0: PVD выключен
1: PVD включён

— Бит 3 **CSBF**: Очистка флага standby, читается как 0.
0: Не влияет
1: Чистит SBF Standby Flag.

— Бит 2 **CWUF:** Очистка флага побудки, читается как 0.
0: Не влияет
1: Чистит WUF Wakeup Flag **после 2 System clock циклов**.

— Бит 1 **PDDS**: Режим *deepsleep*.
Изменяется программно, работает вместе с битом LPDS.
0: Включает режим Stop при входе CPU в Deepsleep. Регулятор зависит от бита LPDS.
1: Включает режим Standby при входе CPU в Deepsleep.

— Бит 0 **LPDS**: Экономный *deepsleep*.
Изменяется программно, работает вместе с битом PDDS.
0: В режиме Stop регулятор напряжения включён
1: В режиме Stop регулятор напряжения экономит электричество.

5.4.2. Регистр состояния и управления (PWR_CSR)

Смещение адреса: 0x04
По сбросу: 0x0000 0000 (выходом из Standby не сбрасывается)
Для чтения нужны дополнительные циклы APB.

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							EWUP	Reserved					PVDO	SBF	WUF
							rw						r	r	r

— Биты 31:9 Резерв, не трогать.

— Бит 8 **EWUP**: Разрешение ножки WKUP.
Изменяется программно.
0: Ножка WKUP используется как I/O и из Standby не будит
1: Ножка WKUP выводит из Standby по переднему фронту сигнала.
NB: Очищается системным Сбросом.

— Биты 7:3 Резерв, не трогать.

— Бит 2 **PVDO:** Выход детектора питания.
Изменяется аппаратно. Достоверен только при включённом PVD (бит PVDE)
0: V_{DD}/V_{DDA} больше порога PVD в битах PLS[2:0]
1: V_{DD}/V_{DDA} меньше порога PVD в битах PLS[2:0].

— Бит 1 **SBF**: Флаг Standby.
Ставится аппаратно, снимается только POR/PDR или установкой бита CSBF в регистре PWR_CR.
0: Standby не было.
1: Standby был.

— Бит 0 **WUF**: Флаг побудки.
Изменяется аппаратно, системным сбросом или битом CWUF в регистре PWR_CR.
0: Побудки не было
1: Разбудили ножкой WKUP или сигналом RTC alarm.
NB: Если ножку WKUP разрешают битом EWUP, а уровень WKUP уже высокий, то появляется дополнительное событие побудки.

5.4.3. Карта регистров PWR

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PWR_CR	Reserved																							DBP	PLS [2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	Reset value																								0	0	0	0	0	0	0	0	0
0x004	PWR_CSR	Reserved																							EWUP	Reserved					PVDO	SBF	WUF
	Reset value																								0						0	0	0

6. Регистры резервного хранения BKP

6.1. Введение в BKP

Регистры резервного хранения (backup registers) это сорок два 16-бит регистра для 84 байт данных пользователя.

Они реализованы в отдельном домене, который питается от V_{BAT} при выключенном V_{DD}. Они не сбрасываются ни выходом из Standby ни сбросами системы и питания.

Кроме того, управляющие регистры BKP управляющие регистры используются для обнаружения Вмешательства (Tamper) и калибровки RTC.

После сброса доступ к регистрам BKP и RTC запрещён, защищая от паразитной записи. Разрешается доступ так:

- дать питание и тактирование интерфейсу битами PWREN и BKPEN в регистре RCC_APB1ENR
- разрешить доступ к BKP и RTC битом DBP в регистре PWR_CR.

6.2. Основные свойства BKP

- 20-байтовые регистры данных (в устройствах низкой и средней плотности) или 84-байтовые регистры данных (в сетевых устройствах, высокой и XL-плотности)
- Регистр статуса/управления для определения вмешательства с возможностью прерывания
- Регистр калибровки RTC
- Возможность вывода RTC Calibration Clock, импульса RTC Alarm или импульса Секунд на ножку TAMPER - PC13 (если не назначена для детектора вмешательства)

6.3. Описание работы BKP

6.3.1. Обнаружение вмешательства (Tamper)

Ножка TAMPER генерирует событие обнаружения вмешательства при изменении состояния с 0 на 1 или с 1 на 0 в зависимости от бита TPAL в регистре BKP_CR). Это событие сбрасывает все регистры данных BKP.

Однако, чтобы не терять событие Вмешательства, сигнал обнаружения логически умножен (AND) с разрешением Tamper для контроля события, произошедшего до установки разрешения.

- При TPAL =0: Если ножка TAMPER имеет высокий уровень до разрешения битом TPE, то после разрешения возникает новое событие даже без переднего фронта на ножке TAMPER
- При TPAL =1: Если ножка TAMPER имеет низкий уровень до разрешения битом TPE, то после разрешения возникает новое событие даже без заднего фронта на ножке TAMPER

Бит TPIE в регистре BKP_CSR разрешает прерывание по событию Вмешательства.

После обнаружения и очистки события Вмешательства ножку TAMPER надо выключить и снова включить битом TPE перед записью в регистры BKP_DRx. Это предохраняет от записи в регистры BKP_DRx при наличии сигнала на ножке TAMPER. Это эквивалентно обнаружению уровня сигнала.

NB: Обнаружение Вмешательства активно при выключенном V_{DD}. Дабы случайно не стереть регистры резервного хранения ножку TAMPER надо снаружи подключать к правильному уровню.

6.3.2. Калибровка RTC

Битом CCO регистра BKP_RTCCR на ножку TAMPER можно вывести такты RTC, делённые на 64.

Битами CAL[6:0] такты можно замедлить до 121 импульса/мин.

Подробности калибровки RTC есть в AN2604 "*STM32F101xx and STM32F103xx RTC calibration*".

6.4. Регистры BKP

Регистры периферии доступны полусловами и словами. Все регистры словные, но активно только младшее полуслово.

6.4.1. Регистры данных (BKP_DRx) (x = 1 ..42)

Смещение адреса: 0x04 до 0x28, 0x40 до 0xBC

По сбросу: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

NB: Регистры BKP_DRx сбросами Системы и Питания или выходом из режима Standby не сбрасываются. А обнуляются они сбросом резервного домена или событием на ножке TAMPER (если активна).

6.4.2. Регистр калибровки RTC (BKP_RTCCR)

Смещение адреса: 0x2C

По сбросу: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						ASOS	ASOE	CCO	CAL[6:0]						
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:10 Резерв, не трогать.

— Бит 9 **ASOS**: Выбор вывода Alarm или секунд.

Если стоит бит ASOE, то в зависимости от бита ASOS на ножку TAMPER выводятся:

0: Импульсы RTC Alarm

1: Импульсы RTC Second.

NB: Очищается только сбросом всего домена.

— Бит 8 **ASOE:** Разрешение вывода импульсов Alarm или секунд.

Длительность импульса равна одному такту RTC. При установленном бите ASOE ножку TAMPER разрешать нельзя.

NB: Очищается только сбросом всего домена.

— Бит 7 **CCO**: Вывод тактов калибровки.

0: Не влияет.

1: На ножку TAMPER выводятся такты RTC, делённые на 64. При установленном бите CCO ножку TAMPER разрешать нельзя. Тогда получите нежелательное Вмешательство.

NB: При выключенном V_{DD} бит сбрасывается.

— Биты 6:0 **CAL**: Значение калибровки.

Это число пропускаемых импульсов на каждые 2^20 тактов.

Этим RTC замедляется шагами по 1000000/2^20 тактов/мин. от 0 до тактов/мин.

6.4.3. Регистр управления BKP (BKP_CR)

Смещение адреса: 0x30

По сбросу: 0x0000 0000

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

— Биты 15:10 Резерв, не трогать.

— Бит 1 **TPAL**: Уровень активности ножки TAMPER.
0: Регистры BKP сбрасываются высоким уровнем (если стоит бит TPE).
1: Регистры BKP сбрасываются низким уровнем (если стоит бит TPE).
— Бит 0 **TPE:** Разрешение ножки TAMPER.
0: Ножка TAMPER свободна для I/O
1: Ножка TAMPER активна для определения Вмешательства.

NB: Одновременная установка битов TPAL и TPE безопасна, но одновременное их стирание может вызвать ложное событие Вмешательства. Так что стирать нужно сначала TPE, затем TPAL.

6.4.4. Регистр управления/состояния BKP (BKP_CSR)

Смещение адреса: 0x34

По сбросу: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TIF	TEF	Reserved					TPIE	CTI	CTE
						r	r						rw	w	w

— Биты 15:10 Резерв, не трогать.
— Бит 9 **TIF**: Флаг прерывания TAMPER.
Ставится аппаратно при появлении события и стоящем бите TPIE. Снимается записью 1 в бит CTI (чистит и прерывание). Снимается и при стирании бита TPIE.
0: Прерывания не было.
1: Прерывание было.
NB: Снимается только системным сбросом и выходом из Standby.
— Бит 8 **TEF:** Флаг события TAMPER.
Ставится аппаратно при появлении события. Снимается записью 1 в бит CTE.
0: События TAMPER не было
1: Ножка TAMPER активна для определения Вмешательства.
— Биты 7:3 Резерв, не трогать.
— Бит 2 **TPIE:** Разрешение прерываний ножки TAMPER.
0: Прерывание TAMPER запрещено
1: Прерывание TAMPER разрешено (также нужно установить бит TPE в регистре BKP_CR).
NB: Прерывание Tamper не выводит из экономных режимов. Бит снимается только системным сбросом и выходом из Standby.
— Бит 1 **CTI:** Очистка прерывания TAMPER. (Читается как 0).
0: Не влияет
1: Чистит прерывание и флаг TIF.
— Бит 0 **CTE:** Очистка события ножки TAMPER.
0: Не влияет
1: Сбрасывает событие и детектор TAMPER

7. Сброс и тактирование устройств разной плотности (RCC)

7.1. Сброс

Есть три типа сброса: системный сброс, сброс питания и сброс домена резервного хранения.

7.1.1. Системный сброс

Сбрасывает все регистры в предписанное состояние кроме флагов сброса в регистре CSR и регистров домена резервного хранения.

Системный сброс генерируется в следующих случаях:

1. Низкий уровень на ножке NRST (внешний сброс)
2. Конец счёта Оконного сторожевого таймера (сброс WWDG)
3. Конец счёта Независимого сторожевого таймера (сброс IWDG reset)
4. Программный сброс (SW сброс)
5. Сброс Управления питанием

Источники идентифицируются флагами в регистре RCC_CSR (см. *Секцию 7.3.10*).

Программный сброс

Для запуска программного сброса надо поставить бит SYSRESETREQ в регистре Cortex®-M3 Application Interrupt and Reset Control Register. См. *STM32F10xxx Cortex®-M3 programming manual*.

Сброс управления питанием

Есть два вида сброса:

1. Сброс по входу в режим Standby:

   Он разрешается очисткой бита nRST_STDBY в байтах Опций Пользователя. В этом случае при успешном выполнении последовательности входа в режим Standby устройство сбрасывается вместо входа в режим Standby.

2. Сброс по входу в режим Stop:

   Он разрешается очисткой бита nRST_STOP в байтах Опций Пользователя. В этом случае при успешном выполнении последовательности входа в режим Stop устройство сбрасывается вместо входа в режим Stop.

Опции Пользователя приведены в *STM32F10xxx Flash programming manual*.

7.1.2. Сброс питания

Генерируется в следующих случаях:

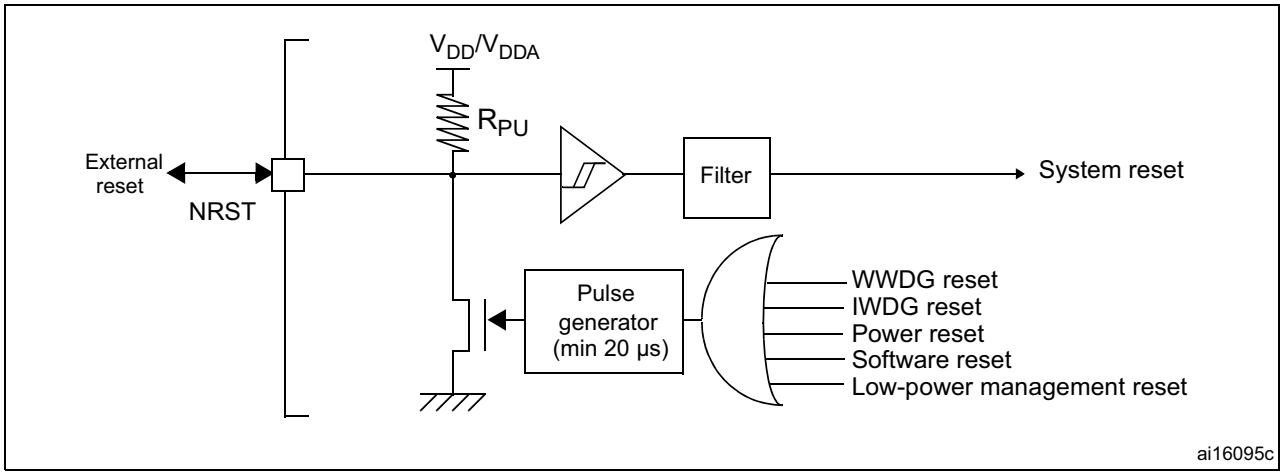
1. Вкл./Выкл. питания (сброс POR/PDR)
2. Выход из режима Standby

Устанавливает все регистры в предписанное состояние, кроме домена резервного хранения.

Эти источники работают как ножка NRST, которая удерживается в низком состоянии на время фазы задержки. Вектор программы сброса расположен по адресу 0x0000_0004 в карте памяти.

Сигнал системного сброса выдаётся на ножку NRST. Генератор импульса гарантирует минимальную длительность импульса 20 µs для обоих источников (внешнего или внутреннего). В случае внешнего сброса импульс генерируется на всё время удержания ножки NRST низкой.

Упрощённая схема сброса.



7.1.3. Сброс резервного домена

Здесь есть два типа сброса:

1. Программный сброс установкой бита BDRST в регистре RCC_BDCR.
2. Включение одного из обоих выключенных V_{DD} или V_{BAT}.

7.2. Тактирование

Для системных тактов (SYSCLK) можно использовать три источника:

- такты генератора HSI
- такты генератора HSE
- такты PLL

Устройства имеют два вторичных источника тактов:

- 40 kHz низкоскоростной внутренний RC (LSI RC), для независимого сторожевого таймера и необязательно RTC для Автопробуждения из режима Stop/Standby.
- 32.768 kHz низкоскоростной внешний кварцевый генератор (LSE crystal) для возможного тактирования часов реального времени (RTCCLK).

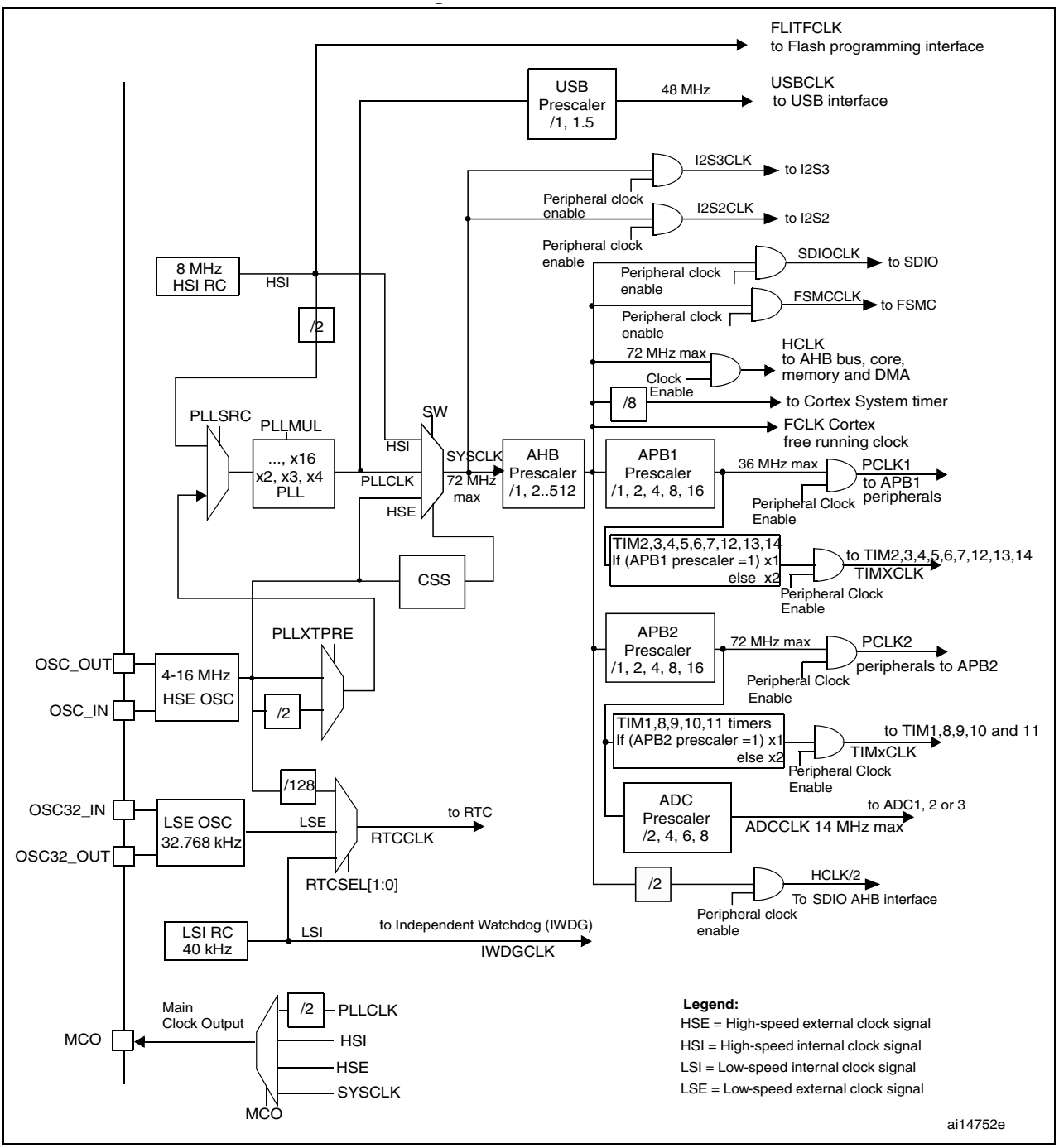
Все источники можно включать и выключать независимо во имя экономии электронов.

Несколько предделителей допускают конфигурацию частот доменов AHB, высокоскоростного APB (APB2) и низкоскоростного APB (APB1). Максимальная частота доменов AHB и APB2 равна 72 MHz. Максимальная допустимая частота домена APB1 равна 36 MHz. Интерфейс SDIO AHB тактируется фиксированной частотой равной HCLK/2.

RCC кормит системный таймер (SysTick) внешними тактами AHB HCLK/8. SysTick может работать от них или от HCLK, что выбирается в регистре управления и состояния SysTick. ADC тактируются частотой высокоскоростного домена (APB2) делённой by 2, 4, 6 или 8.

Такты программирования флэш памяти (FLITFCLK) это всегда HSI.

Дерево тактирования



1. При питании PLL от HSI максимальная частота может достигать 64 MHz.
2. Электрические характеристики внутренного и внешнего источников описаны в руководствах устройств.

Тактирование таймеров автоматически фиксируется аппаратурой. Есть два варианта:

1. если предделитель APB равен 1, то таймеры питаются от частоты своего домена APB.
2. иначе, они кормятся удвоенной (×2) частотой своего домена APB.

FCLK работает как *Cortex®-M3's free-running clock*. За деталями лезьте в Arm® *Cortex-M3 r1p1 Technical Reference Manual (TRM)*.

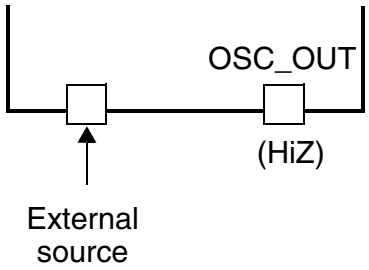
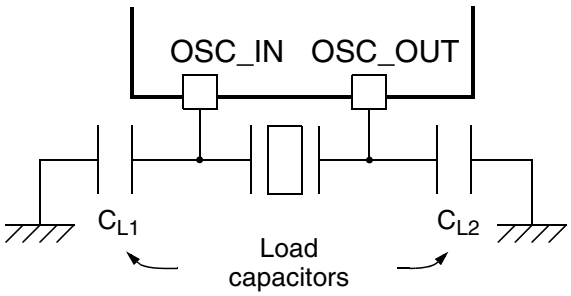
7.2.1. Такты HSE

Высокоскоростной внешний тактовый сигнал (HSE) можно получить из двух источников:

- внешний кварцевый/керамический резонатор HSE
- внешние такты HSE пользователя

Резонатор и конденсаторы нагрузки надо размещать как можно ближе к ножкам генератора для снижения помех и стабилизации времени запуска. Ёмкости нагрузки должны соответствовать выбранному генератору.

Источники HSE/LSE

Clock source	Hardware configuration
External clock	
Crystal/Ceramic resonators	

Внешний источник (HSE bypass)

В этом режиме внешний источник должен иметь частоту до 25 MHz. Он включается битами `HSEBYP` и `HSEON` в регистре `RCC_CR`. Внешний сигнал (меандр, синус или треугольник) с ~50% заполнением подаётся на ножку OSC_IN, ножка OSC_OUT должна оставаться в третьем (hi-Z) состоянии.

Внешний кварцевый/керамический резонатор (HSE crystal)

Выгода 4 - 16 MHz внешнего генератора в очень точных главных тактах.

Стабильность внешнего генератора показывается флагом `HSERDY` в регистре `RCC_CR`. При включении такты не выпускаются на волю пока аппаратура не поставит этот бит. При этом может быть вызвано прерывание, если разрешено в регистре `RCC_CIR`.

HSE Crystal включается/выключается битом `HSEON` в регистре `RCC_CR`.

7.2.2. Такты HSI

Тактовый сигнал HSI выдаётся внутренним 8 MHz RC генератором как системные такты или, поделив на 2, входом для PLL (ФАПЧ).

Выгода HSI RC генератора в низкой стоимости. Да и запускается быстрее, чем HSE crystal генератор, но даже после калибровки точность всё равно ниже кварцевого или керамического.

Калибровка

Частоты RC генератора каждого чипа калибруются фирмой ST до 1% точности при TA=25°C.

После сброса, заводская калибровка загружается в биты `HSICAL[7:0]` регистра `RCC_CR`.

Если ваша система работает при колебаниях напряжения и температуры, то частота RC генератора может меняться. Подстроить частоту HSI можно битами `HSITRIM[4:0]` в регистре `RCC_CR`.

Флаг `HSIRDY` в регистре `RCC_CR` показывает стабильность HSI RC. При включении такты HSI RC не выпускаются на волю пока аппаратура не поставит этот бит.

HSI RC включается/выключается битом `HSION` в регистре `RCC_CR`.

Сигнал HSI можно использовать как резервный (Auxiliary clock) при сбое HSE. См. *Секцию 7.2.7: Система безопасности тактирования (CSS)*.

7.2.3. PLL (ФАПЧ)

Внутренний PLL используется для умножения частоты HSI RC или HSE crystal.

Конфигурацию PLL (выбор HSI/2 или HSE на входе PLL и множитель) нужно делать до включения PLL. После включения PLL эти параметры изменить нельзя.

При готовности PLL можно вызвать прерывание, если оно разрешено в регистре `RCC_CIR`.

Если в системе используется USB, то PLL должно программировать на частоту 48 или 72 MHz для выдачи 48 MHz USBCLK.

7.2.4. Такты LSE

LSE crystal это 32.768 kHz низкоскоростной внешний кварцевый или керамический резонатор. Его выгода в высокой точности сигнала для RTC и другой периферии.

Включается/выключается LSE битом `LSEON` в регистре резервного домена `RCC_BDCR`.

Флаг `LSERDY` в регистре `RCC_BDCR` показывает стабильность LSE crystal. При включении такты LSE crystal не выпускаются на волю пока аппаратура не поставит этот бит. При этом может быть вызвано прерывание, если разрешено в регистре `RCC_CIR`.

Внешний источник (LSE bypass)

Внешний источник частотой до 1 MHz выбирается установкой битов `LSEBYP` и `LSEON` в регистре `RCC_BDCR`. Внешний сигнал (меандр, синус или треугольник) с ~50% заполнением подаётся на ножку OSC32_IN, ножка OSC32_OUT должна оставаться в третьем (hi-Z) состоянии.

7.2.5. Такты LSI

Генератор LSI RC частотой около 40 kHz (между 30 kHz и 60 kHz) продолжает работать в режимах Stop и Standby ради независимого сторожевого таймера (IWDG) и блока авто-побудки (AWU).

Включается/выключается LSI RC битом `LSION` в регистре `RCC_CSR`.

Флаг `LSIRDY` в регистре `RCC_CSR` показывает стабильность LSE crystal. При включении такты LSE crystal не выпускаются на волю пока аппаратура не поставит этот бит. При этом может быть вызвано прерывание, если разрешено в регистре `RCC_CIR`.

NB: Калибровка LSI возможна только в устройствах высокой, XL-плотности и сетевых.

Калибровка LSI

Она позволяет получить сигнал приемлемой точности для RTC и IWDG.

Калибровка выполняется измерением частоты LSI по отношению у входным тактам TIM5 (TIM5CLK) с точностью генератора HSE. Теперь можно программно подстроить 20-бит предделитель RTC или задержку IWDG.

Процедура калибровки LSI:

1. Включить таймер TIM5 и его канал 4 на захват входа
2. Установить бит `TIM5CH4_IREMAP` в регистре `AFIO_MAPR` для подачи тактов LSI на TIM5 - 4.

3. Измерить частоту тактов LSI через событие или прерывание TIM5 Capture/compare 4.
4. С помощью измерения обновить 20-бит предделитель RTC и/или вычислить задержку IWDG.

7.2.6. Выбор SysClk

После системного сброса такты системы берутся от HSI. Источник системных тактов, подключенный напрямую, или через PLL остановить нельзя.

Переключение на новый источник тактов возможно только если он готов (стабилен после запуска или PLL зафиксирован). Если выбран неготовый источник, то он подключится только после готовности. Биты состояния в регистре `RCC_CR` показывают готовые генераторы и кто выбран системным.

7.2.7. Система безопасности тактирования (CSS)

CSS активируется программно и включается после задержки запуска HSE, выключается после остановки генератора.

Если в тактах HSE обнаруживается сбой, то он автоматически останавливается, таймерам (TIM1 и TIM8) посылается событие отключения входа и генерируется прерывание (Clock Security System Interrupt CSSI), подключённое к вектору исключения NMI.

NB: Если при включённой CSS возникает сбой HSE, прерывание NMI происходит независимо от того, очищен ли бит удержания CSS. Следовательно в NMI ISR нужно очистить прерывание CSS установкой бита `CSSC` в регистре `RCC_CIR`.

Если генератор HSE служит источником системных тактов напрямую или через PLL, то обнаруженный сбой переключает тактирование на HSI и выключает HSE и PLL (в случае его использования).

7.2.8. Такты RTC

Источником тактов RTCCLK могут быть HSE/128, LSE или LSI. Он выбирается битами `RTCSEL[1:0]` в регистре `RCC_BDCR`. Этот выбор нельзя изменить без сброса домена резервного хранения.

Генератор LSE расположен в резервном домене тактируется, а HSE и LSI нет. Следовательно:

- Если RTC тактируется от LSE:
  - RTC продолжает работать при отключении V_{DD}, питаясь от V_{BAT}.
- Если блок авто-побудки (AWU) тактируется от LSI:
  - Состояние AWU при отключении V_{DD} не гарантируется. См. *Секцию 7.2.5*.
- Если RTC тактируется от HSE/128:
  - При отключении V_{DD} или регулятора напряжения (выключения 1.8 V домена) состояние RTC не гарантируется.
  - Нужно установить бит `DPB` (отключение защиты записи) в регистре `PWR_CR` (*Секция 5.4.1*).

7.2.9. Такты сторожевого таймера

При аппаратном или программном запуске независимого сторожевого таймера (IWDG) генератор LSI включается и не выключается. Тактирование на IWDG пропускается после задержки запуска LSI.

7.2.10. Вывод тактового сигнала

Существует возможность выводить на внешнюю ножку тактовый сигнал микроконтроллера (MCO). Регистры конфигурации соответствующего порты GPIO должны быть переключены в альтернативный режим. Можно выводить один из 4 сигналов.

- SYSCLK
- HSI
- HSE
- PLL/2

Он выбирается битами `MCO[2:0]` в регистре `RCC_CFGR`.

7.3. Регистры RCC

7.3.1. Регистр управления (RCC_CR)

Смещение адреса: 0x00

По сбросу: 0x0000 XX83 где X не определено.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						PLL RDY	PLLON	Reserved				CSS ON	HSE BYP	HSE RDY	HSE ON
						r	rw					rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res.	HSI RDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

— Биты 31:26 Резерв, не трогать.

— Бит 25 **PLLRDY**: Флаг фиксации PLL.

Ставится аппаратно.

0: PLL unlocked

1: PLL locked

— Бит 24 **PLLON:** Включение PLL.

Ставится и снимается программно.

0: PLL OFF

1: PLL ON

— Биты 23:20 Резерв, не трогать.

— Бит 19 **CSSON:** Включение системы безопасности.

Управляется программно. При стоящем CSSON детектор тактов аппаратно включается при готовом HSE и выключается при отказе HSE

0: Детектор OFF

1: Детектор ON (ON если HSE готов , OFF если нет).

— Бит 18 **HSEBYP:** Шунт (обход) **HSE**.

Управляется программно. Бит HSEON должен стоять. Бит HSEBYP можно писать только при выключенном HSE.

0: Внешний генератор 4-16 MHz не обойдён

1: Внешний генератор 4-16 MHz обойдён

— Бит 17 **HSERDY:** Флаг готовности HSE.

Ставится аппаратно при стабильном HSE. Снимается через 6 тактов HSE после снятия HSEON.

0: HSE не готов

1: HSE готов

— Бит 16 **HSEON:** Включение HSE.

Управляется программно. Снимается аппаратно при входе в режим Stop или Standby. Если HSE прямо или непрямо используется как источник системных тиков, то снять его невозможно.

0: HSE генератор OFF

1: HSE генератор ON

— Биты 15:8 **HSICAL[7:0]:** Калибровка HSI.

Заводские установки инициализируются при запуске.

— Биты 7:3 **HSITRIM[4:0]:** Подстройка HSI.

Программная подстройка частоты внутреннего HSI RC при колебаниях напряжения и температуры.

Значение по умолчанию 16, при добавлении к значению HSICAL подстраивает HSI на 8 MHz ± 1%. Шаг подстройки ($F_{hsitrim}$) около 40 kHz между двумя последовательными шагами HSICAL.

— Бит 2 Резерв, не трогать.

— Бит 1 **HSIRDY:** Флаг готовности HSI.

Ставится аппаратно при стабильности внутреннего 8 MHz RC генератора. Снимается через 6 тактов HSI после снятия HSI.

0: HSI не готов

1: HSI готов

— Бит 0 **HSION:** Включение HSI.

Управляется программно. Ставится аппаратно при выходе из режимов Stop или Standby и в случае отказа внешнего 4-16 MHz генератора, используемого для системных тактов. Этот бит невозможно снять при прямом или непрямом использовании для системных тактов или выборе стать таковым.

0: HSI OFF

1: HSI ON

7.3.2. Регистр конфигурации (RCC_CFR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000.

Доступ: 0 ≤ такты ожидания ≤ 2, слово, полуслово или байт. 1 или 2 тактов ожидания появляются только при переключении источника тактов.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					MCO[2:0]			Res.	USB PRE	PLLMUL[3:0]				PLL XTPRE	PLL SRC
					rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPRE[1:0]		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

— Биты 31:27 Резерв, не трогать.

— Биты 26:24 **MCO**: Вывод тактов микроконтроллера.

0xx: Нету

100: System clock (SYSCLK)

101: HSI

110: HSE

111: PLL / 2

NB: Здесь могут пропущены несколько тактов при запуске или переключении источника MCO.

При подаче MCO тактов SYSCLK убедитесь, что частота не превышает 50 MHz (максимум для IO).

— Бит 23 Резерв, не трогать.

— Бит 22 **USBPRE:** Предделитель USB.

Управляется программно для выдачи 48 MHz тактов USB. Должен быть правильно установлен до разрешения тактов USB в регистре RCC_APB1ENR. При включённом USB не изменяется.

0: PLL делённый на 1.5

1: PLL как есть .

— Биты 21:18 **PLLMUL:** Множитель PLL.

Пишется программно только при выключенном PLL.

Внимание: Выходная частота PLL не должна превышать 72 MHz.

0000: PLL x 2

0001: PLL x 3

0010: PLL x 4

0011: PLL x 5

0100: PLL x 6

0101: PLL x 7

0110: PLL x 8

0111: PLL x 9

1000: PLL x 10

1001: PLL x 11

1010: PLL x 12

1011: PLL x 13

1100: PLL x 14

1101: PLL x 15

1110: PLL x 16

1111: PLL x 16

— Бит 17 **PLLXTPRE:** Предделитель частоты HSE на входе PLL.
Управляется программно. Изменяется только при выключенном PLL.
0: HSE не делится
1: HSE делится 2

— Бит 16 **PLLSRC**: Источник тактов PLL.
Управляется программно. Изменяется только при выключенном PLL.
0: HSI / 2
1: HSE

— Биты 15:14 **ADCPRE**: Предделитель ADC.
Пишется программно.
00: PCLK2 / 2
01: PCLK2 / 4
10: PCLK2 / 6
11: PCLK2 / 8

— Биты 13:11 **PPRE2**: Предделитель APB2 для получения PCLK2.
Пишется программно.
0xx: HCLK как есть
100: HCLK / 2
101: HCLK / 4
110: HCLK / 8
111: HCLK / 16

— Биты 10:8 **PPRE1**: Предделитель APB1 для получения PCLK1.
Пишется программно.
Внимание: Частота PCLK1 не должна превышать 36 MHz.
0xx: HCLK как есть
100: HCLK / 2
101: HCLK / 4
110: HCLK / 8
111: HCLK / 16

— Биты 7:4 **HPRE**: Предделитель AHB.
Пишется программно.
0xxx: SYSCLK как есть
1000: SYSCLK / 2
1001: SYSCLK / 4
1010: SYSCLK / 8
1011: SYSCLK / 16
1100: SYSCLK / 64
1101: SYSCLK / 128
1110: SYSCLK / 256
1111: SYSCLK / 512
NB: Если предделитель AHB не равен 1, то буфер предвыборки нужно включать.

— Биты 3:2 **SWS**: Состояние переключателя системных тактов.
Ставится аппаратно.
00: HSI
01: HSE
10: PLL
11: не применимо

— Биты 1:0 **SW**: Переключатель системных тактов.
Программно ставится для выбора источника SYSCLK.
Аппаратно переключается на HSI при выходе из режима Stop и Standby или при отказе генератора HSE при включённой системе контроля тактирования CSS.
00: HSI
01: HSE
10: PLL
11: не дозволено.

7.3.3. Регистр прерываний (RCC_CIR)

Смещение адреса: 0x08

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CSSC	Reserved		PLL RDYC	HSE RDYC	HSI RDYC	LSE RDYC	LSI RDYC
								w			w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			PLL RDYIE	HSE RDYIE	HSI RDYIE	LSE RDYIE	LSI RDYIE	CSSF	Reserved		PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF
			rw	rw	rw	rw	rw	r			r	r	r	r	r

— Биты 31:24 Резерв, не трогать.

— Бит 23 **CSSC**: Очистка флага прерывания CSS.
Пишется программно. 0: ничего,1: чистит

— Биты 22:21 Резерв, не трогать.

— Бит 20 **PLLRDYC:** Очистка флага прерывания готовности PLL.
Пишется программно. 0: ничего 1: чистит

— Бит 19 **HSERDYC:** Очистка флага прерывания готовности HSE.
Пишется программно. 0: ничего 1: чистит

— Бит 18 **HSIRDYC:** Очистка флага прерывания готовности HSI.
Пишется программно. 0: ничего 1: чистит

— Бит 17 **LSERDYC:** Очистка флага прерывания готовности LSE.
Пишется программно. 0: ничего 1: чистит

— Бит 16 **LSIRDYC:** Очистка флага прерывания готовности LSI.
Пишется программно. 0: ничего 1: чистит

— Биты 15:13 Резерв, не трогать.

— Бит 12 **PLLRDYIE:** Разрешение прерывания готовности PLL.
Ставится и снимается программно. 0: Выключено 1: Включено

— Бит 11 **HSERDYIE:** Разрешение прерывания готовности HSE.
Ставится и снимается программно. 0: Выключено 1: Включено

— Бит 10 **HSIRDYIE:** Разрешение прерывания готовности HSI.
Ставится и снимается программно. 0: Выключено 1: Включено

— Бит 9 **LSERDYIE:** Разрешение прерывания готовности LSE.
Ставится и снимается программно. 0: Выключено 1: Включено

— Бит 8 **LSIRDYIE:** Разрешение прерывания готовности LSI.
Ставится и снимается программно. 0: Выключено 1: Включено

— Бит 7 **CSSF:** Флаг CSS.
Ставится аппаратно при отказе внешнего генератора 4 - 16 MHz. Снимается записью бита CSSC.
0: Отказа нет 1: Отказ есть

— Биты 6:5 Резерв, не трогать.

— Бит 4 **PLLRDYF:** Флаг прерывания готовности PLL.
Ставится аппаратно при фиксации PLL и стоящем PLLRDYDIE. Чистится записью бита PLLRDYC.
0: Прерывания нет 1: Прерывание есть

— Бит 3 **HSERDYF:** Флаг прерывания готовности HSE.
Ставится аппаратно при готовности HSE и стоящем HSERDYDIE. Чистится записью бита HSERDYC.
0: Прерывания нет 1: Прерывание есть

— Бит 2 **HSIRDYF:** Флаг прерывания готовности HSI.
Ставится аппаратно при готовности HSI и стоящем HSIRDYDIE. Чистится записью бита HSIRDYC.
0: Прерывания нет 1: Прерывание есть

— Бит 1 **LSERDYF:** Флаг прерывания готовности LSE.
Ставится аппаратно при готовности LSE и стоящем LSERDYDIE. Чистится записью бита LSERDYC.
0: Прерывания нет 1: Прерывание есть

— Бит 0 **LSIRDYF:** Флаг прерывания готовности LSI.
Ставится аппаратно при готовности LSI и стоящем LSIRDYDIE. Чистится записью бита LSIRDYC.
0: Прерывания нет 1: Прерывание есть

7.3.4. Регистр сброса периферии APB2 (RCC_APB2RSTR)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт. Сброс записью "1" в бит.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										TIM11 RST	TIM10 RST	TIM9 RST	Reserved		
										rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 RST	USART1 RST	TIM8 RST	SPI1 RST	TIM1 RST	ADC2 RST	ADC1 RST	IOPG RST	IOPF RST	IOPE RST	IOPD RST	IOPC RST	IOPB RST	IOPA RST	Res.	AFIO RST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	Res.	rw

— Биты 31:22 Резерв, не трогать.
— Бит 21 **TIM11RST**: Сброс таймера TIM11.
— Бит 20 **TIM10RST**: Сброс таймера TIM10.
— Бит 19 **TIM9RST**: Сброс таймера TIM9.
— Биты 18:16 Резерв, не трогать.
— Бит 15 **ADC3RST:** Сброс ADC3.
— Бит 14 **USART1RST:** Сброс USART1.
— Бит 13 **TIM8RST:** Сброс таймера TIM8.
— Бит 12 **SPI1RST:** Сброс SPI1.
— Бит 11 **TIM1RST:** Сброс таймера TIM1.
— Бит 10 **ADC2RST:** Сброс ADC2.
— Бит 9 **ADC1RST:** Сброс ADC1.
— Бит 8 **IOPGRST:** Сброс IO порта G.
— Бит 7 **IOPFRST:** Сброс IO порта F.
— Бит 6 **IOPERST:** Сброс IO порта E.
— Бит 5 **IOPDRST:** Сброс IO порта D.
— Бит 4 **IOPCRST:** Сброс IO порта C.
— Бит 3 **IOPBRST:** Сброс IO порта B.
— Бит 2 **IOPARST:** Сброс IO порта A.
— Бит 1 Резерв, не трогать.
— Бит 0 **AFIORST:** Сброс альтернативных функций IO.

7.3.5. Регистр сброса периферии APB1 (RCC_APB1RSTR)

Смещение адреса: 0x10

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт. Сброс записью "1" в бит.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DAC RST	PWR RST	BKP RST	Res.	CAN RST	Res.	USB RST	I2C2 RST	I2C1 RST	UART5 RST	UART4 RST	USART 3 RST	USART 2 RST	Res.
		rw	rw	rw		rw		rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 RST	SPI2 RST	Reserved		WWDG RST	Reserved		TIM14 RST	TIM13 RST	TIM12 RST	TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать.
— Бит 29 **DACRST**: Сброс DAC.
— Бит 28 **PWRRST**: Сброс PWR.

— Бит 27 **BKPRST**: Сброс PWR.
— Бит 26 Резерв, не трогать.
— Бит 25 **CANRST:** Сброс CAN.
— Бит 24 Резерв, не трогать.
— Бит 23 **USBRST:** Сброс USB.
— Бит 22 **I2C2RST:** Сброс I2C2.
— Бит 21 **I2C1RST:** Сброс I2C1.
— Бит 20 **UART5RST:** Сброс UART5.
— Бит 19 **UART4RST:** Сброс UART4.
— Бит 18 **USART3RST:** Сброс USART3.
— Бит 17 **USART2RST:** Сброс USART2.
— Бит 16 Резерв, не трогать.
— Бит 15 **SPI3RST:** Сброс SPI3.
— Бит 14 **SPI2RST:** Сброс **SPI2**.
— Биты 13:12 Резерв, не трогать.
— Бит 11 **WWDGRST:** Сброс WWDG.
— Биты 10:9 Резерв, не трогать.
— Бит 8 **TIM14RST:** Сброс TIM14.
— Бит 7 **TIM13RST:** Сброс TIM13.
— Бит 6 **TIM12RST:** Сброс TIM12.
— Бит 5 **TIM7RST:** Сброс TIM7.
— Бит 4 **TIM6RST:** Сброс TIM6.
— Бит 3 **TIM5RST:** Сброс TIM5.
— Бит 2 **TIM4RST:** Сброс TIM4.
— Бит 1 **TIM3RST:** Сброс TIM3.
— Бит 0 **TIM2RST:** Сброс TIM2.

7.3.6. Регистр разрешения тактов периферии AHB (RCC_AHBENR)

Смещение адреса: 0x14

По сбросу: 0x0000 0014.

Доступ: без ожидания, слово, полуслово и байт.

NB: При выключении тактов периферии их регистры могут не читаться и всегда возвращать 0x0.

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					SDIOEN	Res.	FSMCEN	Res.	CRCEN	Res.	FLITFEN	Res.	SRAMEN	DMA2EN	DMA1EN
					rw		rw		rw		rw		rw	rw	rw

Читается и пишется программно.
0: Выключено
1: Включено

— Биты 31:11 Резерв, не трогать.
— Бит 10 **SDIOEN**: Тактирование SDIO.
— Бит 9 Резерв, не трогать.
— Бит 8 **FSMCEN**: Тактирование FSMC.
— Бит 7 Резерв, не трогать.
— Бит 6 **CRCEN**: Тактирование CRC.
— Бит 5 Резерв, не трогать.
— Бит 4 **FLITFEN**: Тактирование FLITF в режиме Sleep.
— Бит 3 Резерв, не трогать.
— Бит 2 **SRAMEN**: Тактирование SRAM в режиме Sleep.
— Бит 1 **DMA2EN**: Тактирование DMA2.
— Бит 0 **DMA1EN**: Тактирование DMA1.

7.3.7. Регистр разрешения тактов периферии APB2 (RCC_APB2ENR)

Смещение адреса: 0x18

По сбросу: 0x0000 0000.

Доступ: слово, полуслово и байт.

Без ожиданий, но во время доступа к периферии APB2 здесь добавляются циклы ожидания до завершения такого доступа.

NB: При выключении тактов периферии их регистры могут не читаться и всегда возвращать 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										TIM11 EN	TIM10 EN	TIM9 EN	Reserved		
										rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 EN	USART 1EN	TIM8 EN	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	IOPG EN	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.	AFIO EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Читается и пишется программно.

0: Выключено

1: Включено

— Биты 31:22 Резерв, не трогать.
— Бит 21 **TIM11EN**: Тактирование TIM11.
— Бит 20 **TIM10EN**: Тактирование TIM10.
— Бит 19 **TIM9EN**: Тактирование TIM9.
— Биты 18:16 Резерв, не трогать.
— Бит 15 **ADC3EN:** Тактирование ADC3.
— Бит 14 **USART1EN:** Тактирование USART1.
— Бит 13 **TIM8EN:** Тактирование TIM8.
— Бит 12 **SPI1EN:** Тактирование SPI1.
— Бит 11 **TIM1EN:** Тактирование TIM1.
— Бит 10 **ADC2EN:** Тактирование ADC2.
— Бит 9 **ADC1EN:** Тактирование ADC1.
— Бит 8 **IOPGEN:** Тактирование IO порта G.
— Бит 7 **IOPFEN:** Тактирование IO порта F.
— Бит 6 **IOPEEN:** Тактирование IO порта E.
— Бит 5 **IOPDEN:** Тактирование IO порта D.
— Бит 4 **IOPCEN:** Тактирование IO порта C.
— Бит 3 **IOPBEN:** Тактирование IO порта B.
— Бит 2 **IOPAEN:** Тактирование IO порта A.
— Бит 1 Резерв, не трогать.
— Бит 0 **AFIOEN:** Тактирование альтернативных функций IO.

7.3.8. Регистр разрешения тактов периферии APB1 (RCC_APB1ENR)

Смещение адреса: 0x1C

По сбросу: 0x0000 0000.

Доступ: слово, полуслово и байт.

Без ожиданий, но во время доступа к периферии APB1 здесь добавляются циклы ожидания до завершения такого доступа.

NB: При выключении тактов периферии их регистры могут не читаться и всегда возвращать 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DAC EN	PWR EN	BKP EN	Res.	CAN EN	Res.	USB EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART3 EN	USART2 EN	Res.
		rw	rw	rw		rw		rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Reserved		WWD GEN	Reserved		TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Читается и пишется программно.

0: Выключено

1: Включено

— Биты 31:30 Резерв, не трогать.

— Бит 29 **DACEN**: Тактирование DAC.

— Бит 28 **PWREN**: Тактирование PWR.

— Бит 27 **BKPEN**: Тактирование BKP.

— Бит 26 Резерв, не трогать.

— Бит 25 **CANEN:** Тактирование CAN.

— Бит 24 Резерв, не трогать.

— Бит 23 **USBEN:** Тактирование USB.

— Бит 22 **I2C2EN:** Тактирование I2C2.

— Бит 21 **I2C1EN:** Тактирование I2C1.

— Бит 20 **UART5EN:** Тактирование UART5.

— Бит 19 **UART4EN:** Тактирование UART4.

— Бит 18 **USART3EN:** Тактирование USART3.

— Бит 17 **USART2EN:** Тактирование USART2.

— Бит 16 Резерв, не трогать.

— Бит 15 **SPI3EN:** Тактирование SPI3.

— Бит 14 **SPI2EN:** Тактирование SPI2.

— Биты 13:12 Резерв, не трогать.

— Бит 11 **WWDGEN:** Тактирование WWDG.

— Биты 10:9 Резерв, не трогать.

— Бит 8 **TIM14EN:** Тактирование TIM14.

— Бит 7 **TIM13EN:** Тактирование TIM13.

— Бит 6 **TIM12EN:** Тактирование TIM12.

— Бит 5 **TIM7EN:** Тактирование TIM7.

— Бит 4 **TIM6EN:** Тактирование TIM6.

— Бит 3 **TIM5EN:** Тактирование TIM5.

— Бит 2 **TIM4EN:** Тактирование TIM4.

— Бит 1 **TIM3EN:** Тактирование TIM3.

— Бит 0 **TIM2EN:** Тактирование TIM2.

7.3.9. Регистр управления резервным доменом (RCC_BDCR)

Смещение адреса: 0x20

По сбросу Резервного Домена: 0x0000 0000.

Доступ: 0 ≤ тактов ожидания ≤ 3, слово, полуслово и байт. Такты ожидания добавляются при успешном доступе к регистру.

NB: Биты LSEON, LSEBYP, RTCSEL и RTCEN регистра RCC_BDCR лежат в резервном домене. Значит после Сброса они защищены от записи и перед их изменением надо установить бит DBP в регистре PWR_CR. См. *Секцию 5*. Обнуляются они только Сбросом резервного домена и ничем иным (см. *Секцию 7.1.3*).

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC EN	Reserved					RTCSEL[1:0]		Reserved					LSE BYP	LSE RDY	LSEON
rw						rw	rw						rw	r	rw

— Биты 31:17 Резерв, не трогать.

— Бит 16 **BDRST:** Программный сброс резервного домена.

Читается и пишется программно.

0: Ничего

1: Сбрасывает весь домен

— Бит 15 **RTCEN:** Тактирование RTC.

Читается и пишется программно.

0: Выключено

1: Включено

— Биты 14:10 Резерв, не трогать.

— Биты 9:8 **RTCSEL[1:0]:** Выбор источника тактов RTC.

Пишется программно однократно. Сбрасывается только Сбросом всего домена битом BDRST.

00: Нету

01: LSE

10: LSI

11: HSE / 128

— Биты 7:3 Резерв, не трогать.

— Бит 2 **LSEBYP:** Шунт (обход) LSE.

Читается и пишется программно для обхода LSE в режиме отладки. Пишется только при выключенном LSE.

0: LSE не шунтирован

1: LSE шунтирован

— Бит 1 **LSERDY:** Готовность LSE.

Читается и пишется аппаратно при готовности LSE. После снятия бита LSEON бит LSERDY снимется через 6 тактов генератора.

0: Не готово

1: Готово

— Бит 0 **LSEON:** Разрешение LSE.

Читается и пишется программно.

0: Выключено

1: Включено

7.3.10. Регистр состояния/управления (RCC_CSR)

Смещение адреса: 0x24

По сбросу Резервного Домена: 0x0C00 0000.

Доступ: 0 ≤ тактов ожидания ≤ 3, слово, полуслово и байт. Такты ожидания добавляются при успешном доступе к регистру.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	Res.	RMVF	Reserved							
rw	rw	rw	rw	rw	rw		rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														LSI RDY	LSION
														r	rw

Биты флагов ставятся аппаратно. Чистятся записью бита RMVF.

0: Сброса не было

1: Сброс был

— Бит 31 **LPWRRSTF:** Флаг сброса экономичного питания.
— Бит 30 **WWDGRSTF:** Флаг сброса WWDG.
— Бит 29 **IWDGRSTF:** Флаг сброса IWDG.
— Бит 28 **SFTRSTF:** Флаг программного сброса.
— Бит 27 **PORRSTF:** Флаг сброса POR/PDR.
— Бит 26 **PINRSTF:** Флаг сброса PIN.
— Бит 25 Резерв, не трогать.
— Бит 24 **RMVF:** Снятие флагов сброса.

Пишется программно для снятия флагов сброса.

0: Ничего

1: Флаги сброса снимаются

— Биты 23:2 Резерв, не трогать.
— Бит 1 **LSIRDY:** Готовность LSI.

Читается и пишется аппаратно при готовности LSI. После снятия бита LSION бит LSIRDY снимется через 3 такта генератора.

0: Не готово

1: Готово

— Бит 0 **LSION:** Разрешение LSI.

Читается и пишется программно.

0: Выключено

1: Включено

7.3.11. Карта регистров RCC

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RCC_CR	Reserved						PLL RDY	PLL ON	Reserved				CSSON	HSEBYP	HSERDY	HSEON	HSICAL[7:0]								HSITRIM[4:0]					Reserved	HSIRDY	HSION
	Reset value							0	0					0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0		1	1
0x04	RCC_CFGR	Reserved					MCO [2:0]			Reserved	USBPRE	PLLMUL [3:0]				PLLXTPRE	PLLSRC	ADC PRE [1:0]		PPRE2 [2:0]			PPRE1 [2:0]			HPRE[3:0]				SWS [1:0]		SW [1:0]	
	Reset value						0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	RCC_CIR	Reserved								CSSC	Reserved		PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	Reserved			PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Reserved		PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	Reset value									0			0	0	0	0	0				0	0	0	0	0	0			0	0	0	0	0
0x0C	RCC_APB2RSTR	Reserved										TIM11RST	TIM10RST	TIM9RST	Reserved			ADC3RST	USART1RST	TIM8RST	SPI1RST	TIM1RST	ADC2RST	ADC1RST	IOPGRST	IOPFRST	IOPERST	IOPDRST	IOPCRST	IOPBRST	IOPARST	Reserved	AFIORST
	Reset value											0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x010	RCC_APB1RSTR	Reserved		DACRST	PWRRST	BKPRST	Reserved	CANRST	Reserved	USBRST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Reserved	SPI3RST	SPI2RST	Reserved		WWDGRST	Reserved		TIM14RST	TIM13RST	TIM12RST	TM7RST	TM6RST	TM5RST	TIM4RST	TIM3RST	TIM2RST
	Reset value			0	0	0		0		0	0	0	0	0	0	0		0	0			0			0	0	0	0	0	0	0	0	0
0x14	RCC_AHBENR	Reserved																						SDIOEN	Reserved	FSMCEN	Reserved	CRCEN	Reserved	FLITFEN	Reserved	SRAMEN	DMA2EN	DMA1EN
	Reset value																							0		0		0		1		1	0	0
0x18	RCC_APB2ENR	Reserved										TIM11 EN	TIM10 EN	TIM9 EN	Reserve d			ADC3EN	USART1EN	TIM8EN	SPI1EN	TIM1EN	ADC2EN	ADC1EN	IOPGEN	IOPFEN	IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	Reserved	AFIOEN
	Reset value											0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x1C	RCC_APB1ENR	Reserved		DACEN	PWREN	BKPEN	Reserved	CANEN	Reserved	USBEN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Reserved	SPI3EN	SPI2EN	Reserved		WWDGEN	Reserved		TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset value			0	0	0		0		0	0	0	0	0	0	0		0	0			0			0	0	0	0	0	0	0	0	0
0x20	RCC_BDCR	Reserved															BDRST	RTCEN	Reserved				RTC SEL [1:0]		Reserved						LSEBYP	LSERDY	LSEON
	Reset value																0	0					0	0							0	0	0
0x24	RCC_CSR	LPWRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	Reserved	RMVF	Reserved																						LSIRDY	LSION
	Reset value	0	0	0	0	1	1		0																							0	0

8. Сетевые устройства: сброс и тактирование

8.1. Сброс

Есть три типа сброса: системный, питания и резервного домена.

8.1.1. Системный сброс

Сбрасываются все регистры, кроме флагов сброса в регистре RCC_CSR и регистров резервного домена.

Системный сброс выполняется в следующих случаях:

1. Низкий уровень на ножке NRST (внешний сброс)
2. Конец счёта WWDG
3. Конец счёта IWDG reset
4. Программный сброс (SW reset)
5. Сброс управления экономным питанием.

Источник сброса отмечается в регистре RCC_CSR.

Программный сброс

Для этого нужно установить бит SYSRESETREQ в регистре Application Interrupt and Reset Control Register.

Сброс управления питанием

Есть два способа:

1. Вход в режим Standby:
   Разрешается сбросом бита nRST_STDBY в Байтах Опций Пользователя. В этом случае при успешном выполнении последовательности входа в режим Standby устройств просто сбрасывается.
2. Вход в режим Stop:
   Разрешается сбросом бита nRST_STOP в Байтах Опций Пользователя. В этом случае при успешном выполнении последовательности входа в режим Stop устройств просто сбрасывается.

О Байтах Опций Пользователя написано в *STM32F10xxx Flash programming manual*.

8.1.2. Сброс питания

Генерируется в следующих случаях:

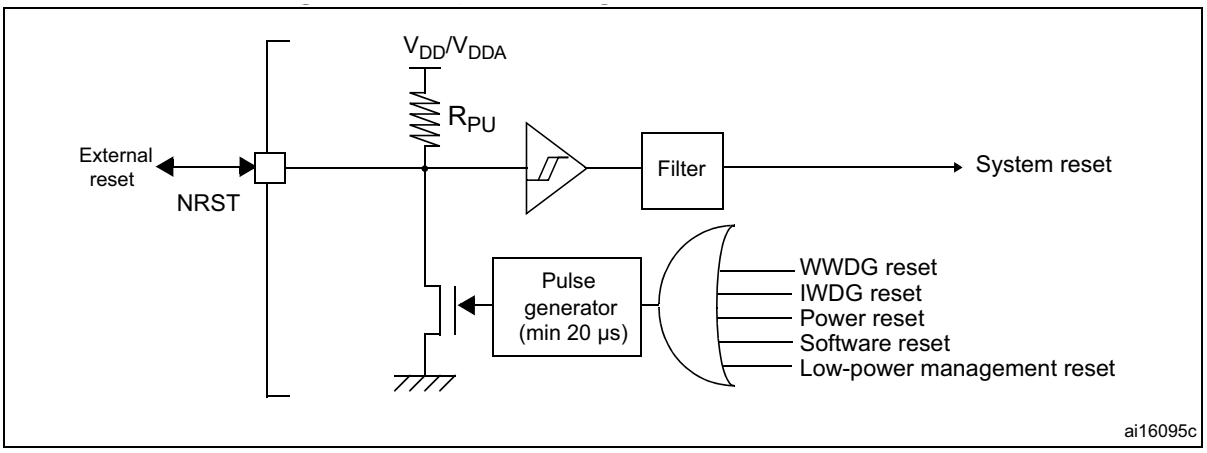
1. Вкл./Выкл. питания (сброс POR/PDR)
2. Выход из режима Standby

Устанавливает все регистры в предписанное состояние, кроме домена резервного хранения.

Эти источники работают как ножка NRST, которая удерживается в низком состоянии на время фазы задержки. Вектор программы сброса расположен по адресу 0x0000_0004 в карте памяти.

Сигнал системного сброса выдаётся на ножку NRST. Генератор импульса гарантирует минимальную длительность импульса 20 µs для обоих источников (внешнего или внутренного). В случае внешнего сброса импульс генерируется на всё время удержания ножки NRST низкой.

Упрощённая схема сброса.



8.1.3. Сброс резервного домена

Здесь есть два типа сброса:

1. Программный сброс установкой бита BDRST в регистре RCC_BDCR.
2. Включение одного из обоих выключенных V_{DD} или V_{BAT}.

8.2. Тактирование

Для системных тактов (SYSCLK) можно использовать три источника:

- такты генератора HSI
- такты генератора HSE
- такты PLL

Устройства имеют два вторичных источника тактов:

- 40 kHz низкоскоростной внутренний RC (LSI RC), для независимого сторожевого таймера и необязательно RTC для Автопробуждения из режима Stop/Standby.
- 32.768 kHz низкоскоростной внешний кварцевый генератор (LSE crystal) для возможного тактирования часов реального времени (RTCCLK).

Во имя экономии электронов все источники можно включать и выключать независимо.

Ради гибкости выбора между внешним и внутренним генераторами для тактов ядра и периферии с наивысшей частотой и гарантии нужных частот для Ethernet и USB OTG FS в развитом контроллере тактирования есть аж 3 PLL.

Отдельный кварцевый генератор 25 MHz может питать всю систему и всю периферию, включая Ethernet и USB OTG FS. Для получения высококачественного звука можно использовать свой кварцевый генератор. В этом случае головной генератор I2S может выдать все стандартные частоты выборки от 8 kHz до 96 kHz с погрешностью менее 0.5%. Подробности см. в описании устройства.

Несколько предделителей допускают конфигурацию частот доменов AHB, высокоскоростного APB (APB2) и низкоскоростного APB (APB1). Максимальная частота доменов AHB и APB2 равна 72 MHz. Максимальная допустимая частота домена APB1 равна 36 MHz.

Вся периферия тактируется от SYSCLK кроме:

- Такты программирования флэш памяти (FLITFCLK) это всегда HSI
- Такты USB OTG FS 48 MHz получаются из PLL VCO (2 × PLLCLK), с последующим программируемым предделителем (делённые на 3 2). Это выбирается битом OTGFSPRE в регистре RCC_CFGR. Для нормальной работы USB OTG FS нужно конфигурировать PLL на выдачу 72 MHz или 48 MHz.
- Такты I2S2 и I2S3 можно получить из SYSCLK или тактов PLL3 VCO (2 × PLL3CLK). Это выбирается битом I2SxSRC в регистре RCC_CFGR2. См. *Секцию 25.4.3*.
- Такты Ethernet MAC (TX, RX и RMII) получают из внешнего PHY. См. *Секцию 29.4.4*. При использовании Ethernet частота тактов AHB должна быть не меньше 25 MHz.

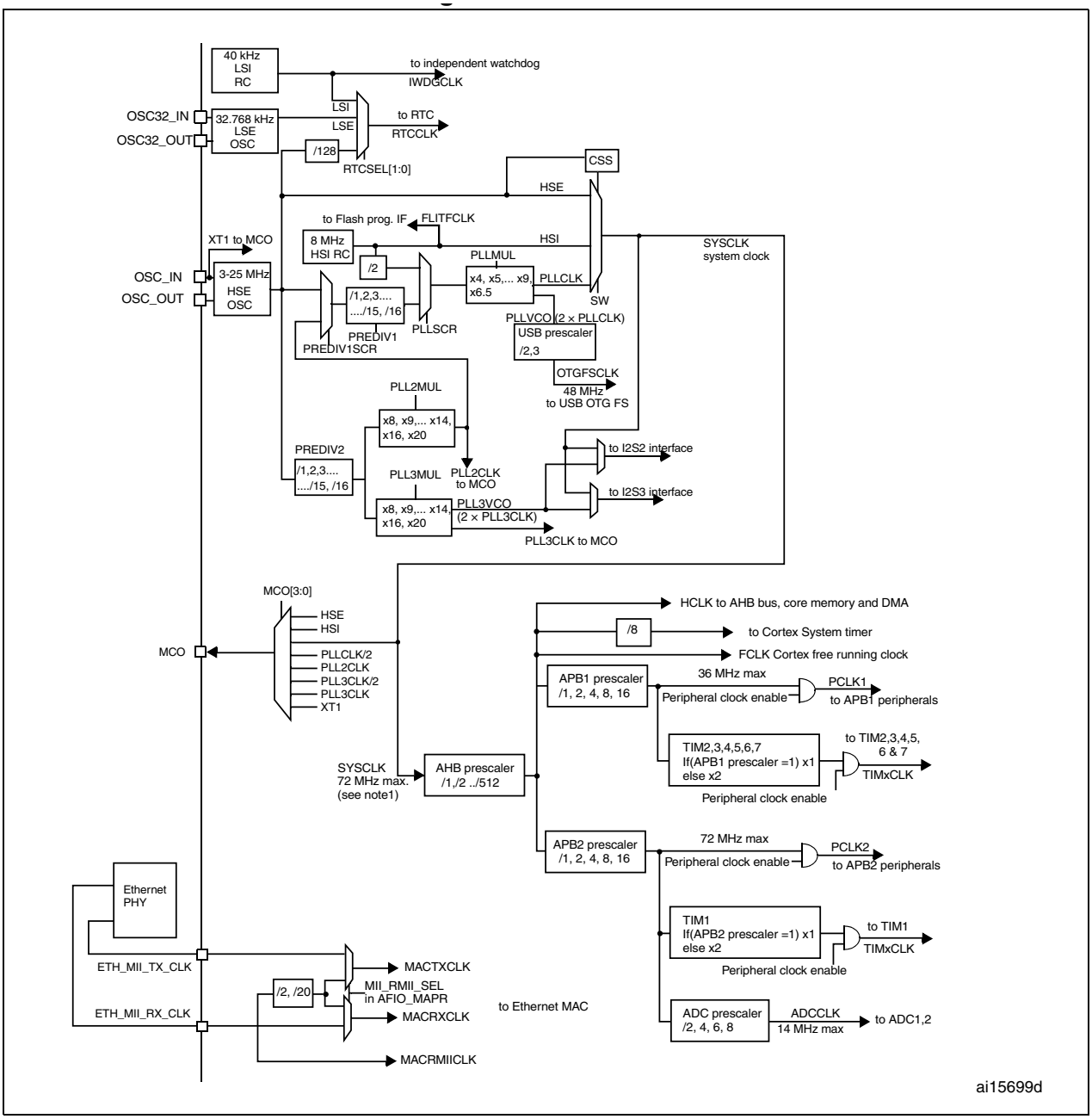
RCC кормит системный таймер (SysTick) внешними тактами AHB HCLK/8. SysTick может работать от них или от HCLK, что выбирается в регистре управления и состояния SysTick. ADC тактируются частотой высокоскоростного домена (APB2) делённой by 2, 4, 6 или 8.

Частоты тактов таймера фиксируются аппаратно:

1. если предделитель APB равен 1, то они равны частоте своего домена APB.
2. иначе они равны удвоенной (×2) частоте своего домена APB.

FCLK работает как независимый тактовый генератор Cortex®-M3. Детали в Arm® *Cortex-M3 r1p1 Technical Reference Manual (TRM)*.

Дерево тактирования



1. При питании PLL от HSI максимальная частота системных тиков может достичь 36 MHz.

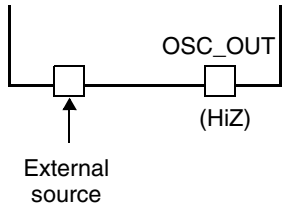
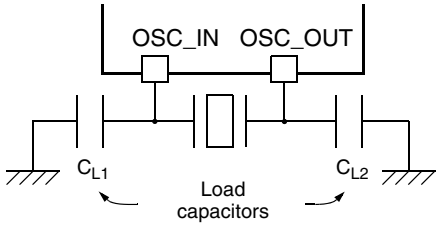
8.2.1. Такты HSE

Высокоскоростной внешний тактовый сигнал (HSE) можно получить из двух источников:

- внешний кварцевый/керамический резонатор HSE
- внешние такты HSE пользователя

Резонатор и конденсаторы нагрузки надо размещать как можно ближе к ножкам генератора для снижения помех и стабилизации времени запуска. Ёмкости нагрузки должны соответствовать выбранному генератору.

Источники HSE/LSE

Clock source	Hardware configuration
External clock	
Crystal/Ceramic resonators	

Внешний источник (HSE bypass)

В этом режиме внешний источник должен иметь частоту 50 MHz. Он включается битами `HSEBYP` и `HSEON` в регистре `RCC_CR`. Внешний сигнал (меандр, синус или треугольник) с ~50% заполнением подаётся на ножку OSC_IN, ножка OSC_OUT должна оставаться в третьем (hi-Z) состоянии.

Внешний кварцевый/керамический резонатор (HSE crystal)

Выгода 3 - 25 MHz внешнего генератора в очень точных главных тактах.

Стабильность внешнего генератора показывается флагом `HSERDY` в регистре `RCC_CR`. При включении такты не выпускаются на волю пока аппаратура не поставит этот бит. При этом может быть вызвано прерывание, если разрешено в регистре `RCC_CIR`.

HSE Crystal включается/выключается битом `HSEON` в регистре `RCC_CR`.

8.2.2. Такты HSI

Тактовый сигнал HSI выдаётся внутренним 8 MHz RC генератором как системные такты или, поделив на 2, входом для PLL (ФАПЧ).

Выгода HSI RC генератора в низкой стоимости. Да и запускается быстрее, чем HSE crystal генератор, но даже после калибровки точность всё равно ниже кварцевого или керамического.

Калибровка

Частоты RC генератора каждого чипа калибруются фирмой ST до 1% точности при TA=25°C.

После сброса, заводская калибровка загружается в биты `HSICAL[7:0]` регистра `RCC_CR`.

Если ваша система работает при колебаниях напряжения и температуры, то частота RC генератора может меняться. Подстроить частоту HSI можно битами `HSITRIM[4:0]` в регистре `RCC_CR`.

Флаг `HSIRDY` в регистре `RCC_CR` показывает стабильность HSI RC. При включении такты HSI RC не выпускаются на волю пока аппаратура не поставит этот бит.

HSI RC включается/выключается битом `HSION` в регистре `RCC_CR`.

Сигнал HSI можно использовать как резервный (Auxiliary clock) при сбое HSE. См. *Секцию 7.2.7: Система безопасности тактирования (CSS)*.

8.2.3. PLL (ФАПЧ)

Главный PLL используется для умножения частоты от:

- HSI RC / 2
- HSE или PLL2 через программируемый делитель.

PLL2 и PLL3 тактируются от HSE через программируемый особый делитель.

Конфигурацию каждого PLL (выбор источника, значение предделителя и множитель) нужно делать до включения PLL. Разрешать их нужно после стабилизации (флаг готовности). После включения PLL эти параметры изменить нельзя.

Готовность PLL может вызвать прерывание, если оно разрешено в регистре RCC_CIR.

При смене источника главного PLL исходный источник можно выключать только после выбора нового битом PLLSRC в регистре RCC_CFGR.

8.2.4. Такты LSE

LSE crystal это 32.768 kHz низкоскоростной внешний кварцевый или керамический резонатор. Его выгода в высокой точности сигнала для RTC и другой периферии.

Включается/выключается LSE битом LSEON в регистре резервного домена RCC_BDCR.

Флаг LSERDY в регистре RCC_BDCR показывает стабильность LSE crystal. При включении такты LSE crystal не выпускаются на волю пока аппаратура не поставит этот бит. При этом может быть вызвано прерывание, если разрешено в регистре RCC_CIR.

Внешний источник (LSE bypass)

Внешний источник частотой до 1 MHz выбирается установкой битов LSEBYP и LSEON в регистре RCC_BDCR. Внешний сигнал (меандр, синус или треугольник) с ~50% заполнением подаётся на ножку OSC32_IN, ножка OSC32_OUT должна оставаться в третьем (hi-Z) состоянии.

8.2.5. Такты LSI

Генератор LSI RC частотой около 40 kHz (между 30 kHz и 60 kHz) продолжает работать в режимах Stop и Standby ради независимого сторожевого таймера (IWDG) и блока авто-побудки (AWU).

Включается/выключается LSI RC битом LSION в регистре RCC_CSR.

Флаг LSIRDY в регистре RCC_CSR показывает стабильность LSE crystal. При включении такты LSE crystal не выпускаются на волю пока аппаратура не поставит этот бит. При этом может быть вызвано прерывание, если разрешено в регистре RCC_CIR.

Калибровка LSI

Она позволяет получить сигнал приемлемой точности для RTC и IWDG.

Калибровка выполняется измерением частоты LSI по отношению у входным тактам TIM5 (TIM5CLK) с точностью генератора HSE. Теперь можно программно подстроить 20-бит предделитель RTC или задержку IWDG.

Процедура калибровки LSI:

1. Включить таймер TIM5 и его канал 4 на захват входа
2. Установить бит TIM5CH4_IREMAP в регистре AFIO_MAPR для подачи тактов LSI на TIM5 - 4.
3. Измерить частоту тактов LSI через событие или прерывание TIM5 Capture/compare 4.
4. С помощью измерения обновить 20-бит предделитель RTC и/или вычислить задержку IWDG.

8.2.6. Выбор SysClk

После системного сброса такты системы берутся от HSI. Источник системных тактов, подключенный напрямую, или через PLL остановить нельзя.

Переключение на новый источник тактов возможно только если он готов (стабилен после запуска или PLL зафиксирован). Если выбран неготовый источник, то он подключится только после готовности. Биты состояния в регистре RCC_CR показывают готовые генераторы и кто выбран системным.

8.2.7. Система безопасности тактирования (CSS)

CSS активируется программно и включается после задержки запуска HSE, выключается после остановки генератора.

Если в тактах HSE обнаруживается сбой, то он автоматически отключается, таймеру TIM1 посылается событие отключения входа и генерируется прерывание (Clock Security System Interrupt CSSI), подключённое к вектору исключения NMI.

NB: Если при включённой CSS возникает сбой HSE, прерывание NMI происходит независимо от того, очищен ли бит удержания CSS. Следовательно в NMI ISR нужно очистить прерывание CSS установкой бита `CSSC` в регистре `RCC_CIR`.

Если генератор HSE служит источником системных тактов напрямую или через PLL/PLL2, то обнаруженный сбой переключает тактирование на HSI и выключает HSE и PLL (в случае его использования).

8.2.8. Такты RTC

Источником тактов RTCCLK могут быть HSE/128, LSE или LSI. Он выбирается битами `RTCSEL[1:0]` в регистре `RCC_BDCR`. Этот выбор нельзя изменить без сброса домена резервного хранения.

Генератор LSE расположен в резервном домене, а HSE и LSI нет. Следовательно:

- Если RTC тактируется от LSE:
  - RTC продолжает работать при отключении V_{DD}, питаясь от V_{BAT}.
- Если блок авто-побудки (AWU) тактируется от LSI:
  - Состояние AWU при отключении V_{DD} не гарантируется. См. *Секцию 7.2.5*.
- Если RTC тактируется от HSE/128:
  - При отключении V_{DD} или регулятора напряжения (выключения 1.8 V домена) состояние RTC не гарантируется.
  - Нужно установить бит `DPB` (отключение защиты записи) в регистре `PWR_CR` (*Секция 5.4.1*).

8.2.9. Такты сторожевого таймера

При аппаратном или программном запуске независимого сторожевого таймера (IWDG) генератор LSI включается и не выключается. Тактирование на IWDG пропускается после задержки запуска LSI.

8.2.10. Вывод тактового сигнала

Существует возможность выводить на внешнюю ножку тактовый сигнал микроконтроллера (MCO). Регистры конфигурации соответствующего порты GPIO должны быть переключены в альтернативный режим. Можно выводить один из 8 сигналов.

- SYSCLK
- HSI
- HSE
- PLL/2
- PLL2
- PLL3/2
- внешний XT1 3-25 MHz генератор (для Ethernet)
- PLL3 (для Ethernet)

Он выбирается битами `MCO[3:0]` в регистре `RCC_CFGR`.

8.3. Регистры RCC

8.3.1. Регистр управления (RCC_CR)

Смещение адреса: `0x00`

По сбросу: `0x0000 XX83` где `X` не определено.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		PLL3 RDY	PLL3 ON	PLL2 RDY	PLL2 ON	PLLRD Y	PLLON	Reserved				CSSON	HSEBY P	HSERDY	HSEON
		r	rw	r	rw	r	rw					rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res.	HSIRDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

— Биты 31:30 Резерв, не трогать.

— Бит 29 **PLL3RDY**: Флаг фиксации PLL3.

Ставится аппаратно.

0: PLL3 unlocked

1: PLL3 locked

— Бит 28 **PLL3ON:** Включение PLL3.

Ставится и снимается программно.

0: PLL3 OFF

1: PLL3 ON

— Бит 27 **PLL2RDY**: Флаг фиксации PLL2.

Ставится аппаратно.

0: PLL2 unlocked

1: PLL2 locked

— Бит 26 **PLL2ON:** Включение PLL2.

Ставится и снимается программно. Аппаратно снимается при входе в режим Stop или Standby. Если PLL2 прямо или непрямо используется как системные такты, то снять его невозможно.

0: PLL2 OFF

1: PLL2 ON

— Бит 25 **PLLRDY**: Флаг фиксации PLL.

Ставится аппаратно.

0: PLL unlocked

1: PLL locked

— Бит 24 **PLLON:** Включение PLL.

Ставится и снимается программно. Аппаратно снимается при входе в режим Stop или Standby. Если PLL прямо или непрямо используется как системные такты, то снять его невозможно. Перед очисткой нужно выключить такты USB OTG FS.

0: PLL OFF

1: PLL ON

— Биты 23:20 Резерв, не трогать.

— Бит 19 **CSSON:** Включение системы безопасности.

Управляется программно. При стоящем CSSON детектор тактов аппаратно включается при готовом HSE и выключается при отказе HSE.

0: Детектор OFF

1: Детектор ON (ON если HSE готов , OFF если нет).

— Бит 18 **HSEBYP:** Шунт (обход) **HSE**.

Управляется программно. Бит HSEON должен стоять. Бит HSEBYP можно писать только при выключенном HSE.

0: Внешний генератор 3-25 MHz не обойдён

1: Внешний генератор 3-25 MHz обойдён

— Бит 17 **HSERDY:** Флаг готовности HSE.

Ставится аппаратно при стабильном HSE. Снимается через 6 тактов HSE после снятия HSEON.

0: HSE не готов

1: HSE готов

— Бит 16 **HSEON:** Включение HSE.

Управляется программно. Снимается аппаратно при входе в режим Stop или Standby. Если HSE прямо или непрямо используется как источник системных тиков, то снять его невозможно.

0: HSE генератор OFF

1: HSE генератор ON

— Биты 15:8 **HSICAL[7:0]:** Калибровка HSI.

Заводские установки инициализируются при запуске.

— Биты 7:3 **HSITRIM[4:0]:** Подстройка HSI.

Программная подстройка частоты внутреннего HSI RC при колебаниях напряжения и температуры.

Значение по умолчанию 16, при добавлении к значению HSICAL подстраивает HSI на 8 MHz ± 1%. Шаг подстройки ($F_{hsitrim}$) около 40 kHz между двумя последовательными шагами HSICAL.

— Бит 2 Резерв, не трогать.

— Бит 1 **HSIRDY:** Флаг готовности HSI.

Ставится аппаратно при стабильности внутреннего 8 MHz RC генератора. Снимается через 6 тактов HSI после снятия HSI.

0: HSI не готов

1: HSI готов

— Бит 0 **HSION:** Включение HSI.

Управляется программно. Ставится аппаратно при выходе из режимов Stop или Standby и в случае отказа внешнего 3-25 MHz генератора, используемого для системных тактов. Этот бит невозможно снять при прямом или непрямом использовании для системных тактов или выборе стать таковым.

0: HSI OFF

1: HSI ON

8.3.2. Регистр конфигурации (RCC_CFGR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000.

Доступ: 0 ≤ такты ожидания ≤ 2, слово, полуслово или байт. 1 или 2 такта ожидания появляются только при переключении источника тактов.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				MCO[3:0]				Res.	OTGFS PRE	PLLMUL[3:0]				PLL XTPRE	PLL SRC
				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC PRE[1:0]		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

— Биты 31:28 Резерв, не трогать.

— Биты 27:24 **MCO**: Вывод тактов микроконтроллера.

Изменяется программно.

00xx: No clock

0100: System clock (SYSCLK)

0101: HSI

0110: HSE

0111: PLL / 2

1000: PLL2

1001: PLL3 / 2

1010: XT1 внешний 3-25 MHz генератор (для Ethernet)

1011: PLL3 (для Ethernet)

NB: Здесь могут пропущены несколько тактов при запуске или переключении источника MCO.

При подаче MCO тактов SYSCLK убедитесь, что частота не превышает 50 MHz (максимум для IO).

— Бит 23 Резерв, не трогать.

— Бит 22 **OTGFSPRE:** Предделитель USB OTG FS.

Управляется программно для выдачи 48 MHz тактов USB. Должен быть правильно установлен до разрешения тактов OTG FS в регистре RCC_APB1ENR. При включённом OTG FS не изменяется.

0: PLL VCO (2 × PLLCLK) / 3 (PLL должен выдавать 72 MHz)

1: PLL VCO (2 × PLLCLK) / 2 (PLL должен выдавать 48 MHz)

— Биты 21:18 **PLLMUL:** Множитель PLL.

Пишется программно только при выключенном PLL.

Внимание: Выходная частота PLL не должна превышать 72 MHz.

000x: Резерв

0010: PLL x 4

0011: PLL x 5

0100: PLL x 6

0101: PLL x 7
0110: PLL x 8
0111: PLL x 9
10xx: Резерв
1100: Резерв
1101: PLL x 6.5
11xx: Резерв

— Бит 17 **PLLXTPRE:** Предделитель частоты HSE на входе PLL.

Управляется программно. Это младший бит делителя PREDIV1. Аппаратно это бит[0] регистра RCC_CFGR2. Если биты[3:1] регистра RCC_CFGR2 не установлены, то этот бит определяет деление на 2 (PLLXTPRE=1) или нет (PLLXTPRE=0). Изменяется только при выключенном PLL.

— Бит 16 **PLLSRC**: Источник тактов PLL.

0: HSI / 2
1: Такты из PREDIV1

NB: При изменении источника главного PLL исходный источник нужно отключать только после выбора нового.

— Биты 15:14 **ADCPRE**: Предделитель ADC.

Пишется программно.

00: PCLK2 / 2
01: PCLK2 / 4
10: PCLK2 / 6
11: PCLK2 / 8

— Биты 13:11 **PPRE2**: Предделитель APB2 для получения PCLK2.

Пишется программно.

0xx: HCLK как есть
100: HCLK / 2
101: HCLK / 4
110: HCLK / 8
111: HCLK / 16

— Биты 10:8 **PPRE1**: Предделитель APB1 для получения PCLK1.

Пишется программно.

Внимание: Частота PCLK1 не должна превышать 36 MHz.

0xx: HCLK как есть
100: HCLK / 2
101: HCLK / 4
110: HCLK / 8
111: HCLK / 16

— Биты 7:4 **HPRE**: Предделитель AHB.

Пишется программно.

0xxx: SYSCLK как есть
1000: SYSCLK / 2
1001: SYSCLK / 4
1010: SYSCLK / 8
1011: SYSCLK / 16
1100: SYSCLK / 64
1101: SYSCLK / 128
1110: SYSCLK / 256
1111: SYSCLK / 512

NB: Если предделитель AHB не равен 1, то буфер предвыборки нужно включать.

Внимание: При включённом Ethernet частота AHB должна быть не менее 25 MHz.

— Биты 3:2 **SWS**: Состояние переключателя системных тактов.

Ставится аппаратно, указывает источник системных тактов.

00: HSI
01: HSE
10: PLL
11: не применимо

— Биты 1:0 **SW**: Переключатель системных тактов.

Программно ставится для выбора источника SYSCLK.

Аппаратно переключается на HSI при выходе из режима Stop и Standby или при отказе генератора HSE при включённой системе контроля тактирования CSS.

00: HSI

01: HSE

10: PLL

11: не дозволено.

8.3.3. Регистр прерываний (RCC_CIR)

Смещение адреса: 0x08

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт. Флаги чистятся записью "1" в нужный бит. Прерывания разрешаются записью "1".

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CSSC	PLL3 RDYC	PLL2 RDYC	PLL RDYC	HSE RDYC	HSI RDYC	LSE RDYC	LSI RDYC
								w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PLL3 RDYIE	PLL2 RDYIE	PLL RDYIE	HSE RDYIE	HSI RDYIE	LSE RDYIE	LSI RDYIE	CSSF	PLL3 RDYF	PLL2 RDYF	PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF
	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r

— Биты 31:24 Резерв, не трогать.

— Бит 23 **CSSC**: Очистка флага прерывания CSS.

— Бит 22 **PLLR3DYC:** Очистка флага прерывания готовности PLL3.

— Бит 21 **PLL2RDYC:** Очистка флага прерывания готовности PLL2.

— Бит 20 **PLLRDYC:** Очистка флага прерывания готовности PLL.

— Бит 19 **HSERDYC:** Очистка флага прерывания готовности HSE.

— Бит 18 **HSIRDYC:** Очистка флага прерывания готовности HSI.

— Бит 17 **LSERDYC:** Очистка флага прерывания готовности LSE.

— Бит 16 **LSIRDYC:** Очистка флага прерывания готовности LSI.

— Бит 15 Резерв, не трогать.

— Бит 14 **PLL3RDYIE:** Разрешение прерывания готовности PLL3.

— Бит 13 **PLL2RDYIE:** Разрешение прерывания готовности PLL2.

— Бит 12 **PLLRDYIE:** Разрешение прерывания готовности PLL.

— Бит 11 **HSERDYIE:** Разрешение прерывания готовности HSE.

— Бит 10 **HSIRDYIE:** Разрешение прерывания готовности HSI.

— Бит 9 **LSERDYIE:** Разрешение прерывания готовности LSE.

— Бит 8 **LSIRDYIE:** Разрешение прерывания готовности LSI.

— Бит 7 **CSSF:** Флаг CSS.

Ставится аппаратно при отказе внешнего генератора 3 - 25 MHz. Снимается записью бита CSSC.

0: Отказа нет

1: Отказ есть

— Биты 6:5 Резерв, не трогать.

— Бит 4 **PLLRDYF:** Флаг прерывания готовности PLL.

Ставится аппаратно при фиксации PLL и стоящем PLLRDYDIE. Чистится записью бита PLLRDYC.

0: Прерывания нет

1: Прерывание есть

— Бит 3 **HSERDYF:** Флаг прерывания готовности HSE.

Ставится аппаратно при готовности HSE и стоящем HSERDYDIE. Чистится записью бита HSERDYC.

0: Прерывания нет

1: Прерывание есть

— Бит 2 **HSIRDYF:** Флаг прерывания готовности HSI.

Ставится аппаратно при готовности HSI и стоящем HSIRDYDIE. Чистится записью бита HSIRDYC.

0: Прерывания нет

1: Прерывание есть

— Бит 1 **LSERDYF:** Флаг прерывания готовности LSE.
Ставится аппаратно при готовности LSE и стоящем LSERDYDIE. Чистится записью бита LSERDYC.
0: Прерывания нет
1: Прерывание есть

— Бит 0 **LSIRDYF:** Флаг прерывания готовности LSI.
Ставится аппаратно при готовности LSI и стоящем LSIRDYDIE. Чистится записью бита LSIRDYC.
0: Прерывания нет
1: Прерывание есть

8.3.4. Регистр сброса периферии APB2 (RCC_APB2RSTR)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт. Устройства сбрасываются записью "1".

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 RST	Res.	SPI1 RST	TIM1 RST	ADC2 RST	ADC1 RST	Reserved		IOPE RST	IOPD RST	IOPC RST	IOPB RST	IOPA RST	Res.	AFIO RST
	rw		rw	rw	rw	rw			rw	rw	rw	rw	rw		rw

— Биты 31:15 Резерв, не трогать.
— Бит 14 **USART1RST:** Сброс USART1.
— Бит 13 Резерв, не трогать.
— Бит 12 **SPI1RST:** Сброс SPI1.
— Бит 11 **TIM1RST:** Сброс таймера TIM1.
— Бит 10 **ADC2RST:** Сброс ADC2.
— Бит 9 **ADC1RST:** Сброс ADC1.
— Бит 8:7 Резерв, не трогать.
— Бит 6 **IOPERST:** Сброс IO порта E.
— Бит 5 **IOPDRST:** Сброс IO порта D.
— Бит 4 **IOPCRST:** Сброс IO порта C.
— Бит 3 **IOPBRST:** Сброс IO порта B.
— Бит 2 **IOPARST:** Сброс IO порта A.
— Бит 1 Резерв, не трогать.
— Бит 0 **AFIORST:** Сброс альтернативных функций IO.

8.3.5. Регистр сброса периферии APB1 (RCC_APB1RSTR)

Смещение адреса: 0x10

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт. Устройства сбрасываются записью "1".

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DAC RST	PWR RST	BKP RST	CAN2 RST	CAN1 RST	Reserved		I2C2 RST	I2C1 RST	UART5 RST	UART4 RST	USART3 RST	USART2 RST	Res.
		rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 RST	SPI2 RST	Reserved		WWDG RST	Reserved					TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw			rw						rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать.
— Бит 29 **DACRST**: Сброс DAC.
— Бит 28 **PWRRST**: Сброс PWR.
— Бит 27 **BKPRST**: Сброс PWR.
— Бит 26 **CAN2RST:** Сброс CAN2.
— Бит 25 **CAN1RST:** Сброс CAN1.
— Бит 24:23 Резерв, не трогать.
— Бит 22 **I2C2RST:** Сброс I2C2.

— Бит 21 **I2C1RST:** Сброс I2C1.
— Бит 20 **UART5RST:** Сброс UART5.
— Бит 19 **UART4RST:** Сброс UART4.
— Бит 18 **USART3RST:** Сброс USART3.
— Бит 17 **USART2RST:** Сброс USART2.
— Бит 16 Резерв, не трогать.
— Бит 15 **SPI3RST:** Сброс SPI3.
— Бит 14 **SPI2RST:** Сброс **SPI2**.
— Биты 13:12 Резерв, не трогать.
— Бит 11 **WWDGRST:** Сброс WWDG.
— Биты 10:6 Резерв, не трогать.
— Бит 5 **TIM7RST:** Сброс TIM7.
— Бит 4 **TIM6RST:** Сброс TIM6.
— Бит 3 **TIM5RST:** Сброс TIM5.
— Бит 2 **TIM4RST:** Сброс TIM4.
— Бит 1 **TIM3RST:** Сброс TIM3.
— Бит 0 **TIM2RST:** Сброс TIM2.

8.3.6. Регистр разрешения тактов периферии AHB (RCC_AHBENR)

Смещение адреса: 0x14

По сбросу: 0x0000 0014.

Доступ: без ожидания, слово, полуслово и байт. Тактирование разрешается записью "1".

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETHMACTXEN	ETHMACEN	Res.	OTGFSEN	Reserved					CRCEN	Res.	FLITFEN	Res.	SRAMEN	DMA2EN	DMA1EN
rw	rw		rw						rw		rw		rw	rw	rw

— Биты 31:17 Резерв, не трогать.
— Бит 16 **ETHMACRXEN**: Тактирование Ethernet MAC RX.
NB: Если включено в режиме RMII, то включается и тактирование RMII для MAC.
— Бит 15 **ETHMACTXEN**: Тактирование Ethernet MAC TX.
NB: Если включено в режиме RMII, то включается и тактирование RMII для MAC.
— Бит 14 **ETHMACEN**: Тактирование Ethernet MAC.
NB: Интерфейс PHY (MII/RMII) нужно выбрать до разрешения тактов MAC.
— Бит 13 Резерв, не трогать.
— Бит 12 **OTGFSEN**: Тактирование USB OTG FS.
— Биты 11:7 Резерв, не трогать.
— Бит 6 **CRCEN**: Тактирование CRC.
— Бит 5 Резерв, не трогать.
— Бит 4 **FLITFEN**: Тактирование FLITF в режиме Sleep.
— Бит 3 Резерв, не трогать.
— Бит 2 **SRAMEN**: Тактирование SRAM в режиме Sleep.
— Бит 1 **DMA2EN**: Тактирование DMA2.
— Бит 0 **DMA1EN**: Тактирование DMA1.

8.3.7. Регистр разрешения тактов периферии APB2 (RCC_APB2ENR)

Смещение адреса: 0x18

По сбросу: 0x0000 0000.

Доступ: слово, полуслово и байт.

Без ожиданий, но во время доступа к периферии APB2 здесь добавляются циклы ожидания до завершения такого доступа. Тактирование разрешается записью "1".

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART 1EN	Res.	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	Reserved		IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.	AFIO EN
	rw		rw	rw	rw	rw			rw	rw	rw	rw	rw		rw

— Биты 31:15 Резерв, не трогать.
— Бит 14 **USART1EN:** Тактирование USART1.
— Бит 13 Резерв, не трогать.
— Бит 12 **SPI1EN:** Тактирование SPI1.
— Бит 11 **TIM1EN:** Тактирование TIM1.
— Бит 10 **ADC2EN:** Тактирование ADC2.
— Бит 9 **ADC1EN:** Тактирование ADC1.
— Биты 8:7 Резерв, не трогать**.**
— Бит 6 **IOPEEN:** Тактирование IO порта E.
— Бит 5 **IOPDEN:** Тактирование IO порта D.
— Бит 4 **IOPCEN:** Тактирование IO порта C.
— Бит 3 **IOPBEN:** Тактирование IO порта B.
— Бит 2 **IOPAEN:** Тактирование IO порта A.
— Бит 1 Резерв, не трогать.
— Бит 0 **AFIOEN:** Тактирование альтернативных функций IO.

8.3.8. Регистр разрешения тактов периферии APB1 (RCC_APB1ENR)

Смещение адреса: 0x1C

По сбросу: 0x0000 0000.

Доступ: слово, полуслово и байт.

Без ожиданий, но во время доступа к периферии APB1 здесь добавляются циклы ожидания до завершения такого доступа. Тактирование разрешается записью "1".

NB: При выключении тактов периферии их регистры могут не читаться и всегда возвращать 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DAC EN	PWR EN	BKP EN	CAN2 EN	CAN1 EN	Reserved		I2C2 EN	I2C1 EN	UART5E N	UART4E N	USART3 EN	USART2 EN	Res.
		rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Reserved		WWD GEN	Reserved					TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw						rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать.
— Бит 29 **DACEN**: Тактирование DAC.
— Бит 28 **PWREN**: Тактирование PWR.
— Бит 27 **BKPEN**: Тактирование BKP.
— Бит 26 **CAN2EN:** Тактирование CAN2.
— Бит 25 **CAN1EN:** Тактирование CAN1.
— Биты 24:23 Резерв, не трогать.
— Бит 22 **I2C2EN:** Тактирование I2C2.
— Бит 21 **I2C1EN:** Тактирование I2C1.
— Бит 20 **UART5EN:** Тактирование UART5.
— Бит 19 **UART4EN:** Тактирование UART4.
— Бит 18 **USART3EN:** Тактирование USART3.
— Бит 17 **USART2EN:** Тактирование USART2.
— Бит 16 Резерв, не трогать.
— Бит 15 **SPI3EN:** Тактирование SPI3.
— Бит 14 **SPI2EN:** Тактирование SPI2.
— Биты 13:12 Резерв, не трогать.

— Бит 11 **WWDGEN:** Тактирование WWDG.
— Биты 10:6 Резерв, не трогать.
— Бит 5 **TIM7EN:** Тактирование TIM7.
— Бит 4 **TIM6EN:** Тактирование TIM6.
— Бит 3 **TIM5EN:** Тактирование TIM5.
— Бит 2 **TIM4EN:** Тактирование TIM4.
— Бит 1 **TIM3EN:** Тактирование TIM3.
— Бит 0 **TIM2EN:** Тактирование TIM2.

8.3.9. Регистр управления резервным доменом (RCC_BDCR)

Смещение адреса: 0x20

По сбросу Резервного Домена: 0x0000 0000.

Доступ: 0 ≤ тактов ожидания ≤ 3, слово, полуслово и байт. Такты ожидания добавляются при успешном доступе к регистру.

NB: Биты LSEON, LSEBYP, RTCSEL и RTCEN регистра RCC_BDCR лежат в резервном домене. Значит после Сброса они защищены от записи и перед их изменением надо установить бит DBP в регистре PWR_CR. См. *Секцию 5*. Обнуляются они только Сбросом резервного домена и ничем иным (см. *Секцию 7.1.3*).

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC EN	Reserved					RTCSEL[1:0]		Reserved					LSE BYP	LSE RDY	LSEON
rw						rw	rw						rw	r	rw

— Биты 31:17 Резерв, не трогать.
— Бит 16 **BDRST:** Программный сброс резервного домена.
0: Ничего
1: Сбрасывает весь домен
— Бит 15 **RTCEN:** Тактирование RTC.
Читается и пишется программно.
0: Выключено
1: Включено
— Биты 14:10 Резерв, не трогать.

— Биты 9:8 **RTCSEL[1:0]:** Выбор источника тактов RTC.
Пишется программно однократно. Сбрасывается только Сбросом всего домена битом BDRST.
00: Нету
01: LSE
10: LSI
11: HSE / 128
— Биты 7:3 Резерв, не трогать.
— Бит 2 **LSEBYP:** Шунт (обход) LSE.
Читается и пишется программно для обхода LSE в режиме отладки. Пишется только при выключенном LSE.
0: LSE не шунтирован
1: LSE шунтирован
— Бит 1 **LSERDY:** Готовность LSE.
Читается и пишется аппаратно при готовности LSE. После снятия бита LSEON бит LSERDY снимется через 6 тактов генератора.
0: Не готово
1: Готово
— Бит 0 **LSEON:** Разрешение LSE.
Читается и пишется программно.
0: Выключено
1: Включено

8.3.10. Регистр состояния/управления (RCC_CSR)

Смещение адреса: 0x24

По сбросу Резервного Домена: 0x0C00 0000.

Доступ: 0 ≤ тактов ожидания ≤ 3, слово, полуслово и байт. Такты ожидания добавляются при успешном доступе к регистру.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	Res.	RMVF	Reserved							
rw	rw	rw	rw	rw	rw		rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														LSI RDY	LSION
														r	rw

— Бит 31 **LPWRRSTF:** Флаг сброса экономичного питания.
Ставится аппаратно при сбросе управления питанием. Чистится записью бита RMVF.
0: Сброса не было
1: Сброс был

— Бит 30 **WWDGRSTF:** Флаг сброса WWDG.
Ставится аппаратно при сбросе WWDG. Чистится записью бита RMVF.
0: Сброса не было
1: Сброс был

— Бит 29 **IWDGRSTF:** Флаг сброса IWDG.
Ставится аппаратно при сбросе IWDG из V_{DD} домена. Чистится записью бита RMVF.
0: Сброса не было
1: Сброс был

— Бит 28 **SFTRSTF:** Флаг программного сброса.
Ставится аппаратно при программном сбросе. Чистится записью бита RMVF.
0: Сброса не было
1: Сброс был

— Бит 27 **PORRSTF:** Флаг сброса POR/PDR.
Ставится аппаратно при сбросе POR/PDR. Чистится записью бита RMVF.
0: Сброса не было
1: Сброс был

— Бит 26 **PINRSTF:** Флаг сброса PIN.
Ставится аппаратно при сбросе с ножки NRST. Чистится записью бита RMVF.
0: Сброса не было
1: Сброс был

— Бит 25 Резерв, не трогать.

— Бит 24 **RMVF:** Снятие флагов сброса.
Пишется программно для снятия флагов сброса.
0: Ничего
1: Флаги сброса снимаются

— Биты 23:2 Резерв, не трогать.

— Бит 1 **LSIRDY:** Готовность LSI.
Читается и пишется аппаратно при готовности LSI. После снятия бита LSION бит LSIRDY снимется через 3 такта генератора.
0: Не готово
1: Готово

— Бит 0 **LSION:** Разрешение LSI.
Читается и пишется программно.
0: Выключено
1: Включено

8.3.11. Регистр сброса периферии AHB (RCC_AHBRSTR)

Смещение адреса: 0x28

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт. Устройство сбрасывается записью "1".

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ETHMACRST	Res.	OTGFSRST	Reserved											
	rw		rw												

— Биты 31:15 Резерв, не трогать.

— Бит 14 **ETHMACRST:** Сброс Ethernet MAC.

— Бит 13 Резерв, не трогать.

— Бит 12 **OTGFSRST:** Сброс USB OTG FS.

— Биты 11:0 Резерв, не трогать.

8.3.12. Второй регистр конфигурации (RCC_CFGR2)

Смещение адреса: 0x2C

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													I2S3SRC	I2S2SRC	PREDIV1SRC
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL3MUL[3:0]				PLL2MUL[3:0]				PREDIV2[3:0]				PREDIV1[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:19 Резерв, не трогать.

— Бит 18 **I2S3SRC:** Источник тактов I2S3.

Читается и пишется программно. Выбирать нужно до включения тактов I2S3.

0: System clock (SYSCLK)

1: PLL3 VCO

— Бит 17 **I2S2SRC:** Источник тактов I2S2.

Читается и пишется программно. Выбирать нужно до включения тактов I2S2.

0: System clock (SYSCLK)

1: PLL3 VCO

— Бит 16 **PREDIV1SRC:** Источник тактов PREDIV1.

Читается и пишется программно. Выбирать нужно до включения тактов PLL.

0: HSE

1: PLL2

— Биты 15:12 **PLL3MUL[3:0]:** Множитель PLL3.

Читается и пишется программно. Выбирать нужно до включения тактов PLL3.

00xx: Резерв

010x: Резерв

0110: PLL3 x 8

0111: PLL3 x 9

1000: PLL3 x 10

1001: PLL3 x 11

1010: PLL3 x 12

1011: PLL3 x 13

1100: PLL3 x 14

1101: Резерв

1110: PLL3 x 16

1111: PLL3 x 20

— Бит 11:8 **PLL2MUL[3:0]**: Множитель PLL2.

Читается и пишется программно. Выбирать нужно до включения тактов PLL2.

00xx: Резерв
010x: Резерв
0110: PLL2 x 8
0111: PLL2 x 9
1000: PLL2 x 10
1001: PLL2 x 11
1010: PLL2 x 12
1011: PLL2 x 13
1100: PLL2 x 14
1101: Резерв
1110: PLL2 x 16
1111: PLL2 x 20

— Биты 7:4 **PREDIV2[3:0]**: Предделитель PREDIV2.

Читается и пишется программно. Выбирать нужно до включения тактов PLL2 и PLL3.

0000: Вход не делится
0001: Вход / 2
0010: Вход / 3
0011: Вход / 4
0100: Вход / 5
0101: Вход / 6
0110: Вход / 7
0111: Вход / 8
1000: Вход / 9
1001: Вход / 10
1010: Вход / 11
1011: Вход / 12
1100: Вход / 13
1101: Вход / 14
1110: Вход / 15
1111: Вход / 16

— Биты 3:0 **PREDIV1[3:0]**: Предделитель PREDIV1.

Читается и пишется программно. Выбирать нужно до включения тактов PLL.

NB: Аппаратно бит(0) это бит(17) регистра RCC_CFGR.

0000: Вход не делится
0001: Вход / 2
0010: Вход / 3
0011: Вход / 4
0100: Вход / 5
0101: Вход / 6
0110: Вход / 7
0111: Вход / 8
1000: Вход / 9
1001: Вход / 10
1010: Вход / 11
1011: Вход / 12
1100: Вход / 13
1101: Вход / 14
1110: Вход / 15
1111: Вход / 16

8.3.13. Карта регистров RCC

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RCC_CR	Reserved		PLL3 RDY	PLL3 ON	PLL2 RDY	PLL2 ON	PLL RDY	PLLON	Reserved				CSSON	HSEBYP	HSERDY	HSEON	HSICAL[7:0]								HSITRIM[4:0]					Reserved	HSIRDY	HSION
	Reset value			0	0	0	0	0	0					0	0	0	0	x	x	x	x	x	x	x	x	1	0	0	0	0		1	1
0x004	RCC_CFGR	Reserved				MCO [3:0]				Reserved	OTGFSPRE	PLLMUL [3:0]				PLLXTPRE	PLLSRC	ADC PRE [1:0]		PPRE2 [2:0]			PPRE1 [2:0]			HPRE[3:0]				SWS [1:0]		SW [1:0]	
	Reset value					0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	RCC_CIR	Reserved								CSSC	PLL3RDYC	PLL2RDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	Reserved	PLL3RDYIE	PLL2RDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	PLL3RDYF	PLL2RDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	Reset value									0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	RCC_APB2RSTR	Reserved																	USART1RST	Reserved	SPI1RST	TIM1RST	ADC2RST	ADC1RST	Reserved		IOPERST	IOPDRST	IOPCRST	IOPBRST	IOPARST	Reserved	AFIORST
	Reset value																		0		0	0	0	0			0	0	0	0	0		0
0x010	RCC_APB1RSTR	Reserved		DACRST	PWRRST	BKPRST	CAN2RST	CAN1RST	Reserved		I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Reserved		SPI3RST	SPI2RST	Reserved		WWDGRST	Reserved			TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
	Reset value			0	0	0	0	0			0	0	0	0	0	0			0	0			0				0	0	0	0	0	0	0
0x014	RCC_AHBENR	Reserved															ETHMACRXEN	ETHMACTXEN	ETHMACEN	Reserved	OTGFSEN	Reserved			CRCEN	Reserved	FLITFEN	Reserved	SRAMEN	DM2AEN	DM1AEN		
	Reset value																0	0	0		0				0		1		1	0	0		
0x018	RCC_APB2ENR	Reserved																	USART1EN	Reserved	SPI1EN	TIM1EN	ADC2EN	ADC1EN	Reserved		IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	Reserved	AFIOEN
	Reset value																		0		0	0	0	0			0	0	0	0	0		0
0x01C	RCC_APB1ENR	Reserved		DACEN	PWREN	BKPEN	CAN2EN	CAN1EN	Reserved		I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Reserved		SPI3EN	SPI2EN	Reserved		WWDGEN	Reserved			TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN	
	Reset value			0	0	0	0	0			0	0	0	0	0	0			0	0			0				0	0	0	0	0	0	
0x020	RCC_BDCR	Reserved															BDRST	RTCEN	Reserved						RTC SEL [1:0]		Reserved					LSEBYP	LSERDY	LSEON
	Reset value																0	0							0	0						0	0	0
0x024	RCC_CSR	LPWRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	Reserved	RMVF	Reserved																						LSIRDY	LSION
	Reset value	0	0	0	0	1	1		0																							0	0
0x028	RCC_AHBSTR	Reserved																	ETHMACRST	Reserved	OTGFSRST	Reserved											
	Reset value																		0		0												
0x02C	RCC_CFGR2	Reserved													I2S3SRC	I2S2SRC	PREDIV1SRC	PLL3MUL [3:0]				PLL2MUL [3:0]				PREDIV2[3:0]				PREDIV1[3:0]			
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9. Порты и альтернативные функции (GPIO и AFIO)

9.1. Работа GPIO

Каждый порт I/O общего назначения имеет два 32-бит регистра конфигурации (GPIOx_CRL, GPIOx_CRH), два 32-бит регистра данных (GPIOx_IDR, GPIOx_ODR), 32-бит регистр установки/сброса (GPIOx_BSRR), 16-бит регистр сброса (GPIOx_BRR) и 32-бит регистр замка (GPIOx_LCKR).

Конкретные характеристики портов даны в описаниях устройств. Каждый бит порта может быть программно установлен в один из режимов:

- Ввод, плавающий
- Ввод, подпорка вверх
- Ввод, подтянутый вниз
- Аналоговый
- Вывод, открытый сток
- Вывод, двухтактный
- Альтернативная функция, двухтактный
- Альтернативная функция, открытый сток

Каждый бит I/O порта свободно программируемый, но регистры доступны только 32-бит словами. Регистры GPIOx_BSRR и GPIOx_BRR предназначена для атомарного доступа чтения-модификации-записи к регистрам GPIO. Таким образом устраняется риск возникновения IRQ между чтением и модификацией.

Рис.13 Базовая структура бита порта I/O.

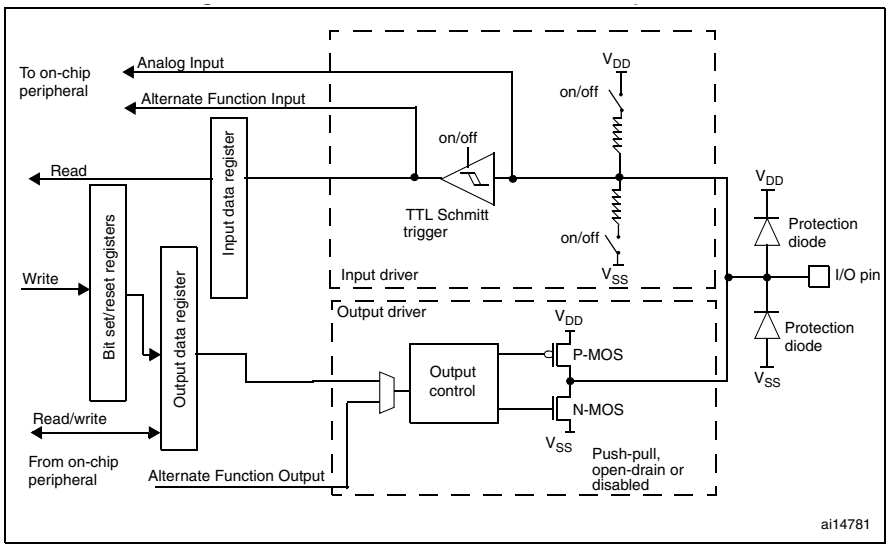
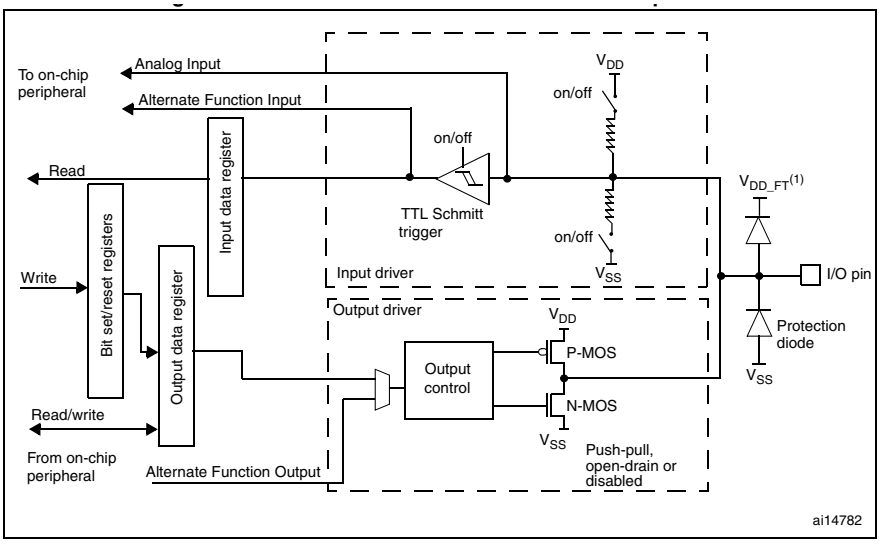


Рис.14 Базовая структура бита 5V порта I/O



1. V_{DD_FT} это потенциал для 5V I/O, отличается от V_{DD}.

Таблица 20. Конфигурация битов порта

Режим конфигурации		CNF1	CNF0	MODE1	MODE0	Регистр PxODR
Вывод общего назначения	Двухтактный	0	0	01 10 11 См. Таблицу 21		0 или 1
	Откр. сток		1			0 или 1
Вывод альтернат.	Двухтактный	1	0			Всё равно
	Откр. сток		1			Всё равно
Ввод	Аналоговый	0	0	00		Всё равно
	Плавающий		1			Всё равно
	Подтяжка вниз	1	0			0
	Подпорка вверх					1

Таблица 21. Биты MODE вывода

MODE[1:0]	Значение
00	Резерв
01	Максимальная частота 10 MHz
10	Максимальная частота 2 MHz
11	Максимальная частота 50 MHz

9.1.1. Вв/Выв общего назначения (GPIO)

Во время и сразу после сброса альтернативные функции отключены и порты I/O ставятся в режим Плавающий Ввод (`CNFx[1:0]=01b, MODEx[1:0]=00b`).

После сброса ножки JTAG стоят во вводе PU/PD:

```
PA15: JTDI в PU
PA14: JTCK в PD
PA13: JTMS в PU
PB4: NJTRST в PU
```

При выводе значения выходного регистра (`GPIOx_ODR`) поступают на ножки I/O. Выходной каскад можно использовать в двухтактном режиме или с открытым стоком (при выводе 0 используется только N-MOS).

Регистр ввода (`GPIOx_IDR`) захватывает данные на ножках I/O каждый цикл APB2.

Все ножки GPIO имеют внутренние резисторы слабой подтяжки вверх и вниз, которые могут подключаться при вводе.

9.1.2. Атомарная установка и снятие битов

При записи регистра `GPIOx_ODR` на уровне битов выключать прерывания нет нужды: можно изменить один или несколько битов за одну атомарную запись APB2. Изменяемые биты выбираются записью ‘1’ в регистры Установки/Сброса (`GPIOx_BSRR`, или только для сброса в `GPIOx_BRR`). Не выбранные биты не изменятся.

9.1.3. Внешние линии прерывания/побудки

Все порты имеют возможность внешнего прерывания. Для этого порт должен быть в режиме ввода. Дальнейшее см. в *Секции 10.2* и *Секции 10.2.3*.

9.1.4. Альтернативные функции (AF)

Для использования альтернативных функций нужно кое-что писать в регистр Конфигурации Битов Порта.

- Для альтернативного ввода внешнего сигнала с ножки нужно и порт поставить в режим ввода (плавающий, с подпоркой или подтянутый вниз).
  NB: Можно программно эмулировать ножку AFI в контроллере GPIO. В этом случае порт нужно поставить в режим альтернативного вывода. И очевидно, что этот порт не должен управляться снаружи (этим займётся контроллер GPIO).
- Для альтернативного вывода порт должно поставить в этот режим (двухтактный или с открытым стоком).
- Для двунаправленных функций порт нужно поставить в альтернативный вывод (двухтактный или с открытым стоком). В этом случае входной каскад переключается в режим плавающего ввода.

Установка бита порта в альтернативный вывод отключает выходной регистр и подключает ножку к выходному сигналу встроенной периферии. Если периферия в этом случае не включена, то выходной сигнал не определён.

9.1.5. Программное перенаправление альтернативных функций

Это делается записью регистров AFIO. См. ниже.

9.1.6. Механизм блокировки GPIO

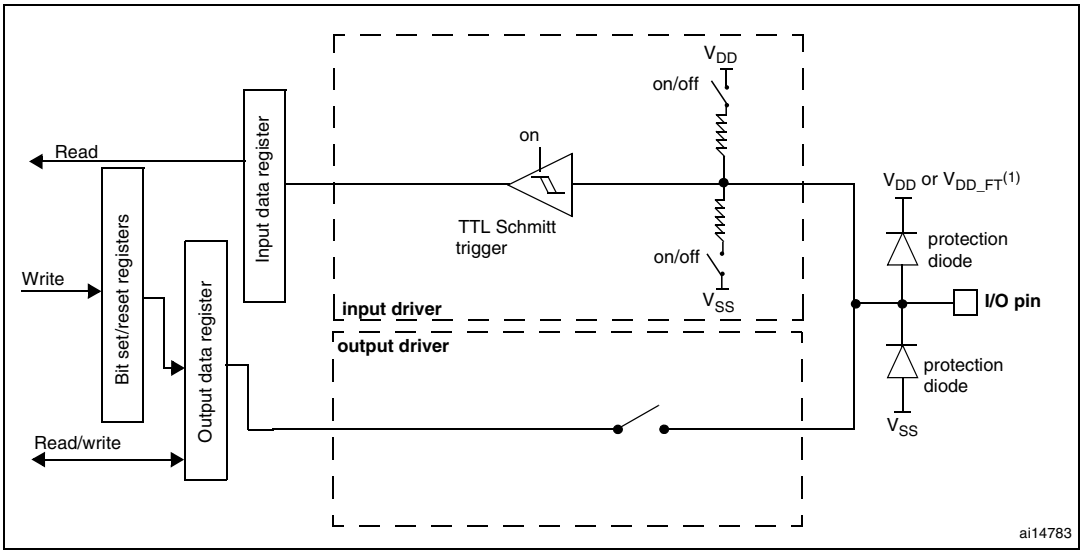
Это позволяет заморозить конфигурацию IO. После прохождения последовательности LOCK для бита порта его изменить нельзя вплоть до следующего сброса.

9.1.7. Конфигурация ввода

При включении порта I/O на ввод:

- Выключается выходной каскад
- Включается входной триггер Шмитта
- В зависимости от конфигурации (подпорка, подтяжка или плавающий) могут включаться нужные резисторы
- Данные на ножке I/O читаются в регистр данных каждый цикл APB2
- Данные получают из регистра данных.

Рис.15 Входная конфигурация бита I/O порта.



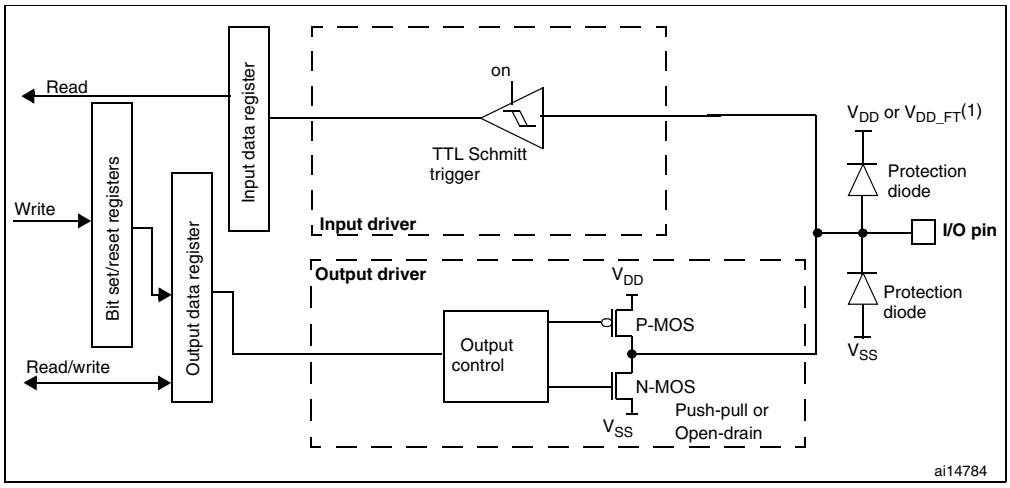
(1)V_{DD_FT} это потенциал для 5V I/O, отличается от V_{DD}.

9.1.8. Конфигурация вывода

При включении порта I/O на вывод:

- Включается выходной каскад:
  - С открытым стоком: “0” в выходном регистре включает N-MOS, а “1” оставляет порт в третьем состоянии Hi-Z (P-MOS никогда не включается)
  - Двухтактный режим: в выходном регистре включает N-MOS, а “1” включает P-MOS
- Включается входной триггер Шмитта
- Выключаются резисторы (подпорка и подтяжка)
- Чтение входного регистра выдаёт состояние в режиме с открытым стоком
- Чтение выходного регистра выдаёт последнее записанное значение.

Рис.16 Выходная конфигурация бита I/O порта.



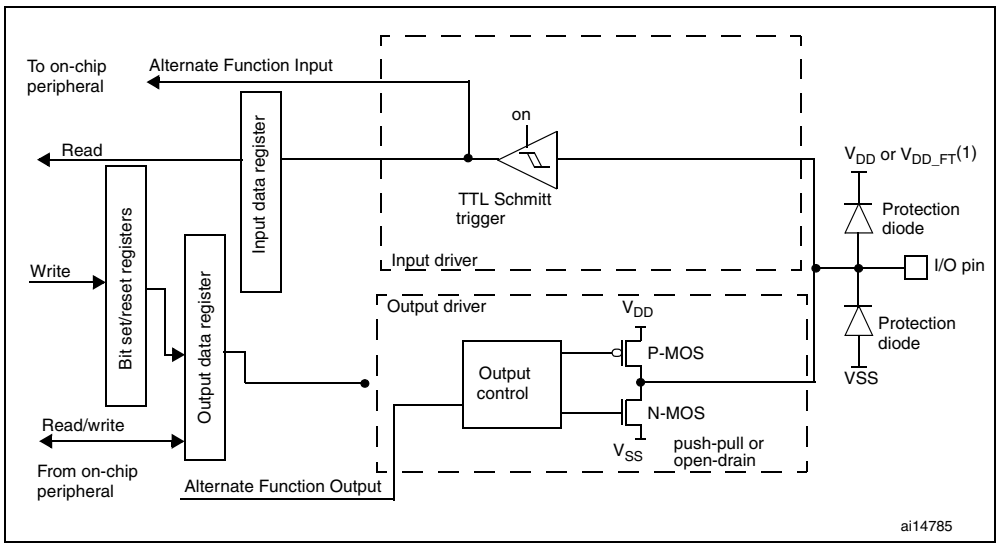
1. V_{DD_FT} это потенциал для 5V I/O, отличается от V_{DD}.

9.1.9. Конфигурация альтернативных функций

При включении альтернативного режима порта:

- Выходной каскад включается в двухтактный режим или с открытым стоком
- Выходной каскад управляется сигналом от периферии
- Включается входной триггер Шмитта
- Выключаются резисторы (подпорка и подтяжка)
- Данные на ножке I/O читаются в регистр данных каждый цикл APB2
- Чтение входного регистра выдаёт состояние в режиме с открытым стоком
- Чтение выходного регистра выдаёт последнее записанное значение.

Рис.17 Альтернативная конфигурация бита I/O порта.



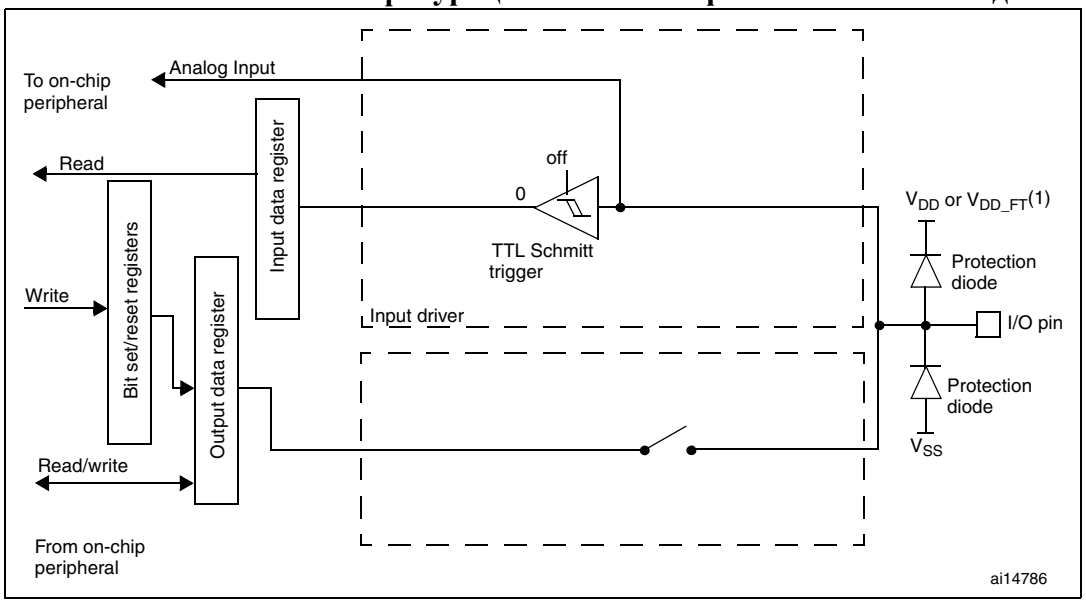
1. V_{DD_FT} это потенциал для 5V I/O, отличается от V_{DD}.

9.1.10. Аналоговая конфигурация

При включении аналогового режима порта:

- Выходной каскад отключается.
- Входной триггер Шмитта отключается, не мешает аналоговому сигналу и постоянно выдаёт “0”.
- Резисторы подпорки и подтяжки отключаются.
- Чтение входного регистра данных выдаёт “0”.

Рис.18 Аналоговая конфигурация бита I/O порта с высоким импедансом.



9.1.11. Конфигурация GPIO для периферии

Таймеры TIM1 и TIM8

Ножки TIM1/8	Конфигурация	Режим GPIO
TIM1/8_CHx	Захват входа канал x	Вход плавающий
	Выход сравнения канал x	Альтернативный двухтактный
TIM1/8_CHxN	Инверсный выход канал x	Альтернативный двухтактный
TIM1/8_BKIN	Вход Останова	Вход плавающий
TIM1/8_ETR	Вход внешнего запуска	Вход плавающий

Таймеры TIM2/3/4/5

Ножки TIMn	Конфигурация	Режим GPIO
TIM2/3/4/5_CHx	Захват входа канал x	Вход плавающий
	Выход сравнения канал x	Альтернативный двухтактный
TIM2/3/4/5_ETR	Вход внешнего запуска	Альтернативный двухтактный

USART

Ножки USART	Конфигурация	Режим GPIO
USARTx_TX(1)	Полный дуплекс	Альтернативный двухтактный
	Синхронный полудуплекс	Альтернативный двухтактный
USARTx_RX	Полный дуплекс	Вход плавающий / с подпоркой
	Синхронный полудуплекс	Свободен. Может быть GPIO
USARTx_CK	Синхронный режим	Альтернативный двухтактный
USARTx_RTS	Аппаратное управление	Альтернативный двухтактный
USARTx_CTS	Аппаратное управление	Вход плавающий / с подпоркой

SPI

Ножки SPI	Конфигурация	Режим GPIO
SPIx_SCK	Ведущий	Альтернативный двухтактный
	Ведомый	Вход плавающий
SPIx_MOSI	Полный дуплекс / ведущий	Альтернативный двухтактный
	Полный дуплекс / ведомый	Вход плавающий / с подпоркой
	Данные двунаправленного симплекса / ведущий	Альтернативный двухтактный
	Данные двунаправленного симплекса / ведомый	Свободен. Может быть GPIO
SPIx_MISO	Полный дуплекс / ведущий	Вход плавающий / с подпоркой
	Полный дуплекс / ведомый (2 точки)	Альтернативный двухтактный
SPIx_MISO	Полный дуплекс / ведомый (многоточечный)	Альтернативный с открытым стоком
	Данные двунаправленного симплекса / ведущий	Свободен. Может быть GPIO
	Данные двунаправленного симплекса / ведомый (2 точки)	Альтернативный двухтактный
	Данные двунаправленного симплекса / ведомый (многоточечный)	Альтернативный с открытым стоком
SPIx_NSS	Аппаратный ведущий / ведомый	Вход плавающий / с подпоркой / с подтяжкой
	Аппаратный ведущий/ NSS выход вкл.	Альтернативный двухтактный
	Программно	Свободен. Может быть GPIO

I2S

Ножки I2S	Конфигурация	Режим GPIO
I2Sx_ WS	Ведущий	Альтернативный двухтактный
	Ведомый	Вход плавающий
I2Sx_CK	Ведущий	Альтернативный двухтактный
	Ведомый	Вход плавающий

Ножки I2S	Конфигурация	Режим GPIO
I2Sx_SD	Передатчик	Альтернативный двухтактный
	Приёмник	Вход плавающий / с подпоркой / с подтяжкой
I2Sx_MCK	Ведущий	Альтернативный двухтактный
	Ведомый	Свободен. Может быть GPIO

I2C

Ножки I2C	Конфигурация	Режим GPIO
I2Cx_SCL	Такты I2C	Альтернативный с открытым стоком
I2Cx_SDA	Данные В/В I2C	Альтернативный с открытым стоком

BxCAN

Ножки BxCAN	Режим GPIO
CAN_TX (Линия данных передачи)	Альтернативный двухтактный
CAN_RX (Линия данных приёма)	Вход плавающий / с подпоркой

USB(1)

Ножки USB	Режим GPIO
USB_DM / USB_DP	При включении USB эти ножки автоматически подключаются к внутреннему USB трансиверу.

1. Эта таблица относится только к устройствам разной плотности.

OTG_FS(1)

Ножки OTG_FS	Конфигурация	Режим GPIO
OTG_FS_SOF	Host	Альтернативный двухтактный, если использован
	Device	Альтернативный двухтактный, если использован
	OTG	Альтернативный двухтактный, если использован
OTG_FS_VBUS(2)	Host	Вход плавающий
	Device	Вход плавающий
	OTG	Вход плавающий
OTG_FS_ID	Host	Не нужен, если программно включён режим Force host (стоит FHMOD в регистре OTG_FS_GUSBCFG)
	Device	Не нужен, если программно включён режим Force device (стоит FDMOD в регистре OTG_FS_GUSBCFG)
	OTG	Вход с подпоркой
OTG_FS_DM	Host	Автоматически управляется USB power-down
	Device	Автоматически управляется USB power-down
	OTG	Автоматически управляется USB power-down

Ножки OTG_FS	Конфигурация	Режим GPIO
OTG_FS_DP	Host	Автоматически управляется USB power-down
	Device	Автоматически управляется USB power-down
	OTG	Автоматически управляется USB power-down

1. Эта таблица относится только к сетевым устройствам.
2. Для ножки OTG_FS_VBUS (PA9), используемой другой общей периферией или как IO общего назначения, нужно включить режим PHY Power-down (очистить бит 16 в регистре OTG_FS_GCCFG).

SDIO

Ножки SDIO	Режим GPIO
SDIO_CK	Альтернативный двухтактный
SDIO_CMD	Альтернативный двухтактный
SDIO[D7:D0]	Альтернативный двухтактный

Вход ADC в GPIO должен быть сконфигурирован аналоговым.

ADC/DAC

Ножки ADC/DAC	Режим GPIO
АЦП/ЦАП	Аналоговый

FSMC

Ножки FSMC	Режим GPIO
FSMC_A[25:0] FSMC_D[15:0]	Альтернативный двухтактный
FSMC_CK	Альтернативный двухтактный
FSMC_NOE FSMC_NWE	Альтернативный двухтактный
FSMC_NE[4:1] FSMC_NCE[3:2] FSMC_NCE4_1 FSMC_NCE4_2	Альтернативный двухтактный
FSMC_NWAIT FSMC_CD	Вход плавающий /с подпоркой
FSMC_NIOS16, FSMC_INTR FSMC_INT[3:2]	Вход плавающий
FSMC_NL FSMC_NBL[1:0]	Альтернативный двухтактный
FSMC_NIORD, FSMC_NIOWR FSMC_NREG	Альтернативный двухтактный

Другие IO

Ножки	Альтернативная функция	Режим GPIO
TAMPER-RTC	Выход RTC	Включается аппаратно при записи регистров BKP_CR и BKP_RTCCR
	Вход события Tamper	
MCO	Выход тактов	Альтернативный двухтактный
Входные линии EXTI	Вход внешних прерываний	Вход плавающий / с подпоркой / с подтяжкой

9.2. Регистры GPIO

Все регистры доступны только как 32-бит слова.

9.2.1. Младший регистр конфигурации порта (GPIOx_CRL) (x=A..G)

Смещение адреса: 0x00

По сбросу: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30, 27:26, 23:22, 19:18, 15:14,11:10, 7:6, 3:2

CNFy[1:0]:Биты конфигурации порта **x** (**y**=0..7)

Пишутся программно. См. Таблицу 20.

Режим ввода (MODE[1:0]=00):

00: Аналоговый

01: Ввод плавающий (по сбросу)

10: Ввод с подпоркой / с подтяжкой

11: Резерв

Режим вывода (MODE[1:0] > 00):

00: Вывод GPIO двухтактный

01: Вывод GPIO открытый сток

10: Вывод альтернативный двухтактный

11: Вывод альтернативный открытый сток

— Биты 29:28, 25:24, 21:20, 17:16, 13:12, 9:8, 5:4, 1:0

MODEy[1:0]: Биты режима порта **x** (**y**=0..7)

Пишутся программно. См. Таблицу 20.

00: Ввод (по сбросу)

01: Вывод, до 10 MHz.

10: Вывод, до 2 MHz.

11: Вывод, до 50 MHz.

9.2.2. Старший регистр конфигурации порта (GPIOx_CRH) (x=A..G)

Смещение адреса: 0x04

По сбросу: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30, 27:26, 23:22, 19:18, 15:14,11:10, 7:6, 3:2

CNFy[1:0]:Биты конфигурации порта **x** (**y**=8..15)

Пишутся программно. См. Таблицу 20.

Режим ввода (MODE[1:0]=00):

00: Аналоговый

01: Ввод плавающий (по сбросу)

10: Ввод с подпоркой / с подтяжкой

11: Резерв

Режим вывода (MODE[1:0] > 00):
00: Вывод GPIO двухтактный
01: Вывод GPIO открытый сток
10: Вывод альтернативный двухтактный
11: Вывод альтернативный открытый сток

— Биты 29:28, 25:24, 21:20, 17:16, 13:12, 9:8, 5:4, 1:0
MODEy[1:0]: Биты режима порта **x** (**y**=8..15)
Пишутся программно. См. Таблицу 20.
00: Ввод (по сбросу)
01: Вывод, до 10 MHz.
10: Вывод, до 2 MHz.
11: Вывод, до 50 MHz.

9.2.3. Регистр данных ввода порта (GPIOx_IDR) (x=A..G)

Смещение адреса: 0x08
По сбросу: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:16 Резерв, не трогать.
— Биты 15:0 **IDRy:** Входные данные (**y**=0..15)
Входные данные порта. Чтение только словом.

9.2.4. Регистр данных вывода порта (GPIOx_ODR) (x=A..G)

Смещение адреса: 0x0C
По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.
— Биты 15:0 **ODRy:** Выходные данные (**y**=0..15)
Выходные данные порта. Запись и чтение только словом.
NB: При атомарном доступе биты ODR можно менять индивидуально записью в регистр GPIOx_BSRR (x = A .. G).

9.2.5. Регистр установки/сброса битов порта (GPIOx_BSRR) (x=A..G)

Смещение адреса: 0x10
По сбросу: 0x0000 0000
Доступен только словом.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

— Биты 31:16 **BRy**: Сброс бита **y** (**y**=0..15).
0: Не изменяется
1: Сбрасывается
— Биты 15:0 **BSRy:** Установка бита **y** (**y**=0..15)
0: Не изменяется
1: Ставится

9.2.6. Регистр сброса битов порта (GPIOx_BRR) (x=A..G)

Смещение адреса: 0x14

По сбросу: 0x0000 0000

Доступен только словом.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

— Биты 31:16 Резерв, не трогать.
— Биты 15:0 **BRy:** Сброс бита **y** (**y**=0..15)
0: Не изменяется
1: Ставится

9.2.7. Регистр блокировки конфигурации порта (GPIOx_LCRR) (x=A..G)

Используется для блокировки конфигурации портов GPIO по битам `LCKR[15:0]` после завершения последовательности записи LOCK в бит 16 (`LCKK`). Во время последовательности записи значения замка `LCKR[15:0]` не должны изменяться. После этого порты невозможно конфигурировать вплоть до следующего сброса. Каждый бит замка блокирует соответствующие 4 бита регистров управления (`CRL`, `CRH`).

Смещение адреса: 0x18

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:17 Резерв, не трогать.
— Бит 16 **LCKK:** Ключ замка.
0: Конфигурация портов возможна
1: Ключ активен. GPIOx_LCKR закрыт до следующего сброса.
Последовательность записи ключа LOCK:
Запись 1
Запись 0
Запись 1
Чтение 0
Чтение 1 (необязательно, но подтверждает активность ключа)
NB: Во время последовательности LOCK значения LCK[15:0] не могут изменяться. Ошибка прерывает последовательность.
— Биты 15:0 **LCKy:** Блокировка порта **y** (**y**=0..15)
Эти биты можно писать только при LCKK=0.
0: Порт **y** не блокирован
1: Порт **y** блокирован

9.3. Альтернативный IO и конфигурация отладки (AFIO)

Удобству программного перенаправления сигналов периферии на разные ножки разных корпусов служит регистр AFIO_MAPR.

9.3.1. Ножки OSC32_IN/OSC32_OUT как GPIO PC14/PC15

Ножки выключенного генератора LSE OSC32_IN и OSC32_OUT можно использовать как GPIO PC14 и PC15, но LSE приоритетнее GPIO.

NB: PC14/PC15 GPIO не работает при выключенном 1.8V домене (режим standby) или при питании резервного домена от V_{BAT} (V_{DD} нету). В этом случает ножки IO ставятся в аналоговый режим. См. *Секцию 5.1.2*.

9.3.2. Ножки OSC_IN/OSC_OUT как GPIO PD0/PD1

Ножки выключенного генератора HSE OSC_IN и OSC_OUT можно использовать как GPIO PD0 и PD1 битом PD01_REMAP в регистре AFIO_MAPR. Это доступно только в корпусах с 36, 48 и 64 ножками (в корпусах со 100 ножками PD0 и PD1 и так есть).

NB: Внешние прерывания/события не перенаправляются. Ножки PD0 и PD1 нельзя использоваться внешних прерываний/событий в корпусах с 36, 48 и 64 ножками.

9.3.3. Перенаправление CAN1

Сигналы CAN можно направить в порты A, B или D, см. Таблицу 34. Порт D не доступен в корпусах с 36, 48 и 64 ножками.

Таблица 34. Перенаправление сигналов CAN1

Альтернативная функция(1)	CAN_REMAP[1:0] = “00”	CAN_REMAP[1:0] = “10”(2)	CAN_REMAP[1:0] = “11”(3)
CAN1_RX или CAN_RX	PA11	PB8	PD0
CAN1_TX или CAN_RX	PA12	PB9	PD1

1. CAN1_RX и CAN1_TX в сетевых устройствах; CAN_RX и CAN_TX в других устройствах с одним CAN.
2. Невозможно в 36-контактном корпусе
3. Это можно только в 100- и 144-ногих корпусах, когда PD0 и PD1 не направлены на OSC-IN и OSC-OUT.

9.3.4. Перенаправление CAN2

CAN2 есть только в сетевых устройствах, см. Таблицу 35.

Таблица 35. Перенаправление сигналов CAN2

Альтернативная функция	CAN2_REMAP = “0”	CAN2_REMAP = “1”
CAN2_RX	PB12	PB5
CAN2_TX	PB13	PB13

9.3.5. Перенаправление JTAG/SWD

Таблица 36. Сигналы интерфейса отладки

Альтернативная функция	Порт GPIO
JTMS / SWDIO	PA13
JTCK / SWCLK	PA14
JTDI	PA15
JTDO / TRACESWO	PB3

Альтернативная функция	Порт GPIO
NJTRST	PB4
TRACECK	PE2
TRACED0	PE3
TRACED1	PE4
TRACED2	PE5
TRACED3	PE6

Перенаправить GPIO при отладке можно битами `SWJ_CFG[1:0]` в регистре `AFIO_MAPR`.

Таблица 37. Сигналы интерфейса отладки

SWJ _CFG [2:0]	Доступные порты отладки	Назначение ножек SWJ I/O pin				
		PA13 / JTMS/ SWDIO	PA14 / JTCK/S WCLK	PA15 / JTDI	PB3 / JTDO/ TRACE SWO	PB4/ NJTRST
000	Full SWJ (JTAG-DP + SW-DP) (По сбросу)	x	x	x	x	x
001	Full SWJ (JTAG-DP + SW-DP) но без NJTRST	x	x	x	x	Свободен
010	JTAG-DP Выключен и SW-DP Включён	x	x	Свободен	Свободен(1)	Свободен
100	JTAG-DP Выключен и SW-DP Выключен	Свободен	Свободен	Свободен	Свободен	Свободен
Другие	Запрещено	—	—	—	—	—

1. Освобождаются только при не асинхронном слежении.

9.3.6. Перенаправление ADC

Таблица 38. Внешний запуск преобразования ADC1(1)

Альтернативная функция	ADC1_ETRGINJ_REMAP = 0	ADC1_ETRGINJ_REMAP = 1
Внешний запуск ADC1	EXTI15	TIM8_CH4

1. Возможно только в устройствах высокой и XL-плотности.

Таблица 39. Внешний запуск регулярного преобразования ADC1(1)

Альтернативная функция	ADC1_ETRGREG_REMAP = 0	ADC1_ETRGREG_REMAP = 1
Внешний запуск ADC1	EXTI11	TIM8_TRGO

1. Возможно только в устройствах высокой и XL-плотности.

Таблица 40. Внешний запуск преобразования ADC2(1)

Альтернативная функция	ADC2_ETRGINJ_REMAP = 0	ADC2_ETRGINJ_REMAP = 1
Внешний запуск ADC1	EXTI15	TIM8_CH4

1. Возможно только в устройствах высокой и XL-плотности.

Таблица 41. Внешний запуск регулярного преобразования ADC1(1)

Альтернативная функция	ADC2_ETRGREG_REMAP = 0	ADC2_ETRGREG_REMAP = 1
Внешний триггер ADC1	EXTI11	TIM8_TRGO

1. Возможно только в устройствах высокой и XL-плотности.

9.3.7. Перенаправление таймеров

Каналы 1 - 4 Таймера 4 можно направить с порта B на порт D. См. описание регистра AFIO_MAPR.

Таблица 42. Перенаправление TIM5(1)

Альтернативная функция	TIM5CH4_IREMAP = 0	TIM5CH4_IREMAP = 1
TIM5_CH4	PA3	LSI подключён к входу TIM5_CH4 для калибровки.

1. Возможно только в устройствах высокой и XL-плотности.

Таблица 43. Перенаправление TIM4

Альтернативная функция	TIM4_REMAP = 0	TIM4_REMAP = 1(1)
TIM4_CH1	PB6	PD12
TIM4_CH2	PB7	PD13
TIM4_CH3	PB8	PD14
TIM4_CH4	PB9	PD15

1. Возможно только в 100- и 144- контактных корпусах.

Таблица 44. Перенаправление TIM3

Альтернативная функция	TIM3_REMAP[1:0] = “00” (нет)	TIM3_REMAP[1:0] = “10” (частичное)	TIM3_REMAP[1:0] = “11” (полное)(1)
TIM3_CH1	PA6	PB4	PC6
TIM3_CH2	PA7	PB5	PC7
TIM3_CH3	PB0		PC8
TIM3_CH4	PB1		PC9

1. Возможно только в 64-, 100- и 144- контактных корпусах.

Таблица 45. Перенаправление TIM2

Альтернативная функция	TIM2_REMAP [1:0] = “00” (нет)	TIM2_REMAP [1:0] = “01” (частичное)	TIM2_REMAP [1:0] = “10” (частичное)(1)	TIM2_REMAP [1:0] = “11” (полное)(1)
TIM2_CH1_ETR(2)	PA0	PA15	PA0	PA15
TIM2_CH2	PA1	PB3	PA1	PB3
TIM2_CH3	PA2		PB10	
TIM2_CH4	PA3		PB11	

1. Невозможно 36- контактных корпусах.
2. TIM_CH1 и TIM_ETR стоят на одной ножке, но не могут использоваться одновременно (отсюда обозначение TIM2_CH1_ETR).

Таблица 46. Перенаправление TIM1

Альтернативная функция	TIM1_REMAP[1:0] = “00” (нет)	TIM1_REMAP[1:0] = “01” (частичное)	TIM1_REMAP[1:0] = “11” (полное)(1)
TIM1_ETR	PA12		PE7
TIM1_CH1	PA8		PE9
TIM1_CH2	PA9		PE11
TIM1_CH3	PA10		PE13
TIM1_CH4	PA11		PE14
TIM1_BKIN	PB12(2)	PA6	PE15
TIM1_CH1N	PB13	PA7	PE8
TIM1_CH2N	PB14(2)	PB0	PE10
TIM1_CH3N	PB15(2)	PB1	PE12

(1) Возможно только в 100- и 144- контактных корпусах.
(2) Невозможно 36- контактных корпусах.

Таблица 47. Перенаправление TIM9(1)

Альтернативная функция	TIM9_REMAP = 0	TIM9_REMAP = 1
TIM9_CH1	PA2	PE5
TIM9_CH2	PA3	PE6

1. См. *Секцию 9.4.7*.

Таблица 48. Перенаправление TIM10(1)

Альтернативная функция	TIM10_REMAP = 0	TIM10_REMAP = 1
TIM11_CH1	PB8	PF6

1. См. *Секцию 9.4.7*.

Таблица 49. Перенаправление TIM11(1)

Альтернативная функция	TIM11_REMAP = 0	TIM11_REMAP = 1
TIM10_CH1	PB9	PF7

1. См. *Секцию 9.4.7*.

Таблица 50. Перенаправление TIM13(1)

Альтернативная функция	TIM13_REMAP = 0	TIM13_REMAP = 1
TIM13_CH1	PA6	PF8

1. См. *Секцию 9.4.7*.

Таблица 51. Перенаправление TIM14(1)

Альтернативная функция	TIM14_REMAP = 0	TIM14_REMAP = 1
TIM14_CH1	PA7	PF9

1. См. *Секцию 9.4.7*.

9.3.8. Перенаправление USART

См. описание регистра AFIO_MAPR.

Таблица 52. Перенаправление USART3

Альтернативная функция	USART3_REMAP[1:0] = “00” (нет)	USART3_REMAP[1:0] = “01” (частичное)(1)	USART3_REMAP[1:0] = “11” (полное)(2)
USART3_TX	PB10	PC10	PD8
USART3_RX	PB11	PC11	PD9
USART3_CK	PB12	PC12	PD10
USART3_CTS	PB13		PD11
USART3_RTS	PB14		PD12

(1) Возможно только в 640, 100- и 144- контактных корпусах.
(2) Невозможно 100- и 144- контактных корпусах.

Таблица 53. Перенаправление USART2

Альтернативная функция	USART2_REMAP = 0	USART2_REMAP = 1(1)
USART2_CTS	PA0	PD3
USART2_RTS	PA1	PD4
USART2_TX	PA2	PD5
USART2_RX	PA3	PD6
USART2_CK	PA4	PD7

(1) Возможно только в 100- и 144- контактных корпусах.

Таблица 54. Перенаправление USART1

Альтернативная функция	USART1_REMAP = 0	USART1_REMAP = 1
USART1_TX	PA9	PB6
USART1_RX	PA10	PB7

9.3.9. Перенаправление I2C1

См. описание регистра AFIO_MAPR.

Таблица 55. Перенаправление I2C1(1)

Альтернативная функция	I2C1_REMAP = 0	I2C1_REMAP = 1(1)
I2C1_SCL	PB6	PB8
I2C1_SDA	PB7	PB9

(1) Возможно только в 36- контактных корпусах.

9.3.10. Перенаправление SPI1

См. описание регистра AFIO_MAPR.

Таблица 56. Перенаправление SPI1

Альтернативная функция	SPI1_REMAP = 0	SPI1_REMAP = 1
SPI1_NSS	PA4	PA15
SPI1_SCK	PA5	PB3
SPI1_MISO	PA6	PB4
SPI1_MOSI	PA7	PB5

9.3.11. Перенаправление SPI3/I2S3

См. описание регистра AFIO_MAPR.

Таблица 57. Перенаправление SPI3/I2S3

Альтернативная функция	SPI3_REMAP = 0	SPI3_REMAP = 1
SPI3_NSS / I2S3_WS	PA15	PA4
SPI3_SCK / I2S3_CK	PB3	PC10
SPI3_MISO	PB4	PC11
SPI3_MOSI / I2S3_SD	PB5	PC12

9.3.12. Перенаправление Ethernet

См. описание регистра AFIO_MAPR.

Таблица 58. Перенаправление ETH

Альтернативная функция	ETH_REMAP = 0	ETH_REMAP = 1
RX_DV-CRS_DV	PA7	PD8
RXD0	PC4	PD9
RXD1	PC5	PD10
RXD2	PB0	PD11
RXD3	PB1	PD12

9.4. Регистры AFIO

NB: Для работы с регистрами AFIO_EVCR, AFIO_MAPR и AFIO_EXTICRX нужно сначала разрешить тактирование AFIO. См. *Секцию 7.3.7*. Доступен словами.

9.4.1. Регистр управления событием (AFIO_EVCR)

Смещение адреса: 0x00

По сбросу: 0x0000 0000

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

— Биты 31:8 Резерв, не трогать.

— Бит 7 **EVOE:** Разрешение вывода события.
Изменяется программно. подключение события EVENTOUT задаётся битами PORT[2:0] и PIN[3:0].

— Биты 6:4 **PORT[2:0]**: Выбор порта для EVENTOUT
Изменяется программно.
000: PA
001: PB
010: PC
011: PD
100: PE

— Биты 3:0 **PIN[3:0]**: Выбор ножки (x=A..E)

Изменяется программно.

0000: Px0

0001: Px1

0010: Px2

0011: Px3

…

1111: Px15

9.4.2. Регистр картирования AF и отладки (AFIO_MAPR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000

Устройства разной плотности

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					SWJ_CFG[2:0]			Reserved			ADC2_ETRGREG_REMAP	ADC2_ETRGINJ_REMAP	ADC1_ETRGREG_REMAP	ADC1_ETRGINJ_REMAP	TIM5CH4_IREMAP
					w	w	w				rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_REMAP	CAN_REMAP[1:0]		TIM4_REMAP	TIM3_REMAP[1:0]		TIM2_REMAP[1:0]		TIM1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:27 Резерв, не трогать.

— Биты 26:24 **SWJ_CFG[2:0]:** Конфигурация последовательной линии JTAG

Только запись. SWJ (Serial Wire JTAG) поддерживает доступ JTAG или SWD порта отладки. Состояние после сброса - SWJ ON без слежения. Так можно включать режим JTAG или SW посылкой определённой последовательности на ножку JTMS / JTCK.

000: Полный SWJ (JTAG-DP + SW-DP): Состояние Сброса

001: Полный SWJ (JTAG-DP + SW-DP) без NJTRST

010: JTAG-DP Выключен и SW-DP Включён

100: JTAG-DP Выключен и SW-DP Выключен

Другое: Без эффекта

— Биты 23:21 Резерв, не трогать.

— Бит 20 **ADC2_ETRGREG_REMAP**: Перенаправление триггера регулярного запуска ADC2

Изменяется программно. Если сброшен, то регулярный запуск ADC2 подключён к EXTI11, иначе внешнее событие запуска подключено к TIM8_TRGO.

— Бит 19 **ADC2_ETRGINJ_REMAP:** Перенаправление триггера однократного запуска ADC2

Изменяется программно. Если сброшен, то однократный запуск ADC2 подключён к EXTI15, внешнее событие запуска подключено к TIM8_Channel4.

— Бит 18 **ADC1_ETRGREG_REMAP**: Перенаправление триггера регулярного запуска ADC1

Изменяется программно. Если сброшен, то регулярный запуск ADC2 подключён к EXTI11, иначе внешнее событие запуска подключено к TIM8_TRGO.

— Бит 17 **ADC1_ETRGINJ_REMAP:** Перенаправление триггера однократного запуска ADC1

Изменяется программно. Если сброшен, то однократный запуск ADC1 подключён к EXTI15, иначе внешнее событие запуска подключено к TIM8_Channel4.

— Бит 16 **TIM5CH4_IREMAP**: Перенаправление TIM5 канал 4

Изменяется программно. Если сброшен, то TIM5_CH4 подключён к PA3, иначе к TIM5_CH4 подключён LSI для калибровки.

NB: Этот бит доступен только в устройствах высокой плотности.

— Бит 15 **PD01_REMAP:** Перенаправление PortD0/PortD1 на OSC_IN/OSC_OUT

Изменяется программно. Если не используется генератор HSE (работает внутренний 8 MHz RC) PD0 и PD1 можно направить на OSC_IN и OSC_OUT. Доступно в 36-, 48- и 64-контактных корпусах (PD0 и PD1 существуют в 100- и 144-контактных корпусах).

0: PD0 и PD1 не перенаправляются

1: PD0 направляется на OSC_IN, PD1 направляется на OSC_OUT

— Биты 14:13 **CAN_REMAP[1:0]:** Перенаправление CAN

Изменяется программно. Работает в устройствах с одним CAN интерфейсом.

00: CAN_RX направляется на PA11, CAN_TX направляется на PA12

01: Не используется

10: CAN_RX направляется на PB8, CAN_TX направляется на PB9 (не в 36-контактном корпусе)

11: CAN_RX направляется на PD0, CAN_TX направляется на PD1

— Бит 12 **TIM4_REMAP:** Перенаправление TIM4

Изменяется программно.

0: Не переносится (TIM4_CH1/PB6, TIM4_CH2/PB7, TIM4_CH3/PB8, TIM4_CH4/PB9)

1: Полный перенос (TIM4_CH1/PD12, TIM4_CH2/PD13, TIM4_CH3/PD14, TIM4_CH4/PD15)

NB: TIM4_ETR на PE0 не переносится.

— Биты 11:10 **TIM3_REMAP[1:0]**: Перенаправление TIM3

Изменяется программно. Переносит TIM3 каналы 1 до 4 на GPIO.

00: Не переносится (CH1/PA6, CH2/PA7, CH3/PB0, CH4/PB1)

01: Не используется

10: Частичный перенос (CH1/PB4, CH2/PB5, CH3/PB0, CH4/PB1)

11: Полный перенос (CH1/PC6, CH2/PC7, CH3/PC8, CH4/PC9)

NB: TIM3_ETR на PE0 не уходит.

— Биты 9:8 **TIM2_REMAP[1:0]**: Перенаправление TIM2

Изменяется программно. Переносит TIM2 каналы 1 до 4 и внешний запуск (ETR) на GPIO.

00: Не переносится (CH1/ETR/PA0, CH2/PA1, CH3/PA2, CH4/PA3)

01: Частичный перенос (CH1/ETR/PA15, CH2/PB3, CH3/PA2, CH4/PA3)

10: Частичный перенос (CH1/ETR/PA0, CH2/PA1, CH3/PB10, CH4/PB11)

11: Полный перенос (CH1/ETR/PA15, CH2/PB3, CH3/PB10, CH4/PB11)

— Биты 7:6 **TIM1_REMAP[1:0]:** Перенаправление TIM1

Изменяется программно. Переносит TIM1 каналы 1 до 4, 1N до 3N, внешний запуск (ETR) и вход Break (BKIN) на GPIO.

00: Не переносится (ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PB12, CH1N/PB13, CH2N/PB14, CH3N/PB15)

01: Частичный перенос (ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PA6, CH1N/PA7, CH2N/PB0, CH3N/PB1)

10: Не используется

11: Полный перенос (ETR/PE7, CH1/PE9, CH2/PE11, CH3/PE13, CH4/PE14, BKIN/PE15, CH1N/PE8, CH2N/PE10, CH3N/PE12)

— Биты 5:4 **USART3_REMAP[1:0]**: Перенаправление USART3

Изменяется программно. Переносит USART3 CTS, RTS,CK,TX и RX на GPIO.

00: Не переносится (TX/PB10, RX/PB11, CK/PB12, CTS/PB13, RTS/PB14)

01: Частичный перенос (TX/PC10, RX/PC11, CK/PC12, CTS/PB13, RTS/PB14)

10: Не используется

11: Полный перенос (TX/PD8, RX/PD9, CK/PD10, CTS/PD11, RTS/PD12)

— Бит 3 **USART2_REMAP:** Перенаправление USART2

Изменяется программно. Переносит USART2 CTS, RTS,CK,TX и RX на GPIO.

0: Не переносится (CTS/PA0, RTS/PA1, TX/PA2, RX/PA3, CK/PA4)

1: Переносится (CTS/PD3, RTS/PD4, TX/PD5, RX/PD6, CK/PD7)

— Бит 2 **USART1_REMAP:** Перенаправление USART1

Изменяется программно. Переносит USART1 TX и RX на GPIO.

0: Не переносится (TX/PA9, RX/PA10)

1: Переносится (TX/PB6, RX/PB7)

— Бит 1 **I2C1_REMAP:** Перенаправление I2C1

Изменяется программно. Переносит I2C1 SCL и SDA на GPIO.

0: Не переносится (SCL/PB6, SDA/PB7)

1: Переносится (SCL/PB8, SDA/PB9)

— Бит 0 **SPI1_REMAP:** Перенаправление SPI1

Изменяется программно. Переносит SPI1 NSS, SCK, MISO, MOSI на GPIO.

0: Не переносится (NSS/PA4, SCK/PA5, MISO/PA6, MOSI/PA7)

1: Переносится (NSS/PA15, SCK/PB3, MISO/PB4, MOSI/PB5)

Сетевые устройства

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PTP_PPS_REMAP	TIM2ITR1_IREMAP	SPI3_REMAP	Res.	SWJ_CFG[2:0]			MII_RMII_SEL	CAN2_REMAP	ETH_REMAP	Reserved				TIM5CH4_IREMAP
	rw	rw	rw		w	w	w	rw	rw	rw					rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_REMAP	CAN1_REMAP[1:0]		TIM4_REMAP	TIM3_REMAP[1:0]		TIM2_REMAP[1:0]		TIM1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 Резерв, не трогать.

— Бит 30 **PTP_PPS_REMAP:** Перенаправление Ethernet PTP PPS

Изменяется программно. Переносит Ethernet MAC PPS_PTS на PB5.

0: PTP_PPS не уходит на PB5.

1: PTP_PPS уходит на PB5.

NB: Только в сетевых устройствах.

— Бит 29 **TIM2ITR1_IREMAP:** Перенаправление внутреннего триггера 1 TIM2

Изменяется программно.

0: Внутренне соединяет TIM2_ITR1 на Ethernet PTP для калибровки.

1: Внутренне соединяет USB OTG SOF (Start of Frame) на TIM2_ITR1 для калибровки.

NB: Только в сетевых устройствах.

— Бит 28 **SPI3_REMAP:** Перенаправление SPI3/I2S3

Изменяется программно. Переносит SPI3_NSS/I2S3_WS, SPI3_SCK/I2S3_CK, SPI3_MISO, SPI3_MOSI/I2S3_SD на GPIO.

0: Не переносит (SPI_NSS-I2S3_WS/PA15, SPI3_SCK-I2S3_CK/PB3, SPI3_MISO/PB4, SPI3_MOSI-I2S3_SD/PB5)

1: Переносит (SPI3_NSS-I2S3_WS/PA4, SPI3_SCK-I2S3_CK/PC10, SPI3_MISO/PC11, SPI3_MOSI-I2S3_SD/PC12)

NB: Только в сетевых устройствах.

— Бит 27 Резерв, не трогать.

— Биты 26:24 **SWJ_CFG[2:0]:** Конфигурация последовательной линии JTAG

Только запись. SWJ (Serial Wire JTAG) поддерживает доступ JTAG или SWD порта отладки. Состояние после сброса - SWJ ON без слежения. Так можно включать режим JTAG или SW посылкой определённой последовательности на ножку JTMS / JTCK.

000: Полный SWJ (JTAG-DP + SW-DP): Состояние Сброса

001: Полный SWJ (JTAG-DP + SW-DP) без NJTRST

010: JTAG-DP Выключен и SW-DP Включён

100: JTAG-DP Выключен и SW-DP Выключен

Другое: Без эффекта

— Бит 23 **MII_RMII_SEL:** Выбор MII или RMII

Изменяется программно.

0: Подключает Ethernet MAC к MII PHY

1: Подключает Ethernet MAC к RMII PHY

NB: Только в сетевых устройствах.

— Бит 22 **CAN2_REMAP:** Перенаправление CAN2 I/O

Изменяется программно. Перенаправляет ножки CAN2_TX и CAN2_RX.

0: Не переносит (CAN2_RX/PB12, CAN2_TX/PB13)

1: Переносит (CAN2_RX/PB5, CAN2_TX/PB6)

NB: Только в сетевых устройствах.

— Бит 21 **ETH_REMAP:** Перенаправление EthernetMAC I/O

Изменяется программно. Подключает Ethernet MAC I/O к PHY.

0: Не переносит (RX_DV-CRS_DV/PA7, RXD0/PC4, RXD1/PC5, RXD2/PB0, RXD3/PB1)

1: Переносит (RX_DV-CRS_DV/PD8, RXD0/PD9, RXD1/PD10, RXD2/PD11, RXD3/PD12)

NB: Только в сетевых устройствах.

— Биты 20:17 Резерв, не трогать.

— Бит 16 **TIM5CH4_IREMAP**: Перенаправление TIM5 канал 4

Изменяется программно. Если сброшен, то TIM5_CH4 подключён к PA3, иначе к TIM5_CH4 подключён LSI для калибровки.

— Бит 15 **PD01_REMAP:** Перенаправление PortD0/PortD1 на OSC_IN/OSC_OUT

Изменяется программно. Если не используется генератор HSE (работает внутренний 8 MHz RC) PD0 и PD1 можно направить на OSC_IN и OSC_OUT. Доступно в 36-, 48- и 64-контактных корпусах (PD0 и PD1 существуют в 100- и 144-контактных корпусах).

0: PD0 и PD1 не перенаправляются

1: PD0 направляется на OSC_IN, PD1 направляется на OSC_OUT

— Биты 14:13 **CAN1_REMAP[1:0]:** Перенаправление CAN1

Изменяется программно.

00: CAN1_RX направляется на PA11, CAN_TX направляется на PA12

01: Не используется

10: CAN1_RX направляется на PB8, CAN_TX направляется на PB9 (не в 36-контактном корпусе)

11: CAN1_RX направляется на PD0, CAN_TX направляется на PD1

— Бит 12 **TIM4_REMAP:** Перенаправление TIM4

Изменяется программно.

0: Не переносится (TIM4_CH1/PB6, TIM4_CH2/PB7, TIM4_CH3/PB8, TIM4_CH4/PB9)

1: Полный перенос (TIM4_CH1/PD12, TIM4_CH2/PD13, TIM4_CH3/PD14, TIM4_CH4/PD15)

NB: TIM4_ETR на PE0 не переносится.

— Биты 11:10 **TIM3_REMAP[1:0]**: Перенаправление TIM3

Изменяется программно. Переносит TIM3 каналы 1 до 4 на GPIO.

00: Не переносится (CH1/PA6, CH2/PA7, CH3/PB0, CH4/PB1)

01: Не используется

10: Частичный перенос (CH1/PB4, CH2/PB5, CH3/PB0, CH4/PB1)

11: Полный перенос (CH1/PC6, CH2/PC7, CH3/PC8, CH4/PC9)

NB: TIM3_ETR на PE0 не уходит.

— Биты 9:8 **TIM2_REMAP[1:0]**: Перенаправление TIM2

Изменяется программно. Переносит TIM2 каналы 1 до 4 и внешний запуск (ETR) на GPIO.

00: Не переносится (CH1/ETR/PA0, CH2/PA1, CH3/PA2, CH4/PA3)

01: Частичный перенос (CH1/ETR/PA15, CH2/PB3, CH3/PA2, CH4/PA3)

10: Частичный перенос (CH1/ETR/PA0, CH2/PA1, CH3/PB10, CH4/PB11)

11: Полный перенос (CH1/ETR/PA15, CH2/PB3, CH3/PB10, CH4/PB11)

— Биты 7:6 **TIM1_REMAP[1:0]:** Перенаправление TIM1

Изменяется программно. Переносит TIM1 каналы 1 до 4, 1N до 3N, внешний запуск (ETR) и вход Break (BKIN) на GPIO.

00: Не переносится (ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PB12, CH1N/PB13, CH2N/PB14, CH3N/PB15)

01: Частичный перенос (ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PA6, CH1N/PA7, CH2N/PB0, CH3N/PB1)

10: Не используется

11: Полный перенос (ETR/PE7, CH1/PE9, CH2/PE11, CH3/PE13, CH4/PE14, BKIN/PE15, CH1N/PE8, CH2N/PE10, CH3N/PE12)

— Биты 5:4 **USART3_REMAP[1:0]**: Перенаправление USART3

Изменяется программно. Переносит USART3 CTS, RTS,CK,TX и RX на GPIO.

00: Не переносится (TX/PB10, RX/PB11, CK/PB12, CTS/PB13, RTS/PB14)

01: Частичный перенос (TX/PC10, RX/PC11, CK/PC12, CTS/PB13, RTS/PB14)

10: Не используется

11: Полный перенос (TX/PD8, RX/PD9, CK/PD10, CTS/PD11, RTS/PD12)

— Бит 3 **USART2_REMAP:** Перенаправление USART2

Изменяется программно. Переносит USART2 CTS, RTS,CK,TX и RX на GPIO.

0: Не переносится (CTS/PA0, RTS/PA1, TX/PA2, RX/PA3, CK/PA4)

1: Переносится (CTS/PD3, RTS/PD4, TX/PD5, RX/PD6, CK/PD7)

— Бит 2 **USART1_REMAP:** Перенаправление USART1

Изменяется программно. Переносит USART1 TX и RX на GPIO.

0: Не переносится (TX/PA9, RX/PA10)

1: Переносится (TX/PB6, RX/PB7)

— Бит 1 **I2C1_REMAP:** Перенаправление I2C1
Изменяется программно. Переносит I2C1 SCL и SDA на GPIO.
0: Не переносится (SCL/PB6, SDA/PB7)
1: Переносится (SCL/PB8, SDA/PB9)

— Бит 0 **SPI1_REMAP:** Перенаправление SPI1
Изменяется программно. Переносит SPI1 NSS, SCK, MISO, MOSI на GPIO.
0: Не переносится (NSS/PA4, SCK/PA5, MISO/PA6, MOSI/PA7)
1: Переносится (NSS/PA15, SCK/PB3, MISO/PB4, MOSI/PB5)

9.4.3. Регистр 1 конфигурации внешнего прерывания (AFIO_EXTICR1)

Смещение адреса: 0x08

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **EXTIx[3:0]:** Конфигурация EXTI x (x= 0 до 3)
Программно выбирает источник внешнего прерывания EXTIx. См. *Секцию 10.2.5*.
0000: PA[x]
0001: PB[x]
0010: PC[x]
0011: PD[x]
0100: PE[x]
0101: PF[x]
0110: PG[x]

9.4.4. Регистр 2 конфигурации внешнего прерывания (AFIO_EXTICR2)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **EXTIx[3:0]:** Конфигурация EXTI x (x= 4 to 7)
Программно выбирает источник внешнего прерывания EXTIx.
0000: PA[x]
0001: PB[x]
0010: PC[x]
0011: PD[x]
0100: PE[x]
0101: PF[x]
0110: PG[x]

9.4.5. Регистр 3 конфигурации внешнего прерывания (AFIO_EXTICR3)

Смещение адреса: 0x10

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **EXTIx[3:0]:** Конфигурация EXTI x (x= 8 to 11)

Программно выбирает источник внешнего прерывания EXTIx.

0000: PA[x]
0001: PB[x]
0010: PC[x]
0011: PD[x]
0100: PE[x]
0101: PF[x]
0110: PG[x]

9.4.6. Регистр 4 конфигурации внешнего прерывания (AFIO_EXTICR4)

Смещение адреса: 0x14

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **EXTIx[3:0]:** Конфигурация EXTI x (x= 12 to 15)

Программно выбирает источник внешнего прерывания EXTIx.

0000: PA[x]
0001: PB[x]
0010: PC[x]
0011: PD[x]
0100: PE[x]
0101: PF[x]
0110: PG[x]

9.4.7. Регистр 2 картирования AF и отладки (AFIO_MAPR2)

Смещение адреса: 0x1C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					FSMC_NADV	TIM14_REMAP	TIM13_REMAP	TIM11_REMAP	TIM10_REMAP	TIM9_REMAP	Reserved				
					rw	rw	rw	rw	rw	rw					

— Биты 31:11 Резерв, не трогать.

— Бит 10 **FSMC_NADV:** Подключение NADV

Программно управляет необязательным сигналом FSMC_NADV.

0: NADV подключён к выходу (по умолчанию)
1: NADV не подключён. Ножка I/O свободна.

— Бит 9 **TIM14_REMAP:** Перенаправление TIM14

Программно посылает TIM14_CH1 на GPIO

0: Не послал (PA7)
1: Послал (PF9)

— Бит 8 **TIM13_REMAP:** Перенаправление TIM13
Программно посылает TIM13_CH1 на GPIO
0: Не послал (PA6)
1: Послал (PF8)

— Бит 7 **TIM11_REMAP:** Перенаправление TIM11
Программно посылает TIM11_CH1 на GPIO
0: Не послал (PB9)
1: Послал (PF7)

— Бит 6 **TIM10_REMAP:** Перенаправление TIM10
Программно посылает TIM10_CH1 на GPIO
0: Не послал (PB8)
1: Послал (PF6)

— Бит 5 **TIM9_REMAP:** Перенаправление TIM9
Программно посылает TIM9_CH1 и TIM9_CH2 на GPIO
0: Не послал (TIM9_CH1 на PA2 и TIM9_CH2 на PA3)
1: Послал (TIM9_CH1 на PE5 и TIM9_CH2 на PE6)

— Биты 4:0 Резерв, не трогать.

9.5. Карта регистров GPIO и AFIO

Таблица 59. GPIO

0x00	GPIOx_CRL	CNF7 [1:0]		MODE7 [1:0]		CNF6 [1:0]		MODE6 [1:0]		CNF5 [1:0]		MODE5 [1:0]		CNF4 [1:0]		MODE4 [1:0]		CNF3 [1:0]		MODE3 [1:0]		CNF2 [1:0]		MODE2 [1:0]		CNF1 [1:0]		MODE1 [1:0]		CNF0 [1:0]		MODE0 [1:0]	
	Reset value	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0x04	GPIOx_CRH	CNF15 [1:0]		MODE15 [1:0]		CNF14 [1:0]		MODE14 [1:0]		CNF13 [1:0]		MODE13 [1:0]		CNF12 [1:0]		MODE12 [1:0]		CNF11 [1:0]		MODE11 [1:0]		CNF10 [1:0]		MODE10 [1:0]		CNF9 [1:0]		MODE9 [1:0]		CNF8 [1:0]		MODE8 [1:0]	
	Reset value	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0x08	GPIOx_IDR	Reserved																IDRy															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOx_ODR	Reserved																ODRy															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPIOx_BSRR	BR[15:0]																BSR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	GPIOx_BRR	Reserved																BR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	GPIOx_LCKR	Reserved															LCKK	LCK[15:0]															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Таблица 60. Карта регистров AFIO

0x00	AFIO_EVCR	Reserved																									EVOE	PORT[2:0]			PIN[3:0]		
	Reset value																										0	0	0	0	0	0	0
0x04	AFIO_MAPR low-, medium-, high- and XL-density devices	Reserved					SWJ_CFG[2]	SWJ_CFG[1]	SWJ_CFG[0]	Reserved			ADC2_ETRGREG_REMAP	ADC2_ETRGINJ_REMAP	ADC1_ETRGREG_REMAP	ADC1_ETRGINJ_REMAP	TIM5CH4_IREMAP	PD01_REMAP	CAN1_REMAP[1]	CAN1_REMAP[0]	TIM4_REMPAP	TIM3_REMPAP[1]	TIM3_REMPAP[0]	TIM2_REMPAP[1]	TIM2_REMPAP[0]	TIM1_REMPAP[1]	TIM1_REMPAP[0]	USART3_REMAP[1]	USART3_REMAP[0]	USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP
	Reset value						0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	AFIO_MAPR connectivity line devices	Reserved	PTP_PPS_REMAP	TIM2ITR1_IREMAP	SPI3_REMAP	Reserved	SWJ_CFG[2]	SWJ_CFG[1]	SWJ_CFG[0]	MII_RMII_SEL	CAN2_REMAP	ETH_REMAP	Reserved				TIM5CH4_IREMAP	PD01_REMAP	CAN1_REMAP[1]	CAN1_REMAP[0]	TIM4_REMPAP	TIM3_REMPAP[1]	TIM3_REMPAP[0]	TIM2_REMPAP[1]	TIM2_REMPAP[0]	TIM1_REMPAP[1]	TIM1_REMPAP[0]	USART3_REMAP[1]	USART3_REMAP[0]	USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP
	Reset value		0	0	0		0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	AFIO_MAPR	Reserved					SWJ_CFG[2:0]			Reserved							TIM5CH4_IREMAP	PD01_REMAP	Reserved		TIM4_REMAP	TIM3_REMAP[1:0]		TIM2_REMAP[1:0]		TIM1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP
	Reset value						0	0	0								0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	AFIO_EXTICR1	Reserved																EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	AFIO_EXTICR2	Reserved																EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	AFIO_EXTICR3	Reserved																EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	AFIO_EXTICR4	Reserved																EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	AFIO_MAPR2	Reserved																					FSMC_NADV	TIM14_REMAP	TIM13_REMAP	TIM11_REMAP	TIM10_REMAP	TIM9_REMAP	Reserved				
	Reset value																						0	0	0	0	0	0					
0x1C	AFIO_MAPR2	Reserved																		MISC_REMAP	TIM12_REMAP	TIM67_DAC_DMA_REMAP	FSMC_NADV	TIM14_REMAP	TIM13_REMAP	Res.			TIM1_DMA_REMAP	CEC_REMAP	TIM17_REMAP	TIM16_REMAP	TIM15_REMAP
	Reset value																			0	0	0	0	0	0				0	0	0	0	0

10. Прерывания и события

10.1. Контроллер вложенных прерываний (NVIC)

Свойства

- 68 прерываний (не включая 16 линий Cortex®-M3)
- 16 программируемых уровней приоритета (4 бита)
- Низкая задержка
- Управление питанием
- Регистры управления

NVIC и ядро тесно связаны, отсюда низкая задержка и обработка запоздавших прерываний.
Все прерывания и исключения в системе обрабатываются именно NVIC.

10.1.1. Регистр калибровки SysTick

Содержит число 9000, что даёт опорную базу 1 ms с SysTick, установленным на 9 MHz (HCLK/8).

10.1.2. Векторы прерываний и исключений

Таблица 61. Сетевые устройства

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
—	—	—	—	Резерв	0x0000_0000
—	-3	фикс.	Сброс	Сброс	0x0000_0004
—	-2	фикс.	NMI	Не маскируемое прерывание. Здесь же и CSS	0x0000_0008
—	-1	фикс.	HardFault	Все классы сбоев	0x0000_000C
—	0	устан.	MemManage	Управление памятью	0x0000_0010
—	1	устан.	BusFault	Сбой предвыборки и доступа к памяти	0x0000_0014
—	2	устан.	UsageFault	Неопределённая команда или недопустимое состояние	0x0000_0018
—	—	—	—	Резерв	0x0000_001C - 0x0000_002B
—	3	устан.	SVCall	Вызов Супервизора	0x0000_002C
—	4	устан.	Debug Monitor	Монитор отладка	0x0000_0030
—	—	—	—	Резерв	0x0000_0034
—	5	устан.	PendSV	Задерживаемый запрос системы	0x0000_0038
—	6	устан.	SysTick	Системный таймер	0x0000_003C
0	7	устан.	WWDG	Оконный сторожевой таймер	0x0000_0040
1	8	устан.	PVD	PVD через EXTI Line детектор	0x0000_0044
2	9	устан.	TAMPER	Вмешательство	0x0000_0048
3	10	устан.	RTC	Глобальное прерывание RTC	0x0000_004C
4	11	устан.	FLASH	Глобальное прерывание FLASH	0x0000_0050
5	12	устан.	RCC	Глобальное прерывание RCC	0x0000_0054
6	13	устан.	EXTI0	EXTI Line0	0x0000_0058

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
7	14	устан.	EXTI1	EXTI Line1	0x0000_005C
8	15	устан.	EXTI2	EXTI Line2	0x0000_0060
9	16	устан.	EXTI3	EXTI Line3	0x0000_0064
10	17	устан.	EXTI4	EXTI Line4	0x0000_0068
11	18	устан.	DMA1_Channel1	Глобальное прерывание DMA1_Channel1	0x0000_006C
12	19	устан.	DMA1_Channel2	Глобальное прерывание DMA1_Channel2	0x0000_0070
13	20	устан.	DMA1_Channel3	Глобальное прерывание DMA1_Channel3	0x0000_0074
14	21	устан.	DMA1_Channel4	Глобальное прерывание DMA1_Channel4	0x0000_0078
15	22	устан.	DMA1_Channel5	Глобальное прерывание DMA1_Channel5	0x0000_007C
16	23	устан.	DMA1_Channel6	Глобальное прерывание DMA1_Channel6	0x0000_0080
17	24	устан.	DMA1_Channel7	Глобальное прерывание DMA1_Channel7	0x0000_0084
18	25	устан.	ADC1_2	Глобальное прерывание ADC1 and ADC2	0x0000_0088
19	26	устан.	CAN1_TX	Прерывания CAN1_TX	0x0000_008C
20	27	устан.	CAN1_RX0	Прерывания CAN1_RX0	0x0000_0090
21	28	устан.	CAN1_RX1	Прерывание CAN1_RX1	0x0000_0094
22	29	устан.	CAN1_SCE	Прерывание CAN1_SCE	0x0000_0098
23	30	устан.	EXTI9_5	Прерывания EXTI Line[9:5]	0x0000_009C
24	31	устан.	TIM1_BRK	Прерывание TIM1_BRK	0x0000_00A0
25	32	устан.	TIM1_UP	Прерывание TIM1_UP	0x0000_00A4
26	33	устан.	TIM1_TRG_COM	Прерывание TIM1 Trigger and Commutation	0x0000_00A8
27	34	устан.	TIM1_CC	Прерывание TIM1 Capture Compare	0x0000_00AC
28	35	устан.	TIM2	Глобальное прерывание TIM2	0x0000_00B0
29	36	устан.	TIM3	Глобальное прерывание TIM3	0x0000_00B4
30	37	устан.	TIM4	Глобальное прерывание TIM4	0x0000_00B8
31	38	устан.	I2C1_EV	Событие I^2C1	0x0000_00BC
32	39	устан.	I2C1_ER	Прерывание ошибки I^2C1	0x0000_00C0
33	40	устан.	I2C2_EV	Событие I^2C2	0x0000_00C4
34	41	устан.	I2C2_ER	Прерывание ошибки I^2C2	0x0000_00C8
35	42	устан.	SPI1	Глобальное прерывание SPI1	0x0000_00CC
36	43	устан.	SPI2	Глобальное прерывание SPI2	0x0000_00D0
37	44	устан.	USART1	Глобальное прерывание USART1	0x0000_00D4
38	45	устан.	USART2	Глобальное прерывание USART2	0x0000_00D8
39	46	устан.	USART3	Глобальное прерывание USART3	0x0000_00DC
40	47	устан.	EXTI15_10	Прерывания EXTI Line[15:10]	0x0000_00E0

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
41	48	устан.	RTCAlarm	Прерывание RTC alarm через EXTI	0x0000_00E4
42	49	устан.	OTG_FS_WKUP	Прерывание USB On-The-Go FS Wakeup через EXTI line	0x0000_00E8
—	—	—	—	Резерв	0x0000_00EC - 0x0000_0104
50	57	устан.	TIM5	Глобальное прерывание TIM5	0x0000_0108
51	58	устан.	SPI3	Глобальное прерывание SPI3	0x0000_010C
52	59	устан.	UART4	Глобальное прерывание UART4	0x0000_0110
53	60	устан.	UART5	Глобальное прерывание UART5	0x0000_0114
54	61	устан.	TIM6	Глобальное прерывание TIM6	0x0000_0118
55	62	устан.	TIM7	Глобальное прерывание TIM7	0x0000_011C
56	63	устан.	DMA2_Channel1	Глобальное прерывание DMA2_Channel1	0x0000_0120
57	64	устан.	DMA2_Channel2	Глобальное прерывание DMA2_Channel2	0x0000_0124
58	65	устан.	DMA2_Channel3	Глобальное прерывание DMA2_Channel3	0x0000_0128
59	66	устан.	DMA2_Channel4	Глобальное прерывание DMA2_Channel4	0x0000_012C
60	67	устан.	DMA2_Channel5	Глобальное прерывание DMA2_Channel5	0x0000_0130
61	68	устан.	ETH	Глобальное прерывание Ethernet	0x0000_0134
62	69	устан.	ETH_WKUP	Прерывание Ethernet Wakeup через EXTI line	0x0000_0138
63	70	устан.	CAN2_TX	Прерывания CAN2_TX	0x0000_013C
64	71	устан.	CAN2_RX0	Прерывания CAN2_RX0	0x0000_0140
65	72	устан.	CAN2_RX1	Прерывания CAN2_RX1	0x0000_0144
66	73	устан.	CAN2_SCE	Прерывание CAN2_SCE	0x0000_0148
67	74	устан.	OTG_FS	Глобальное прерывание USB On The Go FS	0x0000_014C

Таблица 62. Устройства XL-плотности

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
—	—	—	—	Резерв	0x0000_0000
—	-3	фикс.	Сброс	Сброс	0x0000_0004
—	-2	фикс.	NMI	Не маскируемое прерывание. Здесь же и CSS	0x0000_0008
—	-1	фикс.	HardFault	Все классы сбоев	0x0000_000C
—	0	устан.	MemManage	Управление памятью	0x0000_0010
—	1	устан.	BusFault	Сбой предвыборки и доступа к памяти	0x0000_0014
—	2	устан.	UsageFault	Неопределённая команда или недопустимое состояние	0x0000_0018
—	—	—	—	Резерв	0x0000_001C - 0x0000_002B
—	3	устан.	SVCall	Вызов Супервизора	0x0000_002C

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
—	4	устан.	Debug Monitor	Монитор отладка	0x0000_0030
—	—	—	—	Резерв	0x0000_0034
—	5	устан.	PendSV	Задерживаемый запрос системы	0x0000_0038
—	6	устан.	SysTick	Системный таймер	0x0000_003C
0	7	устан.	WWDG	Оконный сторожевой таймер	0x0000_0040
1	8	устан.	PVD	PVD через EXTI Line детектор	0x0000_0044
2	9	устан.	TAMPER	Вмешательство	0x0000_0048
3	10	устан.	RTC	Глобальное прерывание RTC	0x0000_004C
4	11	устан.	FLASH	Глобальное прерывание FLASH	0x0000_0050
5	12	устан.	RCC	Глобальное прерывание RCC	0x0000_0054
6	13	устан.	EXTI0	EXTI Line0	0x0000_0058
7	14	устан.	EXTI1	EXTI Line1	0x0000_005C
8	15	устан.	EXTI2	EXTI Line2	0x0000_0060
9	16	устан.	EXTI3	EXTI Line3	0x0000_0064
10	17	устан.	EXTI4	EXTI Line4	0x0000_0068
11	18	устан.	DMA1_Channel1	Глобальное прерывание DMA1_Channel1	0x0000_006C
12	19	устан.	DMA1_Channel2	Глобальное прерывание DMA1_Channel2	0x0000_0070
13	20	устан.	DMA1_Channel3	Глобальное прерывание DMA1_Channel3	0x0000_0074
14	21	устан.	DMA1_Channel4	Глобальное прерывание DMA1_Channel4	0x0000_0078
15	22	устан.	DMA1_Channel5	Глобальное прерывание DMA1_Channel5	0x0000_007C
16	23	устан.	DMA1_Channel6	Глобальное прерывание DMA1_Channel6	0x0000_0080
17	24	устан.	DMA1_Channel7	Глобальное прерывание DMA1_Channel7	0x0000_0084
18	25	устан.	ADC1_2	Глобальное прерывание ADC1 and ADC2	0x0000_0088
19	26	устан.	USB_HP_CAN_TX	Приоритетное USB или CAN_TX	0x0000_008C
20	27	устан.	USB_LP_CAN_RX0	Не приоритетное USB или CAN_RX0	0x0000_0090
21	28	устан.	CAN_RX1	Прерывание CAN_RX1	0x0000_0094
22	29	устан.	CAN_SCE	Прерывание CAN_SCE	0x0000_0098
23	30	устан.	EXTI9_5	Прерывания EXTI Line[9:5]	0x0000_009C
24	31	устан.	TIM1_BRK_TIM9	Прерывание TIM1_BRK и TIM9	0x0000_00A0
25	32	устан.	TIM1_UP_TIM10	Прерывание TIM1 Update и TIM10 глобальное	0x0000_00A4
26	33	устан.	TIM1_TRG_COM_ TIM11	Прерывание TIM1 Trigger и TIM11 глобальное	0x0000_00A8
27	34	устан.	TIM1_CC	Прерывание TIM1 Capture Compare	0x0000_00AC
28	35	устан.	TIM2	Глобальное прерывание TIM2	0x0000_00B0
29	36	устан.	TIM3	Глобальное прерывание TIM3	0x0000_00B4

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
30	37	устан.	TIM4	Глобальное прерывание TIM4	0x0000_00B8
31	38	устан.	I2C1_EV	Событие I²C1	0x0000_00BC
32	39	устан.	I2C1_ER	Прерывание ошибки I²C1	0x0000_00C0
33	40	устан.	I2C2_EV	Событие I²C2	0x0000_00C4
34	41	устан.	I2C2_ER	Прерывание ошибки I²C2	0x0000_00C8
35	42	устан.	SPI1	Глобальное прерывание SPI1	0x0000_00CC
36	43	устан.	SPI2	Глобальное прерывание SPI2	0x0000_00D0
37	44	устан.	USART1	Глобальное прерывание USART1	0x0000_00D4
38	45	устан.	USART2	Глобальное прерывание USART2	0x0000_00D8
39	46	устан.	USART3	Глобальное прерывание USART3	0x0000_00DC
40	47	устан.	EXTI15_10	Прерывания EXTI Line[15:10]	0x0000_00E0
41	48	устан.	RTCAlarm	Прерывание RTC alarm через EXTI	0x0000_00E4
42	49	устан.	USBWakeUp	Прерывание USB wakeup через EXTI line	0x0000_00E8
43	50	устан.	TIM8_BRK_TIM12	Прерывание TIM8 Break TIM12 глобальное	0x0000_00EC
44	51	устан.	TIM8_UP_TIM13	Прерывание TIM8 Update и TIM13 глобальное	0x0000_00F0
45	52	устан.	TIM8_TRG_COM_ TIM14	Прерывание TIM8 Trigger и TIM14 глобальное	0x0000_00F4
46	53	устан.	TIM8_CC	Прерывание TIM8 Capture Compare	0x0000_00F8
47	54	устан.	ADC3	Глобальное прерывание ADC3	0x0000_00FC
48	55	устан.	FSMC	Глобальное прерывание FSMC	0x0000_0100
49	56	устан.	SDIO	Глобальное прерывание SDIO	0x0000_0104
50	57	устан.	TIM5	Глобальное прерывание TIM5	0x0000_0108
51	58	устан.	SPI3	Глобальное прерывание SPI3	0x0000_010C
52	59	устан.	UART4	Глобальное прерывание UART4	0x0000_0110
53	60	устан.	UART5	Глобальное прерывание UART5	0x0000_0114
54	61	устан.	TIM6	Глобальное прерывание TIM6	0x0000_0118
55	62	устан.	TIM7	Глобальное прерывание TIM7	0x0000_011C
56	63	устан.	DMA2_Channel1	Глобальное прерывание DMA2_Channel1	0x0000_0120
57	64	устан.	DMA2_Channel2	Глобальное прерывание DMA2_Channel2	0x0000_0124
58	65	устан.	DMA2_Channel3	Глобальное прерывание DMA2_Channel3	0x0000_0128
59	66	устан.	DMA2_Channel4_5	Глобальное прерывание DMA2_Channel4 и DMA2_Channel5	0x0000_012C

Таблица 63. Другие устройства STM32F10xx

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
—	—	—	—	Резерв	0x0000_0000
—	-3	фикс.	Сброс	Сброс	0x0000_0004
—	-2	фикс.	NMI	Не маскируемое прерывание. Здесь же и CSS	0x0000_0008
—	-1	фикс.	HardFault	Все классы сбоев	0x0000_000C
—	0	устан.	MemManage	Управление памятью	0x0000_0010
—	1	устан.	BusFault	Сбой предвыборки и доступа к памяти	0x0000_0014
—	2	устан.	UsageFault	Неопределённая команда или недопустимое состояние	0x0000_0018
—	—	—	—	Резерв	0x0000_001C - 0x0000_002B
—	3	устан.	SVCall	Вызов Супервизора	0x0000_002C
—	4	устан.	Debug Monitor	Монитор отладка	0x0000_0030
—	—	—	—	Резерв	0x0000_0034
—	5	устан.	PendSV	Задерживаемый запрос системы	0x0000_0038
—	6	устан.	SysTick	Системный таймер	0x0000_003C
0	7	устан.	WWDG	Оконный сторожевой таймер	0x0000_0040
1	8	устан.	PVD	PVD через EXTI Line детектор	0x0000_0044
2	9	устан.	TAMPER	Вмешательство	0x0000_0048
3	10	устан.	RTC	Глобальное прерывание RTC	0x0000_004C
4	11	устан.	FLASH	Глобальное прерывание FLASH	0x0000_0050
5	12	устан.	RCC	Глобальное прерывание RCC	0x0000_0054
6	13	устан.	EXTI0	EXTI Line0	0x0000_0058
7	14	устан.	EXTI1	EXTI Line1	0x0000_005C
8	15	устан.	EXTI2	EXTI Line2	0x0000_0060
9	16	устан.	EXTI3	EXTI Line3	0x0000_0064
10	17	устан.	EXTI4	EXTI Line4	0x0000_0068
11	18	устан.	DMA1_Channel1	Глобальное прерывание DMA1_Channel1	0x0000_006C
12	19	устан.	DMA1_Channel2	Глобальное прерывание DMA1_Channel2	0x0000_0070
13	20	устан.	DMA1_Channel3	Глобальное прерывание DMA1_Channel3	0x0000_0074
14	21	устан.	DMA1_Channel4	Глобальное прерывание DMA1_Channel4	0x0000_0078
15	22	устан.	DMA1_Channel5	Глобальное прерывание DMA1_Channel5	0x0000_007C
16	23	устан.	DMA1_Channel6	Глобальное прерывание DMA1_Channel6	0x0000_0080
17	24	устан.	DMA1_Channel7	Глобальное прерывание DMA1_Channel7	0x0000_0084
18	25	устан.	ADC1_2	Глобальное прерывание ADC1 and ADC2	0x0000_0088
19	26	устан.	USB_HP_CAN_TX	Приоритетное USB или CAN_TX	0x0000_008C

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
20	27	устан.	USB_LP_CAN_RX0	Не приоритетное USB или CAN_RX0	0x0000_0090
21	28	устан.	CAN_RX1	Прерывание CAN_RX1	0x0000_0094
22	29	устан.	CAN_SCE	Прерывание CAN_SCE	0x0000_0098
23	30	устан.	EXTI9_5	Прерывания EXTI Line[9:5]	0x0000_009C
24	31	устан.	TIM1_BRK	Прерывание TIM1_BRK	0x0000_00A0
25	32	устан.	TIM1_UP	Прерывание TIM1 Update	0x0000_00A4
26	33	устан.	TIM1_TRG_COM	Прерывание TIM1 Trigger	0x0000_00A8
27	34	устан.	TIM1_CC	Прерывание TIM1 Capture Compare	0x0000_00AC
28	35	устан.	TIM2	Глобальное прерывание TIM2	0x0000_00B0
29	36	устан.	TIM3	Глобальное прерывание TIM3	0x0000_00B4
30	37	устан.	TIM4	Глобальное прерывание TIM4	0x0000_00B8
31	38	устан.	I2C1_EV	Событие I^2C1	0x0000_00BC
32	39	устан.	I2C1_ER	Прерывание ошибки I^2C1	0x0000_00C0
33	40	устан.	I2C2_EV	Событие I^2C2	0x0000_00C4
34	41	устан.	I2C2_ER	Прерывание ошибки I^2C2	0x0000_00C8
35	42	устан.	SPI1	Глобальное прерывание SPI1	0x0000_00CC
36	43	устан.	SPI2	Глобальное прерывание SPI2	0x0000_00D0
37	44	устан.	USART1	Глобальное прерывание USART1	0x0000_00D4
38	45	устан.	USART2	Глобальное прерывание USART2	0x0000_00D8
39	46	устан.	USART3	Глобальное прерывание USART3	0x0000_00DC
40	47	устан.	EXTI15_10	Прерывания EXTI Line[15:10]	0x0000_00E0
41	48	устан.	RTCAlarm	Прерывание RTC alarm через EXTI	0x0000_00E4
42	49	устан.	USBWakeUp	Прерывание USB wakeup через EXTI line	0x0000_00E8
43	50	устан.	TIM8_BRK	Прерывание TIM8 Break	0x0000_00EC
44	51	устан.	TIM8_UP	Прерывание TIM8 Update	0x0000_00F0
45	52	устан.	TIM8_TRG_COM	Прерывание TIM8 Trigger	0x0000_00F4
46	53	устан.	TIM8_CC	Прерывание TIM8 Capture Compare	0x0000_00F8
47	54	устан.	ADC3	Глобальное прерывание ADC3	0x0000_00FC
48	55	устан.	FSMC	Глобальное прерывание FSMC	0x0000_0100
49	56	устан.	SDIO	Глобальное прерывание SDIO	0x0000_0104
50	57	устан.	TIM5	Глобальное прерывание TIM5	0x0000_0108
51	58	устан.	SPI3	Глобальное прерывание SPI3	0x0000_010C
52	59	устан.	UART4	Глобальное прерывание UART4	0x0000_0110
53	60	устан.	UART5	Глобальное прерывание UART5	0x0000_0114

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
54	61	устан.	TIM6	Глобальное прерывание TIM6	0x0000_0118
55	62	устан.	TIM7	Глобальное прерывание TIM7	0x0000_011C
56	63	устан.	DMA2_Channel1	Глобальное прерывание DMA2_Channel1	0x0000_0120
57	64	устан.	DMA2_Channel2	Глобальное прерывание DMA2_Channel2	0x0000_0124
58	65	устан.	DMA2_Channel3	Глобальное прерывание DMA2_Channel3	0x0000_0128
59	66	устан.	DMA2_Channel4_5	Глобальное прерывание DMA2_Channel4 и DMA2_Channel5	0x0000_012C

10.2. Контроллер внешних прерываний/событий (EXTI)

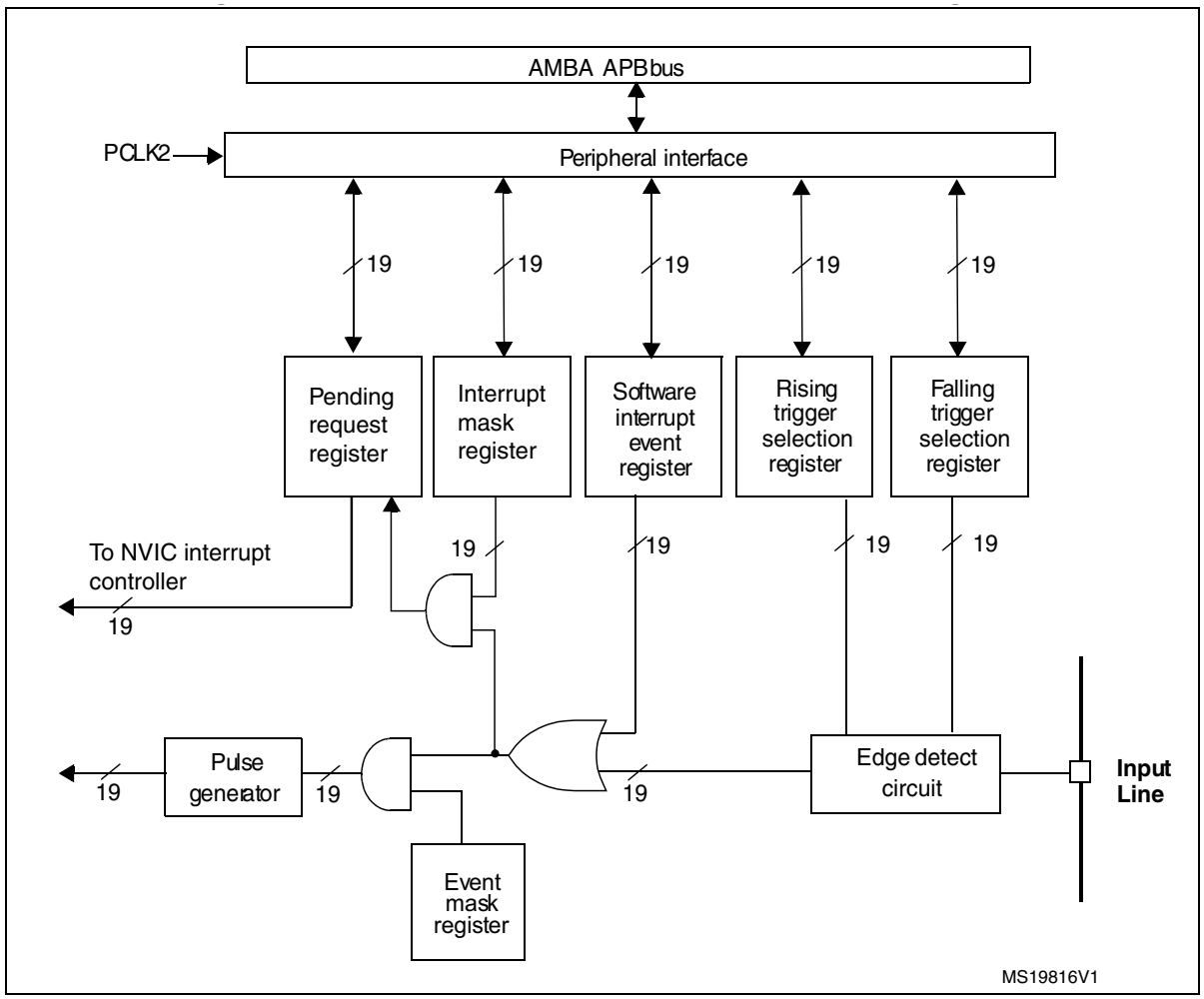
Контроллер внешних прерываний/событий содержит до 20 детекторов фронта запроса событий/прерываний в сетевых устройствах и до 19 детекторов в других устройствах. Каждая входная линия может быть независимо установлена на выбор типа (событие или прерывание) и тип фронта (передний, задний или оба). Маскируются линии независимо. Для запросов прерывания есть регистр удержания.

10.2.1. Основные свойства

Вот они:

- Независимый триггер и маска для каждой линии
- Специальный бит статуса для каждой линии прерывания
- Возбуждение до 20 программных запросов событий/прерываний
- Обнаружение внешних сигналов короче такта APB2. См. описания устройств.

10.2.2. Блок-схема



10.2.3. Событие побудки

Разбудить ядро STM32F10xxx из состояния WFE можно изнутри и снаружи:

- разрешив прерывание в регистре управления периферии (но не в NVIC) и разрешив бит SEVONPEND в регистре управления системы. При выходе из WFE надо очистить биты удержания в периферии и в NVIC IRQ.
- поставив внешнюю или внутреннюю линию EXTI в режим события. При выходе из WFE биты удержания в периферии и в NVIC IRQ чистить не надо (их там и нет).

В сетевых устройствах будить можно и событием Ethernet wakeup.

10.2.4. Функциональное описание

Если захотелось прерывания, то его надо сконфигурировать и разрешить двумя регистрами триггеров, определив желаемый фронт и записав "1" в маску. При появлении на внешней линии прерывания нужного фронта генерируется прерывание и ставится бит удержания нужной линии прерывания. Запрос сбрасывается записью ‘1’ в регистр удержания.

Если захотелось события, то его линию надо сконфигурировать и разрешить двумя регистрами триггеров, определив желаемый фронт и записав "1" в регистр маски события. При появлении на линии события нужного фронта генерируется импульс события, но бит удержания линии события не ставится.

Программно событие/прерывание можно вызвать записью ‘1’ в регистр прерывания/события.

Аппаратное прерывание.

Дабы получить прерывание от 20 линий внешних источников надо:

- Поставить биты маски 20 линий прерывания (EXTI_IMR)
- Поставить биты Выбора Запуска линий прерывания (EXTI_RTSR и EXTI_FTSR)
- Поставить биты разрешения и маски канала NVIC IRQ, отведённого для контроллера EXTI, чтобы поступающее по одной из 20 линий прерывание было правильно обработано.

Аппаратное событие.

Чтобы получить событие от 20 линий внешних источников надо:

- Поставить биты маски 20 линий события (EXTI_EMR)
- Поставить биты Выбора Запуска линий события (EXTI_RTSR и EXTI_FTSR)

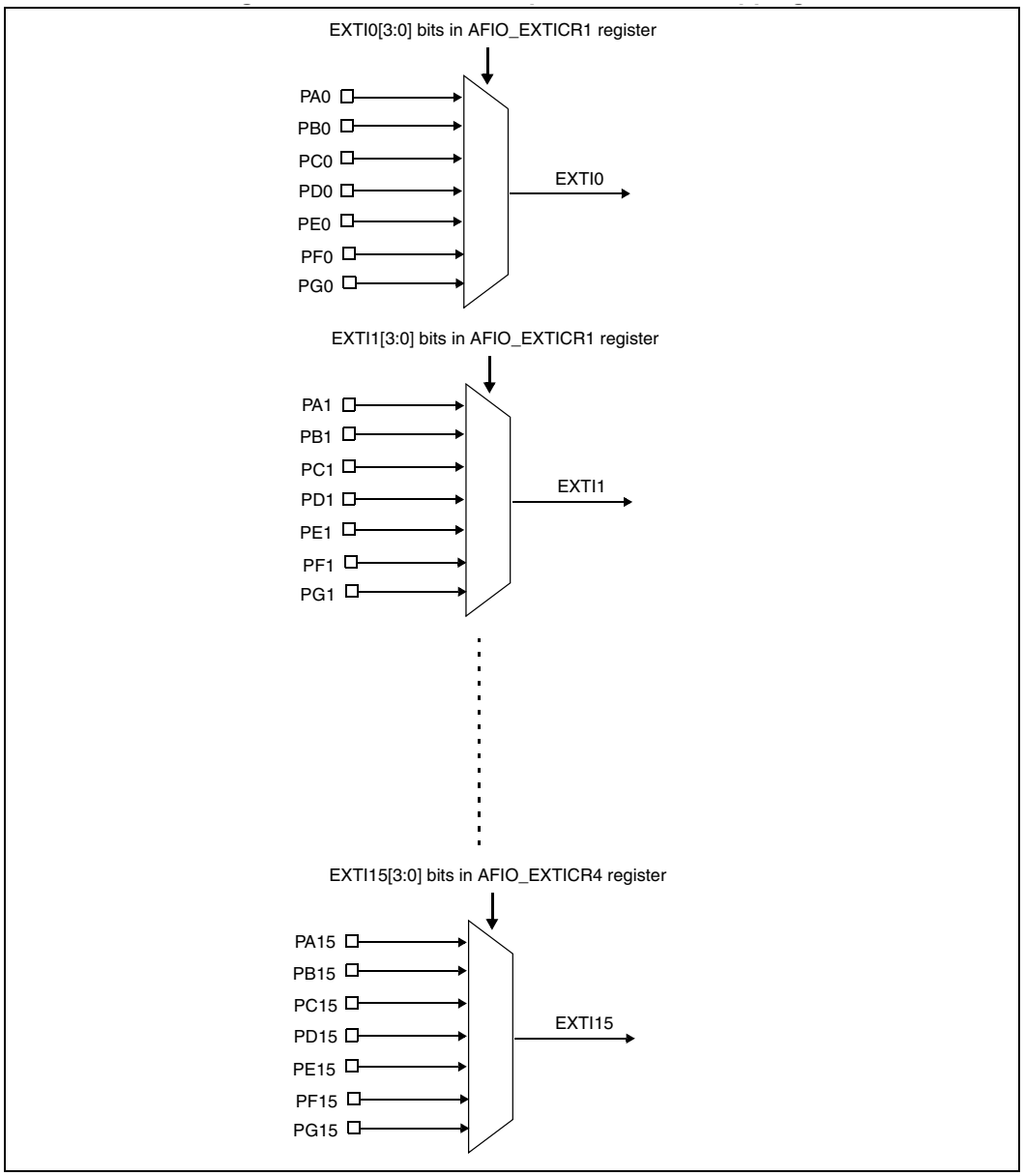
Программное событие/прерывание.

По этим 20 линиям можно получить и программное событие/прерывание:

- Поставить биты маски 20 линий события/прерывания (EXTI_IMR, EXTI_EMR)
- Поставить нужный бит в регистре программного прерывания (EXTI_SWIER)

10.2.5. Подключение линий внешних событий/прерываний

К 16 внешним линиям событий/прерываний 112 GPIO можно подключить так:



1. Для использования AFIO_EXTICRx при картировании линий событий/прерываний на GPIO сначала нужно включить тактирование AFIO в регистре RCC_APB2ENR. См. Секции 7.3.7 и 8.3.7.

Четыре остальные линии EXTI подключены так:

- EXTI line 16 подключена к выходу PVD
- EXTI line 17 подключена к событию RTC Alarm
- EXTI line 18 подключена к событию USB Wakeup
- EXTI line 19 подключена к событию Ethernet Wakeup (в сетевых устройствах)

10.3. Регистры EXTI

10.3.1. Регистр маски прерывания (EXTI_IMR)

Смещение адреса: 0x00

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												MR19	MR18	MR17	MR16
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:20 Резерв, не трогать.

— Биты 19:0 **MRx:** Маска прерывания линии x

0: Запрос прерывания на Line x закрыт

1: Запрос прерывания на Line x открыт

NB: Бит 19 используется в сетевых устройствах, иначе - резерв.

10.3.2. Регистр маски события (EXTI_EMR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												MR19	MR18	MR17	MR16
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:20 Резерв, не трогать.

— Биты 19:0 **MRx:** Маска события на линии x

0: Запрос события на Line x закрыт

1: Запрос события на Line x открыт

NB: Бит 19 используется в сетевых устройствах, иначе - резерв.

10.3.3. Регистр выбора переднего фронта запуска (EXTI_RTSR)

Смещение адреса: 0x08

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												TR19	TR18	TR17	TR16
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:20 Резерв, не трогать.

— Биты 19:0 **TRx:** Маска события/прерывания по переднему фронту на линии **x**

0: Запрос события/прерывания на линии **x** закрыт

1: Запрос события/прерывания на линии **x** открыт

NB: Бит 19 используется в сетевых устройствах, иначе - резерв.

NB: Внешние линии побудки запускаются фронтом, так что глитчи здесь недопустимы. Если передний фронт появляется во время записи в EXTI_RTSR, то бит удержания не ставится.

Одна линия может срабатывать по обоим фронтам сигнала, регистров разрешения то два.

10.3.4. Регистр выбора заднего фронта запуска (EXTI_RTSR)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												TR19	TR18	TR17	TR16
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:20 Резерв, не трогать.

— Биты 19:0 **TRx:** Маска события/прерывания по заднему фронту на линии **x**

0: Запрос события/прерывания на линии **x** закрыт

1: Запрос события/прерывания на линии **x** открыт

NB: Бит 19 используется в сетевых устройствах, иначе - резерв.

NB: Внешние линии побудки запускаются фронтом, так что глитчи здесь недопустимы. Если передний фронт появляется во время записи в EXTI_RTSR, то бит удержания не ставится.

Одна линия может срабатывать по обоим фронтам сигнала, регистров разрешения то два.

10.3.5. Регистр программного события/прерывания (EXTI_SWIER)

Смещение адреса: 0x10

По сбросу: 0x0000 0000

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

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

— Биты 31:20 Резерв, не трогать.

— Биты 19:0 **SWIERx:** Программное прерывание на линии **x**

Если прерывание на линии разрешено в регистре EXTI_IMR, то запись '1' в ранее нулевой бит этого регистра (передний фронт) ставит соответствующий бит в регистре EXTI_PR и соответственно просит о прерывании.

NB: Бит 19 используется в сетевых устройствах, иначе - резерв.

10.3.6. Регистр удержания (EXTI_PR)

Смещение адреса: 0x14

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												PR19	PR18	PR17	PR16
												rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

— Биты 31:20 Резерв, не трогать.

— Биты 19:0 **PRx:** Бит удержания на линии **x**

0: Запроса не было.

1: Запрос был.

Стирается записью "1" в соответствующий бит.

NB: Бит 19 используется в сетевых устройствах, иначе - резерв.

10.3.7. Карта регистров EXTI_PR

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	EXTI_IMR	Reserved												MR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	EXTI_EMR	Reserved												MR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	EXTI_RTSR	Reserved												TR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	EXTI_FTSR	Reserved												TR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	EXTI_SWIER	Reserved												SWIER[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	EXTI_PR	Reserved												PR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

11. Аналого-цифровой преобразователь (ADC)

11.1. Введение в АЦП

АЦП последовательного приближения имеет до 18 мультиплексируемых каналов для 16 внешних и 2 внутренних сигналов. A/D преобразование разных каналов может быть однократным, непрерывным, сканирующим или прерывным. Результат преобразования пишется в 16-бит регистр, выравненным вправо или влево.

Аналоговый сторож может отслеживать входной аналоговый сигнал на выход за заданные верхний и нижний пределы.

Входные такты АЦП получаются из PCLK2 после предделителя. Частота не должна превышать 14MHz.

11.2. Основные свойства АЦП

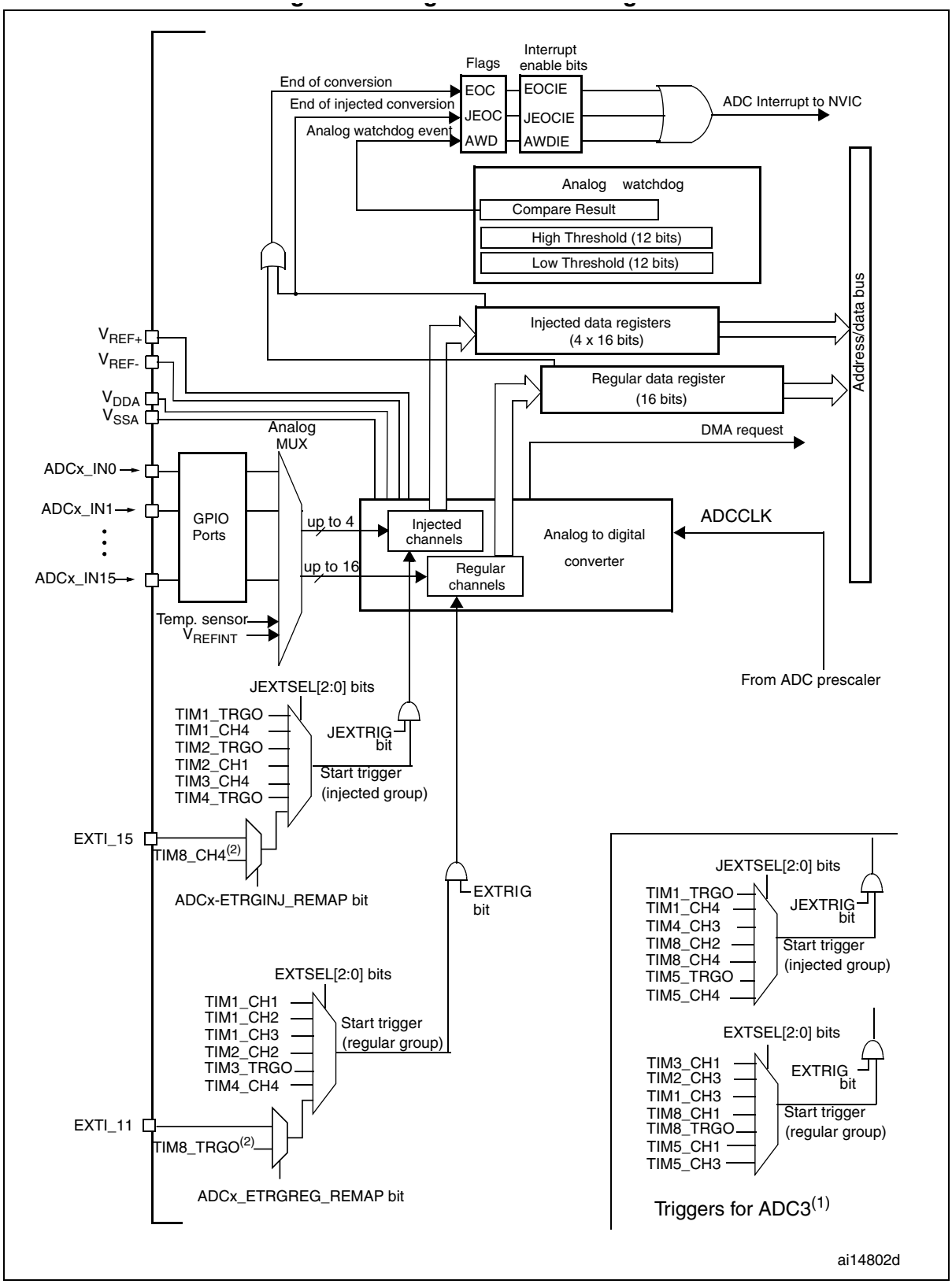
- Разрешение 12-бит
- Прерывание по Концу преобразования, Концу Внешнего преобразования и событию от Аналогового сторожа
- Однократный и непрерывный режимы преобразования
- Автоматический режим сканирования каналов от 0 до ‘n’
- Самокалибровка
- Выравнивание данных с встроенной когеррентностью.
- Программируемое время выборки между каналами
- Опция внешнего запуска регулярного и вставного преобразования
- Прерывное преобразование
- Парный режим (в устройствах с 2 АЦП или более)
- Время преобразования АЦП:
  - STM32F103xx: 1 µs на 56 MHz (1.17 µs на 72 MHz)
  - STM32F101xx: 1 µs на 28 MHz (1.55 µs на 36 MHz)
  - STM32F102xx: 1.2 µs на 48 MHz
  - STM32F105xx и STM32F107xx: 1 µs на 56 MHz (1.17 µs на 72 MHz)

- Питание АЦП: 2.4 V до 3.6 V
- Входной диапазон: $V_{REF-} \leq V_{IN} \leq V_{REF+}$
- Запрос DMA во время регулярного преобразования канала

NB: V_{REF-} , если есть (в зависимости от корпуса), должно подключать к V_{SSA}.

11.3. Функциональное описание АЦП

Блок-схема АЦП



1. Регулярный и вставной запуск ADC3 отличается от таковых в ADC1 и ADC2.
2. TIM8_CH4 и TIM8_TRGO с их битами картирования есть только в устройствах высокой и XL-плотности.

Таблица 65. Ножки АЦП.

Имя	Тип сигнала	Заметки
V_{REF+}	Положительное опорное напряжение	2.4 V ≤V_{REF+} ≤V_{DDA}
V_{DDA}(1)	Аналоговое питание	Эквивалентно V_{DD}, 2.4 V ≤V_{DDA} ≤3.6 V
V_{REF-}	Отрицательное опорное напряжение	V_{REF-} = V_{SSA}
V_{SSA}(1)	Аналоговая земля	Эквивалентна V_{SS}
ADCx_IN[15:0]	Аналоговые сигналы	До 21 аналогового канала(2)

1. V_{DDA} и V_{SSA} надо подключать к V_{DD} и V_{SS}, соответственно.
2. Конкретика есть к описании каждого устройства.

11.3.1. Вкл./Выкл. АЦП

АЦП включается установкой бита ADON в регистре ADC_CR2. При первой установке ADON в первый раз АЦП просыпается из режима Power Down.

Преобразование начинается при второй установке бита ADON после стабилизации АЦП (t_{STAB}).

Преобразование можно остановить сбросом бита ADON. В этом режиме АЦП почти не кушает (несколько µA).

11.3.2. Такты АЦП

Такты ADCCLK синхронны с PCLK2 (такты APB2). Контроллер RCC имеет отдельный предделитель для тактов АЦП.

11.3.3. Выбор канала

Есть 16 мультиплексируемых каналов. Можно организовать преобразования в две группы: регулярную и вставную. Группа состоит из последовательности преобразований любых каналов в любом порядке. Например, Ch3, Ch8, Ch2, Ch2, Ch0, Ch2, Ch2, Ch15.

- **Регулярная группа** содержит до 16 преобразований. Каналы и их порядок выбираются в регистрах ADC_SQRx. Общее число преобразований пишется в биты L[3:0] регистра ADC_SQR1.
- **Вставная группа** содержит до 4 преобразований. Каналы и их порядок выбираются в регистре ADC_JSQR. Общее число преобразований пишется в биты L[1:0] регистра ADC_JSQR.

Если регистры ADC_SQRx или ADC_JSQR изменяются во время преобразования, то текущее преобразование прерывается и выдаётся импульс запуска преобразования новой группы.

Внутренние каналы Датчик температуры/V_{REFINT}

Датчик температуры подключён к ADCx_IN16, а внутреннее опорное напряжение V_{REFINT} к ADCx_IN17. Эти два канала можно включать в обе группы.

NB: Датчик и V_{REFINT} доступны только на ведущем ADC1.

11.3.4. Одинарное преобразование

Запускается установкой бита ADON в регистре ADC_CR2 (только регулярный канал) или внешним сигналом (регулярный или вставной канал), а бит CONT равен 0.

При завершении преобразования:

- Регулярного канала:
  - Результат пишется в 16-бит регистр ADC_DR
  - Ставится флаг EOC (Конец Преобразования)
  - Если бит EOCIE стоит, то генерируется прерывание.
- Вставного канала:
  - Результат пишется в 16-бит регистр ADC_DRJ1
  - Ставится флаг JEOC (Конец Вставного Преобразования)
  - Если бит JEOCIE стоит, то генерируется прерывание.

АЦП останавливается.

11.3.5. Непрерывное преобразование

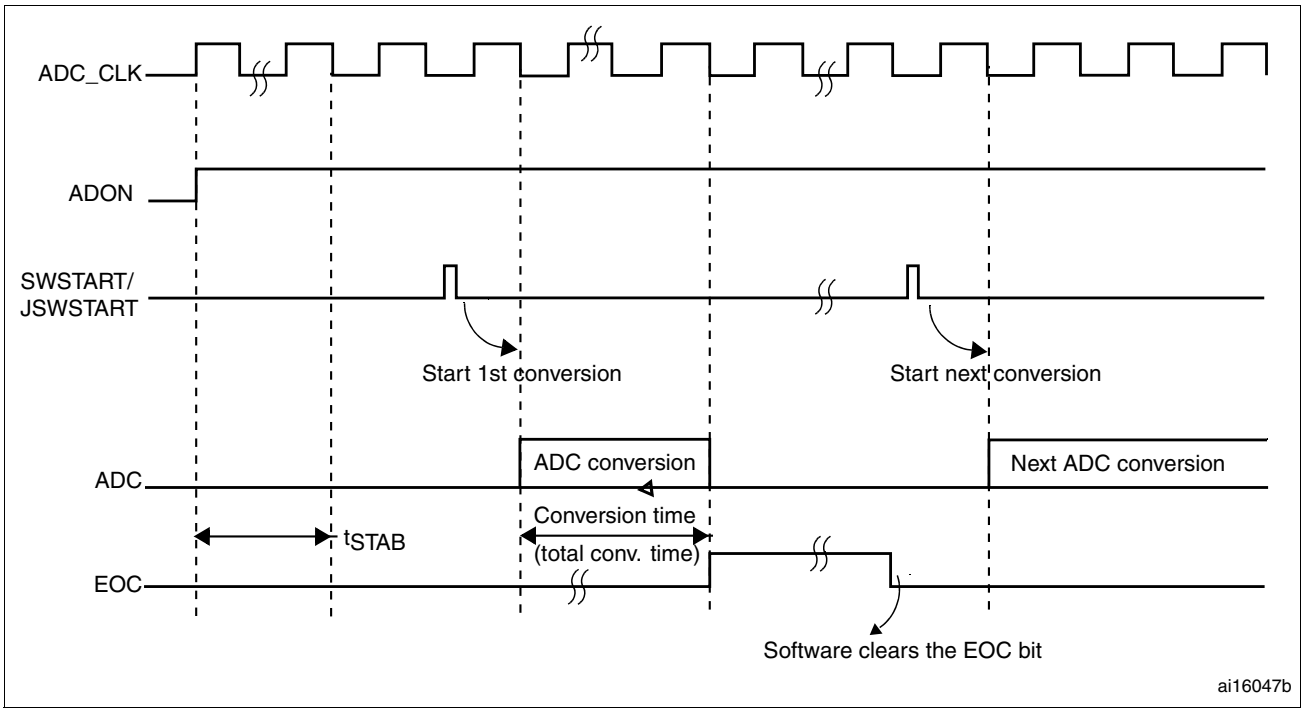
В этом режиме преобразования выполняются одно за другим. Запускается внешним сигналом или установкой бита ADON в регистре ADC_CR2, а бит CONT равен 1.

После каждого преобразования:

- Регулярного канала:
  - Результат пишется в 16-бит регистр ADC_DR
  - Ставится флаг EOC (Конец Преобразования)
  - Если бит EOCIE стоит, то генерируется прерывание.
- Вставного канала:
  - Результат пишется в 16-бит регистр ADC_DRJ1
  - Ставится флаг JEOC (Конец Вставного Преобразования)
  - Если бит JEOCIE стоит, то генерируется прерывание.

11.3.6. Временная диаграмма

Перед преобразованием АЦП нуждается в стабилизации (t_{STAB}). Через 14 тактов после запуска преобразования ставится флаг EOC и результат пишется в регистр ADC_DR.



11.3.7. Аналоговый сторож

Если результат АЦП выходит за верхний и нижний порог, то аналоговый сторож AWD ставит бит состояния. Пороги указываются в младших 12 битах 16-бит регистров ADC_HTR и ADC_LTR. Прерывание разрешается битом AWDIE в регистре ADC_CR1.

Значение порога не зависит от выравнивания, указанного в бите ALIGN регистра ADC_CR2. Сравнение делается раньше. См. *Секцию 11.5*.

Аналоговый сторож может быть разрешён для одного или всех каналов в регистре ADC_CR1.

Охраняемая зона

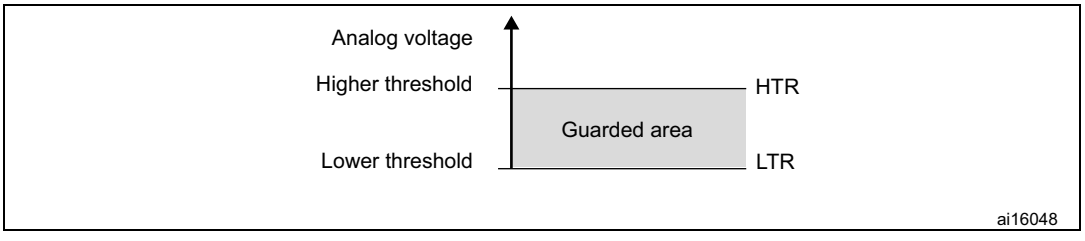


Таблица 66. Выбор каналов для охраны

Охраняемые каналы	Биты регистра ADC_CR1 (x = без разницы)		
	AWDSGL	AWDEN	JAWDEN
Нету	x	0	0
Все вставные	0	0	1
Все регулярные	0	1	0
Все регулярные и вставные	0	1	1
Один(1) вставной	1	0	1
Один(1) регулярный	1	1	0
Один(1) регулярный или вставной	1	1	1

1. Выбирается битами AWDCH[4:0]

11.3.8. Режим сканирования

Сканируется группа аналоговых каналов.

Включается установкой бита SCAN в регистре ADC_CR1. Тогда АЦП сканирует все каналы, указанные в регистрах ADC_SQRx (для регулярных) или ADC_JSQR (для вставных). Преобразование последовательно выполняется для всех каналов группы. Если стоит бит CONT, то преобразование выполняется снова и снова для всей группы.

В этом режиме должно ставить бит DMA для записи результатов в SRAM после каждого обновления регистра ADC_DR.

Данные вставных каналов всегда пишутся в регистры ADC_JDRx.

11.3.9. Управление вставными каналами

Запускаемая вставка

В это случае в регистре ADC_CR1 нужно очистить бит JAUTO и поставить бит SCAN.

1. Внешним сигналом или битом ADON в регистре ADC_CR2 запустить преобразование регулярной группы каналов.
2. Если во время обработки этой группы появляется внешний сигнал вставного запуска, то текущее преобразование останавливается и обрабатывается вставная последовательность в режиме однократного сканирования.
3. Затем восстанавливается регулярная последовательность с прерванного преобразования. Если во время вставной последовательности появляется запуск регулярной, то он не прерывает вставной, и регулярная выполняется после завершения вставной.

NB: При использовании запускаемой вставки интервал между событиями запуска должен быть больше вставляемой последовательности. Например, если длина последовательности равна 28 тактов ADC (2 преобразования и 1.5 такта времени выборки), то минимальный интервал между запусками должен быть не менее 29 тактов ADC.

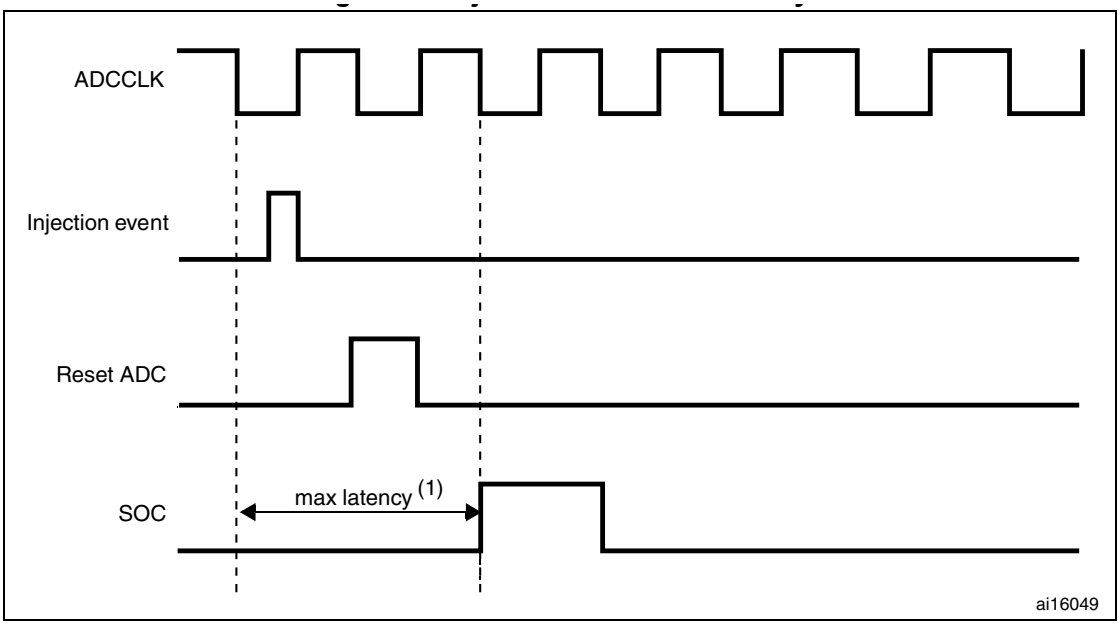
Автовставка

Если бит JAUTO стоит, то вставная группа каналов автоматически обрабатывается после завершения регулярной группы. Так можно выполнить до 20 преобразований, отмеченных в регистрах ADC_SQRx и ADC_JSQR. В этом режиме внешний запуск вставных каналов должен быть выключен. А если ещё стоит и бит CONT, то вся последовательность выполняется постоянно.

При значении прескалеров ADC от 4 до 8 для переключения между любыми последовательностями (регулярная/вставная) добавляется задержка в 1 такт. При предделителе ADC равном 2 и задержка равна 2 тактам ADC.

Автовставку и прерывный режим нельзя использовать одновременно.

Задержки вставной последовательности.



1. Значение максимальной задержки приведено в описаниях STM32F101xx и STM32F103xx.

11.3.10.Прерывный режим

Регулярная группа

Разрешается установкой бита DISCEN в регистре ADC_CR1. Используется для обработки короткой части последовательности преобразований (**n** <=8), записанной в регистрах ADC_SQRx. Значение **n** записано в битах DISCNUM[2:0] регистра ADC_CR1.

Внешний сигнал запускает следующие **n** преобразований последовательности из регистров ADC_SQRx пока не будет обработана вся последовательность. Общая длина последовательности определена в битах L[3:0] регистра ADC_SQR1.

Пример:

n = 3, преобразуемые каналы = 0, 1, 2, 3, 6, 7, 9, 10

1й запуск: преобразуются 0, 1, 2. Событие EOC выдаётся на каждое преобразование

2й запуск: преобразуются 3, 6, 7. Событие EOC выдаётся на каждое преобразование

3й запуск: преобразуются 9, 10. Событие EOC выдаётся на каждое преобразование

4й запуск: преобразуются 0, 1, 2. Событие EOC выдаётся на каждое преобразование

NB. При обработке регулярных групп заворота последовательности через начало не бывает и следующий запуск начинается с первого канала. См. пример ранее.

Вставная группа

Разрешается установкой бита JDISCEN в регистре ADC_CR1. Используется для обработки последовательности каналов, записанной в регистрах ADC_JSQR, один за другим.

Внешний сигнал запускает обработку следующего канала из регистров ADC_JSQR вплоть до конца. Общая длина последовательности определена в битах JL[1:0] регистра ADC_JSQR.

Пример:

n = 1, преобразуемые каналы = 1, 2, 3

1й запуск: канал 1

2й запуск: канал 2

3й запуск: канал 3, выдаются события EOC и JEOC

4й запуск: канал 1

NB. При завершении вставной последовательности очередной сигнал запускает преобразование первого в последовательности.

Прерывный режим и автовставку нельзя использовать одновременно.

Одновременно прерывный режим нужно использовать только для одной из групп (регулярной или вставной).

11.4. Калибровка

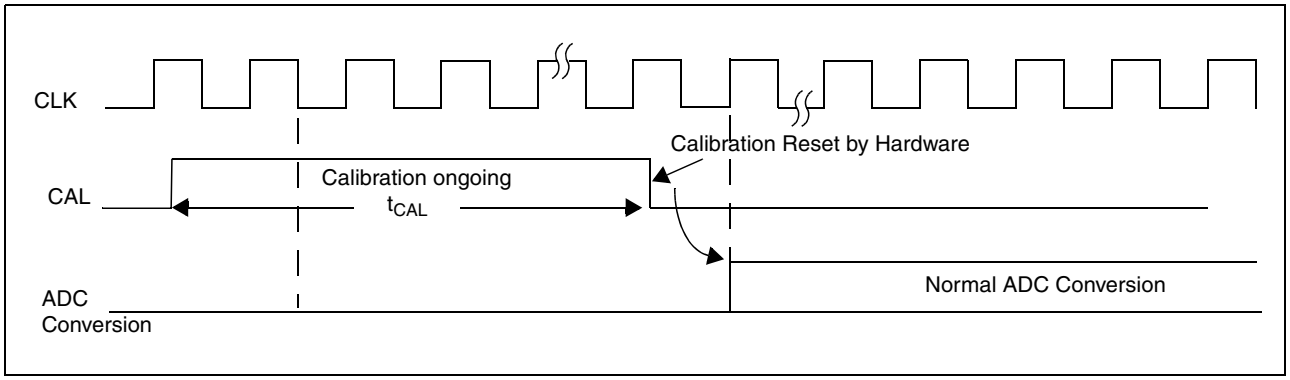
АЦП имеет встроенную самокалибровку, значительно снижающую погрешность преобразования из-за колебаний внутреннего банка емкостей. При калибровке для каждого конденсатора запоминается слово коррекции, которое устраняет вклад в ошибку преобразования каждого конденсатора.

Запускается установкой бита CAL в регистре ADC_CR2 CAL. Сбрасывается он аппаратно, и можно работать. Коды калибровки пишутся в регистр ADC_DR после завершения фазы калибровки.

NB. Рекомендуется калибровать АЦП единожды после каждого включения питания.

До калибровки АЦП он должен быть включён (ADON bit = ‘1’) не менее двух тактов АЦП.

Временная диаграмма калибровки.



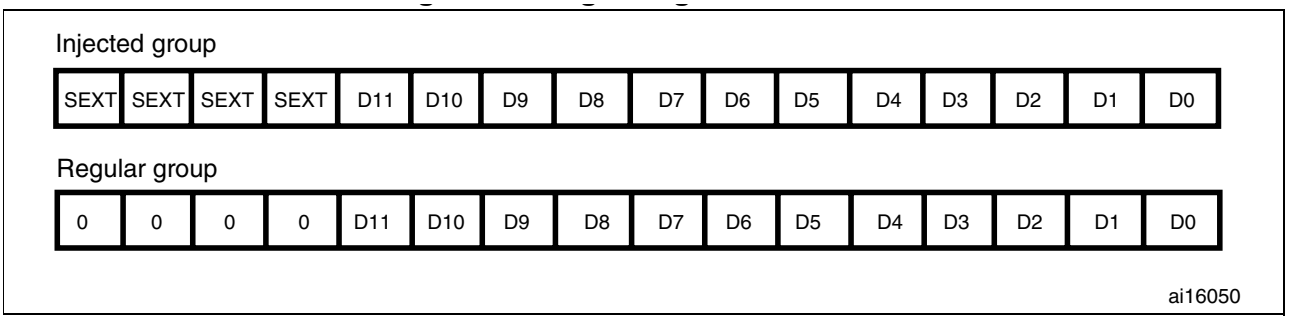
11.5. Выравнивание данных

Выравнивание данных вправо или влево выбирается битом ALIGN в регистре ADC_CR2.

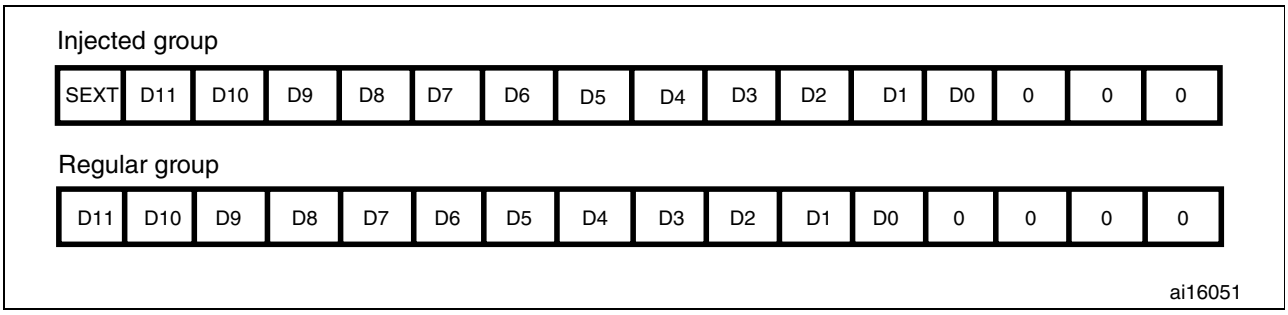
Данные вставных каналов имеют знак относительно смещения, записанного в регистр ADC_JOFRx и разрядность уменьшается на бит знака. Бит SEXT это расширенный знак.

Данные регулярных каналов беззнаковые и 12-разрядные.

Правое выравнивание.



Левое выравнивание.



11.6. Время выборки каналов

Время выборки входного сигнала каждого канала АЦП задаётся числом тактов ADC_CLK , определяемых битами SMP[2:0] в регистрах ADC_SMPR1 и ADC_SMPR2.

Общее время преобразования равно:

T_{conv} = Время выборки + 12.5 тактов

Пример:

При ADCCLK = 14 MHz и времени выборки 1.5 такта:

T_{conv} = 1.5 + 12.5 = 14 тактов = 1 μs

11.7. Внешний запуск преобразования

Внешний запуск преобразования разрешается битом EXTTRIG. Биты EXTSEL[2:0] и JEXTSEL[2:0] выбирают одно из 8 возможных событий запуска для регулярной и вставной групп.

NB: Внешний сигнал запуска обработки регулярных и вставных групп срабатывает только по переднему фронту.

Таблица 67. Внешний запуск регулярных каналов ADC1 и ADC2.

Источник	Тип	EXTSEL[2:0]
TIM1_CC1 event	Внутренний сигнал встроенных таймеров	000
TIM1_CC2 event		001
TIM1_CC3 event		010
TIM2_CC2 event		011
TIM3_TRGO event		100
TIM4_CC4 event		101
EXTI line 11/TIM8_TRGO event(1)(2)	Внешняя ножка/Внутренний от таймеров	110
SWSTART	Программный бит	111

1. TIM8_TRGO event есть только в устройствах XL-плотности.
2. Выбор EXTI line11 или TIM8_TRGO для регулярных каналов ADC1 и ADC2 делается битами ADC1_ETRGREG_REMAP и ADC2_ETRGREG_REMAP.

Таблица 68. Внешний запуск вставных каналов ADC1 и ADC2.

Источник	Тип	JEXTSEL[2:0]
TIM1_TRGO event	Внутренний сигнал встроенных таймеров	000
TIM1_CC4 event		001
TIM2_TRGO event		010
TIM2_CC1 event		011
TIM3_CC4 event		100
TIM4_TRGO event		101
EXTI line 15/TIM8_CC4 event(1)(2)	Внешняя ножка/Внутренний от таймеров	110
JSWSTART	Программный бит	111

1. TIM8_CC4 event есть только в устройствах высокой и XL-плотности.
2. Выбор EXTI line15 или TIM8_CC4 для вставных каналов ADC1 и ADC2 делается битами ADC1_ETRGREG_REMAP и ADC2_ETRGREG_REMAP.

Таблица 69. Внешний запуск регулярных каналов ADC3.

Источник	Тип	EXTSEL[2:0]
TIM3_CC1 event	Внутренний сигнал встроенных таймеров	000
TIM2_CC3 event		001
TIM1_CC3 event		010
TIM8_CC1 event		011
TIM8_TRGO event		100
TIM5_CC1 event		101
TIM5_CC3 event		110
SWSTART	Программный бит	111

Таблица 70. Внешний запуск вставных каналов ADC3.

Источник	Тип	JEXTSEL[2:0]
TIM1_TRGO event	Внутренний сигнал встроенных таймеров	000
TIM1_CC4 event		001
TIM4_CC3 event		010
TIM8_CC2 event		011
TIM8_CC4 event		100
TIM5_TRGO event		101
TIM5_CC4 event		110
JSWSTART	Программный бит	111

Программный запуск выполняется установкой битов `SWSTART` или `JSWSTART` в `ADC_CR2`.

Преобразование регулярной группы может прерываться запуском вставной группы.

11.8. Запрос DMA

Поскольку результаты преобразования регулярных каналов пишутся в один регистр, то для сохранения результатов нескольких каналов надо использовать DMA.

Запрос DMA для передачи регистра `ADC_DR` генерируется только концом преобразования регулярных каналов.

NB. Использование DMA доступно только для ADC1 и ADC3. Результаты ADC2 можно передавать в парном режиме с ведущим ADC1.

11.9. Парный режим АЦП

При наличии нескольких АЦП их можно парой запускать попеременно или одновременно в зависимости от битов `DUALMOD[2:0]` в регистре `ADC1_CR1` как ведущий ADC1 и ведомый ADC2.

NB: Если в парном режиме преобразование надо запускать внешним сигналом, то извне надо запускать только ведущий, а ведомый программно, во избежание ложного запуска ведомого. Но внешний запуск нужно разрешать и ведущему и ведомому АЦП.

Есть шесть возможных режимов:

- Вставной одновременный
- Регулярный одновременный
- Быстрый чередующийся

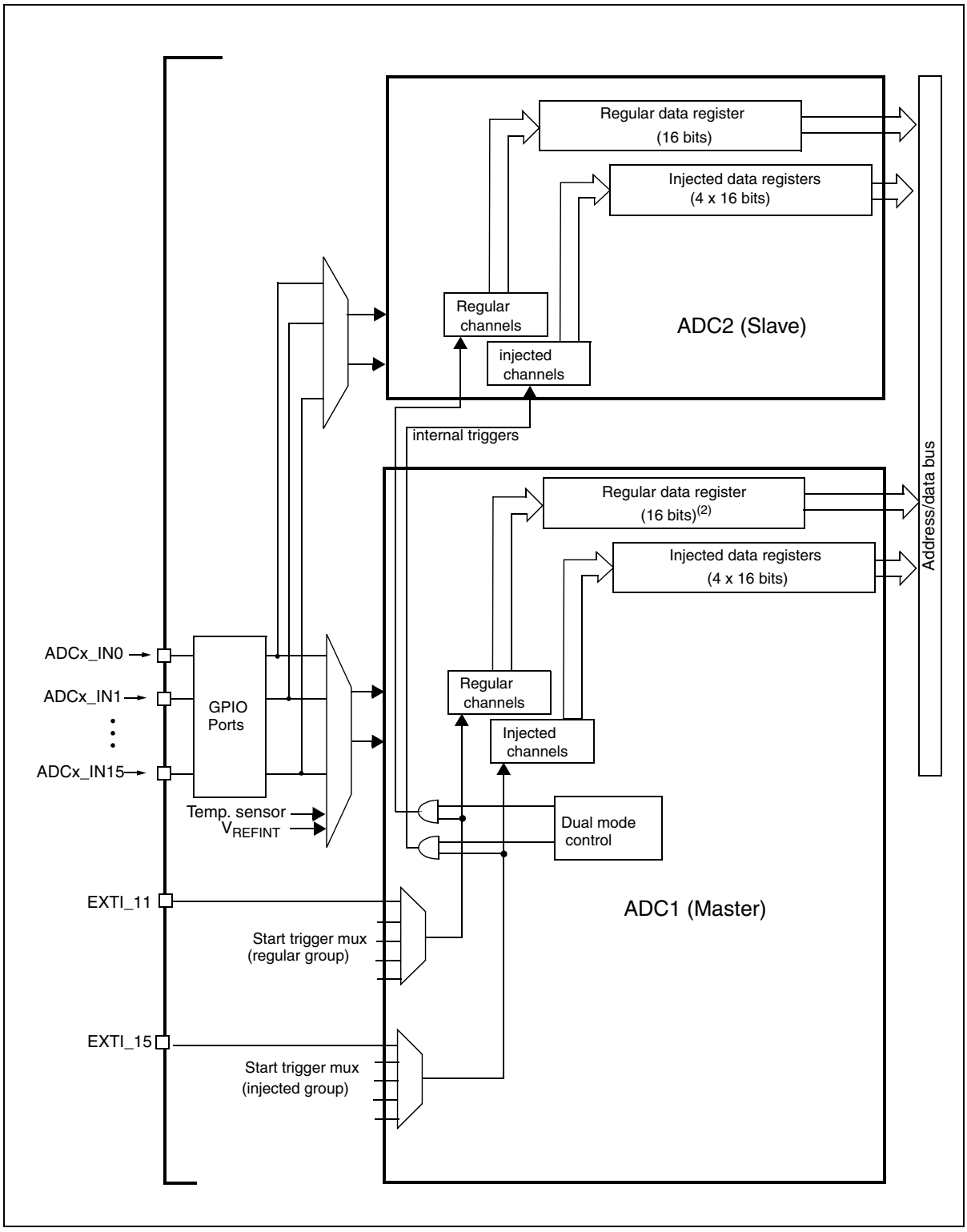
– Медленный чередующихся
– Попеременный запуск
– Независимый

Их можно сочетать:

– Вставной одновременный + Регулярный одновременный
– Регулярный одновременный + Переменный запуск
– Вставной одновременный + Чередующийся

NB: В парном режиме для чтения данных ведомого из регистра данных ведомого надо установить бит DMA даже если он не используется для передачи данных регулярного канала.

Блок-схема парного режима(1)



1. Внешний запуск ADC2 здесь не указан.
2. В некоторых парных режимах регистр данных ADC1 содержит данные и ADC1 и ADC2 в 32-х битах.

11.9.1. Вставной одновременный режим

Внешний сигнал запуска подаётся через мультиплексор вставной группы ADC1 (биты JEXTSEL[2:0] в регистр ADC1_CR2). Одновременно запуск поступает и на ADC2.

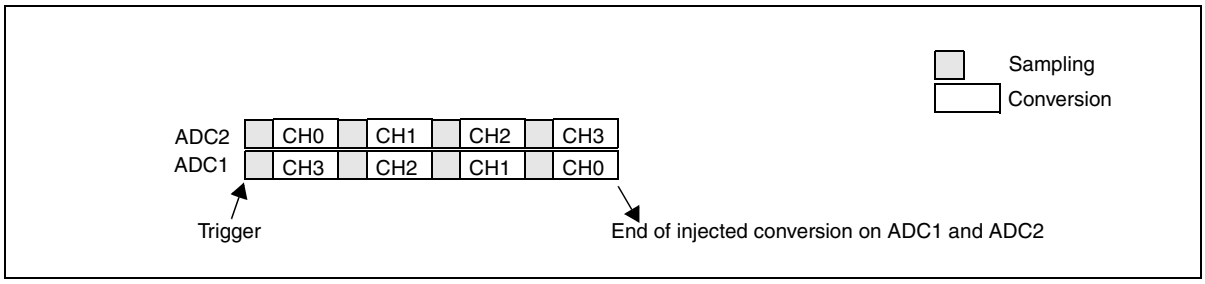
NB: Не преобразуйте один канал на двух АЦП (тогда времена выборки не совпадают).

По концу преобразования на ADC1 или ADC2:

- Результаты пишутся в регистры ADC_JDRx обоих АЦП.
- При получении обоих результатов генерируется прерывание JEOC (если разрешено в одном из АЦП).

NB: Времена выборки обоих каналов обоих АЦП должны совпадать.

Одновременный вставной режим на 4 каналах.



11.9.2. Регулярный одновременный режим

Внешний сигнал запуска подаётся через мультиплексор регулярной группы ADC1 (биты EXTSEL[2:0] в регистр ADC1_CR2). Одновременно запуск поступает и на ADC2.

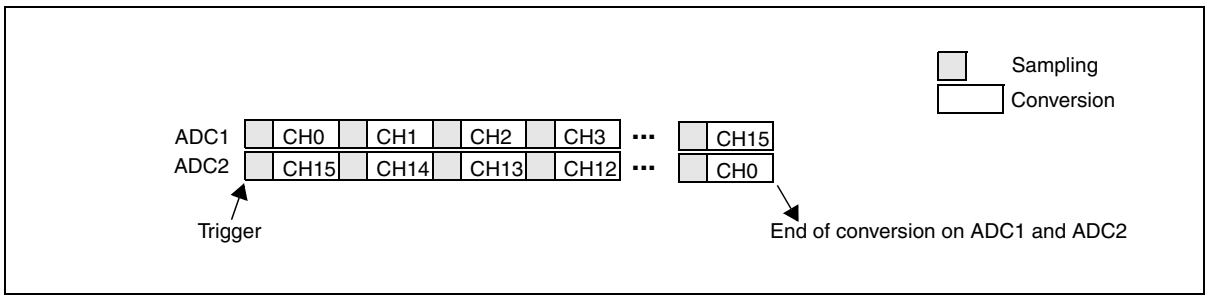
NB: Не преобразуйте один канал на двух АЦП (тогда времена выборки не совпадают).

По концу преобразования на ADC1 или ADC2:

- Генерируется запрос 32-бит DMA передачи (если стоит бит DMA) регистра ADC1_DR в SRAM с результатом ADC2 в старшем полуслове и ADC1 в младшем.
- При получении обоих результатов генерируется прерывание EOC (если разрешено в одном из АЦП).

NB: Времена выборки обоих каналов обоих АЦП должны совпадать.

Одновременный регулярный режим на 16 каналах.



11.9.3. Быстрый чередующийся режим

Этот режим предназначен только для регулярных групп (обычно один канал). Внешний сигнал запуска подаётся через мультиплексор регулярной группы ADC1. После запуска:

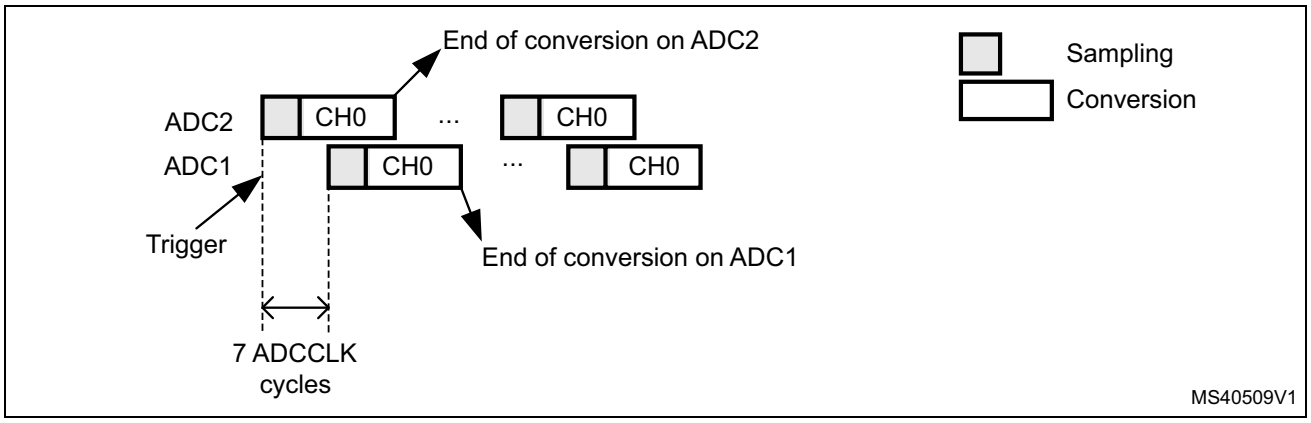
- ADC2 стартует немедленно
- ADC1 стартует после задержки в 7 тактов ADC.

Если в ADC1 и ADC2 стоит бит CONT, АЦП работают постоянно.

После генерации прерывания EOC в ADC1 (если разрешено битом EOCIE) ставится запрос 32-бит DMA передачи (если стоит бит DMA) регистра ADC1_DR в SRAM с результатом ADC2 в старшем полуслове и ADC1 в младшем.

NB: Максимально разрешённое время выборки составляет меньше 7 тактов ADCCLK во избежание перекрытия фаз выборки ADC1 и ADC2 при работе с одним каналом.

Постоянный быстрый чередующийся режим с 1 каналом



11.9.4. Медленный чередующийся режим

Этот режим предназначен только для регулярных групп (обычно один канал). Внешний сигнал запуска подаётся через мультиплексор регулярной группы ADC1. После запуска:

- ADC2 стартует немедленно
- ADC1 стартует после задержки в 14 тактов ADC.
- ADC2 стартует после второй задержки в 14 тактов ADC и так далее

NB: Максимально разрешённое время выборки составляет меньше 14 тактов ADCCLK во избежание перекрытия со следующим преобразованием.

Прерывание `EOC` генерируется ADC1 (если разрешено битом `EOCIE`), генерируется запрос 32-бит DMA передачи (если стоит бит `DMA`) регистра `ADC1_DR` в SRAM с результатом ADC2 в старшем полуслове и ADC1 в младшем.

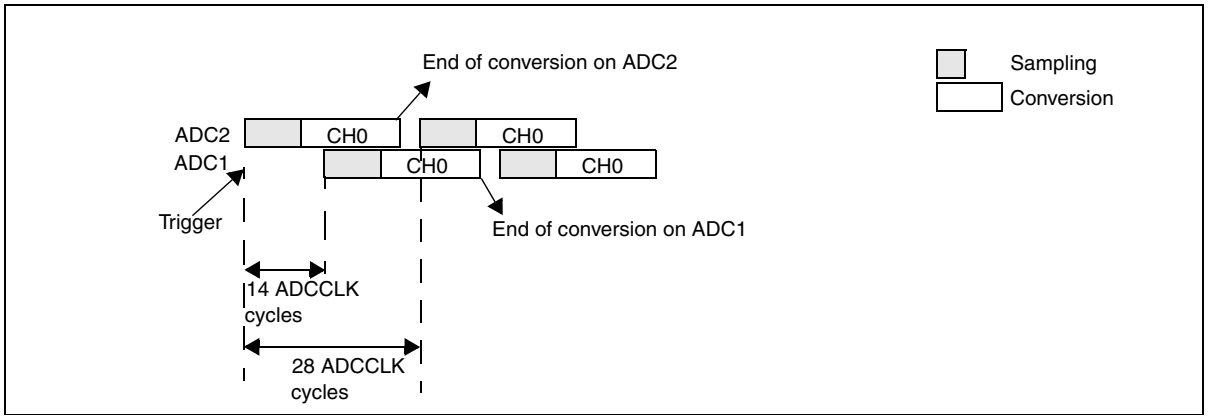
После генерации прерывания `EOC` в ADC1 (если разрешено битом `EOCIE`) ставится запрос 32-бит DMA передачи (если стоит бит `DMA`) регистра `ADC1_DR` в SRAM с результатом ADC2 в старшем полуслове и ADC1 в младшем.

ADC2 автоматически стартует после 28 тактов ADC.

Бит `CONT` ставить нельзя.

NB: В чередующихся режимах появление внешнего запуска вставных каналов преступно.

Медленный чередующийся режим с 1 каналом



11.9.5. Режим попеременного запуска

Этот режим предназначен только для вставных групп (обычно один канал). Внешний сигнал запуска подаётся через мультиплексор вставной группы ADC1. После запуска:

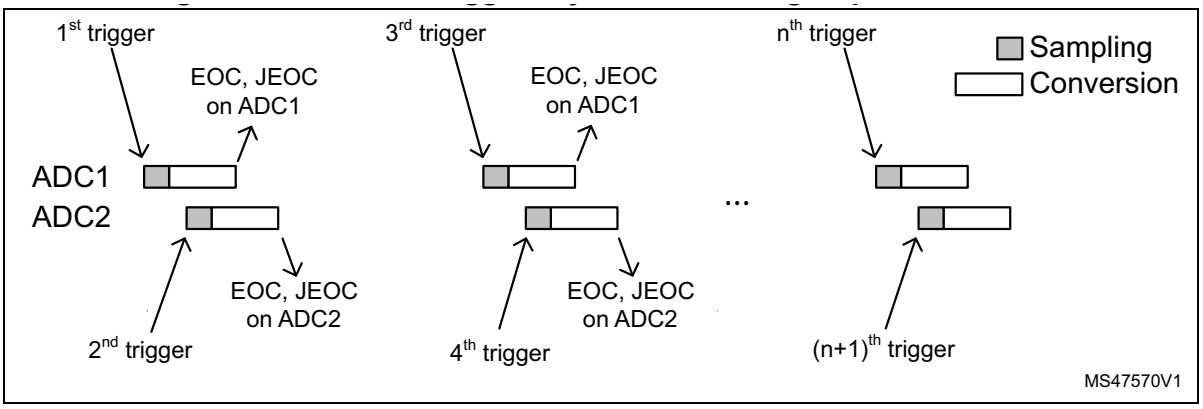
- После первого запуска ADC1 преобразует все каналы своей вставной группы.
- После второго запуска ADC2 преобразует все каналы своей вставной группы.
- И так далее.

Прерывание `JEOC`, если разрешено, генерируется после обработки всей вставной группы ADC1.

Прерывание `JEOC`, если разрешено, генерируется после обработки всей вставной группы ADC2.

Следующий сигнал запуска после завершения обеих групп возобновляет обработку вставных каналов с ADC1.

Попеременный режим с вставными группами на каждом АЦП



В прерывном вставном режиме обоих АЦП:

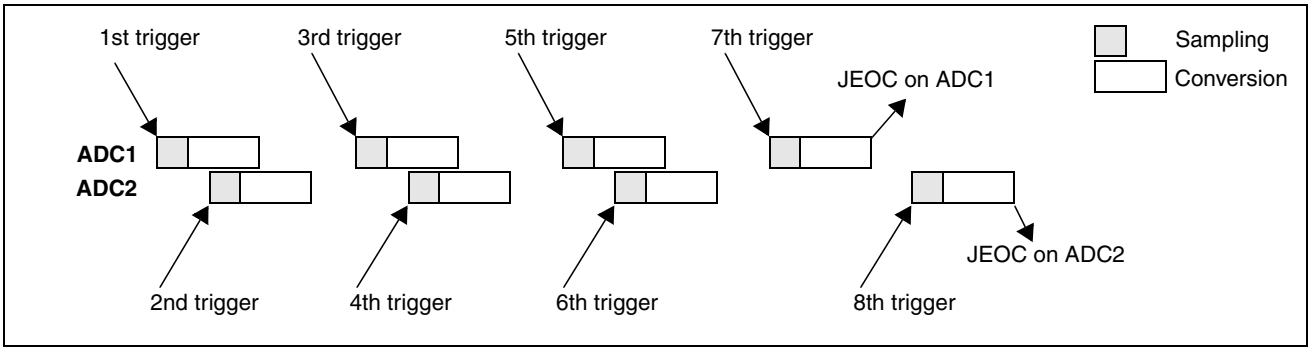
- После первого запуска ADC1 преобразует первый канал своей вставной группы.
- После второго запуска ADC2 преобразует первый канал своей вставной группы.
- И так далее.

Прерывание JEOC, если разрешено, генерируется после обработки всей вставной группы ADC1.

Прерывание JEOC, если разрешено, генерируется после обработки всей вставной группы ADC2.

Следующий сигнал запуска после завершения обеих групп возобновляет обработку вставных каналов с ADC1.

Попеременный прерывный режим с вставными группами по 4 на каждом АЦП



11.9.6. Независимый режим

Тут оба АЦП работают раздельно, без всякой синхронизации.

11.9.7. Комбинированный регулярно/вставной одновременный режим

Есть возможность прерывать одновременное преобразование регулярной группы одновременным преобразованием вставной.

NB: В этом режиме время выборки одновременно обрабатываемых на ACD1 и ADC2 каналов должно точно совпадать.

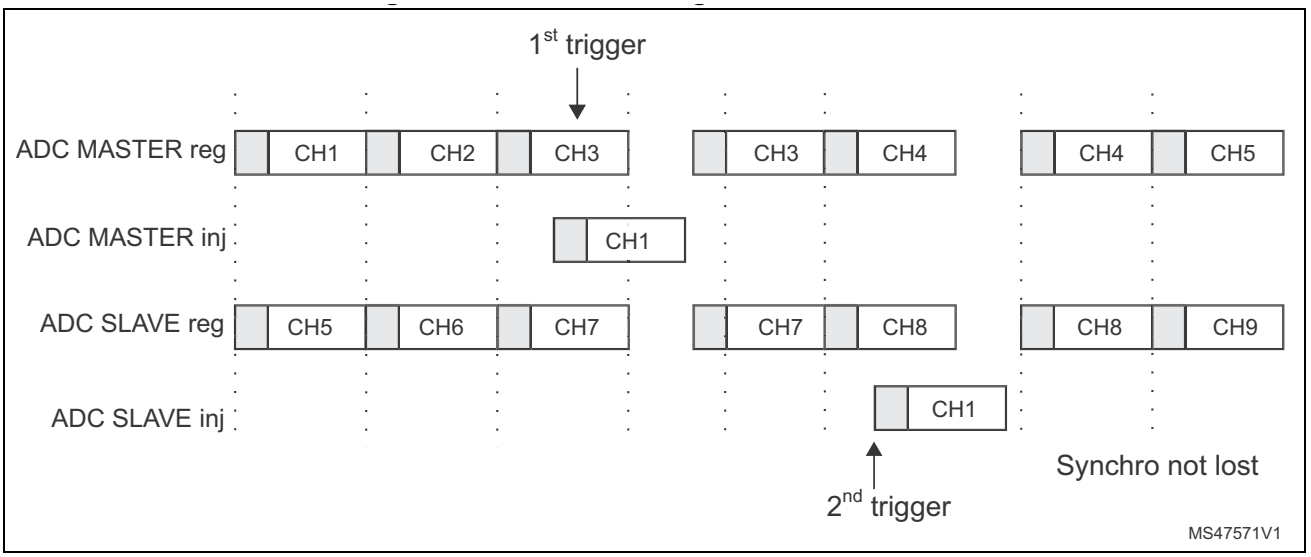
11.9.8. Регулярный одновременный + Попеременный режим

Есть возможность прерывать одновременное преобразование регулярной группы запуском попеременного преобразования вставной.

Попеременное вставное преобразование начинается сразу после сигнала запуска. Если регулярное преобразование уже работает, то оно останавливается на обоих каналах (ведущем и ведомом) и синхронно восстанавливается после завершения вставной группы.

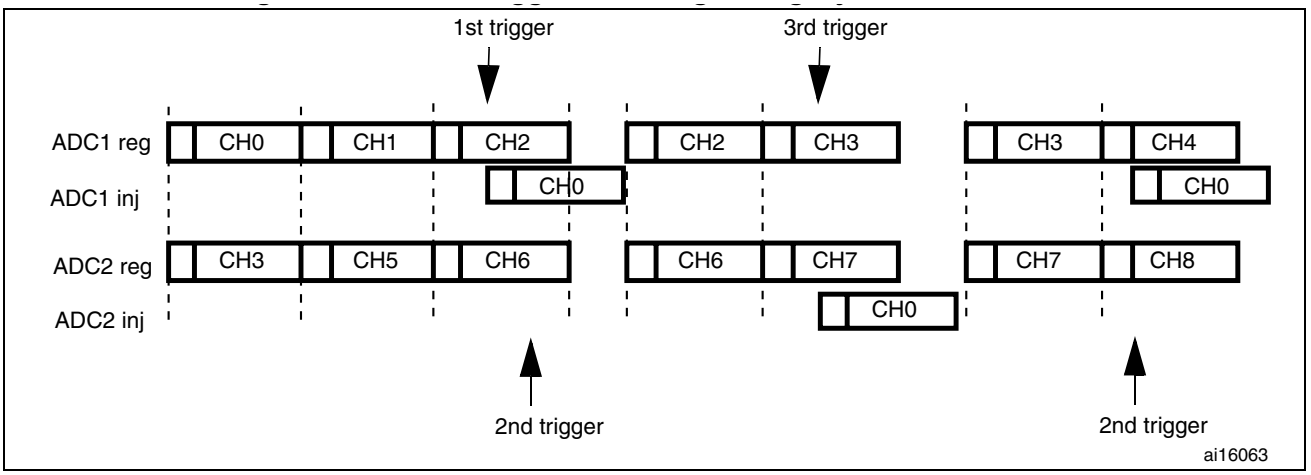
NB: В этом режиме время выборки одновременно обрабатываемых на ACD1 и ADC2 каналов должно точно совпадать.

Попеременный + Регулярный одновременный



Сигнал запуска, пришедший во время обработки вставной последовательности, прервавшей регулярную, игнорируется. На следующей картинке игнорируется второй запуск.

Запуск во время обработки вставной группы

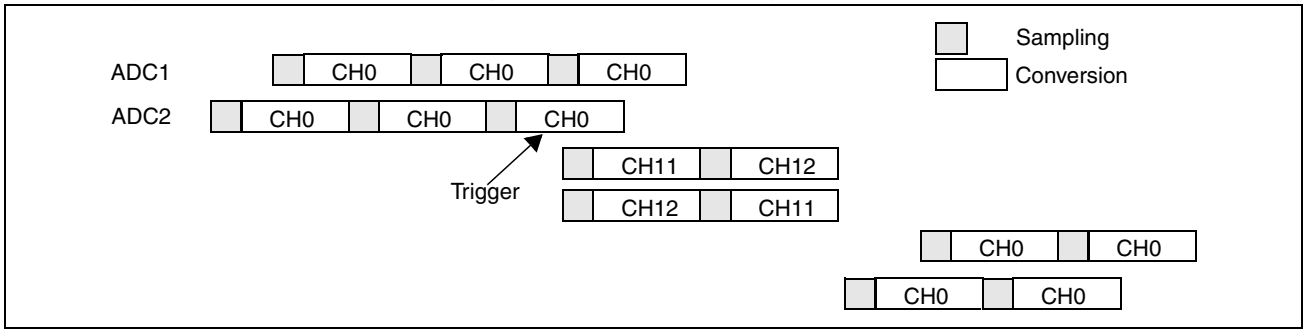


11.9.9. Вставной одновременный + Чередующийся режим

Есть возможность прерывать чередующееся преобразование запуском вставной. Восстанавливается оно после завершения вставной последовательности.

NB: Если предделитель тактов ADC равен 4, то в чередующемся режиме равномерное распределение времён выборки не восстанавливается: 8 и 6 тактов ADC вместо 7 с последующими 7 тактами.

Чередующийся режим одного канала и вставной CH11 и CH12



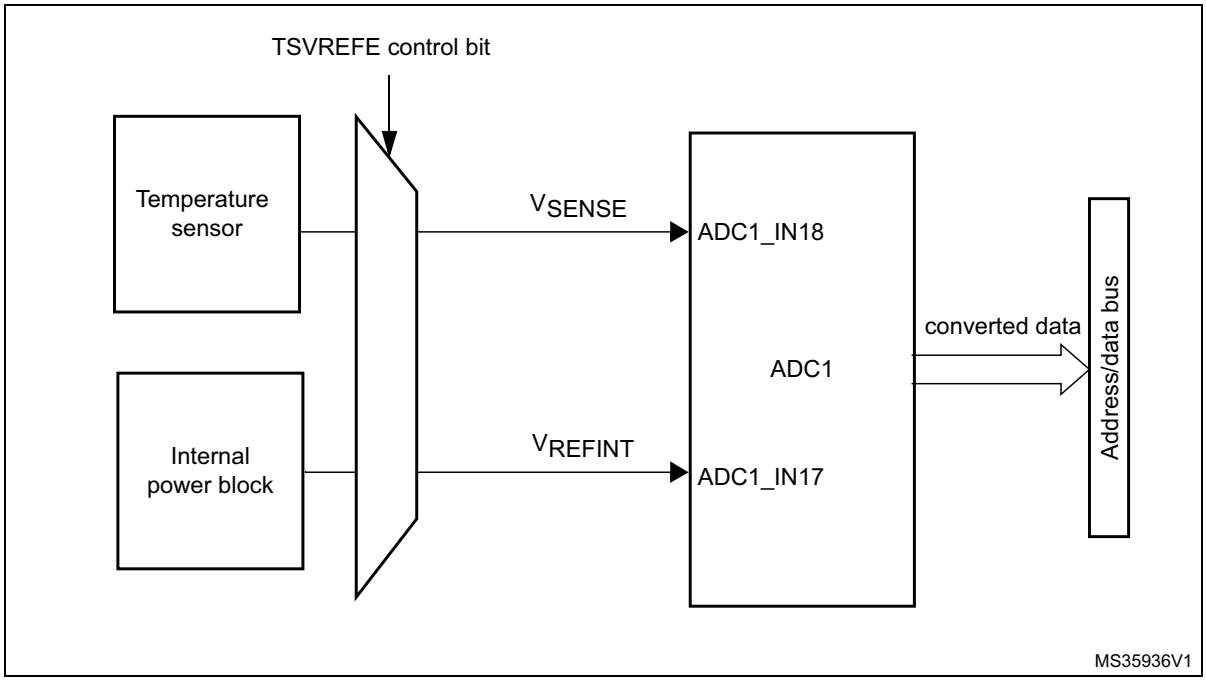
11.10. Датчик температуры

Встроенный датчик температуры подключён к входному каналу ADCx_IN16. Рекомендуемое время выборки равно 17.1 µs.

NB: Бит TSVREFE разрешает оба внутренних канала: ADCx_IN16 (датчик температуры) и ADCx_IN17 (V_{REFINT}).

Выходное напряжение датчика температуры линейно температуре. Смещение этой линии зависит от кристалла и может достигать 45 °C. Так что им можно поглядывать на колебания температуры, а точно измерять температуру нужно внешним датчиком.

Блок-схема датчика температуры и V_{REFINT}



Чтение температуры

Надо:

1. Выбрать канал ADCx_IN16.
2. Выбрать время выборки 17.1 µs
3. Установить бит TSVREFE в регистре ADC_CR2 для включения датчика.
4. Запустить АЦП битом ADON (или снаружи).
5. Прочитать данные V_{SENSE} из регистра АЦП
6. Получить температуру по формуле:

$$T\ (°C) = \{(V_{25} - VS_{ENSE}) / Avg\_Slope\} + 25.$$

где,

V_{25} = значение V_{SENSE} для 25° C и

Avg_Slope = Средний наклон кривой температуры (в mV/°C или µV/°C).

Действительные значения V_{25} and Avg_Slope см. в Электрических характеристиках.

NB: Для уменьшения задержки включения V_{SENSE} и АЦП биты ADON и TSVREFE нужно ставить одновременно.

11.11. Прерывания АЦП

Разрешённые прерывания генерируются для конца преобразования групп и аналогового сторожа.

NB: Прерывания ADC1 и ADC2 имеют один вектор, а ADC3 отдельный вектор.

В регистре ADC_SR есть не связанные с прерываниями флаги:

- JSTRT (Старт преобразования вставных групп)
- STRT (Старт преобразования регулярных групп)

Таблица 71. Прерывания АЦП.

Прерывание	Флаг	Бит разрешения
Конец преобразования регулярной группы	EOC	EOCIE
Конец преобразования вставной группы	JEOC	JEOCIE
Бит статуса Аналогового сторожа встал	AWD	AWDIE

11.12. Регистры АЦП

Доступны только словами.

11.12.1. Регистр состояния АЦП (ADC_SR)

Смещение адреса: 0x00

По сбросу: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											STRT	JSTRT	JEOC	EOC	AWD
											rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

— Биты 31:5 Резерв, не трогать.

— Бит 4 **STRT:** Флаг старта регулярных каналов.
Ставится аппаратно, снимается программно записью 0.
0: Не стартовали
1: Стартовали

— Бит 3 **JSTRT:** Флаг старта вставных каналов.
Ставится аппаратно, снимается программно записью 0.
0: Не стартовали
1: Стартовали

— Бит 2 **JEOC:** Конец преобразования вставных каналов.
Ставится аппаратно, снимается программно записью 0.
0: Нет конца
1: Есть конец

— Бит 1 **EOC:** Конец преобразования.
Ставится аппаратно, снимается программно записью 0.
0: Нет конца
1: Есть конец

— Бит 0 **AWD:** Флаг аналогового сторожа.
Ставится аппаратно, снимается программно записью 0.
0: Не случалось
1: Случилось

11.12.2. Регистр 1 управления АЦП (ADC_CR1)

Смещение адреса: 0x04

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								AWDEN	JAWDEN	Reserved		DUALMOD[3:0]			
								rw	rw			rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.

— Бит 23 **AWDEN:** Разрешение сторожа для регулярных каналов.

Ставится и снимается программно.

0: Нельзя

1: Можно

— Бит 22 **JAWDEN:** Разрешение сторожа для вставных каналов.

Ставится и снимается программно.

0: Нельзя

1: Можно

— Биты 21:20 Резерв, не трогать.

— Биты 19:16 **DUALMOD[3:0]**: Выбор парного режима.

Ставится и снимается программно.

0000: Независимый режим.

0001: Регулярный одновременный + Вставной одновременный.

0010: Регулярный одновременный + Попеременный запуск.

0011: Вставной одновременный + Быстрый чередующийся.

0100: Вставной одновременный + Медленный чередующийся.

0101: Вставной одновременный

0110: Регулярный одновременный

0111: Быстрый чередующийся

1000: Медленный чередующийся

1001: Попеременный запуск

NB: В ADC2 и ADC3 это резерв.

В парном режиме изменение конфигурации каналов запускает их от начала, что может привести к потере синхронизации. Так что парный режим надо предварительно отключить.

— Биты 15:13 **DISCNUM[2:0]**: Счётчик каналов Прерывного режима

Ставится и снимается программно.

000: 1 канал

001: 2 канала

…….

111: 8 каналов

— Бит 12 **JDISCEN**: Прерывный режим вставных каналов

Ставится и снимается программно.

0: Нельзя

1: Можно

— Бит 11 **DISCEN**: Прерывный режим регулярных каналов.

Ставится и снимается программно.

0: Нельзя

1: Можно

— Бит 10 **JAUTO:** Автоматическая обработка вставной группы.

Ставится и снимается программно.

0: Нельзя

1: Можно

— Бит 9 **AWDSGL:** Разрешение сторожа для всех или одного канала из битов AWDCH[4:0].

Ставится и снимается программно.

0: Все каналы

1: Один канал

— Бит 8 **SCAN:** Режим Сканирования каналов из регистров ADC_SQRx или ADC_JSQRx.

Ставится и снимается программно.

0: Нельзя

1: Можно

NB: Разрешённое (битами EOCIE или JEOCIE) прерывание EOC или JEOC возникает только после обработки последнего канала последовательности.

— Бит 7 **JEOCIE**: Разрешение прерывания вставных каналов по флагу JEOC.

Ставится и снимается программно.

0: Нельзя

1: Можно

— Бит 6 **AWDIE**: Разрешение прерывания Аналогового сторожа.
Ставится и снимается программно.
0: Нельзя
1: Можно

— Бит 5 **EOCIE:** Разрешение прерывания для EOC.
Ставится и снимается программно.
0: Нельзя
1: Можно

— Биты 4:0 **AWDCH[4:0]:** Выбор каналов АЦП для Аналогового сторожа.
Ставится и снимается программно.
00000: Канал 0
00001: Канал 1
….
01111: Канал 15
10000: Канал 16
10001: Канал 17
Остальные значения в резерве.
NB: Каналы 16 и 17 в ADC1 внутренне подключены к датчику температуры и V_{REFINT}.
Каналы 16 и 17 в ADC2 внутренне подключены к V_{SS}. Каналы 9, 14, 15, 16 и 17 в ADC3 подключены к V_{SS}.

11.12.3.Регистр 2 управления АЦП (ADC_CR2)

Смещение адреса: 0x08

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								TSVREFE	SWSTART	JSWSTART	EXTTRIG	EXTSEL[2:0]			Res.
								rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JEXTTRIG	JEXTSEL[2:0]			ALIGN	Reserved		DMA	Reserved				RSTCAL	CAL	CONT	ADON
rw	rw	rw	rw	rw	Res.		rw					rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.

— Бит 23 **TSVREFE:** Разрешение датчика температуры и V_{REFINT}.
Ставится и снимается программно.
0: Нельзя
1: Можно

— Бит 22 **SWSTART:** Старт обработки группы регулярных каналов
Ставится программно, снимается аппаратно сразу после программного запуска, выбранного в битах EXTSEL[2:0].
0: Сброшено
1: Запуск

— Бит 21 **JSWSTART:** Старт обработки группы вставных каналов
Ставится программно, снимается аппаратно сразу после программного запуска, выбранного в битах JEXTSEL[2:0].
0: Сброшено
1: Запуск

— Бит 20 **EXTTRIG:** Разрешение внешнего запуска регулярных каналов.
Ставится и снимается программно.
0: Нельзя
1: Можно

— Биты 19:17 **EXTSEL[2:0]** Выбор внешнего сигнала запуска обработки регулярных групп
Пишется программно.
Для ADC1 и ADC2:
000: Timer 1 CC1
001: Timer 1 CC2

010: Timer 1 CC3
011: Timer 2 CC2
100: Timer 3 TRGO
101: Timer 4 CC4
110: EXTI line 11/TIM8_TRGO (TIM8_TRGO есть только в устройствах высокой и XL- плотности)
111: SWSTART

Для ADC3:
000: Timer 3 CC1
001: Timer 2 CC3
010: Timer 1 CC3
011: Timer 8 CC1
100: Timer 8 TRGO
101: Timer 5 CC1
110: Timer 5 CC3
111: SWSTART

— Бит 16 Резерв, не трогать.

— Бит 15 **JEXTTRIG**: Разрешение внешнего запуска вставных каналов.
Ставится и снимается программно.
0: Нельзя
1: Можно

— Биты 14:12 **JEXTSEL[2:0]** Выбор внешнего сигнала запуска обработки вставных групп
Пишется программно.
Для ADC1 и ADC2:
000: Timer 1 TRGO
001: Timer 1 CC4
010: Timer 2 TRGO
011: Timer 2 CC1
100: Timer 3 CC4
101: Timer 4 TRGO
110: EXTI line15/TIM8_CC4 (TIM8_CC4 есть только в устройствах высокой и XL- плотности)
111: JSWSTART

Для ADC3:
000: Timer 1 TRGO
001: Timer 1 CC4
010: Timer 4 CC3
011: Timer 8 CC2
100: Timer 8 CC4
101: Timer 5 TRGO
110: Timer 5 CC4
111: JSWSTART

— Бит 11 **ALIGN**: Выравнивание результата.
Ставится и снимается программно.
0: Вправо
1: Влево

— Биты 10:9 Резерв, не трогать.

— Бит 8 **DMA:** Режим ПДП.
Ставится и снимается программно.
0: Нельзя
1: Можно
NB: Только для ADC1 и ADC3.

— Биты 7:4 Резерв, не трогать.
0: Нельзя
1: Можно

— Бит 3 **RSTCAL**: Сброс калибровки.

Ставится программно, снимается аппаратно после инициализации регистров калибровки.

0: Инициализированы

1: Инициализация регистров калибровки

NB: Установка RSTCAL во время преобразования добавляет такты для очистки регистров калибровки.

— Бит 2 **CAL:** Запуск калибровки АЦП.

Ставится программно, снимается аппаратно после завершения калибровки.

0: Калибровка завершена

1: Разрешить калибровку

— Бит 1 **COUNT:** Разрешение непрерывного преобразования.

Ставится и снимается программно. Снятие бита останавливает непрерывное преобразование.

0: Однократное преобразование

1: Непрерывное преобразование

— Бит 0 **ADON:** Вкл./Выкл. АЦП.

Ставится и снимается программно.

Запись 1 при стоящем 0 включает питание АЦП.

Запись 1 при стоящей 1 запускает преобразование. Между включением и запуском нужно выдержать паузу t_{STAB}.

0: Выключает преобразование/калибровку и выключает питание АЦП.

1: Включает АЦП или запускает прерывание

NB: Если одновременно с ADON изменяется любой другой бит, то преобразование не запускается. Это защита от дурака.

11.12.4.Регистр 1 времени выборки АЦП (ADC_SMPR1)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP 15_0	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.

— Биты 19:16 **SMPx[2:0]**: Время выборки канала **x**.

Ставится и снимается программно. Во время цикла выборки эти биты изменять нельзя.

000: 1.5 такта

001: 7.5 тактов

010: 13.5 тактов

011: 28.5 тактов

100: 41.5 такт

101: 55.5 тактов

110: 71.5 такт

111: 239.5 тактов

NB: Каналы 16 и 17 в ADC1 внутренне подключены к датчику температуры и V_{REFINT}.

Каналы 16 и 17 в ADC2 внутренне подключены к V_{SS}. Каналы 9, 14, 15, 16 и 17 в ADC3 подключены к V_{SS}.

11.12.5.Регистр 2 времени выборки АЦП (ADC_SMPR2)

Смещение адреса: 0x10

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
Res.		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP 5_0	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.

— Биты 19:16 **SMPx[2:0]**: Время выборки канала **x**.

Ставится и снимается программно. Во время цикла выборки эти биты изменять нельзя.

000: 1.5 такта
001: 7.5 тактов
010: 13.5 тактов
011: 28.5 тактов
100: 41.5 такт
101: 55.5 тактов
110: 71.5 такт
111: 239.5 тактов

NB: Канал 9 в ADC3 подключен к V_{SS}.

11.12.6.Смещение данных вставных каналов АЦП (ADC_JOFRx) (x=1..4)

Смещение адреса: 0x14 - 0x20

По сбросу: 0x0000 0000

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

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

— Биты 31:12 Резерв, не трогать.

— Биты 11:0 **JOFFSETx[11:0]**: Смещение данных вставного канала x

Пишется программно. Вычитается из результата преобразования вставных каналов, итог пишется в регистр ADC_JDRx.

11.12.7.Верхний порог аналогового сторожа (ADC_HTR)

Смещение адреса: 0x24

По сбросу: 0x0000 0000

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

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

— Биты 31:12 Резерв, не трогать.

— Биты 11:0 **HT[11:0]:** Верхний порог аналогового сторожа

Пишется программно.

NB: Писать можно во время цикла преобразования АЦП. Значение начнёт действовать после завершения следующего преобразования. Запись делается с задержкой, что может создать неопределённость в начало работы нового значения.

11.12.8.Нижний порог аналогового сторожа (ADC_LTR)

Смещение адреса: 0x28

По сбросу: 0x0000 0000

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

— Биты 31:12 Резерв, не трогать.

— Биты 11:0 **LT[11:0]:** Нижний порог аналогового сторожа

Пишется программно.

NB: Писать можно во время цикла преобразования АЦП. Значение начнёт действовать после завершения следующего преобразования. Запись делается с задержкой, что может создать неопределённость в начало работы нового значения.

11.12.9.Регистр 1 регулярной последовательности (ADC_SQR1)

Смещение адреса: 0x2C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								L[3:0]				SQ16[4:1]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ16_0	SQ15[4:0]					SQ14[4:0]					SQ13[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.

— Биты 23:20 **L[3:0]**. Длина регулярной последовательности.

Пишется программно.

— Биты 19:15

— Биты 14:10

— Биты 9:5

— Биты 4:0 **SQx[4:0]:** Номер аналогового канала (0-17) на месте **x** (x=16-13) последовательности.

Пишется программно.

11.12.10.Регистр 2 регулярной последовательности (ADC_SQR2)

Смещение адреса: 0x30

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ12[4:0]					SQ11[4:0]					SQ10[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ10_0	SQ9[4:0]					SQ8[4:0]					SQ7[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать.

— Биты 29:25

— Биты 24:20

— Биты 19:15

— Биты 14:10

— Биты 9:5

— Биты 4:0 **SQx[4:0]:** Номер аналогового канала (0-17) на месте **x** (x=12-7) последовательности.

Пишется программно.

11.12.11.Регистр 3 регулярной последовательности (ADC_SQR3)

Смещение адреса: 0x34

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ6[4:0]					SQ5[4:0]					SQ4[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4_0	SQ3[4:0]					SQ2[4:0]					SQ1[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать.
— Биты 29:25
— Биты 24:20
— Биты 19:15
— Биты 14:10
— Биты 9:5
— Биты 4:0 **SQx[4:0]:** Номер аналогового канала (0-17) на месте **x** (x=6-1) последовательности.
Пишется программно.

11.12.12.Регистр вставной последовательности (ADC_JSQR)

Смещение адреса: 0x38

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										JL[1:0]		JSQ4[4:1]			
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ4_0	JSQ3[4:0]					JSQ2[4:0]					JSQ1[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:22 Резерв, не трогать.
— Биты 21:20 **JL[2:0]**. Длина вставной последовательности.
Пишется программно.
00: 1 преобразование
01: 2 преобразования
10: 3 преобразования
11: 4 преобразования
— Биты 19:15 **JSQ4[4:0]**: Номер аналогового канала (0..17) 4-го преобразования (при JL[1:0]=3)(1)
Пишется программно.
— Биты 14:10 **JSQ3[4:0]**: 3е преобразование (при JL[1:0]=3)
— Биты 9:5 **JSQ2[4:0]**: 2е преобразование (при JL[1:0]=3)
— Биты 4:0 **JSQ1[4:0]**: 1е преобразование (при JL[1:0]=3).

1. При JL=3 (4 канала) порядок обработки такой: JSQ1[4:0] >> JSQ2[4:0] >> JSQ3[4:0] >> JSQ4[4:0]
При JL=2 (3 канала) порядок обработки такой: JSQ2[4:0] >> JSQ3[4:0] >> JSQ4[4:0]
При JL=1 (2 канала) порядок обработки такой: JSQ3[4:0] >> JSQ4[4:0]
При JL=0 (1 канал) преобразуется только канал JSQ4[4:0]

11.12.13.Регистры x данных вставных каналов (ADC_JDRx) (x=1..4)

Смещение адреса: 0x3C - 0x48

По сбросу: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:16 Резерв, не трогать.
— Биты 15:0 **JDATA[15:0]**. Выравненные влево или вправо данные вставного канала.
Только чтение.

11.12.14.Регистр данных регулярных каналов (ADC_DR)

Смещение адреса: 0x4C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC2DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:16 **ADC2DATA[15:0]:** Данные регулярного канала ADC2.
Только чтение.
В парном режиме с ADC1 содержит данные регулярного канала ADC2.
В ADC2 и ADC3 не используются.

— Биты 15:0 **DATA[15:0]**: Выравненные влево или вправо регулярные данные
Только чтение.

11.12.15.Карта регистров АЦП

0x00	ADC_SR	Reserved																												STRT	JSTRT	JEOC	EOC	AWD
	Reset value																													0	0	0	0	0
0x04	ADC_CR1	Reserved														AWDEN	JAWDEN	Reserved	DUALMOD [3:0]				DISC NUM [2:0]			JDISCEN	DISCEN	JAUTO	AWD SGL	SCAN	JEOC IE	AWDIE	EOCIE	AWDCH[4:0]					
	Reset value															0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	ADC_CR2	Reserved														TSVREFE	SWSTART	JSWSTART	EXTTRIG	EXTSEL [2:0]			Reserved	JEXTTRIG	JEXTSEL [2:0]			ALIGN	Reserved	DMA	Reserved					RSTCAL	CAL	CONT	ADON
	Reset value															0	0	0	0	0	0	0		0	0	0	0	0		0					0	0	0	0
0x0C	ADC_SMPR1	Sample time bits SMPx_x																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	ADC_SMPR2	Sample time bits SMPx_x																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	ADC_JOFR1	Reserved																												JOFFSET1[11:0]				
	Reset value																													0	0	0	0	0	0	0	0	0	0	0
0x18	ADC_JOFR2	Reserved																												JOFFSET2[11:0]				
	Reset value																													0	0	0	0	0	0	0	0	0	0	0
0x1C	ADC_JOFR3	Reserved																												JOFFSET3[11:0]				
	Reset value																													0	0	0	0	0	0	0	0	0	0	0
0x20	ADC_JOFR4	Reserved																												JOFFSET4[11:0]				
	Reset value																													0	0	0	0	0	0	0	0	0	0	0
0x24	ADC_HTR	Reserved																												HT[11:0]				
	Reset value																													0	0	0	0	0	0	0	0	0	0	0
0x28	ADC_LTR	Reserved																												LT[11:0]				
	Reset value																													0	0	0	0	0	0	0	0	0	0	0
0x2C	ADC_SQR1	Reserved																	L[3:0]				SQ16[4:0] 16th conversion in regular sequence bits					SQ15[4:0] 15th conversion in regular sequence bits					SQ14[4:0] 14th conversion in regular sequence bits					SQ13[4:0] 13th conversion in regular sequence bits				
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0x30	ADC_SQR2	Reserved		SQ12[4:0] 12th conversion in regular sequence bits					SQ11[4:0] 11th conversion in regular sequence bits					SQ10[4:0] 10th conversion in regular sequence bits					SQ9[4:0] 9th conversion in regular sequence bits					SQ8[4:0] 8th conversion in regular sequence bits					SQ7[4:0] 7th conversion in regular sequence bits				
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	ADC_SQR3	Reserved		SQ6[4:0] 6th conversion in regular sequence bits					SQ5[4:0] 5th conversion in regular sequence bits					SQ4[4:0] 4th conversion in regular sequence bits					SQ3[4:0] 3rd conversion in regular sequence bits					SQ2[4:0] 2nd conversion in regular sequence bits					SQ1[4:0] 1st conversion in regular sequence bits				
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	ADC_JSQR	Reserved										JL[1:0]		JSQ4[4:0] 4th conversion in injected sequence bits					JSQ3[4:0] 3rd conversion in injected sequence bits					JSQ2[4:0] 2nd conversion in injected sequence bits					JSQ1[4:0] 1st conversion in injected sequence bits				
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	ADC_JDR1	Reserved																		JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	ADC_JDR2	Reserved																		JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	ADC_JDR3	Reserved																		JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	ADC_JDR4	Reserved																		JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	ADC_DR	ADC2DATA[15:0]																Regular DATA[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

12. Цифро-аналоговый преобразователь (DAC)

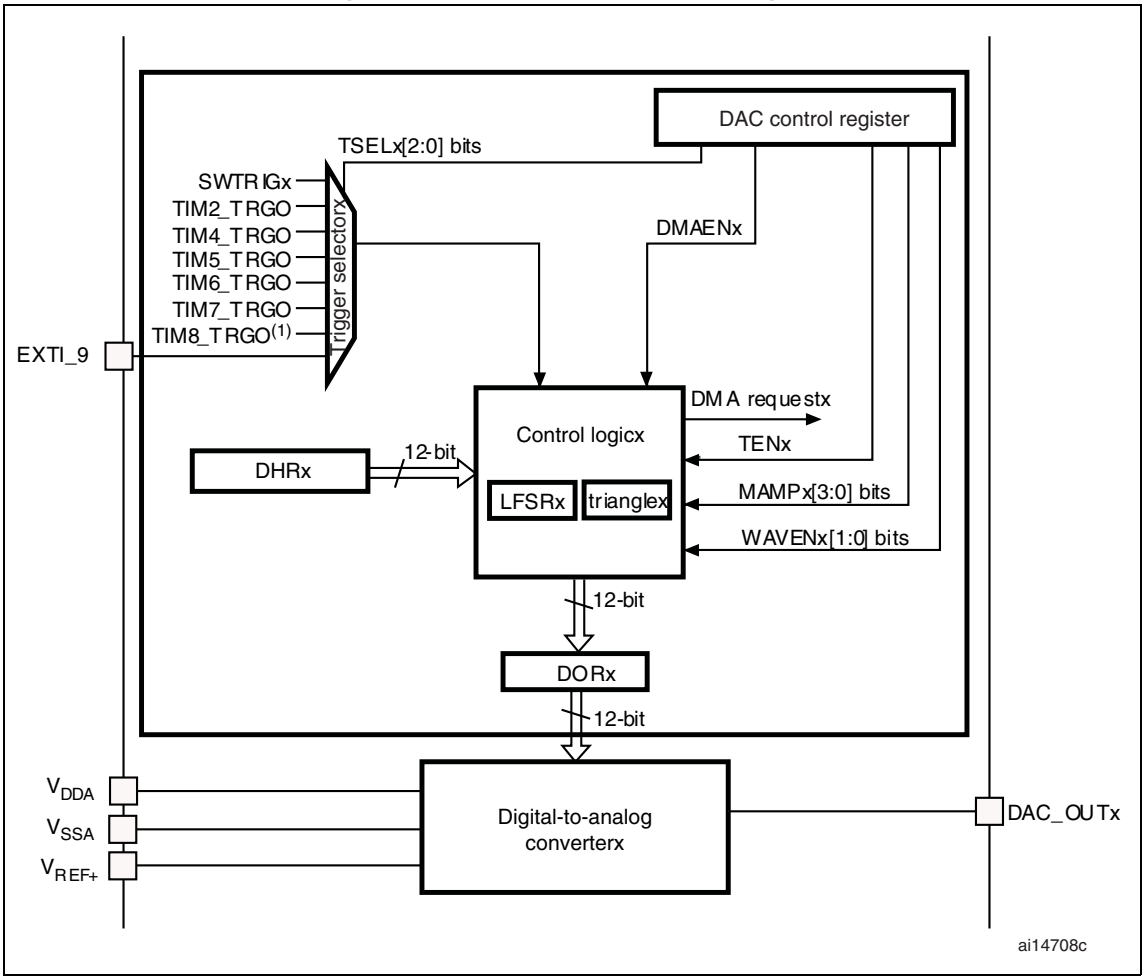
12.1. Введение в ЦАП

ЦАП можно включать в 8- или 12-бит режим и использовать совместно с ПДП. В 12-бит режиме данные могут быть выравненными влево или вправо. ЦАП имеет два канала со своими конверторами. В парном режиме преобразование может выполняться независимо или одновременно, когда оба канала группируются вместе для синхронного обновления операции. Ножка опорного напряжения V_{REF+} общая для АЦП и ЦАП.

12.2. Основные свойства ЦАП

- Два конвертора ЦАП со своими выходными каналами
- Правое или левое выравнивание данных в 12-бит режиме
- Возможность синхронного обновления
- Генерация шумоподобного сигнала
- Генерация треугольного сигнала
- Независимое или одновременное парное преобразование
- Возможность ПДП для обоих каналов
- Внешний запуск преобразования
- Вход опорного напряжения V_{REF+}

Блок-схема ЦАП.



1. В сетевых устройствах ножка TIM8_TRGO заменена на TIM3_TRGO.

Имя	Тип	Заметки
V_{REF+}	Опорное напряжение	2.4 V ≤V_{REF+} ≤V_{DDA} (3.3 V)
V_{DDA}	Аналоговое питание	Аналоговое питание
V_{SSA}	Аналоговая земля	Аналоговая земля
DAC_OUTx	Аналоговый выход	Аналоговый выход

NB. После включения канала ЦАП к его выходу (DAC_OUTx) автоматически подключается соответствующая ножка GPIO (PA4 или PA5). Во избежание паразитных утечек ножки PA4 или PA5 нужно включить в аналоговый режим (AIN).

12.3. Основные свойства ЦАП

12.3.1. Включение канала ЦАП

Каналы ЦАП включаются раздельно битами ENx в регистре DAC_CR. Разрешается канал по истечении времени запуска t_{WAKEUP}.

NB: Бит ENx включает только аналоговую часть канала (DAC channelx), цифровой интерфейс включён даже при снятом бите ENx.

12.3.2. Включение выходных буферов ЦАП

Для уменьшения выходного сопротивления и ради прямого управления внешней нагрузкой имеются два выходных буфера, которые включаются битами BOFFx в регистре DAC_CR.

12.3.3. Формат данных ЦАП

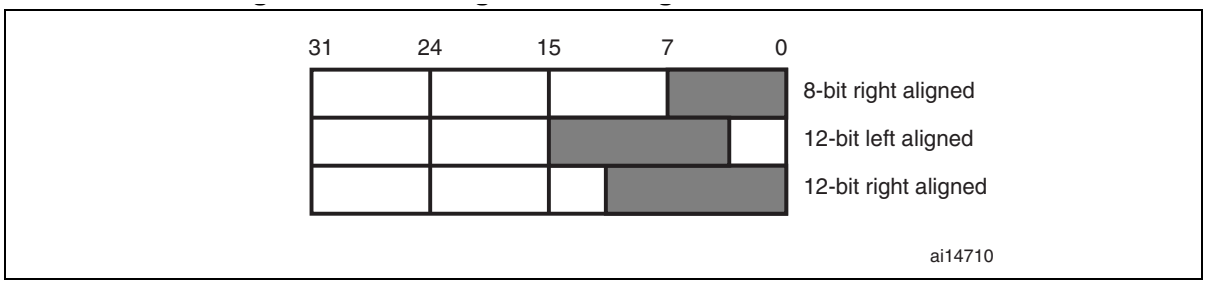
В зависимости от режима данные надо писать в определённые регистры:

• В одиночном режиме канала DAC channelx:

- 8-бит правое выравнивание: данные пишутся в биты DAC_DHR8Rx[7:0], хранятся в DHRx[11:4]
- 12-бит левое выравнивание: данные пишутся в биты DAC_DHR12Lx[15:4], хранятся в DHRx[11:0]
- 12-бит правое выравнивание: данные пишутся в биты DAC_DHR12Rx[11:0], хранятся в DHRx[11:0]

Данные из регистров DAC_DHRyyyx сдвигаются и пишутся в регистры DHRx. Регистры DHRx пишутся в регистры DORx программно или по внешнему сигналу.

Данные в одиночном режиме

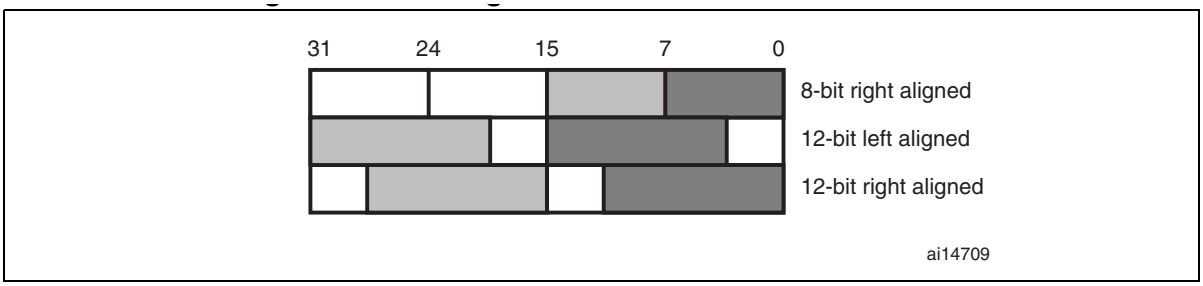


• В парном режиме каналов DAC:

- 8-бит правое выравнивание: DAC channel1 пишется в биты DAC_DHR8RD [7:0] (хранится в DHR1[11:4]), DAC channel2 пишется в биты DAC_DHR8RD [15:8] (хранится в DHR2[11:4])
- 12-бит левое выравнивание: DAC channel1 пишется в биты DAC_DHR12LD [15:4] (хранится в DHR1[11:0]), DAC channel2 пишется в биты DAC_DHR12LD [31:20] (хранится в DHR2[11:0])
- 12-бит правое выравнивание: DAC channel1 пишется в биты DAC_DHR12RD [11:0] (хранится в DHR1[11:0]), DAC channel2 пишется в биты DAC_DHR12LD [27:16] (хранится в DHR2[11:0])

Данные из регистра DAC_DHRyyyD сдвигаются и пишутся в регистры DHR1 и DHR2. Регистры DHR1 и DHR2 пишутся в регистры DOR1 и DOR2 программно или по внешнему сигналу.

Данные в парном режиме



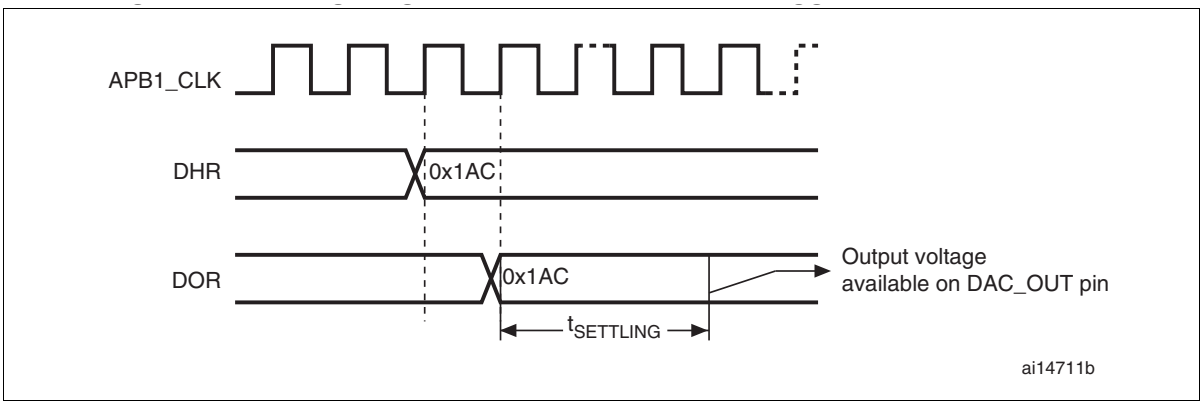
12.3.4. Преобразование ЦАП

Регистры DAC_DORx напрямую недоступны и писать для DAC channelx нужно через регистр DAC_DHRx (в DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12LD или DAC_DHR12LD).

Регистр DAC_DHRx автоматически передаётся в регистр DAC_DORx через один цикл APB1, если не выбран аппаратный запуск (бит TENx в регистре DAC_CR сброшен). Но, при аппаратном запуске (бит TENx в регистре DAC_CR) и появлении сигнала запуска данные передаются на три такта APB1 позже.

После перезаписи DAC_DHRx в DAC_DORx аналоговый сигнал появляется через $t_{SETTLING}$, зависящее от напряжения питания и выходного напряжения.

Временная диаграмма с выключенным запуском TEN=0



12.3.5. Выходное напряжение ЦАП

Преобразование между 0 и V_{REF+} линейное. Выходное напряжение вычисляется по формуле:

$$DACoutput = V_{REF} \times \frac{DOR}{4096}$$

12.3.6. Выбор запуска ЦАП

При стоящем бите TENx источник запуска выбирается битами TSELx[2:0].

Таблица 74. Внешний запуск.

Источник	Тип	TSEL[2:0]
Timer 6 TRGO	Встроенные таймеры	000
Timer 3 TRGO в сетевых устройствах или Timer 8 TRGO в устройствах высокой и XL-плотности		001
Timer 7 TRGO		010
Timer 5 TRGO		011
Timer 2 TRGO		100
Timer 4 TRGO		101
EXTI line9	Внешняя линия	110
SWTRIG	Программный запуск	111

Через три цикла APB1 после появления переднего фронта сигнала TRGO от выбранного таймера или EXTI line 9 данные из регистра DAC_DHRx передаются в регистр DAC_DORx.

При программном запуске преобразование запускается сразу после установки бита SWTRIG. SWTRIG сбрасывается аппаратно сразу после записи DAC_DORx из регистра DAC_DHRx.

NB: Биты TSELx[2:0] нельзя менять при стоящем бите ENx.

Программный запуск занимает только один такт APB1 на передачу DAC_DHRx — DAC_DORx.

12.3.7. Запрос DMA

Для двух каналов ЦАП используются два канала ПДП.

Запрос DMA генерируется только при внешнем запуске (не программном) и установленном бите DMAENx. Затем значение регистра DAC_DHRx передаётся в регистр DAC_DORx.

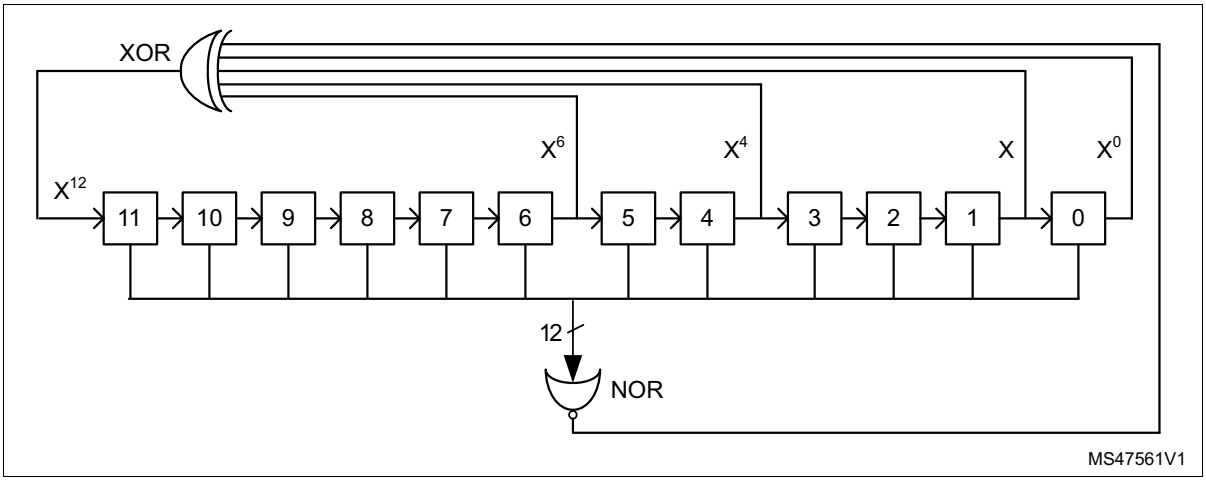
В парном режиме, если установлены оба бита DMAENx, то генерируются два запроса DMA. Если нужен только один запрос DMA, то и ставить нужно один бит DMAENx. Так в парном режиме можно два канала DAC передавать одним каналом DMA по одному запросу.

Запросы DMA от АЦП не имеют очереди, так что поступивший внешний запуск при необслуженном предыдущем запросе обслужен не будет и ошибки не появится.

12.3.8. Генерация шума

Шумоподобный сигнал переменной амплитуды использует Регистр Сдвига с Линейной обратной связью (LFSR). Генератор шума DAC выбирается установкой WAVEx[1:0] в “01”. Предзагруженное значение регистра LFSR равно 0xAAA. Этот регистр обновляется каждые три цикла APB1 после запуска следуя особому алгоритму.

Алгоритм вычисления LFSR.

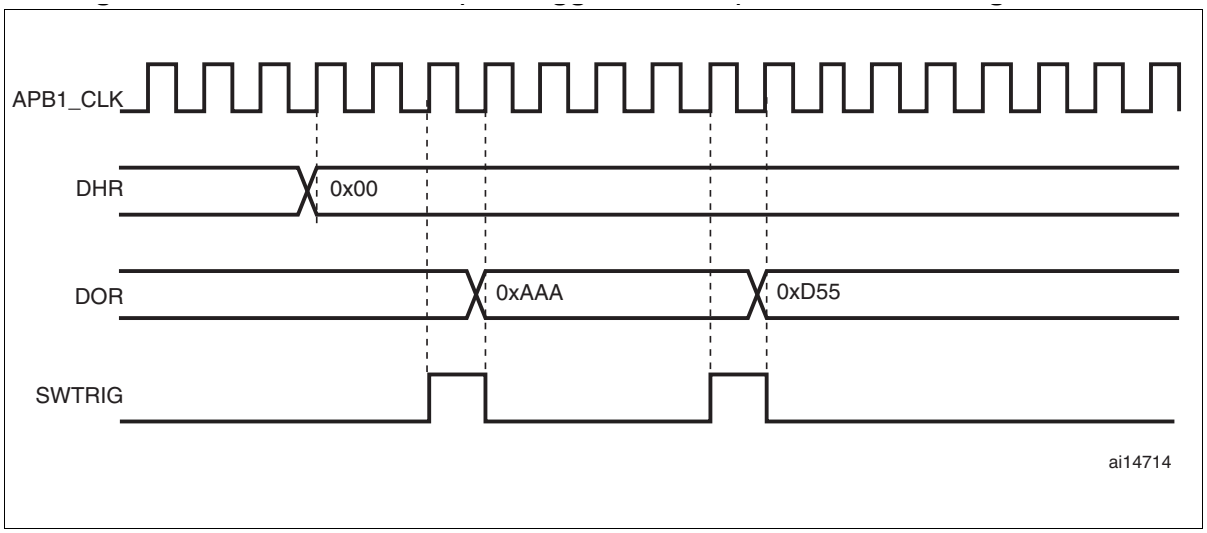


Значение LFSR можно частично или полностью маскировать с помощью битов MAMPx[3:0] в регистре DAC_CR. Оно добавляется к регистру DAC_DHRx без переполнения и затем пересылается в регистр DAC_DORx.

Если LFSR равен 0x0000, то в него вставляется ‘1’ (антиблокировка).

Сбрасывается генератор LFSR сбросом битов WAVEx[1:0].

Программный запуск ЦАП с включённым LFSR.



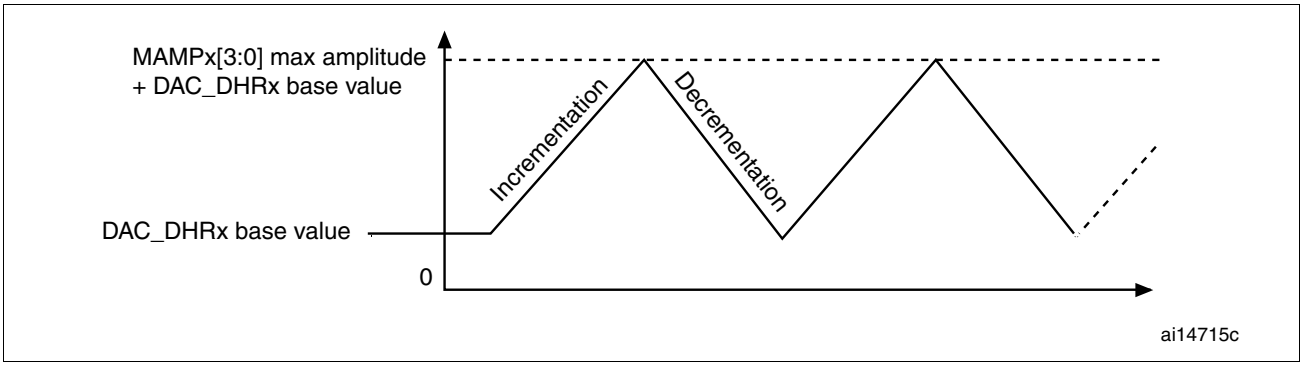
NB. Запуск генератора шума нужно разрешать битом TENx в регистре DAC_CR.

12.3.9. Генерация треугольного сигнала

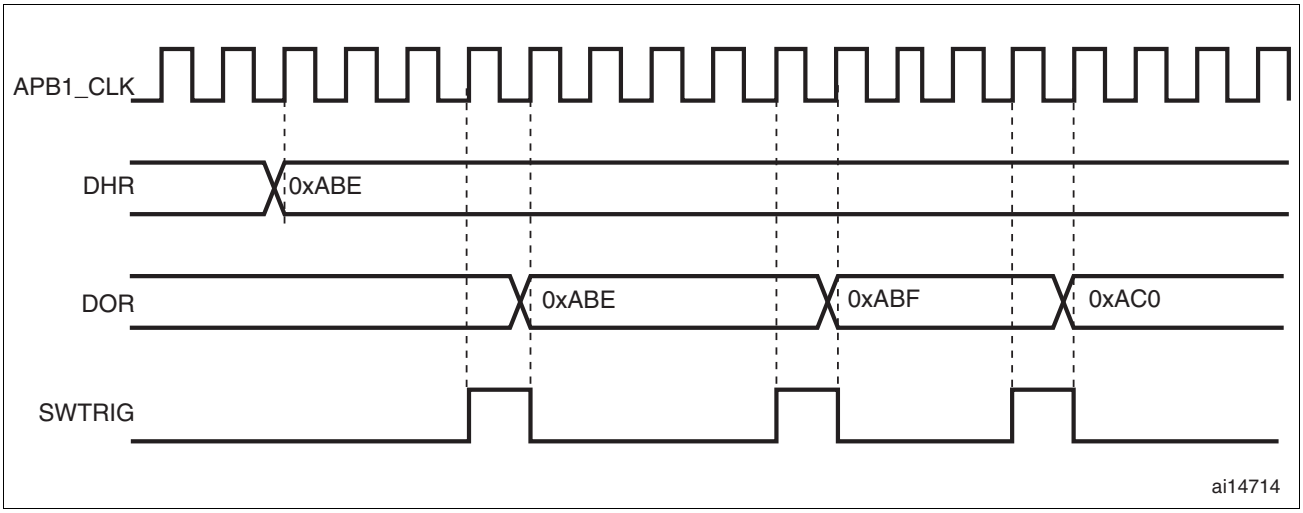
К постоянному или медленно меняющемуся выходному сигналу ЦАП можно добавлять треугольный сигнал малой амплитуды. Генератор треугольника выбирается записью “10” в WAVEx[1:0]. Амплитуда назначается с помощью битов MAMPx[3:0] в регистре DAC_CR. Внутренний реверсивный счётчик инкрементируется каждые три такта APB1 после каждого запуска. Его значение добавляется к регистру DAC_DHRx без переполнения и сумма пишется в DAC_DORx. Счётчик инкрементируется до максимальной амплитуды, заданной битами MAMPx[3:0] и затем инкрементируется до 0, затем снова инкрементируется и т. д.

Останавливается генератор сбросом битов WAVEx[1:0].

Генерация треугольника.



Программный запуск ЦАП с включённым генератором треугольника.



NB. Запуск генератора шума нужно разрешать битом TENx в регистре DAC_CR.
Биты MAMPx[3:0] нужно писать до включение ЦАП, иначе их не изменить.

12.4. Парный режим ЦАП

12.4.1. Независимый запуск ЦАП без генераторов

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать разные источники запуска в битах TSEL1[2:0] и TSEL2[2:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 после сигнала запуска DAC channel1 регистр DHR1 пишется в DAC_DOR1.

Через три такта APB1 после сигнала запуска DAC channel2 регистр DHR2 пишется в DAC_DOR2.

12.4.2. Независимый запуск ЦАП с одинаковым LFSR

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать разные источники запуска в битах TSEL1[2:0] и TSEL2[2:0]
- Записать “01” в биты WAVEx[1:0] обоих каналов и одинаковую маску LFSR в биты MAMPx[3:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 после сигнала запуска DAC channel1 счётчик LFSR1 прибавляется к регистру DHR1 и он пишется в DAC_DOR1. LFSR1 обновляется.

Через три такта APB1 после сигнала запуска DAC channel2 счётчик LFSR2 прибавляется к регистру DHR2 и он пишется в DAC_DOR2. LFSR2 обновляется.

12.4.3. Независимый запуск ЦАП с разными LFSR

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать разные источники запуска в битах TSEL1[2:0] и TSEL2[2:0]
- Записать “01” в биты WAVEx[1:0] обоих каналов и разные маски LFSR в битах MAMP1[3:0] и MAMP2[3:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 после сигнала запуска DAC channel1 счётчик LFSR1 с маской из MAMP1[3:0] прибавляется к регистру DHR1 и он пишется в DAC_DOR1. LFSR1 обновляется.

Через три такта APB1 после сигнала запуска DAC channel2 счётчик LFSR2 с маской из MAMP2[3:0] прибавляется к регистру DHR2 и он пишется в DAC_DOR2. LFSR2 обновляется.

12.4.4. Независимый запуск ЦАП с одинаковым треугольным сигналом

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать разные источники запуска в битах TSEL1[2:0] и TSEL2[2:0]
- Записать “1x” в биты WAVEx[1:0] обоих каналов и одну максимальную амплитуду в биты MAMPx[3:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 после сигнала запуска DAC channel1 счётчик треугольника с одинаковой амплитудой прибавляется к регистру DHR1 и он пишется в DAC_DOR1. Счётчик обновляется.

Через три такта APB1 после сигнала запуска DAC channel2 счётчик треугольника с одинаковой амплитудой прибавляется к регистру DHR2 и он пишется в DAC_DOR2. Счётчик обновляется.

12.4.5. Независимый запуск ЦАП с разным треугольным сигналом

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать разные источники запуска в битах TSEL1[2:0] и TSEL2[2:0]
- Записать “1x” в биты WAVEx[1:0] обоих каналов и разную максимальную амплитуду в биты MAMP1[3:0] и MAMP2[3:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 после сигнала запуска DAC channel1 счётчик треугольника с амплитудой из MAMP1[3:0] прибавляется к регистру DHR1 и он пишется в DAC_DOR1. Счётчик обновляется.

Через три такта APB1 после сигнала запуска DAC channel2 счётчик треугольника с амплитудой из MAMP2[3:0] прибавляется к регистру DHR2 и он пишется в DAC_DOR2. Счётчик обновляется.

12.4.6. Одновременный программный запуск ЦАП

Последовательность запуска:

- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через один такт APB1 регистры DHR1 и DHR2 пишутся в регистры DAC_DOR1 и DAC_DOR2.

12.4.7. Одновременный запуск ЦАП без генераторов

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать один источник запуска обоих каналов в битах TSEL1[2:0] и TSEL2[2:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 регистры DHR1 и DHR2 пишутся в регистры DAC_DOR1 и DAC_DOR2.

12.4.8. Одновременный запуск ЦАП с одинаковым LFSR

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать один источник запуска обоих каналов в битах TSEL1[2:0] и TSEL2[2:0]
- Записать “01” в биты WAVEx[1:0] обоих каналов и одинаковую маску LFSR в биты MAMPx[3:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 счётчик LFSR1 прибавляется к регистру DHR1 и он пишется в DAC_DOR1. LFSR1 обновляется. Одновременно счётчик LFSR2 прибавляется к регистру DHR2 и он пишется в DAC_DOR2. LFSR2 обновляется.

12.4.9. Одновременный запуск ЦАП с разными LFSR

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать один источник запуска обоих каналов в битах TSEL1[2:0] и TSEL2[2:0]
- Записать “01” в биты WAVEx[1:0] обоих каналов и разные маски LFSR в битах MAMP1[3:0] и MAMP2[3:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 после сигнала запуска счётчик LFSR1 с маской из MAMP1[3:0] прибавляется к регистру DHR1 и он пишется в DAC_DOR1. LFSR1 обновляется.

Одновременно счётчик LFSR2 с маской из MAMP2[3:0] прибавляется к регистру DHR2 и он пишется в DAC_DOR2. LFSR2 обновляется.

12.4.10. Одновременный запуск ЦАП с одинаковым треугольным сигналом

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать один источник запуска в битах TSEL1[2:0] и TSEL2[2:0]
- Записать “1x” в биты WAVEx[1:0] обоих каналов и одну максимальную амплитуду в биты MAMPx[3:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 после сигнала запуска DAC channel1 счётчик треугольника с одинаковой амплитудой прибавляется к регистру DHR1 и он пишется в DAC_DOR1. Счётчик обновляется.

Одновременно в DAC channel2 счётчик треугольника с одинаковой амплитудой прибавляется к регистру DHR2 и он пишется в DAC_DOR2. Счётчик обновляется.

12.4.11. Одновременный запуск ЦАП с разным треугольным сигналом

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать один источник запуска в битах TSEL1[2:0] и TSEL2[2:0]
- Записать “1x” в биты WAVEx[1:0] обоих каналов и разную максимальную амплитуду в биты MAMP1[3:0] и MAMP2[3:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 после сигнала запуска DAC channel1 счётчик треугольника с амплитудой из MAMP1[3:0] прибавляется к регистру DHR1 и он пишется в DAC_DOR1. Счётчик обновляется.

Одновременно в DAC channel2 счётчик треугольника с амплитудой из MAMP2[3:0] прибавляется к регистру DHR2 и он пишется в DAC_DOR2. Счётчик обновляется.

12.5. Регистры ЦАП

Доступны только словами.

12.5.1. Регистр управления АЦП (ADC_CR)

Смещение адреса: 0x00

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			DMA EN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2	EN2
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DMA EN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:29 Резерв, не трогать.

— Бит 28 **DMAEN2**: Разрешение DMA DAC channel2

Ставится и снимается программно.

0: Нельзя

1: Можно

— Биты 27:24 **MAMP2[3:0]**: Маска/амплитуда DAC channel2

Пишутся программно.

0000: Немаскированный бит 0 LFSR/ Амплитуда равны 1

0001: Немаскированные биты [1:0] LFSR/ Амплитуда равны 3

0010: Немаскированные биты[2:0] LFSR/ Амплитуда равны 7

0011: Немаскированные биты[3:0] LFSR/ Амплитуда равны 15

0100: Немаскированные биты[4:0] LFSR/ Амплитуда равны 31

0101: Немаскированные биты[5:0] LFSR/ Амплитуда равны 63

0110: Немаскированные биты[6:0] LFSR/ Амплитуда равны 127

0111: Немаскированные биты[7:0] LFSR/ Амплитуда равны 255

1000: Немаскированные биты[8:0] LFSR/ Амплитуда равны 511

1001: Немаскированные биты[9:0] LFSR/ Амплитуда равны 1023

1010: Немаскированные биты[10:0] LFSR/ Амплитуда равны to 2047

≥ 1011: Немаскированные биты[11:0] LFSR/ Амплитуда равны 4095

— Биты 23:22 **WAVE2[1:0]**: Включение генератора шума/треугольника DAC channel2

Пишутся программно.

00: Нету

01: Включён генератор шума

1x: Включён генератор треугольника

NB: Используется при TEN2 = 1 (запуск DAC channel2 разрешён)

— Биты 21:19 **TSEL2[2:0]**: Выбор запуска DAC channel2

Пишутся программно.

000: Timer 6 TRGO

001: Timer 3 TRGO в сетевых, Timer 8 TRGO в устройствах высокой и XL-плотности

010: Timer 7 TRGO

011: Timer 5 TRGO

100: Timer 2 TRGO

101: Timer 4 TRGO

110: External line9

111: Программный запуск

NB: Используется при TEN2 = 1 (запуск DAC channel2 разрешён)

— Бит 18 **TEN2**: Разрешение запуска DAC channel2

Пишется программно.

0: DAC channel2 запускается через один такт APB1 после записи в регистр DAC_DHRx передачей его в регистр DAC_DOR2.

1: DAC channel2 запускается через три такта APB1 после внешнего сигнала передачей DAC_DHRx в регистр DAC_DOR2.

NB: При программном запуске задержка передачи DAC_DHRx в DAC_DOR2 составляют 1 такт APB1.

— Бит 17 **BOFF2**: Выключение выходного буфера DAC channel2

Пишется программно.

0: Выходной буфер DAC channel2 включён

1: Выходной буфер DAC channel2 выключен

— Бит 16 **EN2**: Включение DAC channel2

Пишется программно.

0: DAC channel2 выключен

1: DAC channel2 включён

— Биты 15:13 Резерв, не трогать.

— Бит 12 **DMAEN1**: Разрешение DMA DAC channel1

Ставится и снимается программно.

0: Нельзя

1: Можно

— Биты 11:8 **MAMP1[3:0]**: Маска/амплитуда DAC channel1

Пишутся программно.

0000: Немаскированный бит 0 LFSR/ Амплитуда равны 1

0001: Немаскированные биты [1:0] LFSR/ Амплитуда равны 3

0010: Немаскированные биты[2:0] LFSR/ Амплитуда равны 7

0011: Немаскированные биты[3:0] LFSR/ Амплитуда равны 15

0100: Немаскированные биты[4:0] LFSR/ Амплитуда равны 31

0101: Немаскированные биты[5:0] LFSR/ Амплитуда равны 63

0110: Немаскированные биты[6:0] LFSR/ Амплитуда равны 127

0111: Немаскированные биты[7:0] LFSR/ Амплитуда равны 255

1000: Немаскированные биты[8:0] LFSR/ Амплитуда равны 511

1001: Немаскированные биты[9:0] LFSR/ Амплитуда равны 1023

1010: Немаскированные биты[10:0] LFSR/ Амплитуда равны to 2047

≥ 1011: Немаскированные биты[11:0] LFSR/ Амплитуда равны 4095

— Биты 7:6 **WAVE1[1:0]**: Включение генератора шума/треугольника DAC channel1

Пишутся программно.

00: Нету

01: Включён генератор шума

1x: Включён генератор треугольника

NB: Используется при TEN1 = 1 (запуск DAC channel1 разрешён)

— Биты 5:3 **TSEL1[2:0]**: Выбор запуска DAC channel1

Пишутся программно.

000: Timer 6 TRGO

001: Timer 3 TRGO в сетевых, Timer 8 TRGO в устройствах высокой и XL-плотности

010: Timer 7 TRGO

011: Timer 5 TRGO

100: Timer 2 TRGO

101: Timer 4 TRGO

110: External line9

111: Программный запуск

NB: Используется при TEN1 = 1 (запуск DAC channel1 разрешён)

— Бит 2 **TEN1**: Разрешение запуска DAC channel1

Пишется программно.

0: DAC channel1 запускается через один такт APB1 после записи в регистр DAC_DHRx передачей его в регистр DAC_DOR1.

1: DAC channel1 запускается через три такта APB1 после внешнего сигнала передачей DAC_DHRx в регистр DAC_DOR1.

NB: При программном запуске задержка передачи DAC_DHRx в DAC_DOR1 составляют 1 такт APB1.

— Бит 1 **BOFF1**: Выключение выходного буфера DAC channel1

Пишется программно.

0: Выходной буфер DAC channel1 включён

1: Выходной буфер DAC channel1 выключен

— Бит 0 **EN1**: Включение DAC channel1

Пишется программно.

0: DAC channel1 выключен

1: DAC channel1 включён

12.5.2. Регистр программного запуска АЦП (ADC_SWTRIGR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														SWTRIG2	SWTRIG1
														w	w

— Биты 31:2 Резерв, не трогать.

— Бит 1 **SWTRIG2**: Разрешение программного запуска DAC channel2

Только запись.

0: Нельзя

1: Можно

NB: Сбрасывается аппаратно через один такт после записи DAC_DHR2 в DAC_DOR2.

— Бит 0 **SWTRIG1**: Разрешение программного запуска DAC channel1

Только запись.

0: Нельзя

1: Можно

NB: Сбрасывается аппаратно через один такт после записи DAC_DHR1 в DAC_DOR1.

12.5.3. Регистр хранения правых 12-бит данных канала 1 (DAC_DHR12R1)

Смещение адреса: 0x08

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:12 Резерв, не трогать.

— Биты 11:0 **DACC1DHR[11:0]**: Выравненные вправо 12-бит данные DAC channel1.

12.5.4. Регистр хранения левых 12-бит данных канала 1 (DAC_DHR12L1)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

— Биты 31:16 Резерв, не трогать.
— Биты 15:4 **DACC1DHR[11:0]**: Выравненные влево 12-бит данные DAC channel1.
— Биты 3:0 Резерв, не трогать.

12.5.5. Регистр хранения правых 8-бит данных канала 1 (DAC_DHR8R1)

Смещение адреса: 0x10

По сбросу: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:8 Резерв, не трогать.
— Биты 7:0 **DACC1DHR[7:0]**: Выравненные вправо 8-бит данные DAC channel1.

12.5.6. Регистр хранения правых 12-бит данных канала 2 (DAC_DHR12R2)

Смещение адреса: 0x14

По сбросу: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:12 Резерв, не трогать.
— Биты 11:0 **DACC2DHR[11:0]**: Выравненные вправо 12-бит данные DAC channel2.

12.5.7. Регистр хранения левых 12-бит данных канала 2 (DAC_DHR12L2)

Смещение адреса: 0x18

По сбросу: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

— Биты 31:16 Резерв, не трогать.
— Биты 15:4 **DACC2DHR[11:0]**: Выравненные влево 12-бит данные DAC channel2.
— Биты 3:0 Резерв, не трогать.

12.5.8. Регистр хранения правых 8-бит данных канала 2 (DAC_DHR8R2)

Смещение адреса: 0x1C

По сбросу: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DACC2DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:8 Резерв, не трогать.
— Биты 7:0 **DACC2DHR[7:0]**: Выравненные вправо 8-бит данные DAC channel2.

12.5.9. Парный регистр правых 12-бит данных (DAC_DHR12RD)

Смещение адреса: 0x20

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:28 Резерв, не трогать.
— Биты 27:16 **DACC2DHR[11:0]**: Выравненные вправо 12-бит данные DAC channel2.
— Биты 15:12 Резерв, не трогать.
— Биты 11:0 **DACC1DHR[11:0]**: Выравненные вправо 12-бит данные DAC channel1.

12.5.10.Парный регистр левых 12-бит данных (DAC_DHL12RD)

Смещение адреса: 0x24

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

— Биты 31:20 **DACC2DHR[11:0]**: Выравненные вправо 12-бит данные DAC channel2.
— Биты 19:16 Резерв, не трогать.
— Биты 15:4 **DACC1DHR[11:0]**: Выравненные вправо 12-бит данные DAC channel1.
— Биты 3:0 Резерв, не трогать.

12.5.11.Парный регистр правых 8-бит данных (DAC_DHR8RD)

Смещение адреса: 0x28

По сбросу: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.
— Биты 15:8 **DACC2DHR[7:0]**: Выравненные вправо 8-бит данные DAC channel2.
— Биты 7:0 **DACC2DHR[7:0]**: Выравненные вправо 8-бит данные DAC channel1.

12.5.12.Выходной регистр данных канала 1 (DAC_DOR1)

Смещение адреса: 0x2C

По сбросу: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

— <u>Биты 31:12</u> Резерв, не трогать.
— <u>Биты 11:0</u> **DACC1DOR[11:0]**: Выравненные вправо 12-бит данные DAC channel1.

12.5.13.Выходной регистр данных канала 2 (DAC_DOR2)

Смещение адреса: 0x30

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

— <u>Биты 31:12</u> Резерв, не трогать.
— <u>Биты 11:0</u> **DACC2DOR[11:0]**: Выравненные вправо 12-бит данные DAC channel2.

12.5.14.Карта регистров ЦАП

0x00	**DAC_CR**	Res.			DMAEN2	MAMP2[3:0]				WAVE2[2:0]		TSEL2[2:0]			TEN2	BOFF2	EN2	Res.			DMAEN1	MAMP1[3:0]				WAVE1[2:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**DAC_SWTRIGR**	Reserved																														SWTRIG2	SWTRIG1
	Reset value																															0	0
0x08	**DAC_DHR12R1**	Reserved																				DACC1DHR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**DAC_DHR12L1**	Reserved																DACC1DHR[11:0]												Reserved			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0				
0x10	**DAC_DHR8R1**	Reserved																								DACC1DHR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	**DAC_DHR12R2**	Reserved																				DACC2DHR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x18	**DAC_DHR12L2**	Reserved																DACC2DHR[11:0]												Reserved			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0				
0x1C	**DAC_DHR8R2**	Reserved																								DACC2DHR[7:0]							
																										0	0	0	0	0	0	0	0
0x20	**DAC_DHR12RD**	Reserved				DACC2DHR[11:0]												Reserved				DACC1DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x24	**DAC_DHR12LD**	DACC2DHR[11:0]												Reserved				DACC1DHR[11:0]												Reserved			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x28	**DAC_DHR8RD**	Reserved																DACC2DHR[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**DAC_DOR1**	Reserved																				DACC1DOR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x30	**DAC_DOR2**	Reserved																				DACC2DOR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0

13. Контроллер ПДП (DMA)

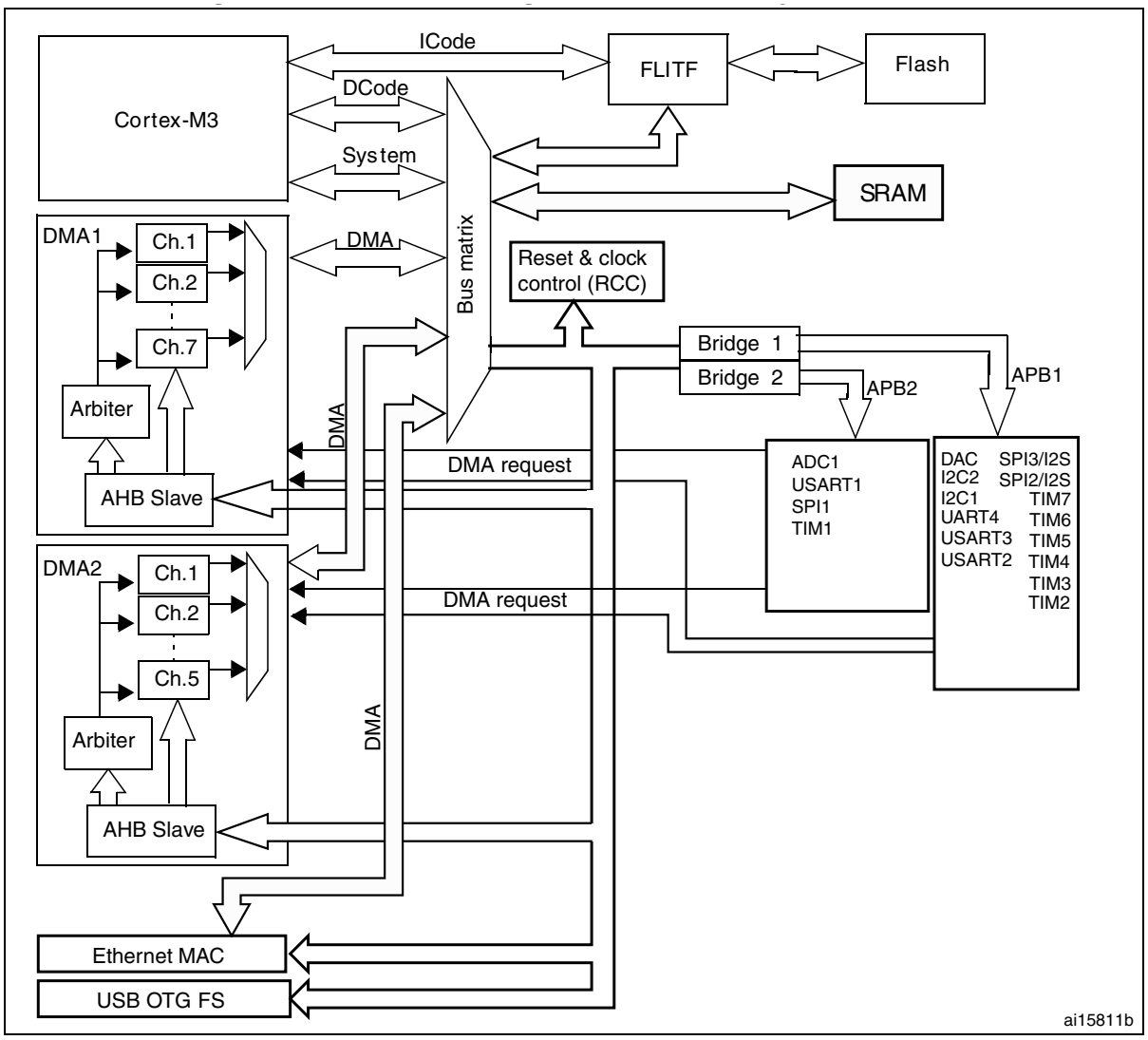
13.1. Введение в DMA

Прямой доступ к памяти работает как при передачах периферия-память, так и память-память. Два контроллера DMA имеют в общем 12 каналов (7 в DMA1 и 5 в DMA2). Приоритет запросов DMA управляется арбитром.

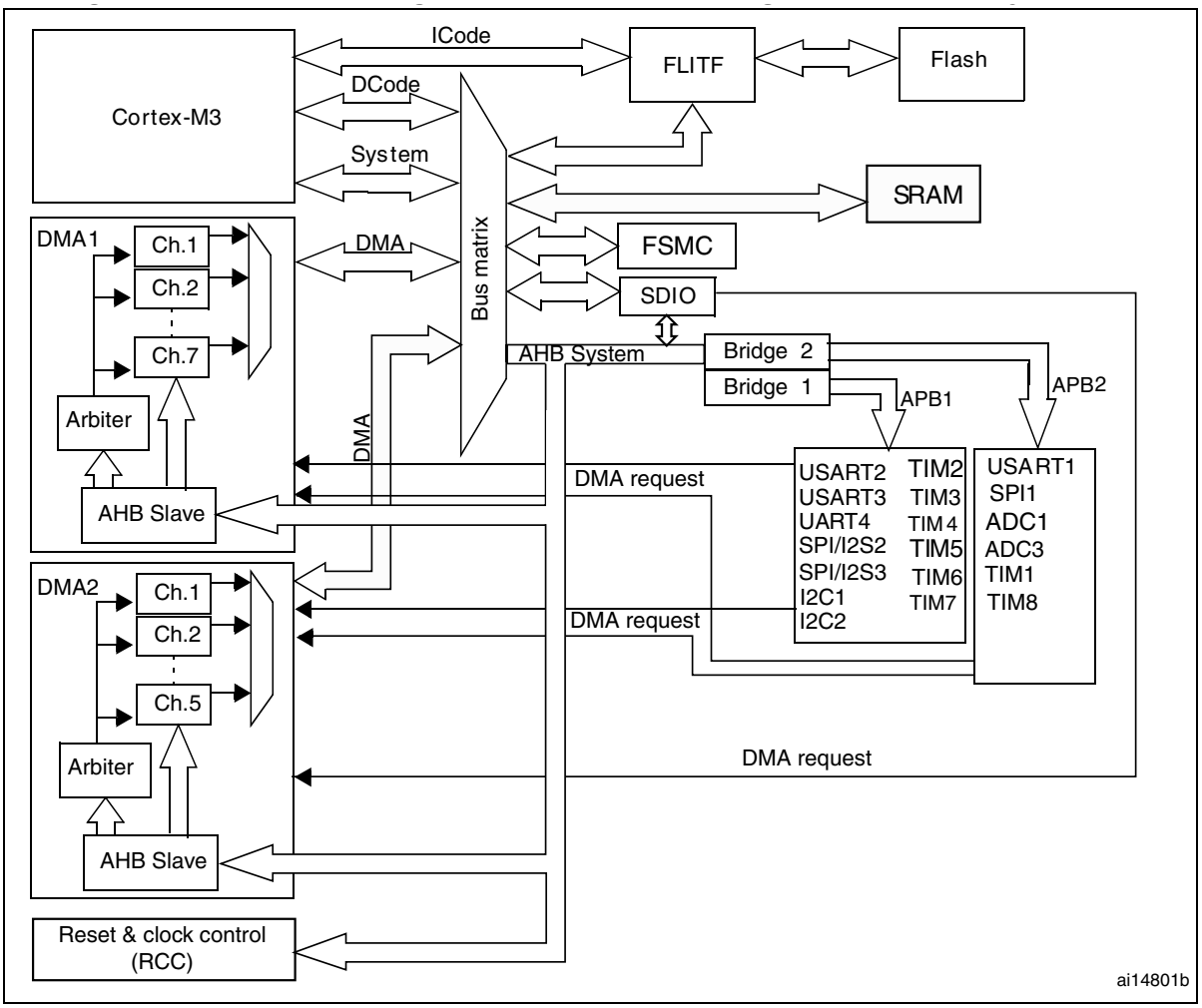
13.2. Основные свойства DMA

- 12 независимых каналов (запросов): 7 в DMA1 и 5 в DMA2
- Каждый канал программно подключается к выделенным запросам DMA, включая программный.
- Приоритеты запросов к одному DMA программируемые (4 уровня: *очень высокий*, *высокий*, *средний*, *низкий*) или аппаратные при равенстве (запрос 1 приоритетнее, чем 2, и т. д.)
- Независимый размер данных источника и получателя (байт, полуслово, слово), эмуляция упаковки и распаковки. Адреса источника/получателя должен быть выравнен на границу размера данных.
- Поддержка кольцевых буферов
- 3 флага событий (Полупередача DMA, Конец передачи DMA и Ошибка передачи DMA) логически складываются (OR) в один запрос прерывания каждого канала.
- Передачи память-память
- Передачи периферия-память, память-периферия и периферия-периферия
- Доступ к Flash, SRAM, периферии APB1, APB2 и AHB как источнику и получателю
- Программируемое число передаваемых данных: до 65536

Блок-схема ПДП в сетевых устройствах.



Блок-схема ПДП в устройствах разной плотности.



1. Контроллер DMA2 есть только в устройствах высокой и XL-плотности.
2. Запросы DMA для ADC3, SPI/I2S3, UART4, SDIO, TIM5, TIM6, DAC, TIM7, TIM8 есть только в устройствах высокой плотности.

13.3. Функциональное свойства DMA

Контролер DMA разделяет системную шину с ядром CPU. Запрос DMA может остановить доступ CPU к системной шине на несколько циклов при обращении к одному получателю (памяти или периферии). Шинная матрица имеет кольцевой арбитраж, оставляя как минимум половину пропускной способности системной шины (и памяти и периферии) для CPU.

13.3.1. Транзакции DMA

После события устройство шлёт запрос контроллеру DMA, который обслуживает запросы согласно приоритетам. Когда контроллер обращается к устройству он шлёт ему подтверждение. Получив подтверждение, устройство снимает запрос. В ответ на снятие запроса контроллер снимает подтверждение. Если запросов больше нет, то устройство может выставлять следующий запрос.

Передача DMA состоит из трёх операций:

- Чтение данных из устройства или памяти по внутреннему регистру текущего адреса. Начальный адрес передачи программно пишется в регистры `DMA_CPARx` или `DMA_CMARx`
- Запись данных в устройство или память по внутреннему регистру текущего адреса. Начальный адрес передачи программно пишется в регистры `DMA_CPARx` или `DMA_CMARx`
- Пост-декремент регистра `DMA_CNDTRx`, с числом оставшихся транзакций.

13.3.2. Арбитр

Арбитр рассматривает запросы согласно их приоритетов и запускает последовательности доступа.

Арбитраж имеет две стадии:

- Программная: В регистрах DMA_CCRx каждому каналу назначается один из четырёх приоритетов:
  - Очень высокий
  - Высокий
  - Средний
  - Низкий
- Аппаратная: При равенстве программных приоритетов двух каналов первым обслуживается канал с меньшим номером.

NB: В устройствах с двумя контроллерами DMA1 приоритетнее DMA2.

13.3.3. Каналы DMA

Все каналы DMA могут передавать заданное количество данных (до 65535) между регистрами устройств, лежащими по фиксированным адресам, и памятью согласно внутреннему декрементному счётчику.

Размеры данных

Задаются битами PSIZE и MSIZE в регистре DMA_CCRx.

Инкремент указателя

Указатели устройств и памяти могут автоматически увеличиваться после каждой передачи согласно битам PINC и MINC в регистре DMA_CCRx. Разрешённый инкремент наращивает указатель данных на 1, 2 или 4 согласно размеру данных. Адрес первой передачи пишется в регистры DMA_CPARx/DMA_CMARx. Во время передачи они не изменяются. Адреса текущей передачи программе недоступны.

В некольцевом режиме запросы DMA, поступившие после обнуления счётчика длины, не обслуживаются. Для его обновления в регистре DMA_CNDTRx канал DMA нужно отключать.

NB: У отключённого канала DMA регистры DMA_CCRx, DMA_CPARx и DMA_CMARx не сбрасываются.

В кольцевом режиме, при обнулении счётчика передач он, как а текущие адреса, автоматически перегружаются начальными значениями из регистров DMA_CNDTRx и DMA_CPARx/DMA_CMARx.

Конфигурация канала.

Надо:

1. Записать адрес регистра устройства в регистр DMA_CPARx.
2. Записать адрес памяти в регистр DMA_CMARx.
3. Записать число передаваемых данных в регистр DMA_CNDTRx.
4. Установить приоритет канала в битах PL[1:0] регистра DMA_CCRx.
5. Выбрать направление передачи, кольцевой режим, режим инкремента памяти/устройства, размер данных памяти/устройства и прерывания после передачи половины и всех данных в регистре DMA_CCRx.
6. Включить канал битом ENABLE в регистр DMA_CCRx.

Вот теперь канал может обслуживать запросы DMA подключённого устройства.

После передачи половины байтов запроса ставится флаг (HTIF) и генерируется прерывание (если разрешено битом HTIE). После завершения всей передачи ставится флаг TCIF и генерируется прерывание (если разрешено битом TCIE).

Кольцевой режим.

Он позволяет работать с кольцевыми буферами и непрерывными потоками данных (вроде сканирования ADC). Включается битом CIRC в регистре DMA_CCRx. При включённом кольцевом режиме после обнуления счётчика передач число передаваемых данных и начальные адреса автоматически перегружаются и запросы DMA продолжают обслуживаться.

Режим память-память.

При стоящем бите MEM2MEM в регистре DMA_CCRx передача запускается установкой бита EN в регистре DMA_CCRx. Останавливается она при обнулении регистра DMA_CNDTRx. Кольцевой режим и Память-Память несовместимы.

13.3.4. Ширина данных, выравнивание и порядок байт

Если PSIZE и MSIZE не равны, то DMA выполняет некоторое выравнивание.

Таблица 76. Ширина данных и порядок байт (PINC = MINC = 1).

Ширина		Адрес/данные источника	Операции	Адрес/данные получателя
Ист.	Получ			
8	8	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: READ B0[7:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B1[7:0] @0x1 then WRITE B1[7:0] @0x1 3: READ B2[7:0] @0x2 then WRITE B2[7:0] @0x2 4: READ B3[7:0] @0x3 then WRITE B3[7:0] @0x3	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3
8	16	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: READ B0[7:0] @0x0 then WRITE 00B0[15:0] @0x0 2: READ B1[7:0] @0x1 then WRITE 00B1[15:0] @0x2 3: READ B3[7:0] @0x2 then WRITE 00B2[15:0] @0x4 4: READ B4[7:0] @0x3 then WRITE 00B3[15:0] @0x6	@0x0 / 00B0 @0x2 / 00B1 @0x4 / 00B2 @0x6 / 00B3
8	32	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: READ B0[7:0] @0x0 then WRITE 000000B0[31:0] @0x0 2: READ B1[7:0] @0x1 then WRITE 000000B1[31:0] @0x4 3: READ B3[7:0] @0x2 then WRITE 000000B2[31:0] @0x8 4: READ B4[7:0] @0x3 then WRITE 000000B3[31:0] @0xC	@0x0 / 000000B0 @0x4 / 000000B1 @0x8 / 000000B2 @0xC / 000000B3
16	8	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: READ B1B0[15:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE B2[7:0] @0x1 3: READ B5B4[15:0] @0x4 then WRITE B4[7:0] @0x2 4: READ B7B6[15:0] @0x6 then WRITE B6[7:0] @0x3	@0x0 / B0 @0x1 / B2 @0x2 / B4 @0x3 / B6
16	16	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: READ B1B0[15:0] @0x0 then WRITE B1B0[15:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE B3B2[15:0] @0x2 3: READ B5B4[15:0] @0x4 then WRITE B5B4[15:0] @0x4 4: READ B7B6[15:0] @0x6 then WRITE B7B6[15:0] @0x6	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6
16	32	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: READ B1B0[15:0] @0x0 then WRITE 0000B1B0[31:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE 0000B3B2[31:0] @0x4 3: READ B5B4[15:0] @0x4 then WRITE 0000B5B4[31:0] @0x8 4: READ B7B6[15:0] @0x6 then WRITE 0000B7B6[31:0] @0xC	@0x0 / 0000B1B0 @0x4 / 0000B3B2 @0x8 / 0000B5B4 @0xC / 0000B7B6
32	8	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B4[7:0] @0x1 3: READ BBBAB9B8[31:0] @0x8 then WRITE B8[7:0] @0x2 4: READ BFBEBDBC[31:0] @0xC then WRITE BC[7:0] @0x3	@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC
32	16	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B1B0[7:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B5B4[7:0] @0x1 3: READ BBBAB9B8[31:0] @0x8 then WRITE B9B8[7:0] @0x2 4: READ BFBEBDBC[31:0] @0xC then WRITE BDBC[7:0] @0x3	@0x0 / B1B0 @0x2 / B5B4 @0x4 / B9B8 @0x6 / BDBC
32	32	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B3B2B1B0[31:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B7B6B5B4[31:0] @0x4 3: READ BBBAB9B8[31:0] @0x8 then WRITE BBBAB9B8[31:0] @0x8 4: READ BFBEBDBC[31:0] @0xC then WRITE BFBEBDBC[31:0] @0xC	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC

Адресация устройств AHB, не поддерживающих запись байтов и полуслов.

При записи байтов или полуслов по шине AHB данных дублируются на незанятые линии шины HWDATA[31:0]. Так что, если ведомое устройство AHB не поддерживает такой записи (HSIZE не используется) и не выдаёт ошибки, то DMA пишет 32 бита HWDATA:

- При полуслове “0xABCD”, шина HWDATA получит “0xABCDABCD” с HSIZE = HalfWord
- При байте “0xAB” шина HWDATA получит “0xABABABAB” с HSIZE = Byte

Так как мост AHB/APB это ведомое 32-бит AHB устройство, что не внимает данным HSIZE, он преобразует байтовые и полусловные операции AHB в 32-операции APB следующим образом:

- Запись байта AHB “0xB0” в 0x0 (0x1, 0x2 или 0x3) преобразуется в запись слова APB “0xB0B0B0B0” в 0x0
- Запись полуслова AHB “0xB1B0” в 0x0 (или 0x2) преобразуется в запись слова APB “0xB1B0B1B0” в 0x0

Например, при записи в регистры резервного хранения APB (16-бит регистры, выравненные на границу 32 бит), размер источника (MSIZE) должен быть “16-bit”, а размер получателя (PSIZE) должен быть “32-bit”.

13.3.5. Обработка ошибок

Ошибка передачи DMA возникает при обращении к резервному адресному пространству. При этом аппаратно снимается бит EN в регистре DMA_CCRx сбойного канала (он выключается), ставится флаг TEIF в регистре DMA_IFR и если флаг TEIE в регистр DMA_CCRx стоит, то генерируется прерывание.

13.3.6. Прерывания

Прерывания генерируются при Половине передачи, Конце передачи и Ошибке передачи для каждого канала. Разрешаются они раздельно.

Таблица 77. Запросы прерываний DMA.

Событие прерывания	Флаг события	Бит разрешения
Половина передачи	HTIF	HTIE
Конец передачи	TCIF	TCIE
Ошибка передачи	TEIF	TEIE

NB: В устройствах высокой и XL-плотности DMA2 Channel4 и DMA2 Channel5 определены на один вектор, а в сетевых устройствах у них вектора разные. Прерывания прочих каналов DMA1 и DMA2 имеют собственные векторы.

13.3.7. Распределение запросов DMA

Контроллер DMA1

Запросы 7 устройств (TIMx[1,2,3,4], ADC1, SPI1, SPI/I2S2, I2Cx[1,2] и USARTx[1,2,3]) логически складываются (OR) перед подачей в DMA1, то есть единовременно должен стоять только один запрос.

Запросы DMA от устройств разрешаются независимо в регистрах управления самих устройств.

Запросы DMA1.

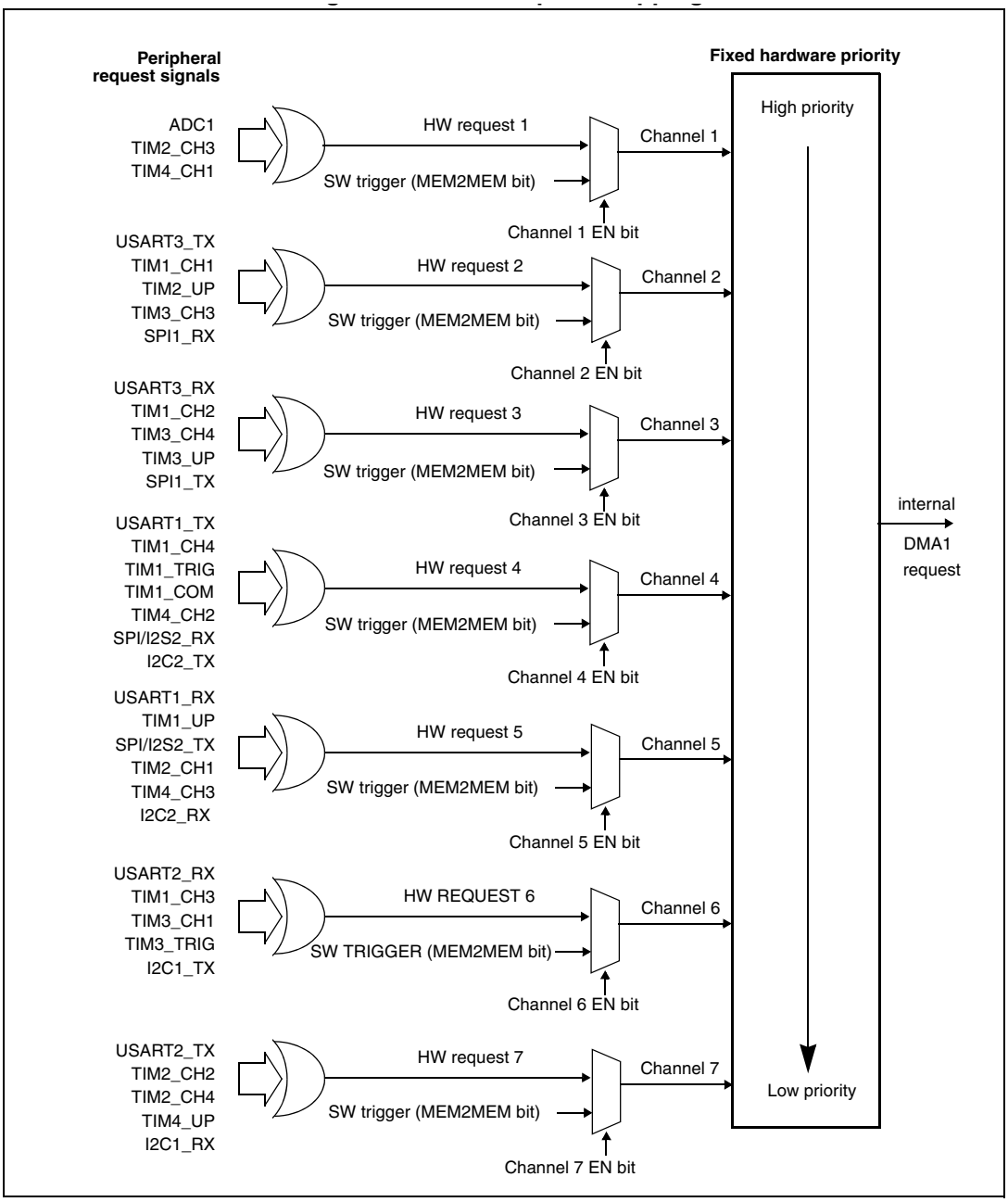


Таблица 78. Распределение устройств по каналам DMA1.

Устр-во	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
ADC1	ADC1	-	-	-	-	-	-
SPI/I^2S	-	SPI1_RX	SPI1_TX	SPI2/ I2S2_RX	SPI2/ I2S2_TX	-	-
USART	-	USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I^2C	-	-	-	I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1	-	TIM1_CH1	-	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	-
TIM2	TIM2_CH3	TIM2_UP	-	-	TIM2_CH1	-	TIM2_CH2 TIM2_CH4
TIM3	-	TIM3_CH3	TIM3_CH4 TIM3_UP	-	-	TIM3_CH1 TIM3_TRIG	-
TIM4	TIM4_CH1	-	-	TIM4_CH2	TIM4_CH3	-	TIM4_UP

Контроллер DMA2.

Запросы 5 устройств (TIMx[5,6,7,8], ADC3, SPI/I2S3, UART4, DAC_Channel[1,2] и SDIO) логически складываются (OR) перед подачей в DMA2, то есть единовременно должен стоять только один запрос.

Запросы DMA от устройств разрешаются независимо в регистрах управления самих устройств.

DMA2 есть только в устройствах высокой, XL-плотности и сетевых.

Запросы DMA2.

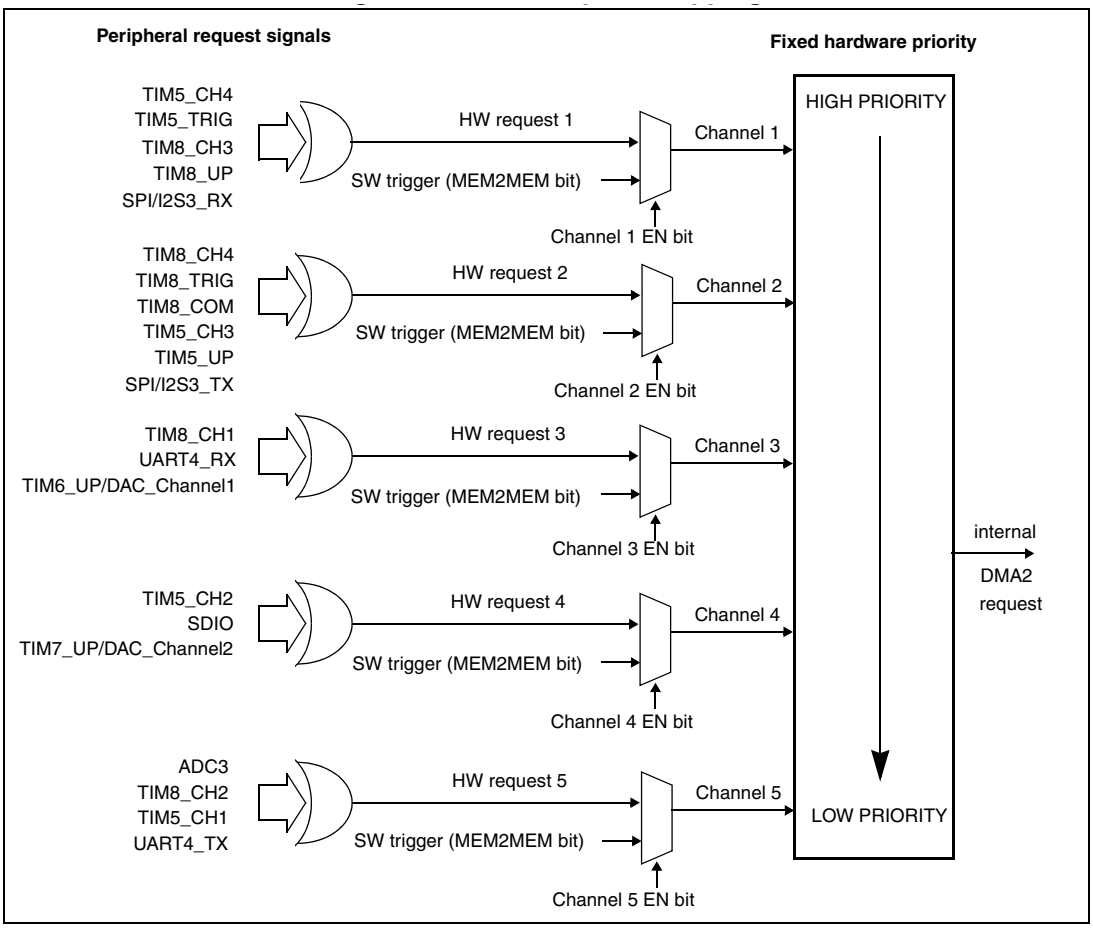


Таблица 78. Распределение устройств по каналам DMA2.

Устр-во	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
ADC3(1)	-	-	-	-	ADC3
SPI/I2S3	SPI/I2S3_RX	SPI/I2S3_TX	-	-	-
UART4	-	-	UART4_RX	-	UART4_TX
SDIO(1)	-	-	-	SDIO	-
TIM5	TIM5_CH4 TIM5_TRIG	TIM5_CH3 TIM5_UP	-	TIM5_CH2	TIM5_CH1
TIM6/ DAC_Channel1	-	-	TIM6_UP/ DAC_Channel1	-	-
TIM7	-	-	-	TIM7_UP/ DAC_Channel2	-
TIM8	TIM8_CH3 TIM8_UP	TIM8_CH4 TIM8_TRIG TIM8_COM	TIM8_CH1	-	TIM8_CH2

1. ADC3, SDIO и TIM8 DMA есть только в устройствах высокой и XL-плотности.

13.4. Регистры DMA

Биты регистров для channel6 и channel7 к DMA2 не относятся.

Регистры доступны байтами (8-бит), полусловами (16-бит) и словами (32-бит).

13.4.1. Регистр состояния прерываний (DMA_ISR)

Смещение адреса: 0x00

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
				r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:28 Резерв, не трогать.

— Биты 27,23,19,15, 11, 7, 3 **TEIFx**: Ошибка передачи канала x (x=1..7)
Ставится аппаратно. Снимается записью 1 с нужный бит регистра DMA_IFCR.
0: TE не было
1: TE было

— Биты 26,22,18,14, 10, 6, 2 **HTIFx:** Половина передачи канала x (x=1..7)
Ставится аппаратно. Снимается записью 1 с нужный бит регистра DMA_IFCR.
0: HT не было
1: HT было

— Биты 25,21,17,13, 9, 5, 1 **TCIFx:** Конец передачи канала x (x=1..7)
Ставится аппаратно. Снимается записью 1 с нужный бит регистра DMA_IFCR.
0: TC не было
1: TC было

— Биты 24,20,16,12, 8, 4, 0 **GIFx:** Глобальное прерывание канала x (x=1..7)
Ставится аппаратно. Снимается записью 1 с нужный бит регистра DMA_IFCR.
0: TE, HT или TC не было
1: TE, HT или TC было

13.4.2. Регистр очистки флагов прерываний (DMA_ISFR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CTEIF 7	CHTIF 7	CTCIF7	CGIF7	CTEIF6	*CHTIF6*	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
				w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTEIF 4	CHTIF 4	CTCIF 4	CGIF4	CTEIF 3	CHTIF 3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

— Биты 31:28 Резерв, не трогать.

— Биты 27,23,19,15, 11, 7, 3 **CTEIFx**: Очистка ошибки передачи канала x (x=1..7)
Ставится и снимается программно.
0: Ничего
1: Чистит флаг в регистре DMA_ISR

— Биты 26,22,18,14, 10, 6, 2 **CHTIFx:** Очистка половины передачи канала x (x=1..7)
Ставится и снимается программно.
0: Ничего
1: Чистит флаг в регистре DMA_ISR

— Биты 25,21,17,13, 9, 5, 1 **CTCIFx:** Очистка конца передачи канала x (x=1..7)
Ставится и снимается программно.
0: Ничего
1: Чистит флаг в регистре DMA_ISR

— Биты 24,20,16,12, 8, 4, 0 **CGIFx:** Очистка глобального прерывания канала x (x=1..7)
Ставится и снимается программно.
0: Ничего
1: Чистит флаги GIF, TEIF, HTIF и TCIF в регистре DMA_ISR

13.4.3. Регистр конфигурации канала x (DMA_CCRx) (x = 1..7)

Смещение адреса: 0x08 + 0d20 × (номер канала – 1)

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2 MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:28 Резерв, не трогать.

— Бит 14 канала x (x=1..7)
Ставится и снимается программно.
0: Ничего
1:

— Биты 13:12 **PL[1:0]:** Приоритет канала x (x=1..7)
Читается и пишется программно.
00: Низкий
01: Средний
10: Высокий
11: Очень высокий

— Биты 11:10 **MSIZE[1:0]:** Размер данных в памяти канала x (x=1..7)
Читается и пишется программно.
00: 8-бит
01: 16-бит
10: 32-бит
11: Резерв

— Биты 9:8 **PSIZE[1:0]**: Размер данных устройства канала x (x=1..7)
Читается и пишется программно.
00: 8-бит
01: 16-бит
10: 32-бит
11: Резерв

— Бит 7 **MINC:** Инкремент адреса памяти канала x (x=1..7)
Ставится и снимается программно.
0: Нельзя
1: Нужно

— Бит 6 **PINC:** Инкремент адреса устройства канала x (x=1..7)
Ставится и снимается программно.
0: Нельзя
1: Нужно

— Бит 5 **CIRC:** Кольцевой режим канала x (x=1..7)
Ставится и снимается программно.
0: Выключен
1:Включён

— Бит 4 **DIR:** Направление передачи канала x (x=1..7)
Ставится и снимается программно.
0: Из устройства
1: Из памяти

— Бит 3 **TEIE:** Разрешение прерывания Ошибки передачи канала x (x=1..7)
Ставится и снимается программно.
0: Нельзя
1: Можно

— Бит 2 **HTIE:** Разрешение прерывания Половины передачи канала x (x=1..7)
Ставится и снимается программно.
0: Нельзя
1: Можно

— Бит 1 **TCIE:** Разрешение прерывания Конца передачи канала x (x=1..7)
Ставится и снимается программно.
0: Нельзя
1: Можно

— Бит 0 **EN:** Включение канала x (x=1..7)
Ставится и снимается программно.
0: Выключен
1: Включён

13.4.4. Регистр числа данных канала x (DMA_CNDTRx) (x = 1..7)

Смещение адреса: 0x0C + 0d20 × (номер канала – 1)

По сбросу: 0x0000 0000

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

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **NDT[15:0]:** Число передаваемых данных канала x (x=1..7)
Число передаваемых данных (0 до 65535). Пишется только при выключенном канале. При разрешённом канале доступен только по чтению, выдаёт оставшееся число данных. Декрементируется после каждой передачи DMA. После завершения передачи может автоматически перегружаться ранее записанным числом в зависимости от установленного режима.
При обнулённом регистре запросы не обслуживаются независимо от того включён канал или нет.

13.4.5. Регистр адреса периферии канала x (DMA_CPARx) (x = 1..7)

Смещение адреса: 0x14 + 0d20 × (номер канала – 1)

По сбросу: 0x0000 0000

При включённом канале писать нельзя.

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

— Биты 31:0 **PA[31:0]:** Адрес периферии канала x (x=1..7)
Базовый адрес регистра данных устройства обмена.
Если PSIZE равен 01 (16-бит), бит PA[0] игнорируется. Доступ автоматически выравнивается на границу полуслова.
Если PSIZE равен 10 (32-bit), биты PA[1:0] игнорируются. Доступ автоматически выравнивается на границу слова.

13.4.6. Регистр адреса памяти канала x (DMA_CPARx) (x = 1..7)

Смещение адреса: 0x10 + 0d20 × (номер канала – 1)

По сбросу: 0x0000 0000

При включённом канале писать нельзя.

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

— Биты 31:0 **MA[31:0]:** Адрес памяти канала x (x=1..7)

Базовый адрес памяти для обмена.

Если MSIZE равен 01 (16-бит), бит MA[0] игнорируется. Доступ автоматически выравнивается на границу полуслова.

Если MSIZE равен 10 (32-bit), биты MA[1:0] игнорируются. Доступ автоматически выравнивается на границу слова.

13.4.7. Карта регистров DMA

0x000	**DMA_ISR**	Reserved				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	**DMA_IFCR**	Reserved				CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	**DMA_CCR1**	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	**DMA_CNDTR1**	Reserved																	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	**DMA_CPAR1**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	**DMA_CMAR1**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	Reserved																																
0x01C	**DMA_CCR2**	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**DMA_CNDTR2**	Reserved																	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	**DMA_CPAR2**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**DMA_CMAR2**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	Reserved																																
0x030	**DMA_CCR3**	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	**DMA_CNDTR3**	Reserved																	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	**DMA_CPAR3**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	**DMA_CMAR3**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	Reserved																																

0x044	**DMA_CCR4**	Reserved																		MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	**DMA_CNDTR4**	Reserved																NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	**DMA_CPAR4**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	**DMA_CMAR4**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	Reserved																																
0x058	**DMA_CCR5**	Reserved																		MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	**DMA_CNDTR5**	Reserved																NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	**DMA_CPAR5**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	**DMA_CMAR5**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	Reserved																																
0x06C	**DMA_CCR6**	Reserved																		MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	**DMA_CNDTR6**	Reserved																NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	**DMA_CPAR6**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078	**DMA_CMAR6**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	Reserved																																
0x080	**DMA_CCR7**	Reserved																		MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	**DMA_CNDTR7**	Reserved																NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	**DMA_CPAR7**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	**DMA_CMAR7**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	Reserved																																

14. Улучшенные таймеры (TIM1 и TIM8)

14.1. Введение в TIM1 и TIM8

Это 16-бит само-перегружаемые таймеры с программируемым предделителем.

Они могут использоваться для измерения длительности входных импульсов (захват входа), генерации сигналов (сравнение выхода, ШИМ, инверсный ШИМ) и пр.

Длина импульсов и сигналов может модулироваться от нескольких микросекунд до миллисекунд с помощью предделителей таймера и тактов RCC.

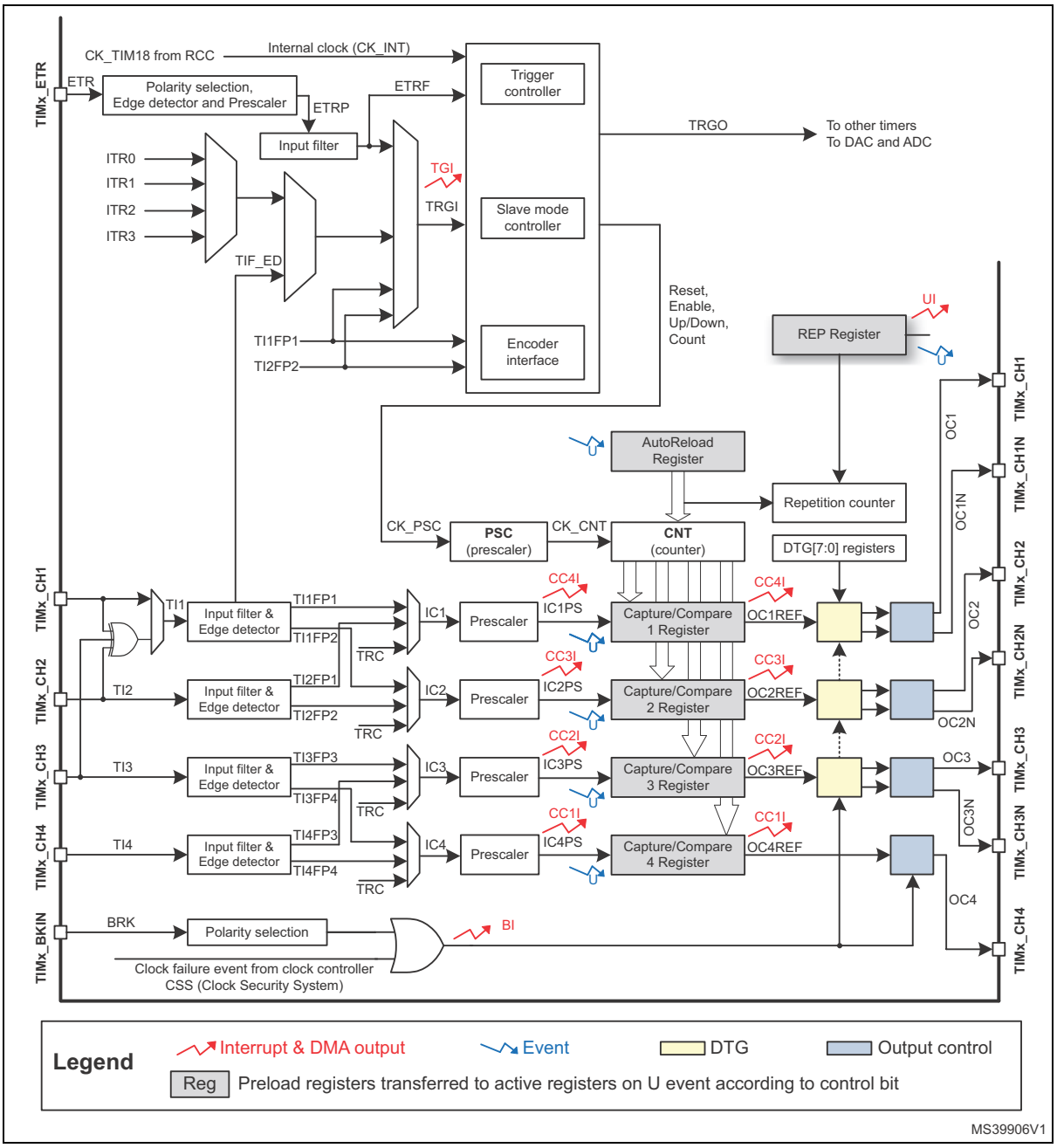
Улучшенные (TIM1 and TIM8) и обычные (TIMx) таймеры полностью независимы и не имеют общих ресурсов. Могут синхронизироваться между собой. См. *Секцию 14.3.20*.

14.2. Основные свойства TIM1 и TIM8

Это:

- 16-бит прямой, обратный, реверсивный счёт с само-перезагрузкой.
- 16-бит программируемый предделитель (можно “налету”) тактов на число от 1 до 65536.
- До 4 независимых каналов для:
  - Захвата входа
  - Сравнение выхода
  - ШИМ генерация (Фронт и Центр)
  - Одиночный импульс
- Инверсные выходы с программируемой задержкой
- Схема синхронизации внешнего управления и объединения таймеров.
- Счётчик повторения для обновления регистров таймера только после нескольких циклов счёта.
- Вход остановки для перевода выходных сигналов таймера в заданное состояние.
- Генерация Прерываний/DMA по следующим событиям:
  - Обновление: переполнение/исчерпание, инициализация счётчика (программно или запуском)
  - Запуск (старт, стоп, инициализация или счёт по сигналу)
  - Захват входа
  - Сравнение выхода
  - Вход остановки.
- Поддерживает инкрементный (квадратурный) кодер и датчики Холла для позиционирования.
- Вход запуска от внешних тактов или шагового управления током.

Блок-схема улучшенных таймеров.



14.3. Функциональное описание таймеров

14.3.1. Узел счёта

Это 16-бит счётчик с регистром само-перезагрузки. Имеет прямой, обратный и реверсивный режимы. Тактирование может поступать через предделитель.

Счётчик, регистр перезагрузки и предделитель доступны программно для чтения и записи даже во время счёта.

Включает:

- Регистр счётчика (TIMx_CNT)
- Регистр предделителя (TIMx_PSC)
- Регистр перезагрузки (TIMx_ARR)
- Регистр повторения счёта (TIMx_RCR)

Регистр перезагрузки предзагружаемый (есть видимый и скрытый регистры). Содержимое видимого регистра пишется в скрытый по каждому событию обновления (UEV), в зависимости от бита разрешения (ARPE) в регистре TIMx_CR1. Оно посылается программно или при переполнении/исчерпании счётчика при снятом бите UDIS в регистре TIMx_CR1.

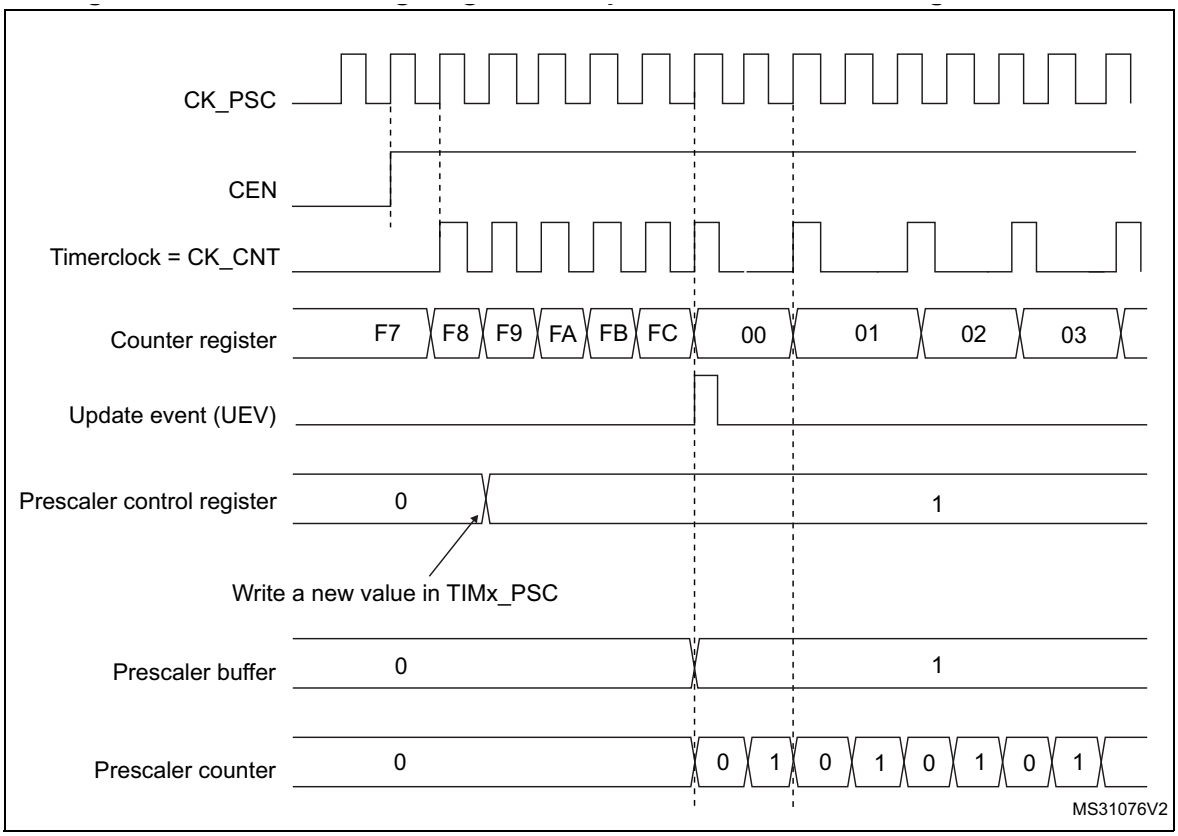
Счётчик тактируется выходом предделителя CK_CNT. Разрешается битом CEN в регистре TIMx_CR1.

Счёт начинается через 1 такт после установки бита CEN в регистре TIMx_CR1.

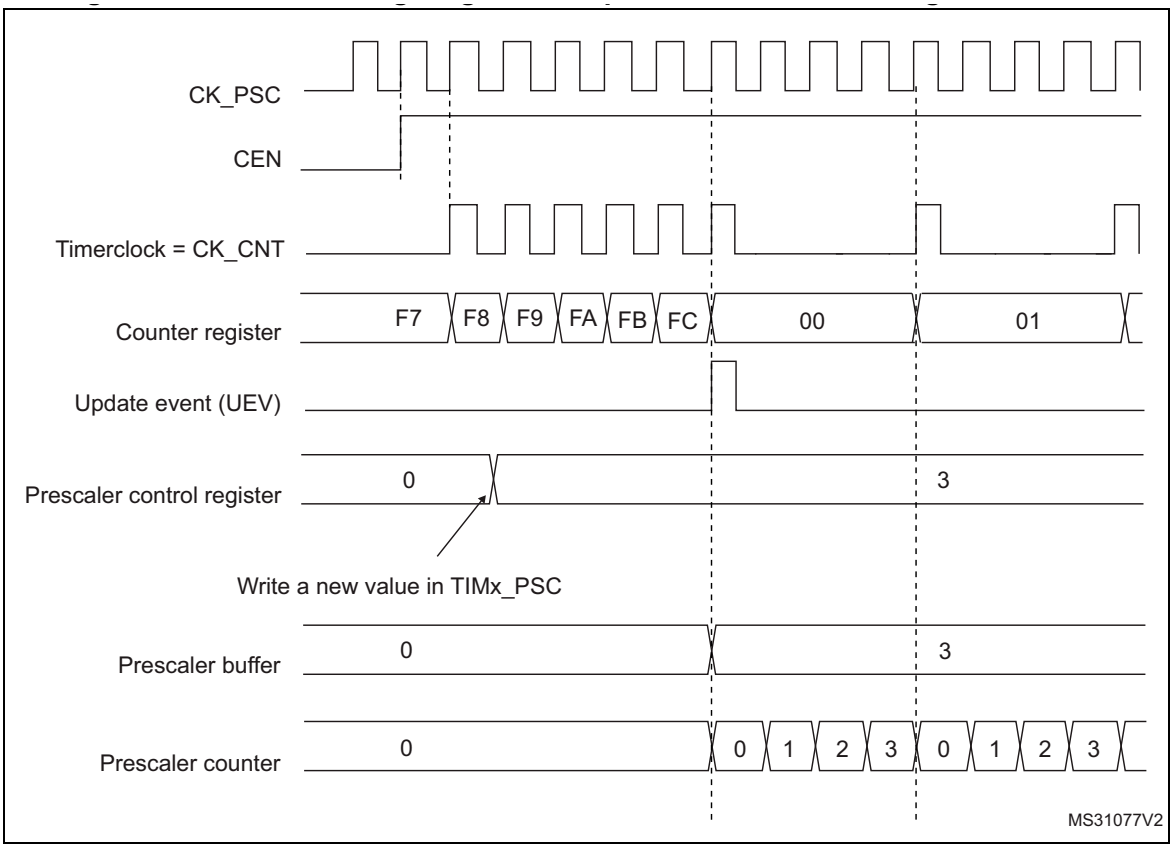
Описание предделителя

Это 16-бит счётчик делителя входной частоты (от 1 до 65536) с видимым регистром TIMx_PSC. его можно менять налету. Новое значение счётчика пишется по следующему событию обновления.

Временная диаграмма с изменением предделителя с 1 на 2.



Временная диаграмма с изменением предделителя с 1 на 4.



14.3.2. Режимы счёта

Прямой счёт

Счёт идёт от 0 до значения перезагрузки (регистр TIMx_ARR), сбрасывается в 0 и выдаёт событие переполнения счётчика.

Если используется счётчик повторений, то событие UEV выдаётся после обнуления заданных повторений плюс 1 (TIMx_RCR+1). Иначе событие обновления выдаётся при каждом переполнении счётчика.

Установка бита UG в регистре TIMx_EGR (программно или контроллером режима ведомого) также выдаёт событие обновления.

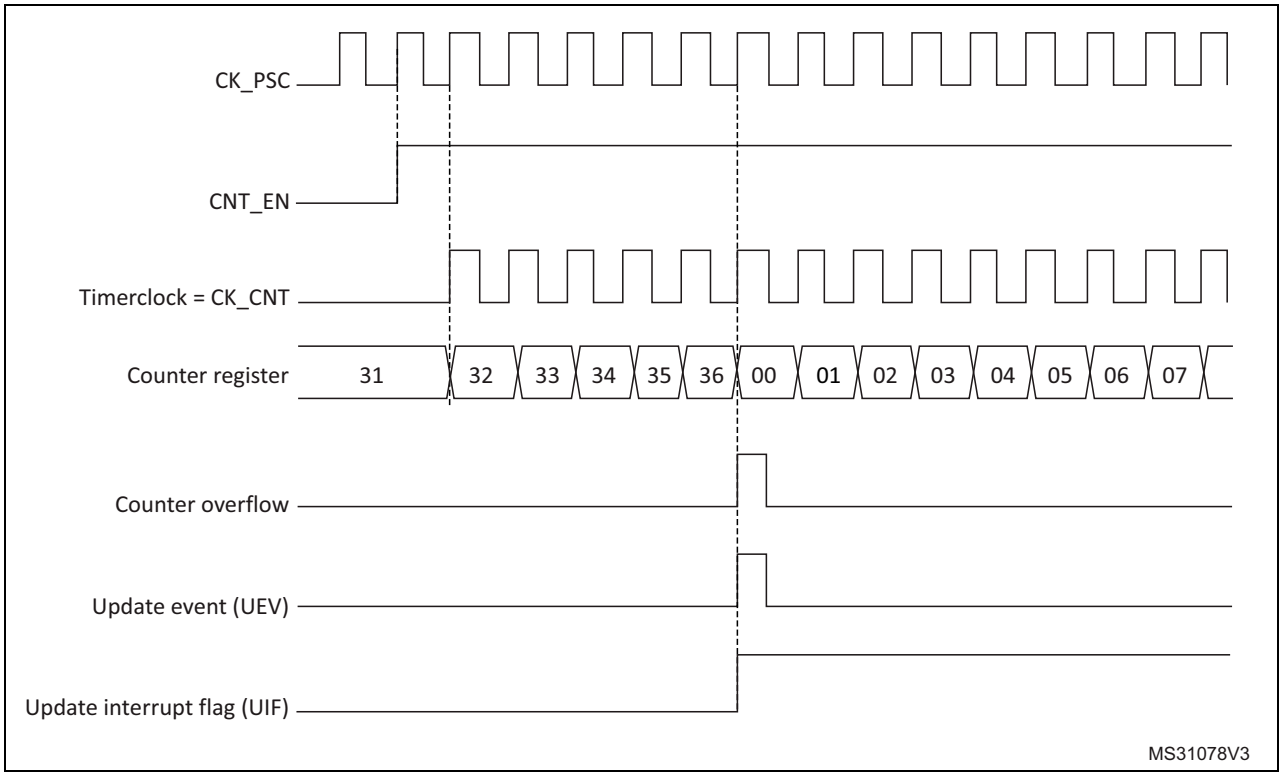
Событие UEV выключается установкой бита UDIS в регистре TIMx_CR1. Этим предупреждается запись скрытого регистра во время изменения регистра предзагрузки. Также событие UEV не появляется при сброшенном бите UDIS 0. Однако, счётчик продолжает считать с 0, равно как и предделитель (не изменив своего значения). Кроме того, если стоит бит URS в регистре TIMx_CR1, то установка бита UG выдаёт событие UEV без установки флага UIF (без прерывания и запроса DMA). Этим предупреждается одновременная выдача прерываний обновления и захвата при очистке счётчика по событию захвата.

По событию обновления обновляются все регистры и ставится флаг обновления (бит UIF в регистре TIMx_SR). Это зависит от бита URS:

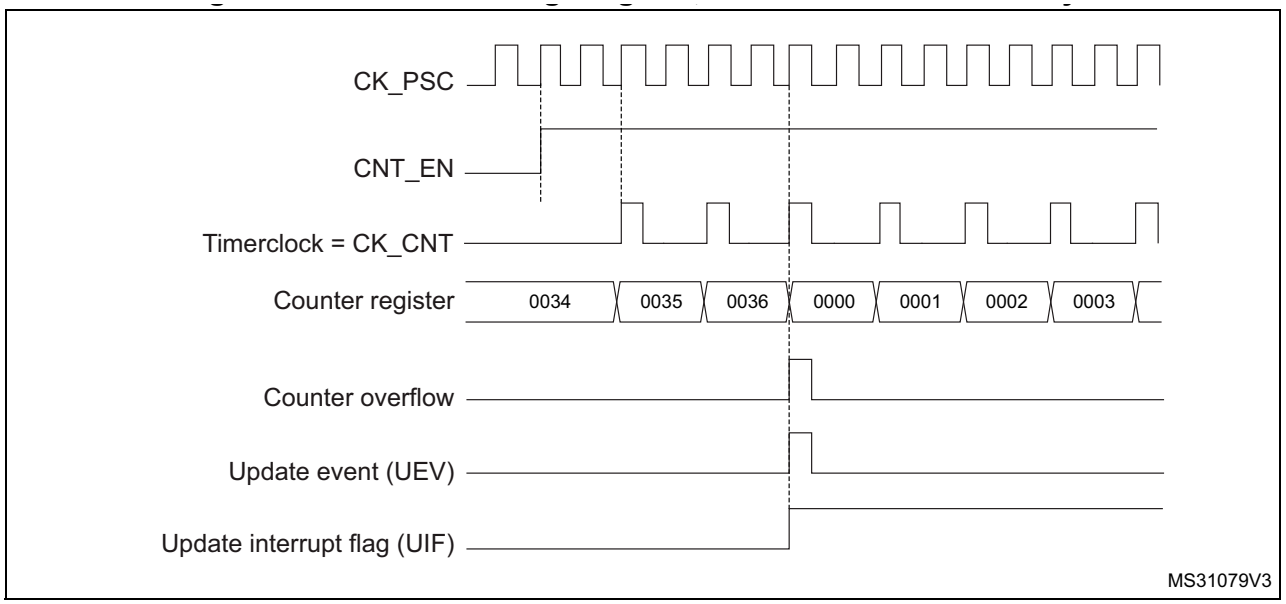
- Счётчик повторений перегружается из регистра TIMx_RCR,
- В скрытый регистр перезагрузки пишется содержимое TIMx_ARR,
- Буфер предделителя перегружается из регистра TIMx_PSC.

Примеры ниже используют TIMx_ARR=0x36.

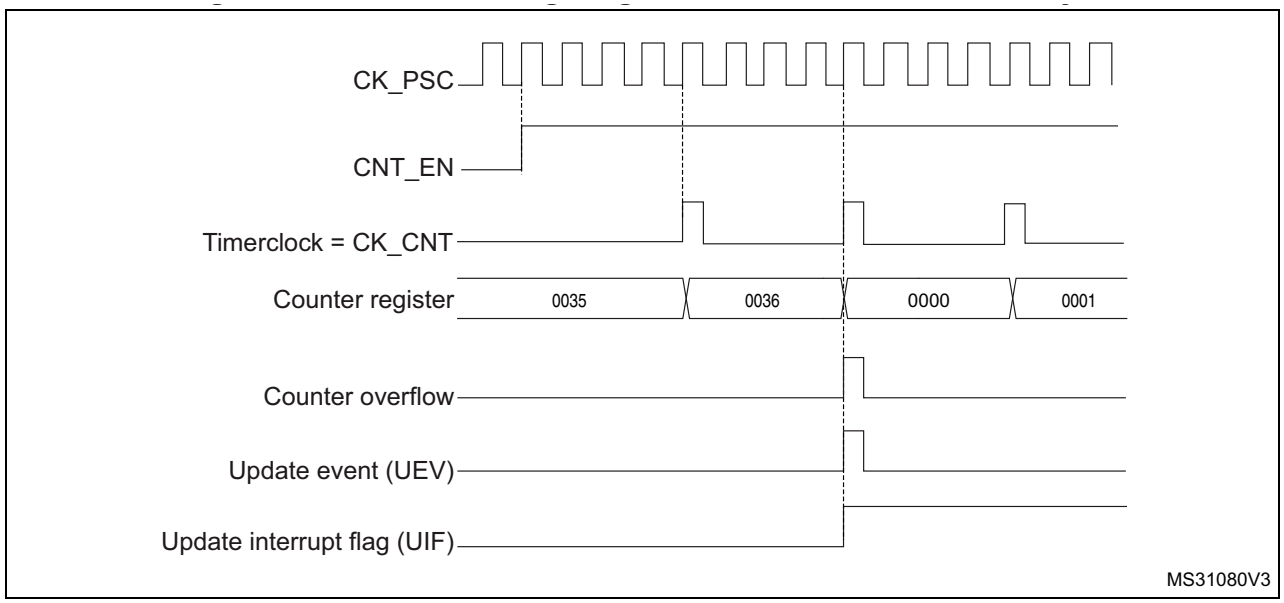
Временная диаграмма с делителем 1.



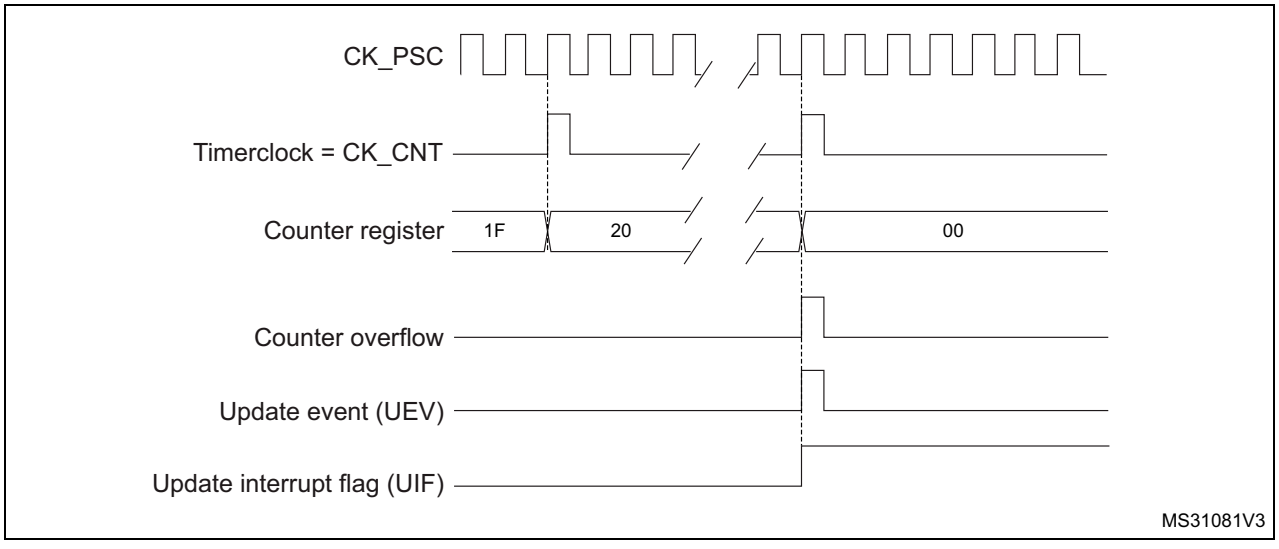
Временная диаграмма с делителем 2.



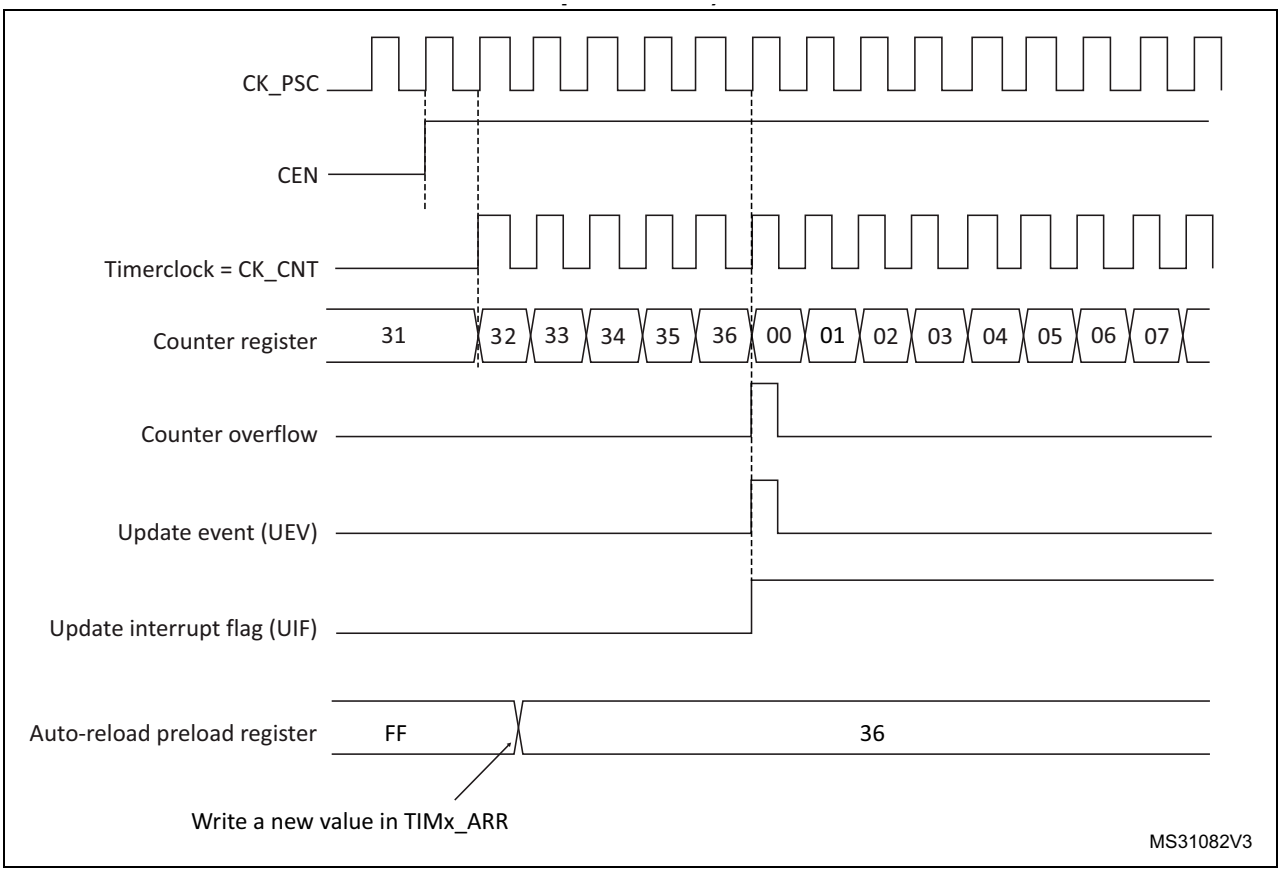
Временная диаграмма с делителем 4.



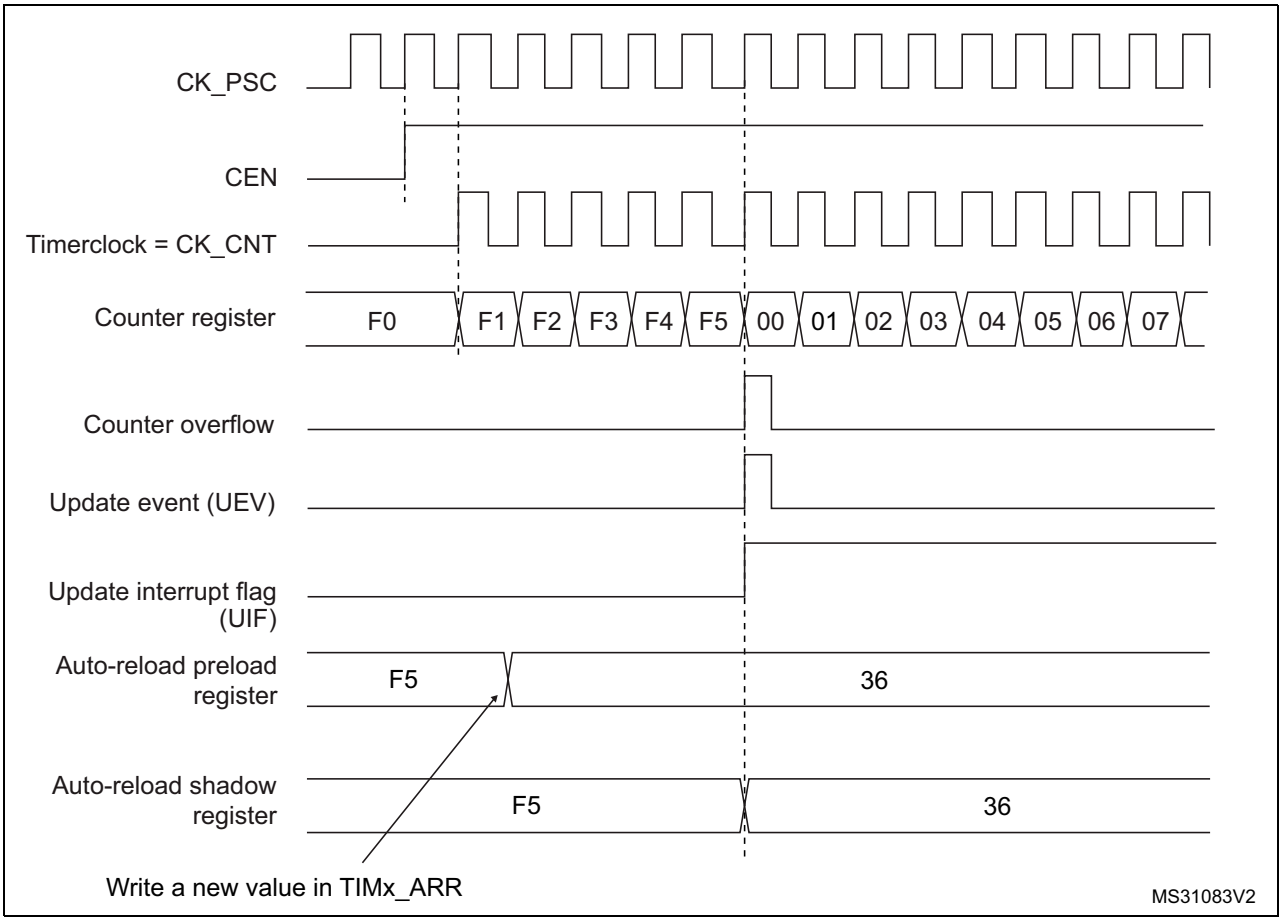
Временная диаграмма с делителем N.



Временная диаграмма события при ARPE=0 (TIMx_ARR не предзагружен).



Временная диаграмма события при ARPE=1 (TIMx_ARR предзагружен).



Обратный счёт

Счёт идёт от значения перезагрузки (регистр TIMx_ARR) до 0, перегружает счётчик и выдаёт событие исчерпания счётчика.

Если используется счётчик повторений, то событие UEV выдаётся после исчерпания заданных повторений плюс 1 (TIMx_RCR+1). Иначе событие обновления выдаётся при каждом исчерпании счётчика.

Установка бита UG в регистре TIMx_EGR (программно или контроллером режима ведомого) также выдаёт событие обновления.

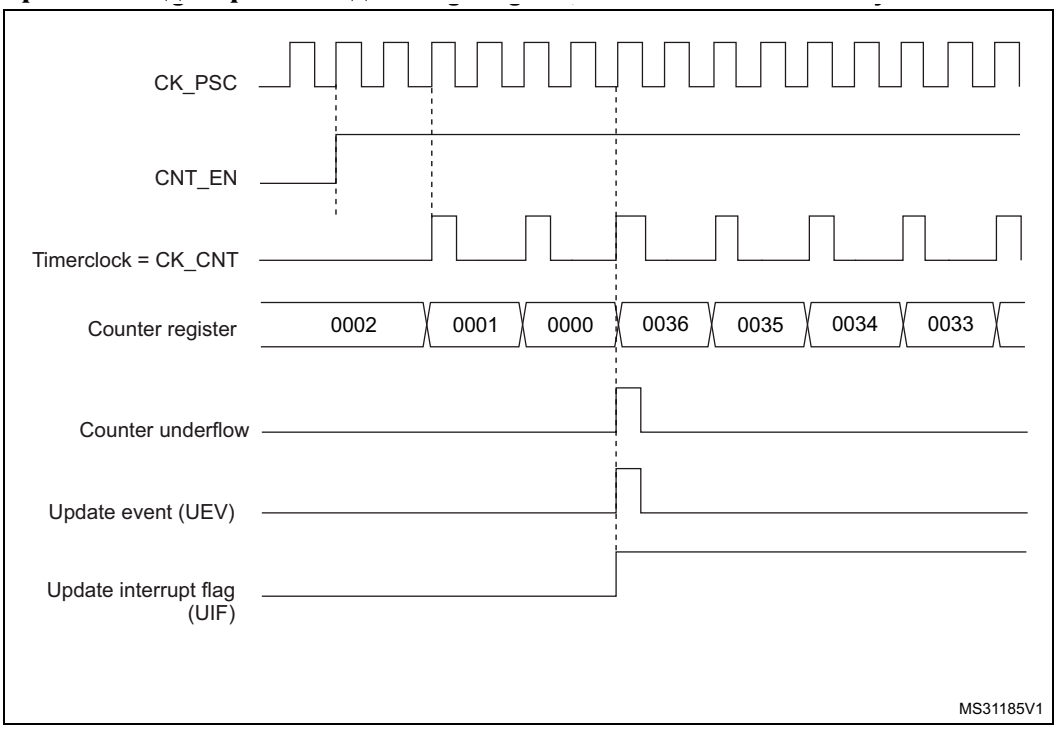
Событие UEV выключается установкой бита UDIS в регистре TIMx_CR1. Этим предупреждается запись скрытого регистра во время изменения регистра предзагрузки. Также событие UEV не появляется при сброшенном бите UDIS 0. Однако, счётчик продолжает считать с значения перезагрузки, тогда как и предделитель с 0 (не изменив своего значения). Кроме того, если стоит бит URS в регистре TIMx_CR1, то установка бита UG выдаёт событие UEV без установки флага UIF (без прерывания и запроса DMA). Этим предупреждается одновременная выдача прерываний обновления и захвата при очистке счётчика по событию захвата.

По событию обновления обновляются все регистры и ставится флаг обновления (бит UIF в регистре TIMx_SR). Это зависит от бита URS:

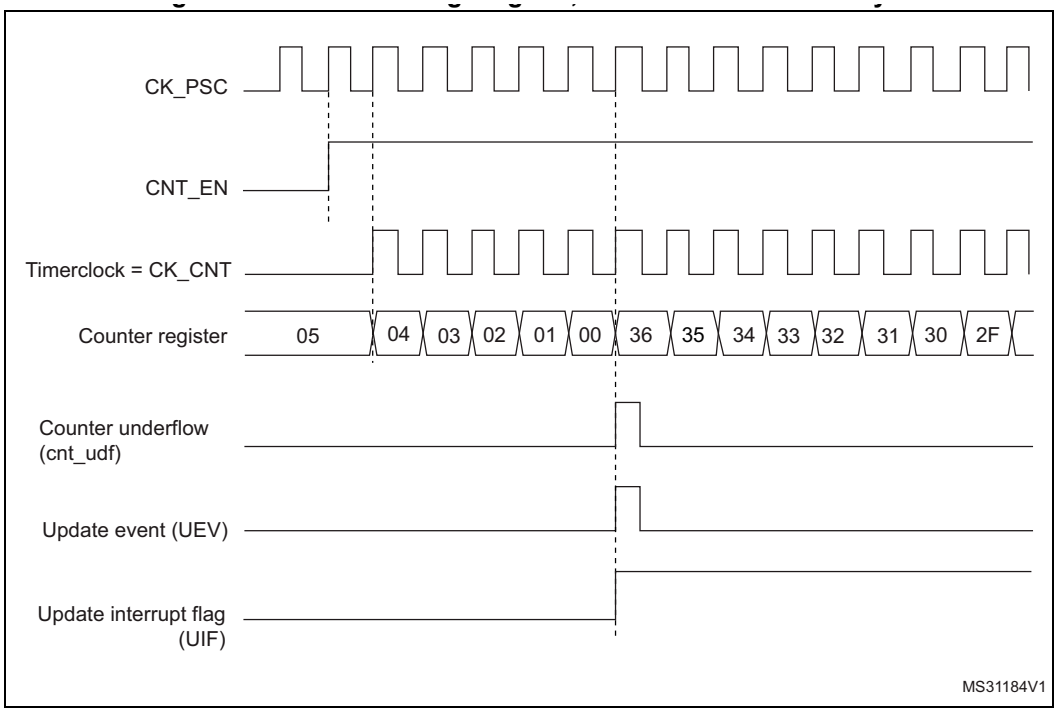
- Счётчик повторений перегружается из регистра TIMx_RCR,
- Буфер предделителя перегружается из регистра TIMx_PSC.
- В скрытый регистр перезагрузки пишется содержимое TIMx_ARR. Скрытый регистр перезагрузки обновляется раньше счётчика и работа продолжается с нового значения.

Примеры ниже используют TIMx_ARR=0x36.

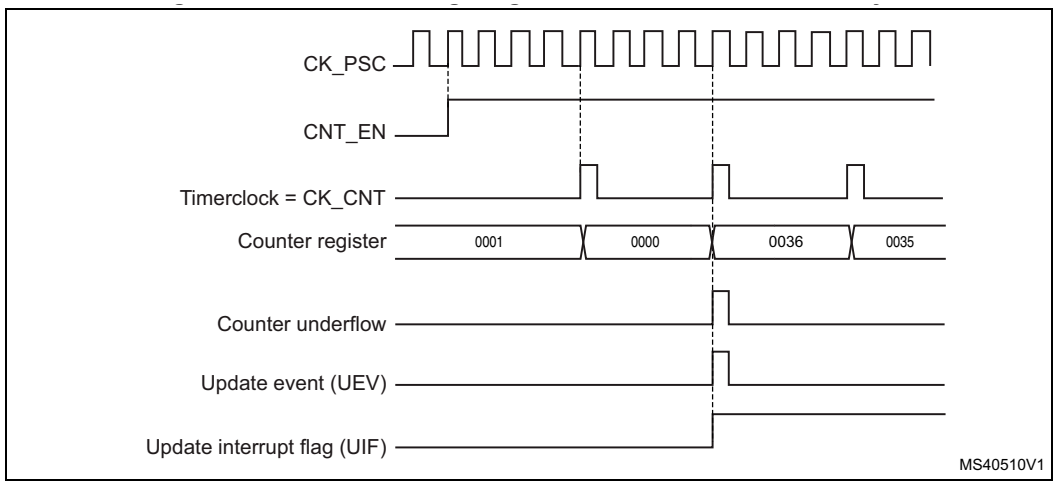
Временная диаграмма с делителем 1.



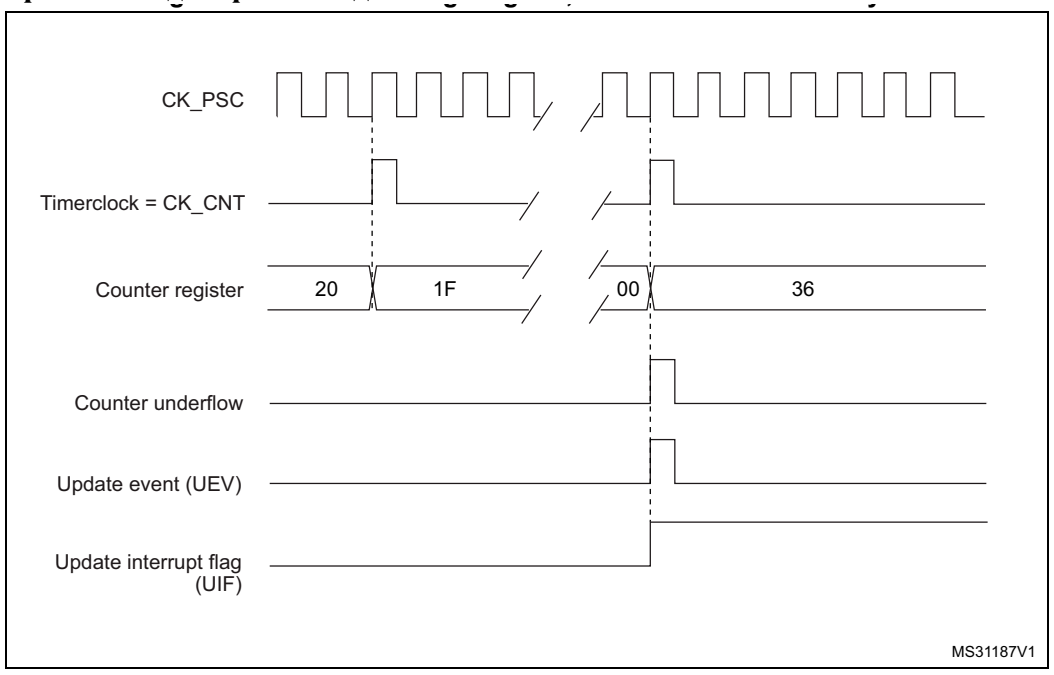
Временная диаграмма с делителем 2.



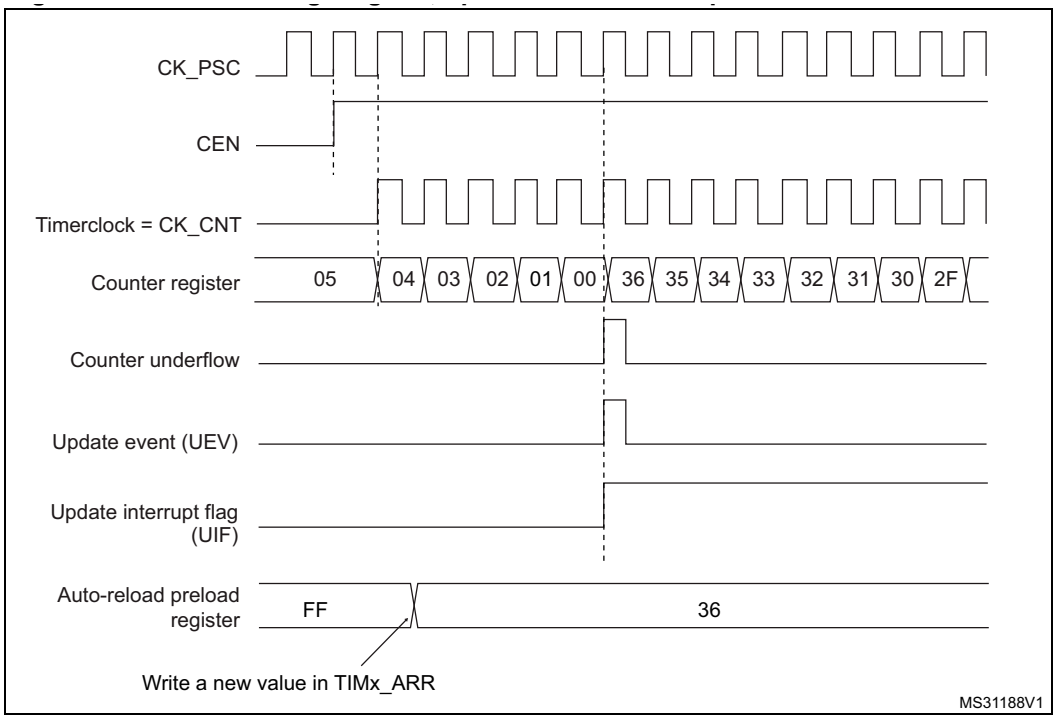
Временная диаграмма с делителем 4.



Временная диаграмма с делителем N.



Временная диаграмма события без счётчика повторений.



Реверсивный счёт

Счёт идёт от 0 до значения перезагрузки минус 1 (регистр TIMx_ARR-1), выдаёт событие переполнения счётчика, затем считает обратно до 1, выдаёт событие исчерпания счётчика и начиняет счёт с 0.

Реверсивный режим включается если биты CMS в регистре TIMx_CR1 не равны '00'. Флаг прерывания Сравнения выхода выводных каналов ставится когда: завершается обратный счёт (Реверсивный режим 1, CMS = "01"), завершается прямой счёт (Реверсивный режим 2, CMS = "10") завершается прямой и обратный счёт (Реверсивный режим 3, CMS = "11").

В этом режиме бит DIR в регистре TIMx_CR1 писать нельзя. Он изменяется аппаратно и указывает текущее направление счёта.

Событие обновления может выдаваться на каждое переполнение и на каждое исчерпание счётчика установкой бита UG в регистре TIMx_EGR (программно или контроллером режима ведомого). В этом случае счётчик и предделитель начинают счёт с 0.

Событие UEV выключается установкой бита UDIS в регистре TIMx_CR1. Этим предупреждается запись скрытого регистра во время изменения регистра предзагрузки. Также событие UEV не появляется при сброшенном бите UDIS 0. Однако, счётчик продолжает считать с вверх и вниз, основываясь на текущем значении предзагрузки.

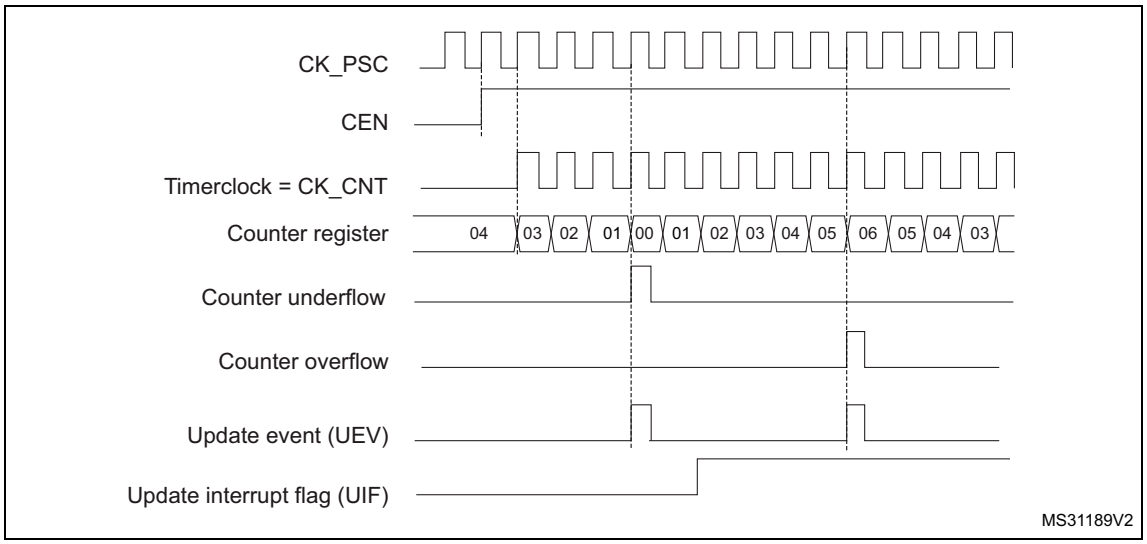
Кроме того, если стоит бит URS в регистре TIMx_CR1, то установка бита UG выдаёт событие UEV без установки флага UIF (без прерывания и запроса DMA). Этим предупреждается одновременная выдача прерываний обновления и захвата при очистке счётчика по событию захвата.

По событию обновления обновляются все регистры и ставится флаг обновления (бит UIF в регистре TIMx_SR). Это зависит от бита URS:

- Счётчик повторений перегружается из регистра TIMx_RCR,
- Буфер предделителя перегружается из регистра TIMx_PSC.
- В скрытый регистр перезагрузки пишется содержимое TIMx_ARR. В случае обновления по переполнению счётчика скрытый регистр перезагрузки обновляется раньше счётчика и работа продолжается с нового значения.

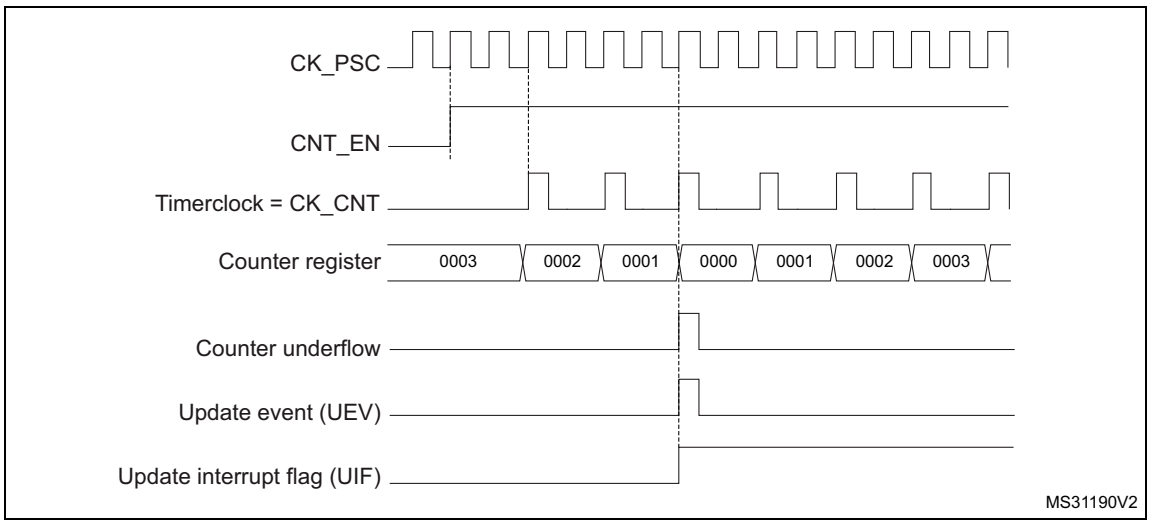
Примеры ниже используют разные частоты тактов.

Временная диаграмма с делителем 1, TIMx_ARR = 0x6.

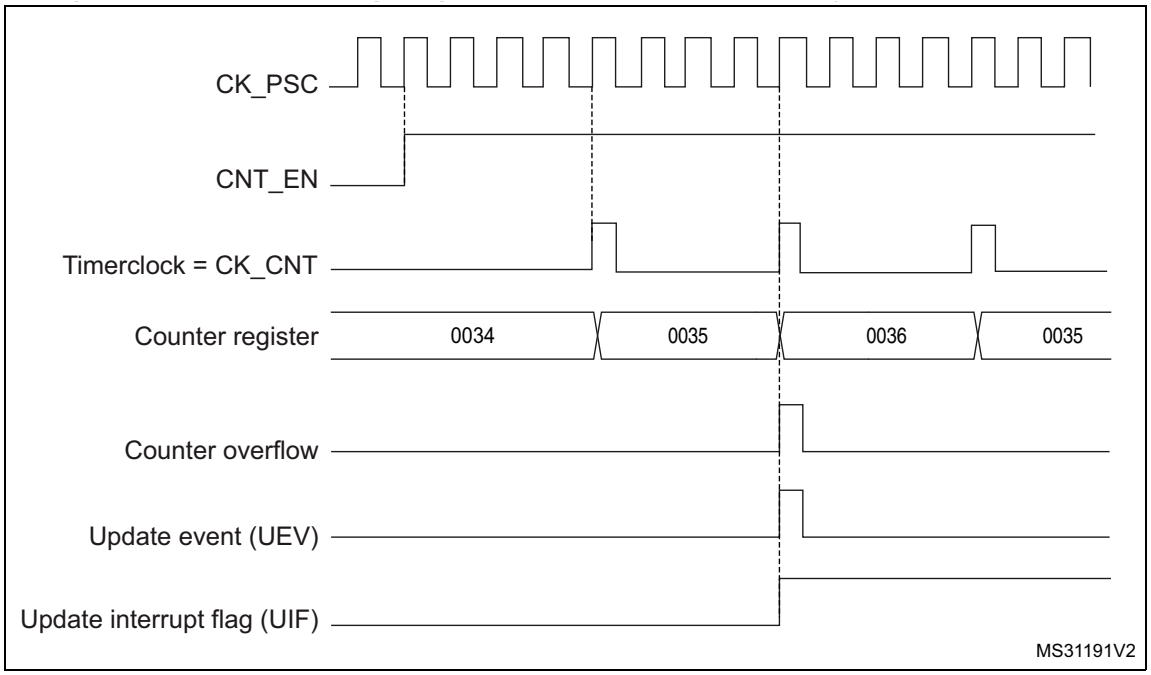


1. Реверсивный режим 1.

Временная диаграмма с делителем 2.

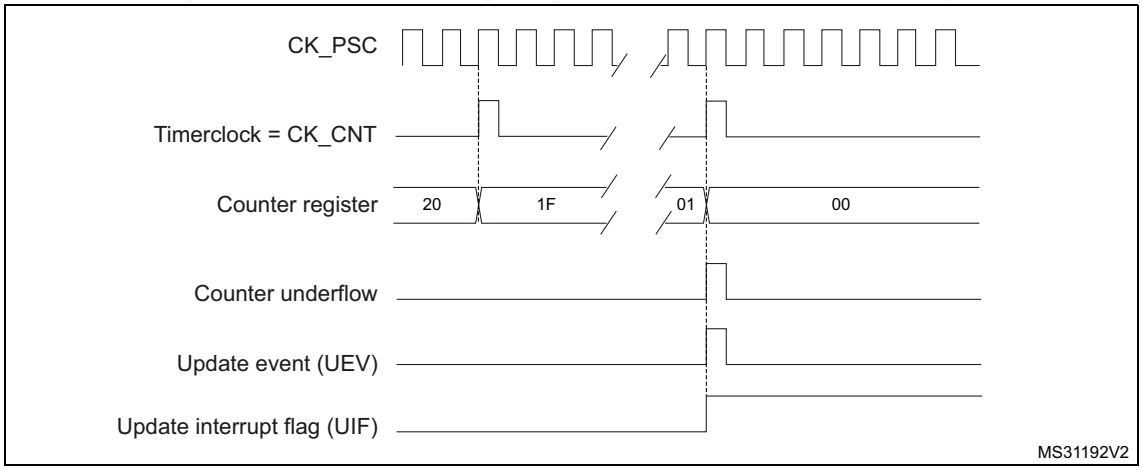


Временная диаграмма с делителем 4, TIMx_ARR=0x36.

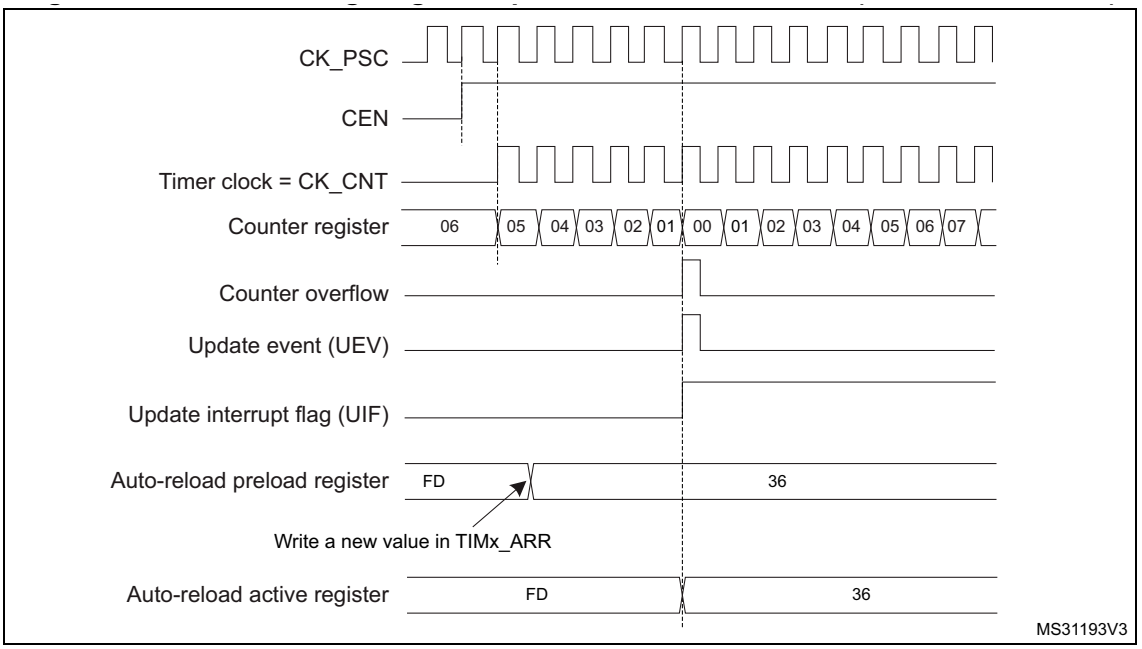


1. Реверсивный режим 2 или 3 с флагом UIF по переполнению.

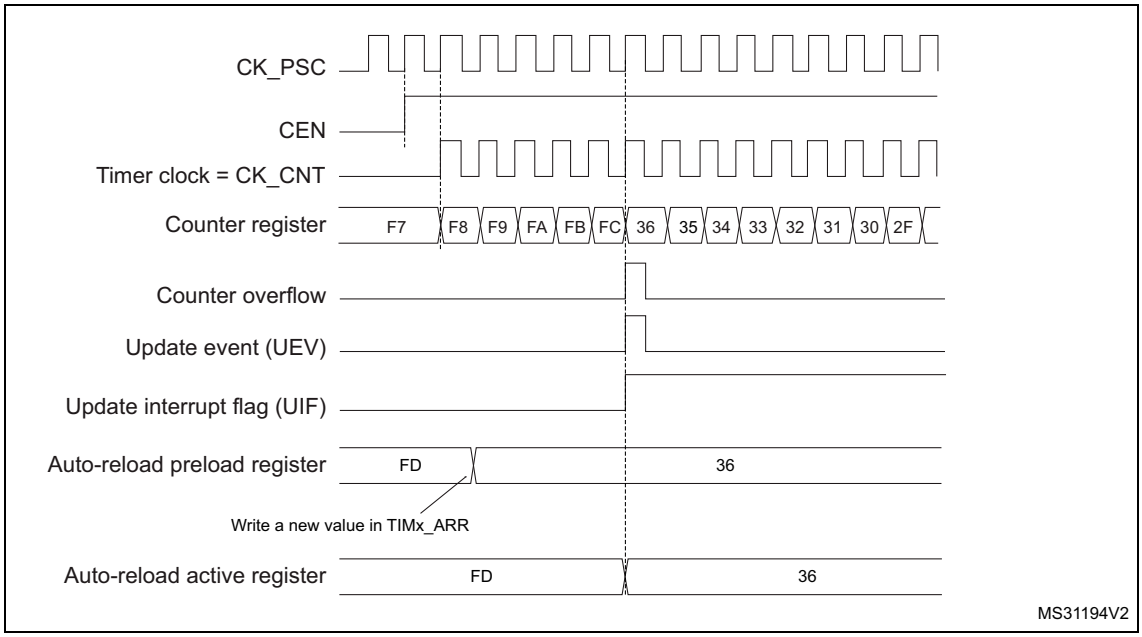
Временная диаграмма с делителем N.



Временная диаграмма, обновление с ARPE=1 (исчерпание).



Временная диаграмма, обновление с ARPE=1 (переполнение).



14.3.3. Счётчик повторения

В действительности событие обновления (UEV) выдаётся только при нулевом счётчике повторения. Это полезно при генерации ШИМ сигналов.

То есть, данные из регистров TIMx_ARR, TIMx_PSC, и TIMx_CCRx пишутся в скрытые регистры каждые N+1 переполнений или исчерпаний счётчика, где N значение счётчика повторений TIMx_RCR.

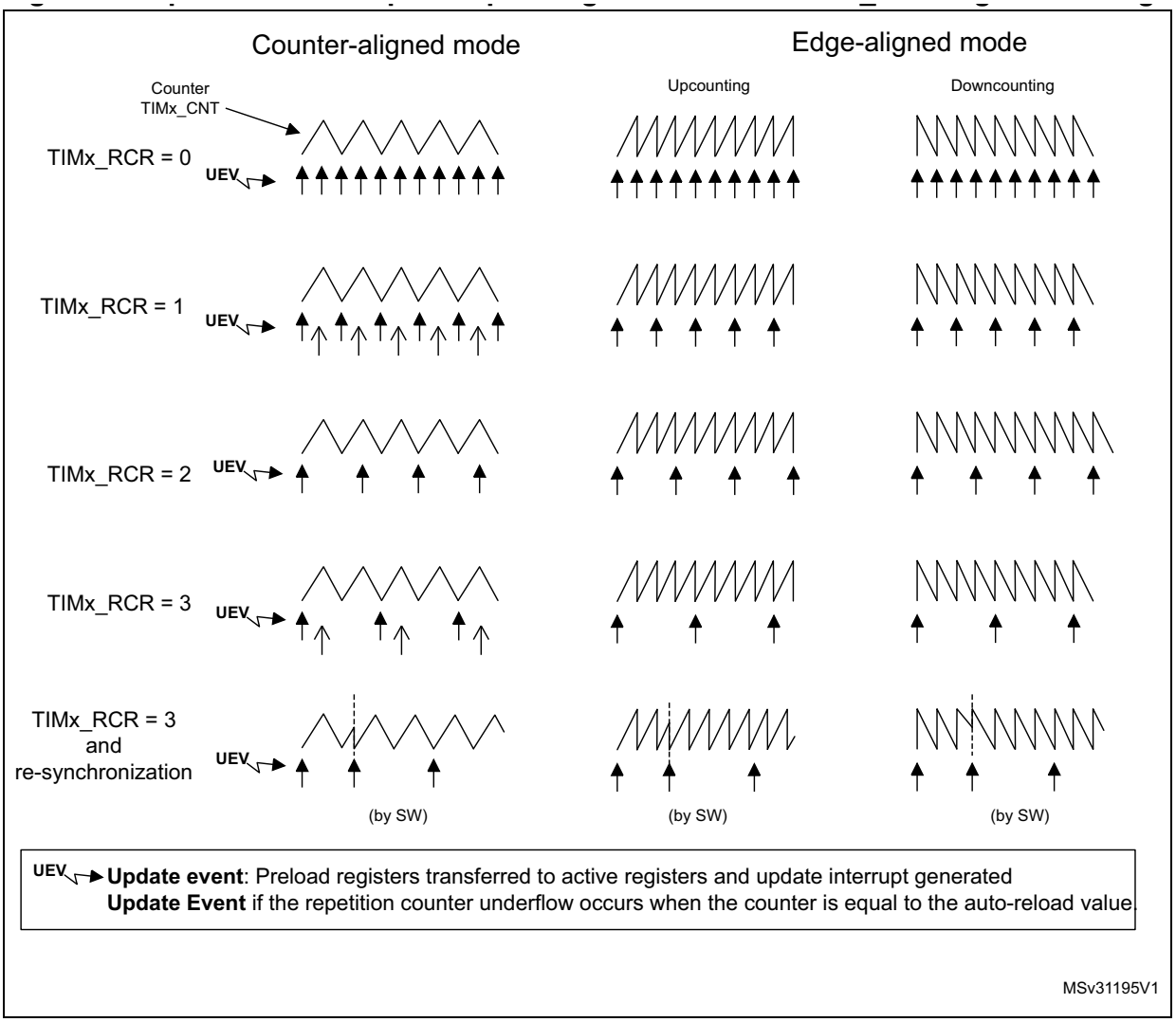
Счётчик повторений декрементируется:

- На каждое переполнение счётчика при прямом счёте,
- На каждое исчерпание счётчика при обратном счёте,
- На каждое переполнение и исчерпание счётчика при реверсивном счёте. Хотя это ограничивает число повторений 128 циклами ШИМ, зато так можно обновлять заполнение цикла дважды за период ШИМ. При обновлении регистров сравнения единожды за период ШИМ в реверсивном режиме максимальное разрешение равно $2xT_{ck}$ из-за симметрии сигнала.

Счётчик повторений сам перегружается из регистра TIMx_RCR. Если событие UEV выдаётся программно (битом UG в регистре TIMx_EGR) или аппаратно контроллером ведомого, то счётчик повторений немедленно загружается из регистра TIMx_RCR, независимо от его содержимого.

В реверсивном режиме при нечётных значениях RCR, событие обновления выдаётся либо при переполнении счётчика, либо при исчерпании, в зависимости от того, когда был записан регистр RCR, до старта счётчика, или после него. Если RCR был записан до старта, то UEV появляется при переполнении, иначе при исчерпании. Например при RCR = 3, событие UEV выдаётся на каждое 4-е переполнение или исчерпание, в зависимости от того, когда был записан RCR.

Обновления в зависимости от установок регистра TIMx_RCR.



14.3.4. Выбор тактов

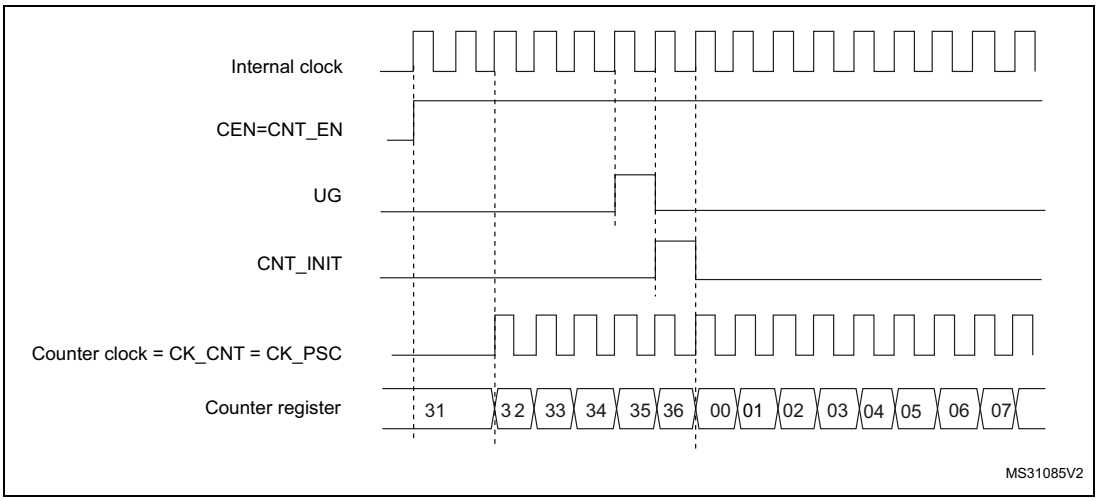
Счётчик может тактироваться из следующих источников:

- Внутренний (CK_INT)
- Внешний режим 1: внешняя ножка
- Внешний режим 2: вход внешнего сигнала ETR
- Входы внутреннего сигнала (ITRx): использование одного таймера предделителем для другого.

Внутренний источник (CK_INT)

Если в ведомом режиме контроллер выключен (SMS=000), то всем управляют только биты CEN, DIR (в регистре TIMx_CR1) и UG (в регистре TIMx_EGR). Они изменяются только программно (кроме UG, снимающегося аппаратно). Сразу после установки бита CEN предделитель начинает получать внутренние такты CK_INT.

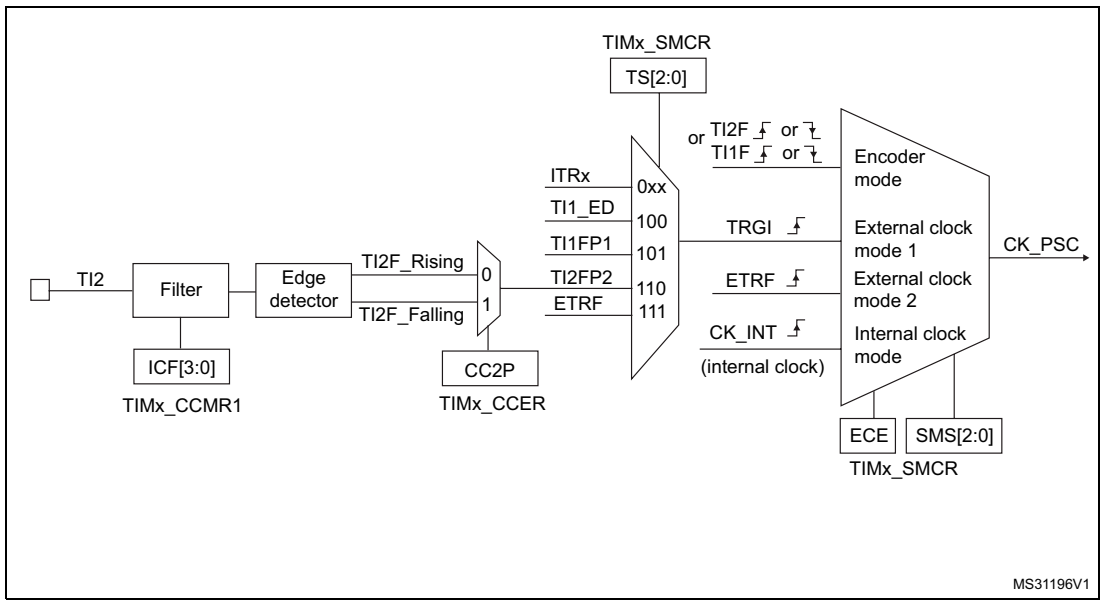
Схема управления в нормальном режиме прямого счёта без предделителя.



Внешний источник режим 1.

Включается при установке SMS=111 в регистре TIMx_SMCR. Счётчик работает от переднего или заднего фронта выбранного входа.

Внешний источник TI2.



Например, процедура включения режима прямого счёта по переднему фронту входа TI2:

1. Включить определение переднего фронта TI2 канала 2 записью CC2S = '01' в регистре TIMx_CCMR1.

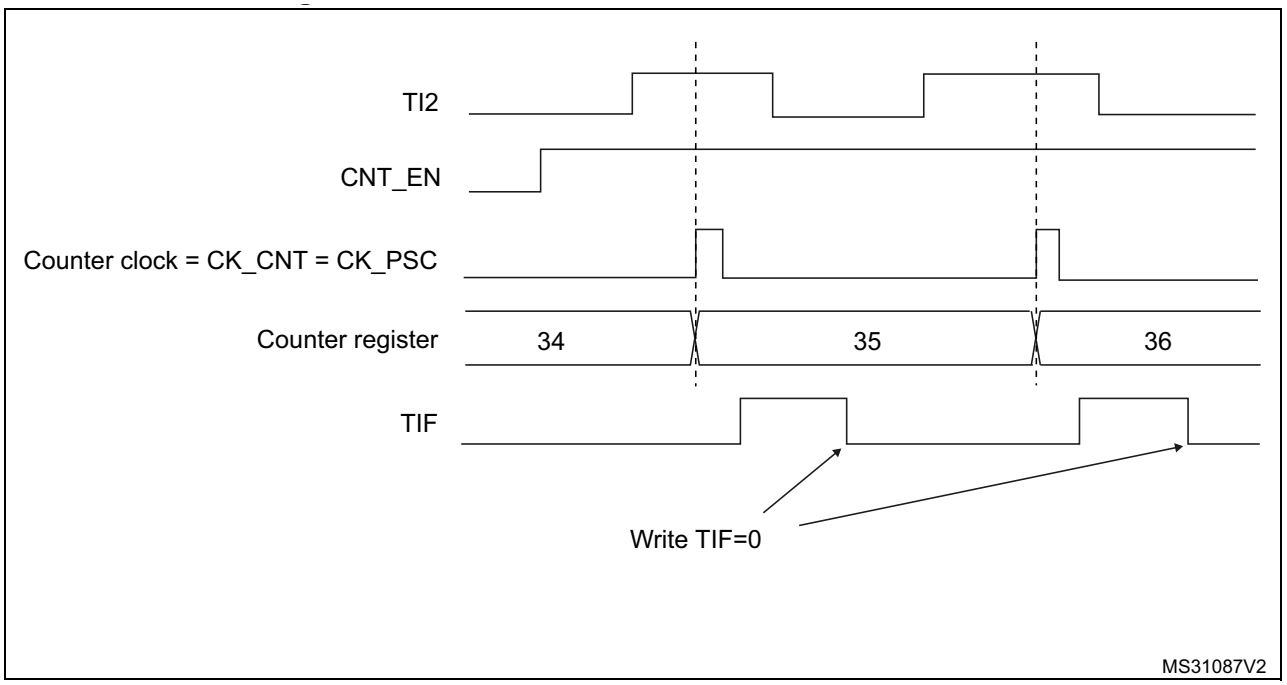
2. Установить длительность входного фильтра битами IC2F[3:0] в регистре TIMx_CCMR1 (если фильтр не нужен, то остаётся IC2F=0000).
3. Выбрать полярность переднего фронта записью CC2P=0 в регистре TIMx_CCER.
4. Включить внешний режим 1 тактирования записью SMS=111 в регистре TIMx_SMCR.
5. Выбрать TI2 как источник сигнала записью TS=110 в регистре TIMx_SMCR.
6. Включить счётчик записью CEN=1 в регистре TIMx_CR1.

Предделитель захвата для запуска не используется и конфигурировать его нужды нет.

По переднему фронту на TI2 счётчик один раз тикает и ставит флаг TIF.

Задержка между передним фронтом на TI2 и реальным тиком появляется из-за схемы ресинхронизации на входе TI2.

Схема управления внешнего режима 1.

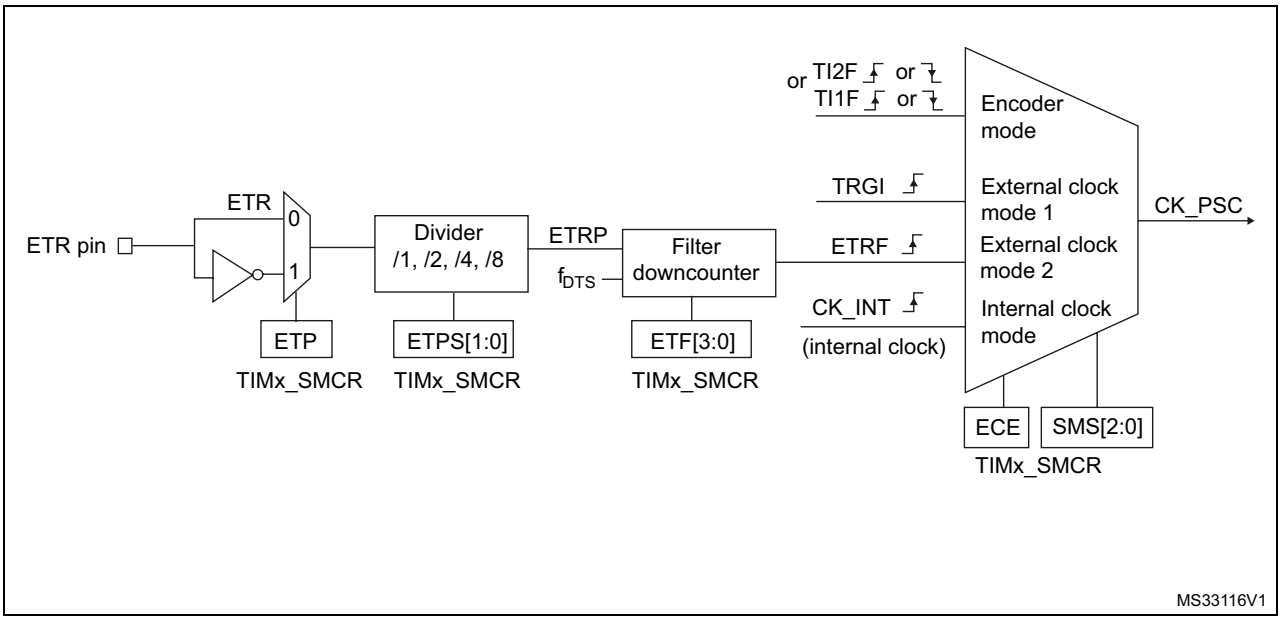


Внешний источник режим 2.

Включается записью ECE=1 в регистре TIMx_SMCR.

Счётчик работает по переднему или заднему фронту входа ETR.

Блок входа внешнего запуска.



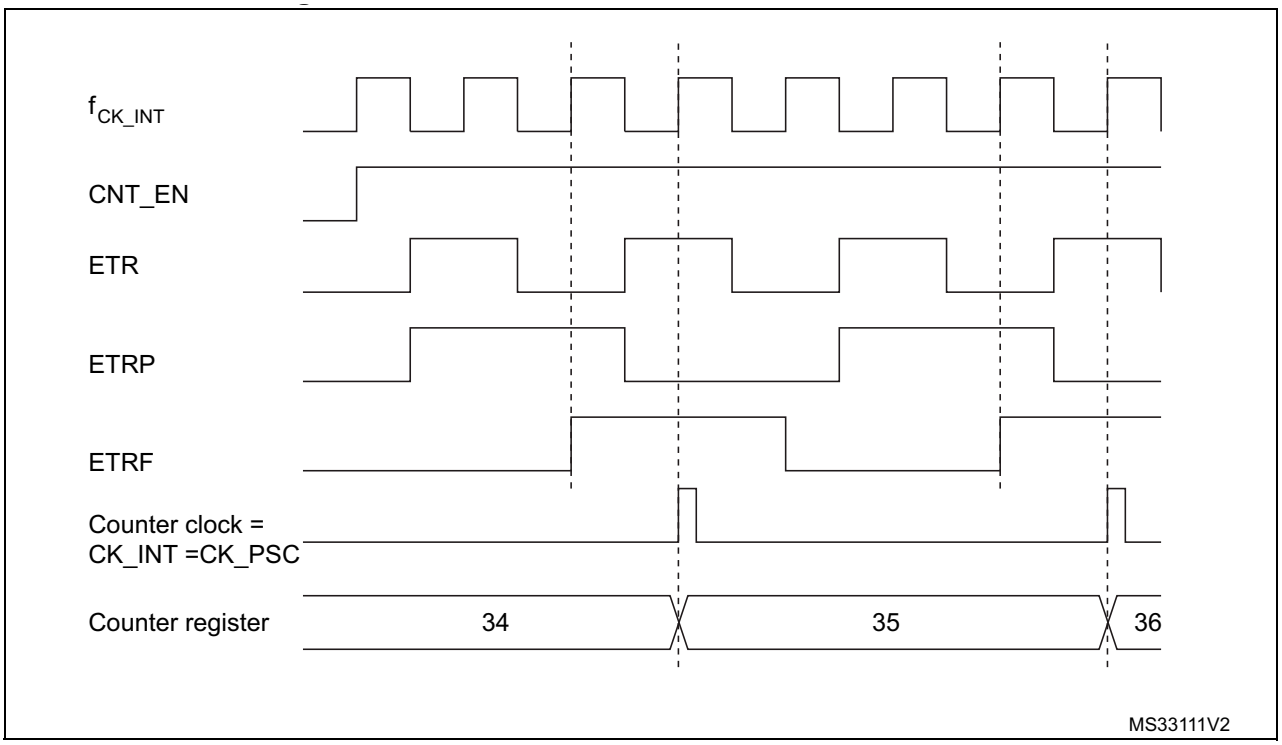
Например, включение режима прямого счёта по каждому 2 переднему фронту на входе ETR:

1. Здесь фильтр не нужен, так что ETF[3:0]=0000 в регистр TIMx_SMCR.
2. В предделитель пишем ETPS[1:0]=01 в регистре TIMx_SMCR
3. Выбираем передний фронт на ножке ETR записью ETP=0 в регистре TIMx_SMCR.
4. Разрешаем внешний режим 2 записью ECE=1 в регистре TIMx_SMCR.
5. Включаем счётчик записью CEN=1 в регистре TIMx_CR1.

Счётчик тикает единожды на каждые 2 передних фронта ETR.

Задержка между передним фронтом на ETR и реальным тиком появляется из-за схемы ресинхронизации на входе сигнала ETRP.

Схема управления с внешним источником режим 2.

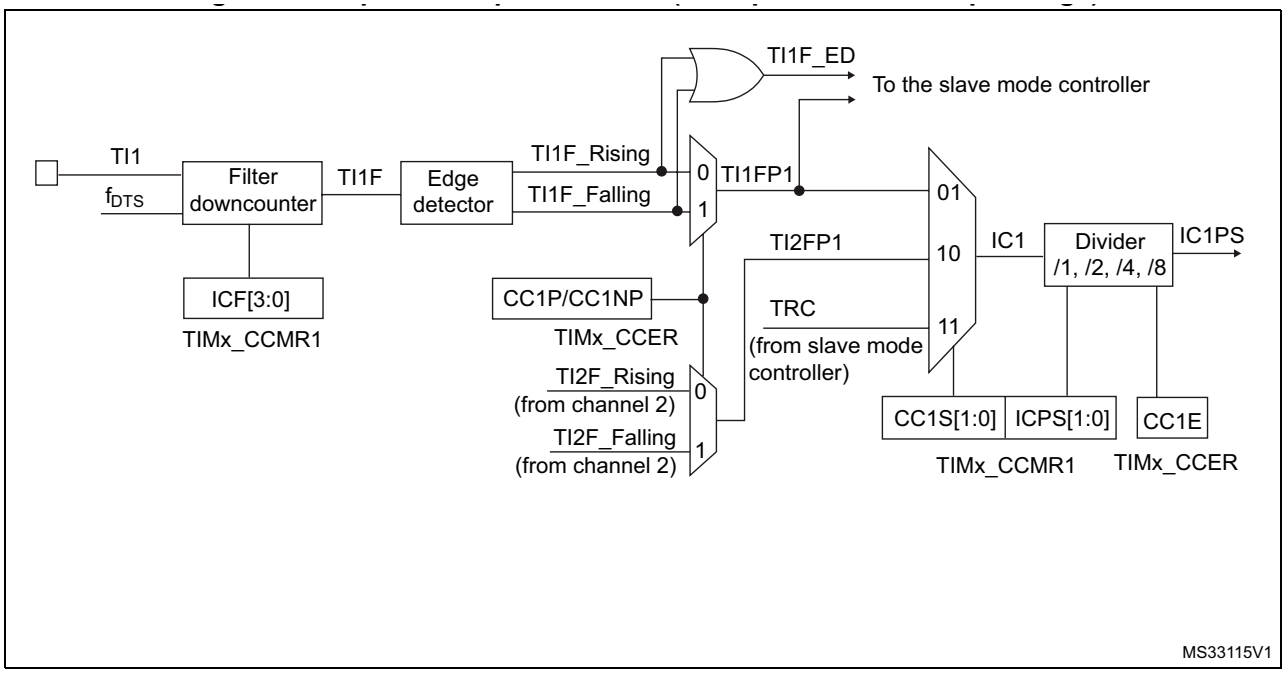


14.3.5. Каналы захвата/сравнения

Каждый канал захвата/сравнения построен вокруг парного регистра (включая невидимый), входного каскада захвата (с цифровым фильтром, мультиплексором и предделителем) и выходного каскада (с компаратором и управлением выходом).

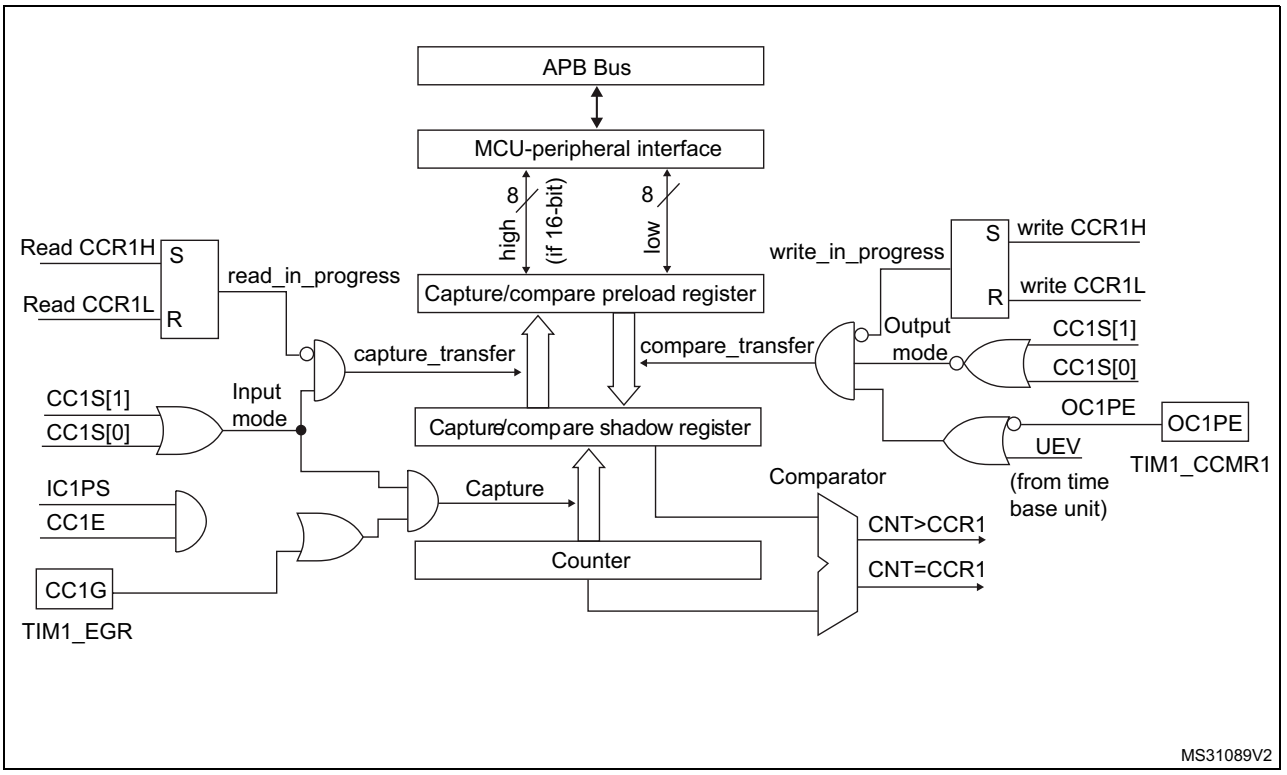
Входной каскад считывает нужный вход TIx для выдачи фильтрованного сигнала TIxF. Далее, детектор фронта с выбором полярности выдаёт сигнал (TIxFPx), что может быть использован как вход запуска контроллером режима ведомого или как команда захвата. Предделитель стоит до регистра захвата (ICxPS).

Входной каскад канала 1.

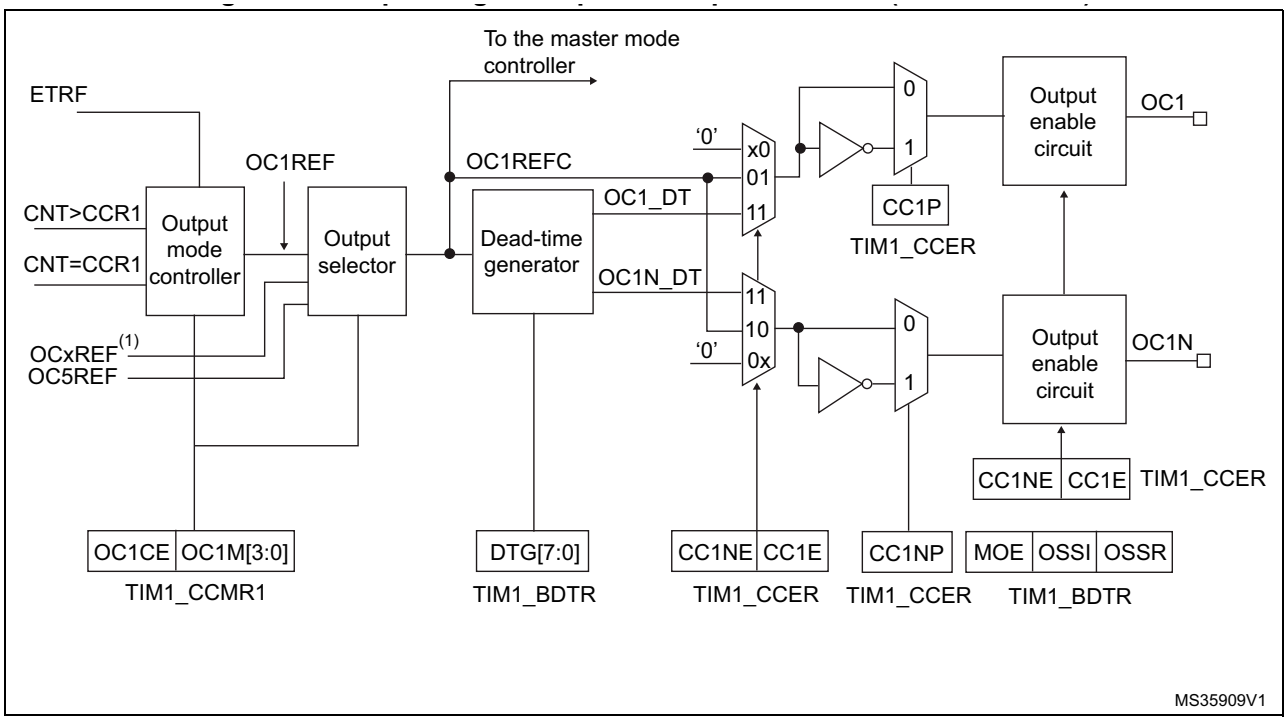


Выходной каскад выдаёт промежуточный сигнал, используемый как опорный: OCxRef (активный высокий). Полярность действует в конце цепочки.

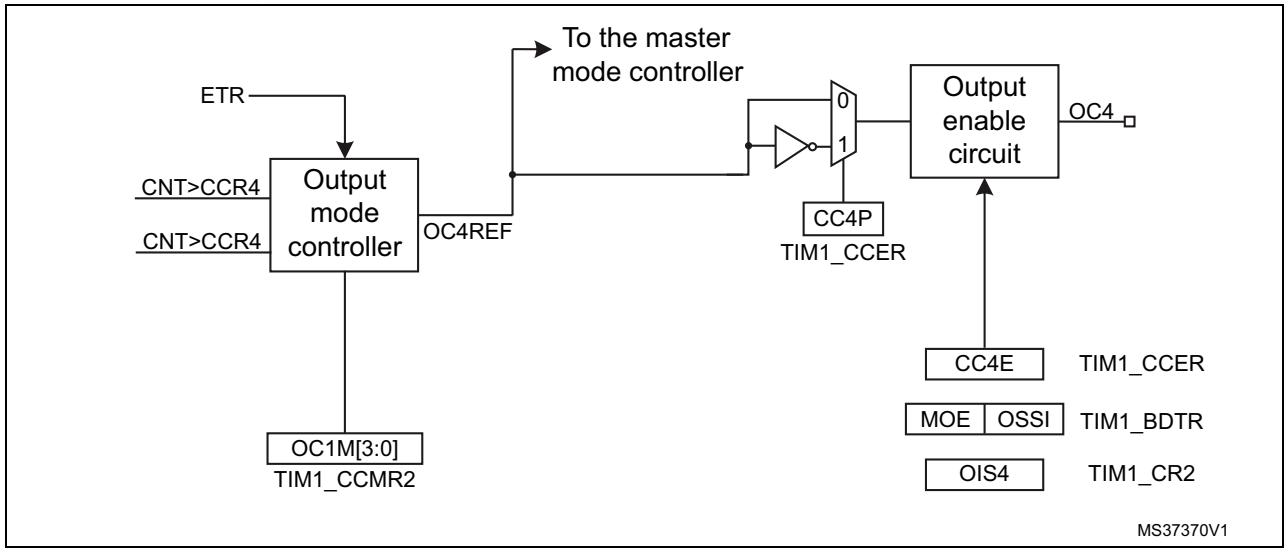
Основная схема канала 1.



Выходной каскад каналов 1 - 3.



Выходной каскад канала 4.



Блок захвата/сравнения состоит из двух регистров: предзагрузки и невидимого. Снаружи доступен только регистр предзагрузки и по чтению и по записи.

Захват выполняется внутренним регистром и результат пишется в регистр предзагрузки.

При сравнении регистр предзагрузки пишется в невидимый, где и сравнивается со счётчиком.

14.3.6. Режим захвата входа

Регистры `TIMx_CCRx` защёлкивают содержимое счётчика после передачи, обнаруженной соответствующим сигналом `ICx`. При этом ставится флаг `CCXIF` в регистре `TIMx_SR` и ставятся запрос DMA и прерывания. Если захват происходит при стоящем флаге `CCxIF`, то ставится флаг перезахвата `CCxOF` в регистре `TIMx_SR`. Флаг `CCxIF` снимается записью в него нуля или чтением захваченных данных из регистра `TIMx_CCRx`. `CCxOF` снимается записью '`0`'.

Далее захватываем содержимое `TIMx_CCR1` по переднему фронту `TI1`. Делаем так:

- Выбираем активный вход: `TIMx_CCR1` подключаем в `TI1` и пишем "`01`" в биты `CC1S` регистра `TIMx_CCMR1`. Биты `CC1S` отличаются от `00`, т. е. канал на вводе и регистр `TIMx_CCR1` доступен только по чтению.

- Ставим длительность входного фильтра битами ICxF в регистре TIMx_CCMRx с учётом входного сигнала на TIx. Допустим, что при переключении входной сигнал нестабилен в течении пяти внутренних тактов. Значит длительность фильтра должна быть больше этих пяти тактов. Мы можем определить передачу на TI1 за 8 последовательных выборок нового уровня на частоте f_{DTS}. И мы пишем 0011 в биты IC1F регистра TIMx_CCMR1.
- Выбираем фронт сигнала на TI1 записью 0 в бит CC1P регистра TIMx_CCER (передний).
- Ставим предделитель. Здесь он нам не нужен (пишем 00 в биты IC1PS регистра TIMx_CCMR1).
- Разрешаем запись счётчика в регистр захвата установкой бита CC1E в TIMx_CCER.
- Если нужно разрешаем выдачу запросов прерывания битом CC1IE в TIMx_DIER и DMA битом CC1DE в регистре TIMx_DIER.

Если захват входа произошёл, то:

- Регистр TIMx_CCR1 получает значение счётчика активной передачи.
- Ставится флаг прерывания CC1IF. Если он ещё стоял, то ставится и флаг CC1OF.
- В зависимости от бита CC1IE генерируется прерывание.
- В зависимости от бита CC1DE генерируется запрос DMA.

Чтобы обработать перезахват рекомендуется читать регистр данных до опроса флага перезахвата.

NB: Запросы прерывания и DMA IC можно выдавать программно установкой битов CCxG в регистре TIMx_EGR.

14.3.7. Режим ввода ШИМ

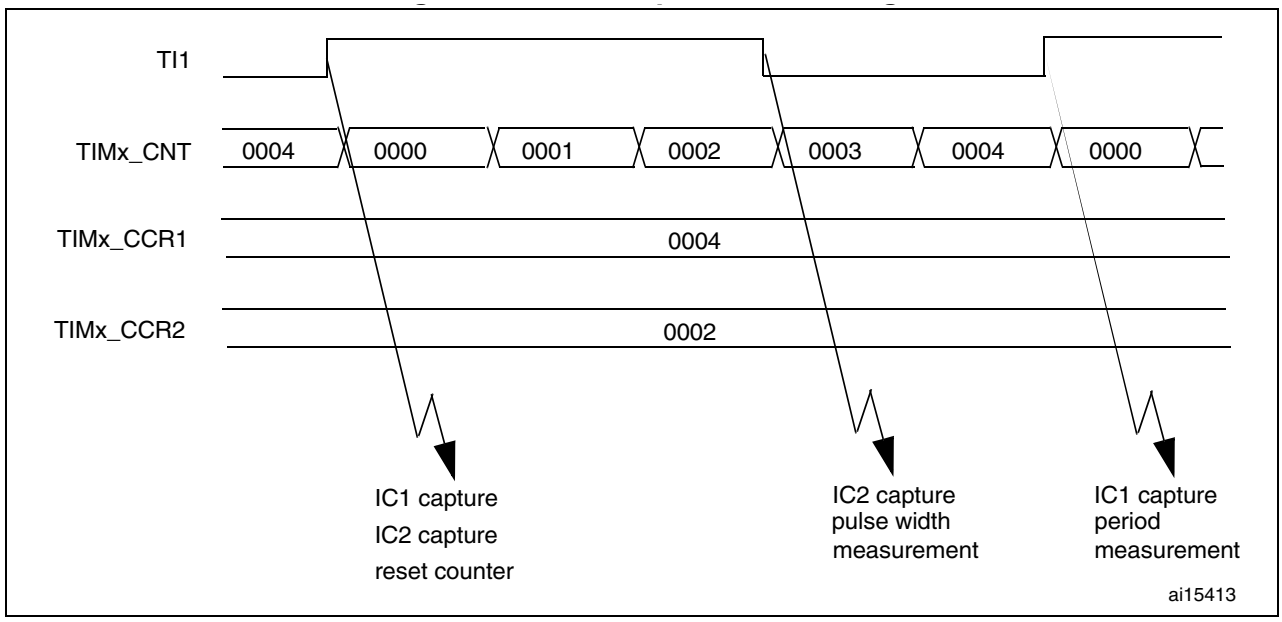
Это часть режима захвата входа. Процедура такая же, кроме:

- Два сигнала ICx направляются на на один вход TIx.
- Полярность этих входов ICx противоположная.
- Сигналом запуска берётся один из двух сигналов TIxFP, а контроллер ведомого режима выключается.

Например, можно измерить период (в регистре TIMx_CCR1) и длительность заполнения (в регистре TIMx_CCR2) ШИМ сигнала на TI1 следующей процедурой (зависит от частоты CK_INT и предделителя):

- Выбрать активный вход для TIMx_CCR1: Записать 01 в биты CC1S регистра TIMx_CCMR1 (TI1).
- Выбрать полярность для TI1FP1 (для захвата в TIMx_CCR1 и очистки счётчика): записать 0 в CC1P (передний фронт).
- Выбрать активный вход для TIMx_CCR2: записать 10 в биты CC2S регистра TIMx_CCMR1 (TI1).
- Выбрать полярность для TI1FP2 (для захвата в TIMx_CCR2): записать 1 в CC2P (задний фронт).
- Выбрать вход запуска: записать 101 в биты TS регистра TIMx_SMCR (TI1FP1).
- Поставить контроллер режима ведомого в сброс: записать 100 в биты SMS регистра TIMx_SMCR.
- Включить захват: поставить биты CC1E и CC2E в регистре TIMx_CCER.

Времянка режима ввода ШИМ.



1. Режим ШИМ может использоваться только с сигналами TIMx_CH1/TIMx_CH2 так как к контроллеру режима ведомого подключены только TI1FP1 и TI2FP2.

14.3.8. Принудительный вывод

В режиме вывода (биты CCxS = 00 в TIMx_CCMRx), каждый выходной сигнал сравнения (OCxREF и затем OCx/OCxN) можно программно поставить в активное или неактивное состояние, независимо от результата сравнения выходного регистра сравнения и счётчика.

Активными выходные сигналы (OCxREF/OCx) ставятся записью 101 в биты OCxM нужного регистра TIMx_CCMRx. Активный сигнал OCxREF всегда высокий, а OCx противоположен биту полярности CCxP.

Сигнал OCxREF снимается записью 100 в биты OCxM регистра TIMx_CCMRx.

Но сравнение внутреннего регистра TIMx_CCRx со счётчиком всё равно выполняется, биты ставятся, генерируются запросы прерывания и DMA. См. ниже.

14.3.9. Режим сравнения выхода

Используется для управления выходным сигналом и определения истечения периода времени.

При совпадении регистра захвата/сравнения со счётчиком эта схема:

- Ставит выходную ножку в заданное состояние (биты OCxM в регистре TIMx_CCMRx) и полярность (бит CCxP в регистре TIMx_CCER). Ножка может хранить состояние(OCxM=000), стать активной (OCxM=001), неактивной (OCxM=010) или переключиться (OCxM=011).
- Ставит флаг прерывания (бит CCxIF в регистре TIMx_SR).
- При установленной маске прерывания (бит CCxIE в регистре TIMx_DIER) выдаёт запрос.
- При установленном разрешении DMA (бит CCxDE в регистре TIMx_DIER) выдаёт нужный (бит CCDS в регистре TIMx_CR2) запрос.

В регистр TIMx_CCRx можно писать как через регистр предзагрузки, так и без него (см. бит OCxPE в регистре TIMx_CCMRx).

В режиме сравнения выхода бит события UEV на выходы OCxREF и OCx не влияет. Разрешение составляет один счёт счётчика. Этот режим можно использовать для выдачи одного импульса.

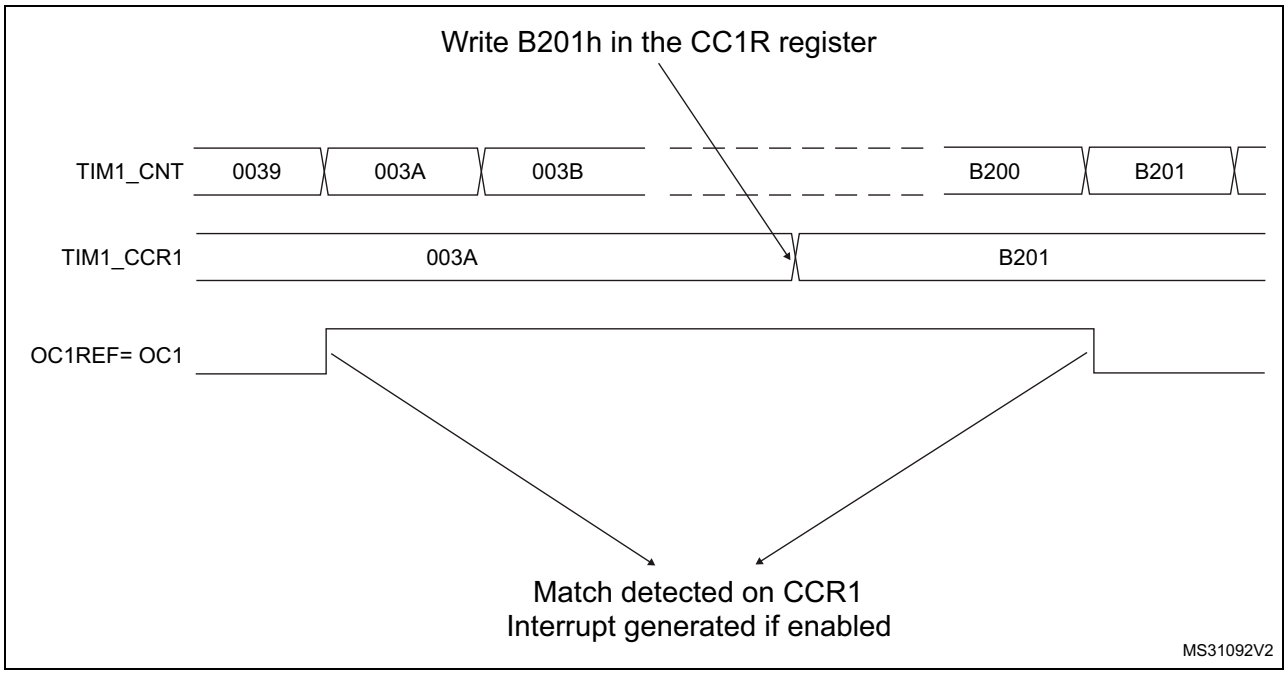
Процедура:

1. Выбрать такты счётчика (внутренние, внешние, предделитель).
2. Записать желаемое в регистры TIMx_ARR и TIMx_CCRx.
3. Если нужно прерывание, то установить бит CCxIE.

4. Выбрать режим выхода. Например:
   - Записать OCxM = 011 для переключения ножки OCx при совпадении CNT и CCRx
   - Записать OCxPE = 0 для отключения регистра предзагрузки
   - Записать CCxP = 0 ради высокого активного уровня
   - Записать CCxE = 1 для разрешения вывода
5. Включить счётчик установкой бита CEN в регистре TIMx_CR1.

При запрещённом регистре предзагрузки (OCxPE='0') регистр TIMx_CCRx можно обновлять в любое время, иначе он будет обновлён только по событию UEV.

Режим сравнения выхода, переключение ножки OC1.



14.3.10.Режим ШИМ

Частота сигнала ШИМ определяется регистром TIMx_ARR, а коэффициент заполнения регистром TIMx_CCRx.

Режим ШИМ может включаться независимо по одному на каждый выход OCx записью '110' (ШИМ1) или '111' (ШИМ2) в биты OCxM регистра TIMx_CCMRx. Соответствующий регистр предзагрузки должно включать битом OCxPE в регистре TIMx_CCMRx и, соответсвенно, регистр авто-загрузки (в режимах прямого и реверсивного счёта) нужно включить битом ARPE в регистре TIMx_CR1.

Поскольку регистр предзагрузки пишется в теневой регистр только по событию обновления, то все регистры нужно заранее инициализировать установкой бита UG в регистре TIMx_EGR.

Полярность OCx (активный высокий или низкий) выбирается битом CCxP в регистре TIMx_CCER. Выходы OCx включаются битами CCxE, CCxNE, MOE, OSSI и OSSR в регистрах TIMx_CCER и TIMx_BDTR.

В режиме ШИМ (1 или 2), TIMx_CNT и TIMx_CCRx сравниваются на выполнение условия TIMx_CCRx ≤ TIMx_CNT или TIMx_CNT ≤ TIMx_CCRx (в зависимости от направления счёта).

Бит CMS в регистре TIMx_CR1 определяет генерацию ШИМ по фронту или по центру.

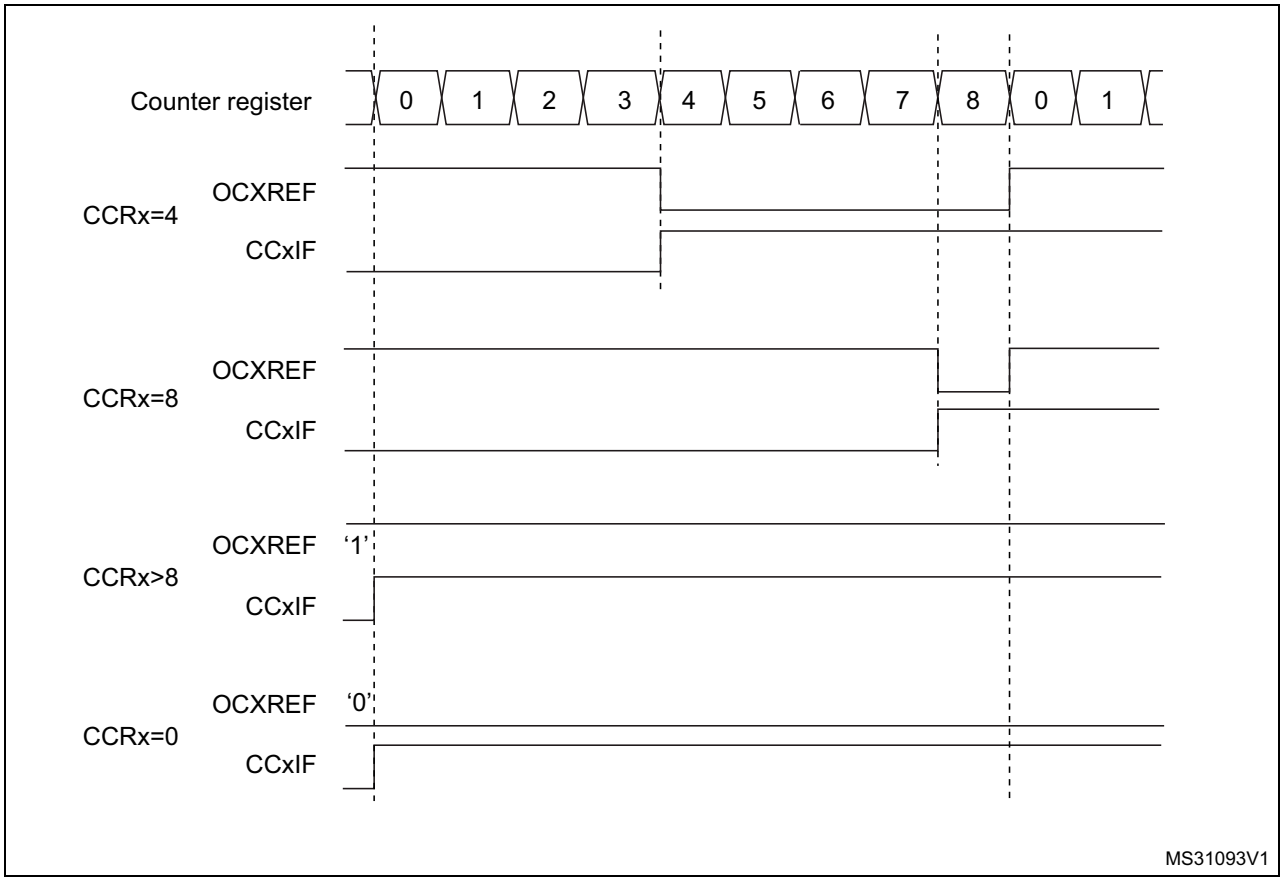
ШИМ по фронту

• Прямой счёт

Он определяется сброшенным битом DIR в регистре TIMx_CR1.

В примере ниже стоит режим ШИМ1. Опорный сигнал OCxREF остаётся высоким при TIMx_CNT < TIMx_CCRx. Если регистр TIMx_CCRx больше значения авто-загрузки (TIMx_ARR) то OCxREF равен '1'. При равенстве значений сравнения, OCxREF равен '0'. В примере ниже TIMx_ARR=8.

ШИМ по фронту (`TIMx_ARR=8`).



- Обратный счёт

  Он определяется стоящим битом `DIR` в регистре `TIMx_CR1`.

  В режиме ШИМ1 опорный сигнал `OCxREF` остаётся низким при `TIMx_CNT` > `TIMx_CCRx`. Если регистр `TIMx_CCRx` больше значения авто-загрузки (`TIMx_ARR`) то `OCxREF` равен '1'. При равенстве значений сравнения, `OCxREF` равен '1'. 0% ШИМ в этом режиме недоступен.

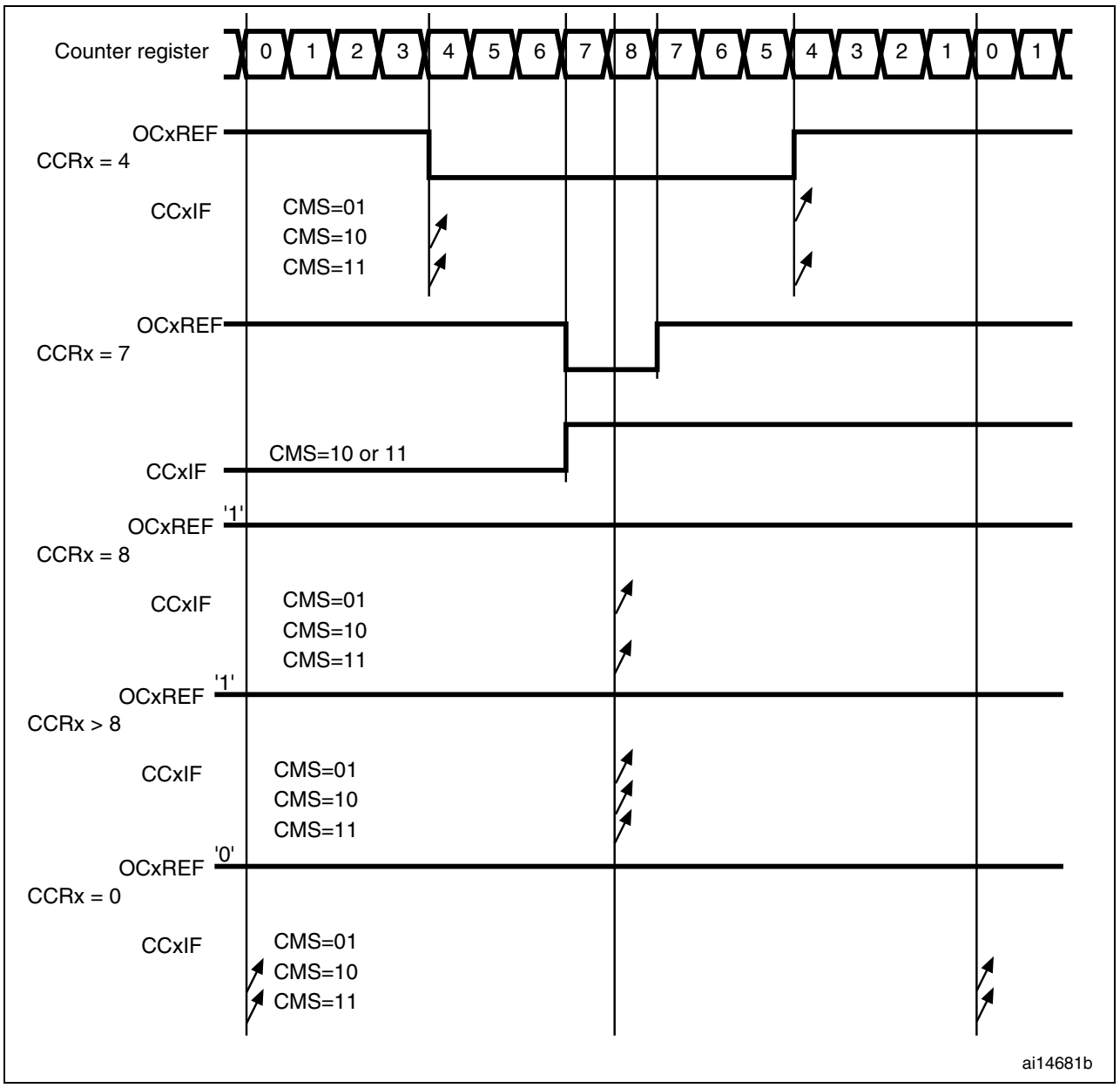
ШИМ по центру

Этот режим включается когда биты `CMS` в регистре `TIMx_CR1` не равны '00' (остальные конфигурации на сигналы `OCxREF`/`OCx` влияют также). Флаг сравнения ставится в любых режимах счёта (прямом, обратном и реверсивном) в зависимости от битов `CMS` в регистре `TIMx_CR1`. Бит направления (`DIR`) в регистр `TIMx_CR1` меняется аппаратно и не дай бог менять его программно.

В примере ниже:

- `TIMx_ARR`=8,
- Стоит режим ШИМ1,
- Флаг ставится при обратном счёте в режиме ШИМ1 по центру (`CMS`=01 в `TIMx_CR1`).

ШИМ по центру (`TIMx_ARR=8`).



Советы:

- При старте режима по центру используется текущее направление счёта (зависит от бита `DIR` в регистре `TIMx_CR1`). Более того, в это время биты `DIR` и `CMS` программно менять нельзя.
- Запись в счётчик работающего режима ой как не рекомендуется. В частности:
  - При `TIMx_CNT`>`TIMx_ARR` направление счёта не изменяется и продолжает считать в том же направлении.
  - При записи 0 или изменении `TIMx_ARR` направление изменяется, но событие `UEV` не выдаётся.
- Самый безопасный способ это программно запустить обновление (битом `UG` в регистре `TIMx_EGR`) прямо перед стартом, а не писать в работающий счётчик.

14.3.11.Инверсные выходы и задержка

Таймеры TIM1 и TIM8 могут выдавать инверсные выходы одного сигнала с некоторой задержкой друг от друга, что могут потребовать характеристики управляемых устройств.

Полярность выходов (основного `OCx` и инверсного `OCxN`) можно выбирать независимо. Этому служат биты `CCxP` и `CCxNP` в регистре `TIMx_CCER`.

Сигналы OCx и OCxN активируются комбинацией битов: CCxE и CCxNE в регистре TIMx_CCER и битами MOE, OISx, OISxN, OSSI и OSSR в регистрах TIMx_BDTR и TIMx_CR2. В частности задержка включается при переходе в состояние IDLE (MOE падает в 0).

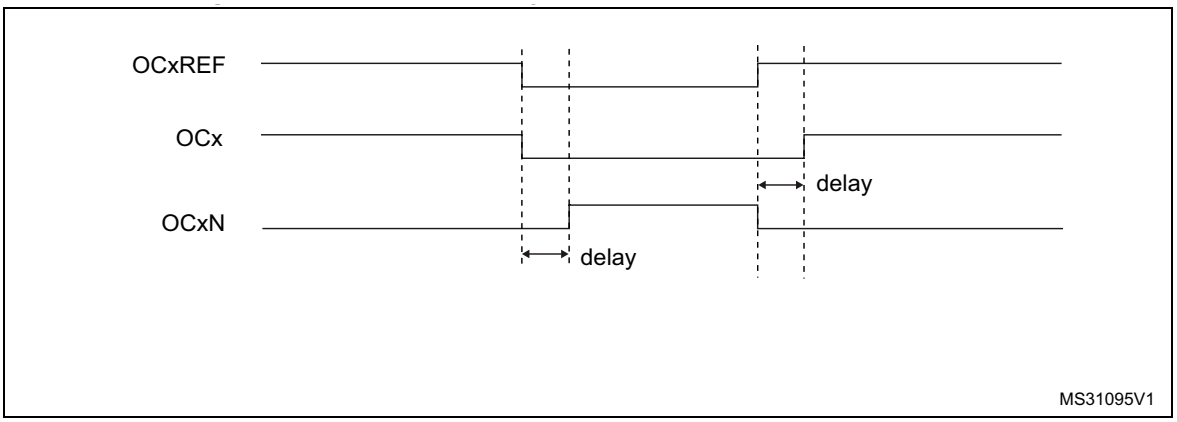
Задержка разрешается установкой обоих битов CCxE и CCxNE, и битом MOE при наличии схемы остановки. Биты DTG[7:0] регистра TIMx_BDTR определяют задержку для всех каналов. От сигнала OCxREF выдаются 2 выхода OCx и OCxN. Если OCx и OCxN активны высокими:

- Сигнал OCx повторяет опорный сигнал, но с задержкой переднего фронта.
- Сигнал OCxN это инверсный опорный сигнал, но с задержкой по противоположному фронту.

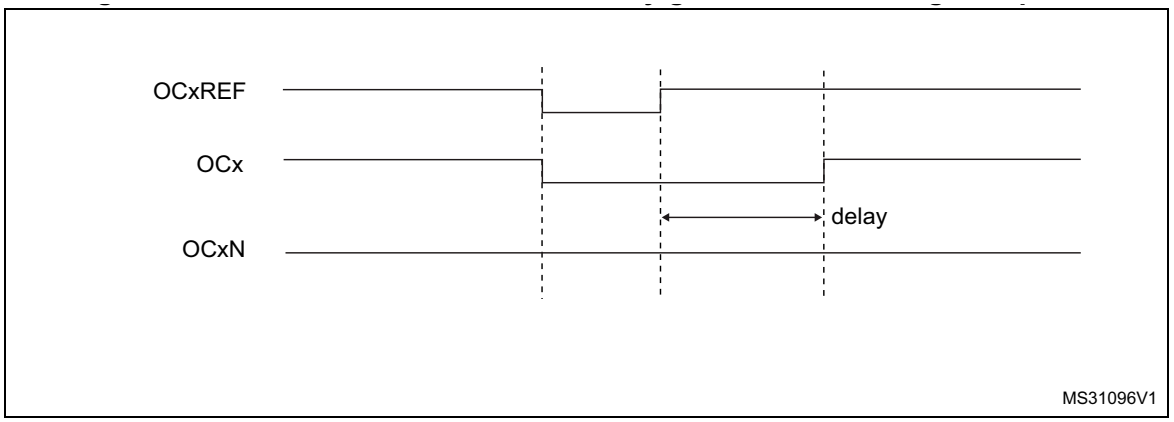
Если задержка больше ширины импульса активного выхода (OCx или OCxN), то соответствующий импульс не генерируется.

Далее показана связь генератора задержки выходных сигналов и опорным сигналом OCxREF. (В этих примерах CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1).

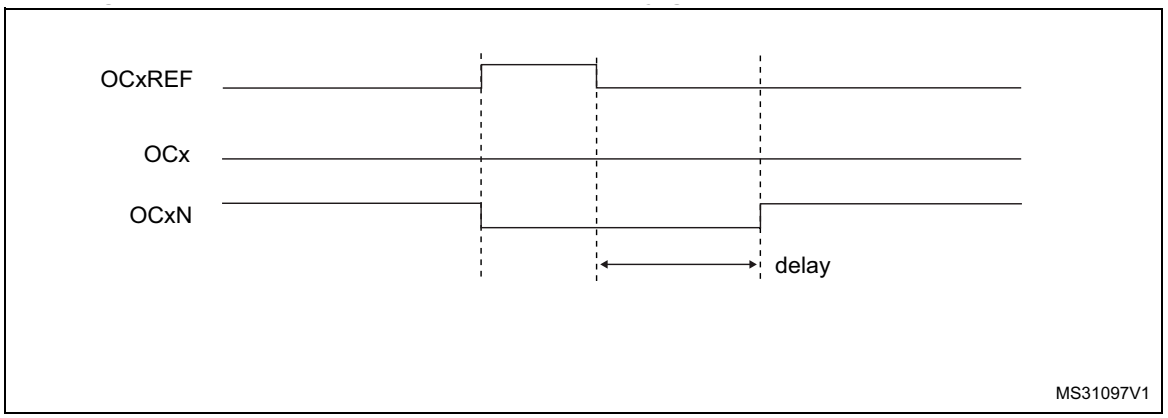
Инверсный выход с задержкой.



Задержка больше инверсного импульса.



Задержка больше прямого импульса.



Задержка определяется битами DTG в регистре TIMx_BDTR одинаковой для всех каналов.

Перенаправление OCxREF на OCx или OCxN

В режиме вывода (принудительном, сравнении выхода или ШИМ) OCxREF можно направлять на выходы OCx или OCxN битами CCxE и CCxNE регистра TIMx_CCER.

Так можно направить какой-либо сигнал (вроде ШИМ и пр.) на один из выходов, оставив инверсный неактивным. Другая возможность это сделать оба выхода неактивными или активными с инверсией и задержкой.

Когда разрешён только OCxN (CCxE=0, CCxNE=1), то он не инвертируется и становится активным при высоком OCxREF. Например, если CCxNP=0, то OCxN= OCxREF. С другой стороны, если разрешены оба сигнала OCx и OCxN (CCxE=CCxNE=1), то OCx становится активным при высоком OCxREF, а инверсный OCxN становится активным при низком OCxREF.

14.3.12.Функция останова

При останове разрешения вывода и неактивные уровни изменяются в соответствии с дополнительными битами управления (MOE, OSSI и OSSR в регистре TIMx_BDTR, OISx и OISxN в регистре TIMx_CR2). В любом случае, выходы OCx и OCxN не могут стать активными одновременно.

Сигнал останова может поступать с внешней ножки или по сбою тактирования от Системы безопасности тактов (CSS).

При выходе из сброса схема останова выключена и бит MOE сброшен. Включается схема останова установкой бита BKE в регистре TIMx_BDTR. Полярность сигнала сброса выбирается битом BKP там же. BKE и BKP можно изменять одновременно. Записанные BKE и BKP работают через 1 такт APB. Следовательно, их чтение истинно через 1 такт APB после записи.

Поскольку задний фронт MOE может быть асинхронным, то между действительным сигналом и синхронным битом в регистре TIMx_BDTR вставлена схема ресинхронизации. Так появляется задержка между синхронным и асинхронным сигналами. В частности, если MOE изменился с 0 на 1, то перед чтением должна быть выполнена пустая команда, поскольку запись асинхронна, а чтение синхронно.

При появлении сигнала останова:

- Бит MOE асинхронно очищается, переключая выходы в неактивное, пустое или сброшенное состояние (выбирается битом OSSI). Это работает даже при выключенном генераторе MCU.
- Все выходные каналы получают уровни, определённые битами OISx в регистре TIMx_CR2 сразу после сброса MOE. Если OSSI=0, то таймер освобождает включённые выходы, иначе они остаются высокими.
- Если используются инверсные выходы:
  - Сначала выходы переходят в неактивное состояние сброса (зависит от полярности). Это действие асинхронно и работает даже при выключенном тактировании.
  - Если тактирование ещё есть, то запускается генератор задержки перевода выходов на уровни, определённые битами OISx и OISxN. Даже в этом случае OCx и OCxN не могут стать активными одновременно. Из-за схемы ресинхронизации MOE, длительность задержки чуточку больше (около 2 ck_tim тактов).
  - Если OSSI=0, то таймер освобождает включённые выходы, иначе они остаются высокими если один из битов CCxE или CCxNE высокий.
- Ставится флаг сброса (BIFв регистре TIMx_SR). Если бит BIE в регистре TIMx_DIER стоит, то генерируется прерывание. Запрос DMA выдаётся если стоит бит BDEв регистр TIMx_DIER.
- Если бит AOE в TIMx_BDTR стоит, то бит MOE автоматически ставится снова по следующему событию UEV. Иначе MOE остаётся низким до следующей записи ‘1’.

NB: Входы останова действуют по уровню. Так что, MOE не ставится при активном входе (ни аппаратно, ни программно). Ну и флаг BIF не сбрасывается.

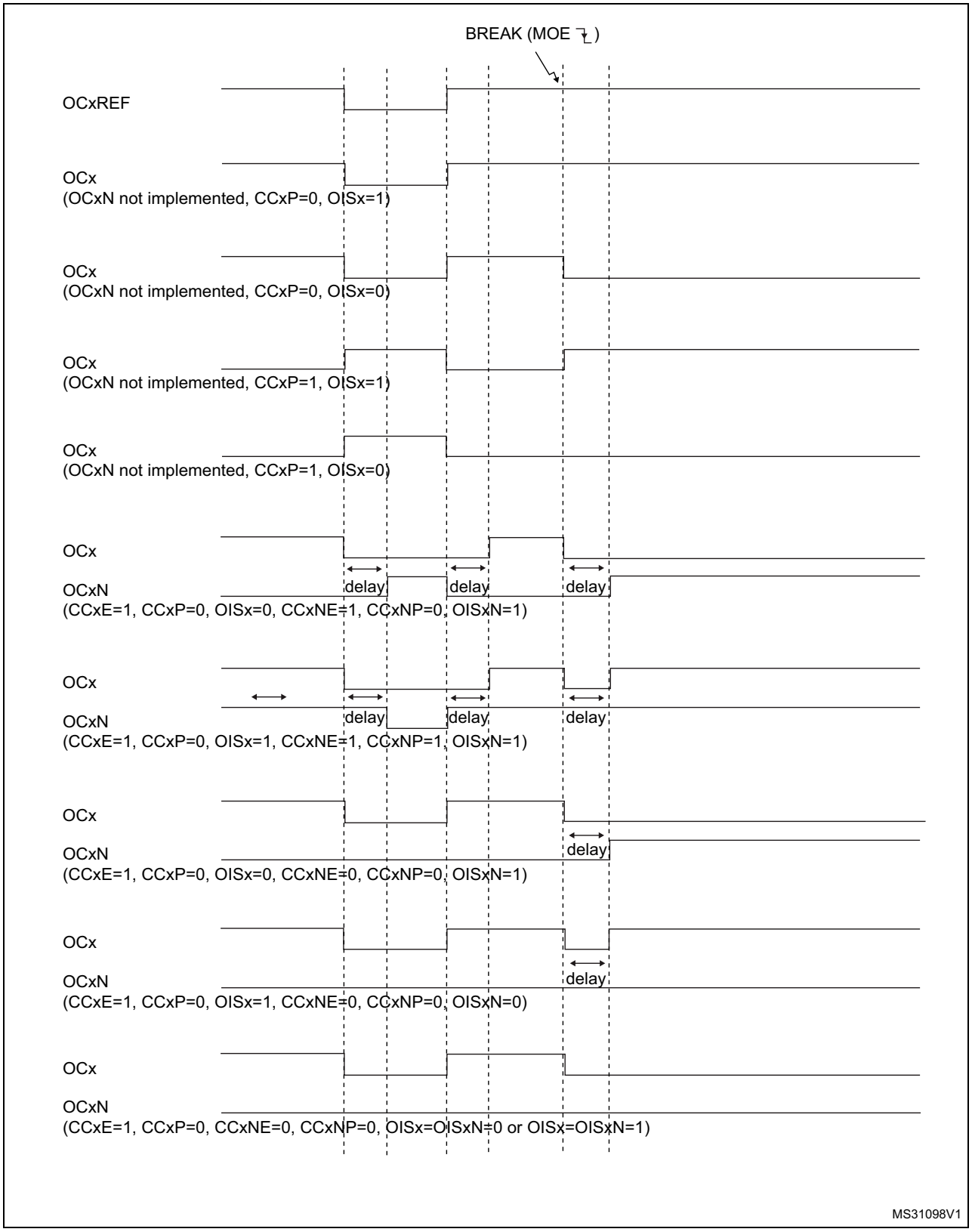
Останов можно генерировать по входу BRK с заданием полярности и битом разрешения BKE в регистре TIMx_BDTR.

Два способа генерации останова:

- По входу BRK с заданием полярности и битом разрешения BKE в регистре TIMx_BDTR.
- Программно битом BG в регистре TIMx_EGR.

Кроме того, в схеме останова реализована защита записи, позволяющая заморозить несколько параметров (длительность задержки, полярность и состояние выключенных OCx/OCxN, конфигурацию OCxM, разрешение и полярность останова). Есть три уровня защиты битами LOCK в регистре TIMx_BDTR. Биты LOCK можно писать единожды после сброса MCU.

Выходные сигналы при останове.



14.3.13.Очистка OCxREF по внешнему событию

Сигнал OCxREF для заданных каналов можно сделать низким, подав высокий уровень на вход ETRF (бит разрешения OCxCE в TIMx_CCMRx стоит в ‘1’). Он остаётся низким вплоть до следующего события UEV.

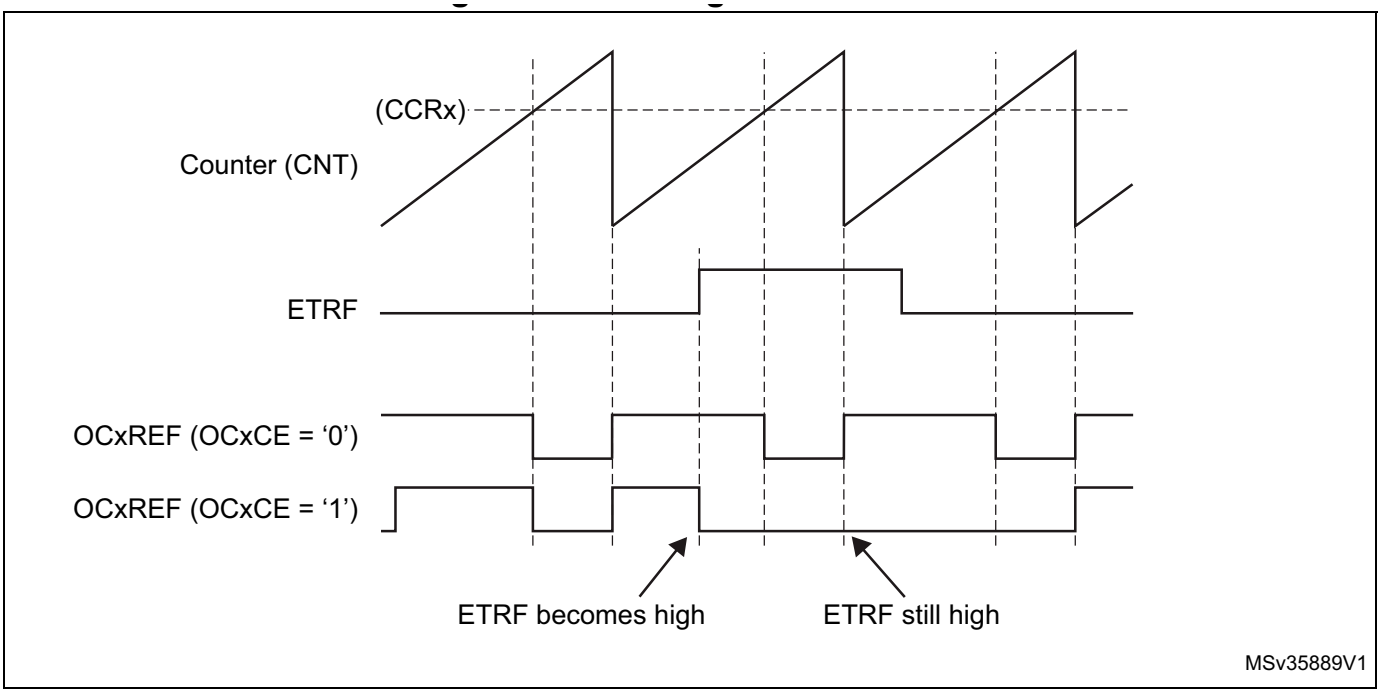
Эта функция работает в режимах сравнения выхода и ШИМ, но не в принудительном режиме.

Например, сигнал ETR можно подключить к к выходу компаратора для управления током. В этом случае ETR нужно конфигурировать так:

1. Выключить предделитель внешнего запуска: записать ’00’ в биты ETPS[1:0] в TIMx_SMCR.
2. Выключить режим 2 внешнего тактирования: снять бит ECE в TIMx_SMCR.
3. Поставить полярность и фильтр внешнего запуска (ETP и ETF) как нужно.

В примере ниже приведён сигнал OCxREF при переходе входа ETRF в высокий уровень для обоих значений OCxCE. Здесь TIMx в режиме ШИМ.

Очистка OCxREF.



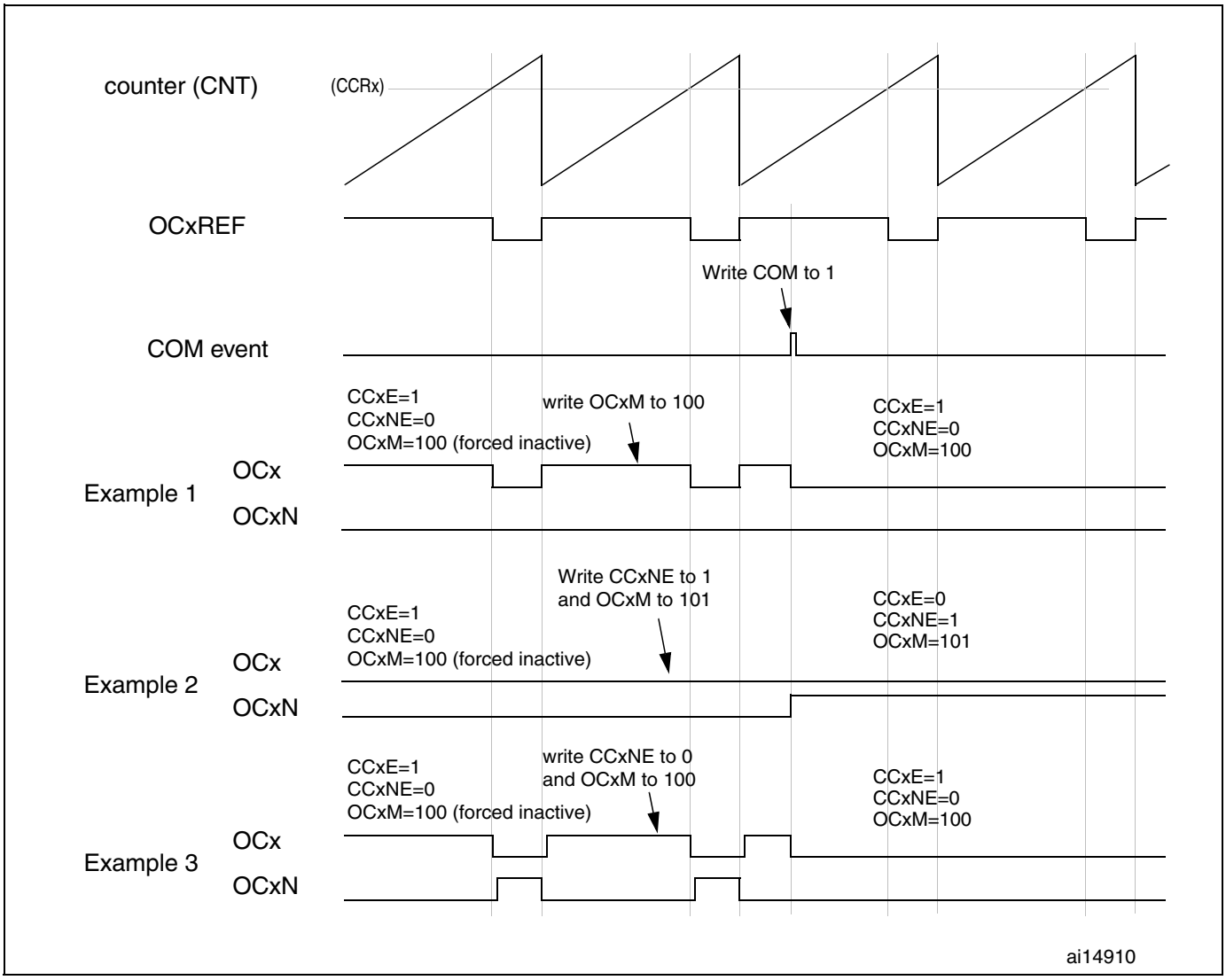
14.3.14.Шести-шаговая ШИМ

При работе с инверсными выходами доступна предзагрузка битов OCxM, CCxE и CCxNE. Они пишутся в теневые биты по событию коммутации COM. Так можно заранее создать конфигурацию и одновременно включить её для всех каналов. COM можно генерировать программно установкой бита COM в регистре TIMx_EGR или аппаратно (по переднему фронту TRGI).

При появлении события COM ставится флаг (COMIF в регистре TIMx_SR), который может вызвать прерывание (если стоит бит COMIE в регистре TIMx_DIER) или запрос DMA (если стоит бит COMDE в регистре TIMx_DIER).

Ниже приведены выходы OCx и OCxN при событии COM в 3 конфигурациях.

Пример COM, (OSSR=1).



14.3.15.Режим одного импульса

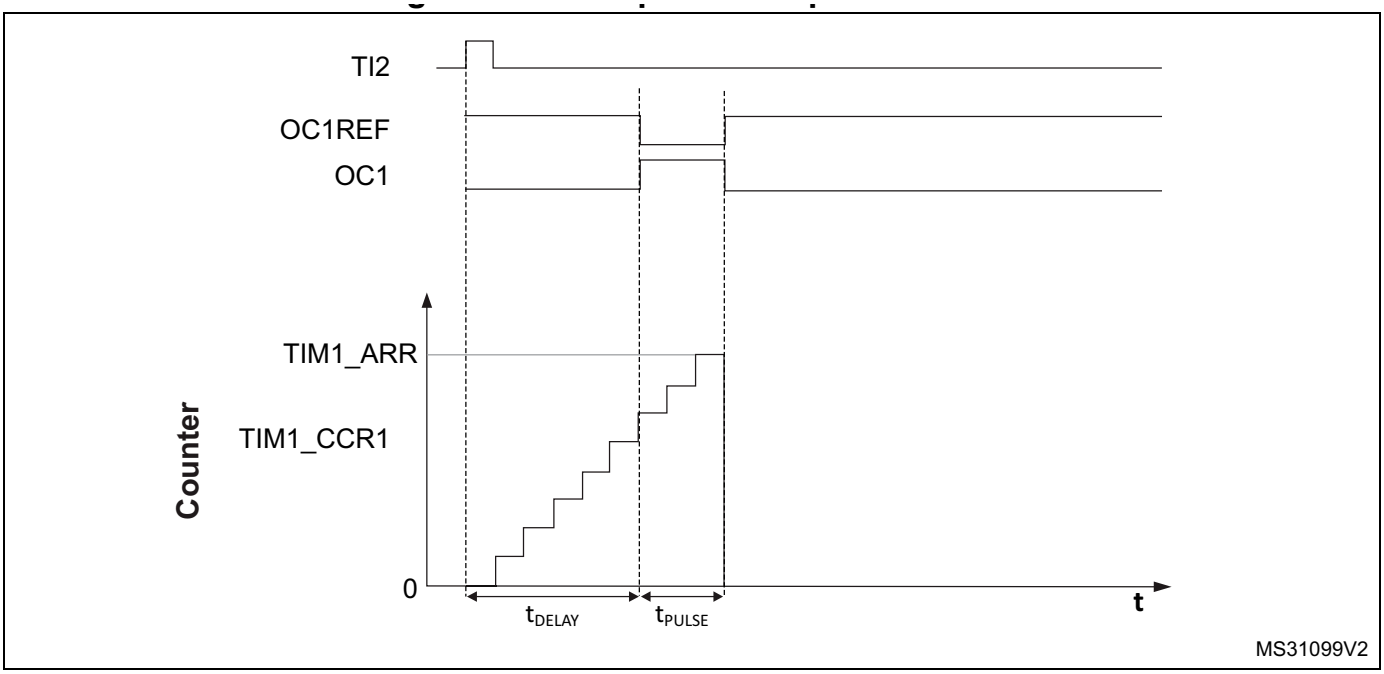
OPM это частный случай предыдущих режимов. Он позволяет счётчику запускаться по сигналу и генерировать импульс программируемой длины с программируемой задержкой.

Запускать счётчик можно из контроллера ведомого режима. Создавать сигнал можно в режиме сравнения выхода и ШИМ. Режим включается установкой бита `OPM` в регистре `TIMx_CR1`. Так счётчик автоматически останавливается по следующему событию `UEV`.

Импульс генерируется правильно только если значение сравнения отличается от начального значения счётчика. Перед стартом (таймер ждёт запуска) конфигурация должна быть:

- При прямом счёте: `CNT` < `CCRx` ≤ `ARR` (в частности, 0 < `CCRx`)
- При обратном счёте: `CNT` > `CCRx`

Режим одного импульса.



Например, выдадим положительный импульс на OC1 длиной t_{PULSE} после задержки t_{DELAY} по переднему фронту на ножке TI2.

Берём TI2FP2 как запуск 1:

- Поставим TI2FP2 на TI2 записью CC2S=’01’ в регистре TIMx_CCMR1.
- TI2FP2 ставим на передний фронт записью CC2P=’0’ в регистре TIMx_CCER.
- Выбираем TI2FP2 как запуск контроллером режима ведомого (TRGI) записью TS=’110’ в регистр TIMx_SMCR.
- TI2FP2 используем как старт записью SMS=‘110’ в TIMx_SMCR (режим запуска).

Сигнал OPM определяем записью регистров сравнения (учитываем частоту тактов и предделитель).

- Определяем t_{DELAY} записью в TIMx_CCR1.
- Определяем t_{PULSE} как разность значений перезагрузки и сравнения (TIMx_ARR - TIMx_CCR1).
- Сигнал создаётся переходом из ‘0’ в ‘1’ при сравнении и переходом из ‘1’ в ‘0’ при достижении счётчиком значения перезагрузки. Для этого, включаем режим ШИМ2 записью OC1M=111 в регистре TIMx_CCMR1. Регистры предзагрузки можно включить записью OC1PE=’1’ в регистре TIMx_CCMR1и ARPE в регистре TIMx_CR1, а можно не включать. В этом случае значение сравнения пишется в регистр TIMx_CCR1, значение перезагрузки в регистр TIMx_ARR, даём событие обновления битом UG и ждём внешнего запуска на TI2. В CC1P здесь пишется ‘0’.

В этом примере биты DIR и CMS регистра TIMx_CR1 должны быть сброшены.

Мы ждём только одного импульса (Однократный режим), так что для остановки счётчика по следующему событию обновления (счётчик переходит через значение перезагрузки и обнуляется) нужно поставить бит OPM регистра TIMx_CR1. Если бит OPM регистра TIMx_CR1 обнулён, то включается Повторяющийся режим.

Частный случай: быстрое включение OCx:

В этом режиме появление фронта на входе TIx ставит бит CEN, чем включает счётчик. Затем совпадение счётчика с значением сравнения переключает выходной сигнал. Но для этого нужны несколько тактов, ограничивая минимально возможное значение t_{DELAY}.

Если нужен сигнал с минимальной задержкой, то нужно ставить бит OCxFE в TIMx_CCMRx. Далее включаются OCxREF (и OCx) не глядя на сравнение. Его новый уровень равен уровню при сравнении. OCxFE работает только в режимах ШИМ1 и ШИМ2.

14.3.16.Режим интерфейса кодера

Режим интерфейса кодера выбирается записью SMS='001' в TIMx_SMCR при счёте по фронтам TI2, SMS='010' по фронтам TI1 и SMS='011' по фронтам TI1 и TI2.

Полярность TI1 и TI2 выбирается битами CC1P и CC2P в регистре TIMx_CCER. Если нужно, то можно использовать входной фильтр.

Для работы с инкрементным кодером используются входы TI1 и TI2. Счётчик тактируется каждой достоверной передачей по TI1FP1 или TI2FP2 (TI1 и TI2 после входного фильтра и выбора полярности, без них TI1FP1= TI1 и TI2FP2= TI2) при стоящем бите CEN в TIMx_CR1. Последовательность сигналов с двух входов создаёт импульсы счёта и сигнал направления. Счёт может быть прямым и обратным, бит DIR в TIMx_CR1 изменяется аппаратно. Бит DIR вычисляется на каждый входной сигнал (TI1 и/или TI2).

Режим интерфейса кодера просто работает по внешним тактам с выбором направления. То есть счётчик непрерывно считает туда-сюда между 0 и перезагружаемым значением в регистре TIMx_ARR. Так что TIMx_ARR нужно писать до старта. Таким способом захват, сравнение, предделитель и запуск вывода продолжают нормально. Режим кодера и Режим внешних тактов 2 несовместимы.

В этом режиме счётчик автоматически следует за скоростью и направлением инкрементного кодера и его содержимое всегда представляет позицию кодера. Направление счёта соответствует направлению вращения подключённого датчика.

Далее приведены возможные комбинации, полагая, что TI1 и TI2 не изменяются одновременно.

Направления счёта.

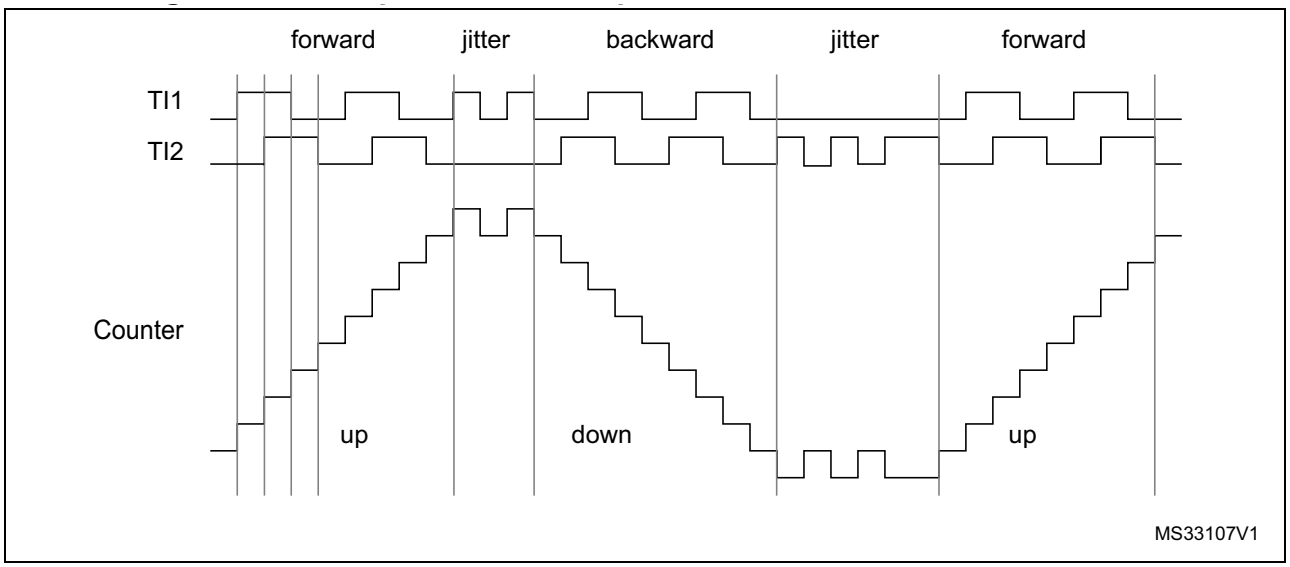
Активный сигнал	Уровень оппозитного сигнала (TI1FP1 для TI2, TI2FP2 для TI1)	Фронт TI1FP1		Фронт TI2FP2	
		Передний	Задний	Передний	Задний
Только TI1	Высокий	Обратный	Прямой	Стоит	Стоит
	Низкий	Прямой	Обратный	Стоит	Стоит
Только TI2	Высокий	Стоит	Стоит	Прямой	Обратный
	Низкий	Стоит	Стоит	Обратный	Прямой
TI1 и TI2	Высокий	Обратный	Прямой	Прямой	Обратный
	Низкий	Прямой	Обратный	Обратный	Прямой

Внешний инкрементный кодер может напрямую подключаться к MCU без дополнительной логики. Но обычно для преобразования дифференциального сигнала в цифровой используются компараторы. Это сильно увеличивает помехозащищённость. Третий выход кодера, показывающий механическую нулевую позицию, можно подключить к входу внешнего прерывания и сбросу счётчика.

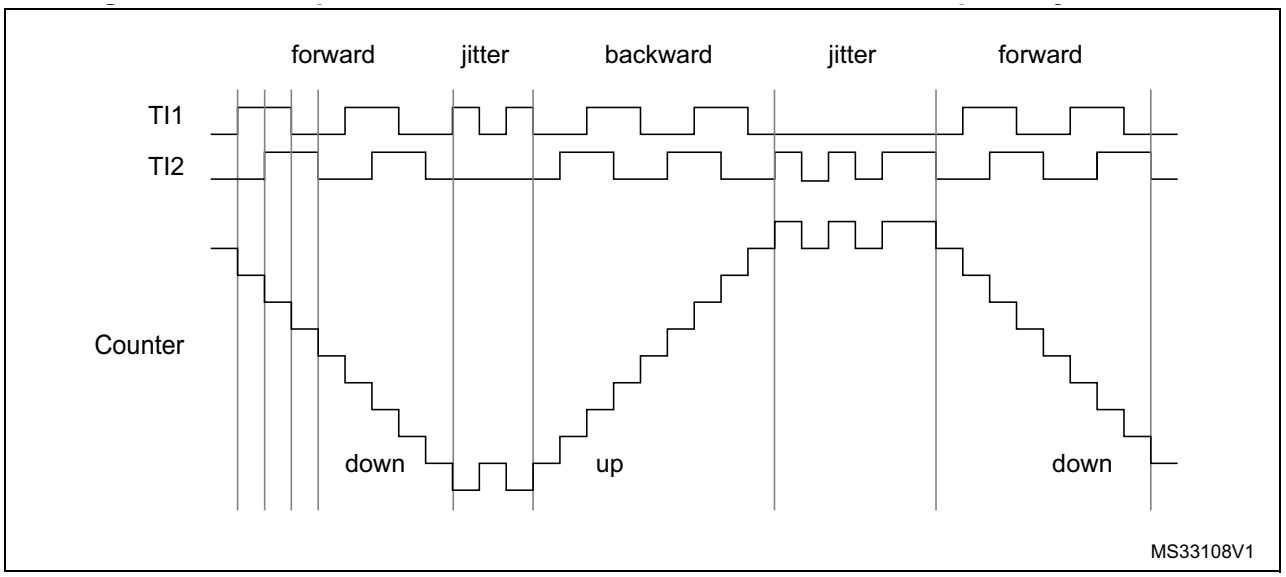
Пример работы счётчика показывает генерацию сигнала счёта и направления. Также показано как компенсируется дребезг на обоих выбранных фронтах. Он может появиться при нахождении датчика возле точек переключения. Использована такая конфигурация:

- CC1S='01' (TIMx_CCMR1, TI1FP1 на TI1).
- CC2S='01' (TIMx_CCMR2, TI1FP2 на TI2).
- CC1P='0', and IC1F = '0000' (TIMx_CCER, TI1FP1 без инверсии, TI1FP1=TI1).
- CC2P='0', and IC2F = '0000' (TIMx_CCER, TI1FP2 без инверсии, TI1FP2= TI2).
- SMS='011' (TIMx_SMCR, оба входа по обоим фронтам).
- CEN='1' (TIMx_CR1, счётчик включён).

Работа счётчика.



Работа счётчика с инверсным TI1FP1 и CC1P='1'.



Таймер в режиме интерфейса кодера показывает текущую позицию счётчика. С помощью второго таймера в режиме захвата, измеряя время между двумя событиями кодера, можно получить динамическую информацию (скорость, ускорение, торможение). Для этого можно использовать третий выход кодера, показывающего механический ноль. В зависимости от времени между двумя событиями счётчик можно читать и регулярно. Это делается регулярной записью счётчика в третий входной регистр (если есть) периодическим сигналом от другого таймера. Можно читать и запросом DMA от часов реального времени.

14.3.17.Функция XOR входов таймера

Бит `TI1S` в регистре `TIMx_CR2` позволяет подключать к входу канала 1 выход вентиля XOR, сочетая три входа `TIMx_CH1`, `TIMx_CH2` and `TIMx_CH3`.

Выход XOR можно использовать со всеми входными функциями таймера, вроде запуска или захвата.

14.3.18.Работа с датчиками Холла

Это выполняется генерацией ШИМ таймером TIM1 или TIM8 и другим таймером TIMx (TIM2, TIM3, TIM4 или TIM5), так сказать "интерфейсным таймером". TIMx захватывает 3 соединённые через XOR (бит `TI1S` в `TIMx_CR2`) входные ножки (`TIMx_CH1`, `TIMx_CH2`, and `TIMx_CH3`) на вход `TI1`.

Контроллер режима ведомого ставится в сброс; вход ведомого TI1F_ED. Так что при переключении одного из 3 входов счётчик начинает с 0. Этим создаётся временная база, запущенная любым изменением на входах Холла.

На “интерфейсном таймере” канал 1 ставится на захват, сигнал захвата TRC. Захваченное значение, соответствующее времени между 2 изменениями входных сигналов, показывает скорость мотора.

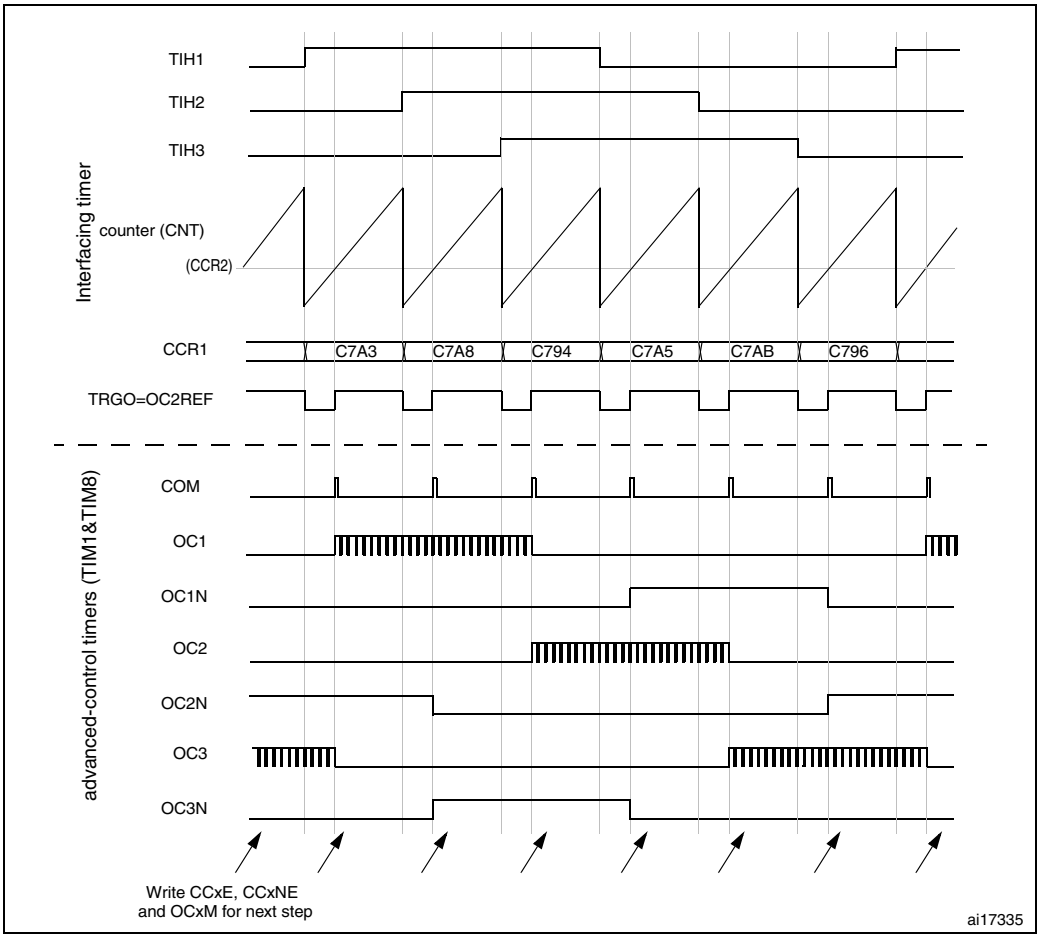
“Интерфейсный таймер” может в режиме выхода выдавать импульс изменения конфигурации продвинутого таймера (TIM1 или TIM8) (событием COM). TIM1 выдаёт ШИМ сигнал для управления мотором. Для этого канал интерфейсного таймера после программируемой задержки должен выдавать положительный импульс (в режиме сравнения выхода или ШИМ). Этот импульс через выход TRGO подаётся на продвинутый таймер (TIM1 или TIM8).

Пример: надо изменить ШИМ конфигурацию TIM1 с программируемой задержкой после каждого изменения входов Холла на таймере TIMx.

- Подключить через XOR три 3 входа таймера на TI1 установкой бита TI1S в TIMx_CR2,
- Создать временную базу: записать в TIMx_ARR максимальное значение (счётчик должен сбрасываться по изменению TI1). Установить предделитель так, чтобы максимальный период счёта превышал время между 2 изменениями датчиков,
- Поставить канал 1 в режим захвата (выбран TRC): записать ‘11’ в биты CC1S регистра TIMx_CCMR1. Можно подключить и цифровой фильтр,
- Поставить канал 1 в режим ШИМ 2 с желаемой задержкой: записать '111’ в биты OC2M и ‘00’ в биты CC2S регистра TIMx_CCMR1,
- Для запуска по TRGO выбрать OC2REF: записать '101’ в биты MMS регистра TIMx_CR2.

В таймере TIM1 правый вход ITR должен стать входом запуска, таймер должен генерировать ШИМ, управляющие сигналы захвата/сравнения предзагружаемые (CCPC=1 в регистре TIMx_CR2) м событие COM управляется от входа запуска (CCUS=1 в регистре TIMx_CR2). Биты управления ШИМ (CCxE, OCxM) пишутся после события COM для следующего шага (это можно сделать в обработчике прерывания по переднему фронту OC2REF).

Пример интерфейса датчиков Холла.



14.3.19.TIMx и синхронизация внешнего запуска

Есть три режима синхронизации TIMx: Сброс, Вентильный и Запуск.

Режим ведомого: Сброс

Счётчик и предделитель можно инициализировать по входу запуска. Кроме того, если бит URS в TIMx_CR1 сброшен, то генерируется событие UEV. Далее обновляются все предзагружаемые регистры (TIMx_ARR, TIMx_CCRx).

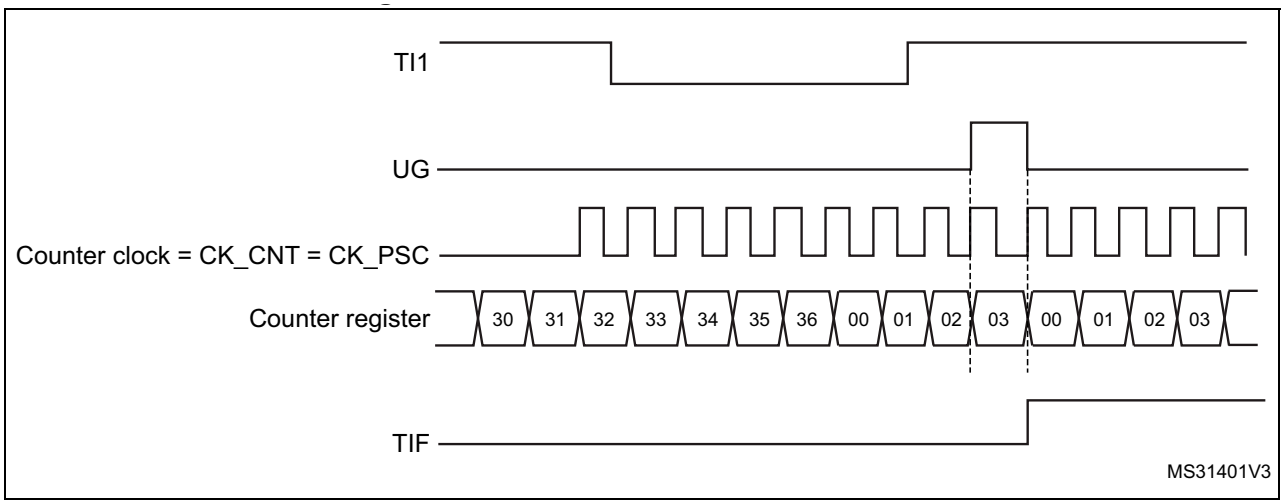
В этом примере счётчик прямого счёта сбрасывается по переднему фронту на входе TI1:

- Ставим канал 1 на передний фронт TI1. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC1S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC1P=0 в регистре TIMx_CCER определяет полярность и передний фронт.
- Ставим таймер в режим сброса записью SMS=100 в регистр TIMx_SMCR. Выбираем вход с TI1 записью TS=101 в регистре TIMx_SMCR.
- Запускаем счётчик записью CEN=1 в регистре TIMx_CR1.

Счётчик начинает считать внутренние такты вплоть переднего фронта TI1. Тогда счётчик сбрасывается с 0, ставится флаг запуска (бит TIF в регистре TIMx_SR) выдаются запросы прерывания и DMA (если разрешены битами TIE и TDE в регистре TIMx_DIER).

На рисунке ниже регистр перезагрузки TIMx_ARR=0x36. Задержка между фронтом TI1 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI1.

Режим Сброса.



Режим ведомого: Вентильный

Счётчик включается уровнем на выбранном входе.

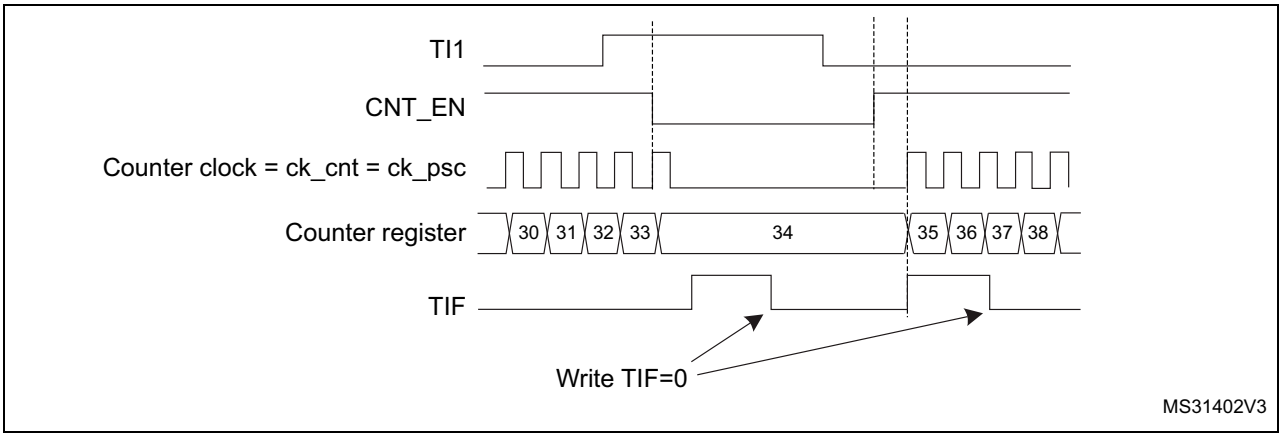
В этом примере счётчик прямого счёта работает только при низком уровне TI1:

- Ставим канал 1 на низкий уровень TI1. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC1S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC1P=1 в регистре TIMx_CCER определяет полярность и низкий уровень.
- Ставим таймер в вентильный режим записью SMS=101 в регистр TIMx_SMCR. Выбираем вход с TI1 записью TS=101 в регистре TIMx_SMCR.
- Запускаем счётчик записью CEN=1 в регистре TIMx_CR1. В этом режиме счётчик не работает при CEN=0, независимо от уровня на входе.

Счётчик начинает считать внутренние такты при низком уровне на TI1 и останавливается при высоком. Флаг запуска (бит TIF в регистре TIMx_SR) ставится при пуске и остановке таймера.

Задержка между фронтом TI1 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI1.

Вентильный режим.



Режим ведомого: Запуск

Счётчик включается событием на выбранном входе.

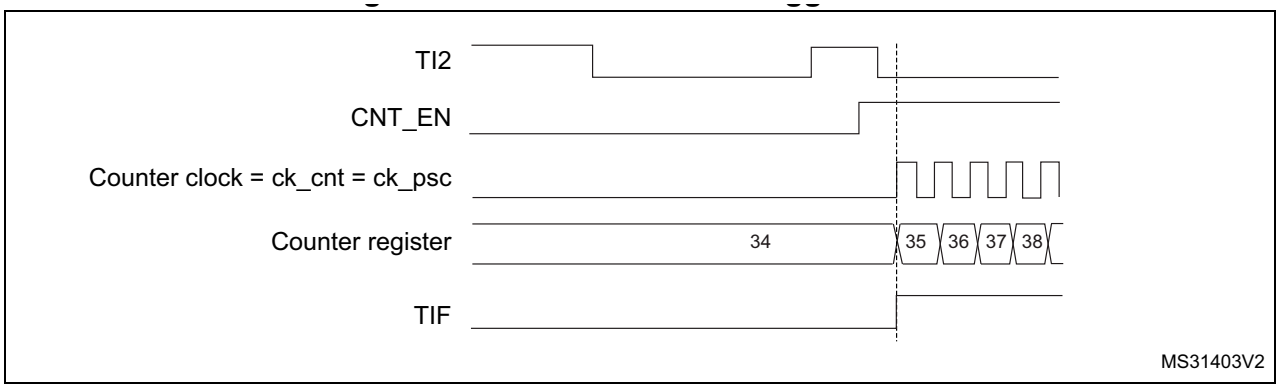
В этом примере счётчик прямого счёта работает по переднему фронту на TI2:

- Ставим канал 2 на передний фронт TI2. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC2S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC2P=1 в регистре TIMx_CCER определяет полярность и низкий уровень.
- Ставим таймер в режим запуска записью SMS=110 в регистр TIMx_SMCR. Выбираем вход с TI2 записью TS=110 в регистре TIMx_SMCR.

Счётчик начинает считать внутренние такты по переднему фронту на TI2 и ставит флаг запуска (бит TIF в регистре TIMx_SR).

Задержка между фронтом TI2 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI2.

Режим Запуска.



Режим ведомого: Внешние такты 2 + Запуск

К режимам ведомого (кроме Внешних тактов 1 и Кодера) можно добавить режим внешнего тактирования 2. В этом случае сигнал ETR выбирается для внешних тактов, а по другому входу можно подавать запуск (в режимах Сброса, Вентильном или Запуска). Не рекомендуем выбирать ETR как TRGI битами TS в регистре TIMx_SMCR.

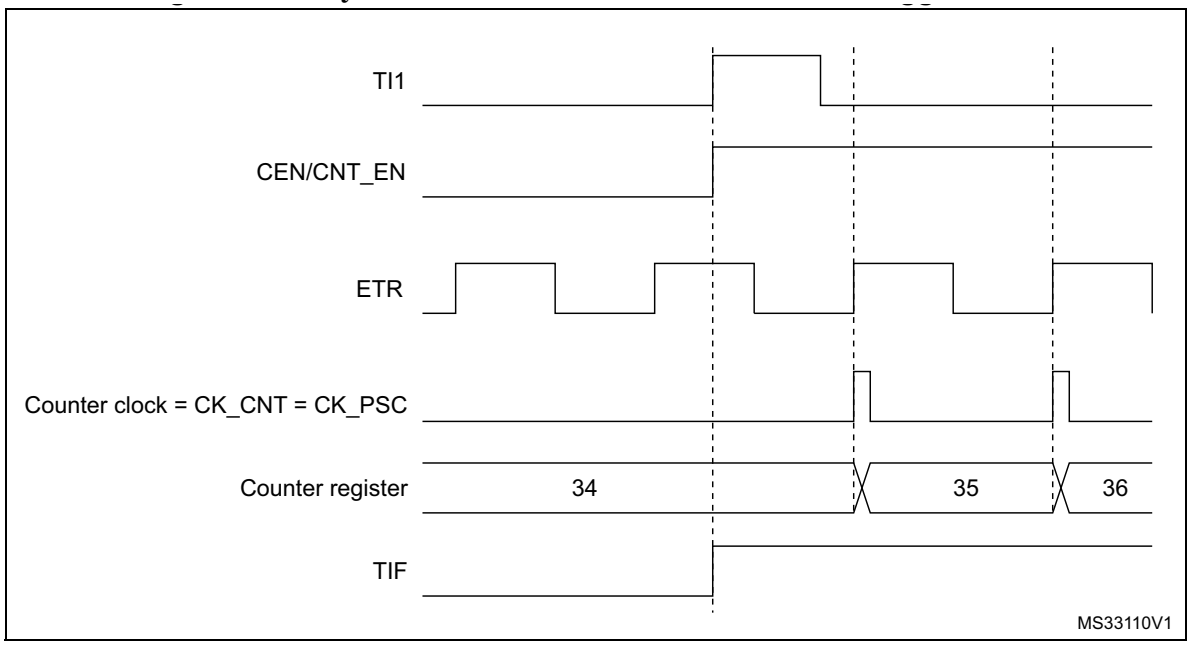
В этом примере счётчик прямого счёта работает по переднему фронту на ETR после переднего фронта запуска на TI1:

1. Ставим внешний запуск в регистре TIMx_SMCR:
   - ETF = 0000: фильтра нет; ETPS = 00: предделитель выключен; ETP = 0: передний фронт на ETR и ECE=1 для внешних тактов 2.
2. Ставим канал 1 на передний фронт TI:
   - IC1F=0000: фильтра нет; предделитель выключен, ничего не делаем; CC1S=01 в TIMx_CCMR1 для захвата входа; CC1P=0 в TIMx_CCER для полярности (и только переднего фронта).
3. Ставим таймер в режим запуска записью SMS=110 в регистр TIMx_SMCR. Выбираем вход с TI1 записью TS=101 в регистре TIMx_SMCR.

Передний фронт TI1 разрешает счёт и ставит флаг TIF. Считаются передние фронты ETR.

Задержка между фронтом ETR и действительным сбросом появляется из-за схемы ресинхронизации на входе ETRP.

Внешние такты 2 + Запуск.



14.3.20.Синхронизация таймера

Таймеры TIM связаны между собой для синхронизации или сцепления.

NB: Такты ведомого таймера нужно включать до получения событий от ведущего и не должны меняться налету.

14.3.21.Режим отладки

В режиме отладки (ядро Cortex-M3 остановлено), счётчик TIMx либо работает, либо останавливается, в зависимости от бита DBG_TIMx_STOP в модуле DBG.

При стоящем счётчике (DBG_TIMx_STOP = 1 в регистре DBGMCU_APBx_FZ) выходы отключаются (как при сброшенном бите MOE). Выходы можно перевести в неактивное состояние (бит OSSI = 1), или возложить управление на контроллер GPIO (бит OSSI = 0) для перевода их в высокоимпедансное состояние (Hi-Z).

14.4. Регистры TIM1 и TIM8

Они доступны полусловами и словами.

14.4.1. Регистр управления 1 (TIMx_CR1)

Смещение адреса: 0x00

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:10 Резерв, не трогать.

— Биты 9:8 **CKD[1:0]:** Деление тактов.

Это отношение тактов таймера (CK_INT) и тактов задержки и выборки (t_{DTS}) в генераторах задержки и цифровых фильтрах (ETR, TIx),

00: $t_{DTS}=t_{CK\_INT}$

01: $t_{DTS}=2*t_{CK\_INT}$

10: $t_{DTS}=4*t_{CK\_INT}$

11: Резерв

— Бит 7 **ARPE**: Разрешение буфера перезагрузки TIMx_ARR

0: Нельзя

1: Нужно

— Бит 6:5 **CMS[1:0]**: Выбор режима по центру

00: По фронту. Прямой или обратный счёт в зависимости от бита направления (DIR).

01: По центру 1. Попеременный прямой и обратный счёт. Флаг сравнения выхода выводных каналов (CCxS=00 в TIMx_CCMRx) ставится только при обратном счёте.

10: По центру 2. Попеременный прямой и обратный счёт. Флаг сравнения выхода выводных каналов (CCxS=00 в TIMx_CCMRx) ставится только при прямом счёте.

11: По центру 3. Попеременный прямой и обратный счёт. Флаг сравнения выхода выводных каналов (CCxS=00 в TIMx_CCMRx) ставится и при прямом и при обратном счёте.

NB: При включённом таймере (CEN=1) переключать режим по фронту в режим по центру нельзя.

— Бит 4 **DIR**: Направление счёта.

0: Прямой

1: Обратный

NB: Бит читается только в режимах По центру и Кодера.

— Бит 3 **OPM**: Режим одного импульса

0: Счётчик не останавливается

1: Счётчик останавливается по следующему событию обновления (снимается бит CEN)

— Бит 2 **URS**: Источник запроса по событию UEV.

Изменяется программно.

0: Разрешённые запросы прерывания или DMA выдаются по:

– Переполнение/Исчерпание счётчика

– Установка бита UG

– Обновление от контроллера режима ведомого

1: Разрешённые запросы прерывания или DMA выдаются только по Переполнению/Исчерпанию счётчика.

— Бит 1 **UDIS:** Выключение выдачи события обновления UEV.

Изменяется программно.

0: Событие UEV выдаётся по:

– Переполнение/Исчерпание счётчика

– Установка бита UG

– Обновление от контроллера режима ведомого

1: Событие UEV не выдаётся, скрытые регистры (ARR, PSC, CCRx) хранят значение. При стоящем бите UG или аппаратном сбросе от контроллера режима ведомого счётчик и предделитель инициализируются.

— Бит 0 **CEN:** Включение счётчика.

0: Выключен

1: Включён

NB: Режимы Внешнего тактирования, Вентильный и Кодера работают только при заранее установленном бите CEN. Режим Запуска ставит его автоматически.

14.4.2. Регистр управления 2 (TIMx_CR2)

Смещение адреса: 0x04

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

— Бит 15 Резерв, не трогать.

— Бит 14 **OIS4:** Пустое состояние 4 (выход OC4) см. бит OIS1.

— Бит 13 **OIS3N**: Пустое состояние 3 (выход OC3N) см. бит OIS1.

— Бит 12 **OIS3**: Пустое состояние 3 (выход OC3) см. бит OIS1.

— Бит 11 **OIS2N**: Пустое состояние 2 (выход OC2N) см. бит OIS1.

— Бит 10 **OIS2**: Пустое состояние 2 (выход OC2) см. бит OIS1.

— Бит 9 **OIS1N**: Пустое состояние 1 (выход OC1N).
0: OC1N=0 после задержки при MOE=0
1: OC1N=1 после задержки при MOE=0
NB: Нельзя изменять при уровнях LOCK 1, 2 или 3 в регистре TIMx_BDTR.

— Бит 8 **OIS1**: Пустое состояние 1 (выход OC1).
0: OC1=0 после задержки (если есть OC1N) при MOE=0
1: OC1=1 после задержки (если есть OC1N) при MOE=0
NB: Нельзя изменять при уровнях LOCK 1, 2 или 3 в регистре TIMx_BDTR.

— Бит 7 **TI1S**: Выбор TI1.
0: Ножка TIMx_CH1 на входе TI1
1: Ножки TIMx_CH1, CH2 и CH3 на входе TI1 (через XOR)

— Бит 6:4 **MMS[1:0]**: Выбор режима ведущего
Выбор информации синхронизации ведомых таймеров от ведущего (TRGO):
000: **Сброс** - На выход запуска (TRGO) направлен бит UG регистра TIMx_EGR. Если сброс задан входом запуска (контроллер ведомого сброшен), то сигнал TRGO задержан по отношению к реальному сбросу.
001: **Разрешение** - На выход запуска (TRGO) направлен сигнал разрешения CNT_EN. Это полезно при одновременном старте нескольких таймеров или при управлении окном работы ведомого таймера. CNT_EN это объединение по OR бита CEN и входа запуска в Вентильном режиме. При подаче CNT_EN по входу запуска сигнал TRGO задерживается, кроме режима ведущий/ведомый (см. бит MSM в регистре TIMx_SMCR).
010: **Обновление** - На выход запуска (TRGO) направлено событие обновления. Например, ведущий таймер может быть предделителем ведомого.
011: **Импульс сравнения** - После захвата или совпадения выход запуска (TRGO) посылает положительный импульс установки флага CC1IF (даже если он стоит).
100: **Сравнение** - На выход запуска (TRGO) направлен сигнал OC1REF
101: **Сравнение** - На выход запуска (TRGO) направлен сигнал OC2REF
110: **Сравнение** - На выход запуска (TRGO) направлен сигнал OC3REF
111: **Сравнение** - На выход запуска (TRGO) направлен сигнал OC4REF
NB: Тактирование ведомого таймера и ADC нужно включать до получения событий от ведущего таймера и не должно меняться налету.

— Бит 3 **CCDS**: Выбор DMA Захвата/Сравнения
0: Запрос CCx DMA посылается по событию CCx
1: Запросы CCx DMA посылаются по событию обновления

— Бит 2 **CCUS**: Выбор обновления битов Захвата/Обновления
0: Предзагружаемые биты (CCPC=1) обновляются только установкой бита COMG
1: Предзагружаемые биты (CCPC=1) обновляются установкой бита COMG или передним фронтом TRGI
NB: Работает только при наличии инверсных выходов.

— Бит 1 Резерв, не трогать.

— Бит 0 **CCPS:** Предзагрузка битов Захвата/Сравнения.
0: Биты CCxE, CCxNE и OCxM не предзагружены
1: Биты CCxE, CCxNE и OCxM предзагружены, после записи они обновляются только по событию COM (бит COMG ставится по переднему фронту на TRGI, в зависимости от бита CCUS).
NB: Работает только при наличии инверсных выходов.

14.4.3. Регистр управления режима ведомого (TIMx_SMCR)

Смещение адреса: 0x08

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	Res.	rw	rw	rw

— Бит 15 **ETP**: Полярность внешнего запуска
0: ETR не инверсный, активен высоким или передним фронтом.
1: ETR инверсный, активен низким или задним фронтом.

— Бит 14 **ECE**: Разрешение внешнего тактирования

0: Внешнее тактирование 2 выключено

1: Внешнее тактирование 2 включено. Счётчик работает от активного фронта ETRF.

NB:

1: Установка бита ECE равнозначна Внешнему тактированию 1 с подключением TRGI к ETRF (SMS=111 и TS=111).

2: Внешнее тактирование 2 можно использовать одновременно с режимами ведомого: Сброс, Вентильный и Запуска. Тем не менее, TRGI нельзя подключать к ETRF (TS не равен 111).

3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

— Биты 13:12 **ETPS[1:0]**: Предделитель внешнего запуска

Частота сигнала ETRP должна быть не больше 1/4 частоты TIMxCLK. Полезно при измерении быстрых внешних сигналов.

00: Предделитель Выкл.

01: ETRP / 2

10: ETRP / 4

11: ETRP / 8

— Биты 11:8 **ETF[3:0]**: Фильтр внешнего запуска

Определяет частоту выборки и длину цифрового фильтра сигнала ETRP. Цифровой фильтр это счётчик, выдающий сигнал на выход после N последовательных событий на входе:

0000: Без фильтра, выборка на частоте f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

— Бит 7 **MSM**: Режим Ведущий/Ведомый

0: Ничего

1: Действие входа запуска (TRGI) задерживается для полной синхронизации текущего таймера и его ведомых (через TRGO). Это полезно при синхронизации нескольких таймеров от одного внешнего события.

— Биты 6:4 **TS[2:0]**: Выбор запуска

000: Внутренний 0 (ITR0)

001: Внутренний 1 (ITR1)

010: Внутренний 2 (ITR2)

011: Внутренний 3 (ITR3)

100: Детектор фронта TI1 (TI1F_ED)

101: Фильтрованный вход 1 (TI1FP1)

110: Фильтрованный вход 2 (TI2FP2)

111: Внешний запуск (ETRF)

NB: Менять их можно только если не используются (например, при SMS=000) во избежание неверного определения фронта.

— Бит 3 Резерв, не трогать.

— Биты 2:0 **SMS[2:0]**: Выбор режима ведомого

При внешнем сигнале активный фронт TRGI подключается к внешнему входу с выбранной полярностью.

000: Режим ведомого выключен - если CEN = '1', то предделитель работает напрямую от внутренних тактов.

001: Режим кодера 1 - Прямой/обратный счёт по фронту TI2FP1 в зависимости от уровня TI1FP2.

010: Режим кодера 2 - Прямой/обратный счёт по фронту TI1FP2 в зависимости от уровня TI2FP1.

011: Режим кодера 3 - Прямой/обратный счёт по фронтам TI1FP1 и TI2FP2 в зависимости от уровня другого входа.

100: Режим сброса - Передний фронт TRGI инициализирует счётчик и запускает обновление регистров.

101: Вентильный режим - Такты счётчика разрешены при высоком TRGI. При низком TRGI счётчик останавливается, но не сбрасывается. Старт и стоп счётчика управляемые.

110: Режим запуска - Счётчик стартует по переднему фронту TRGI (но не сбрасывает). Управляемый только старт.

111: Внешние такты 1 - Счёт передних фронтов TRGI.

NB: Вентильный режим нельзя использовать если входом запуска выбран TI1F_ED (TS='100'). В действительности, TI1F_ED выдаёт 1 импульс на каждую передачу TI1F, тогда как вентильный режим проверяет уровень сигнала запуска.

Такты ведомого таймера нужно включать до приёма событий от ведущего таймера и не должны изменяться налету.

Таблица 82. Подключение внутреннего запуска TIMx.

Ведомый TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM1	TIM5_TRGO	TIM2_TRGO	TIM3_TRGO	TIM4_TRGO
TIM8	TIM1_TRGO	TIM2_TRGO	TIM4_TRGO	TIM5_TRGO

14.4.4. Регистр разрешения прерываний/DMA (TIMx_DIER)

Смещение адреса: 0x0C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Содержимое битов:

0: Нельзя

1: Можно

— Бит 15 Резерв, не трогать.

— Бит 14 **TDE**: Запрос DMA запуска

— Бит 13 **COMDE**: Запрос DMA события COM

— Бит 12 **CC4DE**: Запрос DMA захвата/сравнения 4

— Бит 11 **CC3DE**: Запрос DMA захвата/сравнения 3

— Бит 10 **CC2DE**: Запрос DMA захвата/сравнения 2

— Бит 9 **CC1DE**: Запрос DMA захвата/сравнения 1

— Бит 8 **UDE**: Запрос DMA обновления

— Бит 7 **BIE**: Запрос прерывания останова

— Бит 6 **TIE**: Запрос прерывания запуска

— Бит 5 **COMIE**: Запрос прерывания события COM

— Бит 4 **CC4IE**: Запрос прерывания захвата/сравнения 4

— Бит 3 **CC3IE**: Запрос прерывания захвата/сравнения 3

— Бит 2 **CC2IE**: Запрос прерывания захвата/сравнения 2

— Бит 1 **CC1IE**: Запрос прерывания захвата/сравнения 1

— Бит 0 **UIE**: Запрос прерывания обновления

14.4.5. Регистр состояния (TIMx_SR)

Биты ставятся аппаратно, стираются записью 0.

Перезахват это появление события при стоящем его флаге.

Смещение адреса: 0x10

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0	Res.	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Содержимое битов:

0: Не было

1: Было

— Биты 15:13 Резерв, не трогать.

— Бит 12 **CC4OF**: Флаг перезахвата захвата/сравнения 4

— Бит 11 **CC3OF**: Флаг перезахвата захвата/сравнения 3

— Бит 10 **CC2OF**: Флаг перезахвата захвата/сравнения 2

— Бит 9 **CC1OF**: Флаг перезахвата захвата/сравнения 1

— Бит 8 Резерв, не трогать.

— Бит 7 **BIF**: Флаг прерывания останова

Стирается при неактивном сигнале останова.

— Бит 6 **TIF**: Флаг прерывания запуска

Ставится аппаратно по активному фронту на входе TRGI при работающем контроллере режима ведомого во всех режимах, кроме Вентильного, (тогда по обоим фронтам).

— Бит 5 **COMIF**: Флаг прерывания события COM

Ставится аппаратно по событию COM (биты - CCxE, CCxNE, OCxM - обновились).

— Бит 4 **CC4IF**: Флаг прерывания захвата/сравнения 4

— Бит 3 **CC3IF**: Флаг прерывания захвата/сравнения 3

— Бит 2 **CC2IF**: Флаг прерывания захвата/сравнения 2

— Бит 1 **CC1IF**: Флаг прерывания захвата/сравнения 1

В режиме вывода CC1:

Есть исключения для выравнивания по центру (см. биты CMS в регистре TIMx_CR1).

В режиме ввода CC1:

— Бит 0 **UIF**: Флаг прерывания обновления

Удержание прерывания обновления ставится при:

- Переполнении или исчерпании счётчика и нулевом счётчике и UDIS=0 в регистре TIMx_CR1.
- При программной инициализации CNT битом UG в регистре TIMx_EGR, если URS=0 и UDIS=0 в регистре TIMx_CR1.
- При инициализации CNT событием запуска (см. регистр TIMx_SMCR), если URS=0 и UDIS=0 в регистре TIMx_CR1.

14.4.6. Регистр генерации событий (TIMx_EGR)

Событие генерируется программной установкой бита, снимается он аппаратно.

Смещение адреса: 0x14

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
								w	w	w	w	w	w	w	w

— Биты 15:8 Резерв, не трогать.

— Бит 7 **BG**: Останов.

1: Событие останова. Бит MOE сбрасывается, флаг BIF ставится. Могут возникнуть прерывание и запрос DMA.

— Бит 6 **TG**: Запуск.

1: Ставит флаг TIF в регистре TIMx_SR. Могут возникнуть прерывание и запрос DMA.

— Бит 5 **COMG**: Событие COM.

1: Стоящий бит CCPC разрешает обновление битов CCxE, CCxNE и OCxM

NB: Работает только у каналов с инверсными выходами.

— Бит 4 **CC4G**: Захват/сравнение 4.

— Бит 3 **CC3G**: Захват/сравнение 3.

— Бит 2 **CC2G**: Захват/сравнение 2.

— Бит 1 **CC1G**: Захват/сравнение 1.

1: Выдаёт событие захвата/сравнения:

В режиме вывода CC1:

Ставит флаг CC1IF. Могут возникнуть прерывание и запрос DMA.

В режиме ввода CC1:

Текущее значение счётчика пишется в TIMx_CCR1. Ставит флаг CC1IF. Могут возникнуть прерывание и запрос DMA. При уже стоящем флаге CC1IF ставится и флаг CC1OF.

— Бит 0 **UG**: Обновление.

1: Инициализирует счётчик и обновление регистров. Текущий счётчик предделителя чистится не влияя на само значение. В режиме по центру или при DIR=0 счётчик обнуляется, иначе перегружается из регистра TIMx_ARR (при DIR=1).

14.4.7. Регистр режима захвата/сравнения 1 (TIMx_CCMR1)

Смещение адреса: 0x18

По сбросу: 0x0000

Каналы могут работать на ввод (захват) и вывод (сравнение), что определяется битами `CCxS`. У остальных битов для ввода и вывода значение отличается. `OCxx` обозначает функции вывода, `ICxx` функции ввода. При этом используются разные каскады.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Сравнение выхода:

— Бит 15 **OC2CE:** Разрешение очистки сравнения 2.

— Биты 14:12 **OC2M[2:0]**: Режим сравнения выхода 2.

— Бит 11 **OC2PE**: Разрешение предзагрузки сравнения 2.

— Бит 10 **OC2FE**: Быстрое разрешение сравнения 2.

— Биты 9:8 **CC2S[1:0]**: Выбор Захвата/Сравнения 2.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC2 выходной

01: Канал CC2 входной, IC2 на TI2

10: Канал CC2 входной, IC2 на TI1

11: Канал CC2 входной, IC2 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC2S можно писать только при выключенном канале (CC2E = ‘0’ в TIMx_CCER).

— Бит 7 **OC1CE**: Разрешение очистки сравнения выхода 1.

0: ETRF на OC1REF не влияет

1: Высокий уровень на ETRF стирает OC1REF

— Биты 6:4 **OC1M[2:0]**: Режим сравнения выхода 1.

Определяют поведение выходного опорного сигнала OC1REF, от которого происходят OC1 и OC1N. OC1REF активен высоким, а OC1 и OC1N зависят от битов CC1P и CC1NP.

000: Заморозка - сравнение TIMx_CCR1 и TIMx_CNT на выходы не влияет (режим используется для генерации временной базы).

001: При совпадении TIMx_CNT и TIMx_CCR1 сигнал OC1REF ставится активным (высоким).

010: При совпадении TIMx_CNT и TIMx_CCR1 сигнал OC1REF ставится неактивным (низким).

011: Переключение - при TIMx_CNT=TIMx_CCR1 сигнал OC1REF перебрасывается.

100: Принудительный неактивный - OC1REF ставится низким.

101: Принудительный активный - OC1REF ставится высоким.

110: ШИМ 1 - при прямом счёте канал активен пока TIMx_CNT<TIMx_CCR1. При обратном счёте канал неактивен (OC1REF=‘0’) пока TIMx_CNT>TIMx_CCR1.

111: ШИМ 2 - при прямом счёте канал неактивен пока TIMx_CNT<TIMx_CCR1. При обратном счёте канал активен пока TIMx_CNT>TIMx_CCR1.

NB:

1: Эти биты нельзя изменить пока стоит LOCK уровень 3 (биты LOCK в регистре TIMx_BDTR) и CC1S=’00’ (канал включён на выход).

2: В ШИМ 1 и 2 уровень OCREF изменяется только когда изменяется результат сравнения или при переключении из “Заморозки” в “ШИМ”.

— Бит 3 **OC1PE**: Разрешение регистра предзагрузки.

0: Предзагрузка TIMx_CCR1 выключена. TIMx_CCR1 можно писать в любое время, новое значение начинает действовать незамедлительно.

1: Предзагрузка TIMx_CCR1 включена. Доступ идёт к регистру предзагрузки. TIMx_CCR1 пишется в активный регистр по событию обновления.

NB:

1: Эти биты нельзя изменить пока стоит LOCK уровень 3 (биты LOCK в регистре TIMx_BDTR) и CC1S=’00’ (канал включён на выход).

2: Режим ШИМ без установки регистра предзагрузки можно использовать только в режиме одного импульса (стоит бит OPM в регистре TIMx_CR1). Иначе поведение непредсказуемо.

— Бит 2 **OC1FE**: Разрешение быстрого сравнения.

Ускоряет действие события запуска по входу на выход CC.

0: CC1 работает как обычно (сравнение счётчика и CCR1) даже при включённом запуске. Минимальная задержка переключения выхода CC1 при фронте на входе составляет 5 тактов.

1: Активный фронт запуска по входу действует как сравнение на выходе CC1. Тогда OC ставится в уровень сравнения независимо от его результата. Задержка между импульсом на входе и выходом CC1 сокращается до 3 тактов. OCFE работает только в ШИМ 1 и ШИМ 2.

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC1 выходной

01: Канал CC1 входной, IC1 на TI1

10: Канал CC1 входной, IC1 на TI2

11: Канал CC1 входной, IC1 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC1S можно писать только при выключенном канале (CC1E = ‘0’ в TIMx_CCER).

Захват входа.

— Биты 15:12 **IC2F**: Фильтр захвата входа 2.

— Биты 11:10 **IC2PSC[1:0]** : Предделитель захвата входа 2.

— Биты 9:8 **CC2S[1:0]**: Выбор Захвата/Сравнения 2.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC2 выходной

01: Канал CC2 входной, IC2 на TI2

10: Канал CC2 входной, IC2 на TI1

11: Канал CC2 входной, IC2 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC2S можно писать только при выключенном канале (CC2E = ‘0’ в TIMx_CCER).

— Биты 7:4 **IC1F[3:0]**: Фильтр захвата входа 1.

Определяет частоту выборки и длину цифрового фильтра сигнала TI1. Цифровой фильтр это счётчик, выдающий сигнал на выход после N последовательных событий на входе:

0000: Без фильтра, выборка на частоте f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

— Биты 3:2 **IC1PSC**: Предделитель захвата входа 1

Значение предделителя на входе CC1 (IC1).

По CC1E=’0’ (регистр TIMx_CCER) предделитель сбрасывается.

00: без предделителя, по каждому фронту на входе

01: каждые 2 события

10: каждые 4 события

11: каждые 8 события

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC1 выходной

01: Канал CC1 входной, IC1 на TI1

10: Канал CC1 входной, IC1 на TI2

11: Канал CC1 входной, IC1 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC1S можно писать только при выключенном канале (CC1E = ‘0’ в TIMx_CCER).

14.4.8. Регистр режима захвата/сравнения 2 (TIMx_CCMR2)

Смещение адреса: 0x1C

По сбросу: 0x0000

Каналы могут работать на ввод (захват) и вывод (сравнение), что определяется битами `CCxS`. У остальных битов для ввода и вывода значение отличается. `OCxx` обозначает функции вывода, `ICxx` функции ввода. При этом используются разные каскады.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE.	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Сравнение выхода:

— Бит 15 **OC4CE:** Разрешение очистки сравнения 4.

— Биты 14:12 **OC4M[2:0]**: Режим сравнения выхода 4.

— Бит 11 **OC4PE**: Разрешение предзагрузки сравнения 4.

— Бит 10 **OC4FE**: Быстрое разрешение сравнения 4.

— Биты 9:8 **CC4S[1:0]**: Выбор Захвата/Сравнения 4.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC4 выходной

01: Канал CC4 входной, IC4 на TI4

10: Канал CC4 входной, IC4 на TI3

11: Канал CC4 входной, IC4 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC4S можно писать только при выключенном канале (CC4E = ‘0’ в TIMx_CCER).

— Бит 7 **OC3CE**: Разрешение очистки сравнения выхода 3.

— Биты 6:4 **OC3M[2:0]**: Режим сравнения выхода 3.

— Бит 3 **OC3PE**: Разрешение регистра предзагрузки.

— Бит 2 **OC3FE**: Разрешение быстрого сравнения.

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC3 выходной

01: Канал CC3 входной, IC3 на TI3

10: Канал CC3 входной, IC3 на TI4

11: Канал CC3 входной, IC3 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC3S можно писать только при выключенном канале (CC3E = ‘0’ в TIMx_CCER).

Захват входа.

— Биты 15:12 **IC4F**: Фильтр захвата входа 4.

— Биты 11:10 **IC4PSC[1:0]** : Предделитель захвата входа 4.

— Биты 9:8 **CC4S[1:0]**: Выбор Захвата/Сравнения 4.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC4 выходной

01: Канал CC4 входной, IC4 на TI4

10: Канал CC4 входной, IC4 на TI3

11: Канал CC4 входной, IC4 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC4S можно писать только при выключенном канале (CC4E = ‘0’ в TIMx_CCER).

— Биты 7:4 **IC3F[3:0]**: Фильтр захвата входа 3.

Определяет частоту выборки и длину цифрового фильтра сигнала TI3. Цифровой фильтр это счётчик, выдающий сигнал на выход после N последовательных событий на входе:

0000: Без фильтра, выборка на частоте f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

— Биты 3:2 **IC3PSC**: Предделитель захвата входа 3

— Биты 1:0 **CC3S**: Выбор Захвата/Сравнения 3.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC3 выходной

01: Канал CC3 входной, IC3 на TI3

10: Канал CC3 входной, IC3 на TI4

11: Канал CC3 входной, IC3 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC3S можно писать только при выключенном канале (CC3E = ‘0’ в TIMx_CCER).

14.4.9. Регистр разрешения захвата/сравнения (TIMx_CCER)

Смещение адреса: 0x20

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:14 Резерв, не трогать.

— Бит 13 **CC4P**: Полярность выхода захвата/сравнения 4

— Бит 12 **CC4E**: Разрешение выхода захвата/сравнения 4

— Бит 11 **CC3NP**: Полярность инверсного выхода захвата/сравнения 3

— Бит 10 **CC3NE**: Разрешение инверсного выхода захвата/сравнения 3

— Бит 9 **CC3P**: Полярность выхода захвата/сравнения 3

— Бит 8 **CC3E**:, Разрешение выхода захвата/сравнения 3

— Бит 7 **CC2NP**: Полярность инверсного выхода захвата/сравнения 2

— Бит 6 **CC2NE**: Разрешение инверсного выхода захвата/сравнения 2

— Бит 5 **CC2P**: Полярность выхода захвата/сравнения 2

— Бит 4 **CC2E**: Разрешение выхода захвата/сравнения 2

— Бит 3 **CC1NP**: Полярность инверсного выхода захвата/сравнения 1

0: OC1N активен высоким.

1: OC1N активен низким.

NB: Нельзя писать пока стоит LOCK уровень 2 или 3 (биты LOCK в регистре TIMx_BDTR) и CC1S=”00” (канал выводной).

— Бит 2 **CC1NE**: Разрешение инверсного выхода захвата/сравнения 1

0: Выкл. - OC1N не активен. Уровень OC1N функция битов MOE, OSSI, OSSR, OIS1, OIS1N и CC1E.

1: Вкл. - OC1N выводится на ножку, определённую битами MOE, OSSI, OSSR, OIS1, OIS1N и CC1E.

— Бит 1 **CC1P**: Полярность выхода захвата/сравнения 1

1: Выдаёт событие захвата/сравнения:

В режиме вывода CC1:

0: OC1 активен высоким.

1: OC1 активен низким.

В режиме ввода CC1:

Выбирает для запуска или захвата прямой или инверсный IC1.

0: прямой: захват по переднему фронту IC1. Внешний запуск по IC1 не инвертируется.

1: инверсный: захват по заднему фронту IC1. Внешний запуск по IC1 инвертируется.

NB: Нельзя писать пока стоит LOCK уровень 2 или 3 (биты LOCK в регистре TIMx_BDTR).

— Бит 0 **CC1E**: Разрешение выхода захвата/сравнения 1

В режиме вывода CC1:

0: Выкл. - OC1 не активен. Уровень OC1 функция битов MOE, OSSI, OSSR, OIS1, OIS1N и CC1NE.

1: Вкл. - OC1 выводится на ножку, определённую битами MOE, OSSI, OSSR, OIS1, OIS1N и CC1NE.

В режиме ввода CC1:

0: Захват выключен.

1: Захват включен.

Таблица 83. Управляющие биты OCx and OCxN

Управляющие биты					Состояние выхода(1)	
MOE	**OSSI**	**OSSR**	**CCxE**	**CCxNE**	**Состояние OCx**	**Состояние OCxN**
1	X	0	0	0	Выкл. (таймером не управляется), OCx=0, OCx_EN=0	Выкл. (таймером не управляется), OCxN=0, OCxN_EN=0
		0	0	1	Выкл. (таймером не управляется), OCx=0, OCx_EN=0	OCxREF + Полярность OCxN=OCxREF xor CCxNP, OCxN_EN=1
		0	1	0	OCxREF + Полярность OCx=OCxREF xor CCxP, OCx_EN=1	Выкл. (таймером не управляется) OCxN=0, OCxN_EN=0
		0	1	1	OCREF + Полярность + Задержка OCx_EN=1	Инверсия OCREF + Полярность + Задержка OCxN_EN=1
		1	0	0	Выкл. (таймером не управляется) OCx=CCxP, OCx_EN=0	Выкл. (таймером не управляется) OCxN=CCxNP, OCxN_EN=0
		1	0	1	Включён, неактивное состояние, OCx=CCxP, OCx_EN=1	OCxREF + Полярность OCxN=OCxREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF + Полярность OCx=OCxREF xor CCxP, OCx_EN=1	Включён, неактивное состояние, OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCREF + Полярность + Задержка OCx_EN=1	Инверсия OCREF + Полярность + Задержка OCxN_EN=1
	0		0	0	Output Disabled (not driven by the timer) Asynchronously: OCx=CCxP, OCx_EN=0, OCxN=CCxNP, OCxN_EN=0 Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state.	
	0		0	1		
	0		1	0		

0	0	X	1	1	
	1		0	0	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCxN_EN=1 Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state
	1		0	1	
	1		1	0	
	1		1	1	

1. When both outputs of a channel are not used (CCxE = CCxNE = 0), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

NB: Состояние внешних ножек I/O, подключённых к комплементарным OCx и OCxN, зависит от состояния OCx и OCxN и регистров GPIO и AFIO.

14.4.10.Счётчик TIM1 и TIM8 (TIMx_CNT)

Смещение адреса: 0x24

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **CNT[15:0]**: Значение счётчика.

14.4.11.Предделитель TIM1 и TIM8 (TIMx_PSC)

Смещение адреса: 0x28

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **PSC[15:0]**: Значение предделителя

Частота счёта (CK_CNT) равна f_{CK_PSC} / (PSC[15:0] + 1).

Значение PSC пишется в рабочий регистр предделителя по каждому событию обновления (включая очистку счётчика битом UG регистра TIMx_EGR или запуском в режиме Сброса).

14.4.12.Регистр предзагрузки TIM1 и TIM8 (TIMx_ARR)

Смещение адреса: 0x2C

По сбросу: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **ARR[15:0]**: Значение перезагрузки.

Если значение нулевое, то счётчик блокируется.

14.4.13.Счётчик повторения TIM1 и TIM8 (TIMx_RCR)

Смещение адреса: 0x30

По сбросу: 0x0000

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

— Биты 15:8 Резерв, не трогать.

— Биты 7:0 **REP[7:0]**: Значение счётчика.

Регистр позволяет изменять частоту обновления регистров сравнения и частоту выдачи прерываний.

При каждом обнулении обратного счётчика REP_CNT выдаётся событие обновления. Он перегружается значением REP только по событию U_RC.

То есть в режиме ШИМ (REP+1) соответствует:

– числу периодов ШИМ в режиме по фронту
– числу полупериодов ШИМ в режиме по центру.

14.4.14.Регистр 1 захвата/сравнения TIM1 и TIM8 (TIMx_CCR1)

Смещение адреса: 0x34

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR1[15:0]**: Значение захвата/сравнения

В режиме вывода CC1:

Значение предзагрузки CCR1 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC1PE в регистре TIMx_CCMR1), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC1.

В режиме ввода CC1:

CCR1 содержит значение счётчика по последнему событию захвата 1 (IC1). Здесь TIMx_CCR1 только читается.

14.4.15.Регистр 2 захвата/сравнения TIM1 и TIM8 (TIMx_CCR2)

Смещение адреса: 0x38

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR2[15:0]**: Значение захвата/сравнения

В режиме вывода CC2:

Значение предзагрузки CCR2 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC2PE в регистре TIMx_CCMR2), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC2.

В режиме ввода CC2:

CCR2 содержит значение счётчика по последнему событию захвата 2 (IC2). Здесь TIMx_CCR2 только читается.

14.4.16.Регистр 3 захвата/сравнения TIM1 и TIM8 (TIMx_CCR3)

Смещение адреса: 0x3C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR3[15:0]**: Значение захвата/сравнения

В режиме вывода CC3:

Значение предзагрузки CCR3 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC3PE в регистре TIMx_CCMR3), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC3.

В режиме ввода CC3:

CCR3 содержит значение счётчика по последнему событию захвата 3 (IC3). Здесь TIMx_CCR3 только читается.

14.4.17.Регистр 4 захвата/сравнения TIM1 и TIM8 (TIMx_CCR4)

Смещение адреса: 0x40

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR4[15:0]**: Значение захвата/сравнения

В режиме вывода CC4:

Значение предзагрузки CCR4 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC4PE в регистре TIMx_CCMR4), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC4.

В режиме ввода CC4:

CCR4 содержит значение счётчика по последнему событию захвата 4 (IC4). Здесь TIMx_CCR4 только читается.

14.4.18.Регистр останова и задержки (TIMx_BDTR)

Смещение адреса: 0x44

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

NB:

Поскольку биты AOE, BKP, BKE, OSSI, OSSR и DTG[7:0] могут блокироваться конфигурацией LOCK, то их надо все расставить первой записью в регистр TIMx_BDTR.

— Бит 15 **MOE**: Главное разрешение вывода

Снимается асинхронно аппаратурой пр активном входе останова. Ставится программно или автоматически в зависимости от бита AOE. Работает только с выводными каналами.

0: Выходы OC и OCN выключены или переведены в состояние Простоя.

1: Выходы OC и OCN включаются их битами разрешения (CCxE, CCxNE в регистре TIMx_CCER).

— Бит 14 **AOE**: Автоматическое разрешение вывода

0: MOE ставится только программно

1: MOE ставится программно или автоматически по следующему событию обновления (если неактивен вход останова)

NB: Пока стоит LOCK уровень 1 (биты LOCK в регистре TIMx_BDTR) этот бит изменить невозможно.

— Бит 13 **BKP**: Полярность останова

0: Вход BRK активен низким

1: Вход BRK активен высоким

NB: Пока стоит LOCK уровень 1 (биты LOCK в регистре TIMx_BDTR) этот бит изменить невозможно.

NB: Вступает в действие через 1 такт APB.

— Бит 12 **BKE**: Разрешение останова

0: Входы останова (BRK и сбой CSS) отключены

1: Входы останова (BRK и сбой CSS) включены

NB: Пока стоит LOCK уровень 1 (биты LOCK в регистре TIMx_BDTR) этот бит изменить невозможно.

NB: Вступает в действие через 1 такт APB.

— Бит 11 **OSSR**: Выбор состояния отключения в режиме Run

Используется выводными каналами с комплементарными выходами при MOE=1.

0: Если неактивен, выходы OC/OCN выключены (разрешение OC/OCN =0).

1: Если неактивен, выходы OC/OCN включены с неактивным уровнем когда CCxE=1 или CCxNE=1. Тогда, разрешение OC/OCN =1

NB: Пока стоит LOCK уровень 2 (биты LOCK в регистре TIMx_BDTR) этот бит изменить невозможно.

— Бит 10 **OSSI**: Выбор состояния отключения в режиме Idle (Простой)

Используется выводными каналами при MOE=0.

0: Если неактивен, выходы OC/OCN выключены (разрешение OC/OCN =0).

1: Если неактивен, выходы OC/OCN включены первыми с уровнем Простоя когда CCxE=1 или CCxNE=1. Тогда, разрешение OC/OCN =1

NB: Пока стоит LOCK уровень 2 (биты LOCK в регистре TIMx_BDTR) этот бит изменить невозможно.

— Биты 9:8 **LOCK[1:0]**: Конфигурация защиты записи (LOCK)

00: LOCK OFF - Защиты записи нет.

01: LOCK уровень 1 = Защищены биты DTG в TIMx_BDTR, OISx и OISxN в TIMx_CR2 и BKE/BKP/AOE в регистре TIMx_BDTR.

10: LOCK уровень 2 = LOCK уровень 1 + биты полярности CC (CCxP/CCxNP в TIMx_CCER канала, поставленного на вывод битами CCxS) равно как и биты OSSR OSSI.

11: LOCK уровень 3 = LOCK уровень 2 + биты управления CC (OCxM и OCxPE в TIMx_CCMRx канала, поставленного на вывод битами CCxS).

NB: Биты LOCK можно писать только один раз после сброса. Затем регистр TIMx_BDTR замораживается до следующего сброса.

— Биты 7:0 **DTG[7:0]**: Установка генератора задержки.

Определяет задержку (DT) между комплементарными выходами.

DTG[7:5]=0xx => DT=DTG[7:0] x t_{dtg} с t_{dtg} =t_{DTS}.

DTG[7:5]=10x => DT=(64+DTG[5:0]) x t_{dtg} с T_{dtg}=2 x t_{DTS}.

DTG[7:5]=110 => DT=(32+DTG[4:0]) x t_{dtg} с T_{dtg}=8 x t_{DTS}.

DTG[7:5]=111 => DT=(32+DTG[4:0]) x t_{dtg} с T_{dtg}=16 x t_{DTS}.

Если T_{DTS}=125ns (8MHz), то задержка равна:

0 до 15875 ns шагами по 125 ns,

16 us до 31750 ns шагами по 250 ns,

32 us до 63 us шагами по 1 us,

64 us до 126 us шагами по 2 us

NB: Пока стоит LOCK уровень 1, 2 или 3 (биты LOCK в регистре TIMx_BDTR) эти биты изменить невозможно.

14.4.19.Регистр управления DMA TIM1 и TIM8 (TIMx_DCR)

Смещение адреса: 0x48

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DBL[4:0]					Reserved			DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

— Биты 15:13 Резерв, не трогать.

— Биты 12:8 **DBL[4:0]**: Длина пакета DMA

Это число передач DMA (таймер обнаруживает передачу пакета после выполнения чтения/записи по адресу регистра TIMx_DMAR).

Адрес TIMx_DMAR:

00000: 1 передача

00001: 2 передачи

00010: 3 передачи

…

10001: 18 передач

— Биты 7:5 Резерв, не трогать.

— Биты 4:0 **DBA[4:0]**: Базовый адрес DMA

Это базовый адрес передач DMA (при доступе через адрес TIMx_DMAR). DBA определяется как смещение от адреса регистра TIMx_CR1.

Пример:

00000: TIMx_CR1,

00001: TIMx_CR2,

00010: TIMx_SMCR,

...

Пример: Есть передача: DBL = 7 передач и DBA = TIMx_CR1. Тогда доступ будет к 7 регистрам, начиная с адреса TIMx_CR1.

14.4.20.Регистр адреса полного доступа DMA (TIMx_DMAR)

Смещение адреса: 0x4C

По сбросу: 0x0000

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

— Биты 31:0 **DMAB[31:0]**: Регистр пакетного доступа DMA

Чтение и запись в регистр DMAR обращаются к регистру по адресу
(адрес TIMx_CR1) + (DBA + индекс DMA) x 4

где (адрес TIMx_CR1) это адрес управляющего регистра 1, DBA это базовый адрес DMA в регистре TIMx_DCR, индекс DMA автоматически изменяется при DMA передачах от 0 до DBL из регистра TIMx_DCR.

Пример пакетного DMA

Здесь пакетом DMA полусловами пишутся регистры CCRx (x = 2, 3, 4).

Нужны следующие шаги:

1. Ставим нужный канал DMA:
   - Адрес периферии DMA пишем в регистр DMAR
   - Адрес памяти это адрес данных в RAM для записи в регистры CCRx.
   - Число передач = 3.
   - Выключаем кольцевой режим.
2. Пишем поля DBA и DBL регистра DCR: DBL = 3 передачи, DBA = 0xE.
3. Разрешаем запрос DMA обновления TIMx (ставим бит UDE в регистре DIER).
4. Включаем TIMx
5. Включаем канал DMA

Здесь каждый регистр CCRx обновляется единожды. Если регистры CCRx нужно обновить дважды, то число передач должно быть 6. Буфер в RAM содержит data1, data2, data3, data4, data5 и data6. В регистры CCRx они передаются так: по первому запросу DMA, data1 в CCR2, data2 в CCR3, data3 в CCR4 и по второму запросу обновления DMA, data4 в CCR2, data5 в CCR3 в data6 ів CCR4.

14.4.21.Карта регистров TIM1 и TIM8

0x00	**TIMx_CR1**	Reserved																	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0
0x04	**TIMx_CR2**	Reserved														OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Reserved	CCPC
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**TIMx_SMCR**	Reserved											ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0C	**TIMx_DIER**	Reserved												TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE	
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x10	**TIMx_SR**	Reserved														CC4OF	CC3OF	CC2OF	CC1OF	Reserved	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	
0x14	**TIMx_EGR**	Reserved																			BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG	
	Reset value																				0	0	0	0	0	0	0	0	
0x18	**TIMx_CCMR1** Output Compare mode	Reserved											OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]		
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	TIMx_CCMR1 Input Capture mode	Reserved											IC2F[3:0]				IC2PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1PSC [1:0]		CC1S [1:0]		
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x1C	**TIMx_CCMR2** Output Compare mode	Reserved											O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]		
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	TIMx_CCMR2 Input Capture mode	Reserved											IC4F[3:0]				IC4PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3PSC [1:0]		CC3S [1:0]		
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x20	**TIMx_CCER**	Reserved														CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**TIMx_CNT**	Reserved											CNT[15:0]																
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x28	**TIMx_PSC**	Reserved											PSC[15:0]																
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

0x2C	**TIMx_ARR**	Reserved											ARR[15:0]																
	Reset value												1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0x30	**TIMx_RCR**	Reserved																			REP[7:0]								
	Reset value																				0	0	0	0	0	0	0	0	
0x34	**TIMx_CCR1**	Reserved											CCR1[15:0]																
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x38	**TIMx_CCR2**	Reserved											CCR2[15:0]																
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x3C	**TIMx_CCR3**	Reserved											CCR3[15:0]																
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x40	**TIMx_CCR4**	Reserved											CCR4[15:0]																
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x44	**TIMx_BDTR**	Reserved											MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]								
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x48	**TIMx_DCR**	Reserved															DBL[4:0]					Reserved			DBA[4:0]				
	Reset value																0	0	0	0	0				0	0	0	0	0
0x4C	**TIMx_DMAR**	DMAB[31:0]																											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15. Таймеры общего назначения (TIM2 - TIM5)

15.1. Введение в TIM2 - TIM5

Это 16-бит само-перегружаемые таймеры с программируемым предделителем.

Они могут использоваться для измерения длительности входных импульсов (захват входа), генерации сигналов (сравнение выхода и ШИМ).

Длина импульсов и сигналов может модулироваться от нескольких микросекунд до миллисекунд с помощью предделителей таймера и тактов RCC.

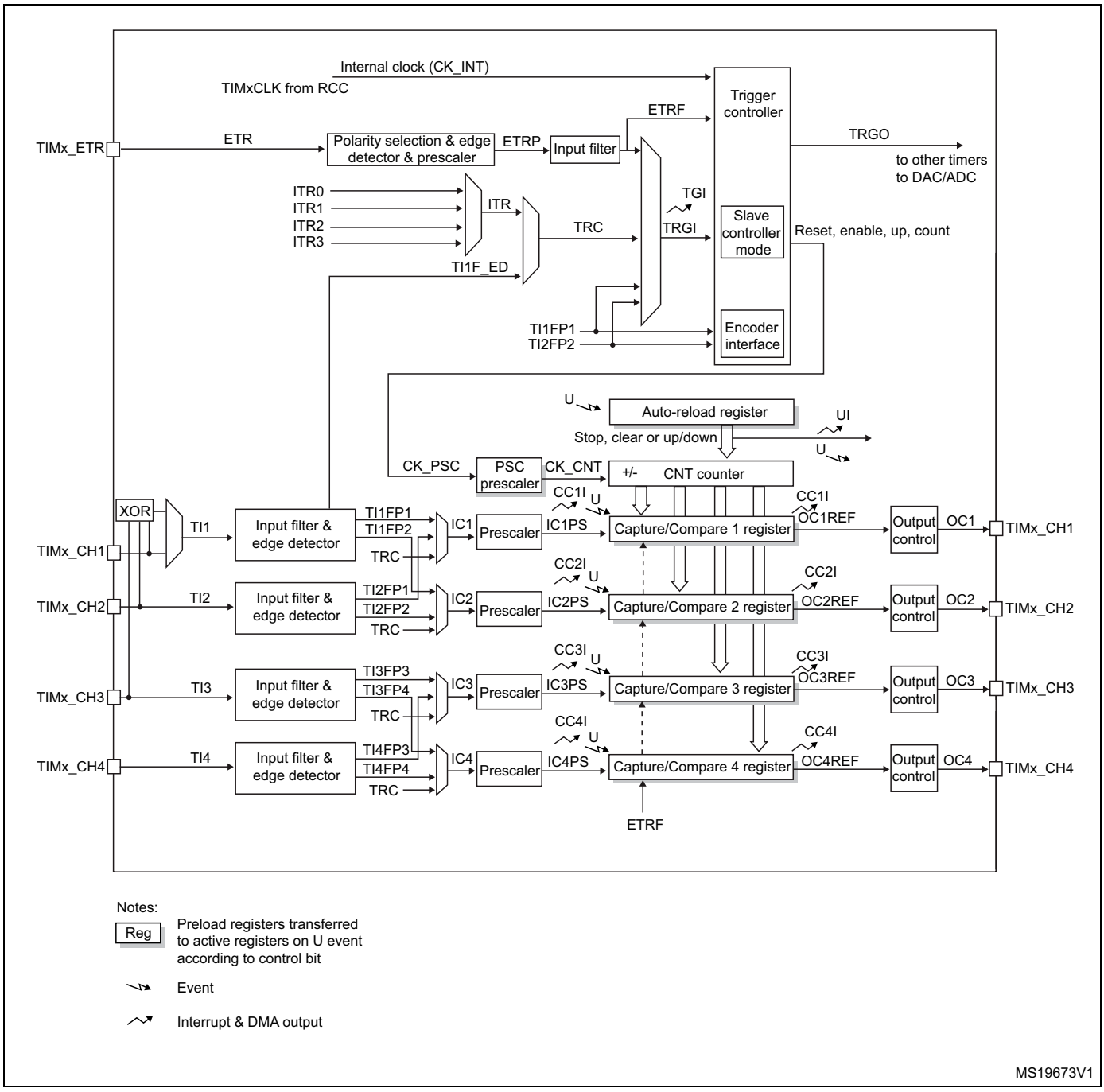
Улучшенные (TIM1 and TIM8) и обычные (TIMx) таймеры полностью независимы и не имеют общих ресурсов. Могут синхронизироваться между собой. См. *Секцию 15.3.15*.

15.2. Основные свойства TIM2 - TIM5

Это:

- 16-бит прямой, обратный, реверсивный счёт с само-перезагрузкой.
- 16-бит программируемый предделитель (можно “налету”) тактов на число от 1 до 65536.
- До 4 независимых каналов для:
  - Захвата входа
  - Сравнение выхода
  - ШИМ генерация (Фронт и Центр)
  - Одиночный импульс
- Инверсные выходы с программируемой задержкой
- Схема синхронизации внешнего управления и объединения таймеров.
- Генерация Прерываний/DMA по следующим событиям:
  - Обновление: переполнение/исчерпание, инициализация счётчика (программно или запуском)
  - Запуск (старт, стоп, инициализация или счёт по сигналу)
  - Захват входа
  - Сравнение выхода
- Поддерживает инкрементный (квадратурный) кодер и датчики Холла для позиционирования.
- Вход запуска от внешних тактов или шагового управления током.

Блок-схема таймера общего назначения.



15.3. Функциональное описание таймеров TIM2 - TIM5

15.3.1. Узел счёта

Это 16-бит счётчик с регистром само-перезагрузки. Имеет прямой, обратный и реверсивный режимы. Тактирование может поступать через предделитель.

Счётчик, регистр перезагрузки и предделитель доступны программно для чтения и записи даже во время счёта.

Включает:

- Регистр счётчика (`TIMx_CNT`)
- Регистр предделителя (`TIMx_PSC`)
- Регистр перезагрузки (`TIMx_ARR`)

Регистр перезагрузки предзагружаемый (есть видимый и скрытый регистры). Содержимое видимого регистра пишется в скрытый постоянно или по каждому событию обновления (`UEV`), в зависимости от бита разрешения (`ARPE`) в регистре `TIMx_CR1`. Оно посылается программно или при переполнении/исчерпании счётчика при снятом бите `UDIS` в регистре `TIMx_CR1`.

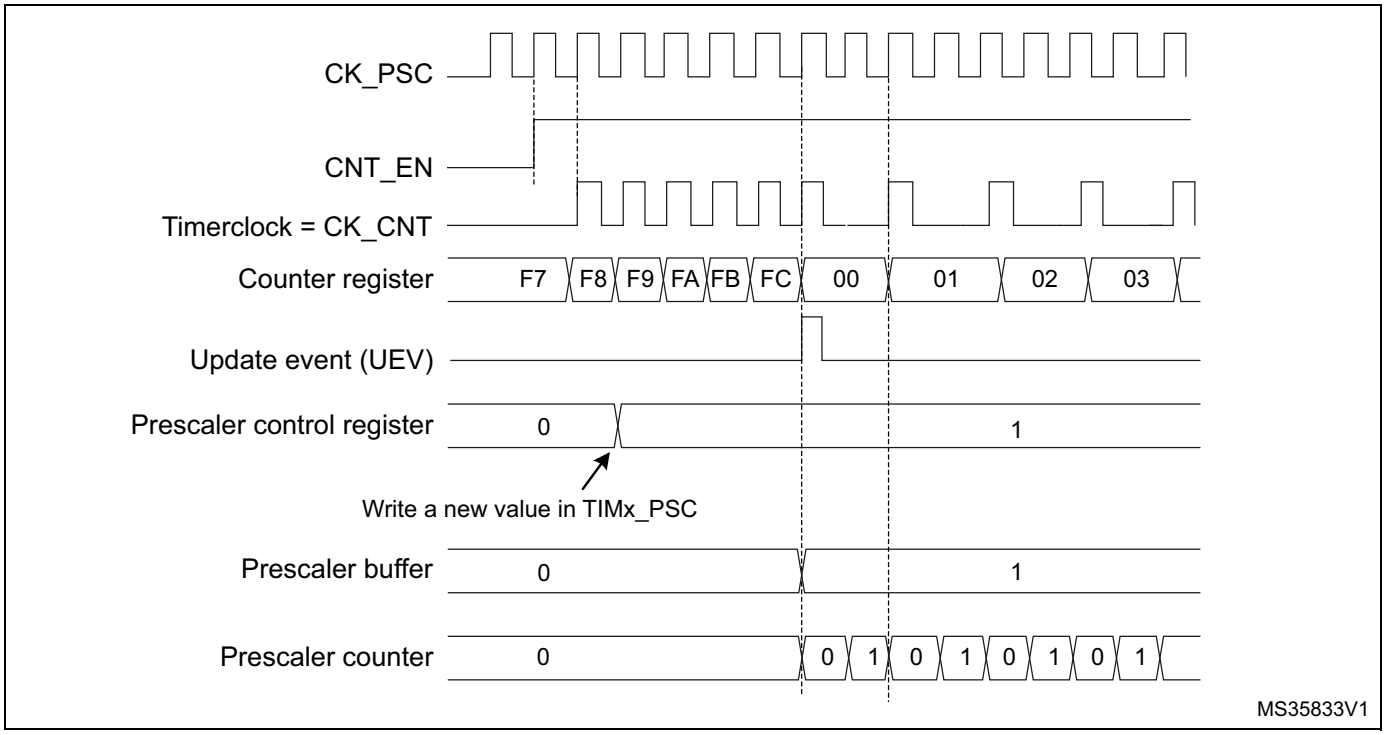
Счётчик тактируется выходом предделителя CK_CNT. Разрешается битом CEN в регистре TIMx_CR1.

Счёт начинается через 1 такт после установки бита CEN в регистре TIMx_CR1.

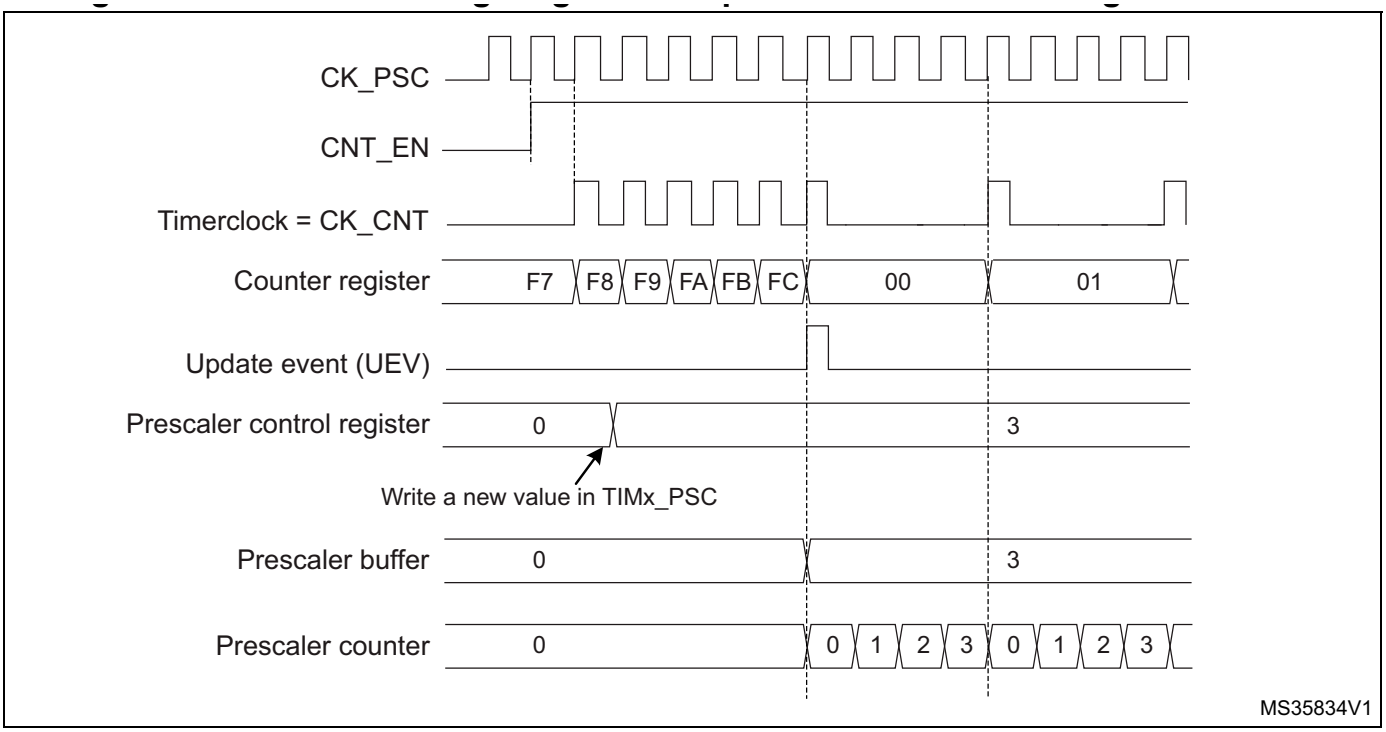
Описание предделителя

Это 16-бит счётчик делителя входной частоты (от 1 до 65536) с видимым регистром TIMx_PSC. Его можно менять налету. Новое значение счётчика пишется по следующему событию обновления.

Временная диаграмма с изменением предделителя с 1 на 2.



Временная диаграмма с изменением предделителя с 1 на 4.



15.3.2. Режимы счёта

Прямой счёт

Счёт идёт от 0 до значения перезагрузки (регистр TIMx_ARR), сбрасывается в 0 и выдаёт событие переполнения счётчика.

Событие обновления UEV выдаётся при каждом переполнении счётчика или установкой бита UG в регистре TIMx_EGR (программно или контроллером режима ведомого).

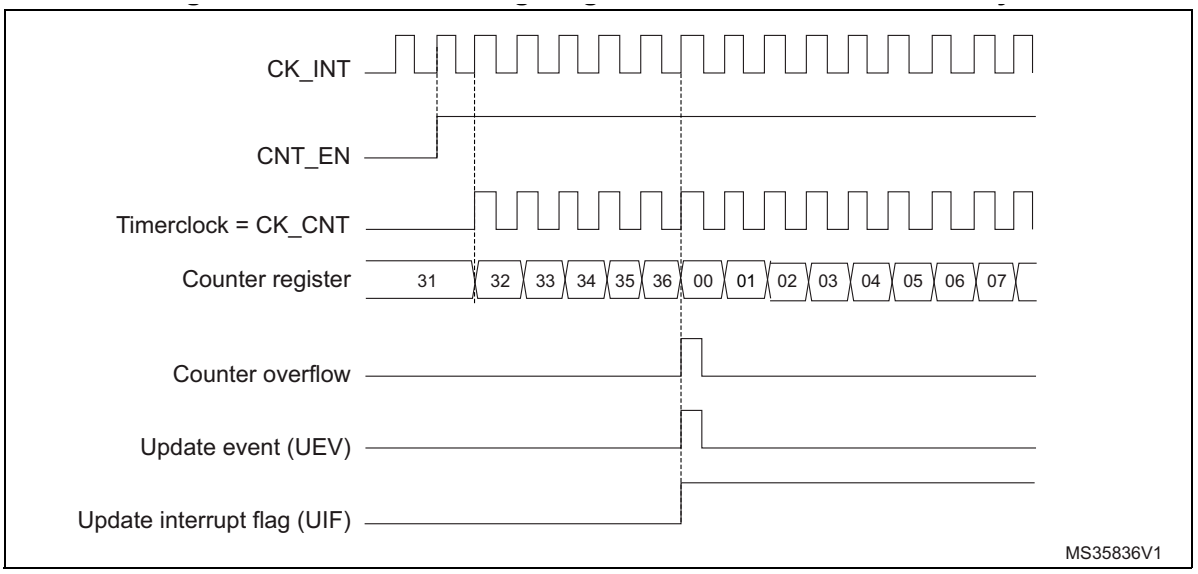
Событие UEV выключается установкой бита UDIS в регистре TIMx_CR1. Этим предупреждается запись скрытого регистра во время изменения регистра предзагрузки. Также событие UEV не появляется при сброшенном бите UDIS 0. Однако, счётчик продолжает считать с 0, равно как и предделитель (не изменив своего значения). Кроме того, если стоит бит URS в регистре TIMx_CR1, то установка бита UG выдаёт событие UEV без установки флага UIF (без прерывания и запроса DMA). Этим предупреждается одновременная выдача прерываний обновления и захвата при очистке счётчика по событию захвата.

По событию обновления обновляются все регистры и ставится флаг обновления (бит UIF в регистре TIMx_SR). Это зависит от бита URS:

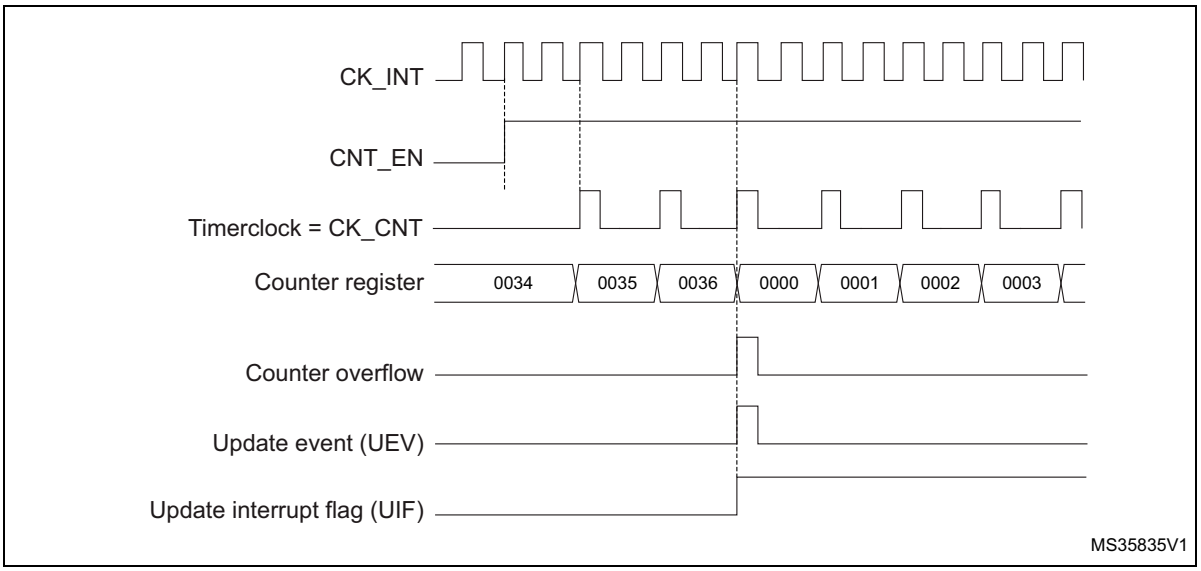
- В скрытый регистр перезагрузки пишется содержимое TIMx_ARR,
- Буфер предделителя перегружается из регистра TIMx_PSC.

Примеры ниже используют TIMx_ARR=0x36.

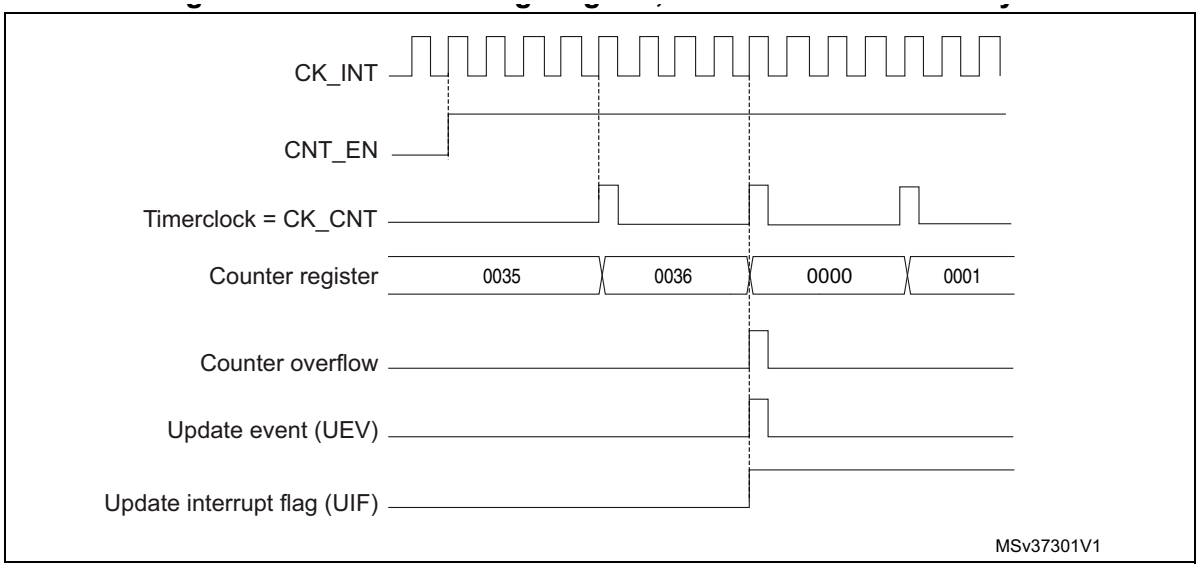
Временная диаграмма с делителем 1.



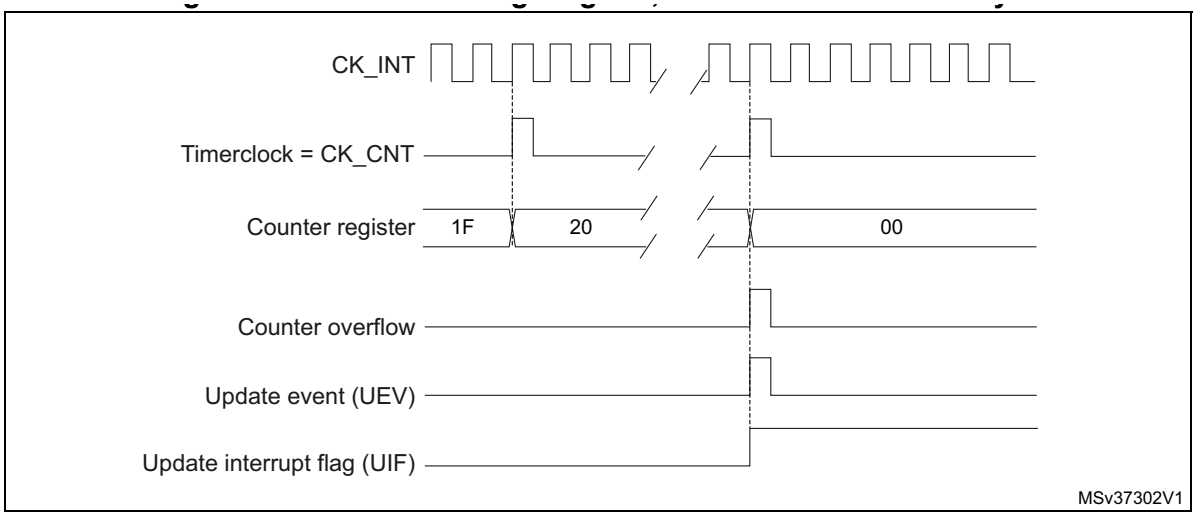
Временная диаграмма с делителем 2.



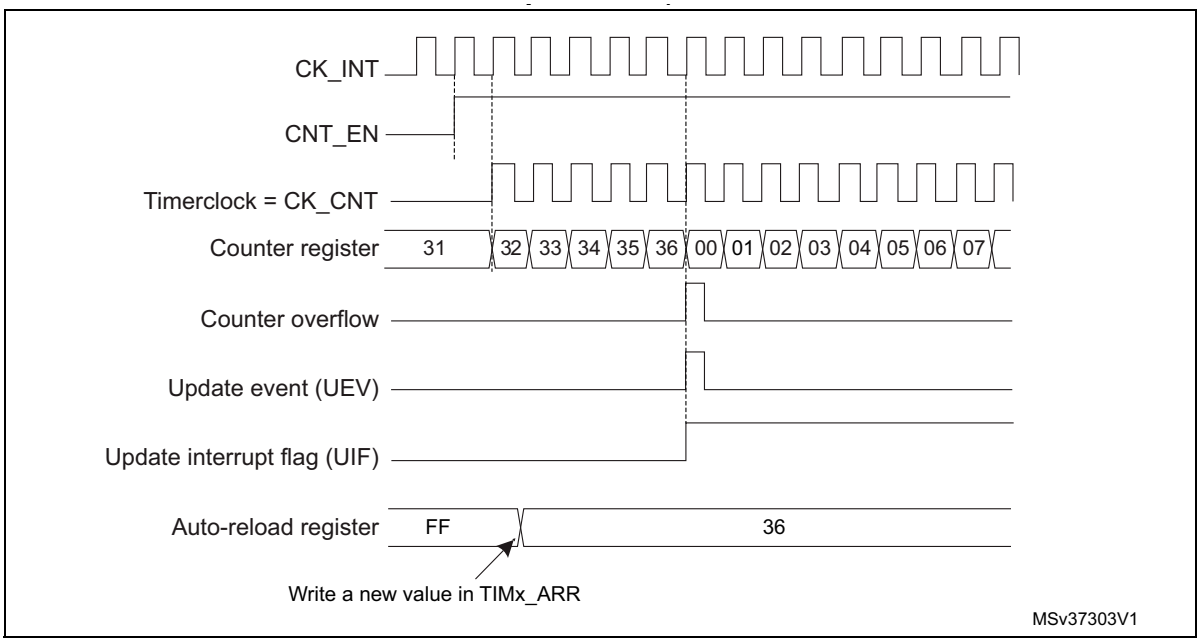
Временная диаграмма с делителем 4.



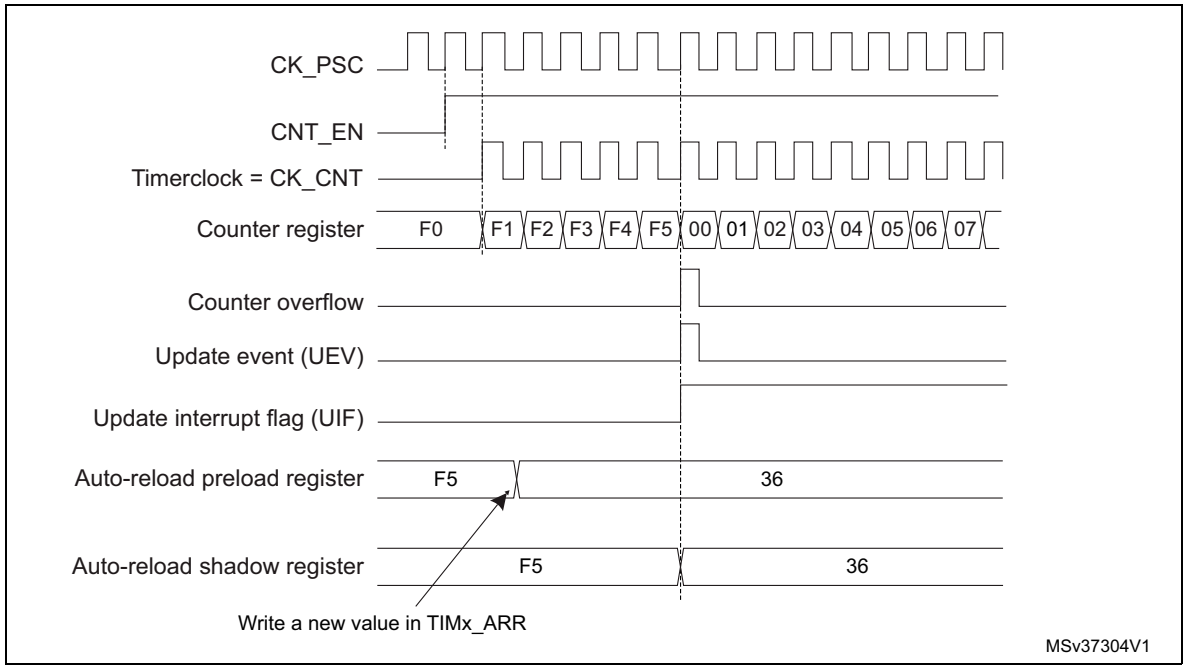
Временная диаграмма с делителем N.



Временная диаграмма события при ARPE=0 (TIMx_ARR не предзагружен).



Временная диаграмма события при ARPE=1 (TIMx_ARR предзагружен).



Обратный счёт

Счёт идёт от значения перезагрузки (регистр TIMx_ARR) до 0, перегружает счётчик и выдаёт событие исчерпания счётчика.

Событие обновления UEV выдаётся при каждом исчерпании счётчика или установкой бита UG в регистре TIMx_EGR (программно или контроллером режима ведомого).

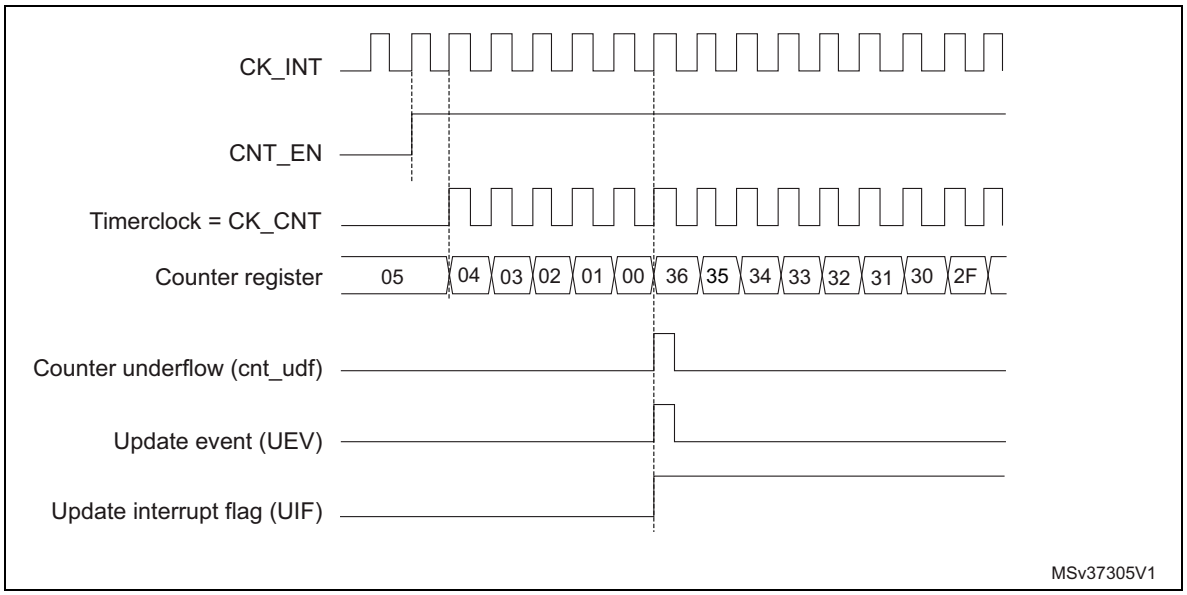
Событие UEV выключается установкой бита UDIS в регистре TIMx_CR1. Этим предупреждается запись скрытого регистра во время изменения регистра предзагрузки. Также событие UEV не появляется при сброшенном бите UDIS 0. Однако, счётчик продолжает считать с значения перезагрузки, также как и предделитель с 0 (не изменив своего значения). Кроме того, если стоит бит URS в регистре TIMx_CR1, то установка бита UG выдаёт событие UEV без установки флага UIF (без прерывания и запроса DMA). Этим предупреждается одновременная выдача прерываний обновления и захвата при очистке счётчика по событию захвата.

По событию обновления обновляются все регистры и ставится флаг обновления (бит UIF в регистре TIMx_SR). Это зависит от бита URS:

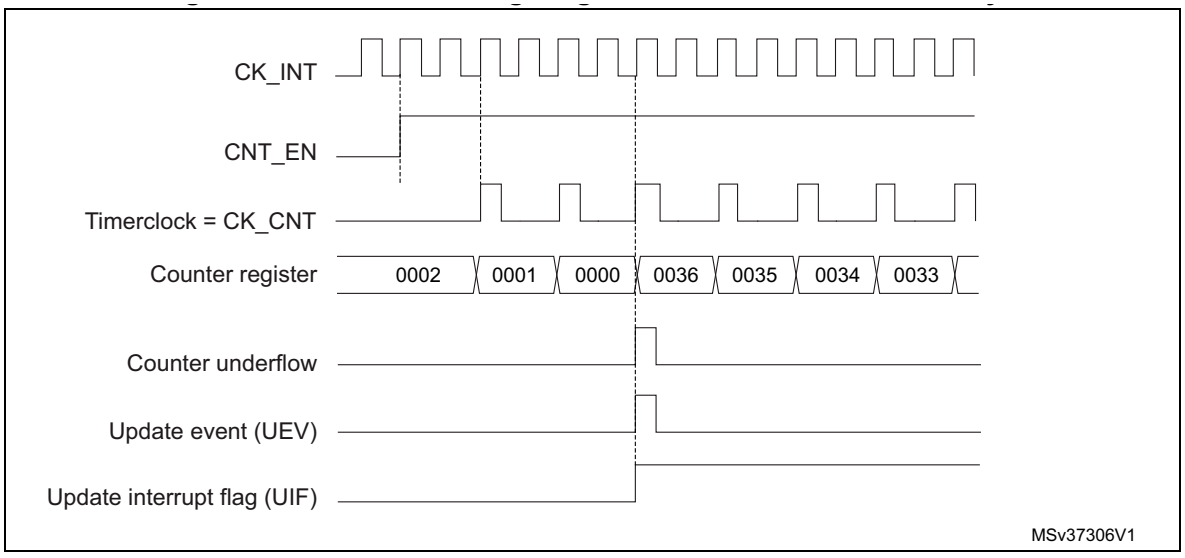
- Буфер предделителя перегружается из регистра TIMx_PSC.
- В скрытый регистр перезагрузки пишется содержимое TIMx_ARR. Скрытый регистр перезагрузки обновляется раньше счётчика и работа продолжается с нового значения.

Примеры ниже используют TIMx_ARR=0x36.

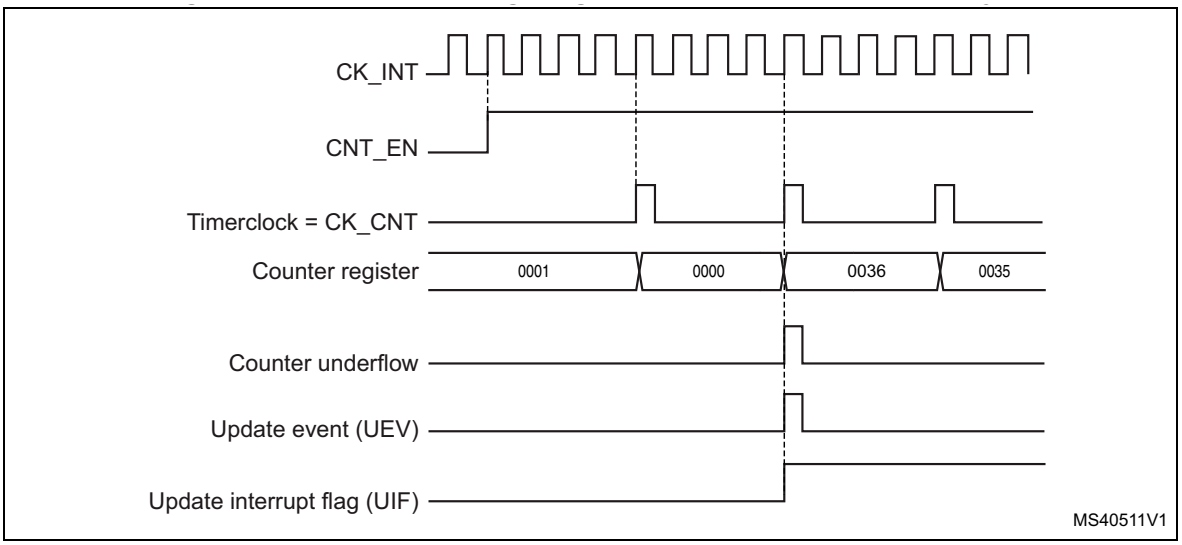
Временная диаграмма с делителем 1.



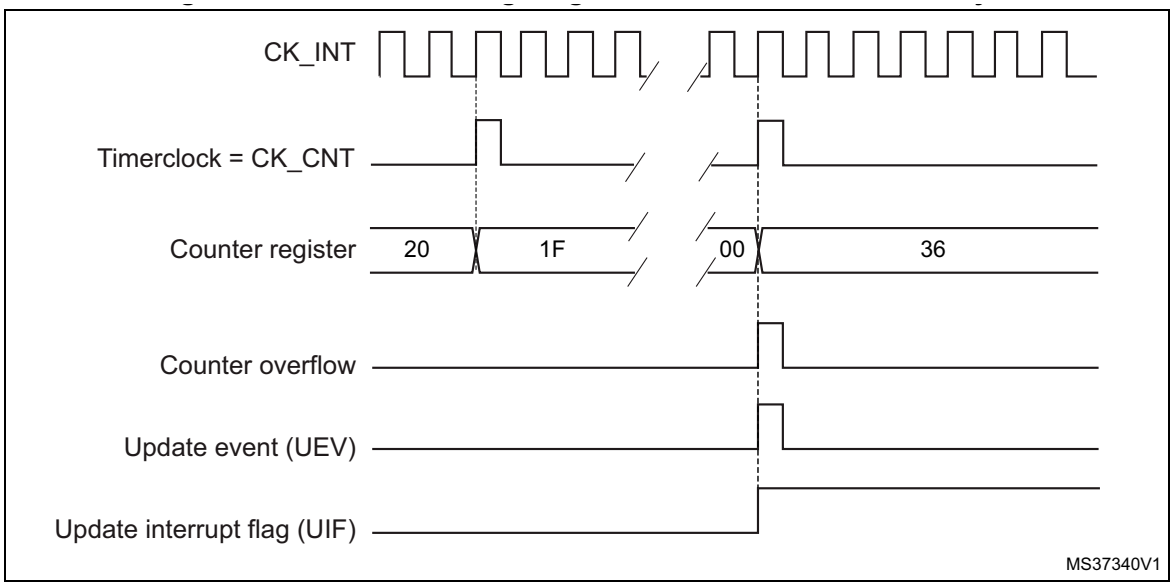
Временная диаграмма с делителем 2.



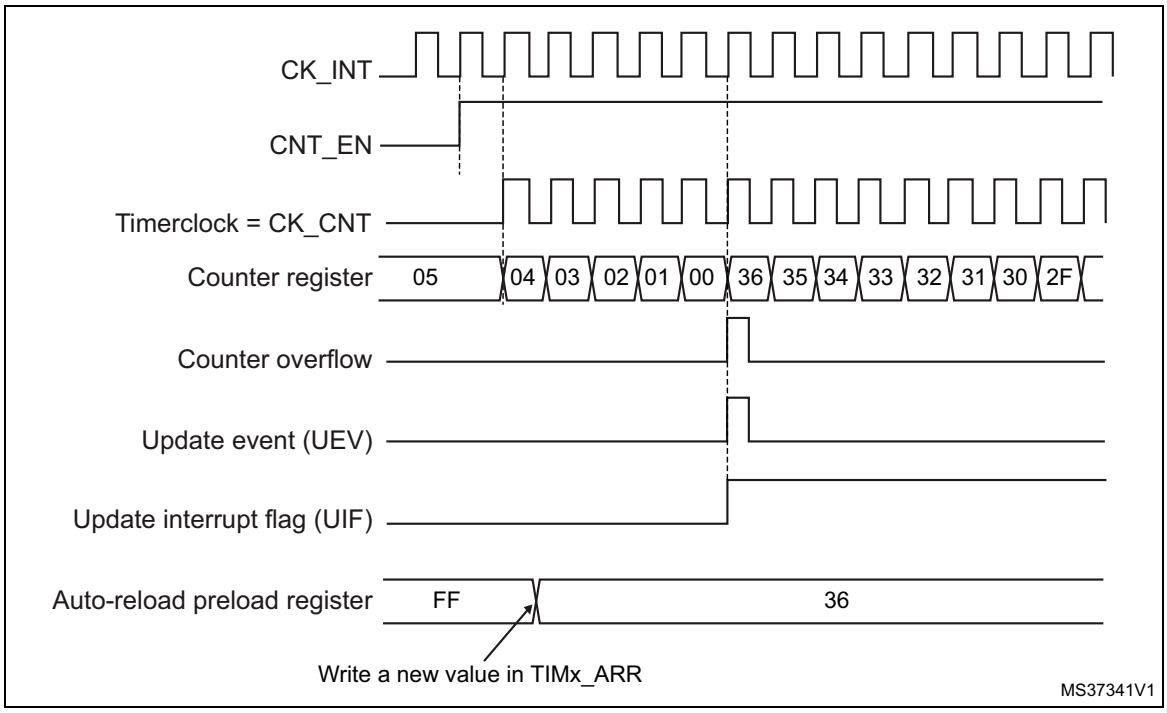
Временная диаграмма с делителем 4.



Временная диаграмма с делителем N.



Временная диаграмма события обновления.



Реверсивный счёт

Счёт идёт от 0 до значения перезагрузки минус 1 (регистр TIMx_ARR-1), выдаёт событие переполнения счётчика, затем считает обратно до 1, выдаёт событие исчерпания счётчика и начиняет счёт с 0.

Реверсивный режим включается если биты CMS в регистре TIMx_CR1 не равны '00'. Флаг прерывания Сравнения выхода выводных каналов ставится когда: завершается обратный счёт (Реверсивный режим 1, CMS = "01"), завершается прямой счёт (Реверсивный режим 2, CMS = "10") завершается прямой и обратный счёт (Реверсивный режим 3, CMS = "11").

В этом режиме бит DIR в регистре TIMx_CR1 писать нельзя. Он изменяется аппаратно и указывает текущее направление счёта.

Событие обновления может выдаваться на каждое переполнение и на каждое исчерпание счётчика установкой бита UG в регистре TIMx_EGR (программно или контроллером режима ведомого). В этом случае счётчик и предделитель начинают счёт с 0.

Событие UEV выключается установкой бита UDIS в регистре TIMx_CR1. Этим предупреждается запись скрытого регистра во время изменения регистра предзагрузки. Также событие UEV не появляется при сброшенном бите UDIS 0. Однако, счётчик продолжает считать с вверх и вниз, основываясь на текущем значении предзагрузки.

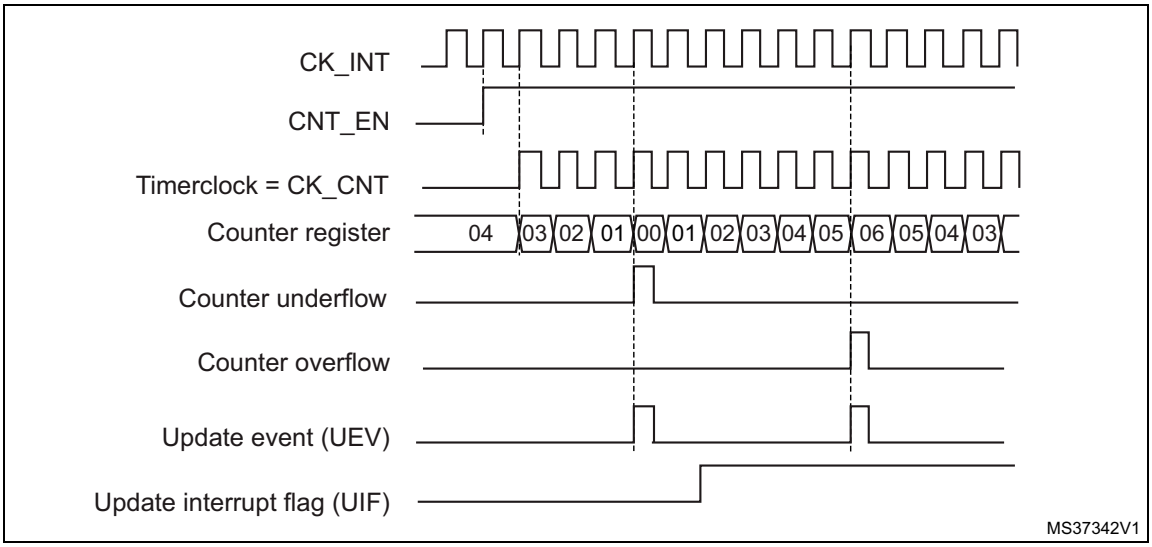
Кроме того, если стоит бит URS в регистре TIMx_CR1, то установка бита UG выдаёт событие UEV без установки флага UIF (без прерывания и запроса DMA). Этим предупреждается одновременная выдача прерываний обновления и захвата при очистке счётчика по событию захвата.

По событию обновления обновляются все регистры и ставится флаг обновления (бит UIF в регистре TIMx_SR). Это зависит от бита URS:

- Буфер предделителя перегружается из регистра TIMx_PSC.
- В скрытый регистр перезагрузки пишется содержимое TIMx_ARR. В случае обновления по переполнению счётчика скрытый регистр перезагрузки обновляется раньше счётчика и работа продолжается с нового значения.

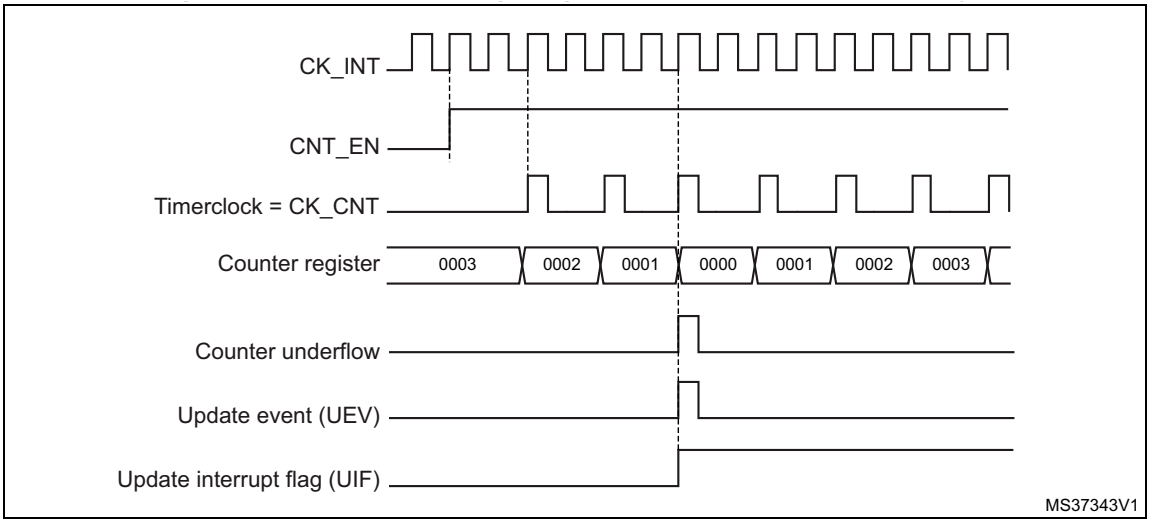
Примеры ниже используют разные частоты тактов.

Временная диаграмма с делителем 1, TIMx_ARR = 0x6.

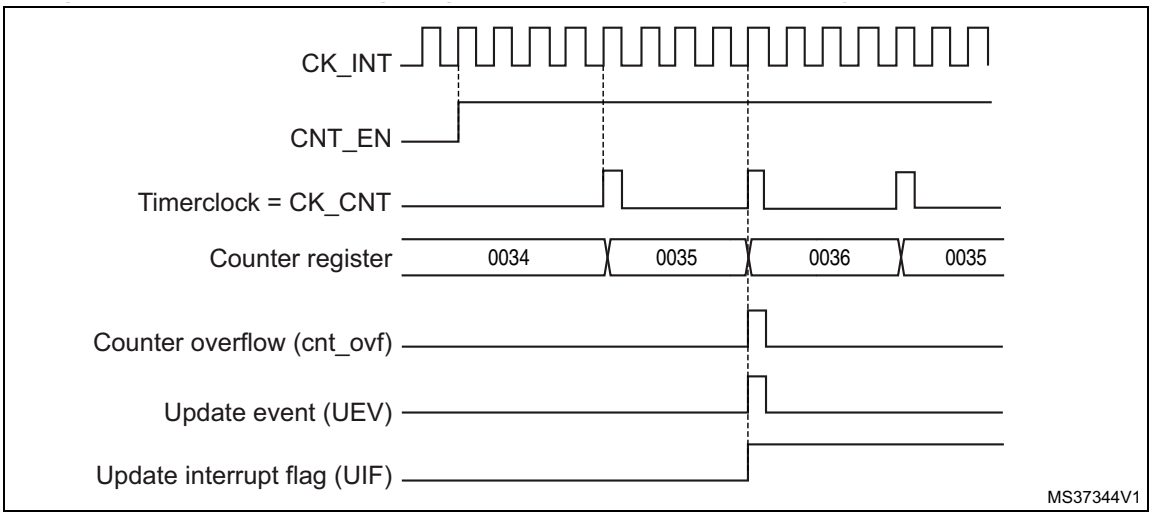


1. Реверсивный режим 1.

Временная диаграмма с делителем 2.

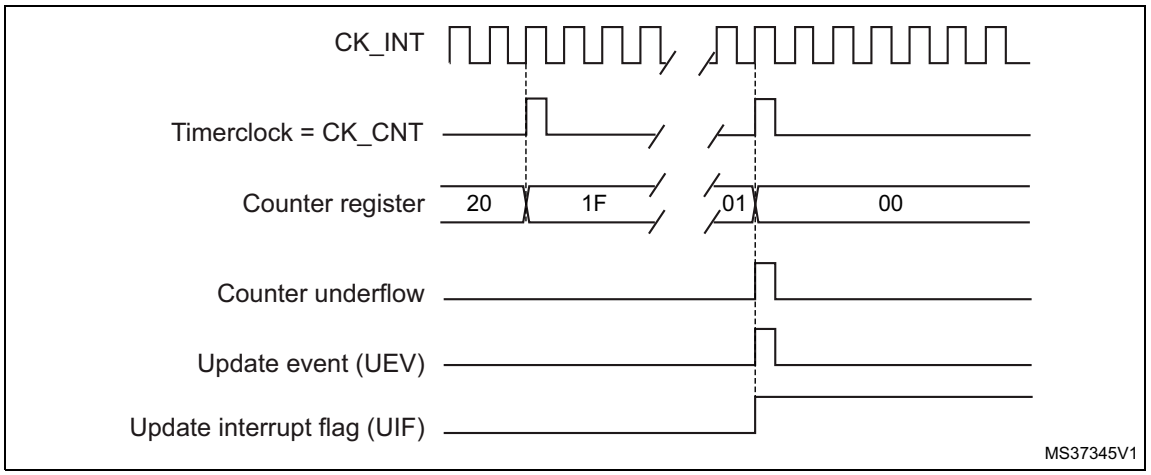


Временная диаграмма с делителем 4, TIMx_ARR=0x36.

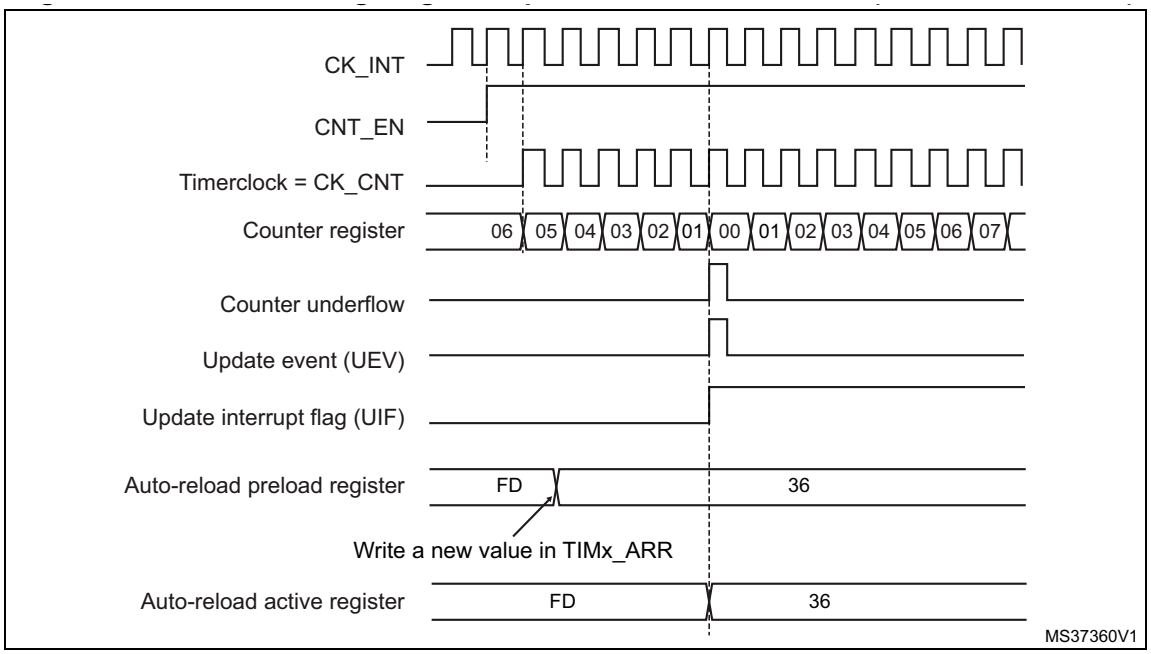


1. Реверсивный режим 2 или 3 с флагом UIF по переполнению.

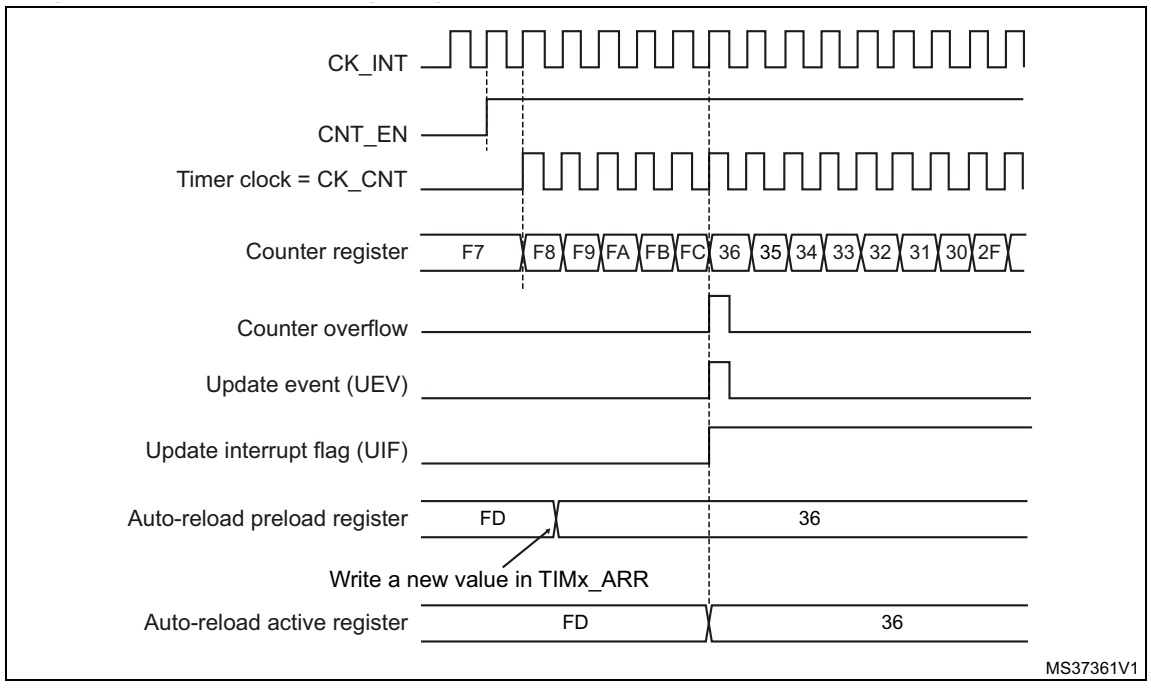
Временная диаграмма с делителем N.



Временная диаграмма, обновление с ARPE=1 (исчерпание).



Временная диаграмма, обновление с ARPE=1 (переполнение).



15.3.3. Выбор тактов

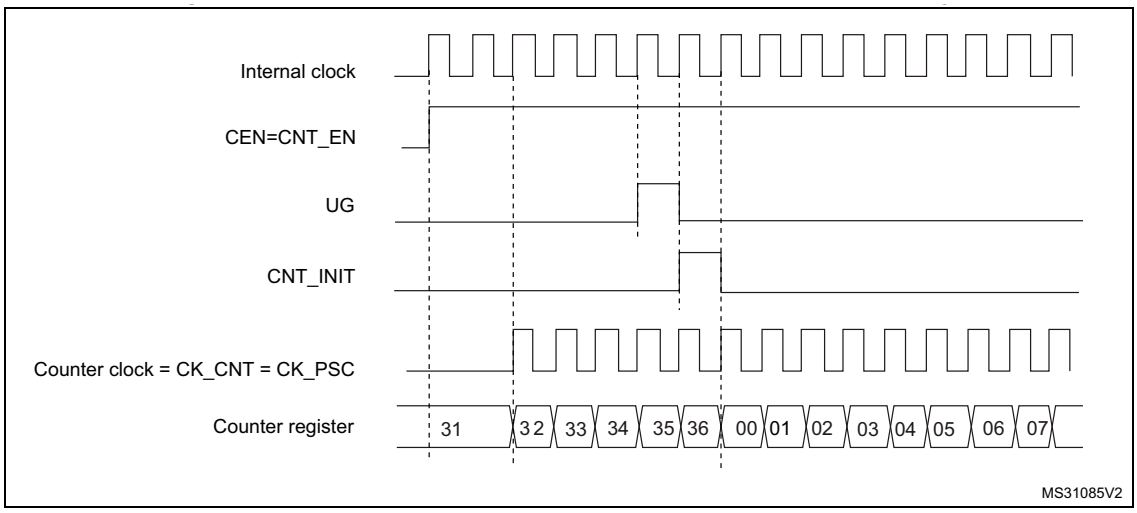
Счётчик может тактироваться из следующих источников:

- Внутренний (CK_INT)
- Внешний режим 1: внешняя ножка TIx
- Внешний режим 2: вход внешнего сигнала ETR
- Входы внутреннего сигнала (ITRx): использование одного таймера предделителем для другого.

Внутренний источник (CK_INT)

Если в ведомом режиме контроллер выключен (SMS=000), то всем управляют только биты CEN, DIR (в регистре TIMx_CR1) и UG (в регистре TIMx_EGR). Они изменяются только программно (кроме UG, снимающегося аппаратно). Сразу после установки бита CEN предделитель начинает получать внутренние такты CK_INT.

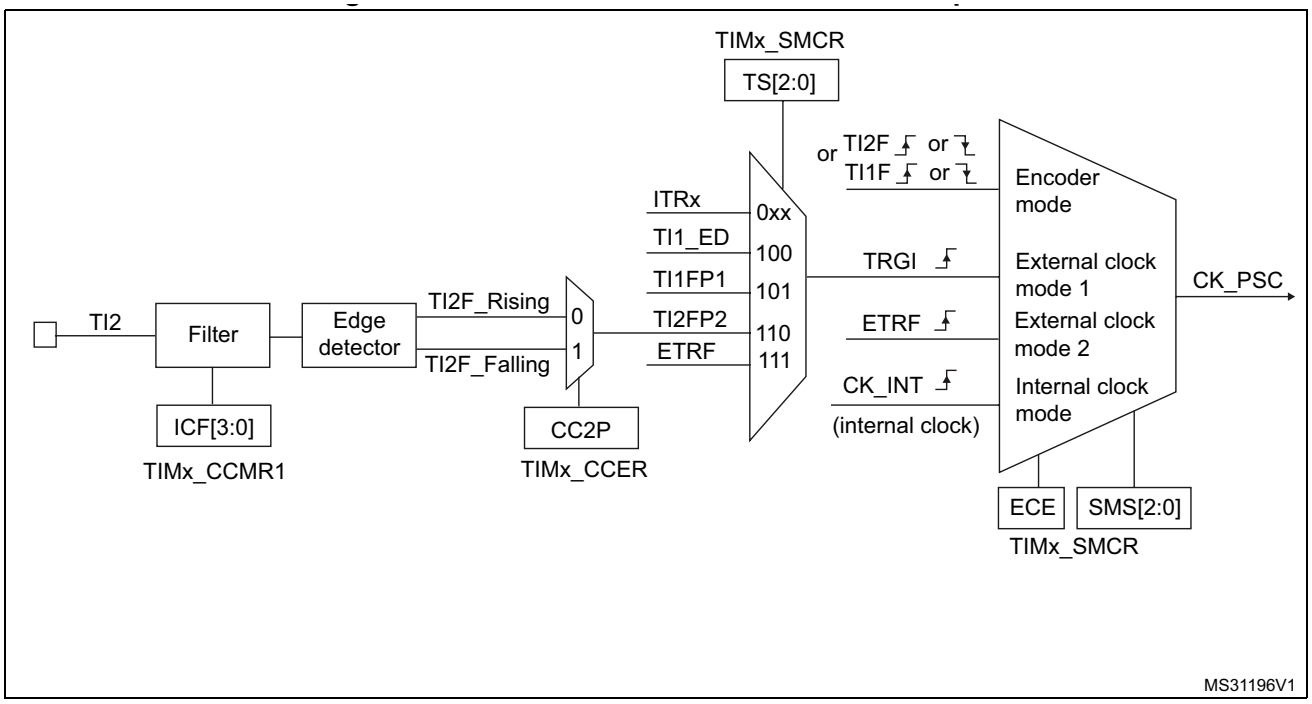
Схема управления в нормальном режиме прямого счёта без предделителя.



Внешний источник режим 1.

Включается при установке SMS=111 в регистре TIMx_SMCR. Счётчик работает от переднего или заднего фронта выбранного входа.

Внешний источник TI2.



Например, процедура включения режима прямого счёта по переднему фронту входа TI2:

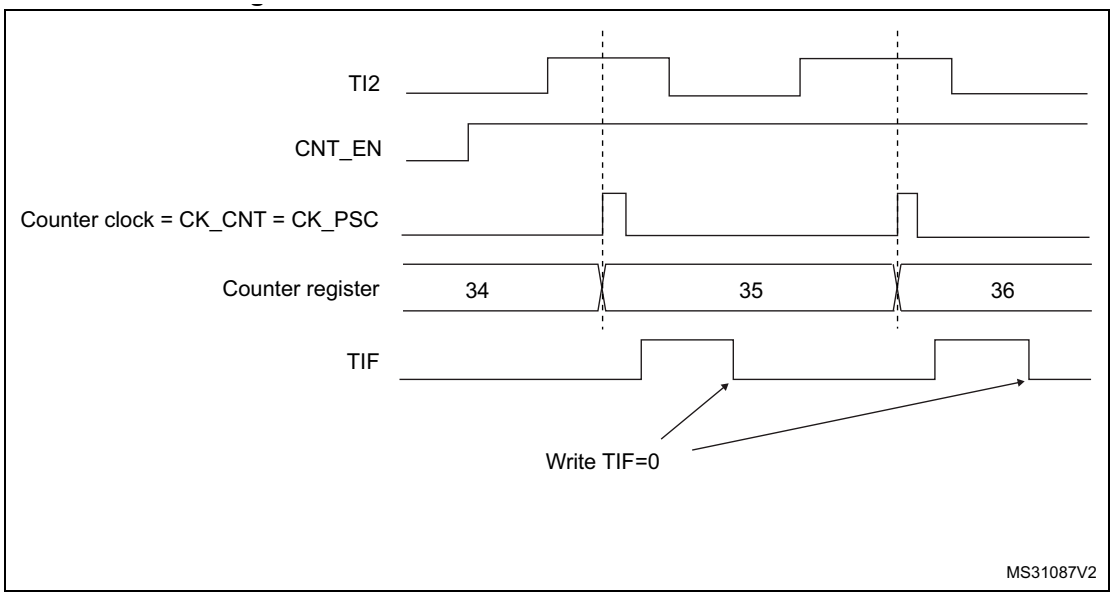
1. Включить определение переднего фронта TI2 канала 2 записью CC2S = ‘01’ в регистре TIMx_CCMR1.
2. Установить длительность входного фильтра битами IC2F[3:0] в регистре TIMx_CCMR1 (если фильтр не нужен, то остаётся IC2F=0000).
3. Выбрать полярность переднего фронта записью CC2P=0 в регистре TIMx_CCER.
4. Включить внешний режим 1 тактирования записью SMS=111 в регистре TIMx_SMCR.
5. Выбрать TI2 как источник сигнала записью TS=110 в регистре TIMx_SMCR.
6. Включить счётчик записью CEN=1 в регистре TIMx_CR1.

Предделитель захвата для запуска не используется, ну и бог с ним.

По переднему фронту на TI2 счётчик один раз тикает и ставит флаг TIF.

Задержка между передним фронтом на TI2 и реальным тиком появляется из-за схемы ресинхронизации на входе TI2.

Схема управления внешнего режима 1.

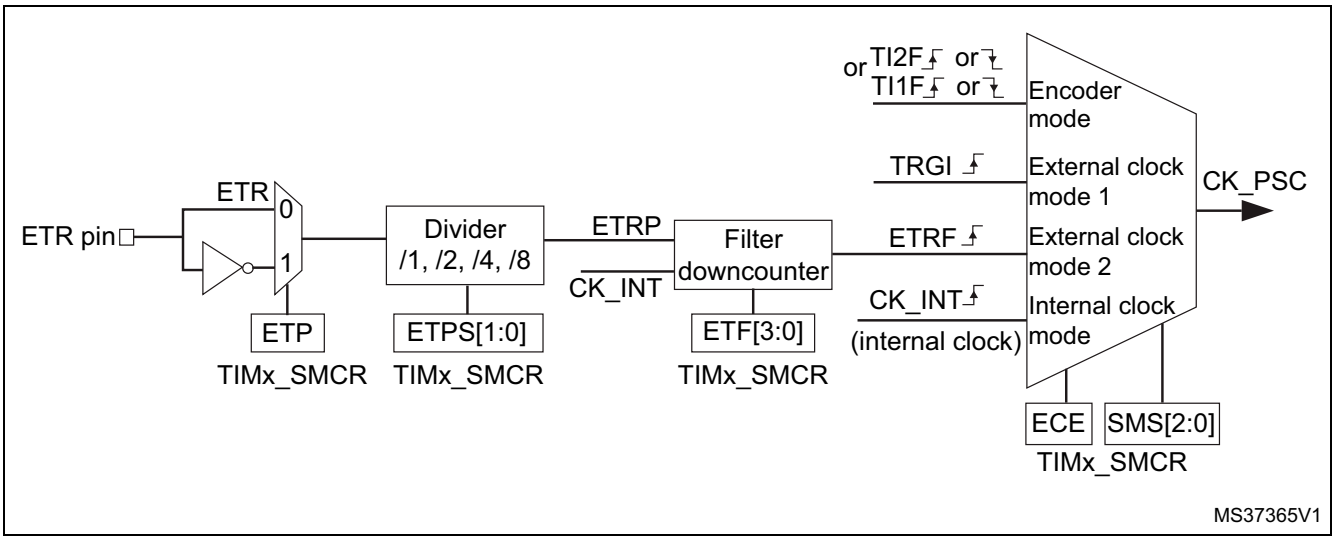


Внешний источник режим 2.

Включается записью ECE=1 в регистре TIMx_SMCR.

Счётчик работает по переднему или заднему фронту входа ETR.

Блок входа внешнего запуска.



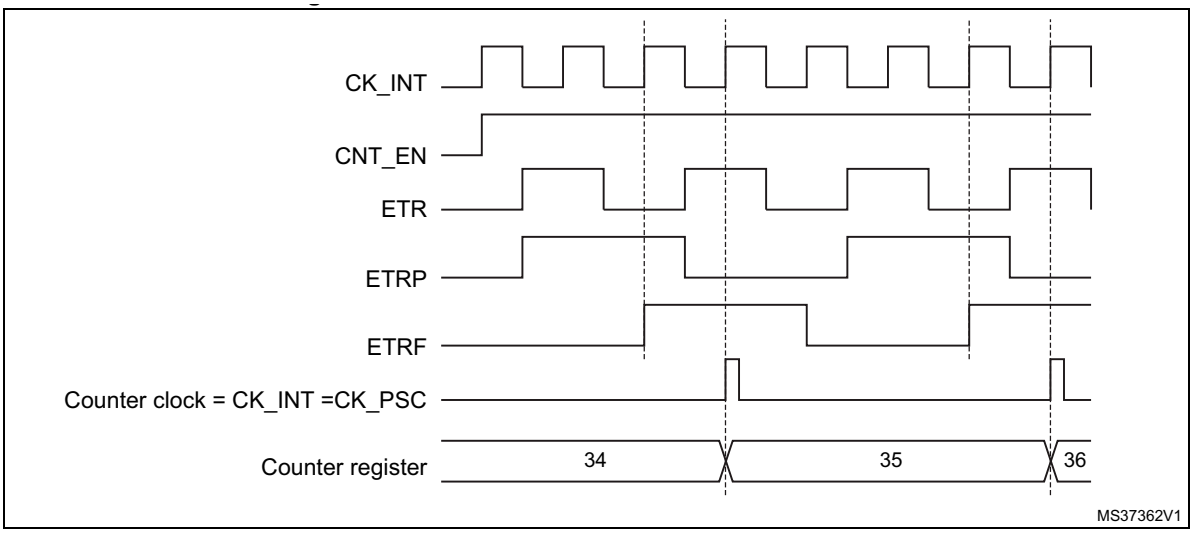
Например, включение режима прямого счёта по каждому 2 переднему фронту на входе ETR:

1. Здесь фильтр не нужен, так что ETF[3:0]=0000 в регистр TIMx_SMCR.
2. В предделитель пишем ETPS[1:0]=01 в регистре TIMx_SMCR
3. Выбираем передний фронт на ножке ETR записью ETP=0 в регистре TIMx_SMCR.
4. Разрешаем внешний режим 2 записью ECE=1 в регистре TIMx_SMCR.
5. Включаем счётчик записью CEN=1 в регистре TIMx_CR1.

Счётчик тикает единожды на каждые 2 передних фронта ETR.

Задержка между передним фронтом на ETR и реальным тиком появляется из-за схемы ресинхронизации на входе сигнала ETRP.

Схема управления с внешним источником режим 2.

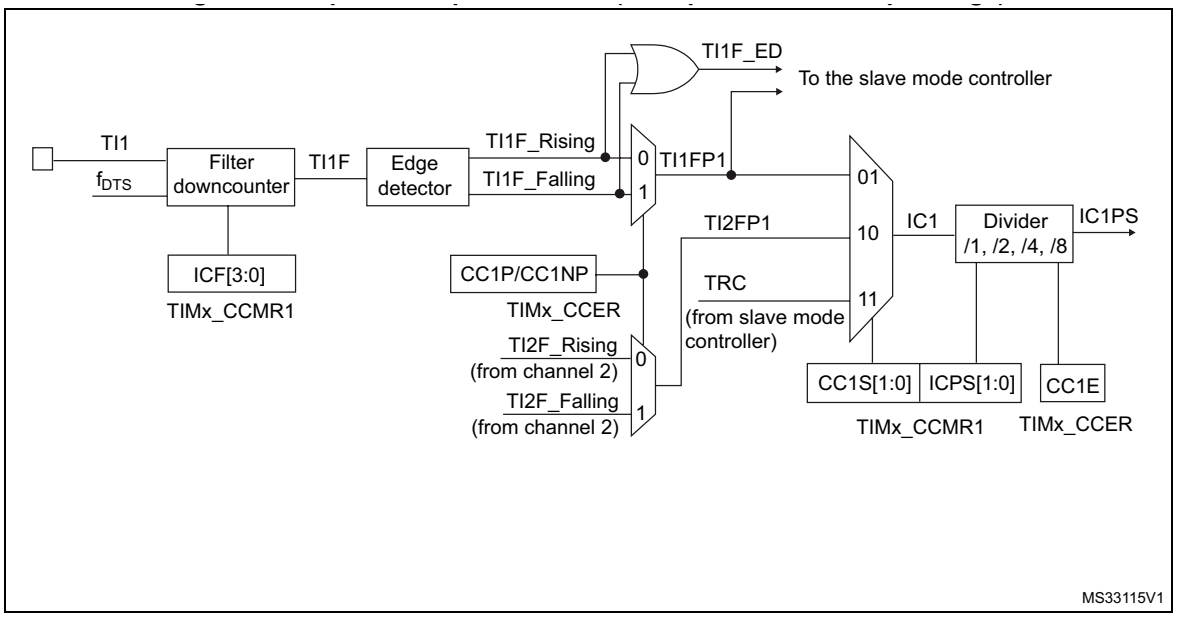


15.3.4. Каналы захвата/сравнения

Каждый канал захвата/сравнения построен вокруг парного регистра (включая невидимый), входного каскада захвата (с цифровым фильтром, мультиплексором и предделителем) и выходного каскада (с компаратором и управлением выходом).

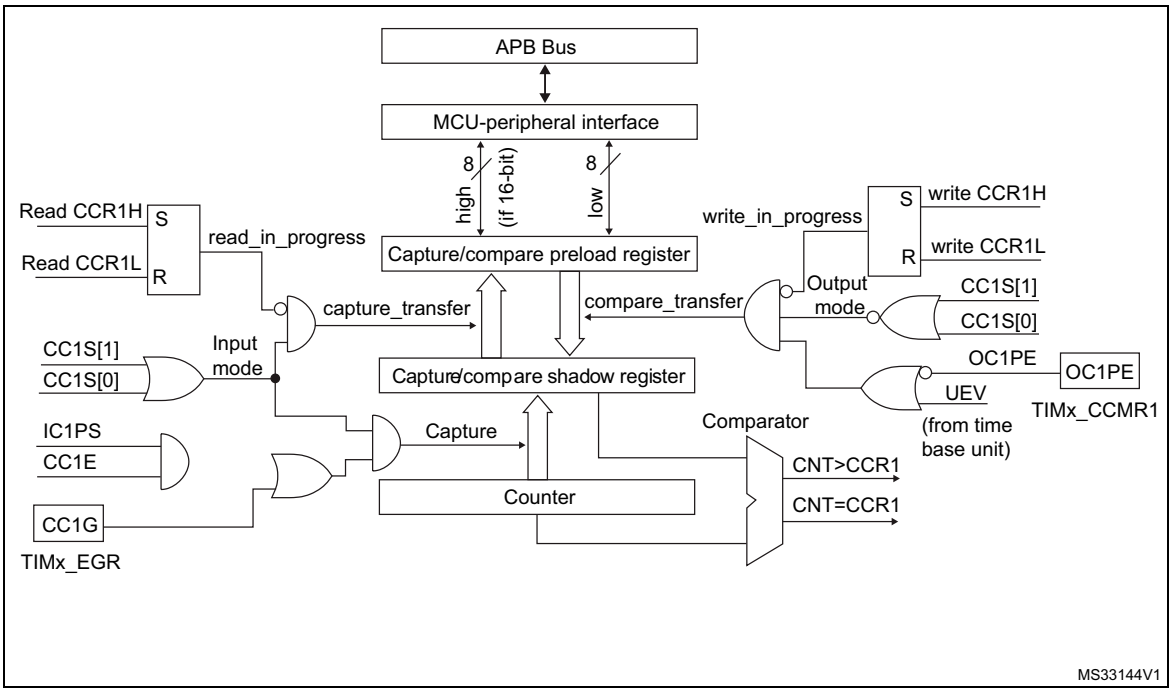
Входной каскад считывает нужный вход TIx для выдачи фильтрованного сигнала TIxF. Далее, детектор фронта с выбором полярности выдаёт сигнал (TIxFPx), что может быть использован как вход запуска контроллером режима ведомого или как команда захвата. Предделитель стоит до регистра захвата (ICxPS).

Входной каскад канала 1.

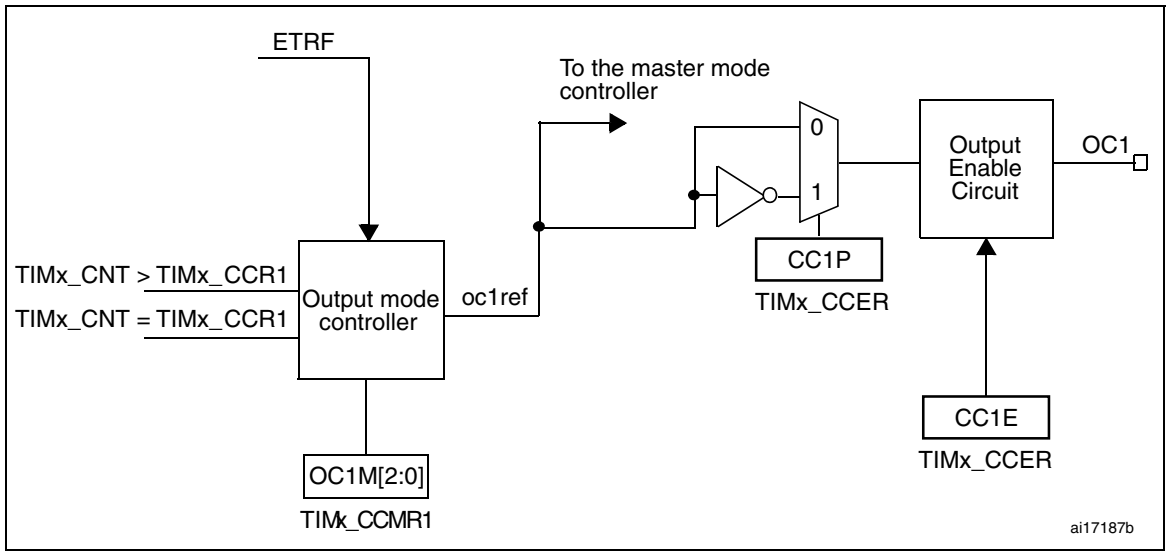


Выходной каскад выдаёт промежуточный сигнал, используемый как опорный: OCxRef (активный высокий). Полярность действует в конце цепочки.

Основная схема канала 1.



Выходной каскад канала захвата/сравнения 1.



Блок захвата/сравнения состоит из двух регистров: предзагрузки и невидимого. Снаружи доступен только регистр предзагрузки и по чтению и по записи.

Захват выполняется внутренним регистром и результат пишется в регистр предзагрузки.

При сравнении регистр предзагрузки пишется в невидимый, где и сравнивается со счётчиком.

15.3.5. Режим захвата входа

Регистры `TIMx_CCRx` защёлкивают содержимое счётчика после передачи, обнаруженной соответствующим сигналом `ICx`. При этом ставится флаг `CCXIF` в регистре `TIMx_SR` и ставятся запрос DMA и прерывания. Если захват происходит при стоящем флаге `CCxIF`, то ставится флаг перезахвата `CCxOF` в регистре `TIMx_SR`. Флаг `CCxIF` снимается записью в него нуля или чтением захваченных данных из регистра `TIMx_CCRx`. `CCxOF` снимается записью ‘`0`’.

Далее захватываем содержимое `TIMx_CCR1` по переднему фронту `TI1`. Делаем так:

- Выбираем активный вход: `TIMx_CCR1` подключаем в `TI1` и пишем "`01`" в биты `CC1S` регистра `TIMx_CCMR1`. Биты `CC1S` отличаются от `00`, т. е. канал на вводе и регистр `TIMx_CCR1` доступен только по чтению.
- Ставим длительность входного фильтра битами `ICxF` в регистре `TIMx_CCMRx` с учётом входного сигнала на `TIx`. Допустим, что при переключении входной сигнал нестабилен в течении пяти внутренних тактов. Значит длительность фильтра должна быть больше этих пяти тактов.

Мы можем определить передачу на TI1 за 8 последовательных выборок нового уровня на частоте f_{DTS}. И мы пишем 0011 в биты IC1F регистра TIMx_CCMR1.

- Выбираем фронт сигнала на TI1 записью 0 в бит CC1P регистра TIMx_CCER (передний).
- Ставим предделитель. Здесь он нам не нужен (пишем 00 в биты IC1PS регистра TIMx_CCMR1).
- Разрешаем запись счётчика в регистр захвата установкой бита CC1E в TIMx_CCER.
- Если нужно разрешаем выдачу запросов прерывания битом CC1IE в TIMx_DIER и DMA битом CC1DE в регистре TIMx_DIER.

Если захват входа произошёл, то:

- Регистр TIMx_CCR1 получает значение счётчика активной передачи.
- Ставится флаг прерывания CC1IF. Если он ещё стоял, то ставится и флаг CC1OF.
- В зависимости от бита CC1IE генерируется прерывание.
- В зависимости от бита CC1DE генерируется запрос DMA.

Чтобы обработать перезахват рекомендуется читать регистр данных до опроса флага перезахвата.

NB: Запросы прерывания и DMA IC можно выдавать программно установкой битов CCxG в регистре TIMx_EGR.

15.3.6. Режим ввода ШИМ

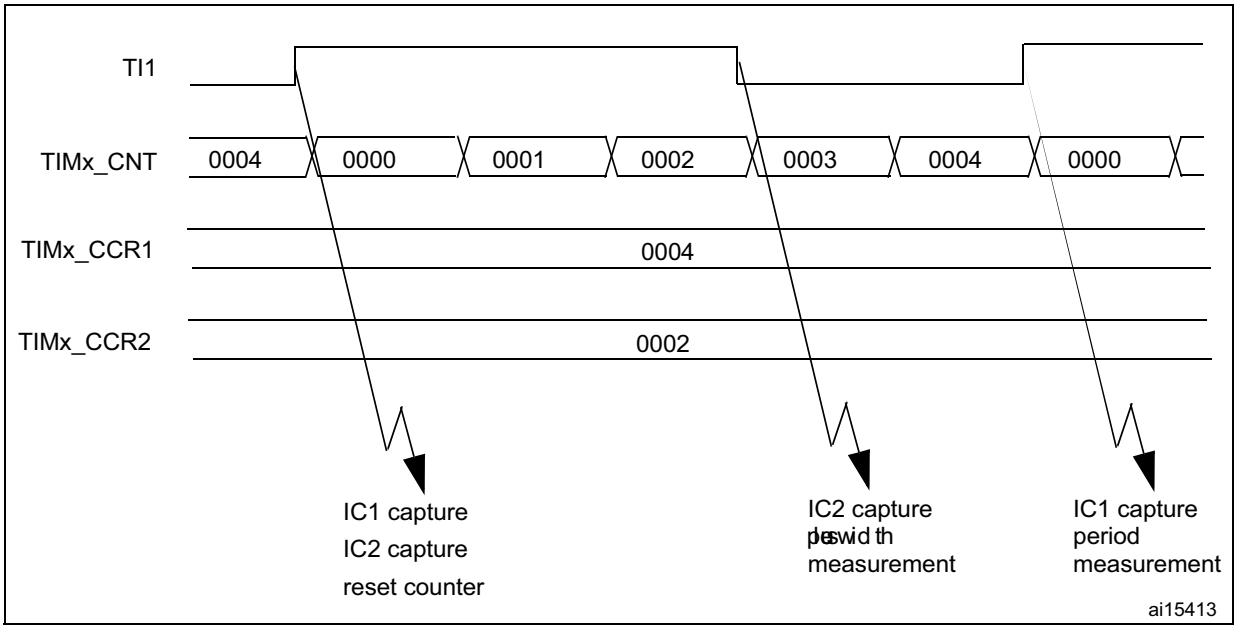
Это часть режима захвата входа. Процедура такая же, кроме:

- Два сигнала ICx направляются на на один вход TIx.
- Полярность этих входов ICx противоположная.
- Сигналом запуска берётся один из двух сигналов TIxFP, а контроллер ведомого режима выключается.

Например, можно измерить период (в регистре TIMx_CCR1) и длительность заполнения (в регистре TIMx_CCR2) ШИМ сигнала на TI1 следующей процедурой (зависит от частоты CK_INT и предделителя):

- Выбрать активный вход для TIMx_CCR1: Записать 01 в биты CC1S регистра TIMx_CCMR1 (TI1).
- Выбрать полярность для TI1FP1 (для захвата в TIMx_CCR1 и очистки счётчика): записать 0 в CC1P (передний фронт).
- Выбрать активный вход для TIMx_CCR2: записать 10 в биты CC2S регистра TIMx_CCMR1 (TI1).
- Выбрать полярность для TI1FP2 (для захвата в TIMx_CCR2): записать 1 в CC2P (задний фронт).
- Выбрать вход запуска: записать 101 в биты TS регистра TIMx_SMCR (TI1FP1).
- Поставить контроллер режима ведомого в сброс: записать 100 в биты SMS регистра TIMx_SMCR.
- Включить захват: поставить биты CC1E и CC2E в регистре TIMx_CCER.

Времянка режима ввода ШИМ.



1. Режим ШИМ может использоваться только с сигналами TIMx_CH1/TIMx_CH2 так как к контроллеру режима ведомого подключены только TI1FP1 и TI2FP2.

15.3.7. Принудительный вывод

В режиме вывода (биты CCxS = 00 в TIMx_CCMRx), каждый выходной сигнал сравнения (OCxREF и затем OCx/OCxN) можно программно поставить в активное или неактивное состояние, независимо от результата сравнения выходного регистра сравнения и счётчика.

Активными выходные сигналы (OCxREF/OCx) ставятся записью 101 в биты OCxM нужного регистра TIMx_CCMRx. Активный сигнал OCxREF всегда высокий, а OCx противоположен биту полярности CCxP.

Пример: CCxP=0 (OCx активный высокий) => OCx ставится высоким.

Сигнал OCxREF снимается записью 100 в биты OCxM регистра TIMx_CCMRx.

Но сравнение внутреннего регистра TIMx_CCRx со счётчиком всё равно выполняется, биты ставятся, генерируются запросы прерывания и DMA. См. ниже.

15.3.8. Режим сравнения выхода

Используется для управления выходным сигналом и определения истечения периода времени.

При совпадении регистра захвата/сравнения со счётчиком эта схема:

- Ставит выходную ножку в заданное состояние (биты OCxM в регистре TIMx_CCMRx) и полярность (бит CCxP в регистре TIMx_CCER). Ножка может хранить состояние(OCxM=000), стать активной (OCxM=001), неактивной (OCxM=010) или переключиться (OCxM=011).
- Ставит флаг прерывания (бит CCxIF в регистре TIMx_SR).
- При установленной маске прерывания (бит CCxIE в регистре TIMx_DIER) выдаёт запрос.
- При установленном разрешении DMA (бит CCxDE в регистре TIMx_DIER) выдаёт нужный (бит CCDS в регистре TIMx_CR2) запрос.

В регистр TIMx_CCRx можно писать как через регистр предзагрузки, так и без него (см. бит OCxPE в регистре TIMx_CCMRx).

В режиме сравнения выхода бит события UEV на выходы OCxREF и OCx не влияет. Разрешение составляет один счёт счётчика. Этот режим можно использовать для выдачи одного импульса.

Процедура:

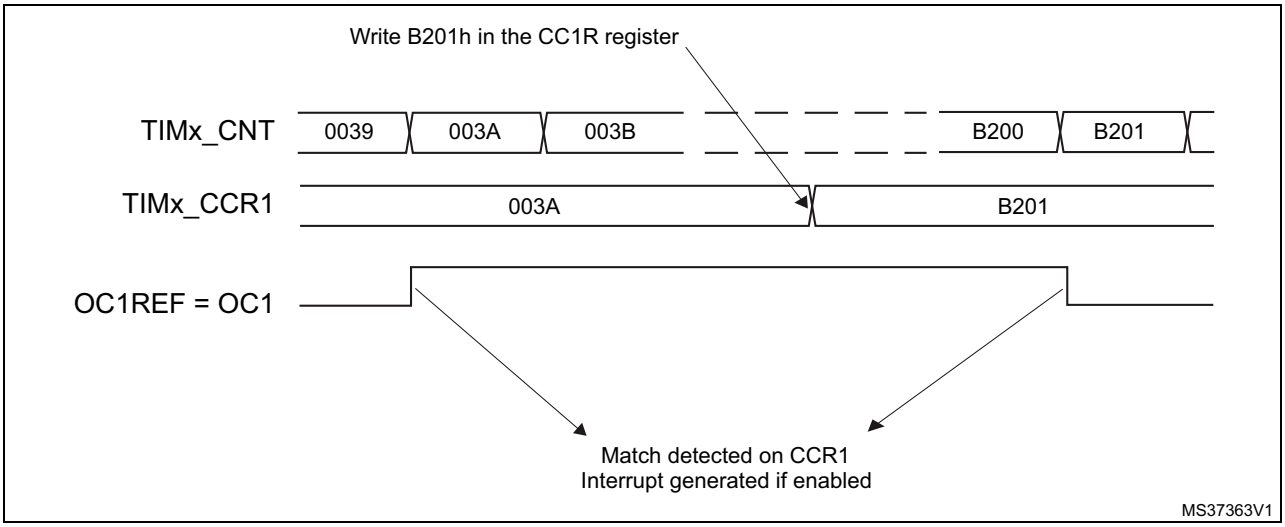
1. Выбрать такты счётчика (внутренние, внешние, предделитель).
2. Записать желаемое в регистры TIMx_ARR и TIMx_CCRx.
3. Если нужно прерывание, то установить бит CCxIE.
4. Выбрать режим выхода. Например:

- Записать OCxM = 011 для переключения ножки OCx при совпадении CNT и CCRx
- Записать OCxPE = 0 для отключения регистра предзагрузки
- Записать CCxP = 0 ради высокого активного уровня
- Записать CCxE = 1 для разрешения вывода

5. Включить счётчик установкой бита CEN в регистре TIMx_CR1.

При запрещённом регистре предзагрузки (OCxPE=’0’) регистр TIMx_CCRx можно обновлять в любое время, иначе он будет обновлён только по событию UEV.

Режим сравнения выхода, переключение ножки OC1.



15.3.9. Режим ШИМ

Частота сигнала ШИМ определяется регистром TIMx_ARR, а коэффициент заполнения регистром TIMx_CCRx.

Режим ШИМ может включаться независимо по одному на каждый выход OCx записью ‘110’ (ШИМ1) или ‘111’ (ШИМ2) в биты OCxM регистра TIMx_CCMRx. Соответствующий регистр предзагрузки должно включать битом OCxPE в регистре TIMx_CCMRx и, соответсвенно, регистр авто-загрузки (в режимах прямого и реверсивного счёта) нужно включить битом ARPE в регистре TIMx_CR1.

Поскольку регистр предзагрузки пишется в теневой регистр только по событию обновления, то все регистры нужно заранее инициализировать установкой бита UG в регистре TIMx_EGR.

Полярность OCx (активный высокий или низкий) выбирается битом CCxP в регистре TIMx_CCER. Выходы OCx включаются битами CCxE, CCxNE, MOE, OSSI и OSSR в регистрах TIMx_CCER и TIMx_BDTR.

В режиме ШИМ (1 или 2), TIMx_CNT и TIMx_CCRx сравниваются на выполнение условия TIMx_CCRx ≤ TIMx_CNT или TIMx_CNT ≤ TIMx_CCRx (в зависимости от направления счёта).

Но для совместимости с ETRF (OCREF очищается внешним сигналом ETR до следующего периода ШИМ), сигнал OCREF выставляется только:

- когда изменяется результат сравнения, или
- при переключении сравнения выхода (биты OCxM в регистре TIMx_CCMRx register) из “заморозки” (без сравнения, OCxM=‘000) в один из режимов ШИМ (OCxM=‘110 или ‘111).

Это программное управление ШИМ при работающем счётчике.

Бит CMS в регистре TIMx_CR1 определяет генерацию ШИМ по фронту или по центру.

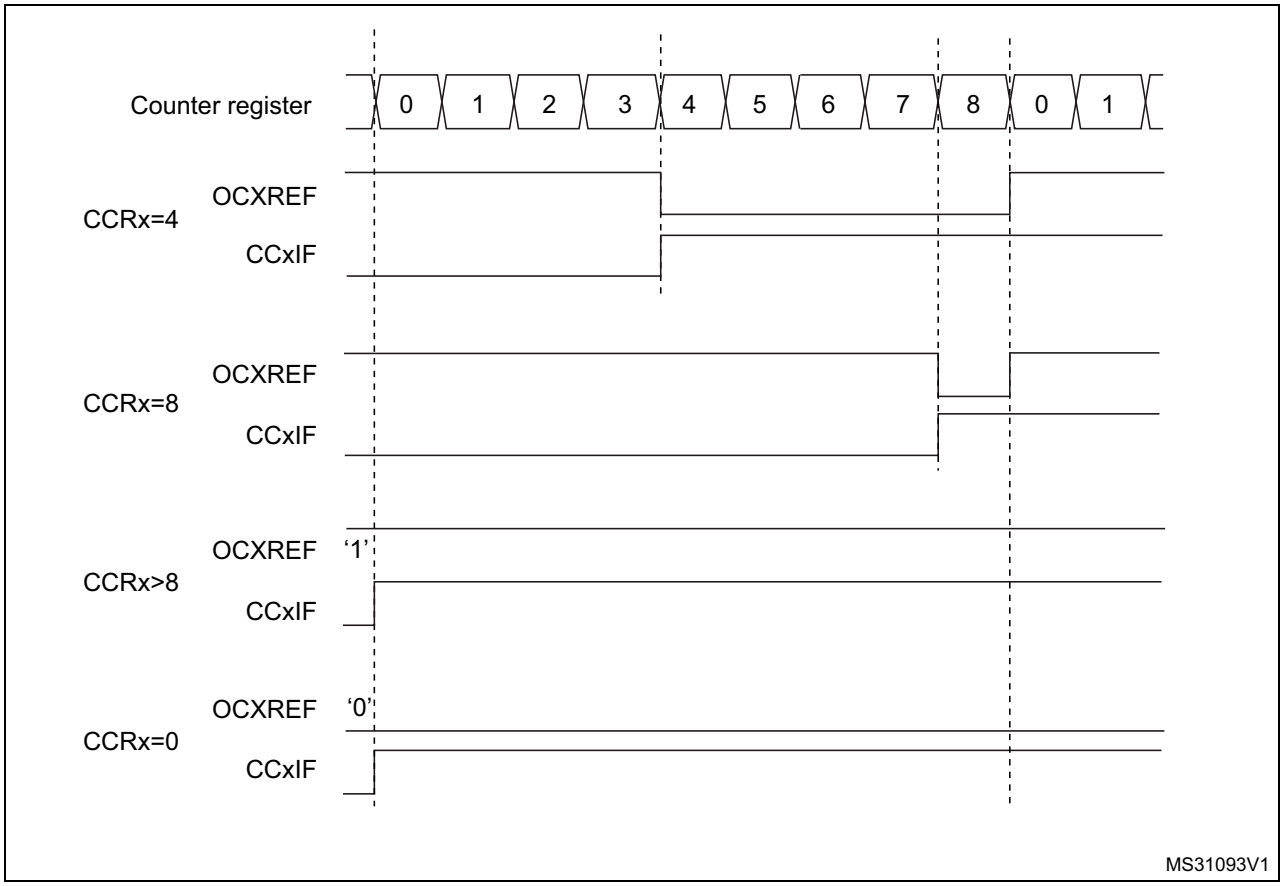
ШИМ по фронту

Прямой счёт

Он определяется сброшенным битом DIR в регистре TIMx_CR1.

В примере ниже стоит режим ШИМ1. Опорный сигнал OCxREF остаётся высоким при TIMx_CNT < TIMx_CCRx. Если регистр TIMx_CCRx больше значения авто-загрузки (TIMx_ARR) то OCxREF равен ‘1’. При равенстве значений сравнения, OCxREF равен ‘0’. В примере ниже TIMx_ARR=8.

ШИМ по фронту (`TIMx_ARR=8`).



Обратный счёт

Он определяется стоящим битом `DIR` в регистре `TIMx_CR1`.

В режиме ШИМ1 опорный сигнал `OCxREF` остаётся низким при `TIMx_CNT` > `TIMx_CCRx`. Если регистр `TIMx_CCRx` больше значения авто-загрузки (`TIMx_ARR`) то `OCxREF` равен '1'. При равенстве значений сравнения, `OCxREF` равен '1'. 0% ШИМ в этом режиме недоступен.

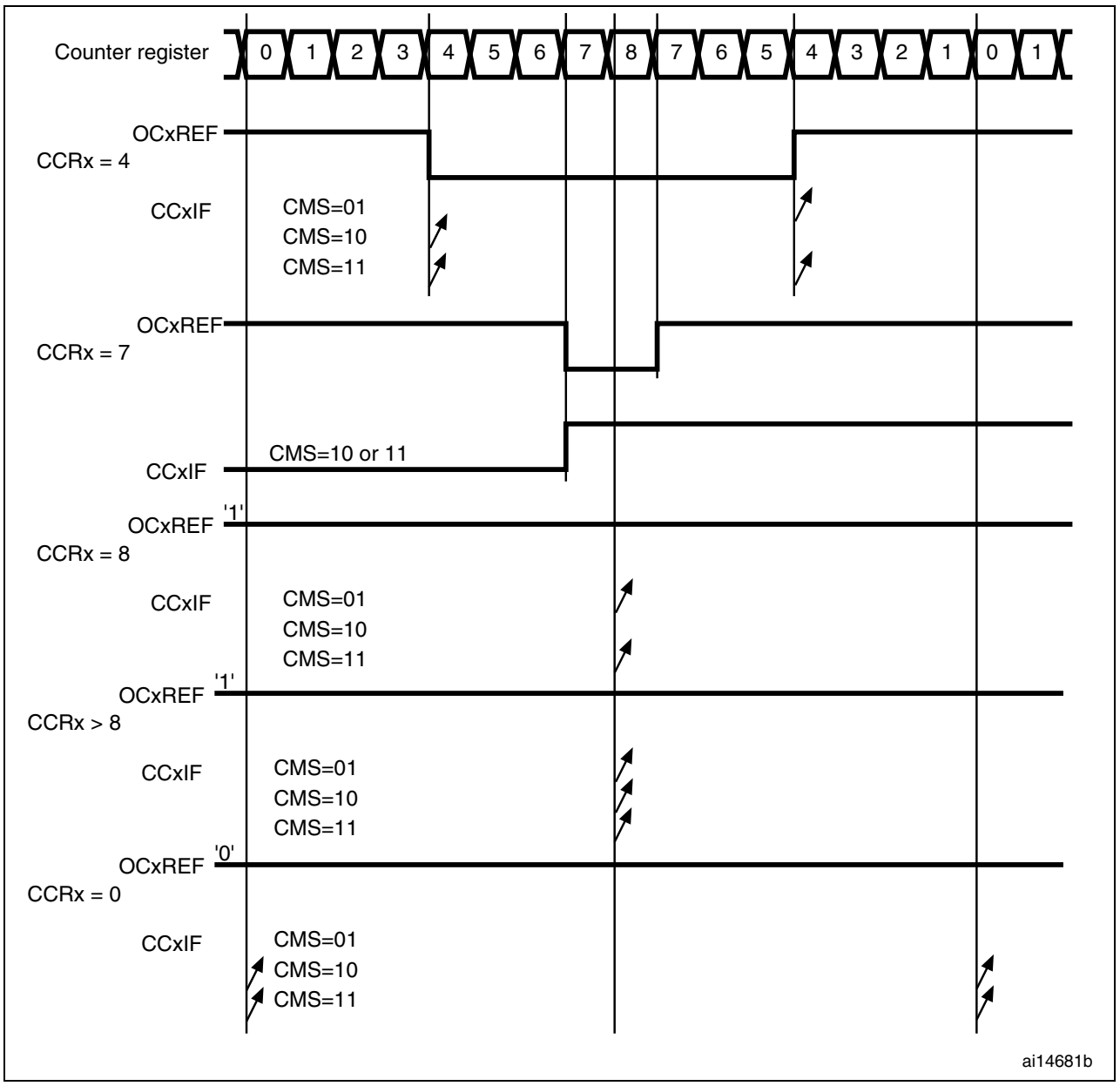
ШИМ по центру

Этот режим включается когда биты `CMS` в регистре `TIMx_CR1` не равны '00' (остальные конфигурации на сигналы `OCxREF`/`OCx` влияют также). Флаг сравнения ставится в любых режимах счёта (прямом, обратном и реверсивном) в зависимости от битов `CMS` в регистре `TIMx_CR1`. Бит направления (`DIR`) в регистр `TIMx_CR1` меняется аппаратно и не дай бог менять его программно.

В примере ниже:

- `TIMx_ARR`=8,
- Стоит режим ШИМ1,
- Флаг ставится при обратном счёте в режиме ШИМ1 по центру (`CMS`=01 в `TIMx_CR1`).

ШИМ по центру (TIMx_ARR=8).



Советы:

- При старте режима по центру используется текущее направление счёта (зависит от бита DIR в регистре TIMx_CR1). Более того, в это время биты DIR и CMS программно менять нельзя.
- Запись в счётчик работающего режима ой как не рекомендуется. В частности:
  - При TIMx_CNT>TIMx_ARR направление счёта не изменяется и продолжает считать в том же направлении.
  - При записи 0 или изменении TIMx_ARR направление изменяется, но событие UEV не выдаётся.
- Самый безопасный способ это программно запустить обновление (битом UG в регистре TIMx_EGR) прямо перед стартом, а не писать в работающий счётчик.

15.3.10.Режим одного импульса (OPM)

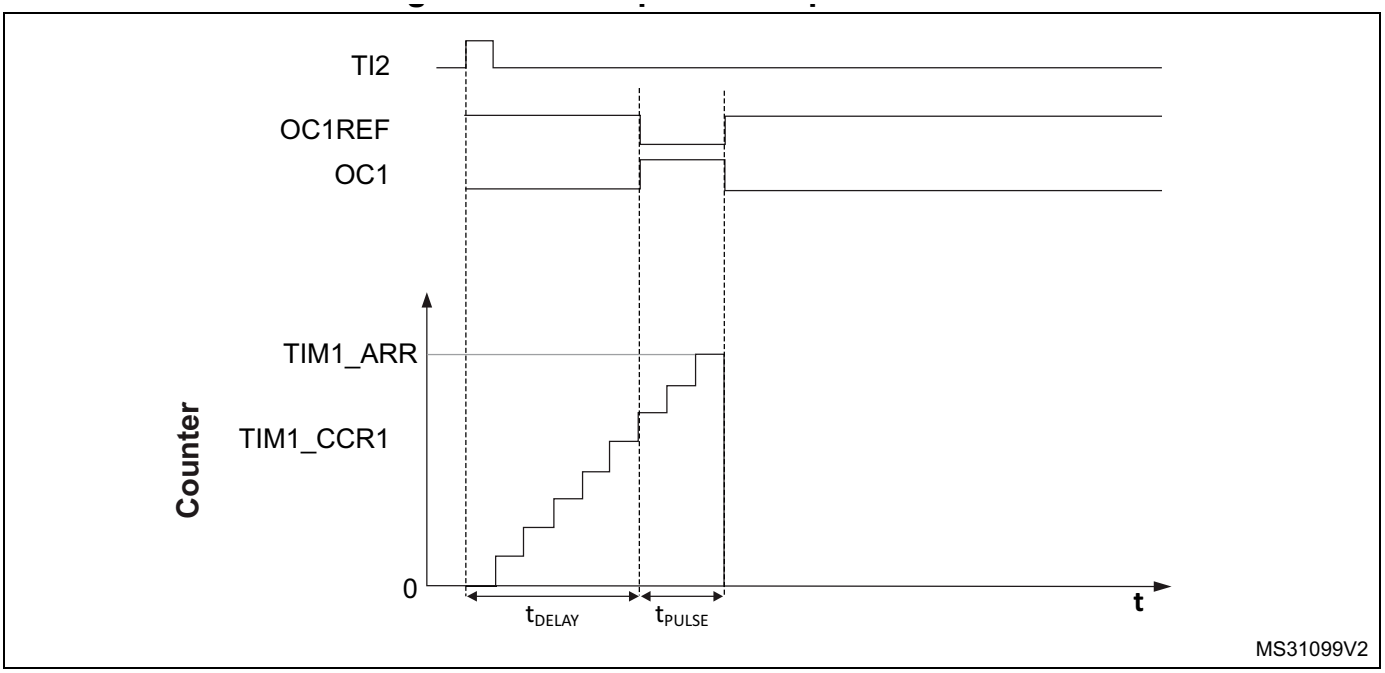
OPM это частный случай предыдущих режимов. Он позволяет счётчику запускаться по сигналу и генерировать импульс программируемой длины с программируемой задержкой.

Запускать счётчик можно из контроллера ведомого режима. Создавать сигнал можно в режиме сравнения выхода и ШИМ. Режим включается установкой бита OPM в регистре TIMx_CR1. Так счётчик автоматически останавливается по следующему событию UEV.

Импульс генерируется правильно только если значение сравнения отличается от начального значения счётчика. Перед стартом (таймер ждёт запуска) конфигурация должна быть:

- При прямом счёте: CNT < CCRx ≤ ARR (в частности, 0 < CCRx)
- При обратном счёте: CNT > CCRx

Режим одного импульса.



Например, выдадим положительный импульс на OC1 длиной t_{PULSE} после задержки t_{DELAY} по переднему фронту на ножке TI2.

Берём TI2FP2 как запуск 1:

- Поставим TI2FP2 на TI2 записью CC2S='01' в регистре TIMx_CCMR1.
- TI2FP2 ставим на передний фронт записью CC2P='0' в регистре TIMx_CCER.
- Выбираем TI2FP2 как запуск контроллером режима ведомого (TRGI) записью TS='110' в регистр TIMx_SMCR.
- TI2FP2 используем как старт записью SMS='110' в TIMx_SMCR (режим запуска).

Сигнал OPM определяем записью регистров сравнения (учитываем частоту тактов и предделитель).

- Определяем t_{DELAY} записью в TIMx_CCR1.
- Ставим t_{PULSE} как разность значений перезагрузки и сравнения (TIMx_ARR - TIMx_CCR1+1).
- Сигнал создаётся переходом из '0' в '1' при сравнении и переходом из '1' в '0' при достижении счётчиком значения перезагрузки. Для этого, включаем режим ШИМ2 записью OC1M=111 в регистре TIMx_CCMR1. Регистры предзагрузки можно включить записью OC1PE='1' в регистре TIMx_CCMR1и ARPE в регистре TIMx_CR1, а можно не включать. В этом случае значение сравнения пишется в регистр TIMx_CCR1, значение перезагрузки в регистр TIMx_ARR, даём событие обновления битом UG и ждём внешнего запуска на TI2. В CC1P здесь пишется '0'.

В этом примере биты DIR и CMS регистра TIMx_CR1 должны быть сброшены.

Мы ждём только одного импульса (Однократный режим), так что для остановки счётчика по следующему событию обновления (счётчик переходит через значение перезагрузки и обнуляется) нужно поставить бит OPM регистра TIMx_CR1. Если бит OPM регистра TIMx_CR1 обнулён, то включается Повторяющийся режим.

Частный случай: быстрое включение OCx:

В этом режиме появление фронта на входе TIx ставит бит CEN, чем включает счётчик. Затем совпадение счётчика с значением сравнения переключает выходной сигнал. Но для этого нужны несколько тактов, ограничивая минимально возможное значение t_{DELAY}.

Если нужен сигнал с минимальной задержкой, то нужно ставить бит OCxFE в TIMx_CCMRx. Далее включаются OCxREF (и OCx) не глядя на сравнение. Его новый уровень равен уровню при сравнении. OCxFE работает только в режимах ШИМ1 и ШИМ2.

15.3.11.Очистка OCxREF по внешнему событию

Сигнал OCxREF для заданных каналов можно сделать низким, подав высокий уровень на вход ETRF (бит разрешения OCxCE в TIMx_CCMRx стоит в ‘1’). Он остаётся низким вплоть до следующего события UEV.

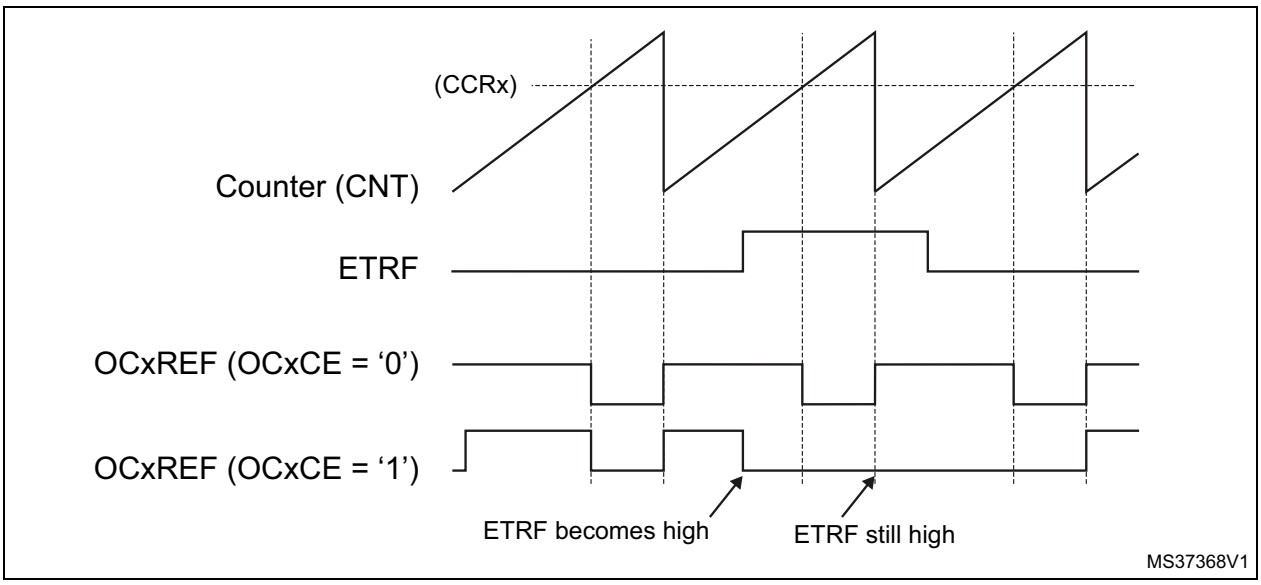
Эта функция работает в режимах сравнения выхода и ШИМ, но не в принудительном режиме.

Например, сигнал ETR можно подключить к выходу компаратора для управления током. В этом случае ETR нужно конфигурировать так:

1. Выключить предделитель внешнего запуска: записать ’00’ в биты ETPS[1:0] в TIMx_SMCR.
2. Выключить режим 2 внешнего тактирования: снять бит ECE в TIMx_SMCR.
3. Поставить полярность и фильтр внешнего запуска (ETP и ETF) как нужно.

В примере ниже приведён сигнал OCxREF при переходе входа ETRF в высокий уровень для обоих значений OCxCE. Здесь TIMx в режиме ШИМ.

Очистка OCxREF.



15.3.12.Режим интерфейса кодера

Режим интерфейса кодера выбирается записью SMS=‘001’ в TIMx_SMCR при счёте по фронтам TI2, SMS=’010’ по фронтам TI1 и SMS=’011’ по фронтам TI1 и TI2.

Полярность TI1 и TI2 выбирается битами CC1P и CC2P в регистре TIMx_CCER. Если нужно, то можно использовать входной фильтр.

Для работы с инкрементным кодером используются входы TI1 и TI2. Счётчик тактируется каждой достоверной передачей по TI1FP1 или TI2FP2 (TI1 и TI2 после входного фильтра и выбора полярности, без них TI1FP1= TI1 и TI2FP2= TI2) при стоящем бите CEN в TIMx_CR1. Последовательность сигналов с двух входов создаёт импульсы счёта и сигнал направления. Счёт может быть прямым и обратным, бит DIR в TIMx_CR1 изменяется аппаратно. Бит DIR вычисляется на каждый входной сигнал (TI1 и/или TI2).

Режим интерфейса кодера просто работает по внешним тактам с выбором направления. То есть счётчик непрерывно считает туда-сюда между 0 и перезагружаемым значением в регистре TIMx_ARR. Так что TIMx_ARR нужно писать до старта. Таким способом захват, сравнение, предделитель и запуск вывода продолжают нормально. Режим кодера и Режим внешних тактов 2 несовместимы.

В этом режиме счётчик автоматически следует за скоростью и направлением инкрементного кодера и его содержимое всегда представляет позицию кодера. Направление счёта соответствует направлению вращения подключённого датчика.

Далее приведены возможные комбинации, полагая, что TI1 и TI2 не изменяются одновременно.

Направления счёта.

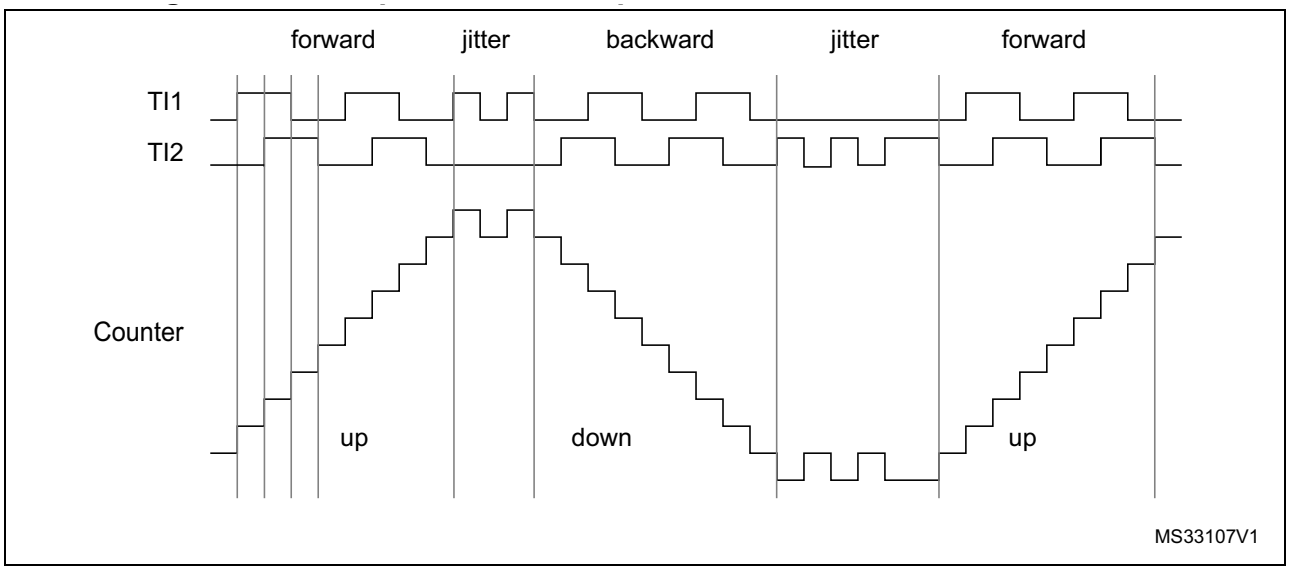
Активный сигнал	Уровень оппозитного сигнала (TI1FP1 для TI2, TI2FP2 для TI1)	Фронт TI1FP1		Фронт TI2FP2	
		Передний	Задний	Передний	Задний
Только TI1	Высокий	Обратный	Прямой	Стоит	Стоит
	Низкий	Прямой	Обратный	Стоит	Стоит
Только TI2	Высокий	Стоит	Стоит	Прямой	Обратный
	Низкий	Стоит	Стоит	Обратный	Прямой
TI1 и TI2	Высокий	Обратный	Прямой	Прямой	Обратный
	Низкий	Прямой	Обратный	Обратный	Прямой

Внешний инкрементный кодер может напрямую подключаться к MCU без дополнительной логики. Но обычно для преобразования дифференциального сигнала в цифровой используются компараторы. Это сильно увеличивает помехозащищённость. Третий выход кодера, показывающий механическую нулевую позицию, можно подключить к входу внешнего прерывания и сбросу счётчика.

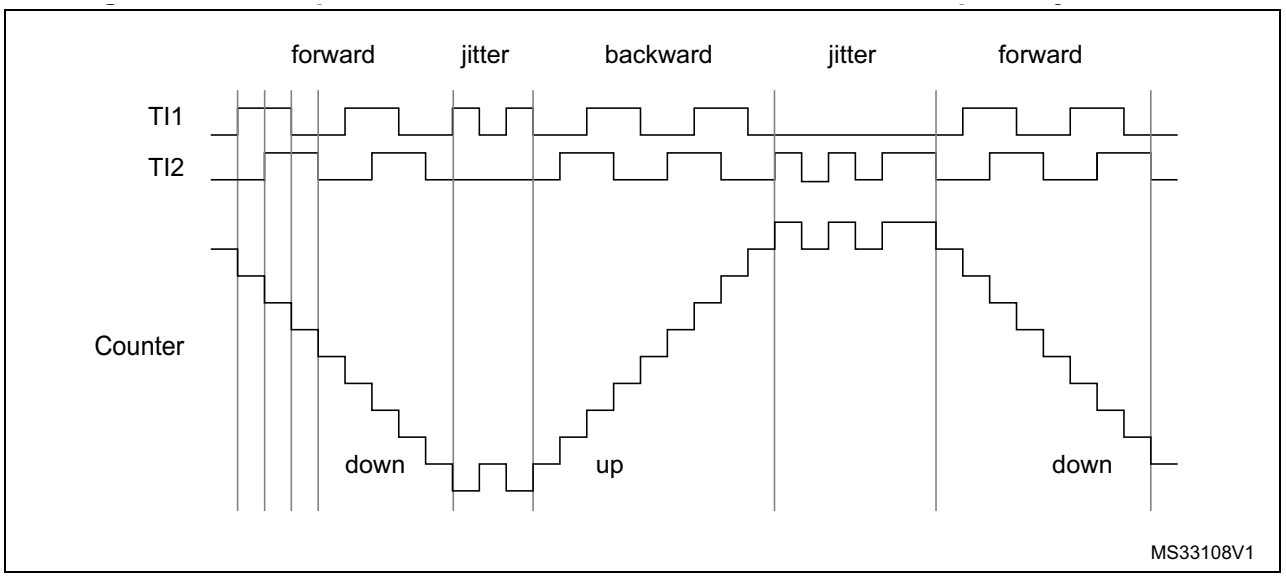
Пример работы счётчика показывает генерацию сигнала счёта и направления. Также показано как компенсируется дребезг на обоих выбранных фронтах. Он может появиться при нахождении датчика возле точек переключения. Использована такая конфигурация:

- CC1S=’01’ (TIMx_CCMR1, TI1FP1 на TI1).
- CC2S=’01’ (TIMx_CCMR2, TI1FP2 на TI2).
- CC1P=’0’, and IC1F = ‘0000’ (TIMx_CCER, TI1FP1 без инверсии, TI1FP1=TI1).
- CC2P=’0’, and IC2F = ‘0000’ (TIMx_CCER, TI1FP2 без инверсии, TI1FP2= TI2).
- SMS=’011’ (TIMx_SMCR, оба входа по обоим фронтам).
- CEN=’1’ (TIMx_CR1, счётчик включён).

Работа счётчика.



Работа счётчика с инверсным TI1FP1 и CC1P=’1’.



Таймер в режиме интерфейса кодера показывает текущую позицию счётчика. С помощью второго таймера в режиме захвата, измеряя время между двумя событиями кодера, можно получить динамическую информацию (скорость, ускорение, торможение). Для этого можно использовать третий выход кодера, показывающего механический ноль. В зависимости от времени между двумя событиями счётчик можно читать и регулярно. Это делается регулярной записью счётчика в третий входной регистр (если есть) периодическим сигналом от другого таймера. Можно читать и запросом DMA от часов реального времени.

15.3.13.Функция XOR входов таймера

Бит TI1S в регистре TIMx_CR2 позволяет подключать к входу канала 1 выход вентиля XOR, сочетая три входа TIMx_CH1, TIMx_CH2 and TIMx_CH3.

Выход XOR можно использовать со всеми входными функциями таймера, вроде запуска или захвата.

15.3.14.Синхронизация внешнего запуска TIMx

Есть три режима синхронизации TIMx: Сброс, Вентильный и Запуск.

Режим ведомого: Сброс

Счётчик и предделитель можно инициализировать по входу запуска. Кроме того, если бит URS в TIMx_CR1 сброшен, то генерируется событие UEV. Далее обновляются все предзагружаемые регистры (TIMx_ARR, TIMx_CCRx).

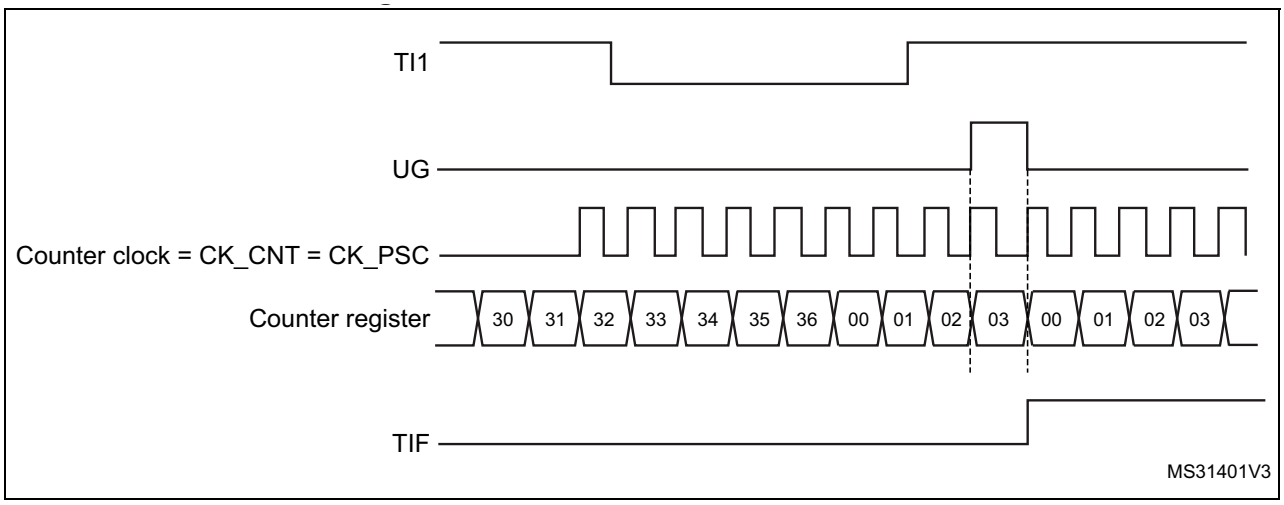
В этом примере счётчик прямого счёта сбрасывается по переднему фронту на входе TI1:

- Ставим канал 1 на передний фронт TI1. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC1S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC1P=0 в регистре TIMx_CCER определяет полярность и передний фронт.
- Ставим таймер в режим сброса записью SMS=100 в регистр TIMx_SMCR. Выбираем вход с TI1 записью TS=101 в регистре TIMx_SMCR.
- Запускаем счётчик записью CEN=1 в регистре TIMx_CR1.

Счётчик начинает считать внутренние такты вплоть переднего фронта TI1. Тогда счётчик сбрасывается с 0, ставится флаг запуска (бит TIF в регистре TIMx_SR) выдаются запросы прерывания и DMA (если разрешены битами TIE и TDE в регистре TIMx_DIER).

На рисунке ниже регистр перезагрузки TIMx_ARR=0x36. Задержка между фронтом TI1 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI1.

Режим Сброса.



Режим ведомого: Вентильный

Счётчик включается уровнем на выбранном входе.

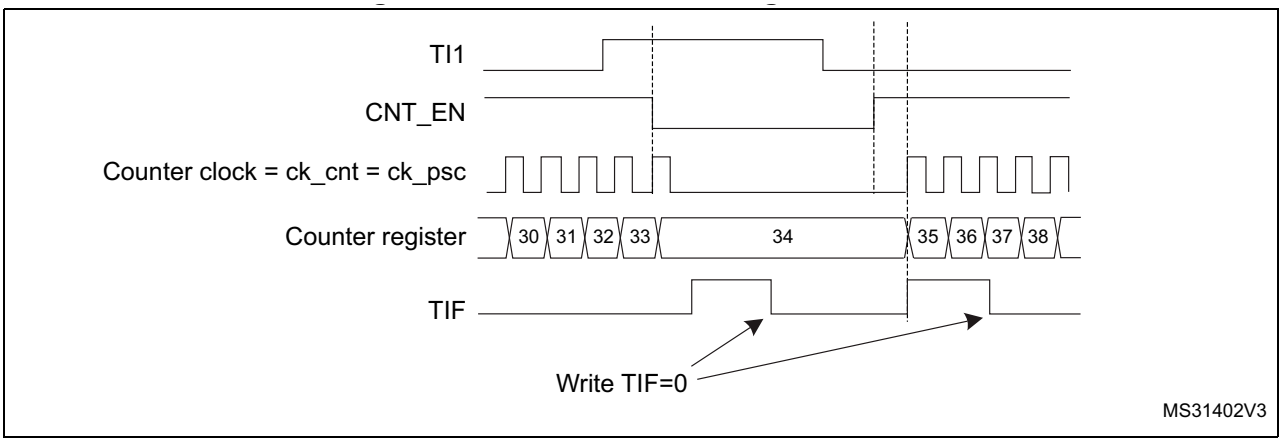
В этом примере счётчик прямого счёта работает только при низком уровне TI1:

- Ставим канал 1 на низкий уровень TI1. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC1S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC1P=1 в регистре TIMx_CCER определяет полярность и низкий уровень.
- Ставим таймер в вентильный режим записью SMS=101 в регистр TIMx_SMCR. Выбираем вход с TI1 записью TS=101 в регистре TIMx_SMCR.
- Запускаем счётчик записью CEN=1 в регистре TIMx_CR1. В этом режиме счётчик не работает при CEN=0, независимо от уровня на входе.

Счётчик начинает считать внутренние такты при низком уровне на TI1 и останавливается при высоком. Флаг запуска (бит TIF в регистре TIMx_SR) ставится при пуске и остановке таймера.

Задержка между фронтом TI1 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI1.

Вентильный режим.



Режим ведомого: Запуск

Счётчик включается событием на выбранном входе.

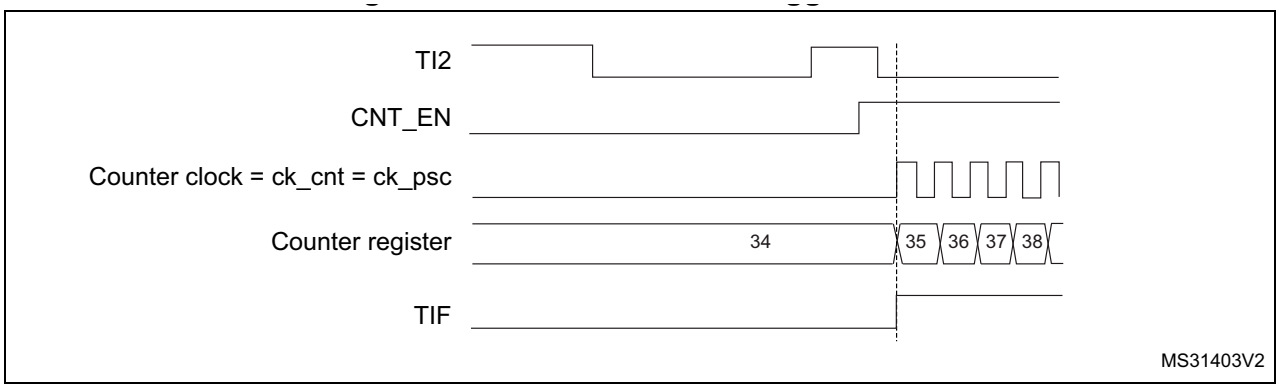
В этом примере счётчик прямого счёта работает по переднему фронту на TI2:

- Ставим канал 2 на передний фронт TI2. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC2S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC2P=1 в регистре TIMx_CCER определяет полярность и низкий уровень.
- Ставим таймер в режим запуска записью SMS=110 в регистр TIMx_SMCR. Выбираем вход с TI2 записью TS=110 в регистре TIMx_SMCR.

Счётчик начинает считать внутренние такты по переднему фронту на TI2 и ставит флаг запуска (бит TIF в регистре TIMx_SR).

Задержка между фронтом TI2 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI2.

Режим Запуска.



Режим ведомого: Внешние такты 2 + Запуск

К режимам ведомого (кроме Внешних тактов 1 и Кодера) можно добавить режим внешнего тактирования 2. В этом случае сигнал ETR выбирается для внешних тактов, а по другому входу можно подавать запуск (в режимах Сброса, Вентильном или Запуска). Не рекомендуем выбирать ETR как TRGI битами TS в регистре TIMx_SMCR.

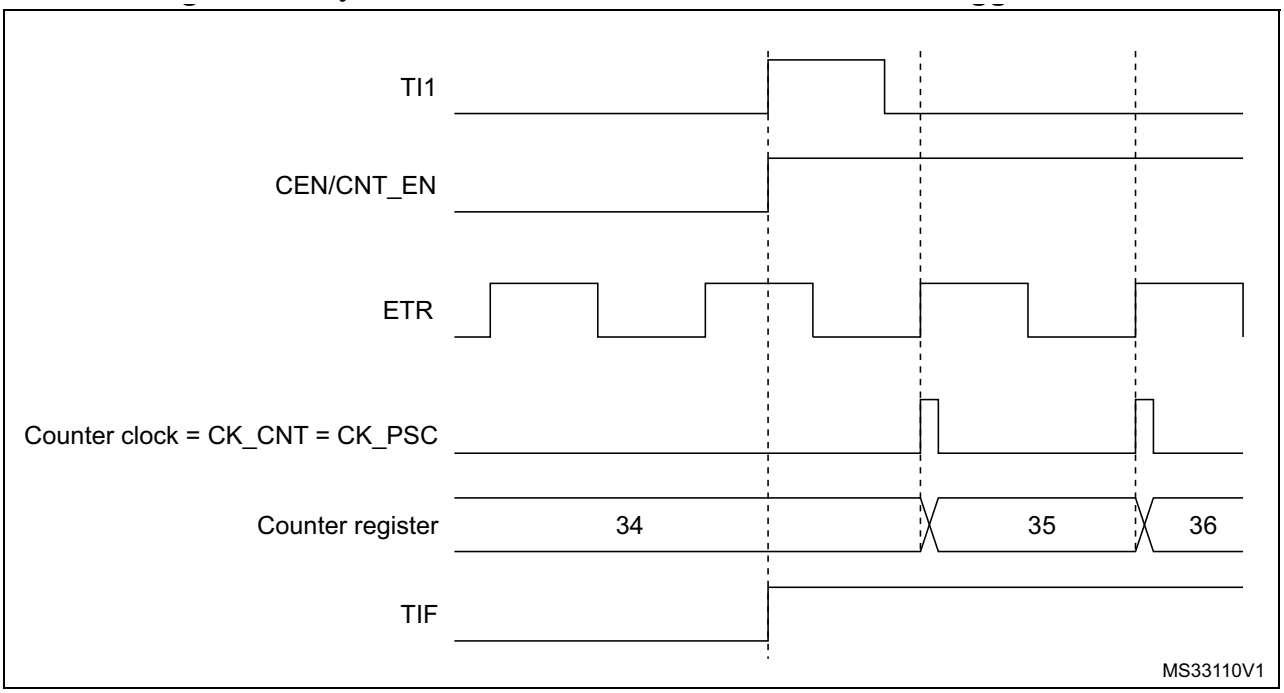
В этом примере счётчик прямого счёта работает по переднему фронту на ETR после переднего фронта запуска на TI1:

1. Ставим внешний запуск в регистре TIMx_SMCR:
   - ETF = 0000: фильтра нет; ETPS = 00: предделитель выключен; ETP = 0: передний фронт на ETR и ECE=1 для внешних тактов 2.
2. Ставим канал 1 на передний фронт TI:
   - IC1F=0000: фильтра нет; предделитель выключен, ничего не делаем; CC1S=01 в TIMx_CCMR1 для захвата входа; CC1P=0 в TIMx_CCER для полярности (и только переднего фронта).
3. Ставим таймер в режим запуска записью SMS=110 в регистр TIMx_SMCR. Выбираем вход с TI1 записью TS=101 в регистре TIMx_SMCR.

Передний фронт TI1 разрешает счёт и ставит флаг TIF. Считаются передние фронты ETR.

Задержка между фронтом ETR и действительным сбросом появляется из-за схемы ресинхронизации на входе ETRP.

Внешние такты 2 + Запуск.

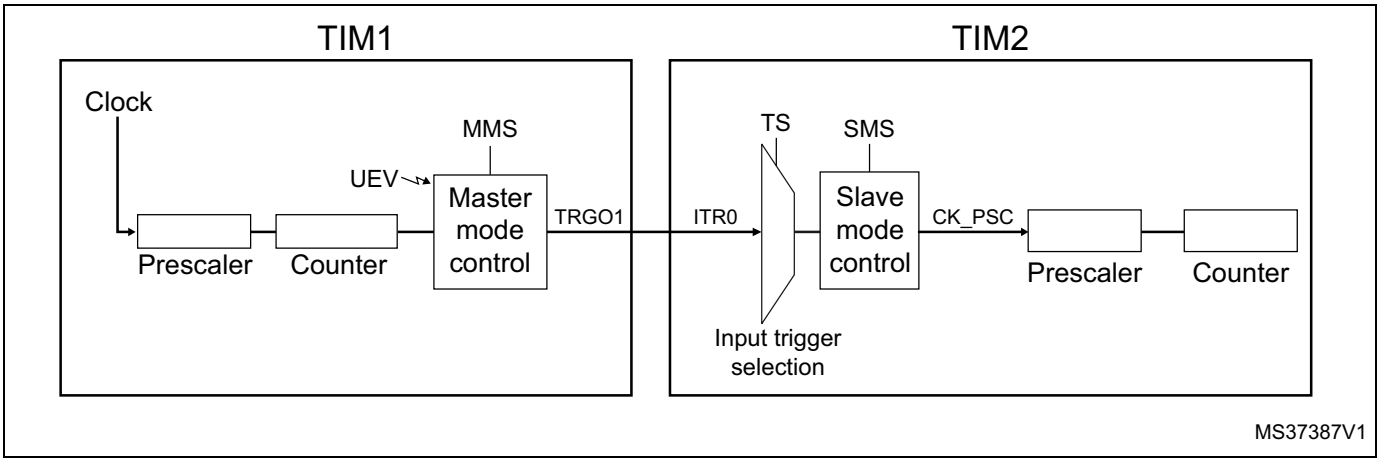


15.3.15.Синхронизация таймера

Таймеры TIM связаны между собой для синхронизации или сцепления. Таймер в режиме Ведущего может сбрасывать, запускать, останавливать другой таймер в режиме Ведомого.

NB: Такты ведомого таймера нужно включать до получения событий от ведущего и не должны меняться налету.

Таймер как предделитель другого.



Здесь TIM1 работает предделителем TIM2:

- Ставим TIM1 ведущим для выдачи периодического сигнала запуска по каждому событию `UEV`. При `MMS`=010 в регистре `TIM1_CR2` передний фронт появляется на выходе `TRGO1`.
- Подключаем выход `TRGO1` от TIM1 идёт на TIM2, стоящий в режиме ведомого с внутренним запуском по `ITR0`. Ставится битами `TS=000` в регистре `TIM2_SMCR`.
- Ставим контоллер ведомого в режим внешних тактов 1 (`SMS`=111 в регистре `TIM2_SMCR`). Отныне TIM2 тактируется передним фронтом сигнала переполнения TIM1.
- Наконец-то включаем оба таймера их битами `CEN` в регистрах `TIMx_CR1`.

NB: Если на TIM1 `OCx` стоит внешним запуском (`MMS`=1xx), то его передний фронт тактирует счётчик TIM2.

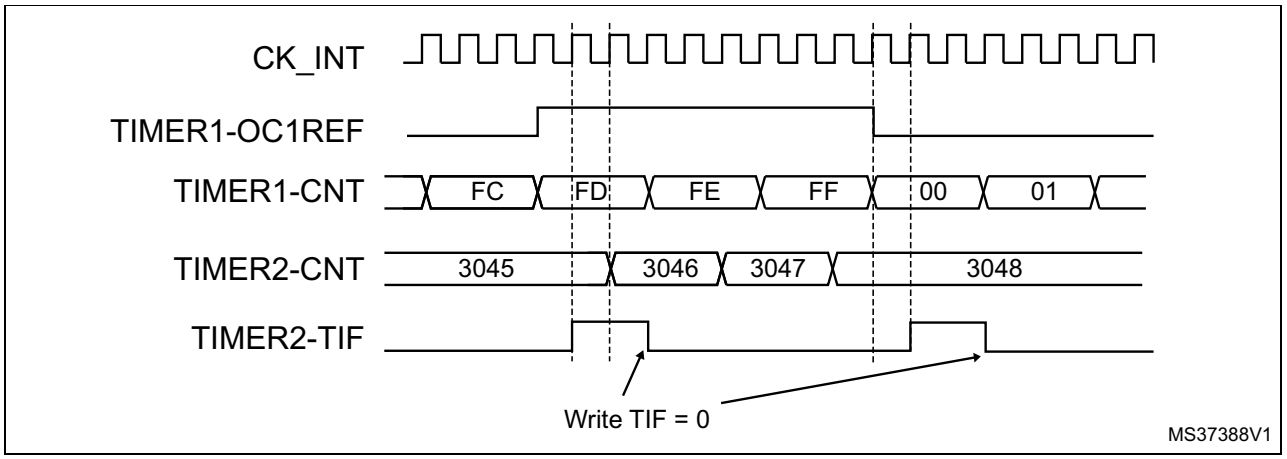
Один таймер разрешает другой.

В этом примере мы разрешаем TIM2 выходом сравнения TIM1. TIM2 считает предделённые внутренние такты только при высоком `OC1REF` от TIM1. Оба счётчика тактируются после предделителя CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

- Ведущий TIM1 посылает `OC1REF` на выход как запуск (`MMS`=100 в `TIM1_CR2`).
- Конфигурируем `OC1REF` у TIM1 (регистр `TIM1_CCMR1`).
- TIM2 получает запуск от TIM1 (`TS`=000 в регистре `TIM2_SMCR`).
- Ставим TIM2 в вентильный режим (`SMS`=101 в регистре `TIM2_SMCR`).
- Разрешаем TIM2 установкой бита `CEN` (регистр `TIM2_CR1`).
- Пускаем TIM1 установкой бита `CEN` (регистр `TIM1_CR1`).

NB: Такты счётчика 2 не синхронизированы со счётчиком 1, режим влияет только на разрешение счётчика TIM2.

Вентильный TIM2 по OC1REF от TIM1.

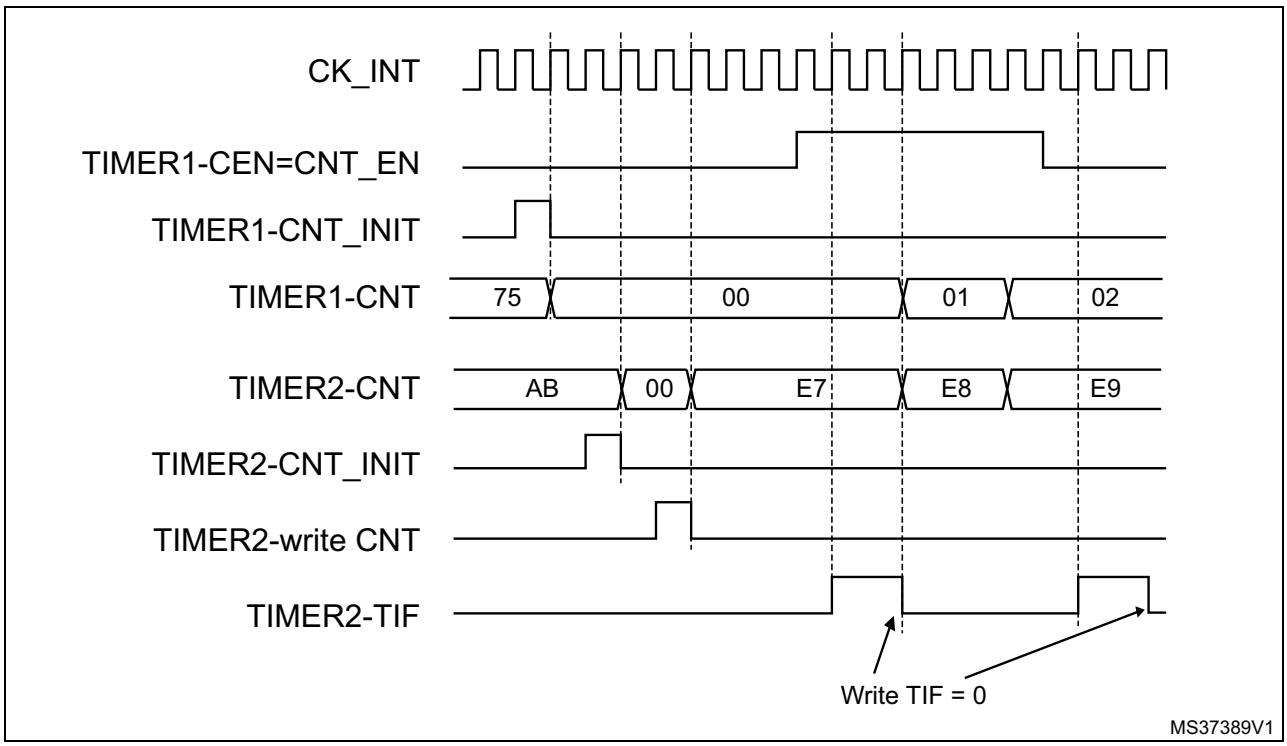


Здесь счётчик и предделитель TIM2 не инициализируются, а работают с текущего состояния. Для этого оба таймера сбрасываются до старта TIM1. Затем можно записать в счётчики любое значение. Сбросить таймеры можно программно битом `UG` в регистрах `TIMx_EGR`.

В следующем примере синхронизируем TIM1 и TIM2. Ведущий TIM1 начинает с 0. Ведомый TIM2 начинает с `0xE7`. Оба предделителя совпадают. TIM2 2 останавливается при выключении TIM1 записью '0' в бит `CEN` регистра `TIM1_CR1`:

- Ведущий TIM1 посылает `OC1REF` на выход как запуск (`MMS`=100 в `TIM1_CR2`).
- Конфигурируем `OC1REF` у TIM1 (регистр `TIM1_CCMR1`).
- TIM2 получает запуск от TIM1 (`TS`=000 в регистре `TIM2_SMCR`).
- Ставим TIM2 в вентильный режим (`SMS`=101 в регистре `TIM2_SMCR`).
- Сбрасываем TIM1 записью '1' в бит `UG` (регистр `TIM1_EGR`).
- Сбрасываем TIM2 записью '1' в бит `UG` (регистр `TIM2_EGR`).
- Пишем `0xE7` в счётчик TIM2 (`TIM2_CNTL`).
- Разрешаем TIM2 установкой бита `CEN` (регистр `TIM2_CR1`).
- Пускаем TIM1 установкой бита `CEN` (регистр `TIM1_CR1`).
- Останавливаем TIM1 снятием бита `CEN` (регистр `TIM1_CR1`).

Вентильный TIM2 с разрешением от TIM1.

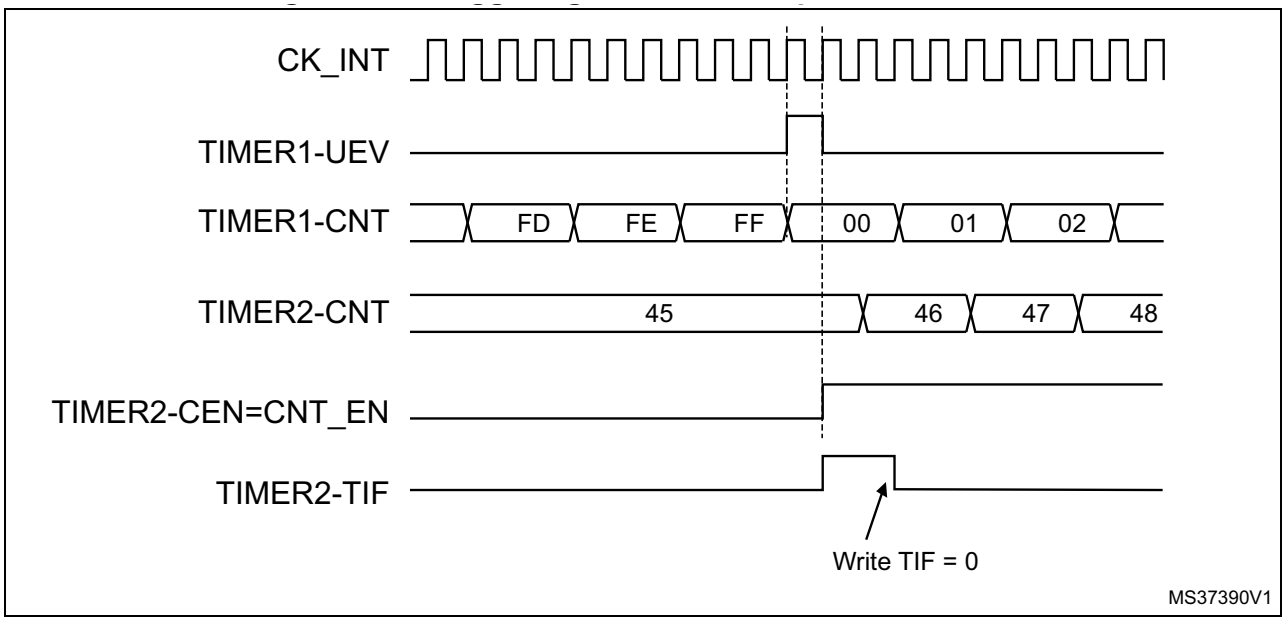


Запуск одного таймера от другого.

В этом примере мы разрешаем TIM2 событием обновления TIM1. TIM2 начинает счёт с текущего состояния (не нулевого) предделёнными внутренними тактами только по событию обновления от TIM1. Когда TIM2 получает сигнал запуска, его бит CEN автоматически встаёт и счётчик работает вплоть до записи ‘0 в бит CEN регистра TIM2_CR1. Оба счётчика тактируются после предделителя CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

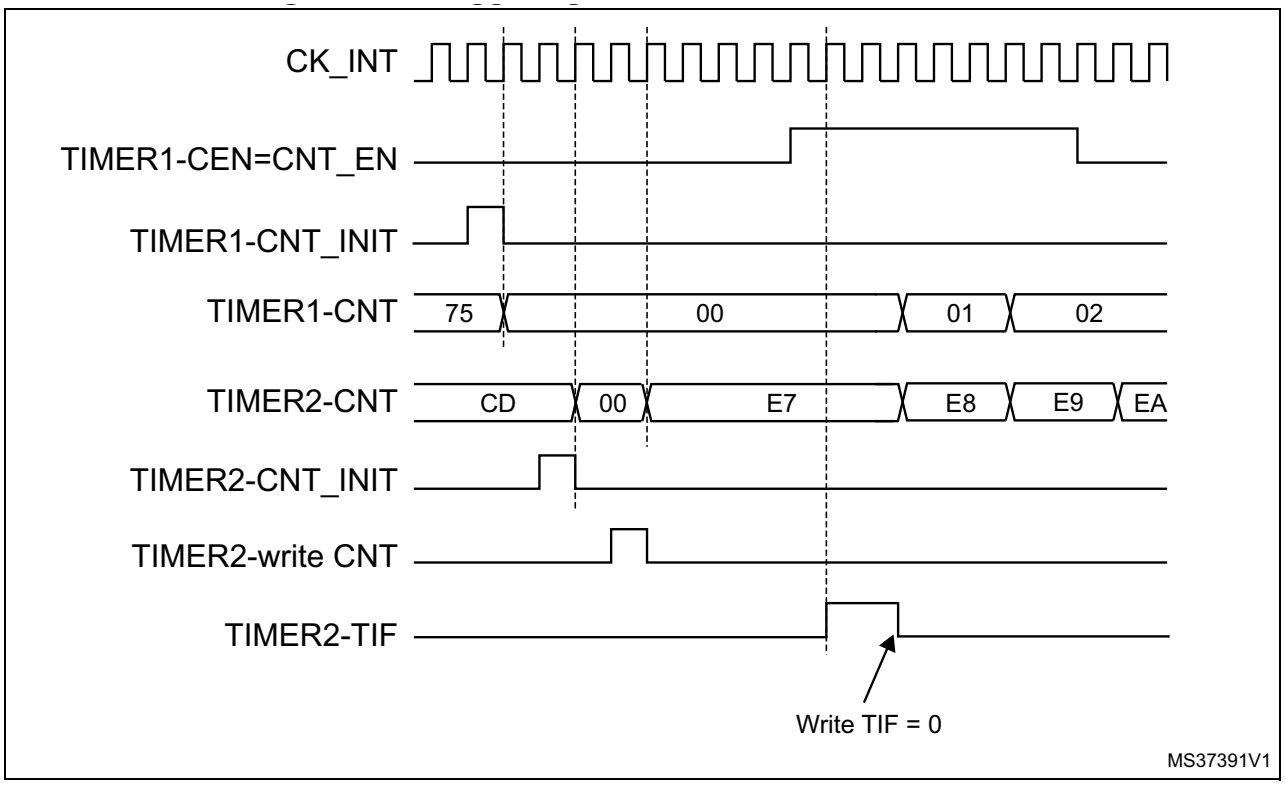
- Ведущий TIM1 посылает UEV на выход как запуск (MMS=010 в TIM1_CR2).
- Ставим период TIM1 (регистр TIM1_ARR).
- TIM2 получает запуск от TIM1 (TS=000 в регистре TIM2_SMCR).
- Ставим TIM2 в режим запуска (SMS=110 в регистре TIM2_SMCR).
- Пускаем TIM1 установкой бита CEN (регистр TIM1_CR1).

Запуск TIM2 событием обновления от TIM1.



Как и в предыдущем примере, можно инициализировать оба счётчика до старта.

Запуск TIM2 разрешением TIM1.



Синхронный старт 2 таймеров по внешнему запуску.

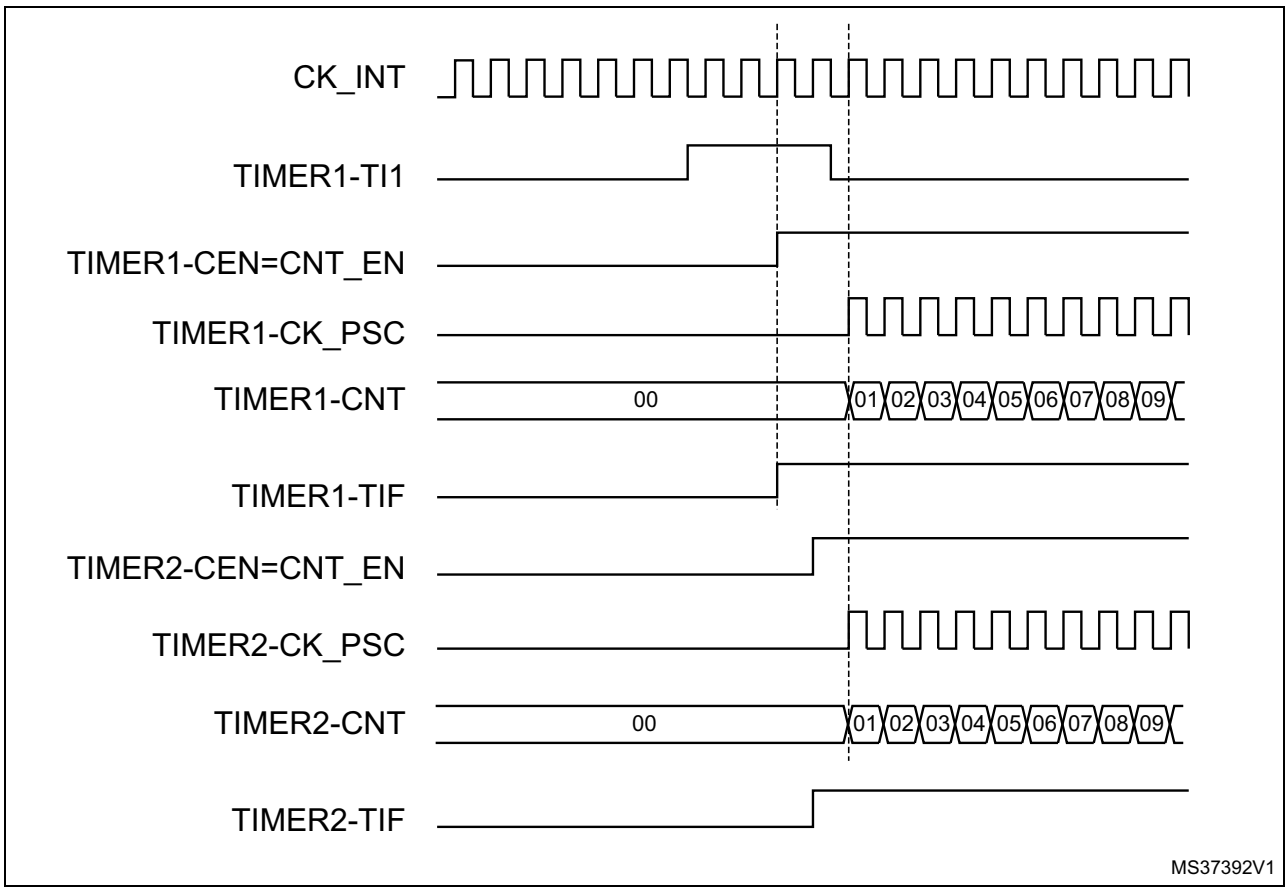
В этом примере передний фронт на входе TI1 TIM1 разрешает и TIM1 и TIM2. Для выравнивания счётчиков TIM1 нужно ставить в режим Ведущий/Ведомый (ведомый для TI1, ведущий для TIM2):

- Ставим TIM1 ведущим для посылки его Разрешения на выход запуска (MMS=001 в TIM1_CR2).
- Ставим TIM1 ведомым для получения запуска от TI1 (TS=100 в TIM1_SMCR).
- Ставим TIM1 в режим запуска (SMS=110 в TIM1_SMCR).
- Ставим TIM1 Ведущим/Ведомым записью MSM=1 (регистра TIM1_SMCR).
- Ставим TIM2 на получение запуска от TIM1 (TS=000 в TIM2_SMCR).
- Ставим TIM2 в режим запуска (SMS=110 в TIM2_SMCR).

По переднему фронту на TI1 (TIM1) оба счётчика начинают работать синхронно по внутренним тактам и оба ставят флаги TIF.

NB: В этом примере оба таймера инициализируются до старта установкой битов UG. Оба счётчика начинают с 0, но можно установить смещение записью в их регистры счётчиков (TIMx_CNT). Ниже видно задержку между CNT_EN и CK_PSC TIM1.

Запуск TIM1 и TIM2 сигналом TI1 TIM1.



15.3.16.Режим отладки

В режиме отладки (ядро Cortex-M3 остановлено), счётчик TIMx либо работает, либо останавливается, в зависимости от бита DBG_TIMx_STOP в модуле DBG.

При стоящем счётчике (DBG_TIMx_STOP = 1 в регистре DBGMCU_APBx_FZ) выходы отключаются (как при сброшенном бите MOE). Выходы можно перевести в неактивное состояние (бит OSSI = 1), или возложить управление на контроллер GPIO (бит OSSI = 0) для перевода их в высокоимпедансное состояние (Hi-Z).

15.4. Регистры TIMx

32-бит регистры надо писать словами. Остальные регистры надо писать полусловами или словами. Читать можно байтами,полусловами или словами.

15.4.1. Регистр управления 1 (TIMx_CR1)

Смещение адреса: 0x00

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:10 Резерв, не трогать.

— Биты 9:8 **CKD[1:0]:** Деление тактов.

Это отношение тактов таймера (CK_INT) и тактов задержки и выборки (t_{DTS}) в генераторах задержки и цифровых фильтрах (ETR, TIx),

00: $t_{DTS}=t_{CK\_INT}$

01: $t_{DTS}=2*t_{CK\_INT}$

10: $t_{DTS}=4*t_{CK\_INT}$

11: Резерв

— Бит 7 **ARPE**: Разрешение буфера перезагрузки TIMx_ARR

0: Нельзя

1: Нужно

— Бит 6:5 **CMS[1:0]**: Выбор режима по центру

00: По фронту. Прямой или обратный счёт в зависимости от бита направления (DIR).

01: По центру 1. Попеременный прямой и обратный счёт. Флаг сравнения выхода выводных каналов (CCxS=00 в TIMx_CCMRx) ставится только при обратном счёте.

10: По центру 2. Попеременный прямой и обратный счёт. Флаг сравнения выхода выводных каналов (CCxS=00 в TIMx_CCMRx) ставится только при прямом счёте.

11: По центру 3. Попеременный прямой и обратный счёт. Флаг сравнения выхода выводных каналов (CCxS=00 в TIMx_CCMRx) ставится и при прямом и при обратном счёте.

NB: При включённом таймере (CEN=1) переключать режим по фронту в режим по центру нельзя.

— Бит 4 **DIR**: Направление счёта.

0: Прямой

1: Обратный

NB: Бит читается только в режимах По центру и Кодера.

— Бит 3 **OPM**: Режим одного импульса

0: Счётчик не останавливается

1: Счётчик останавливается по следующему событию обновления (снимается бит CEN)

— Бит 2 **URS**: Источник запроса по событию UEV.

Изменяется программно.

0: Разрешённые запросы прерывания или DMA выдаются по:

– Переполнение/Исчерпание счётчика
– Установка бита UG
– Обновление от контроллера режима ведомого

1: Разрешённые запросы прерывания или DMA выдаются только по Переполнению/Исчерпанию счётчика.

— Бит 1 **UDIS:** Выключение выдачи события обновления UEV.

Изменяется программно.

0: Событие UEV выдаётся по:

– Переполнение/Исчерпание счётчика
– Установка бита UG
– Обновление от контроллера режима ведомого

1: Событие UEV не выдаётся, скрытые регистры (ARR, PSC, CCRx) хранят значение. При стоящем бите UG или аппаратном сбросе от контроллера режима ведомого счётчик и предделитель инициализируются.

— Бит 0 **CEN:** Включение счётчика.

0: Выключен

1: Включён

NB: Режимы Внешнего тактирования, Вентильный и Кодера работают только при заранее установленном бите CEN. Режим Запуска ставит его автоматически. В режиме Одного импульса бит CEN снимается аппаратно по событию обновления.

15.4.2. Регистр управления 2 (TIMx_CR2)

Смещение адреса: 0x04

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TI1S	MMS[2:0]			CCDS	Reserved		
								rw	rw	rw	rw	rw			

— Биты 15:8 Резерв, не трогать.

— Бит 7 **TI1S:** Выбор ножки для TI1

0: Ножка TIMx_CH1

1: Ножки TIMx_CH1, CH2 и CH3 (через XOR)

— Биты 6:4 **MMS[2:0]:** Режим ведущего

Выбор информации для синхронизации ведомых таймеров (TRGO):

000: **Сброс** - Бит UG в регистре TIMx_EGR. Если на входе запуска идёт сигнал сброса (контроллер режима ведомого в сбросе), то сигнал на TRGO задерживается относительно реального сброса.

001: **Разрешение** - Сигнал разрешения счётчика (CNT_EN). Это полезно для одновременного запуска нескольких таймеров или управления окном ведомого таймера. Сигнал разрешения это объединение по OR бита CEN с входом запуска вентильного режима.

При подаче CNT_EN по входу запуска сигнал TRGO задерживается, кроме режима ведущий/ведомый (см. бит MSM в регистре TIMx_SMCR).

010: **Обновление** - На выход запуска (TRGO) направлено событие обновления. Например, ведущий таймер может быть предделителем ведомого.

011: **Импульс сравнения** - После захвата или совпадения на выход запуска (TRGO) посылается положительный импульс установки флага CC1IF (даже если он стоит).

100: **Сравнение** - Сигнал OC1REF

101: **Сравнение** - Сигнал OC2REF

110: **Сравнение** - Сигнал OC3REF

111: **Сравнение** - Сигнал OC4REF

NB: Тактирование ведомого таймера и ADC нужно включать до получения событий от ведущего таймера и не должно меняться налету.

— Бит 3 **CCDS:** Выбор DMA Захвата/Сравнения

0: Запрос CCx DMA посылается по событию CCx

1: Запросы CCx DMA посылаются по событию обновления

— Биты 2:0 Резерв, не трогать.

15.4.3. Регистр управления режима ведомого (TIMx_SMCR)

Смещение адреса: 0x08

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	Res.	rw	rw	rw

— Бит 15 **ETP**: Полярность внешнего запуска

0: ETR не инверсный, активен высоким или передним фронтом.

1: ETR инверсный, активен низким или задним фронтом.

— Бит 14 **ECE**: Разрешение внешнего тактирования

0: Внешнее тактирование 2 выключено

1: Внешнее тактирование 2 включено. Счётчик работает от активного фронта ETRF.

NB:

1: Установка бита ECE равнозначна Внешнему тактированию 1 с подключением TRGI к ETRF (SMS=111 и TS=111).

2: Внешнее тактирование 2 можно использовать одновременно с режимами ведомого: Сброс, Вентильный и Запуска. Тем не менее, TRGI нельзя подключать к ETRF (TS не равен 111).

3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

— Биты 13:12 **ETPS[1:0]**: Предделитель внешнего запуска

Частота сигнала ETRP должна быть не больше 1/4 частоты TIMxCLK. Полезно при измерении быстрых внешних сигналов.

00: Предделитель Выкл.

01: ETRP / 2

10: ETRP / 4

11: ETRP / 8

— Биты 11:8 **ETF[3:0]**: Фильтр внешнего запуска

Определяет частоту выборки и длину цифрового фильтра сигнала ETRP. Цифровой фильтр это счётчик, выдающий сигнал на выход после N последовательных событий на входе:

0000: Без фильтра, выборка на частоте f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

— Бит 7 **MSM**: Режим Ведущий/Ведомый

0: Ничего

1: Действие входа запуска (TRGI) задерживается для полной синхронизации текущего таймера и его ведомых (через TRGO). Это полезно при синхронизации нескольких таймеров от одного внешнего события.

— Биты 6:4 **TS[2:0]**: Выбор запуска

000: Внутренний 0 (ITR0)

001: Внутренний 1 (ITR1)

010: Внутренний 2 (ITR2)

011: Внутренний 3 (ITR3)

100: Детектор фронта TI1 (TI1F_ED)

101: Фильтрованный вход 1 (TI1FP1)

110: Фильтрованный вход 2 (TI2FP2)

111: Внешний запуск (ETRF)

NB: Менять их можно только если не используются (например, при SMS=000) во избежание неверного определения фронта.

— Бит 3 Резерв, не трогать.

— Биты 2:0 **SMS[2:0]**: Выбор режима ведомого

При внешнем сигнале активный фронт TRGI подключается к внешнему входу с выбранной полярностью.

000: Режим ведомого выключен - если CEN = '1', то предделитель работает напрямую от внутренних тактов.

001: Режим кодера 1 - Прямой/обратный счёт по фронту TI2FP1 в зависимости от уровня TI1FP2.

010: Режим кодера 2 - Прямой/обратный счёт по фронту TI1FP2 в зависимости от уровня TI2FP1.

011: Режим кодера 3 - Прямой/обратный счёт по фронтам TI1FP1 и TI2FP2 в зависимости от уровня другого входа.

100: Режим сброса - Передний фронт TRGI инициализирует счётчик и запускает обновление регистров.

101: Вентильный режим - Такты счётчика разрешены при высоком TRGI. При низком TRGI счётчик останавливается, но не сбрасывается. Старт и стоп счётчика управляемые.

110: Режим запуска - Счётчик стартует по переднему фронту TRGI (но не сбрасывает). Управляемый только старт.

111: Внешние такты 1 - Счёт передних фронтов TRGI.

NB: Вентильный режим нельзя использовать если входом запуска выбран TI1F_ED (TS='100'). В действительности, TI1F_ED выдаёт 1 импульс на каждую передачу TI1F, тогда как вентильный режим проверяет уровень сигнала запуска.

Такты ведомого таймера нужно включать до приёма событий от ведущего таймера и не должны изменяться налету.

Таблица 86. Подключение внутреннего запуска TIMx.

Ведомый TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM2	TIM1	TIM8	TIM3	TIM4
TIM3	TIM1	TIM2	TIM5	TIM4
TIM4	TIM1	TIM2	TIM3	TIM8
TIM5	TIM2	TIM3	TIM4	TIM8

15.4.4. Регистр разрешения прерываний/DMA (TIMx_DIER)

Смещение адреса: 0x0C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Содержимое битов:

0: Нельзя

1: Можно

— Бит 15 Резерв, не трогать.

— Бит 14 **TDE**: Запрос DMA запуска

— Бит 13 Резерв, не трогать.

— Бит 12 **CC4DE**: Запрос DMA захвата/сравнения 4

— Бит 11 **CC3DE**: Запрос DMA захвата/сравнения 3

— Бит 10 **CC2DE**: Запрос DMA захвата/сравнения 2

— Бит 9 **CC1DE**: Запрос DMA захвата/сравнения 1

— Бит 8 **UDE**: Запрос DMA обновления

— Бит 7 Резерв, не трогать.

— Бит 6 **TIE**: Запрос прерывания запуска

— Бит 5 Резерв, не трогать.

— Бит 4 **CC4IE**: Запрос прерывания захвата/сравнения 4

— Бит 3 **CC3IE**: Запрос прерывания захвата/сравнения 3

— Бит 2 **CC2IE**: Запрос прерывания захвата/сравнения 2

— Бит 1 **CC1IE**: Запрос прерывания захвата/сравнения 1

— Бит 0 **UIE**: Запрос прерывания обновления

15.4.5. Регистр состояния (TIMx_SR)

Биты ставятся аппаратно, стираются записью 0.

Перезахват это появление события при стоящем его флаге.

Смещение адреса: 0x10

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Содержимое битов:

0: Не было

1: Было

— Биты 15:13 Резерв, не трогать.

— Бит 12 **CC4OF**: Флаг перезахвата захвата/сравнения 4

— Бит 11 **CC3OF**: Флаг перезахвата захвата/сравнения 3

— Бит 10 **CC2OF**: Флаг перезахвата захвата/сравнения 2

— Бит 9 **CC1OF**: Флаг перезахвата захвата/сравнения 1

— Бит 8:7 Резерв, не трогать.

— Бит 6 **TIF**: Флаг прерывания запуска

Ставится аппаратно по активному фронту на входе TRGI при работающем контроллере режима ведомого во всех режимах, кроме Вентильного, (тогда по обоим фронтам).

— Бит 5 Резерв, не трогать.

— Бит 4 **CC4IF**: Флаг прерывания захвата/сравнения 4

— Бит 3 **CC3IF**: Флаг прерывания захвата/сравнения 3

— Бит 2 **CC2IF**: Флаг прерывания захвата/сравнения 2

— Бит 1 **CC1IF**: Флаг прерывания захвата/сравнения 1

В режиме вывода CC1:

Есть исключения для выравнивания по центру (см. биты CMS в регистре TIMx_CR1).

В режиме ввода CC1:

— Бит 0 **UIF**: Флаг прерывания обновления

Удержание прерывания обновления ставится при:

- Переполнении или исчерпании счётчика и нулевом счётчике и UDIS=0 в регистре TIMx_CR1.
- При программной инициализации CNT битом UG в регистре TIMx_EGR, если URS=0 и UDIS=0 в регистре TIMx_CR1.
- При инициализации CNT событием запуска (см. регистр TIMx_SMCR), если URS=0 и UDIS=0 в регистре TIMx_CR1.

15.4.6. Регистр генерации событий (TIMx_EGR)

Событие генерируется программной установкой бита, снимается он аппаратно.

Смещение адреса: 0x14

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

— Биты 15:7 Резерв, не трогать.

— Бит 6 **TG**: Запуск.

1: Ставит флаг TIF в регистре TIMx_SR. Могут возникнуть прерывание и запрос DMA.

— Бит 5 Резерв, не трогать.

— Бит 4 **CC4G**: Захват/сравнение 4.

— Бит 3 **CC3G**: Захват/сравнение 3.

— Бит 2 **CC2G**: Захват/сравнение 2.

— Бит 1 **CC1G**: Захват/сравнение 1.

1: Выдаёт событие захвата/сравнения:

В режиме вывода CC1:

Ставит флаг CC1IF. Могут возникнуть прерывание и запрос DMA.

В режиме ввода CC1:

Текущее значение счётчика пишется в TIMx_CCR1. Ставит флаг CC1IF. Могут возникнуть прерывание и запрос DMA. При уже стоящем флаге CC1IF ставится и флаг CC1OF.

— Бит 0 **UG**: Обновление.

1: Инициализирует счётчик и обновление регистров. Текущий счётчик предделителя чистится не влияя на само значение. В режиме по центру или при DIR=0 счётчик обнуляется, иначе перегружается из регистра TIMx_ARR (при DIR=1).

15.4.7. Регистр режима захвата/сравнения 1 (TIMx_CCMR1)

Смещение адреса: 0x18

По сбросу: 0x0000

Каналы могут работать на ввод (захват) и вывод (сравнение), что определяется битами `CCxS`. У остальных битов для ввода и вывода значение отличается. `OCxx` обозначает функции вывода, `ICxx` функции ввода. При этом используются разные каскады.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Сравнение выхода:

— Бит 15 **OC2CE:** Разрешение очистки сравнения 2.

— Биты 14:12 **OC2M[2:0]**: Режим сравнения выхода 2.

— Бит 11 **OC2PE**: Разрешение предзагрузки сравнения 2.

— Бит 10 **OC2FE**: Быстрое разрешение сравнения 2.

— Биты 9:8 **CC2S[1:0]**: Выбор Захвата/Сравнения 2.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC2 выходной

01: Канал CC2 входной, IC2 на TI2

10: Канал CC2 входной, IC2 на TI1

11: Канал CC2 входной, IC2 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC2S можно писать только при выключенном канале (CC2E = ‘0’ в TIMx_CCER).

— Бит 7 **OC1CE**: Разрешение очистки сравнения выхода 1.

0: ETRF на OC1REF не влияет

1: Высокий уровень на ETRF стирает OC1REF

— Биты 6:4 **OC1M[2:0]**: Режим сравнения выхода 1.

Определяют поведение выходного опорного сигнала OC1REF, от которого происходят OC1 и OC1N. OC1REF активен высоким, а OC1 и OC1N зависят от битов CC1P и CC1NP.

000: Заморозка - сравнение TIMx_CCR1 и TIMx_CNT на выходы не влияет (режим используется для генерации временной базы).

001: При совпадении TIMx_CNT и TIMx_CCR1 сигнал OC1REF ставится активным (высоким).

010: При совпадении TIMx_CNT и TIMx_CCR1 сигнал OC1REF ставится неактивным (низким).

011: Переключение - при TIMx_CNT=TIMx_CCR1 сигнал OC1REF перебрасывается.

100: Принудительный неактивный - OC1REF ставится низким.

101: Принудительный активный - OC1REF ставится высоким.

110: ШИМ 1 - при прямом счёте канал активен пока TIMx_CNT<TIMx_CCR1. При обратном счёте канал неактивен (OC1REF=‘0’) пока TIMx_CNT>TIMx_CCR1.

111: ШИМ 2 - при прямом счёте канал неактивен пока TIMx_CNT<TIMx_CCR1. При обратном счёте канал активен пока TIMx_CNT>TIMx_CCR1.

NB:

1: Эти биты нельзя изменить пока стоит LOCK уровень 3 (биты LOCK в регистре TIMx_BDTR) и CC1S=’00’ (канал включён на выход).

2: В ШИМ 1 и 2 уровень OCREF изменяется только когда изменяется результат сравнения или при переключении из “Заморозки” в “ШИМ”.

— Бит 3 **OC1PE**: Разрешение регистра предзагрузки.

0: Предзагрузка TIMx_CCR1 выключена. TIMx_CCR1 можно писать в любое время, новое значение начинает действовать незамедлительно.

1: Предзагрузка TIMx_CCR1 включена. Доступ идёт к регистру предзагрузки. TIMx_CCR1 пишется в активный регистр по событию обновления.

NB:

1: Эти биты нельзя изменить пока стоит LOCK уровень 3 (биты LOCK в регистре TIMx_BDTR) и CC1S='00' (канал включён на выход).

2: Режим ШИМ без установки регистра предзагрузки можно использовать только в режиме одного импульса (стоит бит OPM в регистре TIMx_CR1). Иначе поведение непредсказуемо.

— Бит 2 **OC1FE**: Разрешение быстрого сравнения.

Ускоряет действие события запуска по входу на выход CC.

0: CC1 работает как обычно (сравнение счётчика и CCR1) даже при включённом запуске. Минимальная задержка переключения выхода CC1 при фронте на входе составляет 5 тактов.

1: Активный фронт запуска по входу действует как сравнение на выходе CC1. Тогда OC ставится в уровень сравнения независимо от его результата. Задержка между импульсом на входе и выходом CC1 сокращается до 3 тактов. OCFE работает только в ШИМ 1 и ШИМ 2.

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC1 выходной

01: Канал CC1 входной, IC1 на TI1

10: Канал CC1 входной, IC1 на TI2

11: Канал CC1 входной, IC1 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC1S можно писать только при выключенном канале (CC1E = '0' в TIMx_CCER).

Захват входа.

— Биты 15:12 **IC2F**: Фильтр захвата входа 2.

— Биты 11:10 **IC2PSC[1:0]** : Предделитель захвата входа 2.

— Биты 9:8 **CC2S[1:0]**: Выбор Захвата/Сравнения 2.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC2 выходной

01: Канал CC2 входной, IC2 на TI2

10: Канал CC2 входной, IC2 на TI1

11: Канал CC2 входной, IC2 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC2S можно писать только при выключенном канале (CC2E = '0' в TIMx_CCER).

— Биты 7:4 **IC1F[3:0]**: Фильтр захвата входа 1.

Определяет частоту выборки и длину цифрового фильтра сигнала TI1. Цифровой фильтр это счётчик, выдающий сигнал на выход после N последовательных событий на входе:

0000: Без фильтра, выборка на частоте f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

— Биты 3:2 **IC1PSC**: Предделитель захвата входа 1

Значение предделителя на входе CC1 (IC1).

По CC1E='0' (регистр TIMx_CCER) предделитель сбрасывается.

00: без предделителя, по каждому фронту на входе

01: каждые 2 события

10: каждые 4 события

11: каждые 8 события

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC1 выходной

01: Канал CC1 входной, IC1 на TI1

10: Канал CC1 входной, IC1 на TI2

11: Канал CC1 входной, IC1 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC1S можно писать только при выключенном канале (CC1E = ‘0’ в TIMx_CCER).

15.4.8. Регистр режима захвата/сравнения 2 (TIMx_CCMR2)

Смещение адреса: 0x1C

По сбросу: 0x0000

Каналы могут работать на ввод (захват) и вывод (сравнение), что определяется битами `CCxS`. У остальных битов для ввода и вывода значение отличается. `OCxx` обозначает функции вывода, `ICxx` функции ввода. При этом используются разные каскады.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE.	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Сравнение выхода:

— Бит 15 **OC4CE:** Разрешение очистки сравнения 4.

— Биты 14:12 **OC4M[2:0]**: Режим сравнения выхода 4.

— Бит 11 **OC4PE**: Разрешение предзагрузки сравнения 4.

— Бит 10 **OC4FE**: Быстрое разрешение сравнения 4.

— Биты 9:8 **CC4S[1:0]**: Выбор Захвата/Сравнения 4.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC4 выходной

01: Канал CC4 входной, IC4 на TI4

10: Канал CC4 входной, IC4 на TI3

11: Канал CC4 входной, IC4 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC4S можно писать только при выключенном канале (CC4E = ‘0’ в TIMx_CCER).

— Бит 7 **OC3CE**: Разрешение очистки сравнения выхода 3.

— Биты 6:4 **OC3M[2:0]**: Режим сравнения выхода 3.

— Бит 3 **OC3PE**: Разрешение регистра предзагрузки.

— Бит 2 **OC3FE**: Разрешение быстрого сравнения.

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC3 выходной

01: Канал CC3 входной, IC3 на TI3

10: Канал CC3 входной, IC3 на TI4

11: Канал CC3 входной, IC3 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC3S можно писать только при выключенном канале (CC3E = ‘0’ в TIMx_CCER).

Захват входа.

— Биты 15:12 **IC4F**: Фильтр захвата входа 4.

— Биты 11:10 **IC4PSC[1:0]** : Предделитель захвата входа 4.

— Биты 9:8 **CC4S[1:0]**: Выбор Захвата/Сравнения 4.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC4 выходной

01: Канал CC4 входной, IC4 на TI4

10: Канал CC4 входной, IC4 на TI3

11: Канал CC4 входной, IC4 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC4S можно писать только при выключенном канале (CC4E = ‘0’ в TIMx_CCER).

— Биты 7:4 **IC3F[3:0]**: Фильтр захвата входа 3.

Определяет частоту выборки и длину цифрового фильтра сигнала TI3. Цифровой фильтр это счётчик, выдающий сигнал на выход после N последовательных событий на входе:

0000: Без фильтра, выборка на частоте f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

— Биты 3:2 **IC3PSC**: Предделитель захвата входа 3

— Биты 1:0 **CC3S**: Выбор Захвата/Сравнения 3.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC3 выходной

01: Канал CC3 входной, IC3 на TI3

10: Канал CC3 входной, IC3 на TI4

11: Канал CC3 входной, IC3 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC3S можно писать только при выключенном канале (CC3E = ‘0’ в TIMx_CCER).

15.4.9. Регистр разрешения захвата/сравнения (TIMx_CCER)

Смещение адреса: 0x20

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CC4P	CC4E	Reserved		CC3P	CC3E	Reserved		CC2P	CC2E	Reserved		CC1P	CC1E
		rw	rw			rw	rw			rw	rw			rw	rw

— Биты 15:14 Резерв, не трогать.

— Бит 13 **CC4P**: Полярность выхода захвата/сравнения 4

— Бит 12 **CC4E**: Разрешение выхода захвата/сравнения 4

— Биты 11:10 Резерв, не трогать.

— Бит 9 **CC3P**: Полярность выхода захвата/сравнения 3

— Бит 8 **CC3E**:, Разрешение выхода захвата/сравнения 3

— Биты 7:6 Резерв, не трогать.

— Бит 5 **CC2P**: Полярность выхода захвата/сравнения 2

— Бит 4 **CC2E**: Разрешение выхода захвата/сравнения 2

— Биты 3:2 Резерв, не трогать.

— Бит 1 **CC1P**: Полярность выхода захвата/сравнения 1

1: Выдаёт событие захвата/сравнения:

В режиме вывода CC1:

0: OC1 активен высоким.

1: OC1 активен низким.

В режиме ввода CC1:

Выбирает для запуска или захвата прямой или инверсный IC1.

0: прямой: захват по переднему фронту IC1. Внешний запуск по IC1 не инвертируется.

1: инверсный: захват по заднему фронту IC1. Внешний запуск по IC1 инвертируется.

— Бит 0 **CC1E**: Разрешение выхода захвата/сравнения 1

В режиме вывода CC1:

0: Выкл. - OC1 не активен.

1: Вкл. - OC1 выводится на соответствующую ножку.

В режиме ввода CC1:

0: Захват выключен.

1: Захват включен.

Таблица 87. Управляющие биты стандартных OCx каналов

Бит CCxE	Состояние OCx
0	Выключен (OCx=0, OCx_EN=0)
1	OCx=OCxREF + Полярность, OCx_EN=1

NB: Состояние внешних ножек I/O, подключённых к стандартным OCx, зависит от состояния OCx и регистров GPIO и AFIO.

15.4.10.Счётчик TIM2 - TIM5 (TIMx_CNT)

Смещение адреса: 0x24

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **CNT[15:0]**: Значение счётчика.

15.4.11.Предделитель TIM2 - TIM5 (TIMx_PSC)

Смещение адреса: 0x28

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **PSC[15:0]**: Значение предделителя

Частота счёта (CK_CNT) равна f_{CK_PSC} / (PSC[15:0] + 1).

Значение PSC пишется в рабочий регистр предделителя по каждому событию обновления (включая очистку счётчика битом UG регистра TIMx_EGR или запуском в режиме Сброса).

15.4.12.Регистр предзагрузки TIM2 - TIM5 (TIMx_ARR)

Смещение адреса: 0x2C

По сбросу: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **ARR[15:0]**: Значение перезагрузки.

Если значение нулевое, то счётчик блокируется.

15.4.13.Регистр 1 захвата/сравнения TIM2 - TIM5 (TIMx_CCR1)

Смещение адреса: 0x34

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR1[15:0]**: Значение захвата/сравнения

В режиме вывода CC1:

Значение предзагрузки CCR1 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC1PE в регистре TIMx_CCMR1), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC1.

В режиме ввода CC1:

CCR1 содержит значение счётчика по последнему событию захвата 1 (IC1). Здесь TIMx_CCR1 только читается.

15.4.14.Регистр 2 захвата/сравнения TIM2 - TIM5 (TIMx_CCR2)

Смещение адреса: 0x38

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR2[15:0]**: Значение захвата/сравнения

В режиме вывода CC2:

Значение предзагрузки CCR2 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC2PE в регистре TIMx_CCMR2), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC2.

В режиме ввода CC2:

CCR2 содержит значение счётчика по последнему событию захвата 2 (IC2). Здесь TIMx_CCR2 только читается.

15.4.15.Регистр 3 захвата/сравнения TIM2 - TIM5 (TIMx_CCR3)

Смещение адреса: 0x3C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR3[15:0]**: Значение захвата/сравнения

В режиме вывода CC3:

Значение предзагрузки CCR3 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC3PE в регистре TIMx_CCMR3), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC3.

В режиме ввода CC3:

CCR3 содержит значение счётчика по последнему событию захвата 3 (IC3). Здесь TIMx_CCR3 только читается.

15.4.16.Регистр 4 захвата/сравнения TIM2 - TIM5 (TIMx_CCR4)

Смещение адреса: 0x40

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR4[15:0]**: Значение захвата/сравнения

В режиме вывода CC4:

Значение предзагрузки CCR4 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC4PE в регистре TIMx_CCMR4), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC4.

В режиме ввода CC4:

CCR4 содержит значение счётчика по последнему событию захвата 4 (IC4). Здесь TIMx_CCR4 только читается.

15.4.17.Регистр управления DMA (TIMx_DCR)

Смещение адреса: 0x48

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DBL[4:0]					Reserved			DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

— Биты 15:13 Резерв, не трогать.

— Биты 12:8 **DBL[4:0]**: Длина пакета DMA

Это число передач DMA (таймер обнаруживает передачу пакета после выполнения чтения/записи по адресу регистра TIMx_DMAR).

Адрес TIMx_DMAR:

00000: 1 передача

00001: 2 передачи

00010: 3 передачи

…

10001: 18 передач

— Биты 7:5 Резерв, не трогать.

— Биты 4:0 **DBA[4:0]**: Базовый адрес DMA

Это базовый адрес передач DMA (при доступе через адрес TIMx_DMAR). DBA определяется как смещение от адреса регистра TIMx_CR1.

Пример:

00000: TIMx_CR1,

00001: TIMx_CR2,

00010: TIMx_SMCR,

...

Пример: Есть передача: DBL = 7 передач и DBA = TIMx_CR1. Тогда доступ будет к 7 регистрам, начиная с адреса TIMx_CR1.

15.4.18.Регистр адреса полного доступа DMA (TIMx_DMAR)

Смещение адреса: 0x4C

По сбросу: 0x0000

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

— Биты 31:0 **DMAB[31:0]**: Регистр пакетного доступа DMA

Чтение и запись в регистр DMAR обращаются к регистру по адресу
(адрес TIMx_CR1) + (DBA + индекс DMA) x 4

где (адрес TIMx_CR1) это адрес управляющего регистра 1, DBA это базовый адрес DMA в регистре TIMx_DCR, индекс DMA автоматически изменяется при DMA передачах от 0 до DBL из регистра TIMx_DCR.

Пример пакетного DMA

Здесь пакетом DMA полусловами пишутся регистры CCRx (x = 2, 3, 4).

Нужны следующие шаги:

1. Ставим нужный канал DMA:
   - Адрес периферии DMA пишем в регистр DMAR
   - Адрес памяти это адрес данных в RAM для записи в регистры CCRx.
   - Число передач = 3.
   - Выключаем кольцевой режим.
2. Пишем поля DBA и DBL регистра DCR: DBL = 3 передачи, DBA = 0xE.
3. Разрешаем запрос DMA обновления TIMx (ставим бит UDE в регистре DIER).
4. Включаем TIMx
5. Включаем канал DMA

Здесь каждый регистр CCRx обновляется единожды. Если регистры CCRx нужно обновить дважды, то число передач должно быть 6. Буфер в RAM содержит data1, data2, data3, data4, data5 и data6. В регистры CCRx они передаются так: по первому запросу DMA, data1 в CCR2, data2 в CCR3, data3 в CCR4 и по второму запросу обновления DMA, data4 в CCR2, data5 в CCR3 в data6 в CCR4.

15.4.19.Карта регистров TIM2 - TIM5

0x00	TIMx_CR1	Reserved																	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																		0	0	0	0	0	0	0	0	0	0
0x04	TIMx_CR2	Reserved																			TI1S	MMS [2:0]			CCDS	Reserved							
	Reset value																				0	0	0	0	0								
0x08	TIMx_SMCR	Reserved															ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]			
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	
0x0C	TIMx_DIER	Reserved															TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Reserved	TIE	Reserved	CC4IE	CC3IE	CC2IE	CC1IE	UIE		
	Reset value																0	0	0	0	0	0	0		0		0	0	0	0	0		
0x10	TIMx_SR	Reserved																	CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Reserved	CC4IF	CC3IF	CC2IF	CC1IF	UIF		
	Reset value																		0	0	0	0			0		0	0	0	0	0		
0x14	TIMx_EGR	Reserved																							TG	Reserved	CC4G	CC3G	CC2G	CC1G	UG		
	Reset value																								0		0	0	0	0	0		
0x18	TIMx_CCMR1 Output Compare mode	Reserved															OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]		
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	TIMx_CCMR1 Input Capture mode	Reserved															IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]		
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x1C	TIMx_CCMR2 Output Compare mode	Reserved															O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]		
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	TIMx_CCMR2 Input Capture mode	Reserved															IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]		
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x20	TIMx_CCER	Reserved															Reserved			CC4P	CC4E	Reserved		CC3P	CC3E	Reserved		CC2P	CC2E	Reserved		CC1P	CC1E
	Reset value																			0	0			0	0			0	0			0	0

0x24	TIMx_CNT	Reserved	CNT[15:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved	PSC[15:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Reserved	ARR[15:0]													
	Reset value		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	Reserved															
0x34	TIMx_CCR1	Reserved	CCR1[15:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIMx_CCR2	Reserved	CCR2[15:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TIMx_CCR3	Reserved	CCR3[15:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TIMx_CCR4	Reserved	CCR4[15:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	Reserved															
0x48	TIMx_DCR	Reserved									DBL[4:0]					Reserved	DBA[4:0]				
	Reset value										0	0	0	0	0		0	0	0	0	0
0x4C	TIMx_DMAR	Reserved	DMAB[15:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16. Таймеры общего назначения (TIM9 - TIM14)

16.1. Введение в TIM9 - TIM14

Это 16-бит само-перегружаемые таймеры с программируемым предделителем.

Они могут использоваться для измерения длительности входных импульсов (захват входа), генерации сигналов (сравнение выхода и ШИМ).

Длина импульсов и сигналов может модулироваться от нескольких микросекунд до миллисекунд с помощью предделителей таймера и тактов RCC.

Улучшенные (TIM1 and TIM8) и обычные (TIMx) таймеры полностью независимы и не имеют общих ресурсов. Могут синхронизироваться между собой. См. *Секцию 15.3.15*.

16.2. Основные свойства TIM9 - TIM14

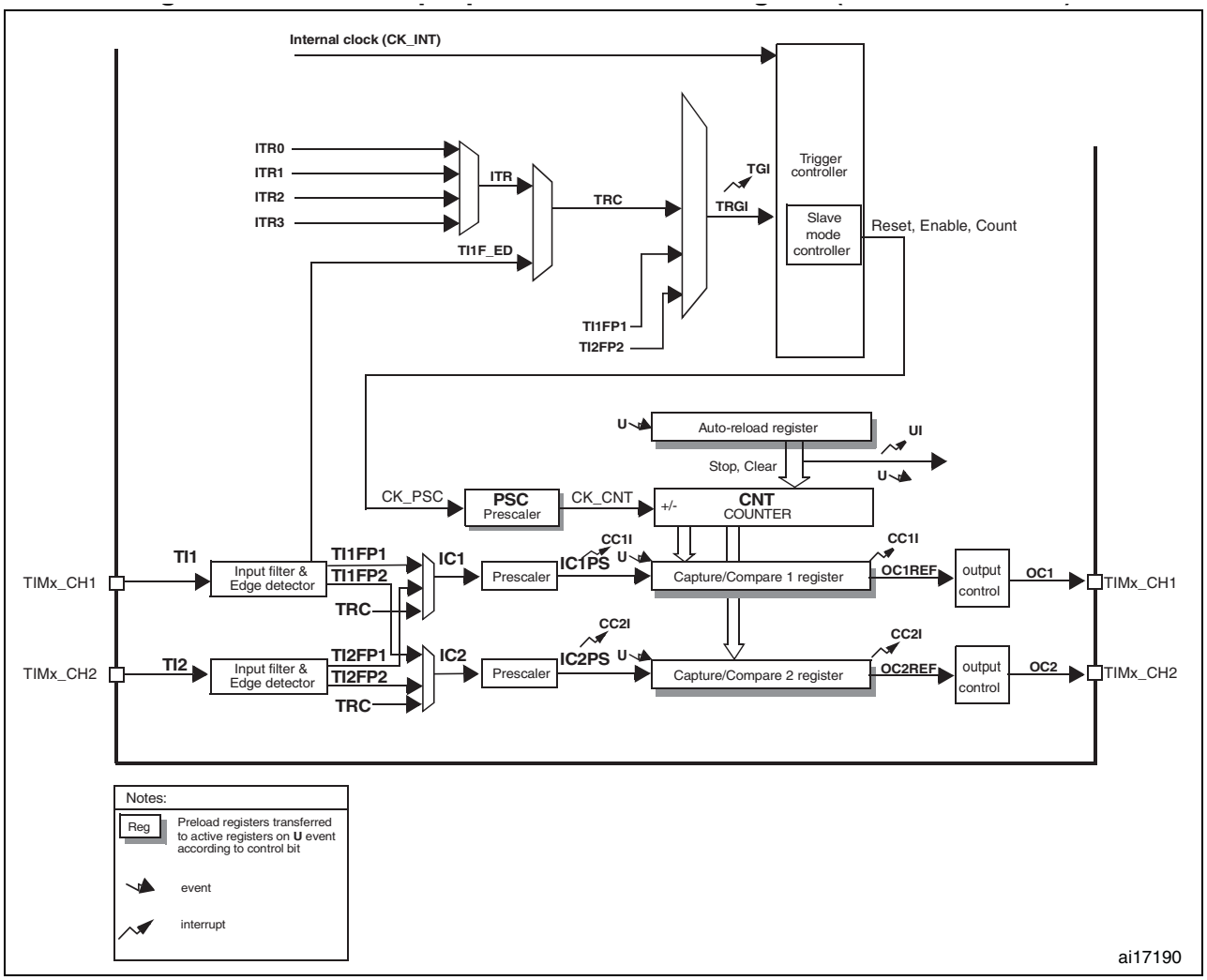
16.2.1. Основные свойства TIM9/TIM12

Это:

- 16-бит прямой счёт с само-перезагрузкой.
- 16-бит программируемый предделитель (можно “налету”) тактов на число от 1 до 65536.
- До 2 независимых каналов для:
  - Захвата входа
  - Сравнение выхода
  - ШИМ генерация (Фронт и Центр)
  - Одиночный импульс

- Схема синхронизации внешнего управления и объединения таймеров.
- Генерация прерываний по следующим событиям:
  - Обновление: переполнение, инициализация счётчика (программно или запуском)
  - Запуск (старт, стоп, инициализация или счёт по сигналу)
  - Захват входа
  - Сравнение выхода

Блок-схема TIM9/TIM12.

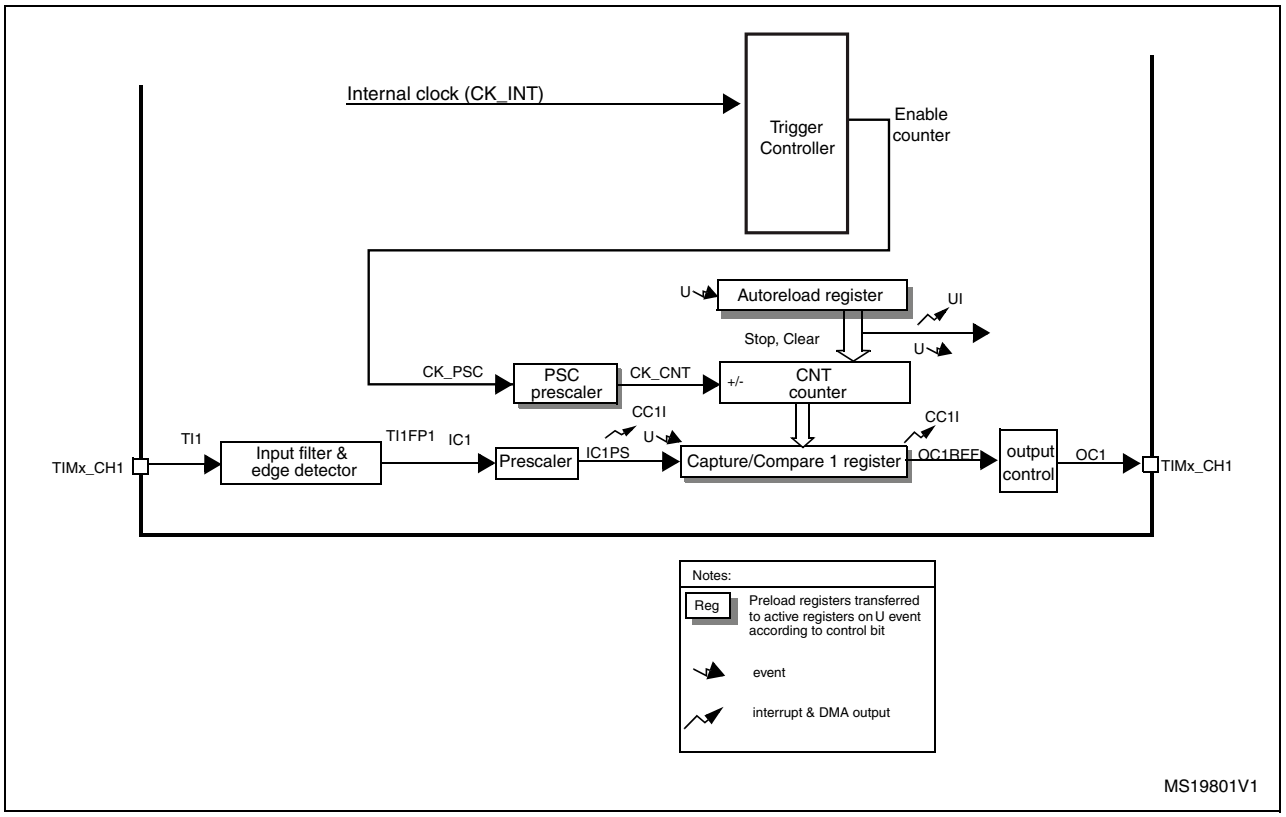


16.2.2. Основные свойства TIM10/TIM11 и TIM13/TIM14

Это:

- 16-бит прямой счёт с само-перезагрузкой.
- 16-бит программируемый предделитель (можно “налету”) тактов на число от 1 до 65536.
- Независимый канал для:
  - Захвата входа
  - Сравнение выхода
  - ШИМ генерация (Фронт и Центр)
  - Одиночный импульс
- Генерация прерываний по следующим событиям:
  - Обновление: переполнение, инициализация счётчика (программно)
  - Захват входа
  - Сравнение выхода

Блок-схема TIM10/11/13/14.



16.3. Функциональное описание таймеров TIM9 - TIM14

16.3.1. Узел счёта

Это прямой 16-бит счётчик с регистром само-перезагрузки и предделителем.

Счётчик, регистр перезагрузки и предделитель доступны программно для чтения и записи даже во время счёта.

Включает:

- Регистр счётчика (TIMx_CNT)
- Регистр предделителя (TIMx_PSC)
- Регистр перезагрузки (TIMx_ARR)

Регистр перезагрузки предзагружаемый (есть видимый и скрытый регистры). Содержимое видимого регистра пишется в скрытый постоянно или по каждому событию обновления (UEV), в зависимости от бита разрешения (ARPE) в регистре TIMx_CR1. Оно посылается программно или при переполнении счётчика при снятом бите UDIS в регистре TIMx_CR1.

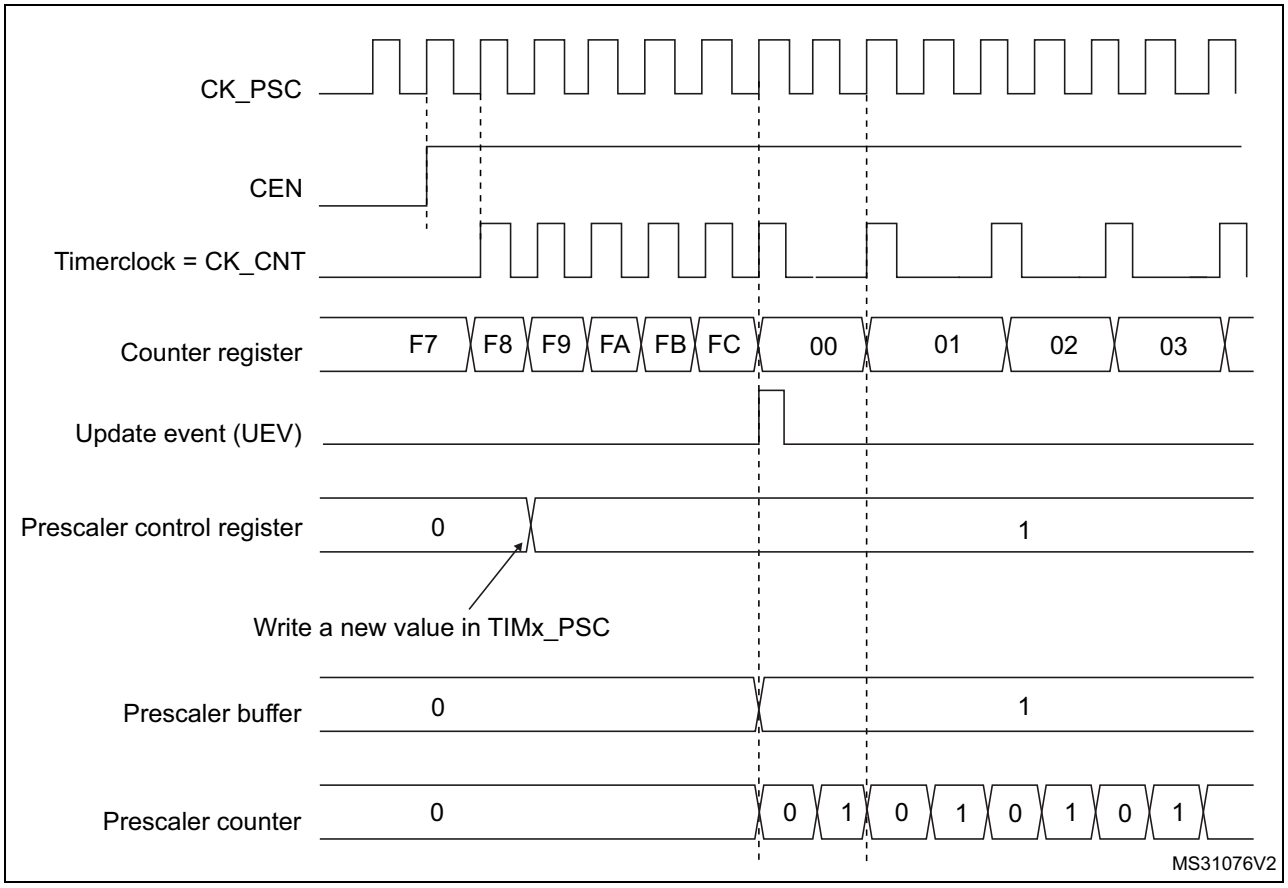
Счётчик тактируется выходом предделителя CK_CNT. Разрешается битом CEN в регистре TIMx_CR1.

Счёт начинается через 1 такт после установки бита CEN в регистре TIMx_CR1.

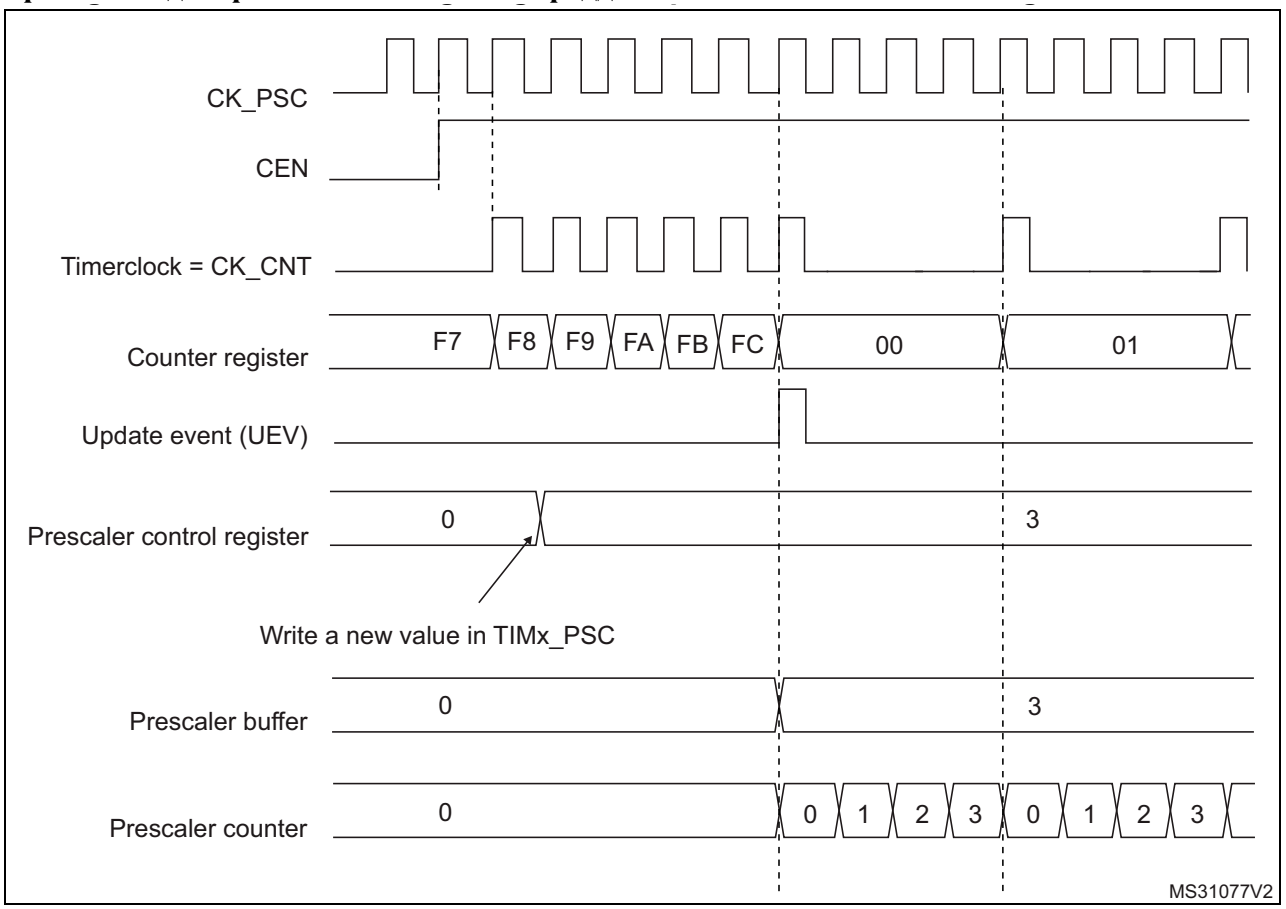
Описание предделителя

Это 16-бит счётчик делителя входной частоты (от 1 до 65536) с видимым регистром TIMx_PSC. Его можно менять налету. Новое значение счётчика пишется по следующему событию обновления.

Временная диаграмма с изменением предделителя с 1 на 2.



Временная диаграмма с изменением предделителя с 1 на 4.



16.3.2. Режимы счёта

Прямой счёт

Счёт идёт от 0 до значения перезагрузки (регистр TIMx_ARR), сбрасывается в 0 и выдаёт событие переполнения счётчика.

Событие обновления UEV выдаётся при каждом переполнении счётчика или установкой бита UG в регистре TIMx_EGR (программно или контроллером режима ведомого).

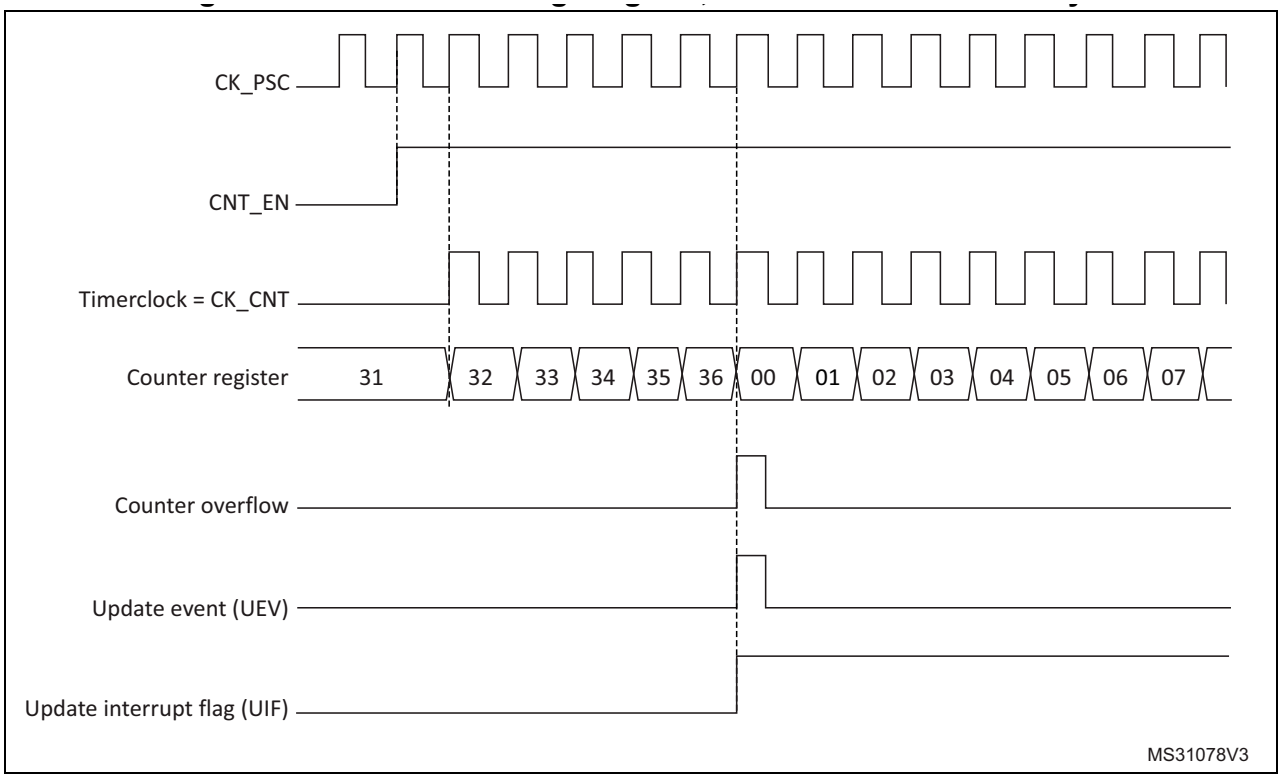
Событие UEV выключается установкой бита UDIS в регистре TIMx_CR1. Этим предупреждается запись скрытого регистра во время изменения регистра предзагрузки. Также событие UEV не появляется при сброшенном бите UDIS 0. Однако, счётчик продолжает считать с 0, равно как и предделитель (не изменив своего значения). Кроме того, если стоит бит URS в регистре TIMx_CR1, то установка бита UG выдаёт событие UEV без установки флага UIF (без прерывания). Этим предупреждается одновременная выдача прерываний обновления и захвата при очистке счётчика по событию захвата.

По событию обновления обновляются все регистры и ставится флаг обновления (бит UIF в регистре TIMx_SR). Это зависит от бита URS:

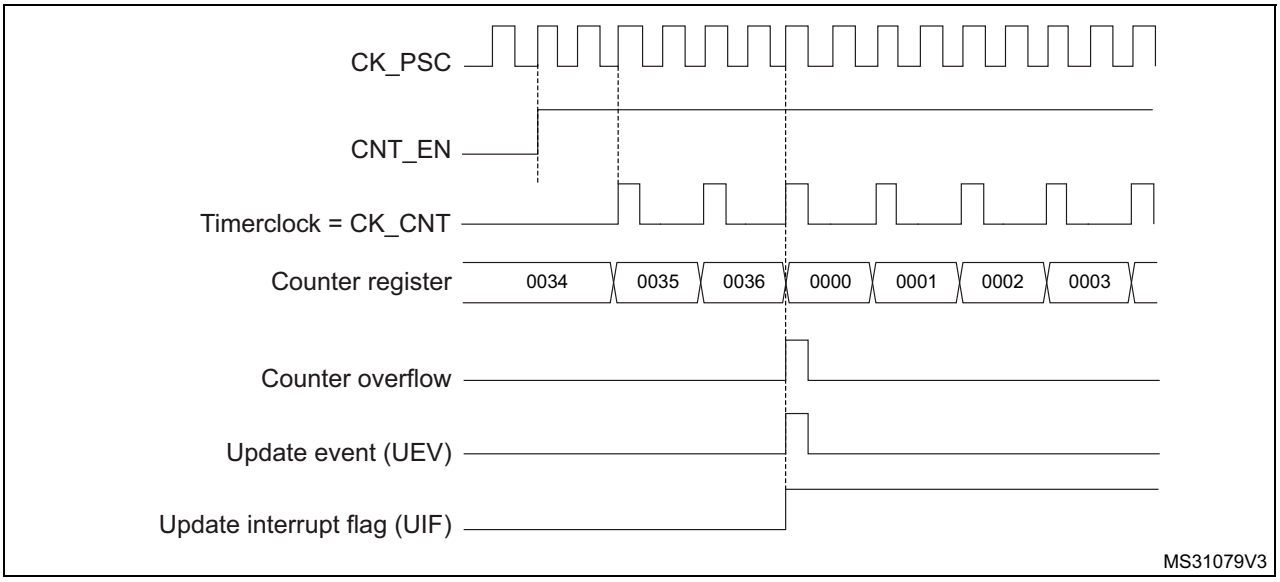
- В скрытый регистр перезагрузки пишется содержимое TIMx_ARR,
- Буфер предделителя перегружается из регистра TIMx_PSC.

Примеры ниже используют TIMx_ARR=0x36.

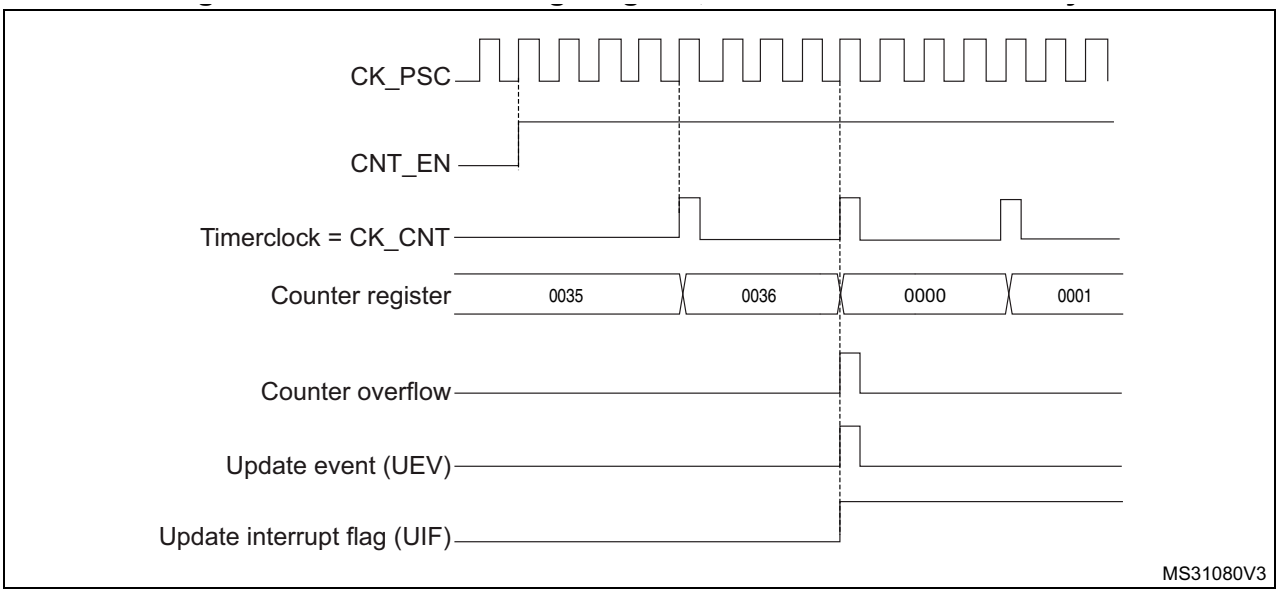
Временная диаграмма с делителем 1.



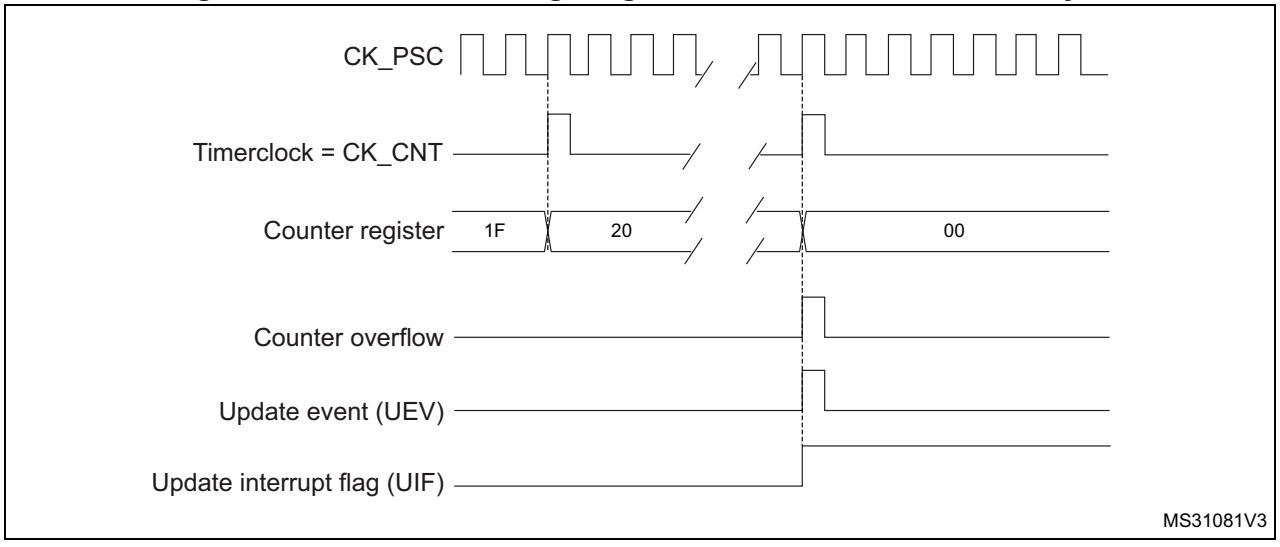
Временная диаграмма с делителем 2.



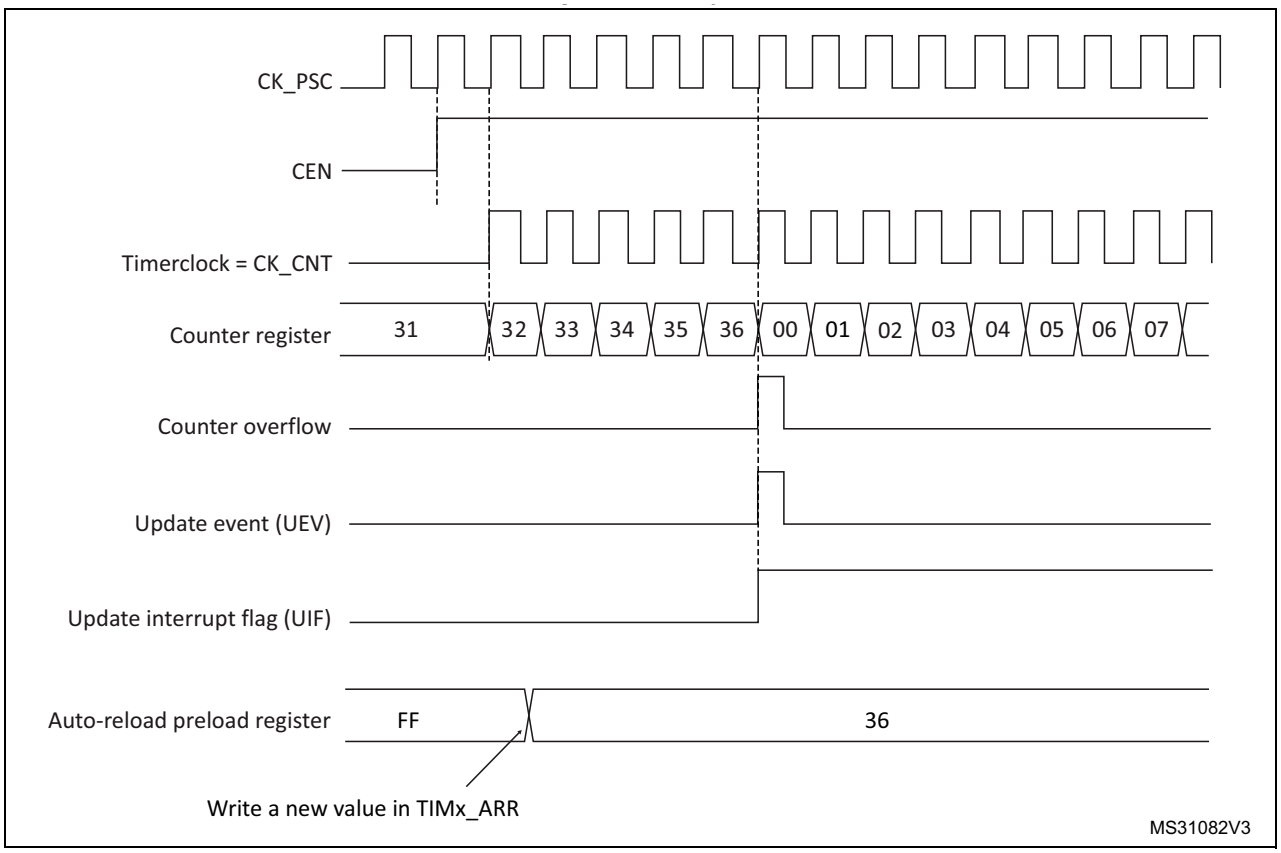
Временная диаграмма с делителем 4.



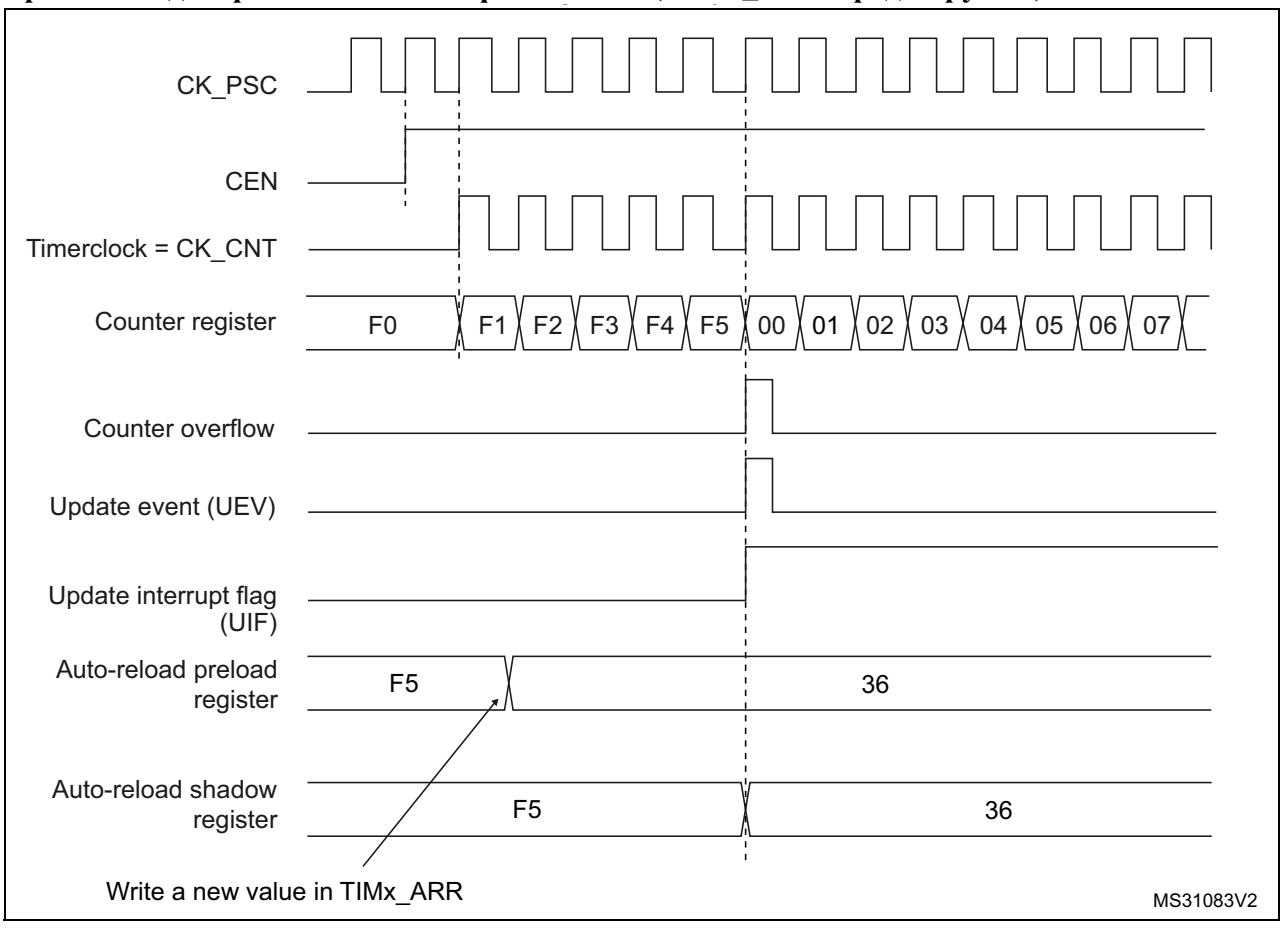
Временная диаграмма с делителем N.



Временная диаграмма события при ARPE=0 (TIMx_ARR не предзагружен).



Временная диаграмма события при ARPE=1 (TIMx_ARR предзагружен).



16.3.3. Выбор тактов

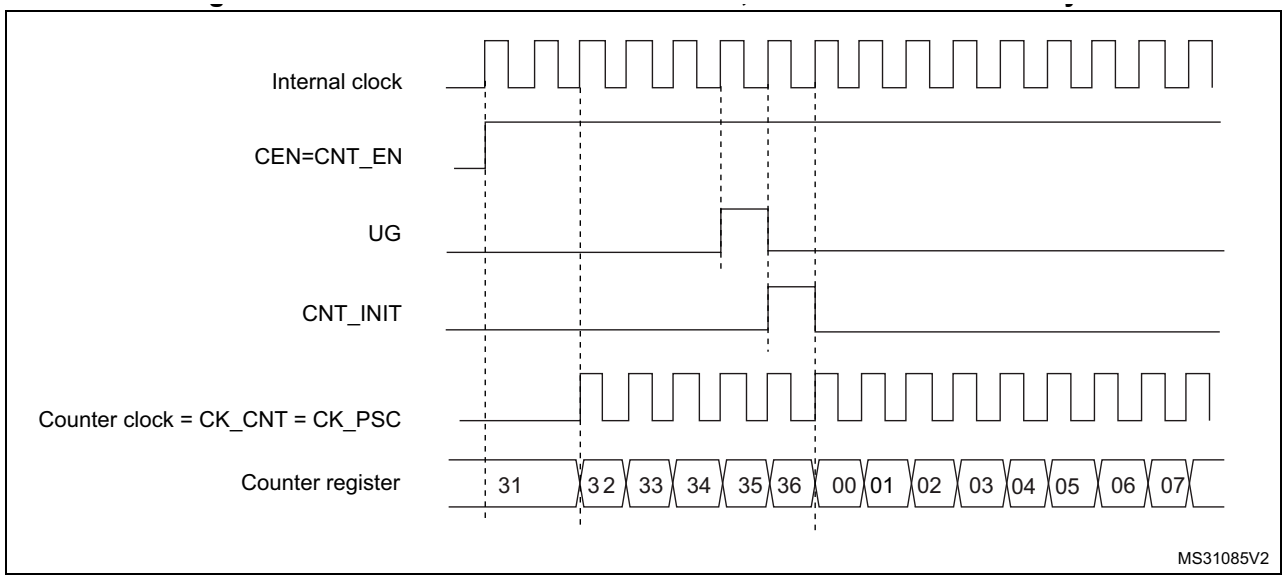
Счётчик может тактироваться из следующих источников:

- Внутренний (CK_INT)
- Внешний режим 1: внешняя ножка TIx
- Входы внутреннего сигнала (ITRx): использование одного таймера предделителем для другого.

Внутренний источник (CK_INT)

Если в ведомом режиме контроллер выключен (SMS=000), то всем управляют только биты CEN, DIR (в регистре TIMx_CR1) и UG (в регистре TIMx_EGR). Они изменяются только программно (кроме UG, снимающегося аппаратно). Сразу после установки бита CEN предделитель начинает получать внутренние такты CK_INT.

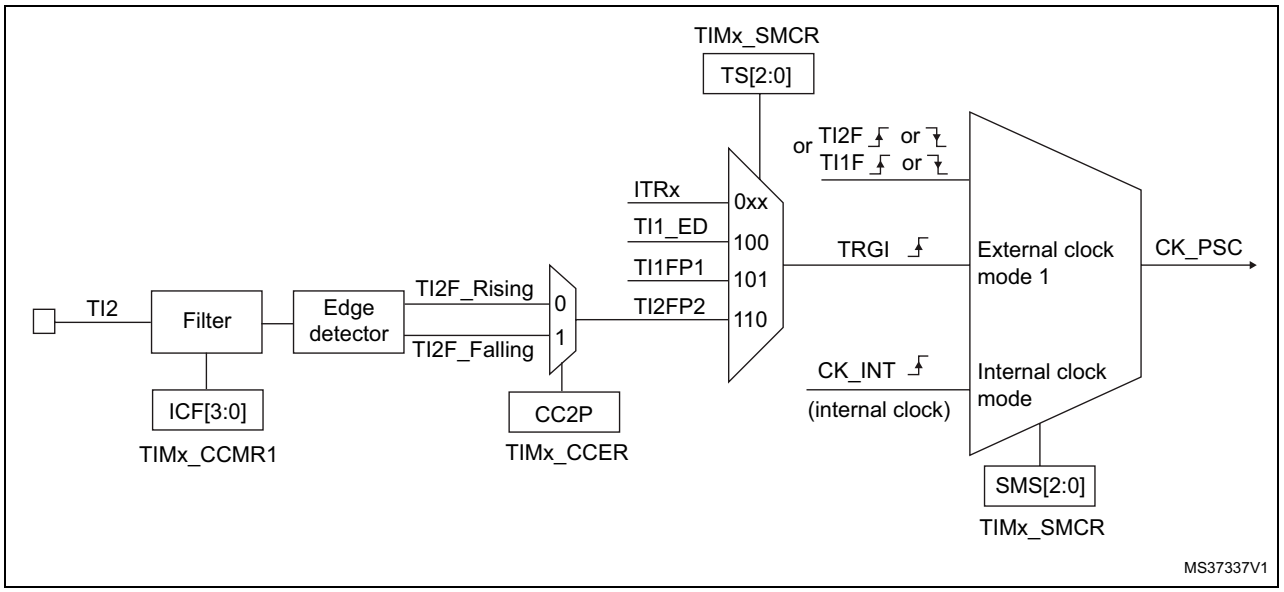
Схема управления в нормальном режиме прямого счёта без предделителя.



Внешний источник режим 1 (TIM9 и TIM12).

Включается при установке SMS=111 в регистре TIMx_SMCR. Счётчик работает от переднего или заднего фронта выбранного входа.

Внешний источник TI2.



Например, процедура включения режима прямого счёта по переднему фронту входа TI2:

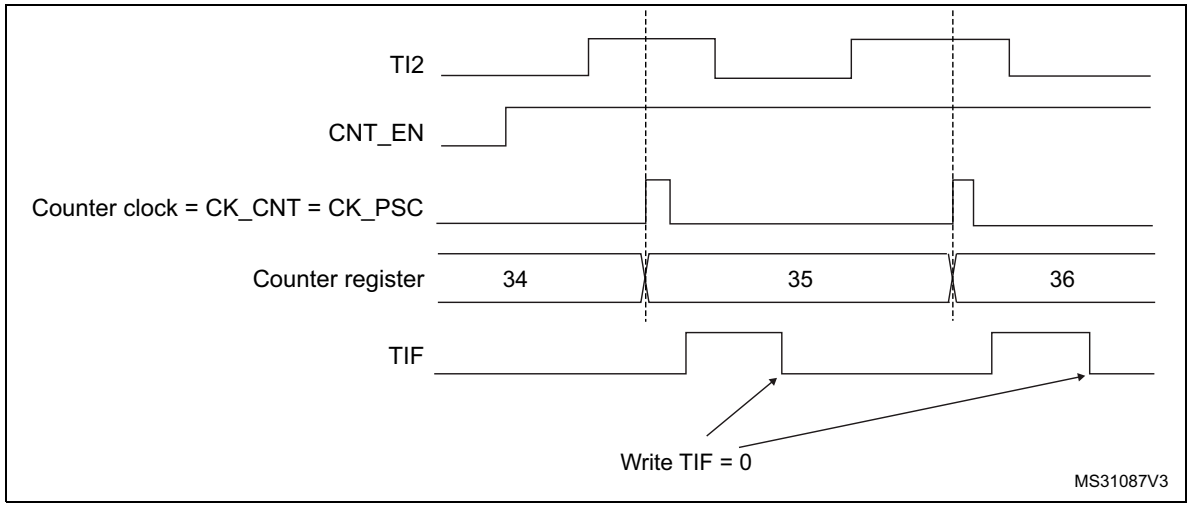
1. Включить определение переднего фронта TI2 канала 2 записью CC2S = ‘01’ в регистре TIMx_CCMR1.
2. Установить длительность входного фильтра битами IC2F[3:0] в регистре TIMx_CCMR1 (если фильтр не нужен, то остаётся IC2F=0000).
3. Выбрать полярность переднего фронта записью CC2P=CC2NP=0 в регистре TIMx_CCER.
4. Включить внешний режим 1 тактирования записью SMS=111 в регистре TIMx_SMCR.
5. Выбрать TI2 как источник сигнала записью TS=110 в регистре TIMx_SMCR.
6. Включить счётчик записью CEN=1 в регистре TIMx_CR1.

Предделитель захвата для запуска не используется, ну и бог с ним.

По переднему фронту на TI2 счётчик один раз тикает и ставит флаг TIF.

Задержка между передним фронтом на TI2 и реальным тиком появляется из-за схемы ресинхронизации на входе TI2.

Схема управления внешнего режима 1.

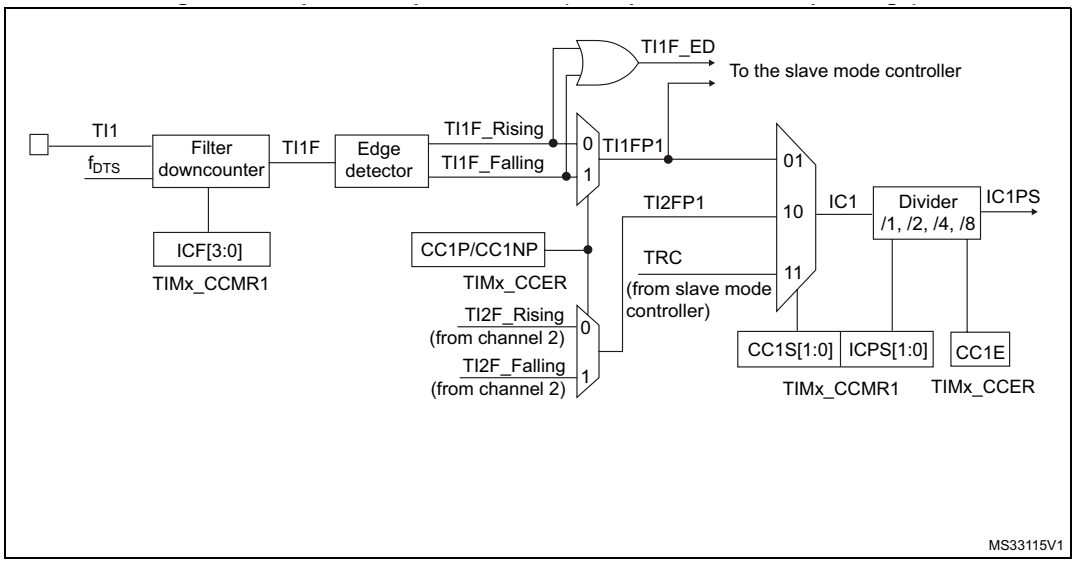


16.3.4. Каналы захвата/сравнения

Каждый канал захвата/сравнения построен вокруг парного регистра (включая невидимый), входного каскада захвата (с цифровым фильтром, мультиплексором и предделителем) и выходного каскада (с компаратором и управлением выходом).

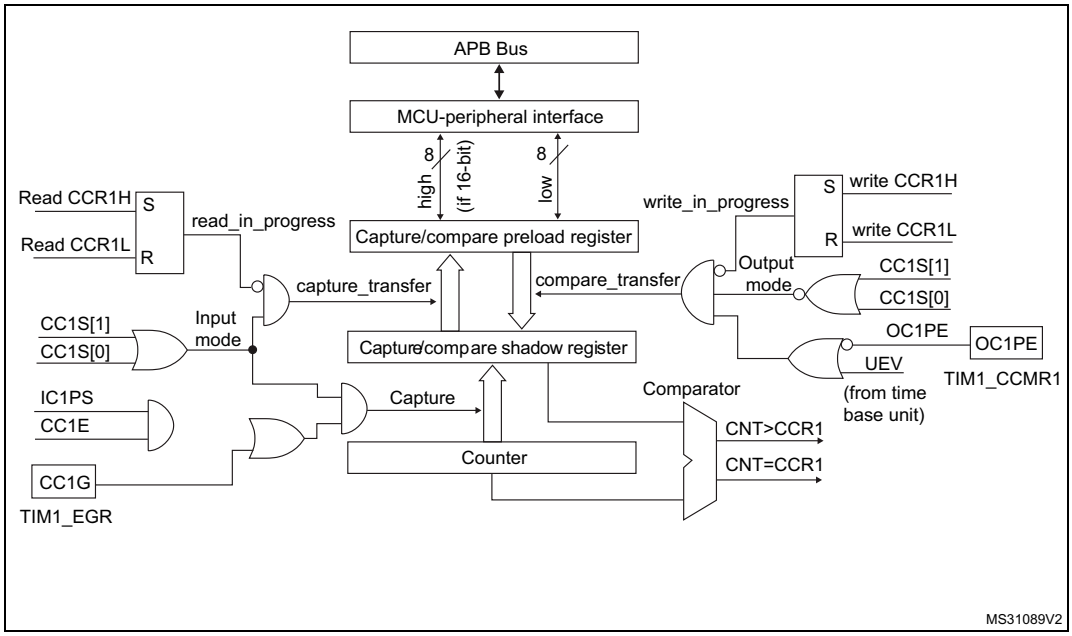
Входной каскад считывает нужный вход TIx для выдачи фильтрованного сигнала TIxF. Далее, детектор фронта с выбором полярности выдаёт сигнал (TIxFPx), что может быть использован как вход запуска контроллером режима ведомого или как команда захвата. Предделитель стоит до регистра захвата (ICxPS).

Входной каскад канала 1.

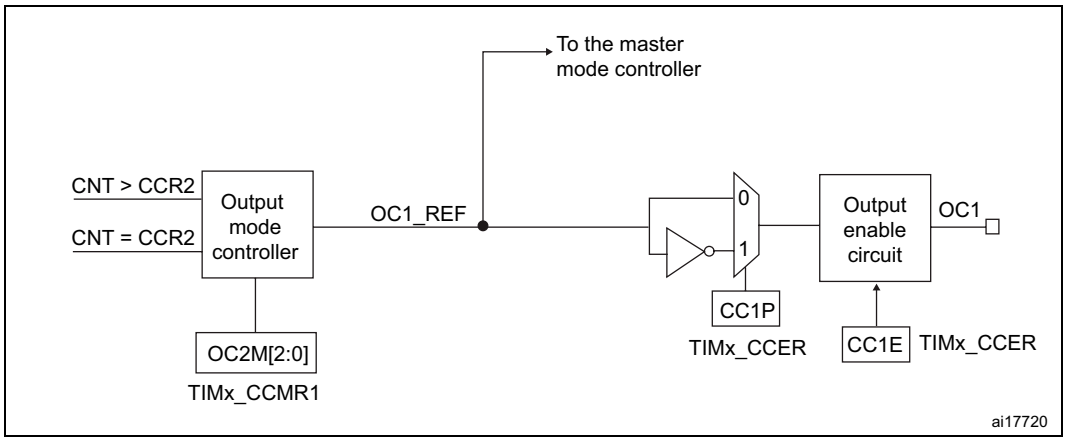


Выходной каскад выдаёт промежуточный сигнал, используемый как опорный: OCxREF (активный высокий). Полярность действует в конце цепочки.

Основная схема канала 1.



Выходной каскад канала захвата/сравнения 1.



Блок захвата/сравнения состоит из двух регистров: предзагрузки и невидимого. Снаружи доступен только регистр предзагрузки и по чтению и по записи.

Захват выполняется внутренним регистром и результат пишется в регистр предзагрузки.

При сравнении регистр предзагрузки пишется в невидимый, где и сравнивается со счётчиком.

16.3.5. Режим захвата входа

Регистры TIMx_CCRx защёлкивают содержимое счётчика после передачи, обнаруженной соответствующим сигналом ICx. При этом ставится флаг CCXIF в регистре TIMx_SR и ставится запрос прерывания. Если захват происходит при стоящем флаге CCxIF, то ставится флаг перезахвата CCxOF в регистре TIMx_SR. Флаг CCxIF снимается записью в него нуля или чтением захваченных данных из регистра TIMx_CCRx. CCxOF снимается записью ‘0’.

Далее захватываем содержимое TIMx_CCR1 по переднему фронту TI1. Делаем так:

- Выбираем активный вход: TIMx_CCR1 подключаем в TI1 и пишем "01" в биты CC1S регистра TIMx_CCMR1. Биты CC1S отличаются от 00, т. е. канал на вводе и регистр TIMx_CCR1 доступен только по чтению.
- Ставим длительность входного фильтра битами ICxF в регистре TIMx_CCMRx с учётом входного сигнала на TIx. Допустим, что при переключении входной сигнал нестабилен в течении пяти внутренних тактов. Значит длительность фильтра должна быть больше этих пяти тактов. Мы можем определить передачу на TI1 за 8 последовательных выборок нового уровня на частоте f_{DTS}. И мы пишем 0011 в биты IC1F регистра TIMx_CCMR1.

- Выбираем фронт сигнала на TI1 записью 0 в бит CC1P регистра TIMx_CCER (передний).
- Ставим предделитель. Здесь он нам не нужен (пишем 00 в биты IC1PS регистра TIMx_CCMR1).
- Разрешаем запись счётчика в регистр захвата установкой бита CC1E в TIMx_CCER.
- Если нужно разрешаем выдачу запросов прерывания битом CC1IE в TIMx_DIER.

Если захват входа произошёл, то:

- Регистр TIMx_CCR1 получает значение счётчика активной передачи.
- Ставится флаг прерывания CC1IF. Если он ещё стоял, то ставится и флаг CC1OF.
- В зависимости от бита CC1IE генерируется прерывание.

Чтобы обработать перезахват рекомендуется читать регистр данных до опроса флага перезахвата.

NB: Запросы прерывания IC можно выдавать программно установкой битов CCxG в регистре TIMx_EGR.

16.3.6. Режим ввода ШИМ (только TIM9/TIM12)

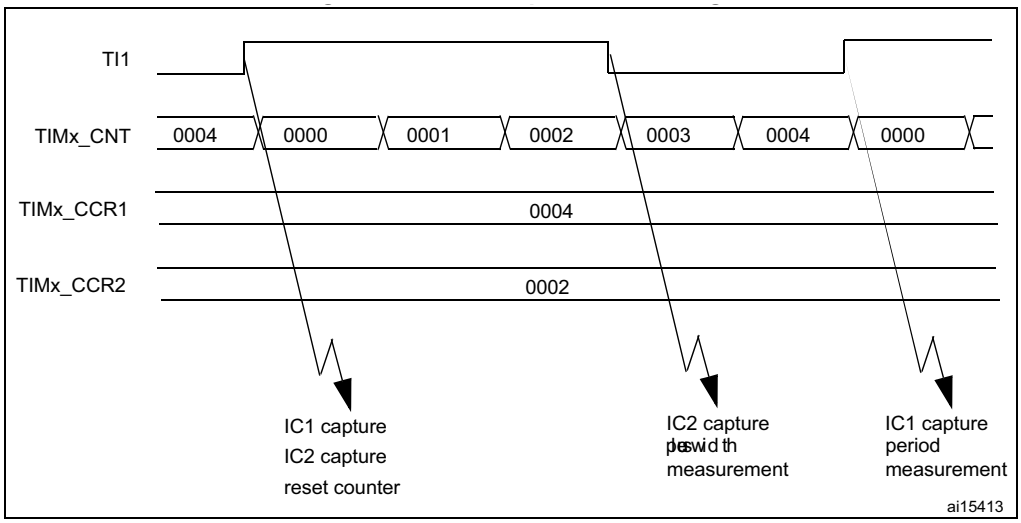
Это часть режима захвата входа. Процедура такая же, кроме:

- Два сигнала ICx направляются на на один вход TIx.
- Полярность фронтов этих входов ICx противоположная.
- Сигналом запуска берётся один из двух сигналов TIxFP, а контроллер ведомого режима выключается.

Например, можно измерить период (в регистре TIMx_CCR1) и длительность заполнения (в регистре TIMx_CCR2) ШИМ сигнала на TI1 следующей процедурой (зависит от частоты CK_INT и предделителя):

- Выбрать активный вход для TIMx_CCR1: Записать 01 в биты CC1S регистра TIMx_CCMR1 (TI1).
- Выбрать полярность для TI1FP1 (для захвата в TIMx_CCR1 и очистки счётчика): записать 0 в CC1P (передний фронт).
- Выбрать активный вход для TIMx_CCR2: записать 10 в биты CC2S регистра TIMx_CCMR1 (TI1).
- Выбрать полярность для TI1FP2 (для захвата в TIMx_CCR2): записать 1 в CC2P (задний фронт).
- Выбрать вход запуска: записать 101 в биты TS регистра TIMx_SMCR (TI1FP1).
- Поставить контроллер режима ведомого в сброс: записать 100 в биты SMS регистра TIMx_SMCR.
- Включить захват: поставить биты CC1E и CC2E в регистре TIMx_CCER.

Времянка режима ввода ШИМ.



1. Режим ШИМ может использоваться только с сигналами TIMx_CH1/TIMx_CH2 так как к контроллеру режима ведомого подключены только TI1FP1 и TI2FP2.

16.3.7. Принудительный вывод

В режиме вывода (биты CCxS = 00 в TIMx_CCMRx), каждый выходной сигнал сравнения (OCxREF и затем OCx/OCxN) можно программно поставить в активное или неактивное состояние, независимо от результата сравнения выходного регистра сравнения и счётчика.

Активными выходные сигналы (OCxREF/OCx) ставятся записью 101 в биты OCxM нужного регистра TIMx_CCMRx. Активный сигнал OCxREF всегда высокий, а OCx противоположен биту полярности CCxP.

Пример: CCxP=0 (OCx активный высокий) => OCx ставится высоким.

Сигнал OCxREF снимается записью 100 в биты OCxM регистра TIMx_CCMRx.

Но сравнение внутреннего регистра TIMx_CCRx со счётчиком всё равно выполняется, биты ставятся, генерируются запросы прерывания и DMA. См. ниже.

16.3.8. Режим сравнения выхода

Используется для управления выходным сигналом и определения истечения периода времени.

При совпадении регистра захвата/сравнения со счётчиком эта схема:

- Ставит выходную ножку в заданное состояние (биты OCxM в регистре TIMx_CCMRx) и полярность (бит CCxP в регистре TIMx_CCER). Ножка может хранить состояние(OCxM=000), стать активной (OCxM=001), неактивной (OCxM=010) или переключиться (OCxM=011).
- Ставит флаг прерывания (бит CCxIF в регистре TIMx_SR).
- При установленной маске прерывания (бит CCxIE в регистре TIMx_DIER) выдаёт запрос.

В регистр TIMx_CCRx можно писать как через регистр предзагрузки, так и без него (см. бит OCxPE в регистре TIMx_CCMRx).

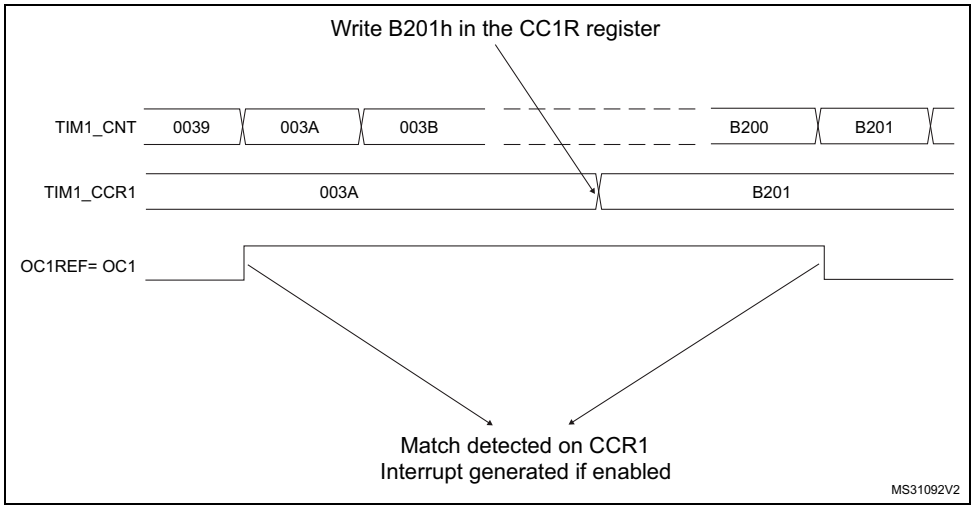
В режиме сравнения выхода бит события UEV на выходы OCxREF и OCx не влияет. Разрешение составляет один счёт счётчика. Этот режим можно использовать для выдачи одного импульса.

Процедура:

1. Выбрать такты счётчика (внутренние, внешние, предделитель).
2. Записать желаемое в регистры TIMx_ARR и TIMx_CCRx.
3. Если нужно прерывание, то установить бит CCxIE.
4. Выбрать режим выхода. Например:
   - Записать OCxM = 011 для переключения ножки OCx при совпадении CNT и CCRx
   - Записать OCxPE = 0 для отключения регистра предзагрузки
   - Записать CCxP = 0 ради высокого активного уровня
   - Записать CCxE = 1 для разрешения вывода
5. Включить счётчик установкой бита CEN в регистре TIMx_CR1.

При запрещённом регистре предзагрузки (OCxPE='0') регистр TIMx_CCRx можно обновлять в любое время, иначе он будет обновлён только по событию UEV.

Режим сравнения выхода, переключение ножки OC1.



16.3.9. Режим ШИМ

Частота сигнала ШИМ определяется регистром TIMx_ARR, а коэффициент заполнения регистром TIMx_CCRx.

Режим ШИМ может включаться независимо по одному на каждый выход OCx записью ‘110’ (ШИМ1) или ‘111’ (ШИМ2) в биты OCxM регистра TIMx_CCMRx. Соответствующий регистр предзагрузки должно включать битом OCxPE в регистре TIMx_CCMRx и, соответсвенно, регистр авто-загрузки (в режимах прямого и реверсивного счёта) нужно включить битом ARPE в регистре TIMx_CR1.

Поскольку регистр предзагрузки пишется в теневой регистр только по событию обновления, то все регистры нужно заранее инициализировать установкой бита UG в регистре TIMx_EGR.

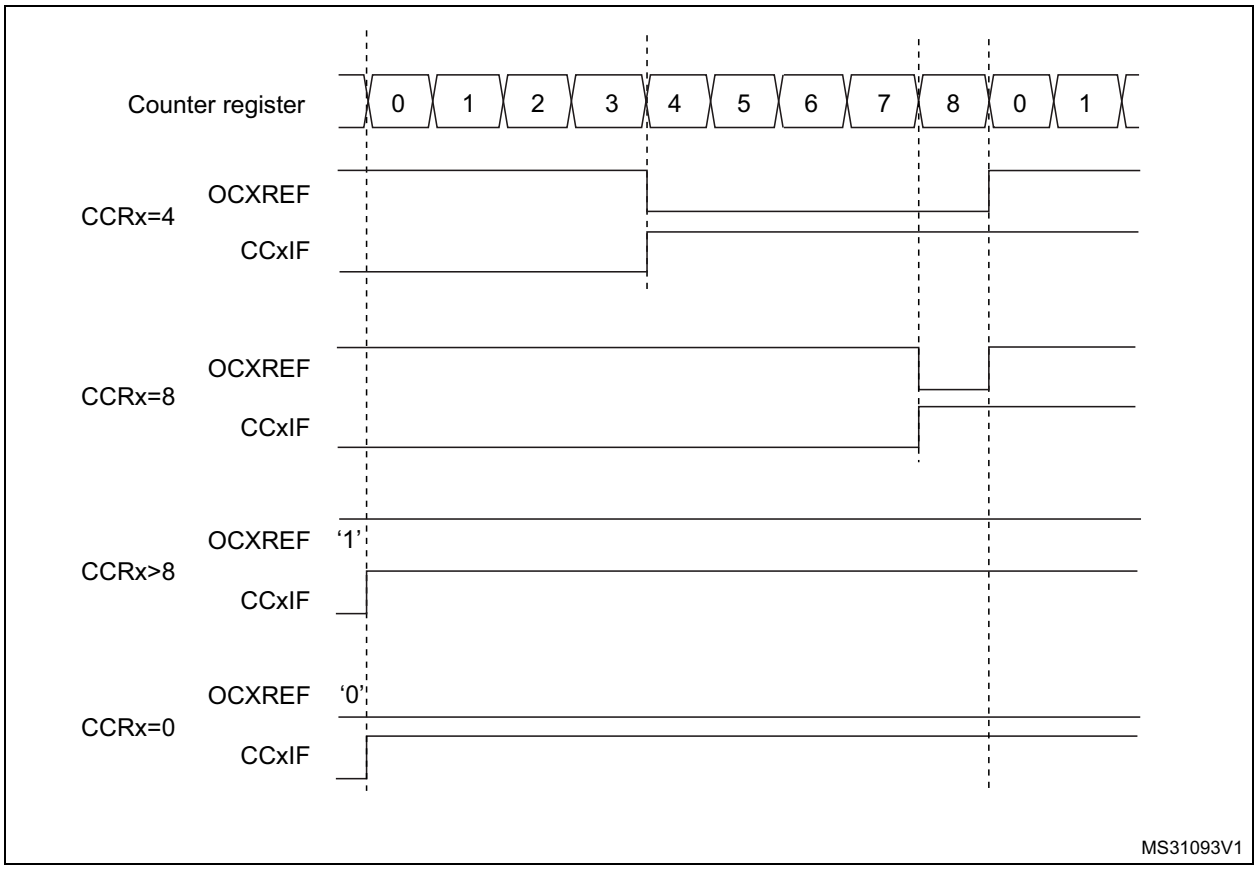
Полярность OCx (активный высокий или низкий) выбирается битом CCxP в регистре TIMx_CCER. Выходы OCx включаются битами CCxE, CCxNE, MOE, OSSI и OSSR в регистрах TIMx_CCER и TIMx_BDTR.

В режиме ШИМ (1 или 2), TIMx_CNT и TIMx_CCRx сравниваются на выполнение условия TIMx_CCRx ≤ TIMx_CNT (только при прямом счёте).

ШИМ по фронту

В примере ниже стоит режим ШИМ1. Опорный сигнал OCxREF остаётся высоким при TIMx_CNT < TIMx_CCRx. Если регистр TIMx_CCRx больше значения авто-загрузки (TIMx_ARR) то OCxREF равен ‘1’. При равенстве значений сравнения, OCxREF равен ‘0’. В примере ниже TIMx_ARR=8.

ШИМ по фронту (TIMx_ARR=8).



16.3.10.Режим одного импульса (OPM)

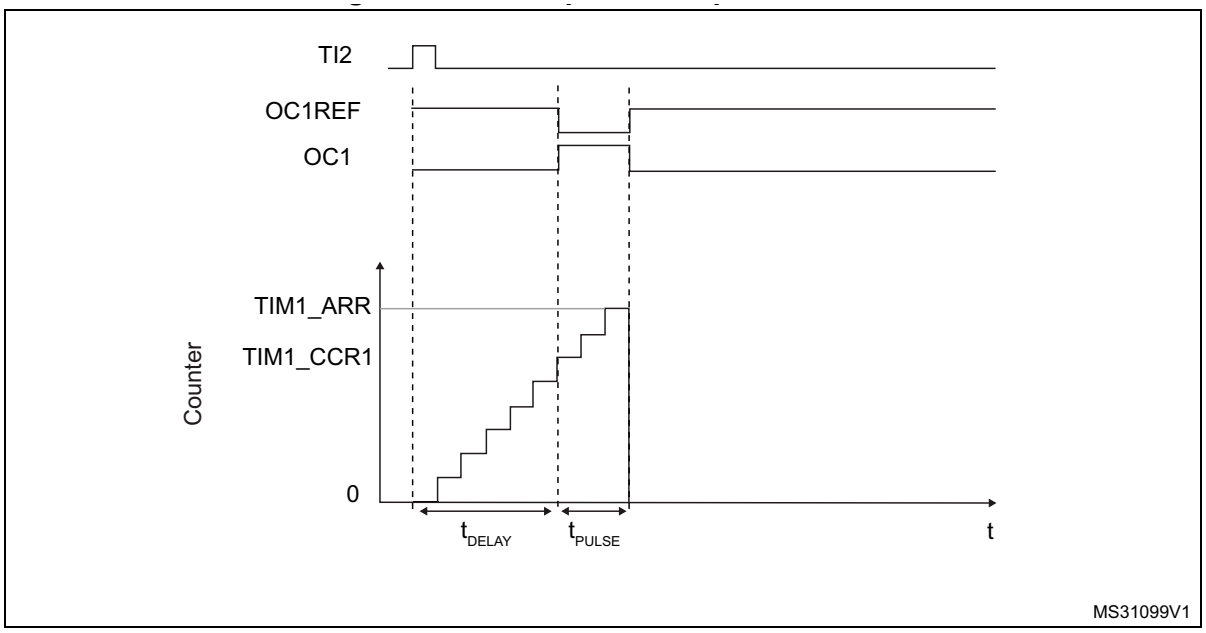
OPM это частный случай предыдущих режимов. Он позволяет счётчику запускаться по сигналу и генерировать импульс программируемой длины с программируемой задержкой.

Запускать счётчик можно из контроллера ведомого режима. Создавать сигнал можно в режиме сравнения выхода и ШИМ. Режим включается установкой бита OPM в регистре TIMx_CR1. Так счётчик автоматически останавливается по следующему событию UEV.

Импульс генерируется правильно только если значение сравнения отличается от начального значения счётчика. Перед стартом (таймер ждёт запуска) конфигурация должна быть:

CNT < CCRx ≤ ARR (в частности, 0 < CCRx)

Режим одного импульса.



Например, выдадим положительный импульс на OC1 длиной t_{PULSE} после задержки t_{DELAY} по переднему фронту на ножке TI2.

Берём TI2FP2 как запуск 1:

- Поставим TI2FP2 на TI2 записью CC2S='01' в регистре TIMx_CCMR1.
- TI2FP2 ставим на передний фронт записью CC2P='0' в регистре TIMx_CCER.
- Выбираем TI2FP2 как запуск контроллером режима ведомого (TRGI) записью TS='110' в регистр TIMx_SMCR.
- TI2FP2 используем как старт записью SMS='110' в TIMx_SMCR (режим запуска).

Сигнал OPM определяем записью регистров сравнения (учитываем частоту тактов и предделитель).

- Определяем t_{DELAY} записью в TIMx_CCR1.
- Ставим t_{PULSE} как разность значений перезагрузки и сравнения (TIMx_ARR - TIMx_CCR1+1).
- Сигнал создаётся переходом из '0' в '1' при сравнении и переходом из '1' в '0' при достижении счётчиком значения перезагрузки. Для этого, включаем режим ШИМ2 записью OC1M=111 в регистре TIMx_CCMR1. Регистры предзагрузки можно включить записью OC1PE='1' в регистре TIMx_CCMR1и ARPE в регистре TIMx_CR1, а можно не включать. В этом случае значение сравнения пишется в регистр TIMx_CCR1, значение перезагрузки в регистр TIMx_ARR, даём событие обновления битом UG и ждём внешнего запуска на TI2. В CC1P здесь пишется '0'.

Мы ждём только одного импульса (Однократный режим), так что для остановки счётчика по следующему событию обновления (счётчик переходит через значение перезагрузки и обнуляется) нужно поставить бит OPM регистра TIMx_CR1. Если бит OPM регистра TIMx_CR1 обнулён, то включается Повторяющийся режим.

Частный случай: быстрое включение OCx:

В этом режиме появление фронта на входе TIx ставит бит CEN, чем включает счётчик. Затем совпадение счётчика с значением сравнения переключает выходной сигнал. Но для этого нужны несколько тактов, ограничивая минимально возможное значение t_{DELAY}.

Если нужен сигнал с минимальной задержкой, то нужно ставить бит OCxFE в TIMx_CCMRx. Далее включаются OCxREF (и OCx) не глядя на сравнение. Его новый уровень равен уровню при сравнении. OCxFE работает только в режимах ШИМ1 и ШИМ2.

16.3.11.Синхронизация внешнего запуска TIM9/12

Есть три режима синхронизации TIMx: Сброс, Вентильный и Запуск.

Режим ведомого: Сброс

Счётчик и предделитель можно инициализировать по входу запуска. Кроме того, если бит URS в TIMx_CR1 сброшен, то генерируется событие UEV. Далее обновляются все предзагружаемые регистры (TIMx_ARR, TIMx_CCRx).

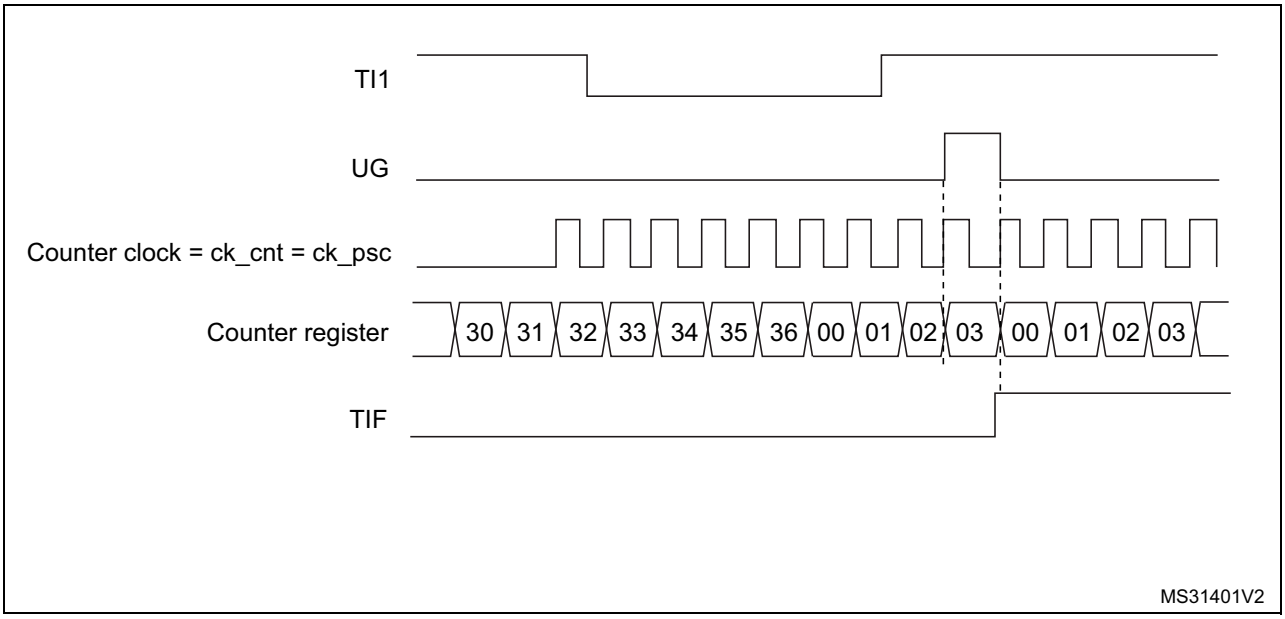
В этом примере счётчик прямого счёта сбрасывается по переднему фронту на входе TI1:

- Ставим канал 1 на передний фронт TI1. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC1S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC1P=0 в регистре TIMx_CCER определяет полярность и передний фронт.
- Ставим таймер в режим сброса записью SMS=100 в регистр TIMx_SMCR. Выбираем вход с TI1 записью TS=101 в регистре TIMx_SMCR.
- Запускаем счётчик записью CEN=1 в регистре TIMx_CR1.

Счётчик начинает считать внутренние такты вплоть переднего фронта TI1. Тогда счётчик сбрасывается с 0, ставится флаг запуска (бит TIF в регистре TIMx_SR) выдаются запросы прерывания и DMA (если разрешены битами TIE и TDE в регистре TIMx_DIER).

На рисунке ниже регистр перезагрузки TIMx_ARR=0x36. Задержка между фронтом TI1 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI1.

Режим Сброса.



Режим ведомого: Вентильный

Счётчик включается уровнем на выбранном входе.

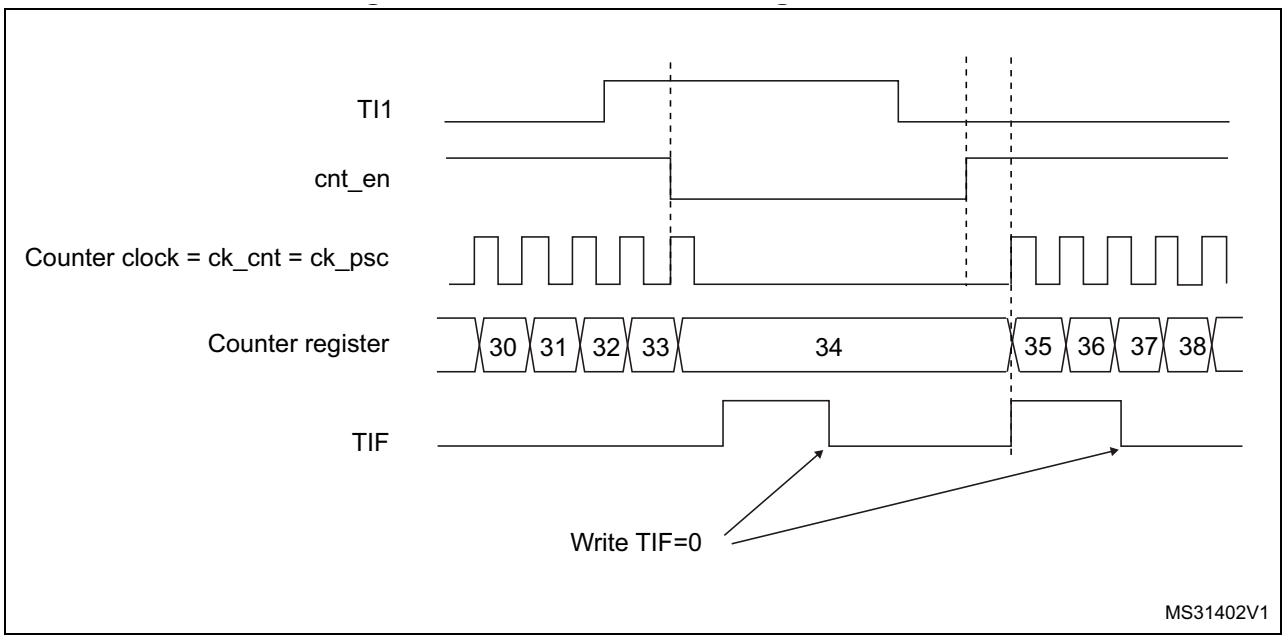
В этом примере счётчик прямого счёта работает только при низком уровне TI1:

- Ставим канал 1 на низкий уровень TI1. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC1S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC1P=1 в регистре TIMx_CCER определяет полярность и низкий уровень.
- Ставим таймер в вентильный режим записью SMS=101 в регистр TIMx_SMCR. Выбираем вход с TI1 записью TS=101 в регистре TIMx_SMCR.
- Запускаем счётчик записью CEN=1 в регистре TIMx_CR1. В этом режиме счётчик не работает при CEN=0, независимо от уровня на входе.

Счётчик начинает считать внутренние такты при низком уровне на TI1 и останавливается при высоком. Флаг запуска (бит TIF в регистре TIMx_SR) ставится при пуске и остановке таймера.

Задержка между фронтом TI1 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI1.

Вентильный режим.



Режим ведомого: Запуск

Счётчик включается событием на выбранном входе.

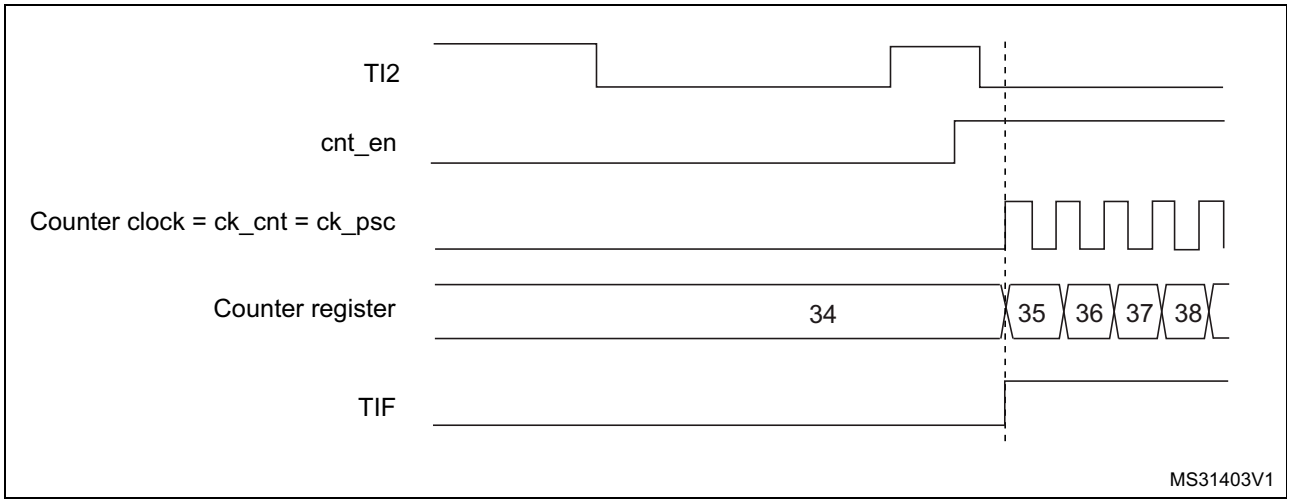
В этом примере счётчик прямого счёта работает по переднему фронту на `TI2`:

- Ставим канал 2 на передний фронт `TI2`. Входной фильтр отключён (`IC1F`=0000). Предделителя нет. Биты `CC2S=01` в регистре `TIMx_CCMR1` выбирают захват входа. Бит `CC2P`=1 в регистре `TIMx_CCER` определяет полярность и низкий уровень.
- Ставим таймер в режим запуска записью `SMS`=110 в регистр `TIMx_SMCR`. Выбираем вход с `TI2` записью `TS`=110 в регистре `TIMx_SMCR`.

Счётчик начинает считать внутренние такты по переднему фронту на `TI2` и ставит флаг запуска (бит `TIF` в регистре `TIMx_SR`).

Задержка между фронтом `TI2` и действительным сбросом появляется из-за схемы ресинхронизации на входе `TI2`.

Режим Запуска.



16.3.12.Синхронизация таймера TIM9/TIM12

Таймеры TIM связаны между собой для синхронизации или сцепления. См. *Секцию 15.3.15*.

NB: Такты ведомого таймера нужно включать до получения событий от ведущего и не должны меняться налету.

16.3.13.Режим отладки

В режиме отладки (ядро Cortex-M3 остановлено), счётчик TIMx либо работает, либо останавливается, в зависимости от бита DBG_TIMx_STOP в модуле DBG.

При стоящем счётчике (DBG_TIMx_STOP = 1 в регистре DBGMCU_APBx_FZ) выходы отключаются (как при сброшенном бите MOE). Выходы можно перевести в неактивное состояние (бит OSSI = 1), или возложить управление на контроллер GPIO (бит OSSI = 0) для перевода их в высокоимпедансное состояние (Hi-Z).

16.4. Регистры TIM9/12

32-бит регистры надо писать словами. Остальные регистры надо писать полусловами или словами. Читать можно байтами,полусловами или словами.

16.4.1. Регистр управления 1 TIM9/12 (TIMx_CR1)

Смещение адреса: 0x00

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
						rw	rw	rw				rw	rw	rw	rw

— Биты 15:10 Резерв, не трогать.

— Биты 9:8 **CKD[1:0]:** Деление тактов.

Это отношение тактов таймера (CK_INT) и тактов задержки и выборки (t_{DTS}) в генераторах задержки и цифровых фильтрах (ETR, TIx),

00: $t_{DTS}=t_{CK\_INT}$

01: $t_{DTS}=2*t_{CK\_INT}$

10: $t_{DTS}=4*t_{CK\_INT}$

11: Резерв

— Бит 7 **ARPE**: Разрешение буфера перезагрузки TIMx_ARR

0: Нельзя

1: Нужно

— Бит 6:4 Резерв, не трогать.

— Бит 3 **OPM**: Режим одного импульса

0: Счётчик не останавливается

1: Счётчик останавливается по следующему событию обновления (снимается бит CEN)

— Бит 2 **URS**: Источник запроса по событию UEV.

Изменяется программно.

0: Разрешённые запросы прерывания выдаются по:

– Переполнение счётчика

– Установка бита UG

1: Разрешённые запросы прерывания выдаются только по Переполнению счётчика.

— Бит 1 **UDIS:** Выключение выдачи события обновления UEV.

Изменяется программно.

0: Событие UEV выдаётся по:

– Переполнение счётчика

– Установка бита UG

1: Событие UEV не выдаётся, скрытые регистры (ARR, PSC, CCRx) хранят значение. При стоящем бите UG счётчик и предделитель инициализируются.

— Бит 0 **CEN:** Включение счётчика.

0: Выключен

1: Включён

NB: В режиме Одного импульса бит CEN снимается аппаратно по событию обновления.

16.4.2. Регистр управления режима ведомого TIM9/12 (TIMx_SMCR)

Смещение адреса: 0x08

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TS[2:0]			Res.	SMS[2:0]		
									rw	rw	rw		rw	rw	rw

— Биты 15:7 Резерв, не трогать.

— Биты 6:4 **TS[2:0]**: Выбор запуска

000: Внутренний 0 (ITR0)

001: Внутренний 1 (ITR1)

010: Внутренний 2 (ITR2)

011: Внутренний 3 (ITR3)

100: Детектор фронта TI1 (TI1F_ED)

101: Фильтрованный вход 1 (TI1FP1)

110: Фильтрованный вход 2 (TI2FP2)

111: Резерв

NB: Менять их можно только если не используются (например, при SMS=000) во избежание неверного определения фронта.

— Бит 3 Резерв, не трогать.

— Биты 2:0 **SMS[2:0]**: Выбор режима ведомого

При внешнем сигнале активный фронт TRGI подключается к внешнему входу с выбранной полярностью.

000: Режим ведомого выключен - если CEN = '1', то предделитель работает напрямую от внутренних тактов.

001: Резерв.

010: Резерв.

011: Резерв.

100: Режим сброса - Передний фронт TRGI инициализирует счётчик и запускает обновление регистров.

101: Вентильный режим - Такты счётчика разрешены при высоком TRGI. При низком TRGI счётчик останавливается, но не сбрасывается. Старт и стоп счётчика управляемые.

110: Режим запуска - Счётчик стартует по переднему фронту TRGI (но не сбрасывает). Управляемый только старт.

111: Внешние такты 1 - Счёт передних фронтов TRGI.

NB: Вентильный режим нельзя использовать если входом запуска выбран TI1F_ED (TS='100'). В действительности, TI1F_ED выдаёт 1 импульс на каждую передачу TI1F, тогда как вентильный режим проверяет уровень сигнала запуска.

Такты ведомого таймера нужно включать до приёма событий от ведущего таймера и не должны изменяться налету.

Таблица 89. Подключение внутреннего запуска TIMx.

Ведомый TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM9	TIM2_TRGO	TIM3_TRGO	TIM10_OC	TIM11_OC
TIM12	TIM4_TRGO	TIM5_TRGO	TIM13_OC	TIM14_OC

16.4.3. Регистр разрешения прерываний TIM9/TIM12 (TIMx_DIER)

Смещение адреса: 0x0C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TIE	Res			CC2IE	CC1IE	UIE
									rw				rw	rw	rw

Содержимое битов:

0: Нельзя

1: Можно

— Биты 15:7 Резерв, не трогать.

— Бит 6 **TIE**: Запрос прерывания запуска

— Биты 5:3 Резерв, не трогать.

— Бит 2 **CC2IE**: Запрос прерывания захвата/сравнения 2

— Бит 1 **CC1IE**: Запрос прерывания захвата/сравнения 1

— Бит 0 **UIE**: Запрос прерывания обновления

16.4.4. Регистр состояния TIM9/TIM12 (TIMx_SR)

Биты ставятся аппаратно, стираются записью 0.

Перезахват это появление события при стоящем его флаге.

Смещение адреса: 0x10

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CC2OF	CC1OF	Reserved		TIF	Reserved			CC2IF	CC1IF	UIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

Содержимое битов:

0: Не было

1: Было

— Биты 15:11 Резерв, не трогать.

— Бит 10 **CC2OF**: Флаг перезахвата захвата/сравнения 2

— Бит 9 **CC1OF**: Флаг перезахвата захвата/сравнения 1

— Бит 8:7 Резерв, не трогать.

— Бит 6 **TIF**: Флаг прерывания запуска

Ставится аппаратно по активному фронту на входе TRGI при работающем контроллере режима ведомого во всех режимах, кроме Вентильного, (тогда по обоим фронтам).

— Биты 5:3 Резерв, не трогать.

— Бит 2 **CC2IF**: Флаг прерывания захвата/сравнения 2

— Бит 1 **CC1IF**: Флаг прерывания захвата/сравнения 1

В режиме вывода CC1:

0: Не было

1: Счётчик TIMx_CNT сравнивается с TIMx_CCR1. При TIMx_CCR1 > TIMx_ARR, флаг CC1IF ставится по переполнению счётчика.

В режиме ввода CC1:

0: Не было

1: В TIMx_CCR1 лежит захваченное значение.

— Бит 0 **UIF**: Флаг прерывания обновления

Удержание прерывания обновления ставится при:

– Переполнении счётчика и UDIS=0 в регистре TIMx_CR1.

– При программной инициализации CNT битом UG в регистре TIMx_EGR, если URS=0 и UDIS=0 в регистре TIMx_CR1.

– При инициализации CNT событием запуска (см. регистр TIMx_SMCR), если URS=0 и UDIS=0 в регистре TIMx_CR1.

16.4.5. Регистр генерации событий (TIMx_EGR)

Событие генерируется программной установкой бита, снимается он аппаратно.

Смещение адреса: 0x14

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TG	Reserved			CC2G	CC1G	UG
									w				w	w	w

— Биты 15:7 Резерв, не трогать.

— Бит 6 **TG**: Запуск.

1: Ставит флаг TIF в регистре TIMx_SR. Может возникнуть прерывание.

— Биты 5:3 Резерв, не трогать.

— Бит 2 **CC2G**: Захват/сравнение 2.

— Бит 1 **CC1G**: Захват/сравнение 1.

0: Ничего.

1: Выдаёт событие захвата/сравнения:

В режиме вывода CC1:

Ставит флаг CC1IF. Может возникнуть прерывание.

В режиме ввода CC1:

Текущее значение счётчика пишется в TIMx_CCR1. Ставит флаг CC1IF. Может возникнуть прерывание. При уже стоящем флаге CC1IF ставится и флаг CC1OF.

— Бит 0 **UG**: Обновление.

1: Инициализирует счётчик и обновление регистров. Текущий счётчик предделителя чистится не влияя на само значение. В режиме по центру или при DIR=0 счётчик обнуляется, иначе перегружается из регистра TIMx_ARR (при DIR=1).

16.4.6. Регистр режима захвата/сравнения 1 TIM9/TIM12 (TIMx_CCMR1)

Смещение адреса: 0x18

По сбросу: 0x0000

Каналы могут работать на ввод (захват) и вывод (сравнение), что определяется битами `CCxS`. У остальных битов для ввода и вывода значение отличается. `OCxx` обозначает функции вывода, `ICxx` функции ввода. При этом используются разные каскады.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Сравнение выхода:

— Бит 15 **OC2CE:** Разрешение очистки сравнения 2.

— Биты 14:12 **OC2M[2:0]**: Режим сравнения выхода 2.

— Бит 11 **OC2PE**: Разрешение предзагрузки сравнения 2.

— Бит 10 **OC2FE**: Быстрое разрешение сравнения 2.

— Биты 9:8 **CC2S[1:0]**: Выбор Захвата/Сравнения 2.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC2 выходной

01: Канал CC2 входной, IC2 на TI2

10: Канал CC2 входной, IC2 на TI1

11: Канал CC2 входной, IC2 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC2S можно писать только при выключенном канале (CC2E = ‘0’ в TIMx_CCER).

— Бит 7 **OC1CE**: Разрешение очистки сравнения выхода 1.

0: ETRF на OC1REF не влияет

1: Высокий уровень на ETRF стирает OC1REF

— Биты 6:4 **OC1M[2:0]**: Режим сравнения выхода 1.

Определяют поведение выходного опорного сигнала OC1REF, от которого происходят OC1 и OC1N. OC1REF активен высоким, а OC1 и OC1N зависят от битов CC1P и CC1NP.

000: Заморозка - сравнение TIMx_CCR1 и TIMx_CNT на выходы не влияет (режим используется для генерации временной базы).

001: При совпадении TIMx_CNT и TIMx_CCR1 сигнал OC1REF ставится активным (высоким).

010: При совпадении TIMx_CNT и TIMx_CCR1 сигнал OC1REF ставится неактивным (низким).

011: Переключение - при TIMx_CNT=TIMx_CCR1 сигнал OC1REF перебрасывается.

100: Принудительный неактивный - OC1REF ставится низким.

101: Принудительный активный - OC1REF ставится высоким.

110: ШИМ 1 - при прямом счёте канал активен пока TIMx_CNT<TIMx_CCR1. При обратном счёте канал неактивен (OC1REF='0') пока TIMx_CNT>TIMx_CCR1.

111: ШИМ 2 - при прямом счёте канал неактивен пока TIMx_CNT<TIMx_CCR1. При обратном счёте канал активен пока TIMx_CNT>TIMx_CCR1.

NB: В ШИМ 1 и 2 уровень OCREF изменяется только когда изменяется результат сравнения или при переключении из "Заморозки" в "ШИМ".

— Бит 3 **OC1PE**: Разрешение регистра предзагрузки.

0: Предзагрузка TIMx_CCR1 выключена. TIMx_CCR1 можно писать в любое время, новое значение начинает действовать незамедлительно.

1: Предзагрузка TIMx_CCR1 включена. Доступ идёт к регистру предзагрузки. TIMx_CCR1 пишется в активный регистр по событию обновления.

NB: Режим ШИМ без установки регистра предзагрузки можно использовать только в режиме одного импульса (стоит бит OPM в регистре TIMx_CR1). Иначе поведение непредсказуемо.

— Бит 2 **OC1FE**: Разрешение быстрого сравнения.

Ускоряет действие события запуска по входу на выход CC.

0: CC1 работает как обычно (сравнение счётчика и CCR1) даже при включённом запуске. Минимальная задержка переключения выхода CC1 при фронте на входе составляет 5 тактов.

1: Активный фронт запуска по входу действует как сравнение на выходе CC1. Тогда OC ставится в уровень сравнения независимо от его результата. Задержка между импульсом на входе и выходом CC1 сокращается до 3 тактов. OCFE работает только в ШИМ 1 и ШИМ 2.

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC1 выходной

01: Канал CC1 входной, IC1 на TI1

10: Канал CC1 входной, IC1 на TI2

11: Канал CC1 входной, IC1 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC1S можно писать только при выключенном канале (CC1E = '0' в TIMx_CCER).

Захват входа.

— Биты 15:12 **IC2F**: Фильтр захвата входа 2.

— Биты 11:10 **IC2PSC[1:0]** : Предделитель захвата входа 2.

— Биты 9:8 **CC2S[1:0]**: Выбор Захвата/Сравнения 2.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC2 выходной

01: Канал CC2 входной, IC2 на TI2

10: Канал CC2 входной, IC2 на TI1

11: Канал CC2 входной, IC2 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC2S можно писать только при выключенном канале (CC2E = '0' в TIMx_CCER).

— Биты 7:4 **IC1F[3:0]**: Фильтр захвата входа 1.

Определяет частоту выборки и длину цифрового фильтра сигнала TI1. Цифровой фильтр это счётчик, выдающий сигнал на выход после N последовательных событий на входе:

0000: Без фильтра, выборка на частоте f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

— Биты 3:2 **IC1PSC**: Предделитель захвата входа 1
Значение предделителя на входе CC1 (IC1).
По CC1E=’0’ (регистр TIMx_CCER) предделитель сбрасывается.
00: без предделителя, по каждому фронту на входе
01: каждые 2 события
10: каждые 4 события
11: каждые 8 события

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.
Определяет направление канала (ввод/вывод) и используемый вход.
00: Канал CC1 выходной
01: Канал CC1 входной, IC1 на TI1
10: Канал CC1 входной, IC1 на TI2
11: Канал CC1 входной, IC1 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.
NB: Биты CC1S можно писать только при выключенном канале (CC1E = ‘0’ в TIMx_CCER).

16.4.7. Регистр разрешения захвата/сравнения (TIMx_CCER)

Смещение адреса: 0x20
По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

— Биты 15:8 Резерв, не трогать.
— Бит 7 **CC2NP**: Полярность инверсного выхода захвата/сравнения 2
— Бит 6 Резерв, не трогать.
— Бит 5 **CC2P**: Полярность выхода захвата/сравнения 2
— Бит 4 **CC2E**: Разрешение выхода захвата/сравнения 2
— Бит 3 **CC1NP**: Полярность инверсного выхода захвата/сравнения 1
Выходной CC1: CC1NP должен быть чистым
Входной CC1: CC1NP вместе с CC1P определяют полярность TI1FP1/TI2FP1 (см. CC1P).
— Бит 2 Резерв, не трогать.
— Бит 1 **CC1P**: Полярность выхода захвата/сравнения 1
1: Выдаёт событие захвата/сравнения:
В режиме вывода CC1:
0: OC1 активен высоким.
1: OC1 активен низким.
В режиме ввода CC1:
Выбирает для запуска или захвата прямой или инверсный IC1.
Биты CC1NP/CC1P определяют полярность TI1FP1 TI2FP1 запуска операции.
00: прямой/передний фронт и уровень
01: обратный/задний фронт и уровень
10: резерв.
11: неинверсные оба фронта и уровень
— Бит 0 **CC1E**: Разрешение выхода захвата/сравнения 1
В режиме вывода CC1:
0: Выкл. - OC1 не активен.
1: Вкл. - OC1 выводится на соответствующую ножку.
В режиме ввода CC1:
0: Захват выключен.
1: Захват включен.

Таблица 90. Управляющие биты стандартных OCx каналов

Бит CCxE	Состояние OCx
0	Выключен (OCx=0, OCx_EN=0)
1	OCx=OCxREF + Полярность, OCx_EN=1

NB: Состояние внешних ножек I/O, подключённых к стандартным OCx, зависит от состояния OCx и регистров GPIO и AFIO.

16.4.8. Счётчик TIM9/12 (TIMx_CNT)

Смещение адреса: 0x24

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **CNT[15:0]**: Значение счётчика.

16.4.9. Предделитель TIM9/12 (TIMx_PSC)

Смещение адреса: 0x28

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **PSC[15:0]**: Значение предделителя

Частота счёта (CK_CNT) равна f_{CK_PSC} / (PSC[15:0] + 1).

Значение PSC пишется в рабочий регистр предделителя по каждому событию обновления (включая очистку счётчика битом UG регистра TIMx_EGR или запуском в режиме Сброса).

16.4.10.Регистр предзагрузки TIM9/12 (TIMx_ARR)

Смещение адреса: 0x2C

По сбросу: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **ARR[15:0]**: Значение перезагрузки.

Если значение нулевое, то счётчик блокируется.

16.4.11.Регистр 1 захвата/сравнения TIM9/12 (TIMx_CCR1)

Смещение адреса: 0x34

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR1[15:0]**: Значение захвата/сравнения

В режиме вывода CC1:

Значение предзагрузки CCR1 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC1PE в регистре TIMx_CCMR1), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC1.

В режиме ввода CC1:

CCR1 содержит значение счётчика по последнему событию захвата 1 (IC1). Здесь TIMx_CCR1 только читается.

16.4.12.Регистр 2 захвата/сравнения TIM9/12 (TIMx_CCR2)

Смещение адреса: 0x38

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR2[15:0]**: Значение захвата/сравнения

В режиме вывода CC2:

Значение предзагрузки CCR2 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC2PE в регистре TIMx_CCMR2), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC2.

В режиме ввода CC2:

CCR2 содержит значение счётчика по последнему событию захвата 2 (IC2). Здесь TIMx_CCR2 только читается.

16.4.13.Карта регистров TIM9/12

0x00	**TIMx_CR1**	Reserved							CKD [1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value								0	0	0				0	0	0	0
0x08	**TIMx_SMCR**	Reserved										TS[2:0]			Reserved	SMS[2:0]		
	Reset value											0	0	0		0	0	0
0x0C	**TIMx_DIER**	Reserved										TIE	Reserved			CC2IE	CC1IE	UIE
	Reset value											0				0	0	0
0x10	**TIMx_SR**	Reserved						CC2OF	CC1OF	Reserved		TIF	Reserved			CC2IF	CC1IF	UIF
	Reset value							0	0			0				0	0	0
0x14	**TIMx_EGR**	Reserved										TG	Reserved			CC2G	CC1G	UG
	Reset value											0				0	0	0
0x18	**TIMx_CCMR1** Output Compare mode	Reserved		OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Reserved	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value			0	0	0	0	0	0	0		0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Reserved	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	Reserved																	
0x20	**TIMx_CCER**	Reserved									CC2NP	Reserved	CC2P	CC2E	CC1NP	Reserved	CC1P	CC1E
	Reset value										0		0	0	0		0	0
0x24	**TIMx_CNT**	Reserved	CNT[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Reserved	PSC[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Reserved	ARR[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved																	
0x34	**TIMx_CCR1**	Reserved	CCR1[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIMx_CCR2**	Reserved	CCR2[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C to 0x4C	Reserved																	

16.5. Регистры TIM10/11/13/14

32-бит регистры надо писать словами. Остальные регистры надо писать полусловами или словами. Читать можно байтами,полусловами или словами.

16.5.1. Регистр управления 1 TIM10/11/13/14 (TIMx_CR1)

Смещение адреса: 0x00

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
						rw	rw	rw				rw	rw	rw	rw

— Биты 15:10 Резерв, не трогать.

— Биты 9:8 **CKD[1:0]:** Деление тактов.

Это отношение тактов таймера (CK_INT) и тактов задержки и выборки (t_{DTS}) в генераторах задержки и цифровых фильтрах (ETR, TIx),

00: $t_{DTS}=t_{CK\_INT}$

01: $t_{DTS}=2*t_{CK\_INT}$

10: $t_{DTS}=4*t_{CK\_INT}$

11: Резерв

— Бит 7 **ARPE**: Разрешение буфера перезагрузки TIMx_ARR

0: Нельзя

1: Нужно

— Бит 6:4 Резерв, не трогать.

— Бит 3 **OPM**: Режим одного импульса

0: Счётчик не останавливается

1: Счётчик останавливается по следующему событию обновления (снимается бит CEN)

— Бит 2 **URS**: Источник запроса по событию UEV.

Изменяется программно.

0: Разрешённые запросы прерывания выдаются по:

– Переполнение счётчика

– Установка бита UG

1: Разрешённые запросы прерывания выдаются только по Переполнению счётчика.

— Бит 1 **UDIS:** Выключение выдачи события обновления UEV.

Изменяется программно.

0: Событие UEV выдаётся по:

– Переполнение счётчика

– Установка бита UG

1: Событие UEV не выдаётся, скрытые регистры (ARR, PSC, CCRx) хранят значение. При стоящем бите UG счётчик и предделитель инициализируются.

— Бит 0 **CEN:** Включение счётчика.

0: Выключен

1: Включён

NB: В режиме Одного импульса бит CEN снимается аппаратно по событию обновления.

16.5.2. Регистр разрешения прерываний TIM10/11/13/14 (TIMx_DIER)

Смещение адреса: 0x0C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1IE	UIE
														rw	rw

Содержимое битов:

0: Нельзя

1: Можно

— Биты 15:2 Резерв, не трогать.

— Бит 6 **TIE**: Запрос прерывания запуска

— Бит 1 **CC1IE**: Запрос прерывания захвата/сравнения 1

— Бит 0 **UIE**: Запрос прерывания обновления

16.5.3. Регистр состояния TIM10/11/13/14 (TIMx_SR)

Биты ставятся аппаратно, стираются записью 0.

Перезахват это появление события при стоящем его флаге.

Смещение адреса: 0x10

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CC1OF	Reserved							CC1IF	UIF
						rc_w0								rc_w0	rc_w0

Содержимое битов:

0: Не было

1: Было

— Биты 15:10 Резерв, не трогать.

— Бит 9 **CC1OF**: Флаг перезахвата захвата/сравнения 1

— Бит 8:3 Резерв, не трогать.

— Бит 2 **CC2IF**: Флаг прерывания захвата/сравнения 2

— Бит 1 **CC1IF**: Флаг прерывания захвата/сравнения 1

В режиме вывода CC1:

0: Не было

1: Счётчик TIMx_CNT сравнивается с TIMx_CCR1. При TIMx_CCR1 > TIMx_ARR, флаг CC1IF ставится по переполнению счётчика.

В режиме ввода CC1:

0: Не было

1: В TIMx_CCR1 лежит захваченное значение.

— Бит 0 **UIF**: Флаг прерывания обновления

Удержание прерывания обновления ставится при:

– Переполнении и UDIS=0 в регистре TIMx_CR1.

– При программной инициализации CNT битом UG в регистре TIMx_EGR, если URS=0 и UDIS=0 в регистре TIMx_CR1.

16.5.4. Регистр генерации событий TIM10/11/13/14 (TIMx_EGR)

Событие генерируется программной установкой бита, снимается он аппаратно.

Смещение адреса: 0x14

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1G	UG
														w	w

— Биты 15:2 Резерв, не трогать.

— Бит 1 **CC1G**: Захват/сравнение 1.

0: Ничего.

1: Выдаёт событие захвата/сравнения:

В режиме вывода CC1:

Ставит флаг CC1IF. Может возникнуть прерывание.

В режиме ввода CC1:

Текущее значение счётчика пишется в TIMx_CCR1. Ставит флаг CC1IF. Может возникнуть прерывание. При уже стоящем флаге CC1IF ставится и флаг CC1OF.

— Бит 0 **UG**: Обновление.

1: Инициализирует счётчик и обновление регистров. Текущий счётчик предделителя чистится не влияя на само значение.

16.5.5. Регистр режима захвата/сравнения TIM10/11/13/14 (TIMx_CCMR1)

Смещение адреса: 0x18

По сбросу: 0x0000

Каналы могут работать на ввод (захват) и вывод (сравнение), что определяется битами CCxS. У остальных битов для ввода и вывода значение отличается. OCxx обозначает функции вывода, ICxx функции ввода. При этом используются разные каскады.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OC1M[2:0]				OC1PE	OC1FE	CC1S[1:0]	
Reserved								IC1F[3:0]				IC1PSC[1:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Сравнение выхода:

— Бит 15:7 Резерв, не трогать.

— Биты 6:4 **OC1M[2:0]**: Режим сравнения выхода 1.

Определяют поведение выходного опорного сигнала OC1REF, от которого происходят OC1 и OC1N. OC1REF активен высоким, а OC1 и OC1N зависят от битов CC1P и CC1NP.

000: Заморозка - сравнение TIMx_CCR1 и TIMx_CNT на выходы не влияет (режим используется для генерации временной базы).

001: При совпадении TIMx_CNT и TIMx_CCR1 сигнал OC1REF ставится активным (высоким).

010: При совпадении TIMx_CNT и TIMx_CCR1 сигнал OC1REF ставится неактивным (низким).

011: Переключение - при TIMx_CNT=TIMx_CCR1 сигнал OC1REF перебрасывается.

100: Принудительный неактивный - OC1REF ставится низким.

101: Принудительный активный - OC1REF ставится высоким.

110: ШИМ 1 - при прямом счёте канал активен пока TIMx_CNT<TIMx_CCR1. При обратном счёте канал неактивен (OC1REF='0') пока TIMx_CNT>TIMx_CCR1.

111: ШИМ 2 - при прямом счёте канал неактивен пока TIMx_CNT<TIMx_CCR1. При обратном счёте канал активен пока TIMx_CNT>TIMx_CCR1.

NB: В ШИМ 1 и 2 уровень OCREF изменяется только когда изменяется результат сравнения или при переключении из "Заморозки" в "ШИМ".

— Бит 3 **OC1PE**: Разрешение регистра предзагрузки.

0: Предзагрузка TIMx_CCR1 выключена. TIMx_CCR1 можно писать в любое время, новое значение начинает действовать незамедлительно.

1: Предзагрузка TIMx_CCR1 включена. Доступ идёт к регистру предзагрузки. TIMx_CCR1 пишется в активный регистр по событию обновления.

NB: Режим ШИМ без установки регистра предзагрузки можно использовать только в режиме одного импульса (стоит бит OPM в регистре TIMx_CR1). Иначе поведение непредсказуемо.

— Бит 2 **OC1FE**: Разрешение быстрого сравнения.

Ускоряет действие события запуска по входу на выход CC.

0: CC1 работает как обычно (сравнение счётчика и CCR1) даже при включённом запуске. Минимальная задержка переключения выхода CC1 при фронте на входе составляет 5 тактов.

1: Активный фронт запуска по входу действует как сравнение на выходе CC1. Тогда OC ставится в уровень сравнения независимо от его результата. Задержка между импульсом на входе и выходом CC1 сокращается до 3 тактов. OCFE работает только в ШИМ 1 и ШИМ 2.

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC1 выходной

01: Канал CC1 входной, IC1 на TI1

10: Канал CC1 входной, IC1 на TI2

11: Канал CC1 входной, IC1 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC1S можно писать только при выключенном канале (CC1E = '0' в TIMx_CCER).

Захват входа.

— Биты 15:8 Резерв, не трогать.

— Биты 7:4 **IC1F[3:0]**: Фильтр захвата входа 1.

Определяет частоту выборки и длину цифрового фильтра сигнала TI1. Цифровой фильтр это счётчик, выдающий сигнал на выход после N последовательных событий на входе:

0000: Без фильтра, выборка на частоте f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

— Биты 3:2 **IC1PSC**: Предделитель захвата входа 1
Значение предделителя на входе CC1 (IC1).
По CC1E='0' (регистр TIMx_CCER) предделитель сбрасывается.
00: без предделителя, по каждому фронту на входе
01: каждые 2 события
10: каждые 4 события
11: каждые 8 событий

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.
Определяет направление канала (ввод/вывод) и используемый вход.
00: Канал CC1 выходной
01: Канал CC1 входной, IC1 на TI1
10: Канал CC1 входной, IC1 на TI2
11: Канал CC1 входной, IC1 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.
NB: Биты CC1S можно писать только при выключенном канале (CC1E = '0' в TIMx_CCER).

16.5.6. Разрешение захвата/сравнения TIM10/11/13/14 (TIMx_CCER)

Смещение адреса: 0x20
По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CC1NP	Res.	CC1P	CC1E
												rw		rw	rw

— Биты 15:4 Резерв, не трогать.
— Бит 3 **CC1NP**: Полярность инверсного выхода захвата/сравнения 1
Выходной CC1: CC1NP должен быть чистым
Входной CC1: CC1NP вместе с CC1P определяют TI1FP1/TI2FP1 полярность (см. CC1P).
— Бит 2 Резерв, не трогать.
— Бит 1 **CC1P**: Полярность выхода захвата/сравнения 1
1: Выдаёт событие захвата/сравнения:
В режиме вывода CC1:
0: OC1 активен высоким.
1: OC1 активен низким.
В режиме ввода CC1:
Выбирает для запуска или захвата прямой или инверсный IC1.
Биты CC1NP/CC1P определяют полярность TI1FP1 TI2FP1 запуска операции.
00: прямой/передний фронт и уровень
01: обратный/задний фронт и уровень
10: резерв.
11: неинверсные оба фронта и уровень

— Бит 0 **CC1E**: Разрешение выхода захвата/сравнения 1

В режиме вывода CC1:

0: Выкл. - OC1 не активен.

1: Вкл. - OC1 выводится на соответствующую ножку.

В режиме ввода CC1:

0: Захват выключен.

1: Захват включен.

Таблица 90. Управляющие биты стандартных OCx каналов

Бит CCxE	Состояние OCx
0	Выключен (OCx=0, OCx_EN=0)
1	OCx=OCxREF + Полярность, OCx_EN=1

NB: Состояние внешних ножек I/O, подключённых к стандартным OCx, зависит от состояния OCx и регистров GPIO и AFIO.

16.5.7. Счётчик TIM10/11/13/14 (TIMx_CNT)

Смещение адреса: 0x24

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **CNT[15:0]**: Значение счётчика.

16.5.8. Предделитель TIM10/11/13/14 (TIMx_PSC)

Смещение адреса: 0x28

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **PSC[15:0]**: Значение предделителя

Частота счёта (CK_CNT) равна f_{CK_PSC} / (PSC[15:0] + 1).

Значение PSC пишется в рабочий регистр предделителя по каждому событию обновления (включая очистку счётчика битом UG регистра TIMx_EGR или запуском в режиме Сброса).

16.5.9. Регистр предзагрузки TIM10/11/13/14 (TIMx_ARR)

Смещение адреса: 0x2C

По сбросу: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **ARR[15:0]**: Значение перезагрузки.

Если значение нулевое, то счётчик блокируется.

16.5.10.Регистр 1 захвата/сравнения TIM10/11/13/14 (TIMx_CCR1)

Смещение адреса: 0x34

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR1[15:0]**: Значение захвата/сравнения

В режиме вывода CC1:

Значение предзагрузки CCR1 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC1PE в регистре TIMx_CCMR1), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC1.

В режиме ввода CC1:

CCR1 содержит значение счётчика по последнему событию захвата 1 (IC1). Здесь TIMx_CCR1 только читается.

16.5.11.Карта регистров TIM10/11/13/14

Offset	Register	31:16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Reserved							CKD [1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value								0	0	0				0	0	0	0
0x08	**TIMx_SMCR**	Reserved																
	Reset value																	
0x0C	**TIMx_DIER**	Reserved															CC1IE	UIE
	Reset value																0	0
0x10	**TIMx_SR**	Reserved							CC1OF	Reserved							CC1IF	UIF
	Reset value								0								0	0
0x14	**TIMx_EGR**	Reserved															CC1G	UG
	Reset value																0	0
0x18	**TIMx_CCMR1** Output compare mode	Reserved										OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value											0	0	0	0	0	0	0
	TIMx_CCMR1 Input capture mode	Reserved									IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value										0	0	0	0	0	0	0	0
0x1C	Reserved																	
0x20	**TIMx_CCER**	Reserved													CC1NP	Reserved	CC1P	CC1E
	Reset value														0		0	0
0x24	**TIMx_CNT**	Reserved	CNT[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Reserved	PSC[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Reserved	ARR[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved																	
0x34	**TIMx_CCR1**	Reserved	CCR1[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 to 0x4C	Reserved																	

17. Базовые таймеры (TIM6 и TIM7)

17.1. Введение в TIM6 и TIM7

Это 16-бит само-перегружаемые таймеры с программируемым предделителем.

Они могут использоваться для создания временной базы и управления ЦАП через подключённые входы запуска.

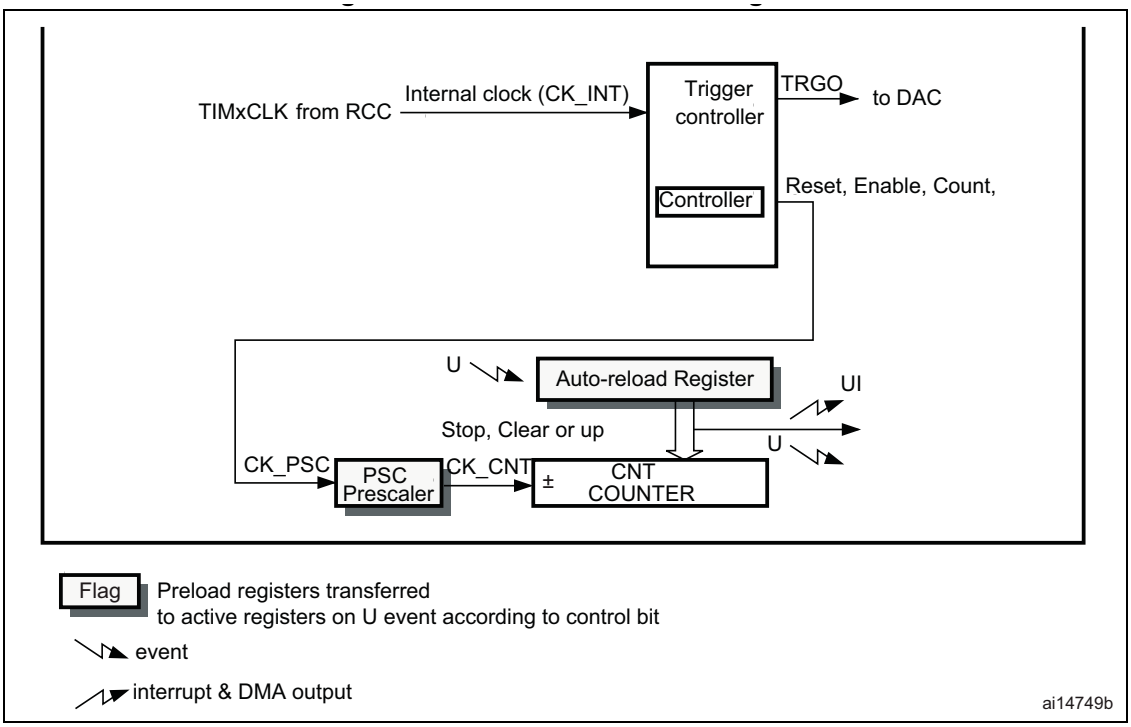
Таймеры полностью независимы и не имеют общих ресурсов.

17.2. Основные свойства TIM6 и TIM7

Это:

- 16-бит прямой счётчик с перезагрузкой.
- 16-бит программируемый предделитель (можно “налету”) тактов на число от 1 до 65536.
- Схема синхронизации запуска ЦАП.
- Генерация Прерываний/DMA по переполнению счётчика.

Блок-схема базового таймера.



17.3. Функциональное описание таймеров TIM6 и TIM7

17.3.1. Узел счёта

Это 16-бит счётчик с регистром перезагрузки. Тактирование может поступать через предделитель.

Счётчик, регистр перезагрузки и предделитель доступны программно для чтения и записи даже во время счёта.

Включает:

- Регистр счётчика (`TIMx_CNT`)
- Регистр предделителя (`TIMx_PSC`)
- Регистр перезагрузки (`TIMx_ARR`)

Регистр перезагрузки предзагружаемый (есть видимый и скрытый регистры). Содержимое видимого регистра пишется в скрытый постоянно или по каждому событию обновления (`UEV`), в зависимости от бита разрешения (`ARPE`) в регистре `TIMx_CR1`. Оно посылается программно или при переполнении/исчерпании счётчика при снятом бите `UDIS` в регистре `TIMx_CR1`.

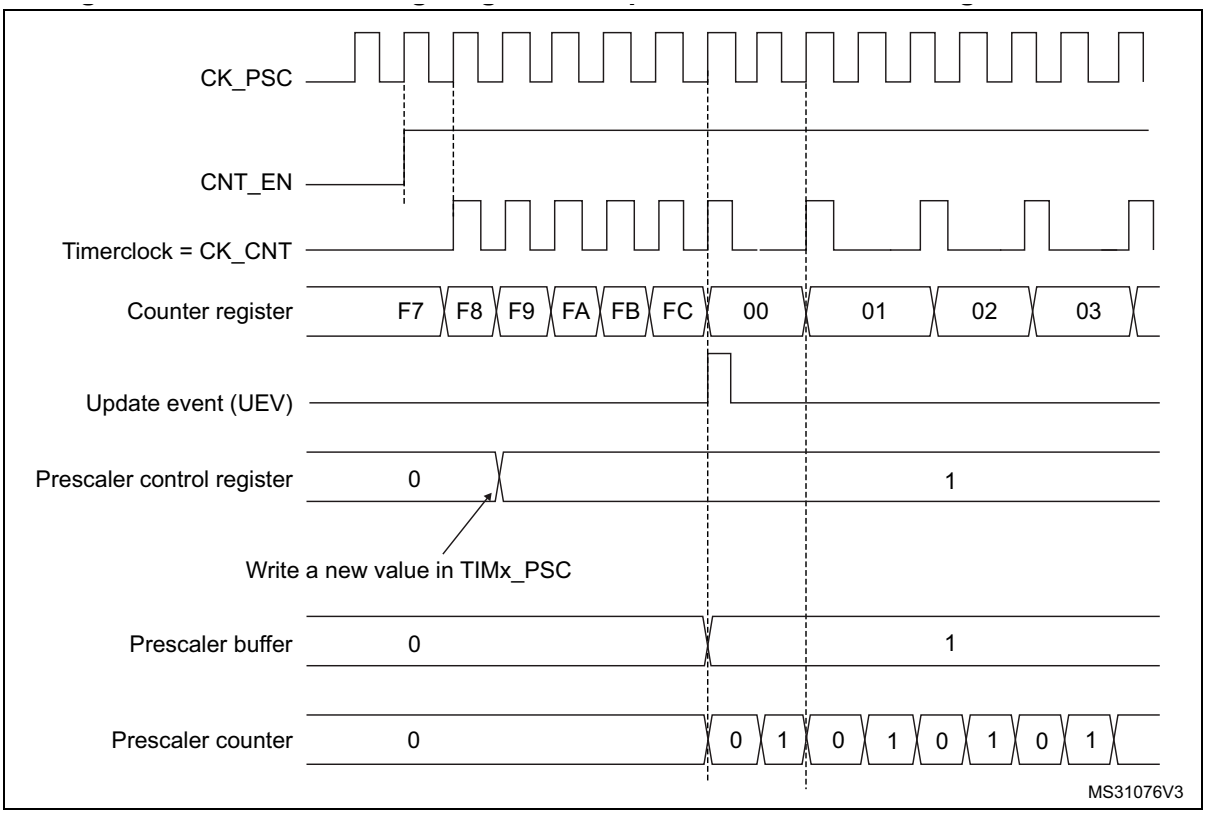
Счётчик тактируется выходом предделителя `CK_CNT`. Разрешается битом `CEN` в регистре `TIMx_CR1`.

Счёт начинается через 1 такт после установки бита `CEN` в регистре `TIMx_CR1`.

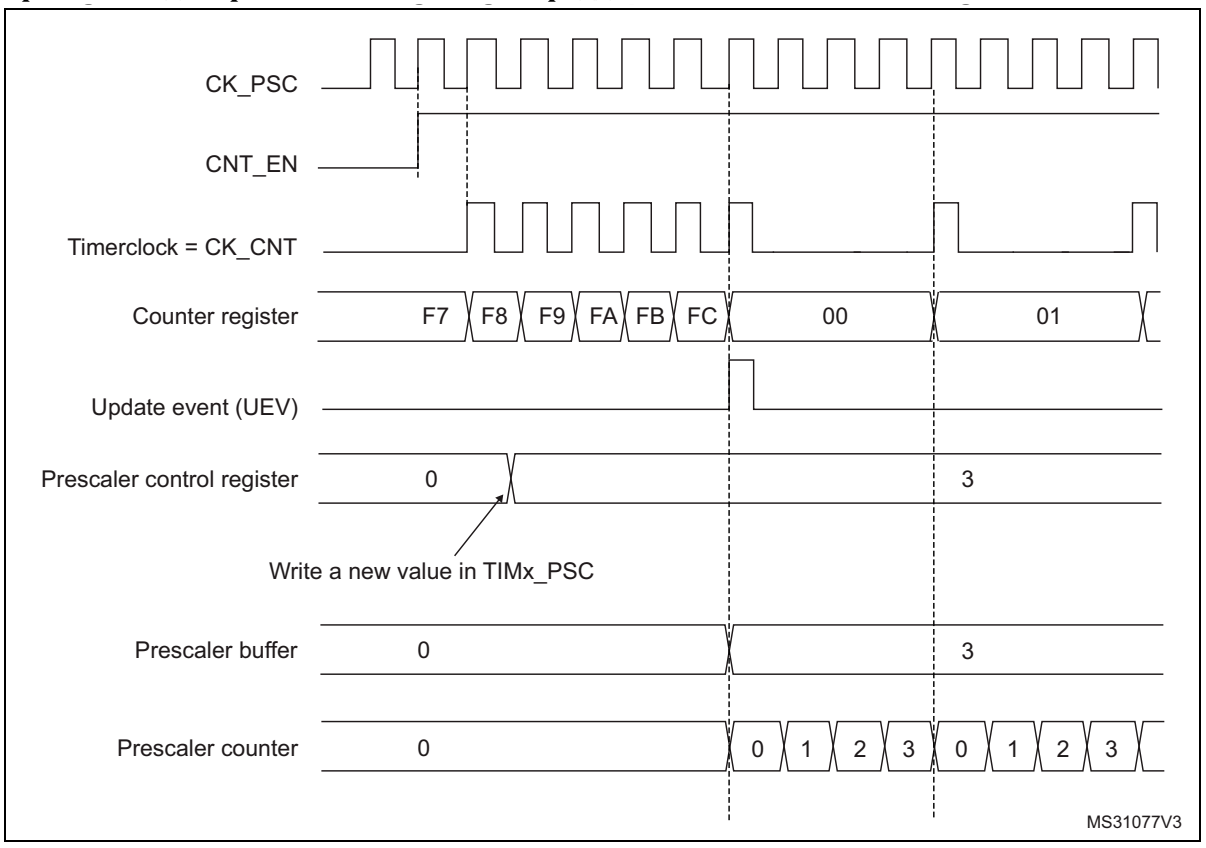
Описание предделителя

Это 16-бит счётчик делителя входной частоты (от 1 до 65536) с видимым регистром `TIMx_PSC`. Его можно менять налету. Новое значение счётчика пишется по следующему событию обновления.

Временная диаграмма с изменением предделителя с 1 на 2.



Временная диаграмма с изменением предделителя с 1 на 4.



17.3.2. Режим счёта

Счёт идёт от 0 до значения перезагрузки (регистр TIMx_ARR), сбрасывается в 0 и выдаёт событие переполнения счётчика.

Событие обновления UEV выдаётся при каждом переполнении счётчика или установкой бита UG в регистре TIMx_EGR (программно или контроллером режима ведомого).

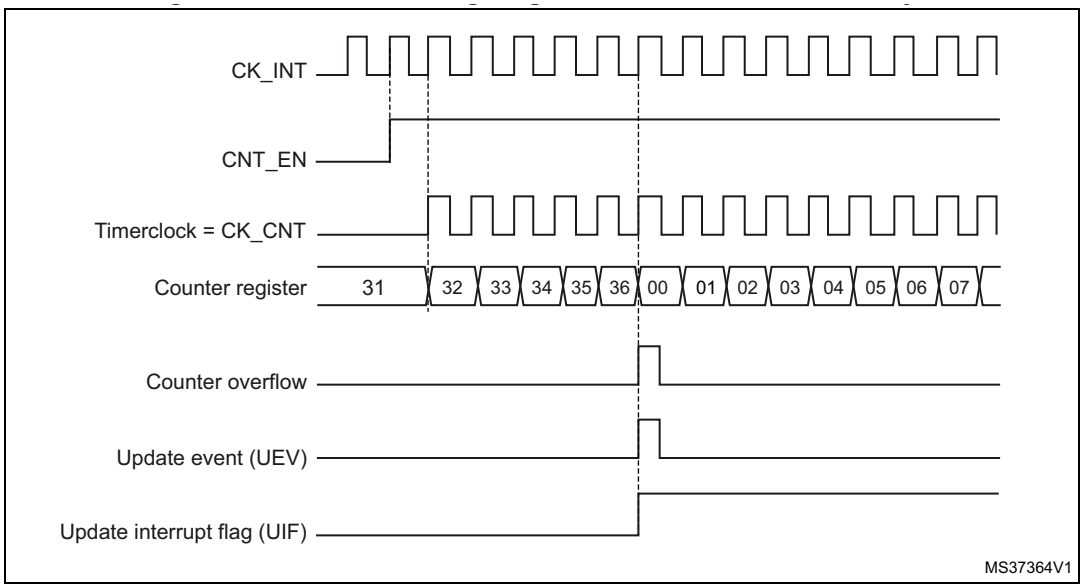
Событие UEV выключается установкой бита UDIS в регистре TIMx_CR1. Этим предупреждается запись скрытого регистра во время изменения регистра предзагрузки. Также событие UEV не появляется при сброшенном бите UDIS 0. Однако, счётчик продолжает считать с 0, равно как и предделитель (не изменив своего значения). Кроме того, если стоит бит URS в регистре TIMx_CR1, то установка бита UG выдаёт событие UEV без установки флага UIF (без прерывания и запроса DMA). Этим предупреждается одновременная выдача прерываний обновления и захвата при очистке счётчика по событию захвата.

По событию обновления обновляются все регистры и ставится флаг обновления (бит UIF в регистре TIMx_SR). Это зависит от бита URS:

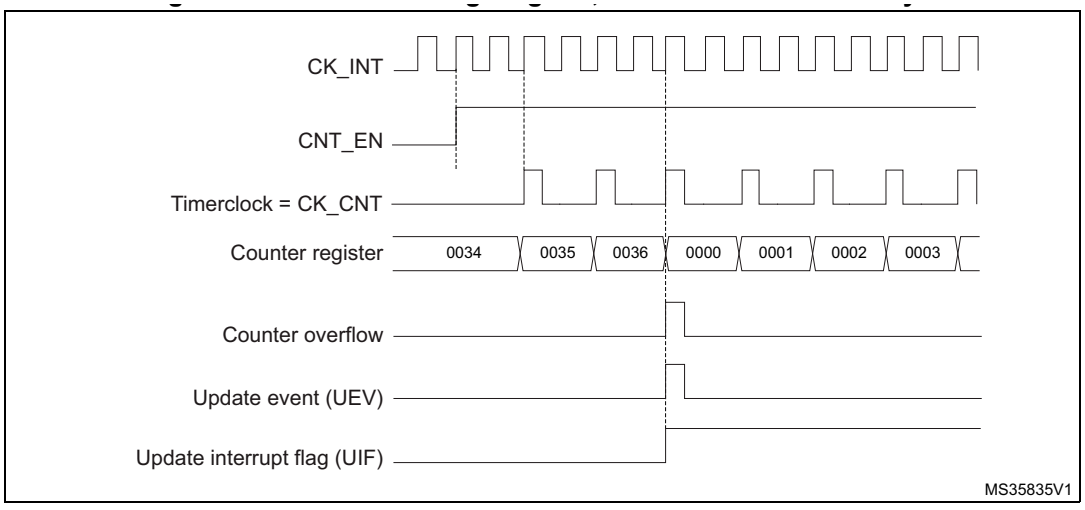
- В скрытый регистр перезагрузки пишется содержимое TIMx_ARR,
- Буфер предделителя перегружается из регистра TIMx_PSC.

Примеры ниже используют TIMx_ARR=0x36.

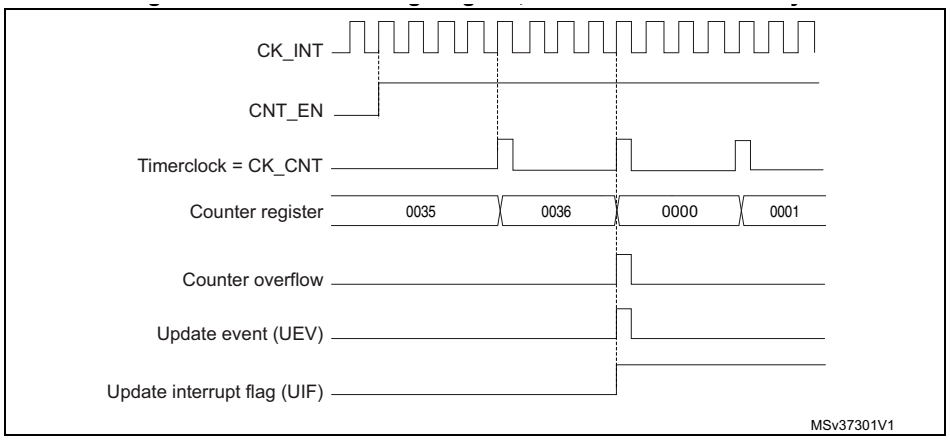
Временная диаграмма с делителем 1.



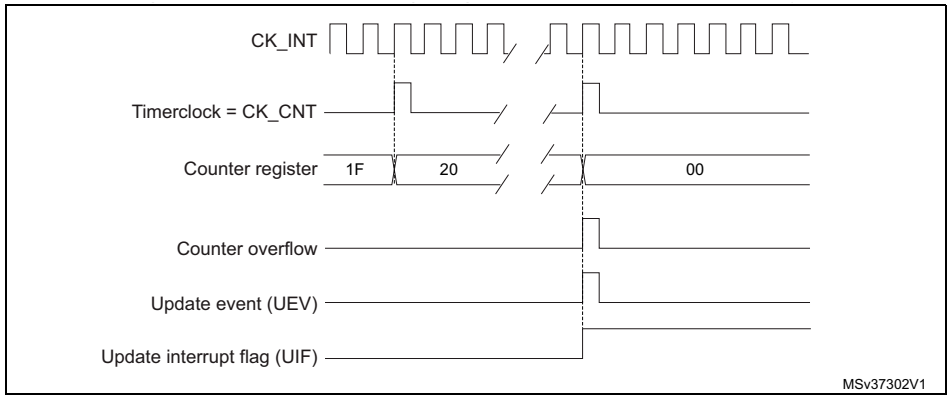
Временная диаграмма с делителем 2.



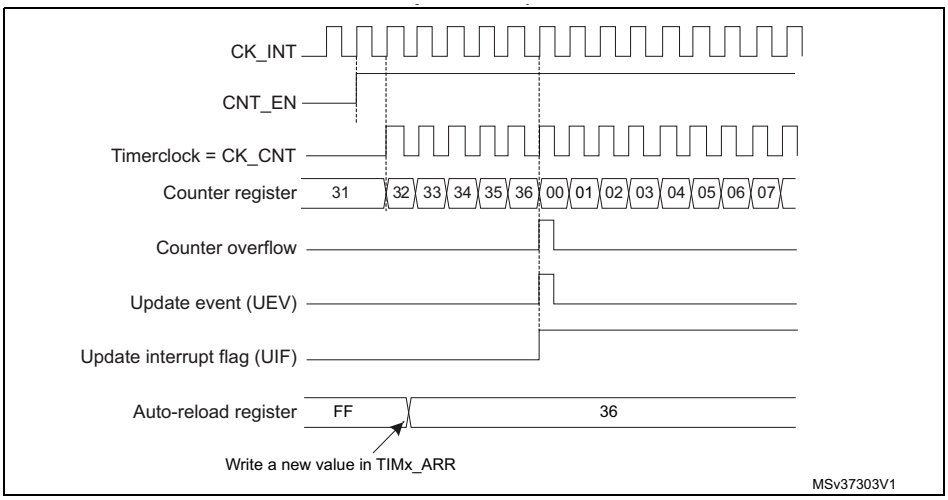
Временная диаграмма с делителем 4.



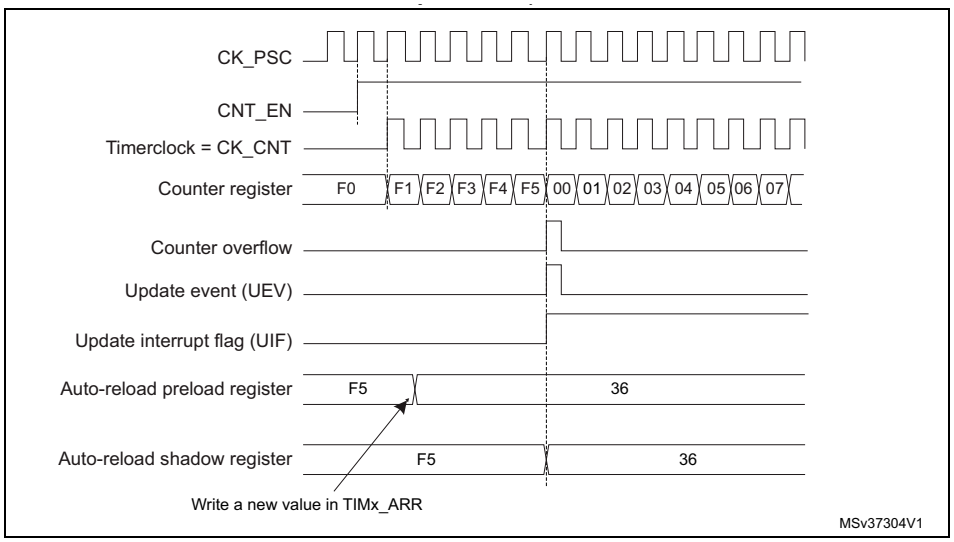
Временная диаграмма с делителем N.



Временная диаграмма события при ARPE=0 (TIMx_ARR не предзагружен).



Временная диаграмма события при ARPE=1 (TIMx_ARR предзагружен).

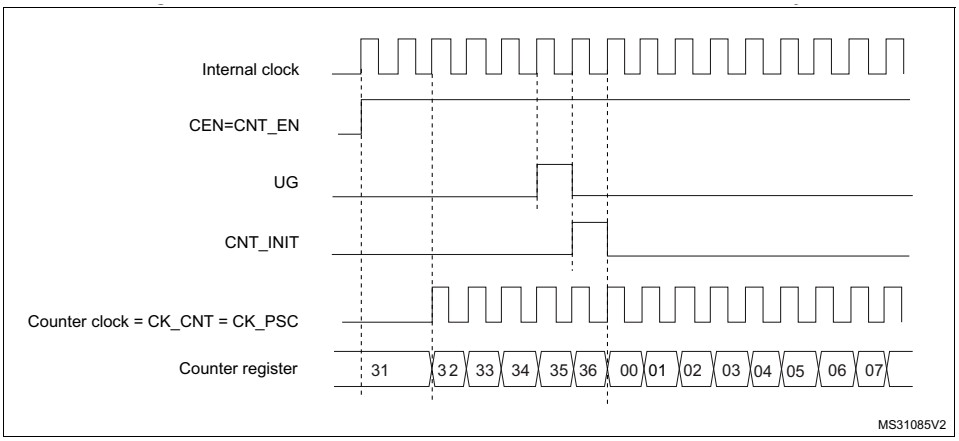


17.3.3. Выбор тактов

Счётчик может тактироваться внутреннего источника (CK_INT)

Всем управляют только биты CEN, DIR (в регистре TIMx_CR1) и UG (в регистре TIMx_EGR). Они изменяются только программно (кроме UG, снимающегося аппаратно). Сразу после установки бита CEN предделитель начинает получать внутренние такты CK_INT.

Режим без предделителя.



17.3.4. Режим отладки

В режиме отладки (ядро Cortex-M3 остановлено), счётчик TIMx либо работает, либо останавливается, в зависимости от бита DBG_TIMx_STOP в модуле DBG.

17.4. Регистры TIM6 и TIM7

32-бит регистры надо писать словами. Остальные регистры надо писать полусловами или словами. Читать можно байтами, полусловами или словами.

17.4.1. Регистр управления 1 TIM6 и TIM7 (TIMx_CR1)

Смещение адреса: 0x00

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ARPE	Reserved			OPM	URS	UDIS	CEN
								rw				rw	rw	rw	rw

— Биты 15:8 Резерв, не трогать.

— Бит 7 **ARPE**: Разрешение буфера перезагрузки TIMx_ARR

0: Нельзя

1: Нужно

— Бит 6:4 Резерв, не трогать.

— Бит 3 **OPM**: Режим одного импульса

0: Счётчик не останавливается

1: Счётчик останавливается по следующему событию обновления (снимается бит CEN)

— Бит 2 **URS**: Источник запроса по событию UEV.

Изменяется программно.

0: Разрешённые запросы прерывания/DMA выдаются по:

– Переполнение счётчика

– Установка бита UG

– Обновление от контроллера режима ведомого

1: Разрешённые запросы прерывания/DMA выдаются только по Переполнению счётчика.

— Бит 1 **UDIS:** Выключение выдачи события обновления UEV.

Изменяется программно.

0: Событие UEV выдаётся по:

– Переполнение счётчика

– Установка бита UG

– Обновление от контроллера режима ведомого

Регистры перегружаются значениями из регистров предзагрузки.

1: Событие UEV не выдаётся, скрытые регистры (ARR, PSC, CCRx) хранят значение. При стоящем бите UG или сигнале от контроллера режима ведомого счётчик и предделитель инициализируются.

— Бит 0 **CEN:** Включение счётчика.

0: Выключен

1: Включён

NB: Вентильный режим работает только при программно установленном бите CEN, режим запуска может ставить бит CEN аппаратно. В режиме Одного импульса бит CEN снимается аппаратно по событию обновления.

17.4.2. Регистр управления 2 (TIMx_CR2)

Смещение адреса: 0x04

По сбросу: 0x0000

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

— Биты 15:7 Резерв, не трогать.

— Биты 6:4 **MMS[2:0]:** Режим ведущего

Выбор информации для синхронизации ведомых таймеров (TRGO):

000: **Сброс** - Бит UG в регистре TIMx_EGR. Если на входе запуска идёт сигнал сброса (контроллер режима ведомого в сбросе), то сигнал на TRGO задерживается относительно реального сброса.

001: **Разрешение** - Сигнал разрешения счётчика (CNT_EN). Это полезно для одновременного запуска нескольких таймеров или управления окном ведомого таймера. Сигнал разрешения это объединение по OR бита CEN с входом запуска вентильного режима.

При подаче CNT_EN по входу запуска сигнал TRGO задерживается, кроме режима ведущий/ведомый (см. бит MSM в регистре TIMx_SMCR).

010: **Обновление** - На выход запуска (TRGO) направлено событие обновления. Например, ведущий таймер может быть предделителем ведомого.

NB: Тактирование ведомого таймера и ADC нужно включать до получения событий от ведущего таймера и не должно меняться налету.

— Биты 3:0 Резерв, не трогать.

17.4.3. Регистр разрешения прерываний TIM6 - TIM7 (TIMx_DIER)

Смещение адреса: 0x0C

По сбросу: 0x0000

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

Содержимое битов:

0: Нельзя

1: Можно

— Биты 15:9 Резерв, не трогать.

— Бит 8 **UDE**: Запрос обновления DMA

— Биты 7:1 Резерв, не трогать.

— Бит 0 **UIE**: Запрос прерывания обновления

17.4.4. Регистр состояния TIM6 - TIM7 (TIMx_SR)

Биты ставятся аппаратно, стираются записью 0.

Перезахват это появление события при стоящем его флаге.

Смещение адреса: 0x10

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															UIF
															rc_w0

Содержимое битов:

0: Не было

1: Было

— Бит 0 **UIF**: Флаг прерывания обновления

Удержание прерывания обновления ставится при:

– Переполнении счётчика и UDIS=0 в регистре TIMx_CR1.

– При программной инициализации CNT битом UG в регистре TIMx_EGR, если URS=0 и UDIS=0 в регистре TIMx_CR1.

17.4.5. Регистр генерации событий TIM6 - TIM7 (TIMx_EGR)

Событие генерируется программной установкой бита, снимается он аппаратно.

Смещение адреса: 0x14

По сбросу: 0x0000

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

— Биты 15:1 Резерв, не трогать.

— Бит 0 **UG**: Обновление.

1: Инициализирует счётчик и обновление регистров. Текущий счётчик предделителя чистится не влияя на само значение.

17.4.6. Счётчик TIM6 - TIM7 (TIMx_CNT)

Смещение адреса: 0x24

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **CNT[15:0]**: Значение счётчика.

17.4.7. Предделитель TIM6 - TIM7 (TIMx_PSC)

Смещение адреса: 0x28

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **PSC[15:0]**: Значение предделителя

Частота счёта (CK_CNT) равна f_{CK_PSC} / (PSC[15:0] + 1).

Значение PSC пишется в рабочий регистр предделителя по каждому событию обновления (включая очистку счётчика битом UG регистра TIMx_EGR или запуском в режиме Сброса).

17.4.8. Регистр предзагрузки TIM6 - TIM7 (TIMx_ARR)

Смещение адреса: 0x2C

По сбросу: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **ARR[15:0]**: Значение перезагрузки.

Если значение нулевое, то счётчик блокируется.

17.4.9. Карта регистров TIM6 - TIM7

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																								ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value																									0				0	0	0	0
0x04	TIMx_CR2	Reserved																								MMS[2:0]			Reserved				
	Reset value																									0	0	0					
0x08	Reserved																																
0x0C	TIMx_DIER	Reserved																							UDE	Reserved							UIE
	Reset value																								0								0
0x10	TIMx_SR	Reserved																															UIF
	Reset value																																0
0x14	TIMx_EGR	Reserved																															UG
	Reset value																																0
0x18	Reserved																																
0x1C	Reserved																																
0x20	Reserved																																
0x24	TIMx_CNT	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Reserved																ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

18. Таймер реального времени (RTC)

18.1. Введение в RTC

Это независимый таймер, состоящий из набора постоянно работающих счётчиков, пригодных для обеспечения функций даты/времени системы.

Ядро и конфигурация RTC (регистр RCC_BDCR) лежат в Резервном домене и сохраняются после сброса и выхода из режима Standby.

После сброса Резервные регистры и RTC отключены и защищены от записи. Разрешают доступ к регистрам и RTC так:

- включаем питание и тактирование BKP установкой битов PWREN и BKPEN в RCC_APB1ENR
- разрешаем доступ к регистрам и RTC битом DBP в регистре PWR_CR.

18.2. Основные свойства RTC

- Программируемый делитель вплоть до 2^{20}
- Бит-программируемый счётчик для длительных измерений
- Раздельные такты: PCLK1 для интерфейса APB1 и для clock (не менее чем в 4 раза медленнее тактов PCLK1)

- Источником тактов RTC могут быть:
  - Такты HSE / 128
  - Такты генератора LSE
  - Такты генератора LSI
- Два раздельных типа сброса:
  - Интерфейс APB1 сбрасывается системным сбросом
  - Ядро RTC (Предделитель, Будильник, Счётчик и Делитель) сбрасывается только сбросом BKP.
- Три выделенных маскируемых линии прерываний:
  - Программно управляемый Будильник.
  - Секундное, с программируемым интервалом (до 1 секунды).
  - Переполнение, переход счётчика через 0.

18.3. Функциональное описание RTC

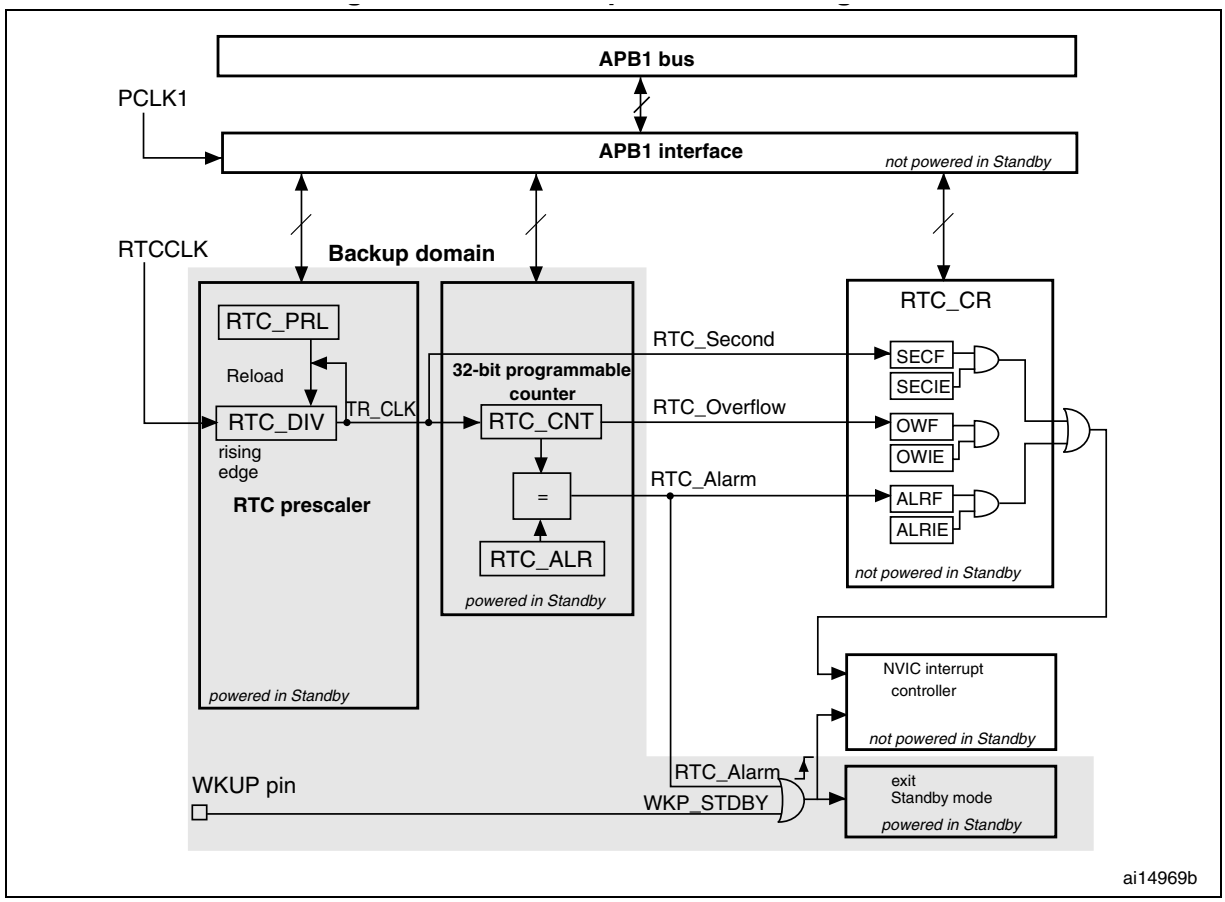
18.3.1. Обзор

RTC состоит из двух основных блоков: интерфейса с шиной APB1 и ядра RTC.

Интерфейс APB1 содержит набор 16-бит регистров, доступных через шину APB1. Тактируется он от шины APB1.

Ядро RTC состоит из двух основных блоков. Первый бок это предделитель, создающий временную базу RTC (TR_CLK), программируемый на период до 1 секунды. Включает программируемый 20-бит делитель (предделитель RTC). Каждый период TR_CLK RTC выдаёт Секундное прерывание, если оно разрешено в регистре RTC_CR. Второй блок это 32-бит программируемый счётчик, инициализируемый на системное время. Оно инкрементируется со скоростью TR_CLK и сравнивается с программируемой датой (в регистре RTC_ALR) для выдачи прерывания Будильника, если разрешено в регистре RTC_CR.

Упрощённая блок-схема RTC.



18.3.2. Сброс регистров RTC

Все системные регистры асинхронно сбрасываются сбросами Системы и Питания, кроме RTC_PRL, RTC_ALR, RTC_CNT и RTC_DIV. Они сбрасываются сбросом Резервного домена.

18.3.3. Чтение регистров RTC

Ядро RTC полностью независимо от интерфейса APB1.

Программно предделитель, счётчик и будильник RTC доступны по шине APB1, но соответствующие регистры внутренне обновляются каждым передним фронтом тактов RTC, ресинхронизированных с тактами RTC APB1. Это правдиво и для флагов RTC.

То есть первое чтение регистров RTC APB1 сразу по включению интерфейса APB1 после сброса до внутреннего обновления. Возможные причины:

- Сброс системы или питания
- Выход MCU из режима Standby
- Выход MCU из режима Stop

В любом случае ядро RTC продолжает работать при выключенном интерфейсе APB1 (сброс, не тактируемый или без питания).

Следовательно для чтения регистров RTC после выключения интерфейса RTC APB1 надо сначала дождаться бита синхронизации RSF в регистре RTC_CRL register to be set by hardware.

Режимы пониженного питания WFI и WFE на интерфейс RTC APB1 не влияют.

18.3.4. Конфигурация регистров RTC

Для записи в регистры RTC_PRL, RTC_ALR, RTC_CNT периферию надо перевести в режим Конфигурации установкой бита CNF в регистре RTC_CRL.

Кроме того, любая запись в регистр RTC возможна только после завершения предыдущей записи. Писать можно только при стоящем бите RTOFF в регистре RTC_CR.

Процедура:

1. Опрашивать RTOFF до появления ‘1’
2. Установить бит CNF
3. Записать нужные регистры RTC
4. Очистить бит CNF
5. Опрашивать RTOFF до появления ‘1’.

Операция записи выполняется только при снятом бите CNF; это занимает не меньше 3 циклов RTCCLK.

18.3.5. Установка флагов RTC

Флаг Секунд RTC (SECF) ставится по каждому такту ядра RTC перед обновлением счётчика RTC.

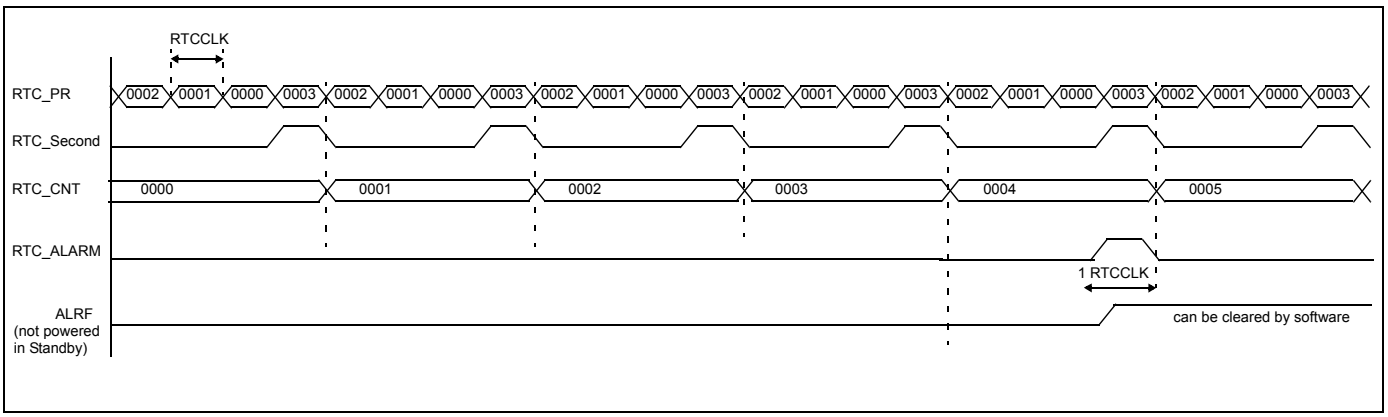
Флаг Переполнения RTC (OWF) ставится по последнему такту ядра RTC перед обнулением счётчика.

RTC_Alarm и флаг Будильника RTC (ALRF) ставится по последнему такту ядра RTC перед достижением счётчиком значения регистра плюс 1 (RTC_ALR + 1).

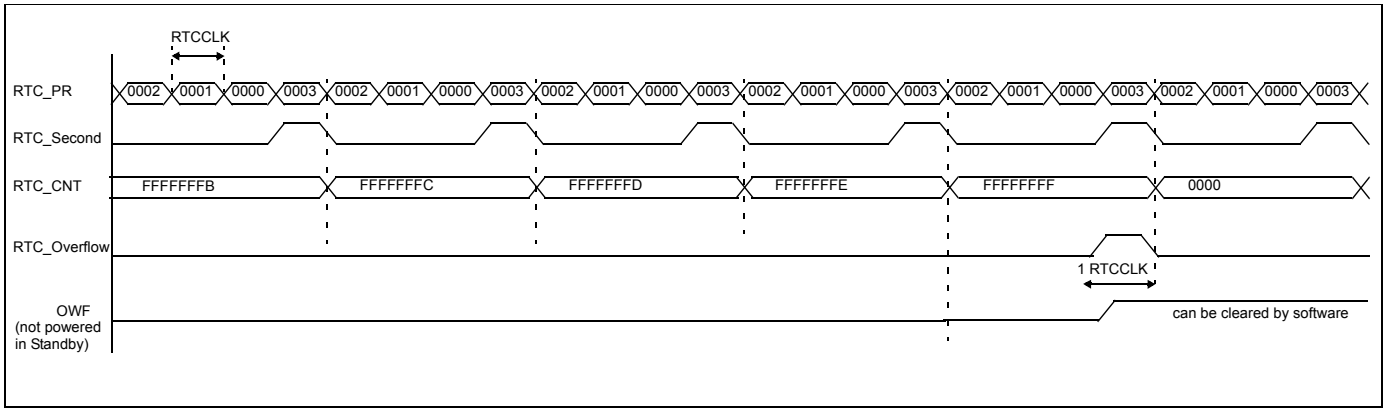
Запись в флаги Будильника и Секунд нужно синхронизировать так:

- Использовать прерывание Будильника и обновлять регистры будильника и счётчика RTC внутри обработчика.
- Ждать установки бита SECF и тогда обновлять эти регистры.

Сигналы Будильника и Секунд с PR=0003, ALARM=00004.



Сигнал Переполнения с PR=0003.



18.4. Регистры RTC

Регистры доступны полусловами или словами.

18.4.1. Старший регистр управления RTC (RTC_CRH)

Смещение адреса: 0x00

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													OWIE	ALRIE	SECIE
													rw	rw	rw

— Биты 15:3 Резерв, не трогать.

— Бит 2 **OWIE**: Разрешение прерывания Переполнения

0: Нельзя

1: Можно

— Бит 1 **ALRE**: Разрешение прерывания Будильника

0: Нельзя

1: Можно

— Бит 0 **SECIE**: Разрешение прерывания Секунд

0: Нельзя

1: Можно

После сброса все прерывания маскированы и можно спокойно писать регистры RTC. Писать биты нужно соблюдая предписанные процедуры конфигурации и установки флагов.

18.4.2. Младший регистр управления RTC (RTC_CRH)

Смещение адреса: 0x04

По сбросу: 0x0020

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RTOFF	CNF	RSF	OWF	ALRF	SECF
										r	rw	rc_w0	rc_w0	rc_w0	rc_w0

— Биты 15:6 Резерв, всегда 0.

— Бит 5 **RTOFF**: Флаг завершения записи RTC

0: Запись не завершена

1: Запись завершена

— Бит 4 **CNF**: Флаг конфигурации

Программно разрешает запись в регистры RTC_CNT, RTC_ALR и RTC_PRL. Реально запись выполняется при снятии бита.

0: Выход из конфигурации (старт обновления регистров RTC).

1: Вход в конфигурацию.

— Бит 3 **RSF**: Флаг синхронизации регистров

Ставятся аппаратно при каждом обновлении регистров RTC_CNT и RTC_DIV. После сброса или остановки тактирования APB1 этот бит нужно программно снимать перед первым чтением, чтобы дождаться синхронизации записи регистров RTC_CNT, RTC_ALR и RTC_PRL.

0: Ещё не синхронизированы

1: Синхронизированы

— Бит 2 **OWF**: Флаг переполнения

Ставится аппаратно при переполнении 32-бит счётчика. При OWIE=1 в регистре RTC_CRH вызывается прерывание. Снимается только программно. Запись ‘1’ бессмысленна.

0: Не было

1: Было

— Бит 1 **ALRF**: Флаг Будильника

Ставится аппаратно при достижении 32-бит счётчиком порога в регистре RTC_ALR. При ALRIE=1 в регистре RTC_CRH вызывается прерывание. Снимается только программно. Запись ‘1’ бессмысленна.

0: Не сработал

1: Сработал

— Бит 0 **SECIF**: Флаг Секунд

Ставится аппаратно при переполнении 32-бит предделителя, инкрементируя счётчик RTC. Это периодический сигнал (обычно 1 секунда). An interrupt is generated При SECIE=1 в регистре RTC_CRH вызывается прерывание. Снимается только программно. Запись ‘1’ бессмысленна.

0: Секунды не было

1: Секунда была

Писать биты нужно соблюдая предписанные процедуры конфигурации и установки флагов.

NB: Все флаги удерживаются вплоть до программного сброса соответствующего запроса RTC_CR обработчиком прерывания.

После сброса все прерывания отключены и можно спокойно писать регистры RTC. Писать биты нужно соблюдая предписанные процедуры конфигурации и установки флагов.

При отключённом тактировании APB1 биты OWF, ALRF, SECF и RSF не обновляются.

Биты OWF, ALRF, SECF и RSF ставятся аппаратно и снимаются программно.

При ALRF = 1 и ALRIE = 1, разрешается глобальное прерывание RTC. Если в контроллере EXTI разрешена и EXTI Line 17, то разрешаются и глобальное прерывание RTC и прерывание Будильника.

При ALRF = 1 и если в контроллере EXTI в режиме прерывания разрешена EXTI Line 17, то вызывается прерывание Будильника. При EXTI Line 17, разрешённой в режиме события, на эту линию выдаётся импульс (без вызова прерывания будильника).

18.4.3. Регистры загрузки предделителя RTC (RTC_PRLH / RTC_PRLL)

Определяют период счёта предделителя. Защищены от записи битом RTOFF в регистре RTC_CR.

Старший регистр загрузки предделителя RTC (RTC_PRLH).

Смещение адреса: 0x08

По сбросу: 0x0000

Доступен только по записи.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PRL[19:16]			
												w	w	w	w

— Биты 15:4 Резерв, всегда 0.

— Биты 3:0 **PRL[19:16]:** Старшие биты предделителя.

Определяет частоту счётчика по формуле: $f_{TR\_CLK} = f_{RTCCLK}/(PRL[19:0]+1)$.

Младший регистр загрузки предделителя RTC (RTC_PRLL).

Смещение адреса: 0x0C

По сбросу: 0x8000

Доступен только по записи.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRL[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

— Биты 15:0 **PRL[15:0]:** Младшие биты предделителя.

Определяет частоту счётчика по формуле: $f_{TR\_CLK} = f_{RTCCLK}/(PRL[19:0]+1)$.

Внимание: Нулевое значение не рекомендуется. Прерывания и флаги RTC ставятся неправильно.

NB: Если f_{RTCCLK} равна 32.768 kHz, то запись 7FFFh в этот регистр даёт период в 1 секунду.

18.4.4. Регистры чтения предделителя RTC (RTC_DIVH / RTC_DIVL)

Каждый период тактов TR_CLK счётчик предделителя перегружается из регистра RTC_PRL. Точное значение счётчика пишется в регистр RTC_DIV по каждому изменению регистров RTC_PRL или RTC_CNT. Отсюда и читается.

Старший регистр чтения предделителя RTC (RTC_DIVH).

Смещение адреса: 0x10

По сбросу: 0x0000

Доступен только по чтению.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												RTC_DIV[19:16]			
												r	r	r	r

— Биты 15:4 Резерв, всегда 0.

— Биты 3:0 **RTC_DIV[19:16]:** Старшие биты предделителя.

Младший регистр чтения предделителя RTC (RTC_DIVL).

Смещение адреса: 0x14

По сбросу: 0x8000

Доступен только по чтению.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_DIV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 15:0 **RTC_DIV[15:0]:** Младшие биты предделителя.

18.4.5. Регистры счётчика RTC (RTC_CNTH / RTC_CNTL)

В ядре RTC есть один 32-бит программируемый счётчик, доступный через два 16-бит регистра; он считает такты TR_CLK от предделителя. Чтение регистра RTC_CNT возвращает текущее значение (системную дату). Регистры защищены от записи битом RTOFF в регистре RTC_CR. Запись в старший (RTC_CNTH) или младший (RTC_CNTL) регистры напрямую грузится в счётчик и перегружает предделитель.

Старший регистр счётчика RTC (RTC_CNTH).

Смещение адреса: 0x18

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_CNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **RTC_CNT[19:16]:** Старшие биты счётчика.

Чтение регистра RTC_CNTH возвращает старшую часть счётчика RTC. Писать в него можно в режиме конфигурации.

Младший регистр счётчика RTC (RTC_CNTL).

Смещение адреса: 0x1C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **RTC_CNT[15:0]:** Младшие биты счётчика.

Чтение регистра RTC_CNTL возвращает младшую часть счётчика RTC. Писать в него можно в режиме конфигурации.

18.4.6. Регистры будильника RTC (RTC_ALRH / RTC_ALRL)

При достижении счётчиком значения регистра RTC_ALR запускается будильник, возбуждая прерывание RTC_alarmIT. Он защищён от записи битом RTOFF в регистр RTC_CR.

Старший регистр будильника RTC (RTC_ALRH).

Смещение адреса: 0x20

По сбросу: 0xFFFF

Только запись.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_ALR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

— Биты 15:0 **RTC_ALR[19:16]:** Старшие биты счётчика.

Это старшая часть будильника RTC. Писать в него можно в режиме конфигурации.

Младший регистр будильника RTC (RTC_ALRL).

Смещение адреса: 0x24

По сбросу: 0xFFFF

Только запись.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_ALR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

— Биты 15:0 **RTC_ALR[15:0]:** Младшие биты счётчика.

Это младшая часть счётчика RTC. Писать в него можно в режиме конфигурации.

18.5. Карта регистров RTC

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RTC_CRH	Reserved																													OWIE	ALRIE	SECIE
	Reset value																														0	0	0
0x04	RTC_CRL	Reserved																										RTOFF	CNF	RSF	OWF	ALRF	SECF
	Reset value																											1	0	0	0	0	0
0x08	RTC_PRLH	Reserved																												PRL[19:16]			
	Reset value																													0	0	0	0
0x0C	RTC_PRLL	Reserved																PRL[15:0]															
	Reset value																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	RTC_DIVH	Reserved																DIV[31:16]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	RTC_DIVL	Reserved																DIV[15:0]															
	Reset value																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	RTC_CNTH	Reserved																CNT[13:16]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	RTC_CNTL	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	RTC_ALRH	Reserved																ALR[31:16]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x24	RTC_ALRL	Reserved																ALR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

19. Независимый сторожевой таймер (IWDG)

19.1. Введение в IWDG

MCU имеют два сторожевых таймера (Независимый и Оконный) для обнаружения программных отказов и запуска системного сброса и прерывания (только Оконный) при истечении заданного интервала времени.

Независимый сторож (IWDG) тактируется собственным выделенным низкочастотным генератором (LSI) и остаётся активным даже при сбое главных тактов. Оконный сторож (WWDG) тактируется предделителем тактов APB1 и может обнаруживать ненормально ранние и поздние события с поведении приложения.

IWDG подходит для полностью независимого контроля приложения, но с меньшей точностью по времени. WWDG пригоден для контроля поведения системы в рамках заданного временного окна.

19.2. Основные свойства IWDG

- Независимый обратный счётчик
- Тактируется независимым RC генератором (может работать в режимах Standby и Stop
- Выдаёт Сброс при достижении счётчиком 0x000

19.3. Функциональное описание IWDG

Если сторож запускается записью 0xCCCC в регистр ключа (IWDG_KR), то счёт начинается с числа 0xFFF. В конце счёта (0x000) выдаётся сигнал сброса IWDG.

Если в регистр IWDG_KR пишется ключ 0xAAAA, то в счётчик загружается значение регистра IWDG_RLR и отключается выдача сброса.

19.3.1. Аппаратный сторож

Если в битах конфигурации выбран “Аппаратный сторож”, то он автоматически запускается по включению питания и при завершении счёта выдаёт сброс независимо от какой-либо записи в регистр ключа.

19.3.2. Защита доступа

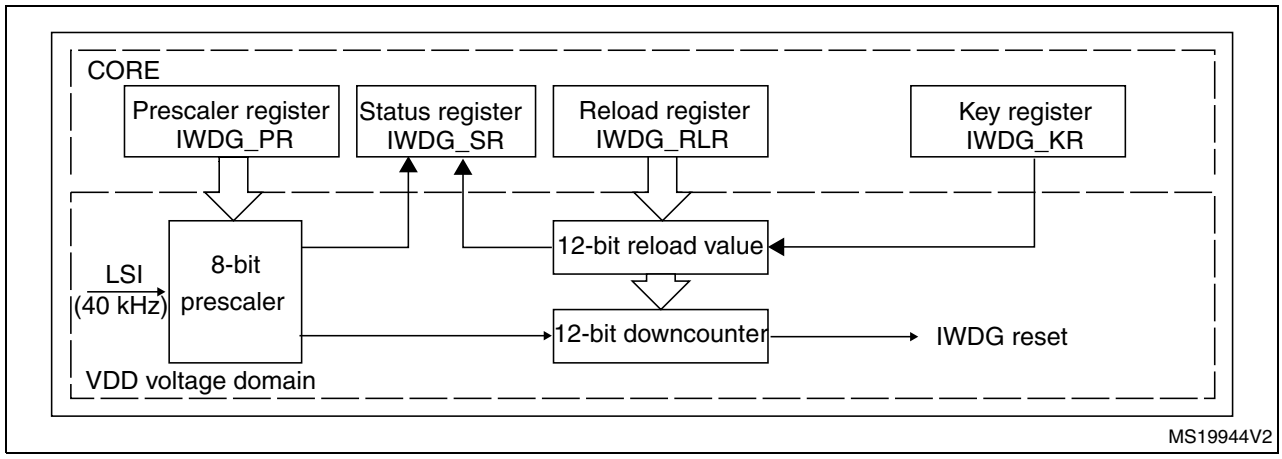
Регистры IWDG_PR и IWDG_RLR защищены от записи. Чтобы изменить их надо сначала записать 0x5555 в регистр IWDG_KR. Запись любого другого значения включает в этот регистр включает защиту записи. Предполагается, что это операций перезагрузки (запись 0xAAAA).

События обновления предделителя и перезагрузки счётчика показываются в регистре состояния.

19.3.3. Режим отладки

В режиме отладки (ядро Cortex-M3 остановлено) счётчик IWDG продолжает работать или останавливается в зависимости от бита конфигурации DBG_IWDG_STOP в модуле DBG.

Блок-схема IWDG.



NB: Функции сторожа, реализованные в домене V_{DD}, продолжают работать в режимах Stop и Standby.

Таблица 96. Min/max таймауты IWDG (в ms) на 40 kHz (LSI)(1).

Предделитель	Биты PR[2:0]	Min таймаут RL[11:0]=0x000	Max таймаут RL[11:0]=0xFFF
/4	0	0.1	409.6
/8	1	0.2	819.2
/16	2	0.4	1638.4
/32	3	0.8	3276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	6 (или 7)	6.4	26214.4

1. Частоту RC генератора LSI можно менять.

19.4. Регистры IWDG

Регистры доступны полусловами или словами.

19.4.1. Регистр ключа (IWDG_KR)

Смещение адреса: 0x00

По сбросу: 0x0000 0000. Сбрасывается режимом Standby.

31–16	15–0
Reserved	KEY[15:0]
	w w w w w w w w w w w w w w w w

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **KEY[15:0]**: Значение ключа (только запись, чтение 0000h)

При записи AAAAh включается регулярный интервал, иначе при достижении 0 выдаётся сброс.

При записи 5555h разрешается доступ к регистрам IWDG_PR и IWDG_RLR.

При записи CCCCh запускает таймер (если не включён аппаратный сторож).

19.4.2. Регистр предделителя (IWDG_PR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000.

31–3	2–0
Reserved	PR[2:0]
	rw rw rw

— Биты 31:3 Резерв, не трогать.

— Биты 2:0 **PR[2:0]:** Значение предделителя

Биты защищены от записи. При записи бит PVU в регистре IWDG_SR должен быть снят.

000: делитель /4

001: делитель /8

010: делитель /16

011: делитель /32

100: делитель /64

101: делитель /128

110: делитель /256

111: делитель /256

NB: Чтение этого регистра возвращает значение предделителя из домена V_{DD}. При незавершённой операции записи значение недостоверно. Читать надо при снятом бите PVU в регистре IWDG_SR.

19.4.3. Регистр перезагрузки (IWDG_RLR)

Смещение адреса: 0x08

По сбросу: 0x0000 0FFF. Сбрасывается режимом Standby.

31–12	11–0
Reserved	RL[11:0]
	rw rw rw rw rw rw rw rw rw rw rw rw

— Биты 31:12 Резерв, не трогать.

— Биты 11:0 **RL[11:0]:** Перегружаемое значение сторожа

Биты защищены от записи. Пишется в счётчик каждый раз после записи AAAAh в регистр IWDG_KR. Обратный отсчёт начинается с этого значения. Период таймаута это функция этого значения и предделителя тактов. При записи бит RVU в регистре IWDG_SR должен быть снят.

NB: Чтение этого регистра возвращает значение перезагрузки из домена V_{DD}. При незавершённой операции записи значение недостоверно. Читать надо при снятом бите PVU в регистре IWDG_SR.

19.4.4. Регистр состояния (IWDG_RLR)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000. Режимом Standby не сбрасывается.

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

— Биты 31:2 Резерв, не трогать.

— Бит 1 **RVU:** Обновление значения перезагрузки

Ставится аппаратно во время операции записи регистра, снимается аппаратно после её завершения в домене V_{DD} (занимает до 5 циклов RC 40 kHz). Писать можно при снятом бите RVU.

— Бит 0 **PVU:** Обновление значения предделителя

Ставится аппаратно во время операции записи регистра, снимается аппаратно после её завершения в домене V_{DD} (занимает до 5 циклов RC 40 kHz). Писать можно при снятом бите PVU.

NB: При использовании нескольких значений перезагрузки или предделителя перед их записью нужно дождаться снятия битов RVU (перезагрузка) или PVU (предделитель). А завершения операции записи ждать не надо, она завершится без вас.

19.4.5. Карта регистров IWDG

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**IWDG_KR**	Reserved																KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**IWDG_PR**	Reserved																														PR[2:0]	
	Reset value																															0	0	0
0x08	**IWDG_RLR**	Reserved																				RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	**IWDG_SR**	Reserved																														RVU	PVU
	Reset value																															0	0

20. Оконный сторожевой таймер (WWDG)

20.1. Введение в WWDG

WWDG используется для обнаружения программных отказов, обычно вызываемых внешним влиянием или непредвиденными логическими условиями. По истечении заданного периода времени схема выдаёт сброс MCU, если только программа не обновляет обратный счётчик до очистки бита `T6`. Сброс MCU также выдаётся если 7-бит обратный счётчик (в регистре управления) обновляется до достижения счётчиком границы окна. То есть счётчик нужно обновлять в границах окна.

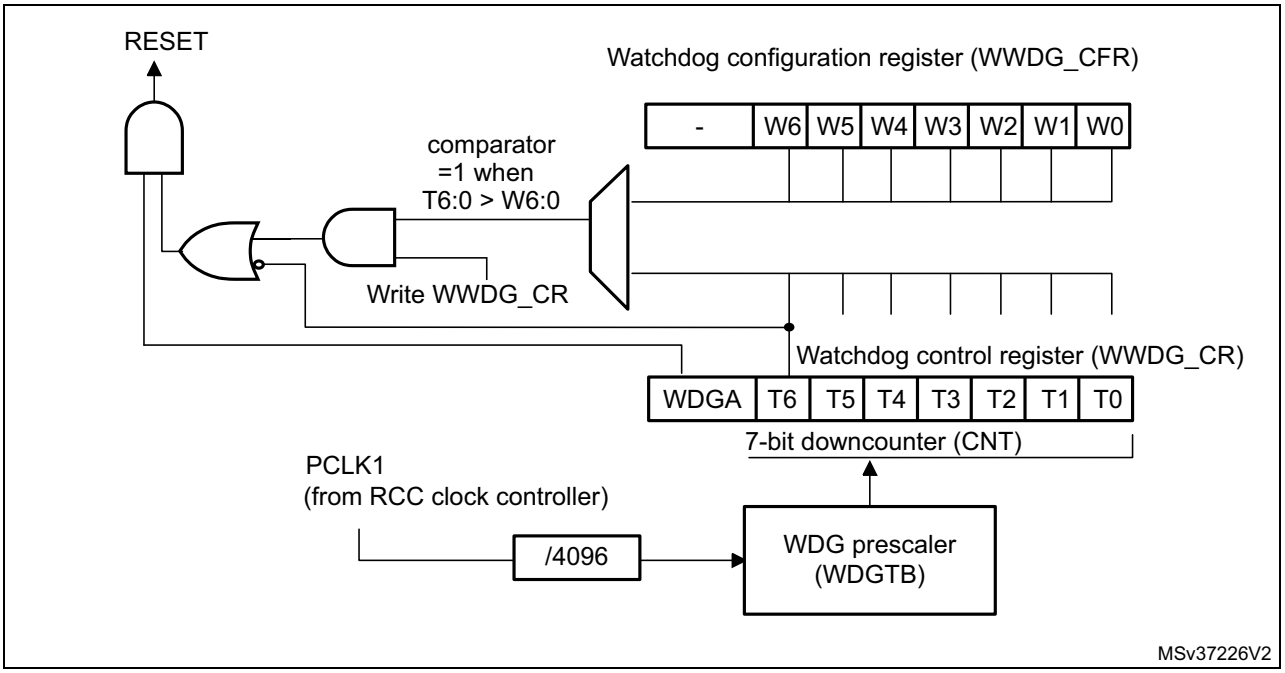
20.2. Основные свойства WWDG

- Программируемый независимый обратный счётчик
- Условный сброс (при активированном стороже)
  - Если значение счётчика становится меньше 0x40
  - Если счётчик перегружается вне окна
- Прерывание раннего пробуждения (EWI): выдаётся (если разрешено при активированном стороже) когда счётчик равен 0x40.

20.3. Функциональное описание WWDG

Сброс выдаётся активированным сторожем (стоит бит `WDGA` в регистре `WWDG_CR`) когда 7-бит счётчик (биты `T[6:0]`) переходят с 0x40 на 0x3F (`T6` очищается). Также он выдаётся при перезаписи счётчика когда он больше границы окна.

Блок-схема сторожа.



Во избежание сброса MCU программа должна регулярно писать в работающий счётчик `WWDG_CR` число в пределах от `0xFF` и `0xC0`. Писать нужно только когда значение счётчика меньше границы окна.

Включение сторожа

По сбросу сторож всегда выключен. Включается он установкой бита `WDGA` в регистре `WWDG_CR`, выключается только сбросом.

Управление счётчиком

Счётчик работает даже при выключенном стороже. При включённом стороже бит `T6` должен стоять во избежание немедленного сброса.

Биты `T[5:0]` содержат число шагов счётчика до выдачи сброса. Из-за неизвестного состояния предделителя на момент записи регистра `WWDG_CR` время лежит в диапазоне между минимальным и максимальным значениями. Регистр конфигурации (`WWDG_CFR`) содержит верхний предел окна: для предупреждения выдачи сброса перегружать счётчик надо когда его значение меньше границы окна и больше `0x3F`.

NB: Для выдачи программного сброса можно использовать бит `T6` (бит `WDGA` стоит, а `T6` чист).

Прерывание сторожа

В случае надобности выполнения каких-либо действий перед реальным сбросом можно использовать Прерывание Ранней Побудки (EWI). Оно разрешается битом `EWI` в регистре `WWDG_CFR`. и вызывается при достижении счётчиком значения `0x40`.

Прерывание EWI можно использовать для выполнения проверки и постепенного выключения системы без выдачи сброса `WWDG`. В этом случае обработчик должен перегрузить счётчик `WWDG` и запустить нужные действия.

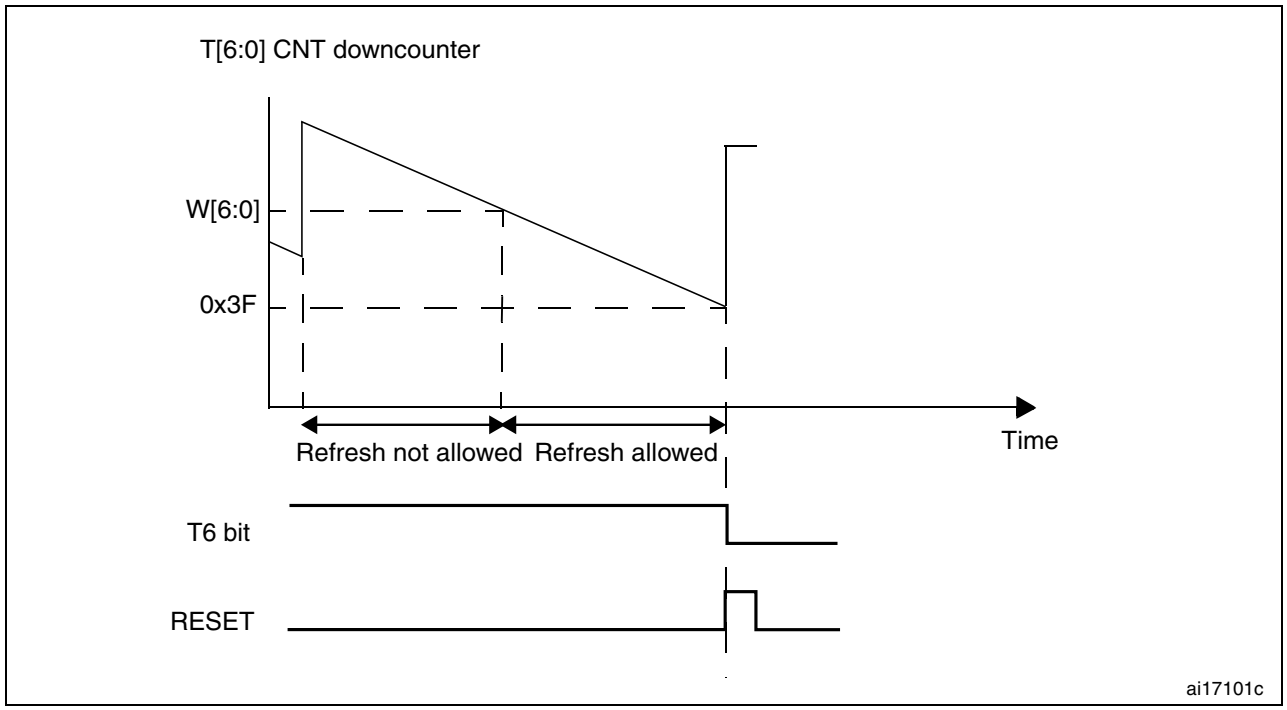
Прерывание EWI очищается снятием бита `EWIF` в регистре `WWDG_SR`.

NB: Если прерывание EWI не может быть по каким-либо причинам обслужено, то сброс `WWDG` таки произойдёт.

20.4. Вычисление таймаута WWDG

Внимание: При записи регистра WWDG_CR всегда ставьте бит T6.

Временная диаграмма Оконного сторожа.



Формула вычисления таймаута:

$$t_{WWDG} = t_{PCLK1} \times 4096 \times 2^{WDGTB[1:0]} \times (T[5:0] + 1) \quad (ms)$$

где:

t_{WWDG}: таймаут WWDG

t_{PCLK1}: период тактов APB1 в ms

4096: значение внутреннего делителя

Например, частота APB1 равна 24 MHz, WDGTB[1:0] равны 3 и T[5:0] равны 63:

$$t_{WWDG} = 1/\ 24000 \times 4096 \times 2^{3} \times (63 + 1) = 21.85ms$$

Таблица 98. Min и Max значения t_{WWDG} при f_{PCLK1}=36MHz.

Предделитель	WDGTB	Min	Max
1	0	113 µs	7.28 ms
2	1	227 µs	14.56 ms
4	2	455 µs	29.12 ms
8	3	910 µs	58.25 ms

20.5. Режим отладки

В режиме отладки (ядро Cortex-M3 остановлено) счётчик WWDG продолжает работать или останавливается в зависимости от бита конфигурации DBG_WWDG_STOP в модуле DBG.

20.6. Регистры WWDG

20.6.1. Регистр ключа (WWDG_CR)

Смещение адреса: 0x00

По сбросу: 0x0000 007F.

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								WDGA	T[6:0]						
								rs	rw						

— Биты 31:8 Резерв, не трогать.

— Бит 7 **WDGA:** Бит активации

Ставится программно, снимается аппаратно по сбросу. При WDGA = 1 может выдаваться сброс.

0: Выключен

1: Включён

— Биты 6:0 **T[6:0]:** 7-бит счётчик (MSB to LSB)

Значение счётчика. Декрементируется каждые (4096 x $2^{WDGTB[1:0]}$) тактов PCLK1. Сброс выдаётся при переходе от 0x40 к 0x3F (T6 очищается).

20.6.2. Регистр конфигурации (WWDG_CFR)

Смещение адреса: 0x04

По сбросу: 0x0000 007F.

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						EWI	WDGTB[1:0]		W[6:0]						
						rs	rw		rw						

— Биты 31:10 Резерв, не трогать.

— Бит 9 **EWI:** Прерывание Ранней Побудки

Если установлен, то при достижении счётчиком 0x40 выдаётся прерывание. Чистится аппаратно после сброса.

— Биты 8:7 **T[6:0]:** База таймера

Изменение базы предделителя:

00: CK Counter Clock (PCLK1 / 4096) / 1

01: CK Counter Clock (PCLK1 / 4096) / 2

10: CK Counter Clock (PCLK1 / 4096) / 4

11: CK Counter Clock (PCLK1 / 4096) / 8 .

— Биты 6:0 **W[6:0]:** 7-бит значение окна

Значение окна чтобы сравнивать со счётчиком.

20.6.3. Регистр состояния (WWDG_SR)

Смещение адреса: 0x08

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EWIF
															rc_w0

— Биты 31:10 Резерв, не трогать.

— Бит 0 **EWIF:** Флаг Прерывания Ранней Побудки

Ставится аппаратно при достижении счётчиком 0x40. Чистится программно записью ‘0’. Запись ‘1’ бестолкова. Ставится также при не разрешённом прерывании.

20.6.4. Карта регистров WWDG

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**WWDG_CR**	Reserved																								WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x04	**WWDG_CFR**	Reserved																						EWI	WDGTB1	WDGTB0	W[6:0]						
	Reset value																							0	0	0	1	1	1	1	1	1	1
0x08	**WWDG_SR**	Reserved																															EWIF
	Reset value																																0

21. Контроллер статической памяти (FSMC)

21.1. Основные свойства FSMC

Блок FSMC это интерфейс с синхронной и асинхронной памятью и 16-бит PC Card. Основное назначение:

- Трансляция передач AHB в протокол внешних устройств
- Согласование временных требований внешних устройств

Все виды внешней памяти используют одни адреса, данные и управляющие сигналы с контроллером. Доступ FSMC выполняется по одному обращению за раз к выбранному внешнему устройству.

Основные свойства FSMC:

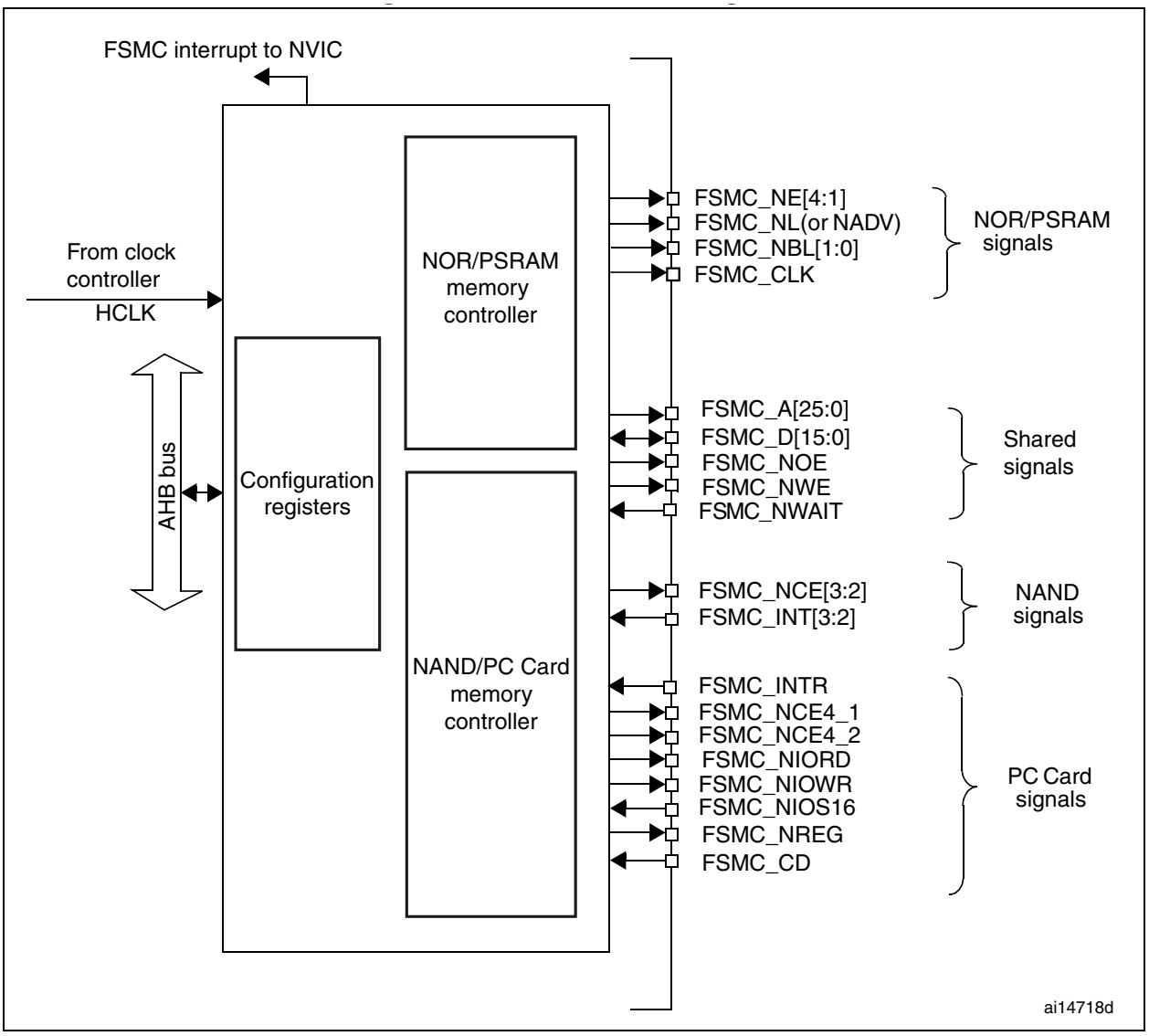
- Интерфейс со статическими устройствами:
  - Статическая память (SRAM)
  - NOR Flash память
  - PSRAM (4 банка памяти)
- Два банка NAND Flash с коррекцией ошибок до 8 КБ данныха
- 16-бит PC Card совместимые устройства
- Поддерживает пакетный режим доступа синхронных устройств (NOR Flash и PSRAM)
- Шина данных 8- или 16-бит ширины
- Независимый выбор чипа для каждого банка памяти
- Независимая конфигурация каждого банка памяти
- Программируемые времянки для множества устройств:
  - Программируемые состояния ожидания (до 15)
  - Программируемые циклы реверсных передач шины (до 15)
  - Программируемые задержки разрешения вывода и записи (до 15)
  - Независимые времянки чтения/записи и протоколов
- Выходы разрешения записи и выбора байтовых линий для устройств PSRAM и SRAM
- Трансляция 32-бит передач AHB в последовательный 16- или 8-бит доступ внешних устройств
- FIFO записи из двух 32-бит слов для хранения данных (не адресов) как буфер пакетных передач записи AHB. Так можно писать в медленную память, быстро освобождая AHB для других операций. Буферизуется только один пакет: при появлении нового пакета или отдельной передачи при незавершённой операции AHB сливается FIFO. FSMC вставит состояния ожидания до завершения текущего доступа.
- Внешнее управление асинхронным ожиданием

Регистры описания устройств FSMC обычно пишутся во время загрузки и не меняются до сброса. Но их можно поменять в любое время.

21.2. Блок-схема FSMC

FSMC состоит из четырёх основных блоков:

- Интерфейс AHB (включая регистры конфигурации FSMC)
- Контроллер NOR Flash/PSRAM
- Контроллер NAND Flash/PC Card
- Интерфейс внешних устройств



21.3. Интерфейс AHB

Интерфейс ведомого AHB позволяет CPU и другим ведущим устройствам на шине обращаться к внешней статической памяти.

Передачи AHB транслируются в протокол внешнего устройства. В частности, 32-бит передачи AHB разбиваются на последовательные 16- или 8-бит передачи. Chip Select удерживается низким для 32-бит выравненных передач или переключается между последовательными 32-бит невыравненными передачами.

FSMC выдаёт ошибку AHB:

- При обращении к не разрешённому банку FSMC
- При обращении к банку NOR Flash со сброшенным битом `FACCEN` в регистре `FSMC_BCRx`.
- При обращении к банку PC Card с низкоим сигналом на входе `FSMC_CD` (Card Presence Detection).

Действие ошибки AHB зависит от ведущего на шине:

- Если это Cortex-M3 CPU, то выдаётся тяжёлый сбой
- Если это DMA, то выдаётся ошибка DMA и этот канал автоматически отключается.

FSMC тактируется от AHB (HCLK).

21.3.1. Поддерживаемая память и передачи

Общие правила передачи

Передачи данных AHB могут быть шириной 8-, 16- или 32-бита, а у внешних устройств она фиксированная. Это может привести к несовместимости.

Стало быть простые правила передачи таковы:

- Размер передачи AHB и данных в памяти равны. Проблем никаких.
- Размер передачи AHB больше размера памяти. FSMC разбивает передачу AHB на несколько меньших с размером данных внешней памяти памяти.
- Размер передачи AHB меньше размера памяти. Асинхронные передачи могут быть несовместимы в зависимости от типа внешнего устройства.
  - Асинхронный доступ к устройствам с выбором байтов (SRAM, ROM, PSRAM).
    a) FSMC дозволяет запись нужных данных с помощью байтовых линий NBL[1:0]
    b) Чтение разрешено. Читаются все байты и ненужные отбрасываются. Линии NBL[1:0] удерживаются низкими.
  - Асинхронный доступ к устройствам без выбора байтов (NOR и 16-бит NAND Flash). Эта ситуация возникает при байтовом доступе к 16-бит Flash памяти. Понятно, байтовый доступ невозможен, так что:
    a) Запись невозможна.
    b) Чтение разрешено. Читаются все байты и ненужные отбрасываются. Линии NBL[1:0] удерживаются низкими.

Регистры конфигурации

Установки регистров FSMC описаны в *Секции 21.5.6* для NOR Flash/PSRAM и *Секции 21.6.8* для NAND Flash/PC Card.

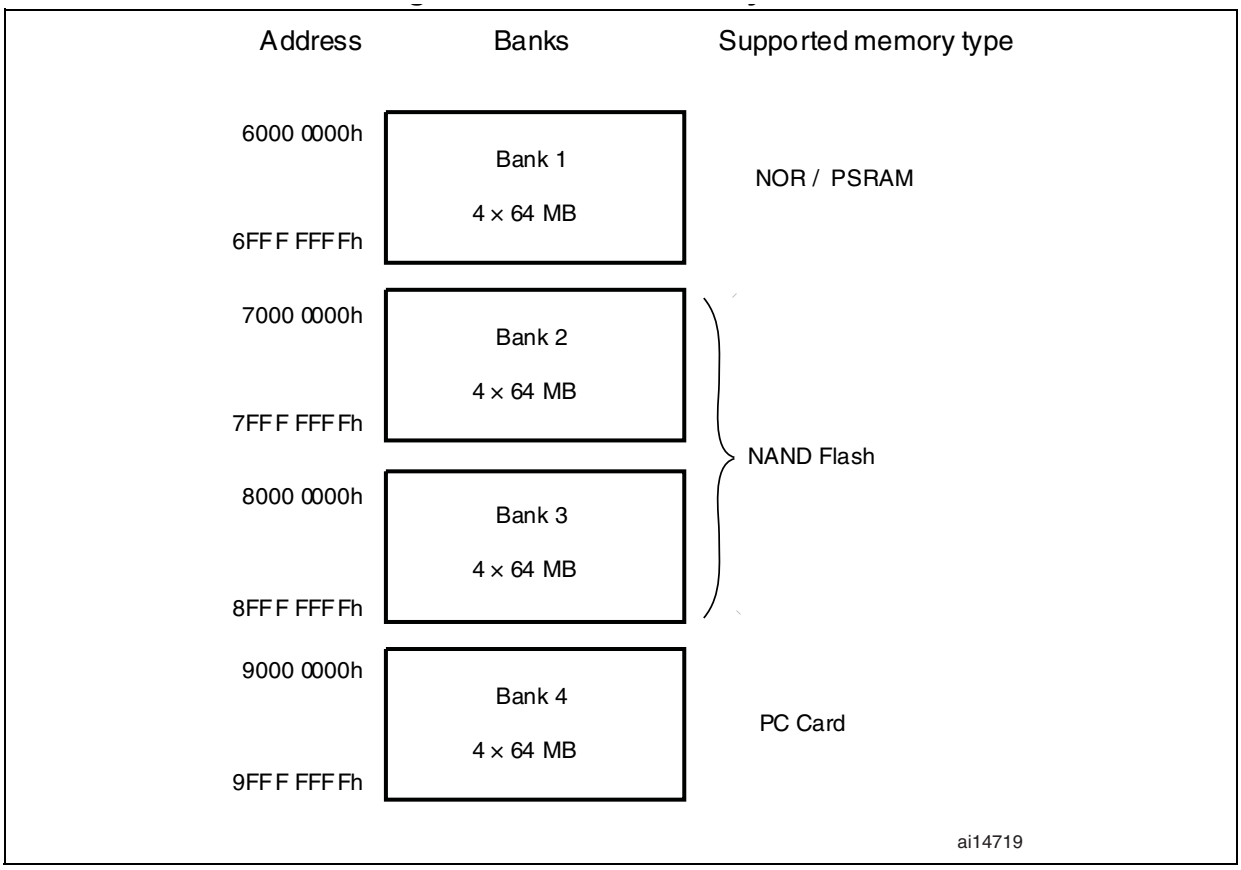
21.4. Отображение адресов внешних устройств

С точки зрения FSMC внешняя память разделяется на 4 банка по 256 Мбайт:

- Банк 1 адресует до 4 NOR Flash или PSRAM устройств памяти. Он разделён на 4 подбанка NOR/PSRAM четырьмя с 4 выделенными Chip Selects:
  - Банк 1 - NOR/PSRAM 1
  - Банк 1 - NOR/PSRAM 2
  - Банк 1 - NOR/PSRAM 3
  - Банк 1 - NOR/PSRAM 4
- Банки 2 и 3 адресуют устройства NAND Flash (по одному на банк)
- Банк 4 адресует устройства PC Card

Тип каждого банка определяется в регистре конфигурации.

Банки памяти FSMC.



21.4.1. Адресация NOR/PSRAM

Биты HADDR[27:26] выбирают банк памяти:

HADDR[27:26](1)	Банк
00	Банк 1 - NOR/PSRAM 1
01	Банк 1 - NOR/PSRAM 2
10	Банк 1 - NOR/PSRAM 3
11	Банк 1 - NOR/PSRAM 4

1. HADDR это внутренние линии адреса AHB, транслируемые во внешнюю память.

Биты HADDR[25:0] содержат адрес внешней памяти. В таблице ниже приведено изменение HADDR при использовании разных типов памяти.

Ширина памяти(1)	Адрес в памяти	Максимальная ёмкость памяти (бит)
8 бит	HADDR[25:0]	64 Mbyte x 8 = 512 Mbit
16 бит	HADDR[25:1] >> 1	64 Mbyte/2 x 16 = 512 Mbit

1. При 16-бит внешней памяти FSMC для адреса внешней памяти FSMC_A[24:0] использует HADDR[25:1].
При ширине памяти (16-бит или 8-бит) FSMC_A[0] надо подключать к адресу внешней памяти A[0].

Поддержка закольцевания NOR Flash/PSRAM.

Кольцевой пакетный режим синхронной памяти не поддерживается. Память нужно ставить в линейный пакетный режим с неопределённой длиной.

21.4.2. Адресация NAND/PC Card

В этом случае доступны три банка, поделённые на пространства.

Адрес начала	Адрес конца	Банк FSMC	Пространство	Регистр времянки
0x9C00 0000	0x9FFF FFFF	Банк 4 - PC card	I/O	FSMC_PIO4 (0xB0)
0x9800 0000	0x9BFF FFFF		Attribute	FSMC_PATT4 (0xAC)
0x9000 0000	0x93FF FFFF		Common	FSMC_PMEM4 (0xA8)
0x8800 0000	0x8BFF FFFF	Банк 3 - NAND Flash	Attribute	FSMC_PATT3 (0x8C)
0x8000 0000	0x83FF FFFF		Common	FSMC_PMEM3 (0x88)
0x7800 0000	0x7BFF FFFF	Банк 2- NAND Flash	Attribute	FSMC_PATT2 (0x6C)
0x7000 0000	0x73FF FFFF		Common	FSMC_PMEM2 (0x68)

В NAND Flash памяти пространства общей памяти и памяти атрибутов подразделяются на три секции, размещённые в младших 256 КБ:

- Секция данных (первые 64 КБ в пространстве памяти общей/атрибутов)
- Секция команд (вторые 64 КБ в пространстве памяти общей/атрибутов)
- Секция адресов (следующие 128 КБ в пространстве памяти общей/атрибутов)

Имя секции	HADDR[17:16]	Диапазон адресов
Секция адресов	1X	0x020000-0x03FFFF
Секция команд	01	0x010000-0x01FFFF
Секция данных	00	0x000000-0x0FFFF

При доступе к NAND Flash памяти используются 3 секции:

- **Для посылки команд**: используется секция команд.
- **Для задания адресов чтения и записи**: используется секция адресов. Поскольку адрес может быть 4 или 5 байтов длиной, то требуются несколько последовательных циклов записи.
- **Для чтения и записи данных**: используется секция данных.

Так как NAND Flash память автоматически инкрементирует адреса, то для последовательного доступа к памяти данных адреса доступа изменять не надо.

21.5. Контроллер NOR Flash/PSRAM

FSMC выдаёт нужную времянку сигналов для следующих типов памяти:

- Асинхронные SRAM и ROM
  - 8-бит
  - 16-бит
  - 32-бит
- PSRAM (Сотовая RAM)
  - Асинхронный режим
  - Пакетный режим синхронного доступа
- NOR Flash
  - Асинхронный режим
  - Пакетный режим синхронного доступа
  - Мультиплексированный или немультиплексированный

FSMC выдаёт раздельные сигналы выбора на банк `NE[4:1]`. Все другие сигналы (адреса, данные и управление) общие.

При синхронном доступе чтении/записи FSMC подаёт на выбранное внешнее устройство такты (CLK), кратные HCLK. Размер банков фиксированный (64 МБ).

Банки конфигурируются специально выделенными регистрами (см. *Секцию 21.5.6*).

Параметры программируемой памяти включают времянку доступа и поддержку управления ожиданием (для PSRAM и NOR Flash в пакетном режиме).

Параметры доступа программируемой NOR/PSRAM.

Параметр	Функция	Режим доступа	Единицы	Min.	Max.
Установка адреса	Длительность фазы установки адреса	Асинхронный	Такты AHB (HCLK)	1	16
Удержание адреса	Длительность фазы удержания адреса	Асинхронный, мультиплекс. В/Выв.	Такты AHB (HCLK)	2	16
Установка данных	Длительность фазы установки данных	Асинхронный	Такты AHB (HCLK)	2	256
Реверс шины	Длительность фазы реверса шины	Асинхронный и синхронные чтение/запись	Такты AHB (HCLK)	1	16
Делитель частоты	Число тактов AHB (HCLK) на один такт памяти (CLK)	Синхронный	Такты AHB (HCLK)	2	16
Задержка данных	Число тактов, подаваемых памяти перед первым данным пакета	Синхронный	Такты памяти (CLK)	2	17

21.5.1. Сигналы интерфейса внешней памяти

Префикс "N" указывает на низкий активный уровень сигнала. NOR Flash и PSRAM адресуются 16-бит словами. Максимальная ёмкость 512 Мбит (26 линий адреса).

NOR Flash, немультиплексированный Вв./Выв.

Сигнал FSMC	Вв./Выв.	Функция
CLK	Выв.	Такты (для синхронного доступа)
A[25:0]	Выв.	Шина адреса
D[15:0]	Вв./Выв.	Двунаправленная шина данных
NE[x]	Выв.	Выбор чипа, x = 1..4
NOE	Выв.	Разрешение выхода
NWE	Выв.	Разрешение записи
NL(=NADV)	Выв.	Разрешение защёлки (в некоторых NOR Flash называется достоверным адресом, NADV)
NWAIT	Вв.	Входной сигнал ожидания от NOR Flash

NOR Flash, мультиплексированный Вв./Выв.

Сигнал FSMC	Вв./Выв.	Функция
CLK	Выв.	Такты (для синхронного доступа)
A[25:16]	Выв.	Шина адреса
AD[15:0]	Вв./Выв.	16-бит мультиплексированная, двунаправленная шина адреса/данных
NE[x]	Выв.	Выбор чипа, x = 1..4
NOE	Выв.	Разрешение выхода
NWE	Выв.	Разрешение записи
NL(=NADV)	Выв.	Разрешение защёлки (в некоторых NOR Flash называется достоверным адресом, NADV)
NWAIT	Вв.	Входной сигнал ожидания от NOR Flash

PSRAM/SRAM, немультиплексированный Вв./Выв.

Сигнал FSMC	Вв./Выв.	Функция
CLK	Выв.	Такты (для синхронного доступа)
A[25:0]	Выв.	Шина адреса
D[15:0]	Вв./Выв.	Двунаправленная шина данных
NE[x]	Выв.	Выбор чипа, x = 1..4 (называемая NCE в PSRAM (Cellular RAM т.е. CRAM))
NOE	Выв.	Разрешение выхода
NWE	Выв.	Разрешение записи
NL(=NADV)	Выв.	Достоверный адрес ввода в PSRAM (имя в памяти NADV)
NWAIT	Вв.	Входной сигнал ожидания от PSRAM
NBL[1]	Выв.	Разрешение старшего байта (имя в памяти NUB)
NBL[0]	Выв.	Разрешение младшего байта (имя в памяти NLB)

21.5.2. Поддерживаемая память и передачи

Таблица ниже даёт пример поддерживаемых устройств, режимов доступа и передач с 16-бит шиной данных для NOR, PSRAM и SRAM. Недоступные (или неподдерживаемые) передачи FSMC выделены красным цветом.

Уст-во	Режим	Ч/З	Размер данных AHB	Размер данных памяти	Можно	Комментарии
NOR Flash (мультипл. и немультипл.)	Асинхр.	Ч	8	16	Д	—
	Асинхр.	З	8	16	Н	—
	Асинхр.	Ч	16	16	Д	—
	Асинхр.	З	16	16	Д	—
	Асинхр.	Ч	32	16	Д	За два обращения FSMC
	Асинхр.	З	32	16	Д	За два обращения FSMC
	Асинхр. страничн.	Ч	-	16	Н	Не поддерживается
	Синхр.	Ч	8	16	Н	—
	Синхр.	Ч	16	16	Д	—
	Синхр.	Ч	32	16	Д	—
PSRAM (мультипл. и немультипл.)	Асинхр.	Ч	8	16	Д	—
	Асинхр.	З	8	16	Д	Используются линии байтов NBL[1:0]
	Асинхр.	Ч	16	16	Д	—
	Асинхр.	З	16	16	Д	—
	Асинхр.	Ч	32	16	Д	За два обращения FSMC
	Асинхр.	З	32	16	Д	Используются линии байтов NBL[1:0]
	Асинхр. страничн.	Ч	-	16	Н	Не поддерживается

Уст-во	Режим	Ч/З	Размер данных AHB	Размер данных памяти	Можно	Комментарии
	Синхр.	Ч	8	16	Н	—
	Синхр.	Ч	16	16	Д	—
	Синхр.	Ч	32	16	Д	—
	Синхр.	З	8	16	Д	Используются линии байтов NBL[1:0]
	Синхр.	З	16/32	16	Д	—
SRAM и ROM	Асинхр.	Ч	8/16	16	Д	—
	Асинхр.	З	8/16	16	Д	Используются линии байтов NBL[1:0]
	Асинхр.	Ч	32	16	Д	За два обращения FSMC
	Асинхр.	З	32	16	Д	За два обращения FSMC. Используются линии байтов NBL[1:0]

21.5.3. Синхронизация сигналов

- Все выходные сигналы контроллера изменяются по переднему фронту HCLK
- В синхронном режиме (чтение и запись), все выходные сигналы изменяются по переднему фронту HCLK. Независимо от значения CLKDIV все выходы меняются так:
  - NOEL/NWEL/NEL/NADVL/NADVH/NBLL/Адрес_Достоверен меняются по заднему фронту FSMC_CLK.
  - NOEH/NWEH/NEH/NOEH/NBLH/Адрес_Недостоверен меняются по переднему фронту FSMC_CLK.

21.5.4. Асинхронные передачи контроллера NOR Flash/PSRAM

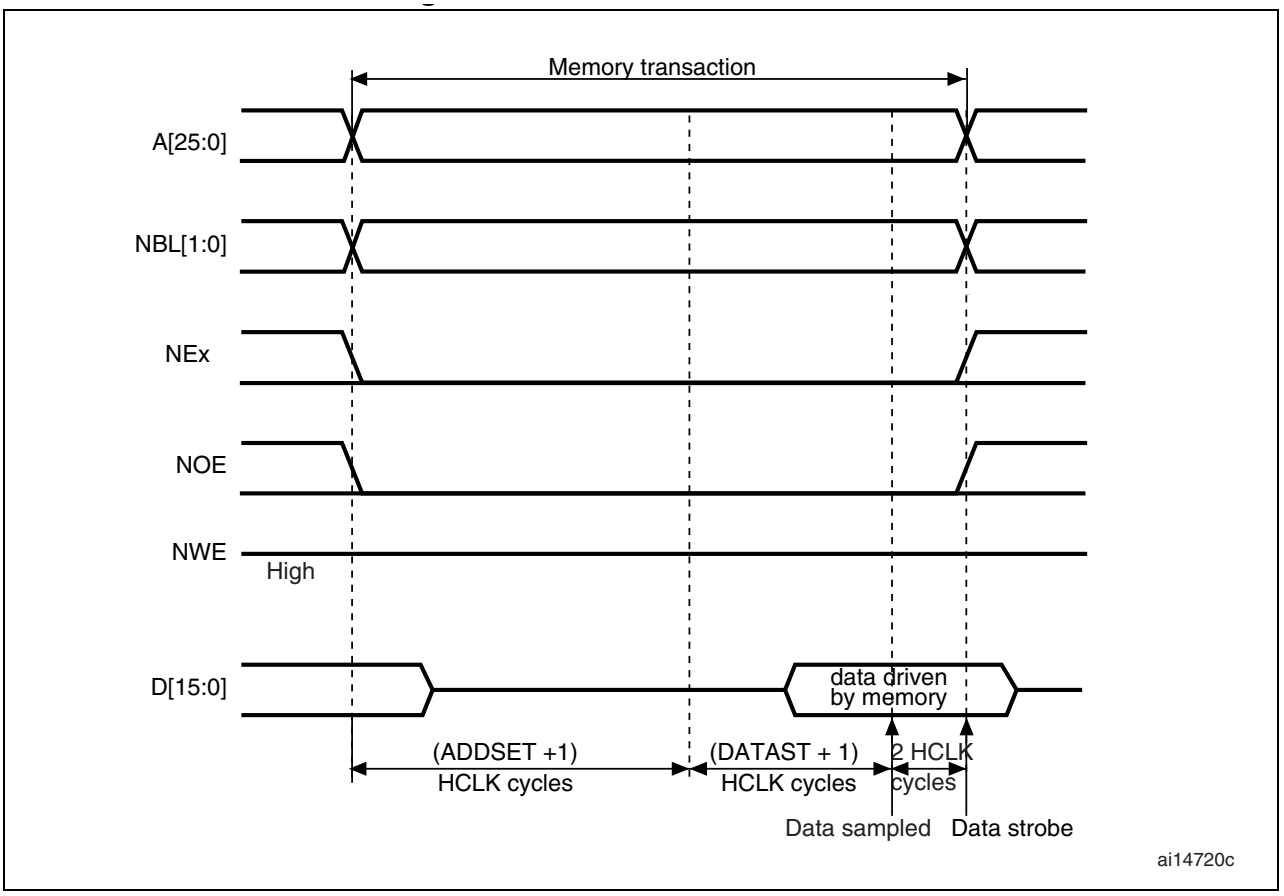
Асинхронная статическая память (NOR Flash, PSRAM, SRAM)

- Сигналя синхронизированы внутренними тактами HCLK, на память они не передаются.
- FSMC всегда считывает данные перед снятием сигналов NOE. Так соблюдается время удержания данных памятью.
- Если разрешён расширенный режим (стоит бит `EXTMOD` в регистре `FSMC_BCRx`), то доступны до 4 расширенных режима (A, B, C и D). При чтении и записи их можно смешивать. Например, читать можно в режиме A, а писать в режиме B.
- Если расширенный режим отключён (бит `EXTMOD` в регистре `FSMC_BCRx` сброшен), то FSMC может работать в режиме 1 или 2:
  - С памятью типа SRAM/PSRAM (`MTYP[0:1]` = `0x0` или `0x01` в `FSMC_BCRx`) по умолчанию используется Режим 1.
  - С памятью типа NOR (`MTYP[0:1]` = `0x10` в `FSMC_BCRx`) по умолчанию используется Режим 2.

Режим 1 - SRAM/PSRAM (CRAM)

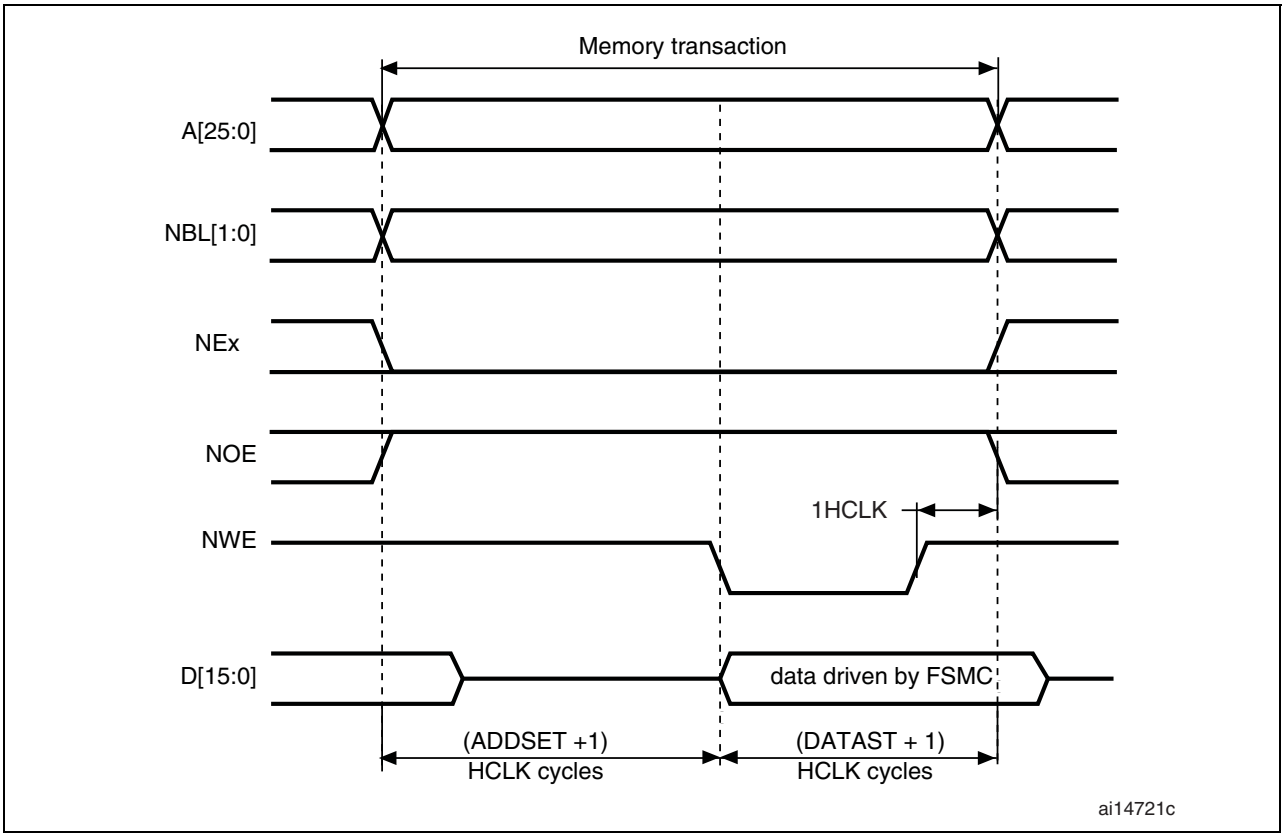
На рисунках ниже показаны передачи чтения и записи в поддерживаемых режимах и нужные установки регистров `FSMC_BCRx` и `FSMC_BTRx`/`FSMC_BWTRx`.

Режим 1, чтение.



1. При чтении NBL[1:0] удерживается низким.

Режим 1, запись.



Один такт HCLK в конце записи гарантирует удержание адреса и данных после снятия сигнала NWE. Из-за этого значение DATAST должно быть больше нуля (DATAST > 0).

Таблица 109. Биты FSMC_BCRx.

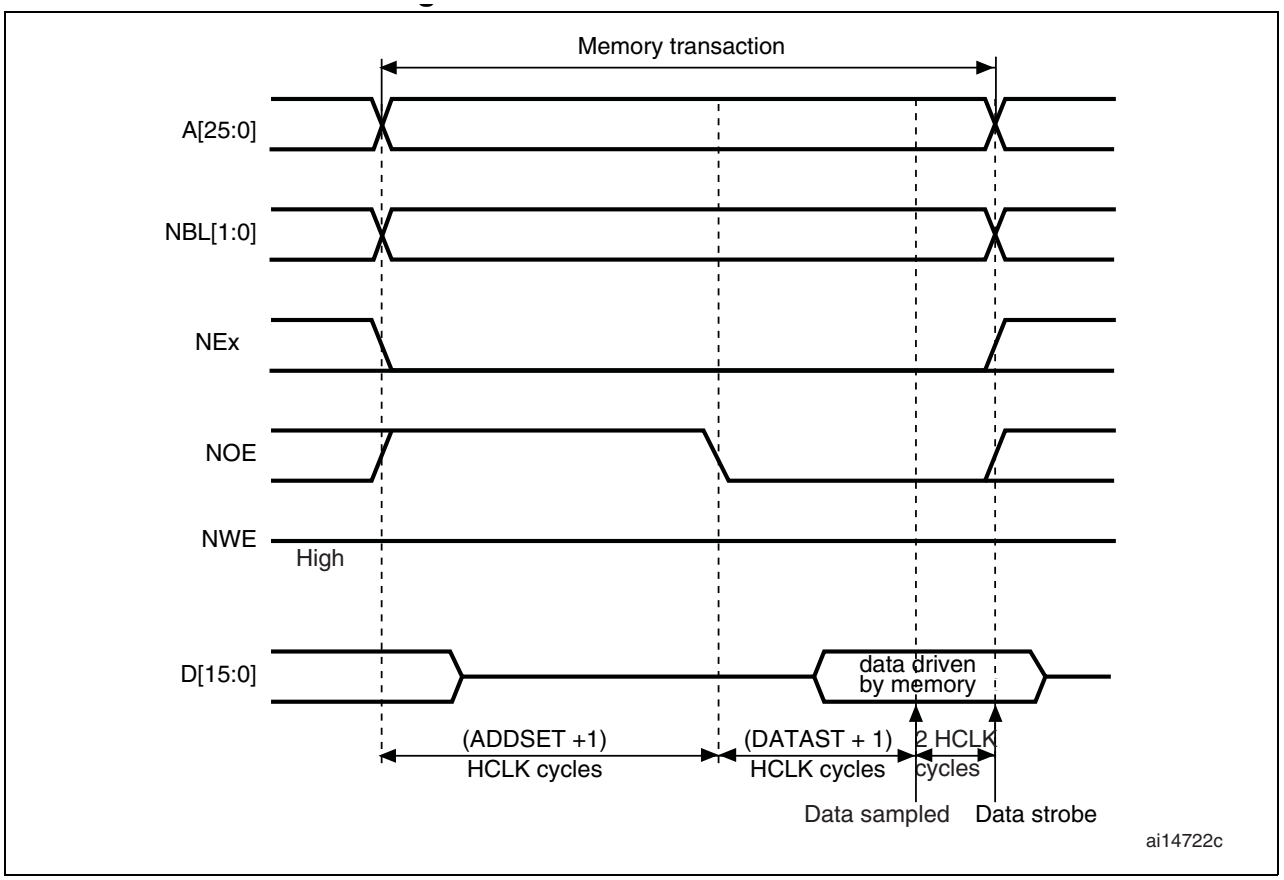
Номер	Имя	Устанавливаемое значение
31-20	Резерв	0x000
19	CBURSTRW	0x0 (в асинхронном режиме не влияет)
18:16	CPSIZE	0x0 (в асинхронном режиме не влияет)
15	ASYNCWAIT	1 при поддержке памятью. Иначе 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (в асинхронном режиме не влияет)
12	WREN	Как нужно
11	WAITCFG	Всё равно.
10	WRAPMOD	0x0
9	WAITPOL	Имеет смысл только при стоящем бите 15
8	BURSTEN	0x0
7	Резерв	0x1
6	FACCEN	Всё равно.
5-4	MWID	Как нужно
3-2	MTYP[0:1]	Как нужно, исключая 0x2 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 110. Биты FSMC_BTRx.

Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	Всё равно.
27-24	CPSIZE	Всё равно.
23-20	CLKDIV	Всё равно.
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+1 циклов HCLK при записи, DATAST+3 циклов HCLK при чтении). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK).

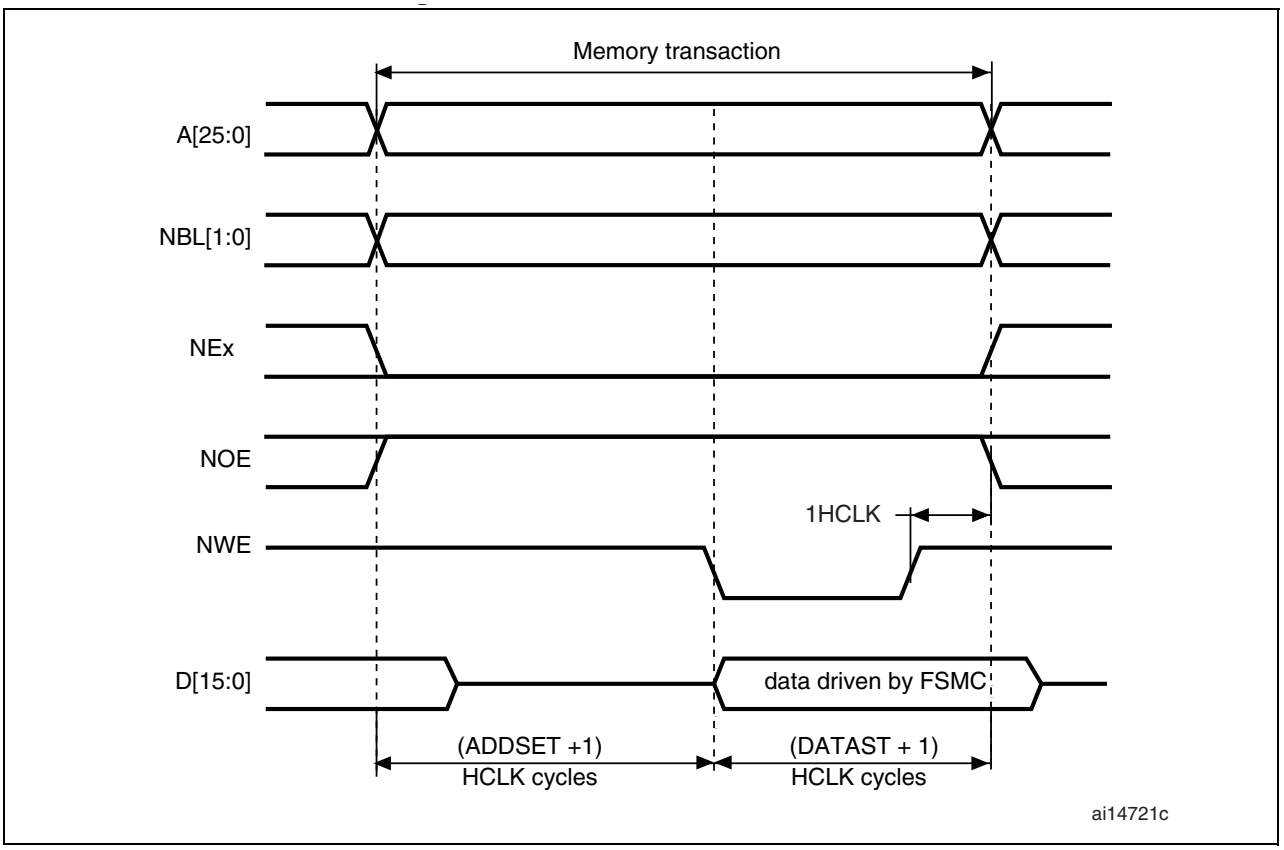
Режим A - SRAM/PSRAM (CRAM), переключение OE.

Режим A, чтение.



1. При чтении NBL[1:0] удерживается низким.

Режим A, запись.



В отличие от Режима 1 сигнал NOE переключается и времена чтения и записи независимые.

Таблица 111. Биты FSMC_BCRx.

Номер	Имя	Устанавливаемое значение
31-20	Резерв	0x000
19	CBURSTRW	0x0 (в асинхронном режиме не влияет)
18:16	CPSIZE	0x0 (в асинхронном режиме не влияет)
15	ASYNCWAIT	1 при поддержке памятью. Иначе 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (в асинхронном режиме не влияет)
12	WREN	Как нужно
11	WAITCFG	Всё равно.
10	WRAPMOD	0x0
9	WAITPOL	Имеет смысл только при стоящем бите 15
8	BURSTEN	0x0
7	Резерв	0x1
6	FACCEN	Всё равно.
5-4	MWID	Как нужно
3-2	MTYP[0:1]	Как нужно, исключая 0x2 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 112. Биты FSMC_BTRx.

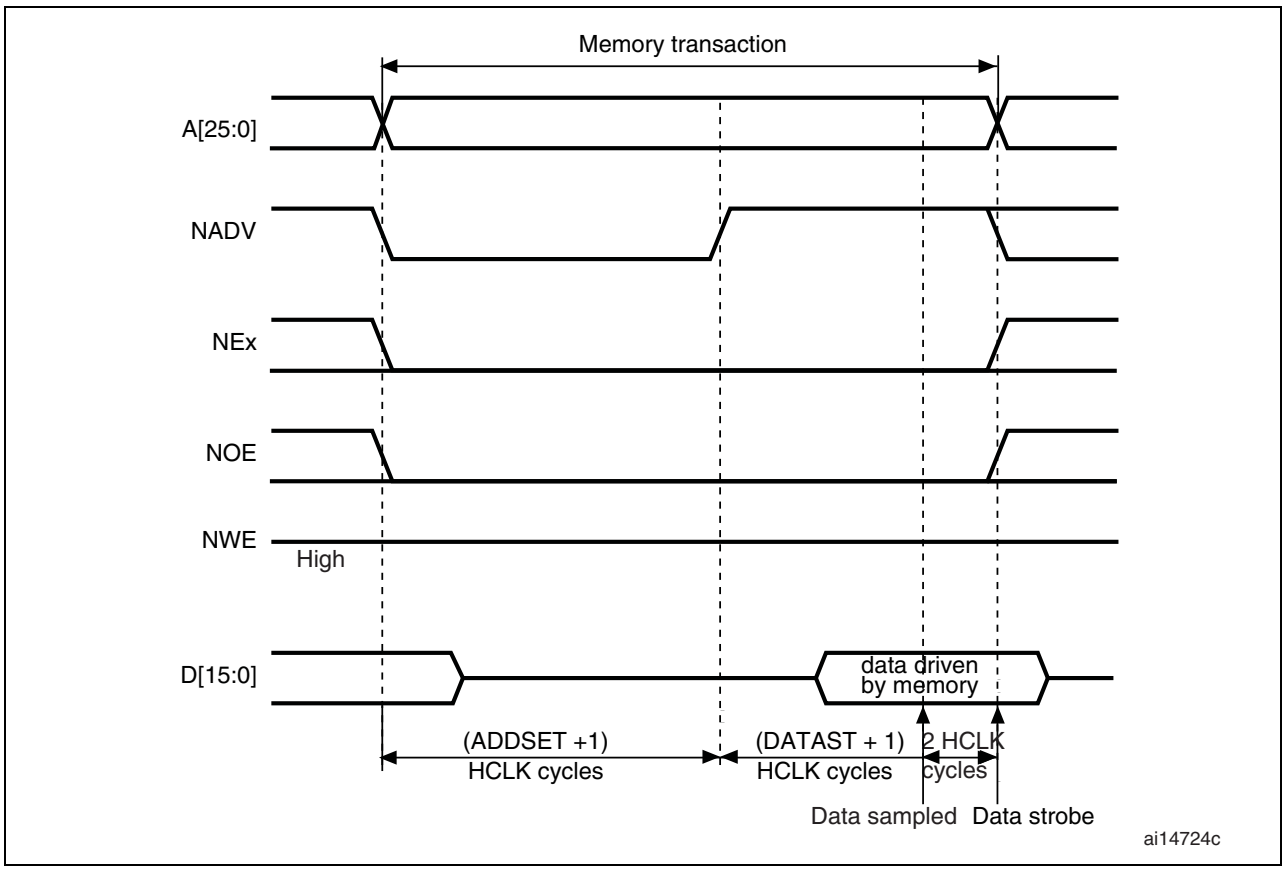
Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x0
27-24	CPSIZE	Всё равно.
23-20	CLKDIV	Всё равно.
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+3 циклов HCLK при чтении). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK).

Таблица 113. Биты FSMC_BWTRx.

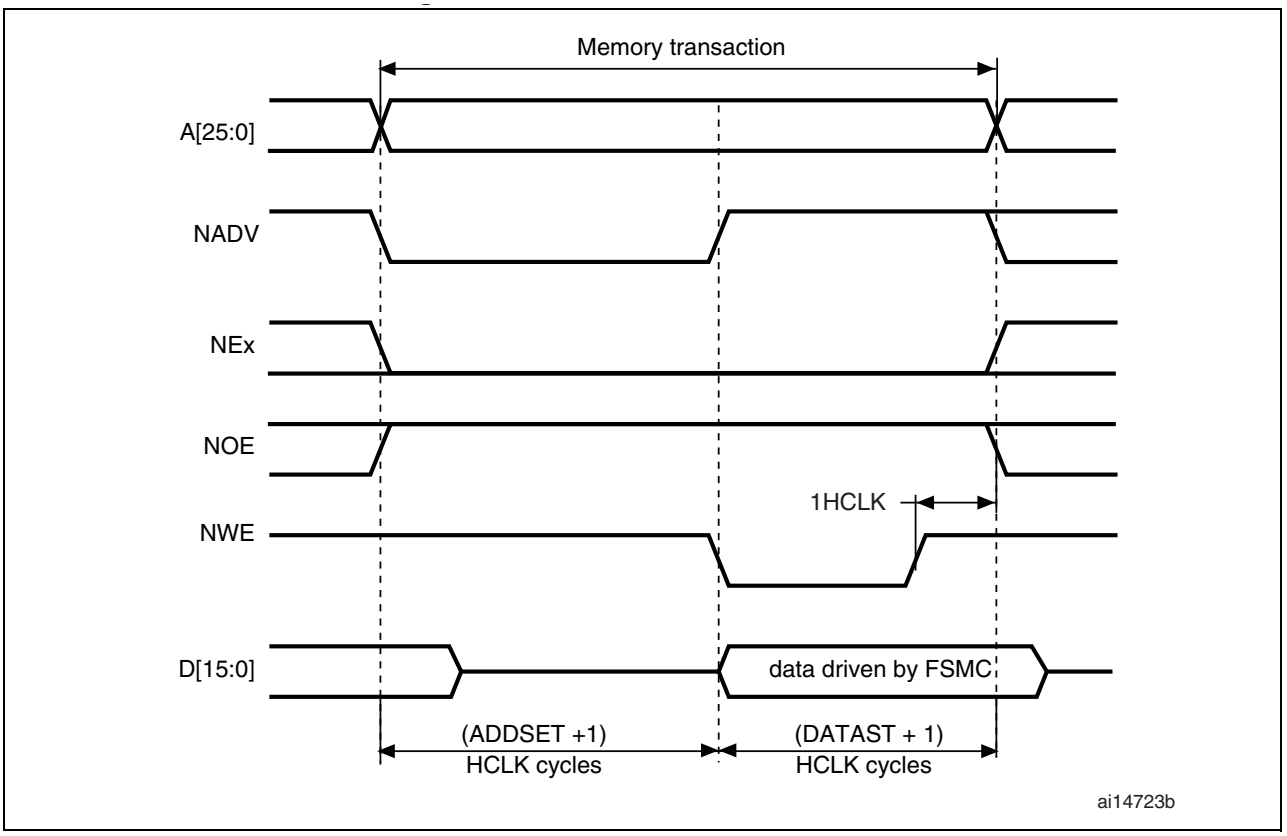
Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Всё равно.
23-20	CLKDIV	Всё равно.
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+1 циклов HCLK при записи, DATAST+3 циклов HCLK при чтении). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK для записи).

Режим 2/B - NOR Flash

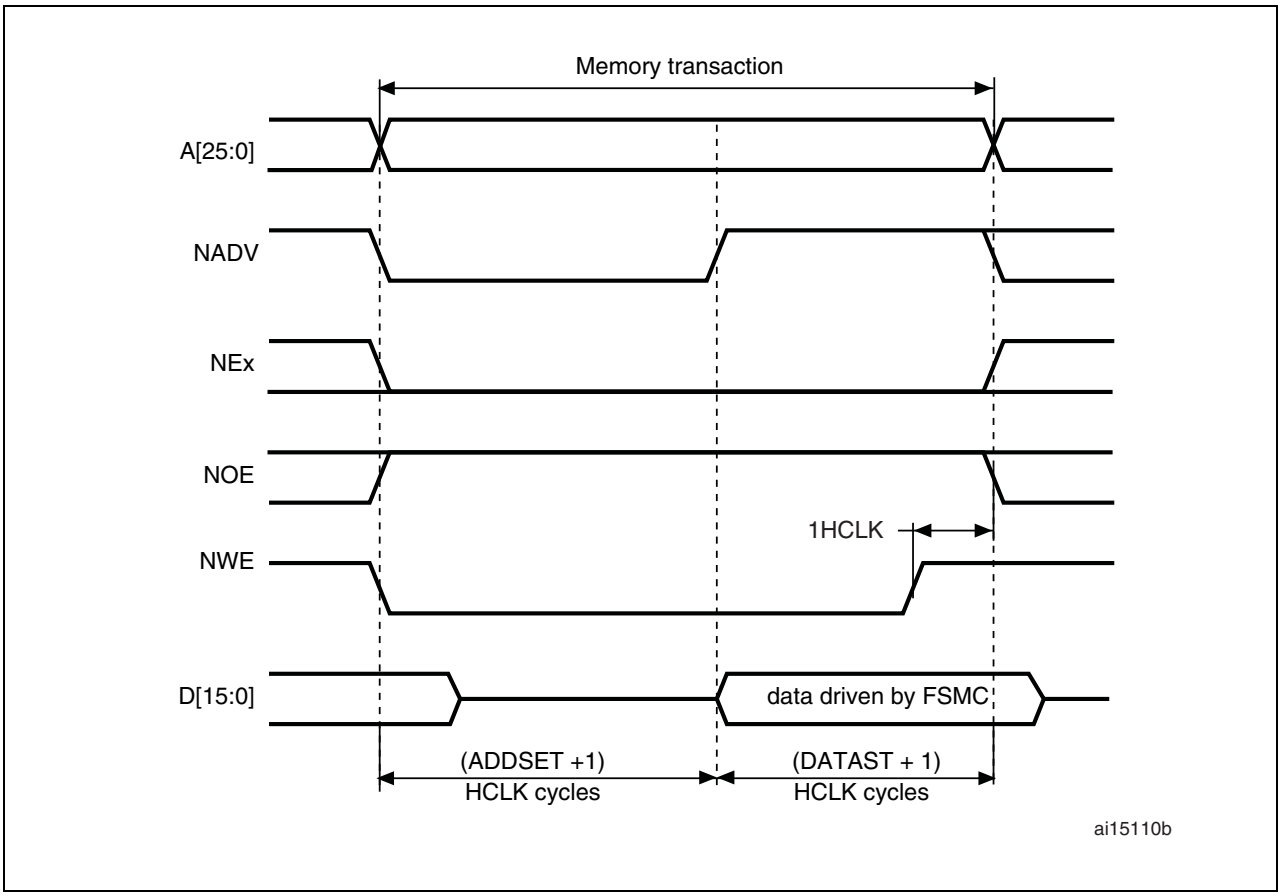
Режим 2/B, чтение.



Режим 2, запись.



Режим B, запись.



Отличие с Режимом 1 в переключении NWE и независимыми временами чтения и записи в расширенном режиме (Mode B).

Таблица 114. Биты FSMC_BCRx.

Номер	Имя	Устанавливаемое значение
31-20	Резерв	0x000
19	CBURSTRW	0x0 (в асинхронном режиме не влияет)
18:16	CPSIZE	0x0 (в асинхронном режиме не влияет)
15	ASYNCWAIT	1 при поддержке памятью. Иначе 0.
14	EXTMOD	0x1 в Режиме B, 0x0 в Режиме 2
13	WAITEN	0x0 (в асинхронном режиме не влияет)
12	WREN	Как нужно
11	WAITCFG	Всё равно.
10	WRAPMOD	0x0
9	WAITPOL	Имеет смысл только при стоящем бите 15
8	BURSTEN	0x0
7	Резерв	0x1
6	FACCEN	0x1
5-4	MWID	Как нужно
3-2	MTYP[0:1]	0x2 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 115. Биты FSMC_BTRx.

Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x1
27-24	DATLAT	Всё равно.
23-20	CLKDIV	Всё равно.
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+3 циклов HCLK при чтении). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK).

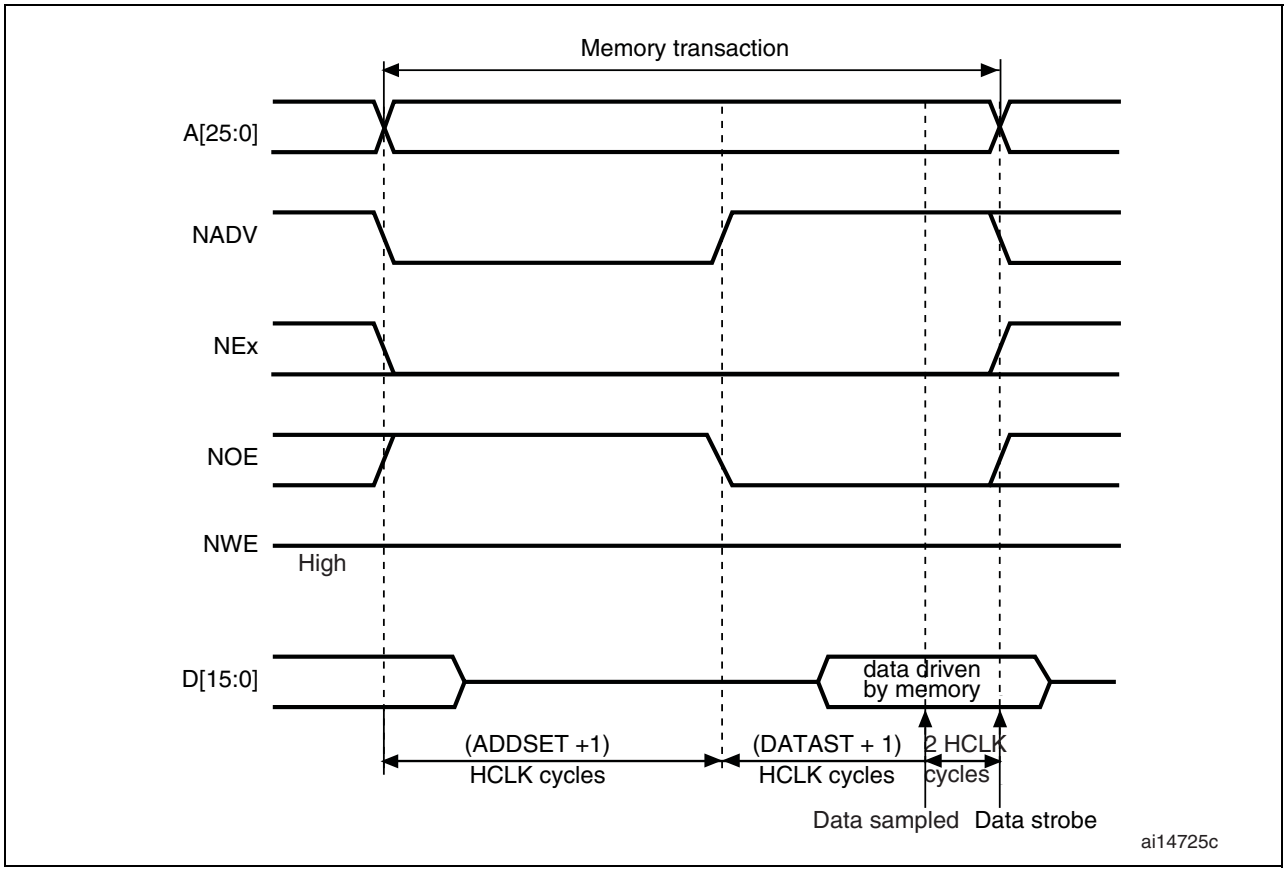
Таблица 116. Биты FSMC_BWTRx.

Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x1
27-24	DATLAT	Всё равно.
23-20	CLKDIV	Всё равно.
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+1 циклов HCLK при записи, DATAST+3 циклов HCLK при чтении). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK для записи).

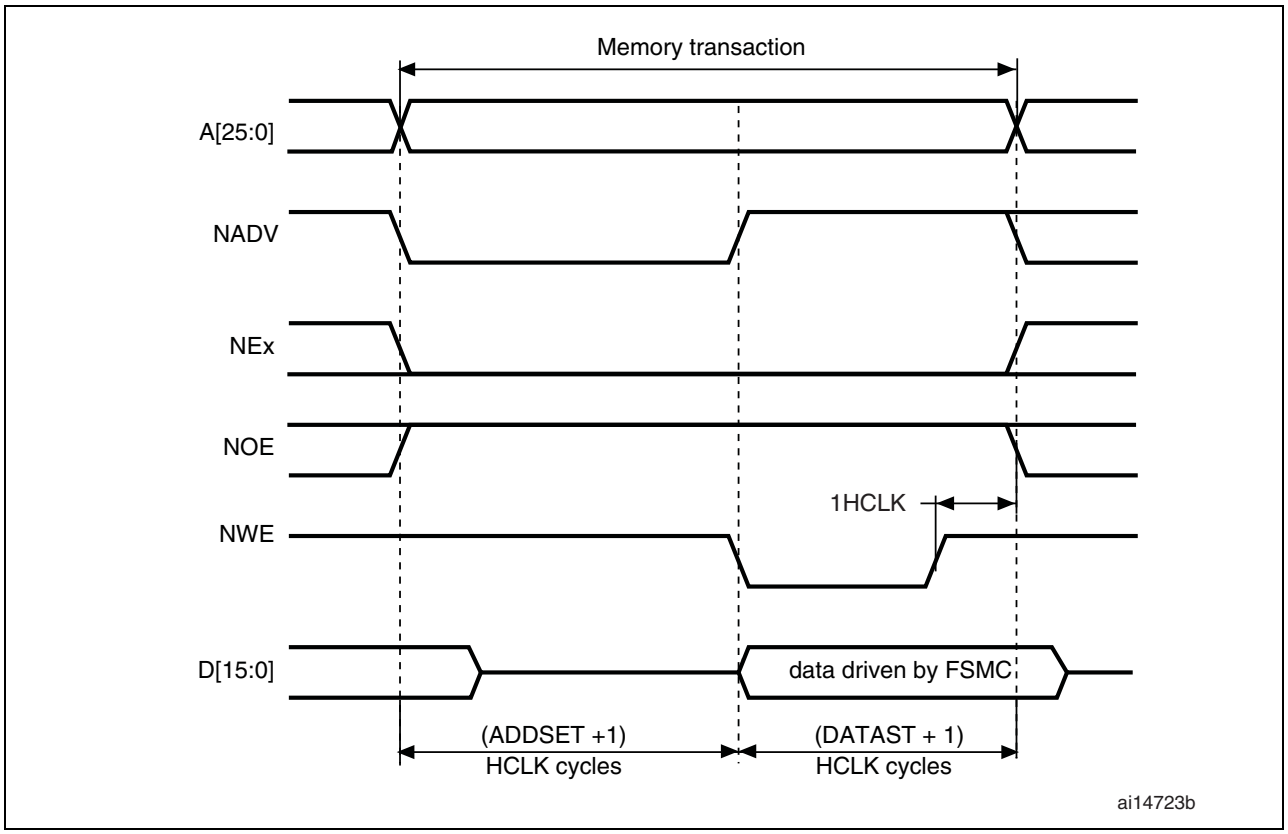
Регистр FSMC_BWTRx достоверен только в расширенном режиме B, иначе его содержимое бестолково.

Режим C - NOR Flash - переключение OE.

Режим C, чтение.



Режим C, запись.



От Режима 1 отличается переключением NOE и независимыми времянками чтения и записи.

Таблица 117. Биты FSMC_BCRx.

Номер	Имя	Устанавливаемое значение
31-20	Резерв	0x000
19	CBURSTRW	0x0 (в асинхронном режиме не влияет)
18:16	CPSIZE	0x0 (в асинхронном режиме не влияет)
15	ASYNCWAIT	1 при поддержке памятью. Иначе 0.
14	EXTMOD	0x1 в Режиме B, 0x0 в Режиме 2
13	WAITEN	0x0 (в асинхронном режиме не влияет)
12	WREN	Как нужно
11	WAITCFG	Всё равно.
10	WRAPMOD	0x0
9	WAITPOL	Имеет смысл только при стоящем бите 15
8	BURSTEN	0x0
7	Резерв	0x1
6	FACCEN	0x1
5-4	MWID	Как нужно
3-2	MTYP[0:1]	0x2 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 118. Биты FSMC_BTRx.

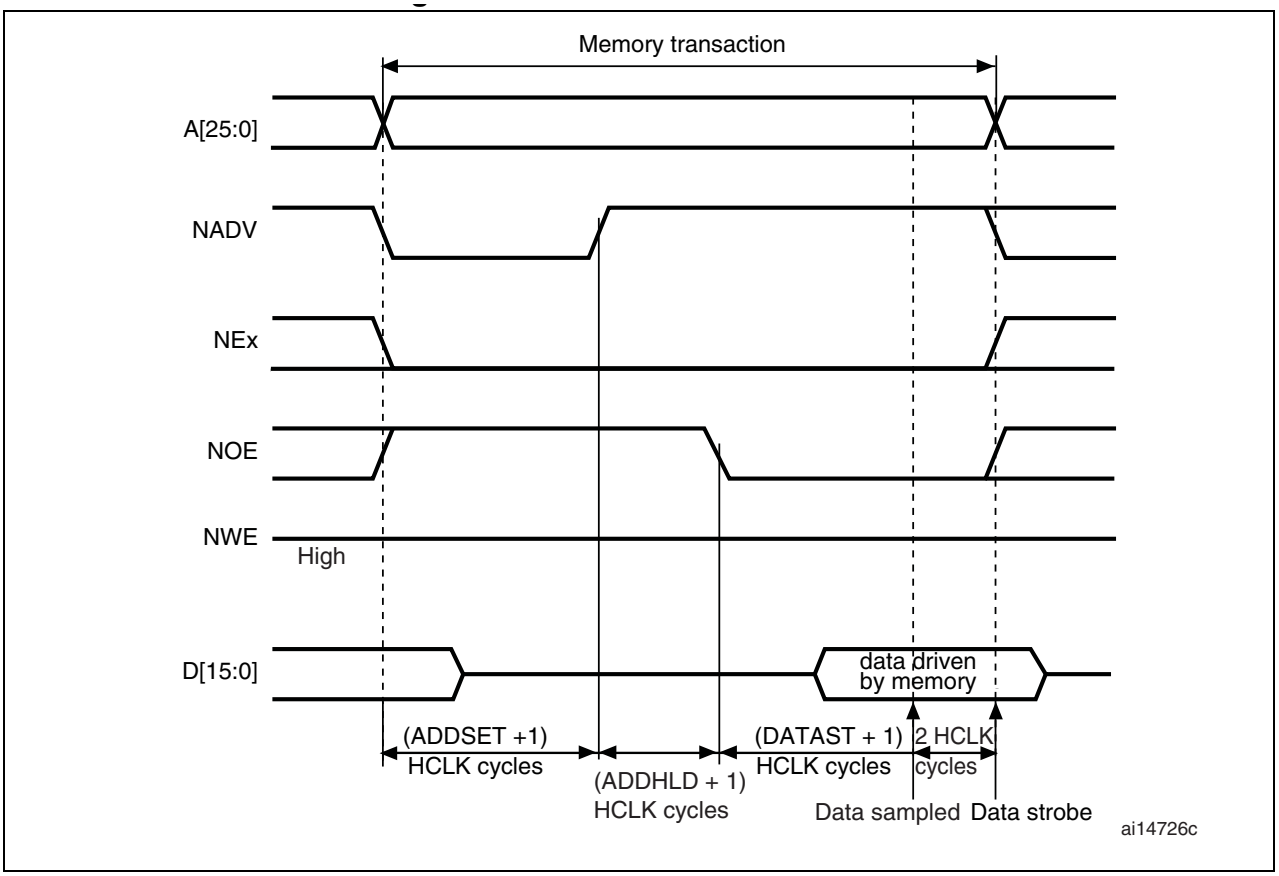
Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x2
27-24	DATLAT	0x0
23-20	CLKDIV	0x0
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+3 циклов HCLK при чтении). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK).

Таблица 119. Биты FSMC_BWTRx.

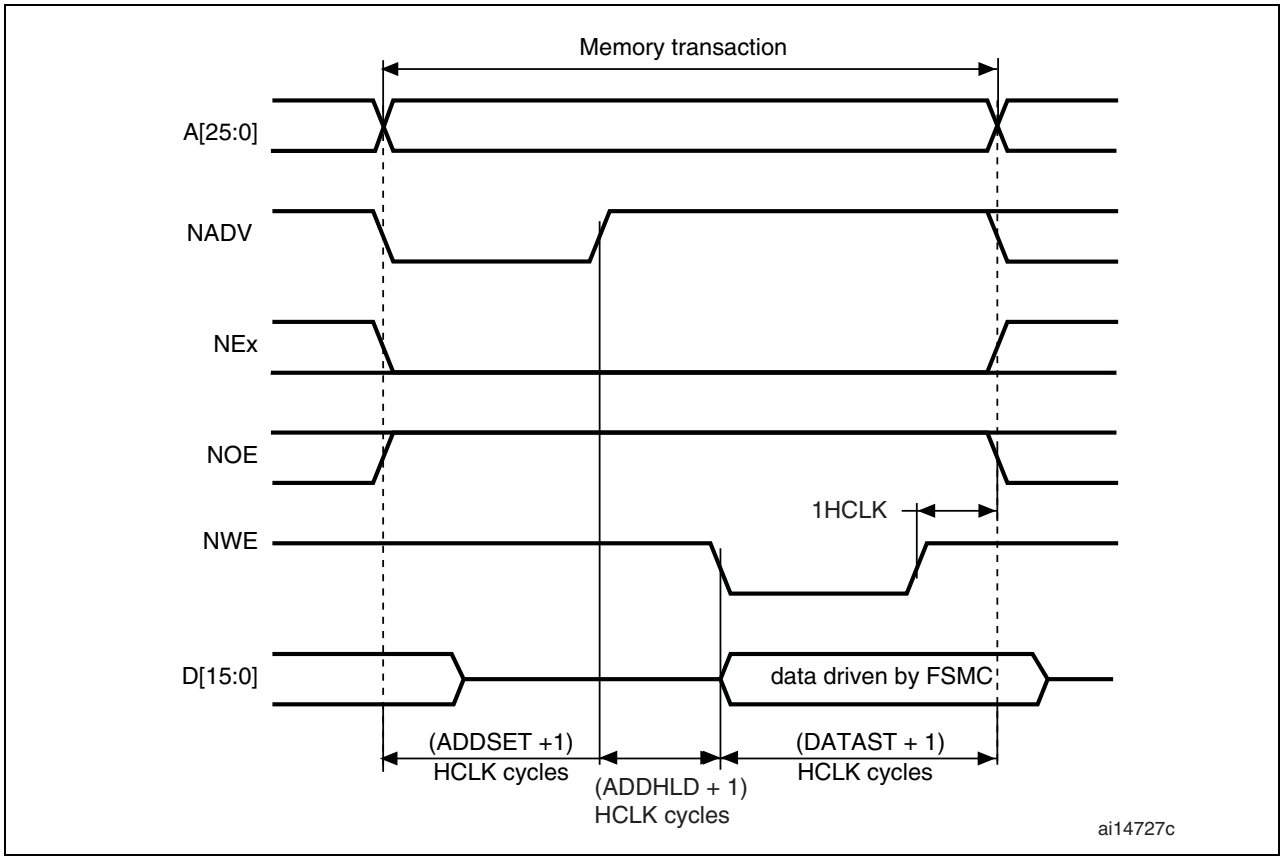
Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x2
27-24	DATLAT	Всё равно.
23-20	CLKDIV	Всё равно.
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+1 циклов HCLK при записи, DATAST+3 циклов HCLK при чтении). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK для записи).

Режим D, асинхронный доступ с расширенным адресом.

Режим D, чтение.



Режим D, запись.



От режима 1 отличается переключением NOE после изменения NADV и независимыми времянками чтения и записи.

Таблица 120. Биты FSMC_BCRx.

Номер	Имя	Устанавливаемое значение
31-20	Резерв	0x000
19	CBURSTRW	0x0 (в асинхронном режиме не влияет)
18:16	CPSIZE	0x0 (в асинхронном режиме не влияет)
15	ASYNCWAIT	1 при поддержке памятью. Иначе 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (в асинхронном режиме не влияет)
12	WREN	Как нужно
11	WAITCFG	Всё равно.
10	WRAPMOD	0x0
9	WAITPOL	Имеет смысл только при стоящем бите 15
8	BURSTEN	0x0
7	Резерв	0x1
6	FACCEN	Соответственно памяти.
5-4	MWID	Как нужно
3-2	MTYP[0:1]	Как нужно
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 121. Биты FSMC_BTRx.

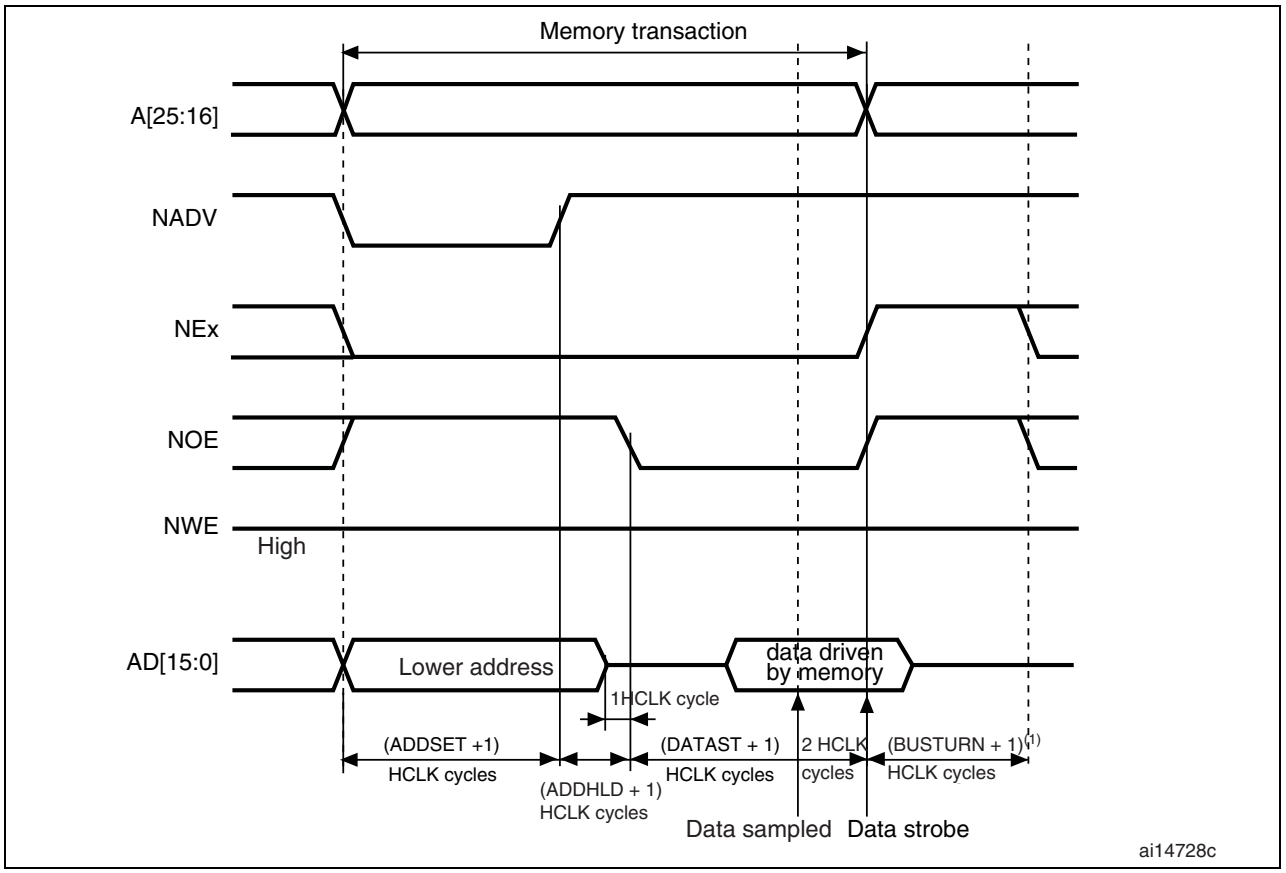
Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x3
27-24	DATLAT	Всё равно.
23-20	CLKDIV	Всё равно.
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+3 циклов HCLK при чтении). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Длительность средней фазы чтения (ADDHLD+1 циклов HCLK).
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK).

Таблица 122. Биты FSMC_BWTRx.

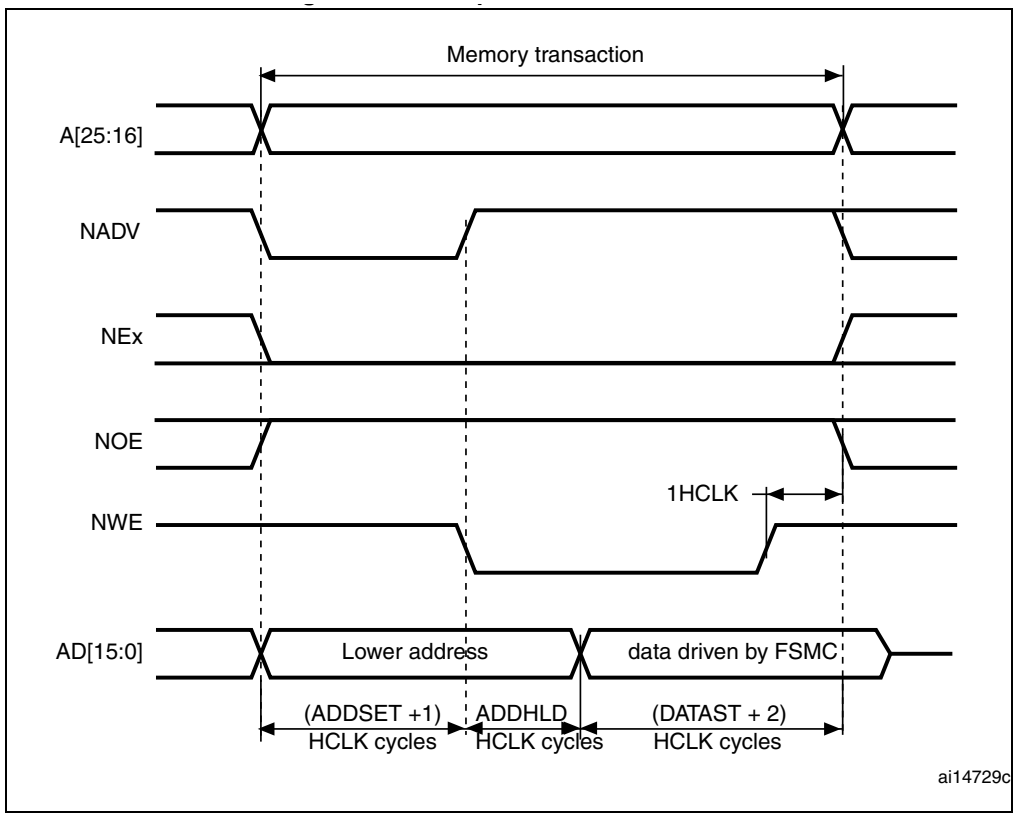
Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x3
27-24	DATLAT	0x0
23-20	CLKDIV	0x0
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+3 циклов HCLK при записи). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Длительность средней фазы записи (ADDHLD+1 циклов HCLK).
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK для записи).

Мультиплексный режим - мультиплексированный асинхронный доступ к NOR Flash памяти.

Мультиплексный режим, чтение.



Мультиплексный режим, запись.



От режима D отличается управлением младшей частью адреса на шине данных.

Таблица 123. Биты FSMC_BCRx.

Номер	Имя	Устанавливаемое значение
31-21	Резерв	0x000
19	CBURSTRW	0x0 (в асинхронном режиме не влияет)
18:16	CPSIZE	0x0 (в асинхронном режиме не влияет)
15	ASYNCWAIT	1 при поддержке памятью. Иначе 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (в асинхронном режиме не влияет)
12	WREN	Как нужно
11	WAITCFG	Всё равно.
10	WRAPMOD	0x0
9	WAITPOL	Имеет смысл только при стоящем бите 15
8	BURSTEN	0x0
7	Резерв	0x1
6	FACCEN	0x1
5-4	MWID	Как нужно
3-2	MTYP[0:1]	0x2 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 124. Биты FSMC_BTRx.

Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Всё равно.
23-20	CLKDIV	Всё равно.
19-16	BUSTURN	Длительность последней фазы доступа (BUSTURN+1 HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+3 циклов HCLK при чтении, DATAST+1 циклов HCLK при записи). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Длительность средней фазы доступа (ADDHLD+1 циклов HCLK). Не может быть равным 0 (минимум 1).
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK).

Управление ожиданием при асинхронном доступе.

Если асинхронная память умеет выставлять сигнал WAIT при своей неготовности к работе, то в регистре FSMC_BCRx надо ставить бит ASYNCWAIT.

Если сигнал WAIT активен (в зависимости от бита WAITPOL), то вторая фаза доступа (установка данных), определённая битами DATAST, удлиняется вплоть до снятия сигнала WAIT. В отличие от фазы данных, первые фазы доступа (Установка и удержание адреса), определённые битами ADDSET[3:0] и ADDHLD, к сигналу WAIT не чувствительны.

Фазу установки данных (DATAST в регистре FSMC_BCRx) нужно определять так, чтобы сигнал WAIT соответствовал следующим условиям:

- Для чтения: WAIT можно обнаружить за 4 такта HCLK до считывания данных или за 6 тактов HCLK до снятия NOE.
- Для записи: WAIT можно обнаружить за 4 такта HCLK до снятия NWE.

1. Память выставляет сигнал WAIT выравненным к переключаемым NOE/NWE:

$$DATAST \geq (4 \times HCLK) + max\_wait\_assertion\_time$$

2. Память выставляет сигнал WAIT выравненным к NEx (или NOE/NWE не переключаются):

если

$$max\_wait\_assertion\_time > фаза\_адреса + фаза\_удержания$$

то

$$DATAST \geq (4 \times HCLK) + \left(max\_wait\_assertion\_time - фаза\_адреса - фаза\_удержания\right)$$

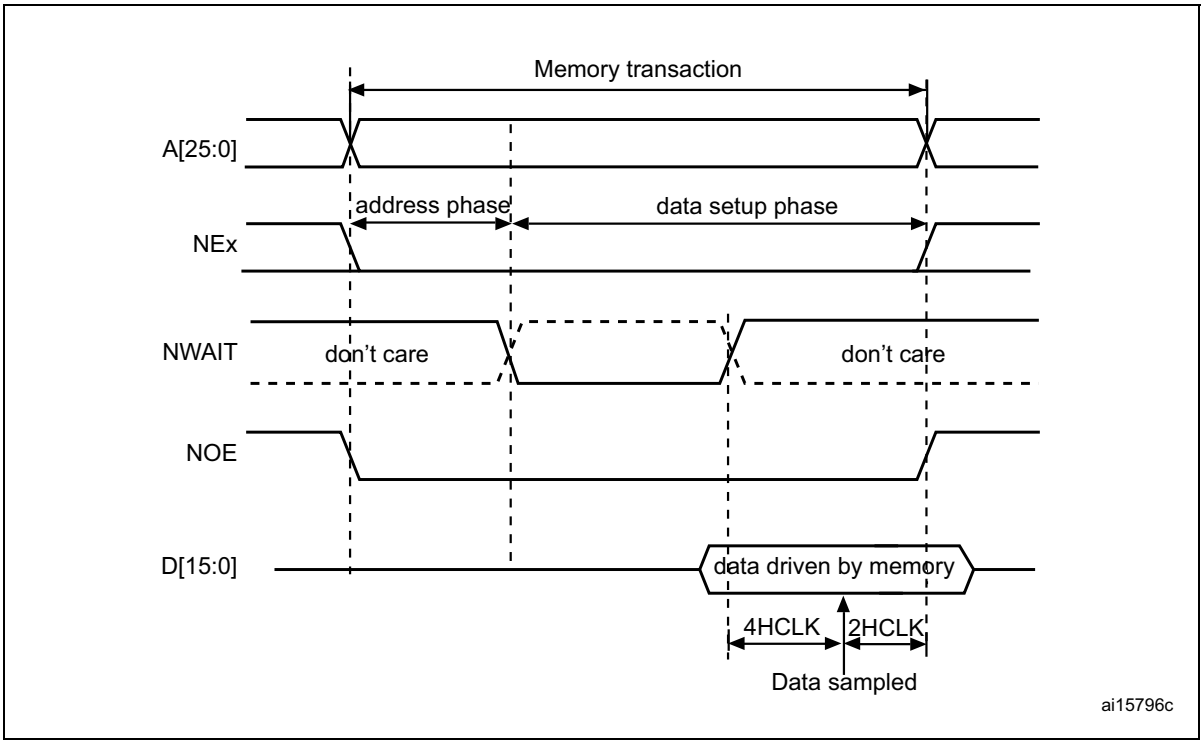
иначе

$$DATAST \geq 4 \times HCLK$$

где max_wait_assertion_time это максимальное время установки сигнала WAIT по низкому уровню на NEx/NOE/NWE.

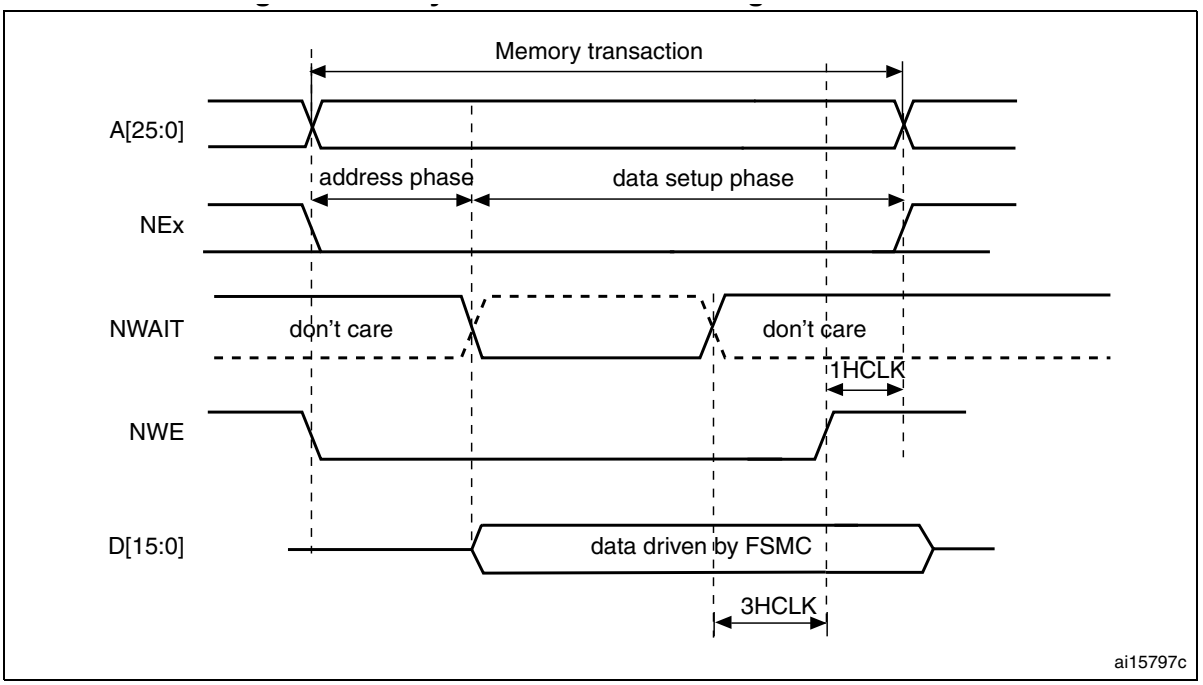
Рисунки ниже показывают число добавляемых тактов HCLK для доступа к памяти после снятия ею сигнала WAIT (независимо от приведённых случаев).

Асинхронное ожидание при чтении.



Полярность NWAIT зависит от бита WAITPOL в регистре FSMC_BCRx.

Асинхронное ожидание при записи.



Полярность NWAIT зависит от бита WAITPOL в регистре FSMC_BCRx.

21.5.5. Синхронные передачи

Кратность тактов памяти, CLK, и тактов HCLK задаётся параметром CLKDIV.

NOR Flash определяет минимальное время от выдачи NADV до переднего фронта CLK. Для его соблюдения FSMC не должен подавать такт на память во время первого цикла синхронного доступа (перед появлением NADV). Так гарантируется появление переднего фронта в середине импульса импульса NADV.

Задержки Данных и NOR Flash.

Задержка данных это число циклов ожидания перед считыванием данных. Значение DATLAT должно быть совместимым с значением в регистре конфигурации NOR Flash. При вычислении задержки данных FSMC не учитывает такты при низком NADV.

Внимание:

Некоторые устройства NOR Flash памяти не учитывают такты низкого NADV при подсчёте запаздывания данных. Так что точное соотношение запаздывания NOR Flash и параметром FMSC DATLAT может быть одним из двух:

- Запаздывание NOR Flash = (DATLAT + 2) тактов CLK
- Запаздывание NOR Flash = (DATLAT + 3) тактов CLK

Некоторые современные типы памяти выставляют NWAIT в фазе задержки. В таких случаях DATLAT может быть минимальным. В результате FSMC читает данные и ждёт достаточно долго для определения достоверности данных. Так FSMC определяет конец задержки памяти и достоверность данных.

Другие типы памяти во время запаздывания NWAIT не выставляют. В этом случае задержки и FSMC и памяти должны быть корректными, чтобы не терять данные в первой фазе доступа к памяти.

Однопакетная передача.

Если выбранный банк включён в пакетный синхронный доступ и, например, AHB запросил однопакетный доступ к 16-бит памяти, то FSMC выполняет пакетную передачу длиной 1 (при 16-бит передаче AHB) или длиной 2 (при 32-бит передаче AHB) и снимает выбор чипа после стробирования последних данных.

Понятно, что такая передача не самая эффективная по сравнению с асинхронным чтением по затратам тактов. Тем не менее, асинхронное чтение с произвольным доступом потребует установки режима доступа к памяти, что в итоге много длиннее.

Переход границы страниц в Cellular RAM 1.5

Cellular RAM 1.5 не разрешает пакетному доступу пересекать границу страниц. FSMC позволяет автоматически автоматически разбивать пакетный доступ при достижении границы страниц установкой битов `CPSIZE` в регистре `FSMC_BCR1` следуя размеру страницы.

Управление ожиданием

Для синхронной NOR Flash памяти состояние `NWAIT` определяется после заданного периода задержки, (DATLAT+2) тактов CLK.

Если `NWAIT` активен (низкий при `WAITPOL` = 0, высокий при `WAITPOL` = 1), то вставляются такты ожидания до снятия NWAIT (высокий при `WAITPOL` = 0, низкий при `WAITPOL` = 1).

Если `NWAIT` неактивен, данные считаются достоверными либо немедленно (бит `WAITCFG` = 1), либо по следующему фронту такта (бит `WAITCFG` = 0).

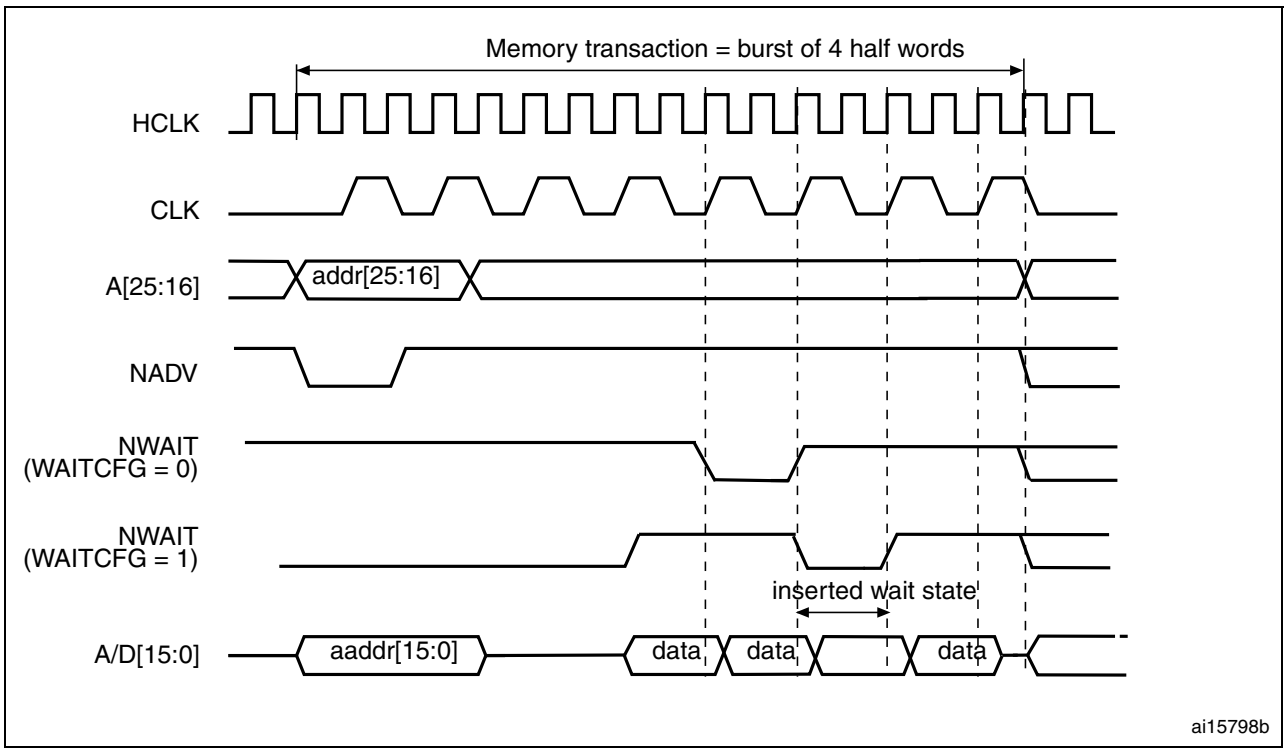
Во время вставки тактов ожидания сигналом `NWAIT` контроллер продолжает посылать такты памяти, сохраняет выбор чипа и разрешение вывода данных, но не рассматривает их достоверными.

Есть две конфигурации сигнала `NWAIT` для NOR Flash в пакетном режиме:

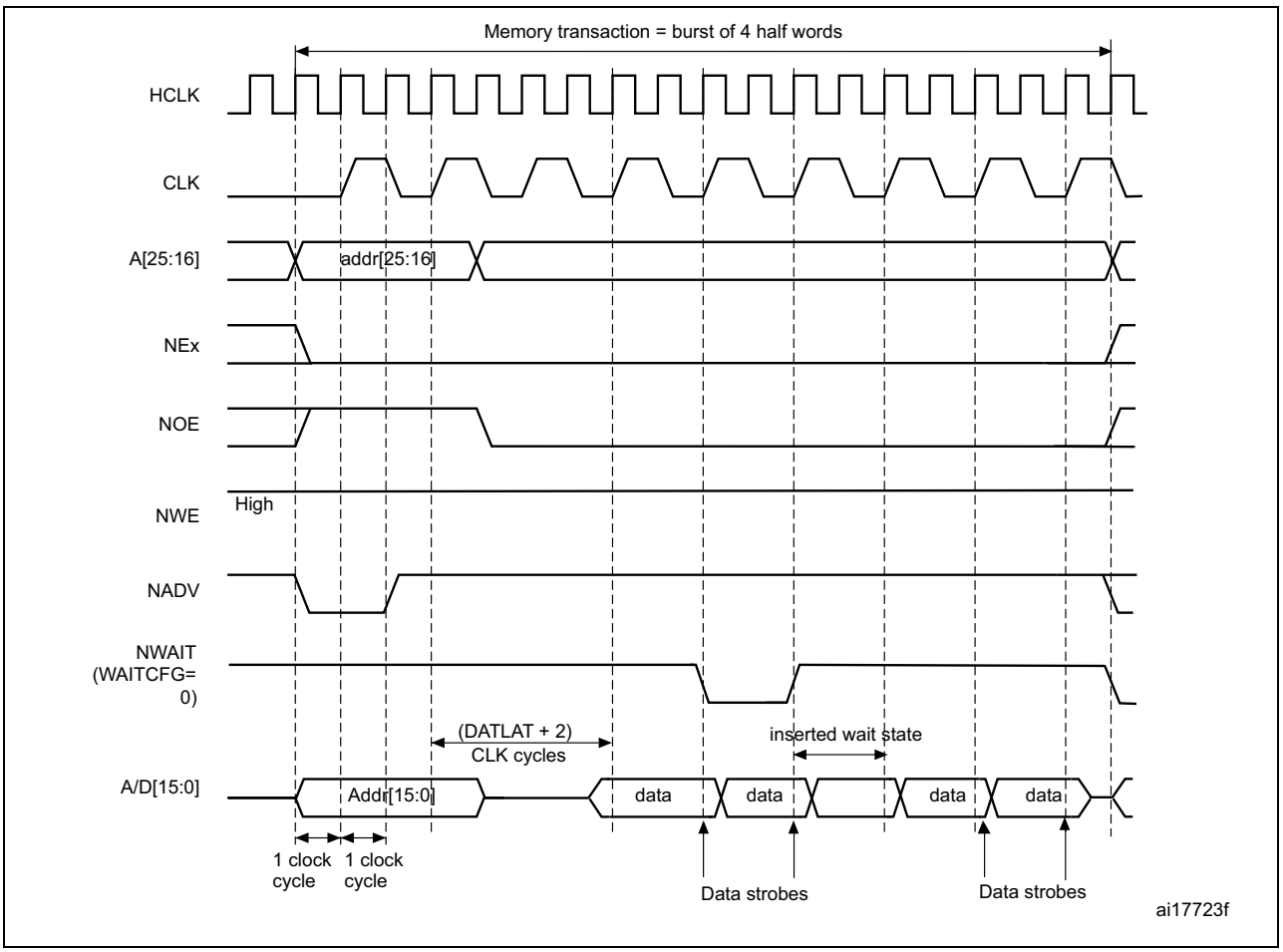
- Flash память выставляет сигнал `NWAIT` за один цикл до состояния ожидания (по умолчанию после сброса).
- Flash память выставляет сигнал `NWAIT` во время состояния ожидания.

Они поддерживаются FSMC раздельно для каждого выбора чипа битом `WAITCFG` в регистрах `FSMC_BCRx` (x = 0..3).

Конфигурации ожидания.



Синхронное мультиплексное чтение - NOR, PSRAM (CRAM).



1. Выходы байтовых линий BL не показаны; для NOR они высокие, для PSRAM (CRAM) низкие.
2. Полярность NWAIT ставится в 0.

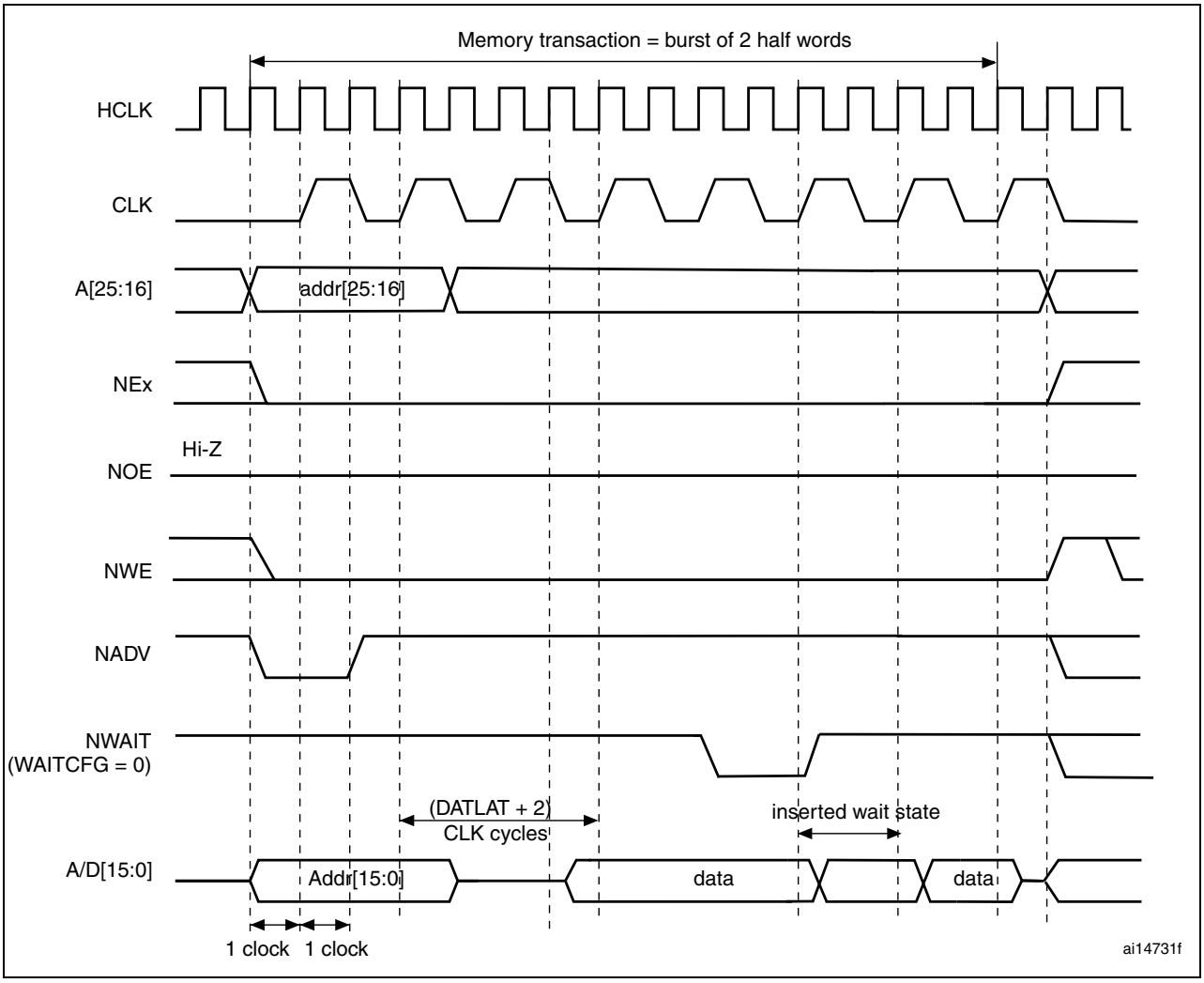
Таблица 125. Биты FSMC_BCRx.

Номер	Имя	Устанавливаемое значение
31-21	Резерв	0x000
19	CBURSTRW	При синхронном чтении не влияет
18:16	CPSIZE	Как нужно (0x1 для CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	1, если память поддерживает, иначе 0.
12	WREN	При синхронном чтении не влияет
11	WAITCFG	Соответственно памяти.
10	WRAPMOD	0x0
9	WAITPOL	Соответственно памяти.
8	BURSTEN	0x1
7	Резерв	0x1
6	FACCEN	Соответственно памяти (NOR Flash).
5-4	MWID	Как нужно
3-2	MTYP[0:1]	0x1 или 0x2
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 126. Биты FSMC_BTRx.

Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Задержка данных.
23-20	CLKDIV	0x0 для CLK = HCLK (не поддерживается) 0x1 для CLK = 2 × HCLK
19-16	BUSTURN	Не влияет.
15-8	DATAST	Всё равно.
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Всё равно.

Синхронная мультиплексная запись - PSRAM (CRAM).



1. Память должна выдавать NWAIT на такт раньше и WAITCFG должен быть равен 0.
2. Полярность NWAIT ставится в 0.
3. Байтовые линии (NBL) не показаны, при активном NEx удерживается низким.

Таблица 127. Биты FSMC_BCRx.

Номер	Имя	Устанавливаемое значение
31-20	Резерв	0x000
19	CBURSTRW	0x1
18:16	CPSIZE	Как нужно (0x1 для CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	1, если память поддерживает, иначе 0.
12	WREN	0x1
11	WAITCFG	0x0
10	WRAPMOD	0x0
9	WAITPOL	Соответственно памяти.
8	BURSTEN	На синхронную запись не влияет.

Номер	Имя	Устанавливаемое значение
7	Резерв	0x1
6	FACCEN	Соответственно памяти.
5-4	MWID	Как нужно
3-2	MTYP[0:1]	0x1
1	MUXEN	Как нужно
0	MBKEN	0x1

Таблица 128. Биты FSMC_BTRx.

Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Задержка данных.
23-20	CLKDIV	0x0 для CLK = HCLK (не поддерживается) 0x1 для CLK = 2 × HCLK
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Всё равно.
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Всё равно.

21.5.6. Регистры управления NOR/PSRAM

Они доступны словами.

Регистры выбора чипов SRAM/NOR-Flash 1..4 (FSMC_BCR1..4)

Смещение адреса: 0xA000 0000 + 8 * (x – 1), x = 1…4

По сбросу: 0x0000 30DB для Банка 1 и 0x0000 30D2 для Банков 2 - 4

Управляющая информация банков SRAM, PSRAM и NOR Flash памяти.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CBURSTRW	CPSIZE[2:0]			ASCYCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
												rw				rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

— Биты 31:20 Резерв, не трогать.

— Бит 19 **CBURSTRW:** Разрешение синхронной пакетной записи Cellular RAM (PSRAM).
Синхронное пакетное чтение разрешается битом BURSTEN в регистре FSMC_BCRx.
0: Запись асинхронная
1: Запись синхронная.

— Биты 18:16 **CPSIZE[2:0]:** Размер страницы CRAM.
Используется Cellular RAM 1.5, не позволяющей пересечения границы страниц пакетными передачами. При установке этих битов FSMC автоматически разбивает пакетные передачи при достижении границы страниц.

000: Без автоматического разделения пакетов (по умолчанию после сброса)

001: 128 байт

010: 256 байт

011: 512 байт

100: 1024 байта

Остальные: резерв.

— Бит 15 **ASYNCWAIT**: Разрешение сигнала ожидания при асинхронных передачах.

0: В асинхронном протоколе сигнал NWAIT не учитывается (по умолчанию после сброса)

1: Сигнал NWAIT учитывается

— Бит 14 **EXTMOD:** Разрешение расширенного режима.

Разрешает учитывать биты времянки записи немультиплексного асинхронного доступа в регистре FSMC_BWTR, устанавливая различные времянки чтения и записи.

0: Значения FSMC_BWTR не учитываются (по умолчанию после сброса)

1: Значения FSMC_BWTR учитываются

NB: При выключенном расширенном режиме FSMC может работать в Режимах 1 или 2:

– Режим 1 используется по умолчанию для SRAM/PSRAM памяти (MTYP [0:1]=0x0 or 0x01)

– Режим 2 используется по умолчанию для NOR памяти (MTYP [0:1]= 0x10).

— Бит 13 **WAITEN**: Разрешение ожидания.

Разрешает вставку тактов ожидания по сигналу NWAIT при синхронном доступе к Flash памяти.

0: Сигнал NWAIT не учитывается, такты ожидания после периода задержки Flash не вставляются

1: Сигнал NWAIT учитывается, вставляются такты ожидания после периода задержки Flash (по умолчанию после сброса)

— Бит 12 **WREN**: Разрешение записи FSMC в банк памяти.

0: Запись запрещена, выдаётся ошибка AHB,

1: Запись разрешена.

— Бит 11 **WAITCFG**: Конфигурация времянки ожидания.

Сигнал NWAIT показывает достоверность данных из памяти, или необходимость вставки состояний ожидания при доступе к Flash памяти в синхронном режиме. Этот бит определяет установку NWAIT за один такт до состояния ожидания или во время его:

0: Сигнал NWAIT ставится за один такт до состояния ожидания (по умолчанию после сброса),

1: Сигнал NWAIT Ставится во время состояния ожидания (не используется в PRAM).

— Бит 10 **WRAPMOD**: Поддержка закольцованных пакетов.

Определяет разбиение закольцованного пакета AHB на два последовательных обращения. Действует только при пакетном доступе.

0: Прямой закольцованный пакет запрещён (по умолчанию после сброса),

1: Прямой закольцованный пакет разрешён.

NB: Бит не имеет значения если CPU и DMA не могут выдавать такие запросы.

— Бит 9 **WAITPOL**: Полярность сигнала ожидания.

Действует только при пакетном доступе:

0: NWAIT активен низким (по умолчанию после сброса),

1: NWAIT активен высоким.

— Бит 8 **BURSTEN**: Разрешение пакетов.

Разрешает синхронный доступ при чтении в пакетном режиме:

0: Пакеты запрещены (по умолчанию после сброса). Чтение в асинхронном режиме.

1: Пакеты разрешены.Чтение в синхронном режиме.

NB: При выключенном расширенном режиме FSMC может работать в Режимах 1 или 2:

– Режим 1 используется по умолчанию для SRAM/PSRAM памяти (MTYP [0:1]=0x0 or 0x01)

– Режим 2 используется по умолчанию для NOR памяти (MTYP [0:1]= 0x10).

— Бит 7 Резерв, не трогать.

— Бит 6 **FACCEN:** Разрешение доступа к NOR Flash

0: Доступ к NOR Flash запрещён

1: Доступ к NOR Flash разрешён (по умолчанию после сброса).

— Биты 5:4 **MWID[1:0]**: Ширина шины данных памяти.

00: 8 бит,

01: 16 бит (по умолчанию после сброса),

10: Резерв, не трогать.

11: Резерв, не трогать.

— Биты 3:2 **MTYP[1:0]:** Тип памяти.
Тип внешней памяти в банке:
00: SRAM (по умолчанию после сброса для Банков 2...4)
01: PSRAM (CRAM)
10: NOR Flash(по умолчанию после сброса для Банка 1)
11: Резерв, не трогать.

— Бит 1 **MUXEN**: Разрешение мультиплекса Адреса/Данных.
Стоящий бит разрешает мультиплексирование Адреса/Данных на шине данных для NOR и PSRAM памяти:
0: Адрес/Данные не мультиплексируются
1: Адрес/Данные мультиплексируются (по умолчанию после сброса).

— Бит 0 **MBKEN**: Разрешение банка памяти.
После сброса разрешён Банк 1, остальные запрещены.
Доступ к запрещённому банку вызывает Ошибку на шине AHB.
0: Соответствующий банк выключен
1: Соответствующий банк включен.

Регистры времянки выбора чипов 1..4 SRAM/NOR-Flash (FSMC_BTR1..4).

Смещение адреса: 0xA000 0000 + 0x04 + 8 * (x – 1), x = 1..4

По сбросу: 0x0FFF FFFF

Биты FSMC_BTRx программно добавляют задержку в конец передач чтения/записи. Она позволяет согласовать минимальное время между последовательными передачами (t_{EHEL} от высокого NEx до низкого FSMC_NEx) и максимального времени освобождения шины данных памятью после операции чтения (t_{EHQZ}).

Содержат управляющую информацию для всех банков SRAM, PSRAM и NOR Flash. Если бит EXTMOD в регистре FSMC_BCRx стоит, то появляются 2 регистра: для чтения (этот регистр) и для записи (FSMC_BWTRx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать.

— Биты 29:28 **ACCMOD[1:0]:** Режим доступа
Определяет режим асинхронного доступа. Работает только при стоящем бите EXTMOD в регистре FSMC_BCRx.
00: Режим A
01: Режим B
10: Режим C
11: Режим D

— Биты 27:24 **DATLAT[3:0]:** Задержка данных для синхронной NOR Flash памяти
Определяет число тактов (+2) синхронной NOR Flash памяти, подаваемых на память перед чтением/записью первых данных.
Выражается не в периодах HCLK, а в периодах FSMC_CLK. В случае PSRAM (CRAM) это поле должно быть нулевым. В асинхронных NOR Flash, SRAM или PSRAM это поле никого не волнует.
0000: Задержка в 2 такта CLK перед первым доступом пакета
1111: Задержка в 17 тактов CLK перед первым доступом пакета (по умолчанию после сброса)

— Биты 23:20 **CLKDIV[3:0]:** Делитель частоты сигнала FSMC_CLK в тактах HCLK
0000: Резерв
0001: Период FSMC_CLK = 2 × HCLK периодов
0010: Период FSMC_CLK = 3 × HCLK периодов
1111: Период FSMC_CLK = 16 × HCLK периодов (по умолчанию после сброса)
В асинхронных NOR Flash, SRAM или PSRAM это поле никого вообще не волнует.

— Биты 19:16 **BUSTURN[3:0]:** Длительность фазы реверса шины

Эти биты программно добавляют задержку в конец передач чтение/запись и запись/чтение. Программируемая задержка вставляется между асинхронными чтением (мультиплекс или режим D) или записью и любым другим асинхронно/синхронным доступом к статическому банку (при чтении банк может быть тем же или другим; при записи банк может быть различным исключая мультиплекс и режим D).

В некоторых случаях, задержка реверса шины фиксирована при любом значении BUSTURN:

- Задержка реверса шины не вставляется между двумя последовательными записями в один банк статической памяти, исключая мультиплекс и режим D.
- Задержка реверса шины в 1 такт FSMC вставляется между:
  - Двумя последовательными асинхронными чтениями из одного банка статической памяти, исключая мультиплекс и режим D.
  - Асинхронным чтением и асинхронной или синхронной записью в любой статический или динамический банк, исключая мультиплекс и режим D.
  - Асинхронным чтением (режимы 1, 2, A, B or C) и чтением из другого статического банка.
- Задержка реверса шины в 2 такта FSMC вставляется между:
  - Двумя последовательными синхронными записями (пакетными или одиночными) в один банк
  - Синхронной записью (пакетной или одиночной) и асинхронным обменом с статическим банком памяти (банк может быть тем же или другим в случае чтения).
  - Двумя последовательными синхронными чтениями (пакетными или одиночными) с последующим синхронным/асинхронным обменом с другим банком статической памяти.
- Задержка реверса шины в 3 такта FSMC вставляется между:
  - Двумя последовательными синхронными записями (пакетными или одиночными) в разные статические банки.
  - Синхронной записью (пакетной или одиночной) и синхронным чтением из одного или разных банков.

  0000: Длительность фазы BUSTURN = 1 добавленный такт HCLK

  …

  1111: Длительность фазы BUSTURN = 16 × HCLK тактов (по умолчанию после сброса)

— Биты 15:8 **DATAST[7:0]:** Длительность фазы данных асинхронного доступа.

0000 0000: Резерв.

0000 0001: Длительность фазы DATAST = 2 × HCLK тактов

0000 0010: Длительность фазы DATAST = 3 × HCLK тактов

…

1111 1111: Длительность фазы DATAST = 256 × HCLK тактов (по умолчанию после сброса)

Пример: Режим 1, чтение, DATAST=1: Фаза данных = DATAST+3 = 4 HCLK тактов.

NB: При синхронном доступе никого не волнует.

— Биты 7:4 **ADDHLD[3:0]:** Длительность фазы удержания адреса.

Используется при мультиплексе и режиме D:

0000: Резерв

0001: Длительность фазы ADDHLD = 2 × HCLK тактов

0010: Длительность фазы ADDHLD = 3 × HCLK тактов

…

1111: Длительность фазы ADDHLD = 16 × HCLK тактов (по умолчанию после сброса)

Пример: Режим D, чтение, ADDHLD=1: Фаза удержания адреса = ADDHLD + 1 = 2 HCLK такта.

NB: При синхронном доступе не используется, фаза всегда равна 1 такту памяти.

— Биты 3:0 **ADDSET[3:0]:** Длительность фазы установки адреса.

Используется в SRAM, ROM и асинхронных NOR Flash и PSRAM:

0000: Длительность фазы ADDSET = 1 × HCLK тактов

…

1111: Длительность фазы ADDSET = × HCLK тактов (по умолчанию после сброса)

Пример: Режим 2, чтение, ADDSET=1: Фаза установки адреса = ADDSET + 1 = 2 HCLK тактов.

NB: При синхронном доступе к NOR Flash и PSRAM никому не интересно.

NB: PSRAM (CRAM) имеет переменную задержку из-за внутренней регенерации. Так что эта память выдаёт сигнал NWAIT на всё нужное время.

С памятью PSRAM (CRAM) поле DATLAT должно быть обнулено, так что FSMC вскоре выходит из задержки и смотрит на сигнал NWAIT от памяти memory, затем при готовности читает или пишет в память.

Этот метод можно использовать и с позднейшими поколениями Flash памяти, выдающими сигнал NWAIT, в отличие от старой Flash памяти.

Регистры времянки записи 1..4 SRAM/NOR-Flash (FSMC_BWTR1..4).

Смещение адреса: 0xA000 0000 + 0x104 + 8 * (x – 1), x = 1..4

По сбросу: 0x0FFF FFFF

Содержит управляющую информацию всех банков памяти SRAM, PSRAM и NOR Flash. Регистр работает только при асинхронной записи (стоит бит `EXTMOD` в регистре `FSMC_BCRx`).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		ACCMOD[2:0]		Reserved								BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
		rw	rw									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать.

— Биты 29:28 **ACCMOD[1:0]:** Режим доступа

Определяет режим асинхронного доступа. Работает только при стоящем бите EXTMOD в регистре FSMC_BCRx.

00: Режим A

01: Режим B

10: Режим C

11: Режим D

— Биты 27:20 Резерв, не трогать.

— Биты 19:16 **BUSTURN[3:0]:** Длительность фазы реверса шины

Программируемая задержка вставляется между асинхронной записью и любым другим асинхронно/синхронным доступом к статическому банку (при чтении банк может быть тем же или другим; при записи банк может быть различным, исключая мультиплекс и режим D).

В некоторых случаях, задержка реверса шины фиксирована при любом значении BUSTURN:

– Задержка реверса шины не вставляется между двумя последовательными записями в один банк статической памяти, исключая мультиплекс и режим D.

– Задержка реверса шины в 2 такта FSMC вставляется между:

  – Двумя последовательными синхронными записями (пакетными или одиночными) в один банк

  – Синхронной записью (пакетной или одиночной) и асинхронным обменом с статическим банком памяти (банк может быть тем же или другим в случае чтения).

– Задержка реверса шины в 3 такта FSMC вставляется между:

  – Двумя последовательными синхронными записями (пакетными или одиночными) в разные статические банки.

  – Синхронной записью (пакетной или одиночной) и синхронным чтением из одного или разных банков.

0000: Длительность фазы BUSTURN = 1 добавленный такт HCLK

…

1111: Длительность фазы BUSTURN = 16 × HCLK тактов (по умолчанию после сброса)

— Биты 15:8 **DATAST[7:0]:** Длительность фазы данных асинхронного доступа.

0000 0000: Резерв.

0000 0001: Длительность фазы DATAST = 2 × HCLK тактов

0000 0010: Длительность фазы DATAST = 3 × HCLK тактов

…

1111 1111: Длительность фазы DATAST = 256 × HCLK тактов (по умолчанию после сброса)

Пример: Режим 1, чтение, DATAST=1: Фаза данных = DATAST+3 = 4 HCLK тактов.

NB: При синхронном доступе никого не волнует.

— Биты 7:4 **ADDHLD[3:0]:** Длительность фазы удержания адреса.

Используется при мультиплексе и режиме D:

0000: Резерв

0001: Длительность фазы ADDHLD = 2 × HCLK тактов

0010: Длительность фазы ADDHLD = 3 × HCLK тактов

…

1111: Длительность фазы ADDHLD = 16 × HCLK тактов (по умолчанию после сброса)

Пример: Режим D, чтение, ADDHLD=1: Фаза удержания адреса = ADDHLD + 1 = 2 HCLK такта.

NB: При синхронном доступе не используется, фаза всегда равна 1 такту памяти.

— Биты 3:0 **ADDSET[3:0]:** Длительность фазы установки адреса.

Используется в SRAM, ROM и асинхронных NOR Flash и PSRAM:

0000: Длительность фазы ADDSET = 1 × HCLK тактов

…

1111: Длительность фазы ADDSET = × HCLK тактов (по умолчанию после сброса)

Пример: Режим 2, чтение, ADDSET=1: Фаза установки адреса = ADDSET + 1 = 2 HCLK тактов.

NB: При синхронном доступе к NOR Flash и PSRAM никому не интересно.

21.6. Контроллер NAND Flash/PC Card

FSMC выдаёт нужную времянку сигналов для следующих типов памяти:

- NAND Flash
  - 8-бит
  - 16-бит
- 16-bit PC Card совместимые устройства

Контроллер NAND/PC Card управляет тремя внешними банками. Банк 2 и банк 3 поддерживают NAND Flash устройства. Банк 4 поддерживает устройства PC Card.

Каждый банк конфигурируется своими регистрами. Программируемые параметры памяти включают времянку доступа и конфигурацию контроля ошибок (ECC).

Таблица 129. Параметры доступа NAND Flash/PC Card.

Параметр памяти	Функция	Режим доступа	Единицы	Min.	Max.
Время установки	Число тактов HCLK установки адреса перед подачей команды	Ч/З	HCLK	1	255
Ожидание	Минимальная длительность подачи команды	Ч/З	HCLK	2	256
Удержание	Число тактов удержания адреса (и данных при записи) после снятия команды	Ч/З	HCLK	1	254
Hi-Z шины данных	Число тактов удержания шиной данных high-Z состояния после начала записи	З	HCLK	0	255

21.6.1. Сигналы интерфейса внешней памяти

Внимание: При использовании PC Card или CompactFlash в режиме I/O, входная ножка NIOS16 должна быть заземлена, а не подключаться к карте, иначе FSMC забастует, откажется работать и пр.

Теоретически ограничений ёмкости нет, так как FSMC может использовать произвольное число тактов адреса.

NB: Префикс “N” указывает на активный низкий уровень сигнала.

Таблица 130. 8-бит NAND Flash

Сигнал FSMC	Вв./Выв.	Функция
A[17]	Ы	Разрешение защёлки адреса NAND Flash (ALE)
A[16]	Ы	Разрешение защёлки команды NAND Flash (CLE)
D[7:0]	В/Ы	8-бит мультиплексная, двунаправленная шина адреса/данных
NCE[x]	Ы	Выбор чипа, x = 2, 3
NOE(= NRE)	Ы	Разрешение выхода (в памяти: разрешение чтения, NRE)
NWE	Ы	Разрешение записи
NWAIT/INT[3:2]	В	Сигнал готов/занят NAND Flash для FSMC

Таблица 131. 16-бит NAND Flash

Сигнал FSMC	Вв./Выв.	Функция
A[17]	Ы	Разрешение защёлки адреса NAND Flash (ALE)
A[16]	Ы	Разрешение защёлки команды NAND Flash (CLE)
D[7:0]	В/Ы	16-бит мультиплексная, двунаправленная шина адреса/данных
NCE[x]	Ы	Выбор чипа, x = 2, 3
NOE(= NRE)	Ы	Разрешение выхода (в памяти: разрешение чтения, NRE)
NWE	Ы	Разрешение записи
NWAIT/INT[3:2]	В	Сигнал готов/занят NAND Flash для FSMC

Таблица 132. 16-бит PC Card

Сигнал FSMC	Вв./Выв.	Функция
A[10:0]	O	Address bus
NIOS16	I	Data transfer in I/O space. It must be shorted to GND (16-bit transfer only)
NIORD	O	Output enable for I/O space
NIOWR	O	Write enable for I/O space
NREG	O	Register signal indicating if access is in Common or Attribute space
D[15:0]	I/O	Bidirectional databus
NCE4_1	O	Chip select 1
NCE4_2	O	Chip select 2 (indicates if access is 16-bit or 8-bit)
NOE	O	Output enable in Common and in Attribute space
NWE	O	Write enable in Common and in Attribute space
NWAIT	I	PC Card wait input signal to the FSMC (memory signal name IORDY)
INTR	I	PC Card interrupt to the FSMC (only for PC Cards that can generate an interrupt)
CD	I	PC Card presence detection. Active high. If an access is performed to the PC Card banks while CD is low, an AHB error is generated.

21.6.2. Поддерживаемая память и передачи NAND Flash / PC Card

В таблице ниже неразрешённые передачи указаны красным цветом.

Уст-во	Режим	Ч/З	Размер данных AHB	Размер данных памяти	Можно	Комментарии
NAND 8-бит	Асинхр.	Ч	8	8	Д	
	Асинхр.	З	8	8	Д	
	Асинхр.	Ч	16	8	Д	За два обращения FSMC
	Асинхр.	З	16	8	Д	За два обращения FSMC
	Асинхр.	Ч	32	8	Д	За 4 обращения FSMC
	Асинхр.	З	32	8	Д	За 4 обращения FSMC
NAND 16-бит	Асинхр.	Ч	8	16	Д	
	Асинхр.	З	8	16	Н	
	Асинхр.	Ч	16	16	Д	
	Асинхр.	З	16	16	Д	
	Асинхр.	Ч	32	16	Д	За два обращения FSMC
	Асинхр.	З	32	16	Д	За два обращения FSMC

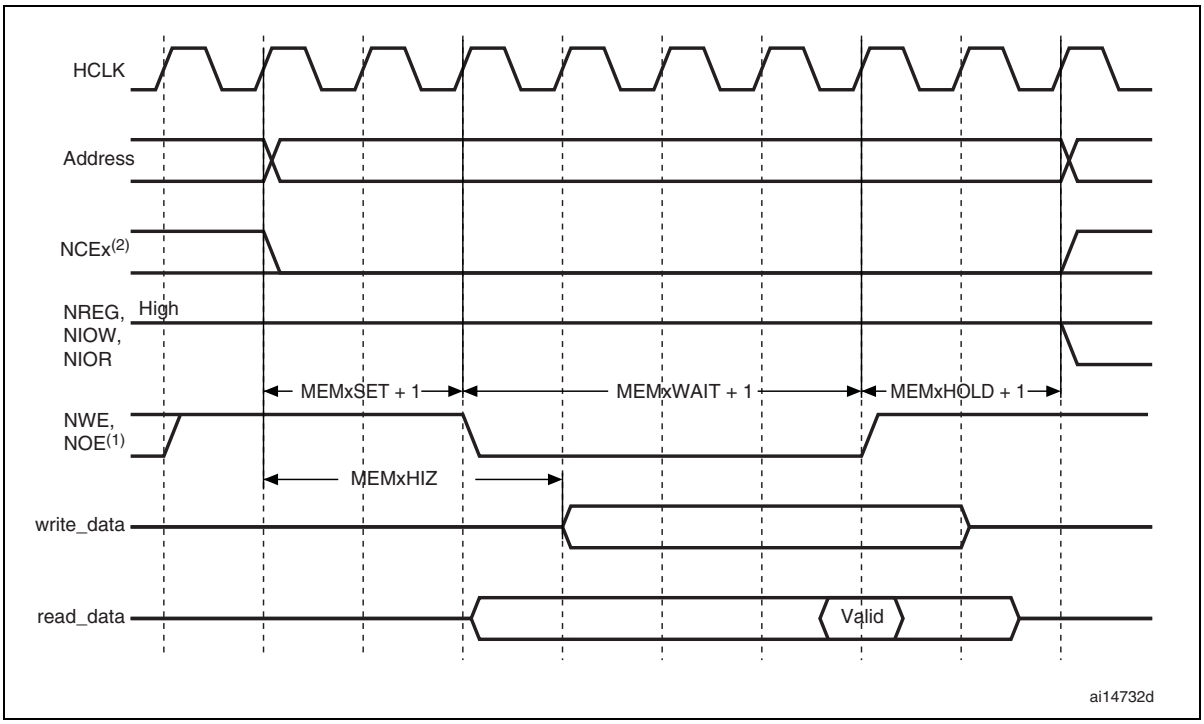
21.6.3. Временные диаграммы NAND Flash / PC Card

Банки памяти PC Card/CompactFlash и NAND Flash управляются регистрами:

- Управления: `FSMC_PCRx`
- Состояния прерываний: `FSMC_SRx`
- Коррекции ошибок (ECC): `FSMC_ECCRx`
- Времянки Общей памяти: `FSMC_PMEMx`
- Времянки памяти Атрибутов: `FSMC_PATTx`
- Времянки памяти Вв./Выв.: `FSMC_PIOx`

Регистры времянки содержат три параметра, определяющих число тактов HCLK для трёх фаз доступа к PC Card/CompactFlash NAND Flash, плюс ещё один, задающий время начала управления шиной данных при записи. Ниже показаны параметры доступа к Общей памяти, так как для памяти Атрибутов и Вв./Выв. (только для PC Card) они такие же.

Времянки доступа к Общей памяти NAND/PC Card.



1. NOE остаётся высоким (неактивным) при записи. NWE остаётся высоким (неактивным) при чтении.
2. NCEx ставится низким при запросе доступа к NAND и держится до обращения к другому банку.

21.6.4. Операции NAND Flash

Сигналы разрешения защёлок команды (CLE) и адреса (ALE) у NAND управляются сигналами адреса FSMC. То есть, CPU пишет команды или адреса в определённые области NAND Flash памяти.

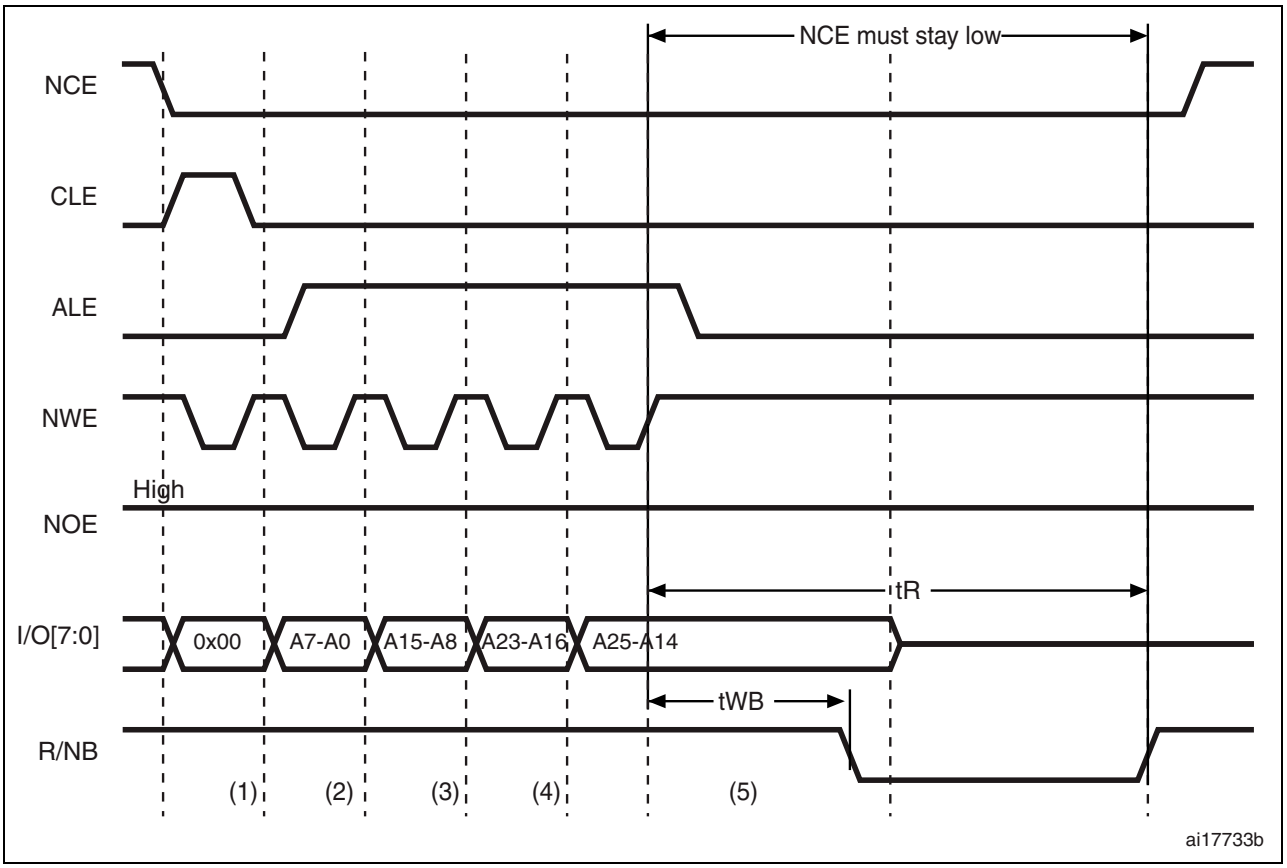
Вот типичное чтение из NAND Flash:

1. Регистрами FSMC_PCRx и FSMC_PMEMx (иногда и FSMC_PATTx) выбрать банк NAND Flash в соответствии с его характеристиками (битами PWID ширину шины данных, и как нужно PTYP = 1, PWAITEN = 0 или 1).
2. CPU пишет один байт команды в общую память (например 0x00 для Samsung NAND Flash). Во время строба записи (низкий импульс на NWE) сигнал на входе CLE активен, и байт рассматривается как команда. Теперь при последующих чтениях страницы команду подавать не нужно.
3. Теперь CPU может послать адрес начала чтения (STARTAD) записью байтов (например, STARTAD[7:0], STARTAD[15:8], STARTAD[23:16] и, наконец, STARTAD[25:24] для 64 Mb x 8 бит NAND Flash) в общую память или пространство атрибутов. Во время строба записи (низкий импульс на NWE) сигнал на входе ALE активен, и байты рассматриваются как начальный адрес чтения. Использование памяти атрибутов позволяет FSMC включать разные времянки вроде предожидания в некоторых NAND Flash.
4. Контроллер ждёт готовности NAND Flash (высокий сигнал R/NB) перед новым обращением (в тот же или другой банк). При этом контроллер держит сигнал NCE активным (низким).
5. Теперь CPU может байт за байтом читать из общей памяти страницу NAND Flash (поле данных + резервное поле).
6. Следующую страницу NAND Flash можно читать без записи команд и адреса тремя способами:
   - просто выполняя операцию шага 5
   - писать новый адрес, начиная с шага 3
   - послать новую команду, начиная с шага 2.

21.6.5. Предожидание NAND Flash

Некоторые устройства NAND Flash требуют чтобы после записи последней части адреса контроллер ждал падения сигнала R/NB.

Доступ к не ‘CE не волнует’ NAND-Flash.



1. CPU записал байт 0x00 по адресу 0x7001 0000.
2. CPU записал байт A7-A0 по адресу 0x7002 0000.
3. CPU записал байт A15-A8 по адресу 0x7002 0000.
4. CPU записал байт A23-A16 по адресу 0x7002 0000.
5. CPU записал байт A25-A24 по адресу 0x7802 0000: FSMC выполняет запись используя определение FSMC_PATT2, где ATTHOLD ≥ 7 (обеспечивая (7+1) × HCLK = 112 ns > t_{WB} max). так гарантируется что NCE остаётся низким вплоть до падения и подъёма R/NB (требуется для NAND Flash, безразличным к NCE).

При надобности это достигается установкой MEMHOLD для соответствия t_{WB}. Но у чтения NAND Flash задержка удержания равна (MEMHOLD + 2) x HCLK тактов, а запись имеет задержку удержания (MEMHOLD) x HCLK тактов.

Чтобы обойти это можно использовать память атрибутов установив ATTHOLD для получения времянки t_{WB}, и оставив MEMHOLD минимальной. Тогда при всех обменах с NAND Flash надо использовать общую память, но последний байт адреса нужно писать а память атрибутов.

21.6.6. Вычисление кода коррекции ошибки (ECC) NAND Flash

Контроллер FSMC PC-Card имеет два блока аппаратного вычисления коррекции ошибок, по одному на блоки памяти 2 и 3. С блоком 4 аппаратное вычисление ECC невозможно. Оба регистра идентичны.

Использованный алгоритм позволяет исправлять ошибку в 1 бите и обнаруживать ошибки 2 битов в 256, 512, 1 024, 2 048, 4 096 и 8 192 передачах. Он основан на кодах Хэмминга с вычислением чётности строк и колонок.

Модули ECC отслеживают шину данных и сигналы NCE и NWE активного банка памяти NAND Flash.

Они работают так:

- При доступе к банкам 2 и 3 NAND Flash перехваченные данные на шине D[15:0] запоминаются для последующих вычислений.
- При доступе по любым другим адресам логика ECC отдыхает, так что команды и сами адреса в вычислениях не участвуют.

После передачи желаемого числа байт из регистров FSMC_ECCR2/3 читается результат вычисления и они сбрасываются обнулением бита ECCEN. Для вычисления нового блока бит ECCEN в регистре FSMC_PCR2/3 нужно поставить.

Вычисляем ECC:

1. Ставим бит ECCEN в регистре FSMC_PCR2/3.
2. Пишем данные в страницу памяти NAND Flash. Блок ECC в это время вычисляет ECC.
3. Сохраняем значение ECC из FSMC_ECCR2/3 в переменной.
4. Снимаем и снова ставим бит ECCEN в регистре FSMC_PCR2/3 для обратного чтения данных. Блок ECC в это время вычисляет ECC.
5. Читаем новое значение ECC из регистра FSMC_ECCR2/3.
6. Если оба значения ECC равны, то можно спокойно вздохнуть, иначе программа исправления ошибок скажет вам, может ли она это сделать.

21.6.7. Операции PC Card/CompactFlash

Адресные пространства и доступ к памяти

FSMC поддерживает Compact Flash и PC Cards в режимах Памяти и I/O (но не True IDE).

Compact Flash и PC Card имеют 3 пространства памяти:

- Общее
- Атрибутов
- Память I/O

Ножки nCE2 и nCE1 (FSMC_NCE4_2 и FSMC_NCE4_1) выбирают карту и словный или байтовый доступ: nCE2 выбирает нечётный байт на D15-8 и nCE1 выбирает чётный байт на D7-0 при A0=0 или нечётный байт на D7-0 при A0=1. Полное слово на D15-0 выбирается при обоих низких nCE2 и nCE1.

Пространство памяти выбирается низким nOE при чтении или низким nWE при записи, в сочетании с низкими nCE2/nCE1 и nREG.

- Если при доступе к памяти nREG=1, то используется общая память
- Если при доступе к памяти nREG=0, то используется память атрибутов

Пространство I/O выбирается низким nIORD при чтении или низким nIOWR при записи [вместо nOE/nWE для пространства памяти], в сочетании с nCE2/nCE1. При этом nREG должен быть низким.

Для 16-бит PC Card есть три вида доступа:

- К общей памяти: 8-бит по чётным адресам или 16-бит AHB.
  8-бит доступ по нечётным адресам не поддерживается и не ведёт к выставлению низкого nCE2.
  32-бит запросы AHB транслируются в два 16-бит обращения к памяти.
- К памяти атрибутов с конфигурацией PC Card по 8-бит запросу AHB по чётным адресам.
  16-бит доступ AHB преобразуется в одно 8-бит обращение к памяти: nCE1 будет низким, а nCE2 высоким и достоверным будет только байт на D7-0. 32-бит доступ AHB преобразуется в два 8-bit обращения к памяти по чётным адресам: nCE1 будет низким, а nCE2 будет высоким и достоверным будет только чётный байт.
- Доступ к пространству I/O выполняется 8- или 16-бит запросами AHB.

Таблица 134. Сигналы и типы доступа 16-бит PC Card.

nCE2	nCE1	nREG	nOE/nWE	nIORD /nIOWR	A10	A9	A7-1	A0	Space	Access Type	Allowed/not Allowed
1	0	1	0	1	X	X	X-X	X	Common Memory Space	Read/Write byte on D7-D0	YES
0	1	1	0	1	X	X	X-X	X		Read/Write byte on D15-D8	Not supported
0	0	1	0	1	X	X	X-X	0		Read/Write word on D15-D0	YES
X	0	0	0	1	0	1	X-X	0	Attribute Space	Read or Write Configuration Registers	YES
X	0	0	0	1	0	0	X-X	0		Read or Write CIS (Card Information Structure)	YES
1	0	0	0	1	X	X	X-X	1	Attribute Space	Invalid Read or Write (odd address)	YES
0	1	0	0	1	X	X	X-X	x		Invalid Read or Write (odd address)	YES
1	0	0	1	0	X	X	X-X	0	I/O space	Read Even Byte on D7-0	YES
1	0	0	1	0	X	X	X-X	1		Read Odd Byte on D7-0	YES
1	0	0	1	0	X	X	X-X	0		Write Even Byte on D7-0	YES
1	0	0	1	0	X	X	X-X	1		Write Odd Byte on D7-0	YES
0	0	0	1	0	X	X	X-X	0		Read Word on D15-0	YES
0	0	0	1	0	X	X	X-X	0		Write word on D15-0	YES
0	1	0	1	0	X	X	X-X	X		Read Odd Byte on D15-8	Not supported
0	1	0	1	0	X	X	X-X	X		Write Odd Byte on D15-8	Not supported

Банк 4 FSMC позволяет доступ к этим 3 пространствам. См. *Секцию 21.4.2* и *Таблицу 102*.

Ожидание

Если ожидание разрешено битом PWAITEN в регистре FSMC_PCRx, то CompactFlash и PC Card могут запросить у FSMC расширение фазы доступа, указанное битами у MEMWAITx/ATTWAITx/IOWAITx, выставляя сигнал nWAIT после активации nOE/nWE или nIORD/nIOWR. Для правильного определения nWAIT биты MEMWAITx/ATTWAITx/IOWAITx нужно определять так:

xxWAITx >= 4 + max_wait_assertion_time/HCLK

где max_wait_assertion_time максимальное время падения nWAIT после падения nOE/nWE или nIORD/nIOWR.

После снятия nWAIT FSMC Расширяет фазу ожидания на 4 такта HCLK.

21.6.8. Управляющие регистры NAND Flash/PC Card

Они доступны словами (32 бита).

Регистры управления 2..4 NAND Flash/PC Card (FSMC_PCR2..4).

Смещение адреса: 0xA0000000 + 0x40 + 0x20 * (x – 1), x = 2..4

По сбросу: 0x0000 0018

31–20	19–17	16–13	12–9	8–7	6	5–4	3	2	1	0
Reserved	ECCPS[2:0]	TAR[2:0]	TCLR[2:0]	Res.	ECCEN	PWID[1:0]	PTYP	PBKEN	PWAITEN	Reserved
	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

— Биты 31:20 Резерв, не трогать.

— Биты 19:17 **ECCPS[2:0]:** Размер страницы расширенного ECC.

000: 256 байт
001: 512 байт
010: 1024 байта
011: 2048 байт
100: 4096 байт
101: 8192 байта

— Биты 16:13 **TAR[2:0]:** Задержка от ALE до RE в тактах AHB (HCLK).

Время: t_ar = (TAR + SET + 4) × THCLK где THCLK это период такта HCLK

0000: 1 так HCLK (по умолчанию)
1111: 16 тактов HCLK

NB: SET это MEMSET или ATTSET в зависимости от пространства адресов.

— Биты 12:9 **TCLR[2:0]:** Задержка от CLE до RE в тактах AHB (HCLK).

Время: t_clr = (TCLR + SET + 4) × THCLK где THCLK это период такта HCLK

0000: 1 так HCLK (по умолчанию)
1111: 16 тактов HCLK

NB: SET это MEMSET или ATTSET в зависимости от пространства адресов.

— Биты 8:7 Резерв, не трогать.

— Бит 6 **ECCEN**: Разрешение работы ECC

0: Логика ECC отключена и сброшена (по умолчанию после сброса),
1: Логика ECC включена.

— Биты 5:4 **PWID[1:0]:** Ширина шины данных внешней памяти.

00: 8 бит
01: 16 бит (по умолчанию после сброса). Для PC Card обязательно.
10: Резерв, не трогать.
11: Резерв, не трогать.

— Бит 3 **PTYP**: Тип памяти банка.

0: PC Card, CompactFlash, CF+ или PCMCIA
1: NAND Flash (по умолчанию после сброса)

— Бит 2 **PBKEN**: Включение банка памяти PC Card/NAND Flash.

Обращение к отключённому банку вызывает ERROR на шине AHB.

0: Выключен (по умолчанию после сброса)
1: Включён

— Бит 1 **PWAITEN**: Разрешение ожидания.

0: Нельзя
1: Можно

NB: Для PC Card биты MEMWAITx/ATTWAITx/IOWAITx должны ставиться так:

xxWAITx ≥ 4 + max_wait_assertion_time/HCLK

где max_wait_assertion_time это максимальное время между падением nWAIT и падением nOE/nWE или nIORD/nIOWR.

— Бит 0 Резерв, не трогать.

Регистр 2..4 состояния и прерываний FIFO (FSMC_SR2..4)

Смещение адреса: 0xA000 0000 + 0x44 + 0x20 * (x-1), x = 2..4

По сбросу: 0x0000 0040

FSMC использует FIFO при записи в память для хранения до 16 слов данных от AHB. При этом AHB быстро освобождается для передач другой периферии пока FSMC сливает FIFO в память. При вычислении ECC надо использовать один бит состояния FIFO.

ECC вычисляется при записи в память, так что достоверное значение получается при пустом FIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS
																									r	rw	rw	rw	rw	rw	rw

— Биты 31:20 Резерв, не трогать.

— Бит 6 **FEMPT:** FIFO пуст.

Только чтение.

0: Что-то есть

1: FIFO пуст

— Бит 5 **IFEN:** Разрешение определения заднего фронта прерывания

0: Нельзя

1: Можно

— Бит 4 **ILEN:** Разрешение определения высокого уровня прерывания

0: Нельзя

1: Можно

— Бит 3 **IREN:** Разрешение определения переднего фронта прерывания

0: Нельзя

1: Можно

— Бит 2 **IFS:** Флаг заднего фронта прерывания

Ставится аппаратно, снимается программно.

0: Не было

1: Случилось

NB: Ставится программно (ЛАЖА КАКАЯ-ТО, но там так и написано). Может снимается записью 1?

— Бит 1 **ILS:** Флаг высокого уровня прерывания

Ставится аппаратно, снимается программно.

0: Не было

1: Случилось

— Бит 0 **IRS:** Флаг переднего фронта прерывания

Ставится аппаратно, снимается программно.

0: Не было

1: Случилось

NB: Ставится программно (ОПЯТЬ ЛАЖА, но там ИМЕННО ТАК). Может снимается записью 1?

Регистр 2..4 времянки общей памяти (FSMC_PMEM2..4)

Смещение адреса: 0xA000 0000 + 0x48 + 0x20 * (x – 1), x = 2..4

По сбросу: 0xFCFC FCFC

Каждый регистр чтения/записи `FSMC_PMEMx` (x = 2..4) содержит определение времянки банка X PC Card или NAND Flash, используемой при доступе к общей памяти 16-бит PC Card/CompactFlash, или при записи команд и адреса NAND Flash и при чтении/записи данных.

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

— Биты 31:24 **MEMHIZx[7:0]:** Время состояния HiZ шины данных общей памяти X.
Число тактов HCLK (+1 только для NAND) удержания состояния HiZ на шине данных после старта записи в общее пространство (сокет X) PC Card/NAND Flash. Работает только при записи:
0000 0000: 1 такт HCLK
1111 1110: 255 тактов HCLK
1111 1111: Резерв

— Биты 23:16 **MEMHOLDx[7:0]**: Время удержания общей памяти X.
При чтении общей памяти NAND Flash определяют число тактов (HCLK+2) удержания адреса после снятия команды (NWE, NOE).
При записи общей памяти NAND Flash определяют число тактов (HCLK) удержания данных после снятия команды (NWE, NOE).
0000 0000: Резерв.
0000 0001: 1 такт HCLK при записи, 3 такта HCLK при чтении.
1111 1110: 254 такта HCLK при записи, 256 тактов HCLK при чтении.
1111 1111: Резерв.

— Биты 15:9 **MEMWAITx[7:0]:** Время ожидания общей памяти X.
При чтении и записи общей памяти PC Card/NAND Flash определяют минимальное число тактов HCLK (+1) выдачи команд (NWE, NOE) для сокета X. Длительность команд может расширяться сигналом ожидания (NWAIT) после заданного значения HCLK:
0000 0000: Резерв
0000 0001: 2 такта HCLK (+ такт ожидания, вставленный снятием NWAIT)
1111 1110: 255 тактов HCLK (+ такт ожидания, вставленный снятием NWAIT)
1111 1111: Резерв.

— Биты 8:0 **MEMSETx[7:0]:** Время установки общей памяти X.
При чтении и записи общей памяти PC Card/NAND Flash определяют число тактов HCLK (+1 для PC Card, +2 для NAND) установки адреса перед подачей команд (NWE, NOE) для сокета X:
0000 0000: 1 такт HCLK
1111 1110: 255 тактов HCLK
1111 1111: Резерв

Регистр 2..4 времянки памяти атрибутов (FSMC_PATT2..4)

Смещение адреса: 0xA000 0000 + 0x4C + 0x20 * (x – 1), x = 2..4

По сбросу: 0xFCFC FCFC

Каждый регистр чтения/записи `FSMC_PATTx` (x = 2..4) содержит определение времянки банка X PC Card/CompactFlash или NAND Flash, используемой при 8-бит доступе к памяти атрибутов PC Card/CompactFlash, или при записи последней части адреса NAND Flash, если при этом времянка должна отличаться от предыдущих стадий.

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

— Биты 31:24 **ATTHIZx[7:0]:** Время состояния HiZ шины данных памяти атрибутов X.
Число тактов HCLK (+1 только для NAND) удержания состояния HiZ на шине данных после старта записи в пространство атрибутов (сокет X) PC Card/NAND Flash. Работает только при записи:
0000 0000: 0 тактов HCLK
1111 1110: 255 тактов HCLK
1111 1111: Резерв

— Биты 23:16 **ATTHOLDx[7:0]**: Время удержания памяти атрибутов X.
При чтении памяти атрибутов PC Card/NAND Flash определяют число тактов (HCLK+2) удержания адреса после снятия команды (NWE, NOE).
При записи памяти атрибутов PC Card/NAND Flash определяют число тактов (HCLK) удержания данных после снятия команды (NWE, NOE).
0000 0000: Резерв.
0000 0001: 1 такт HCLK при записи, 3 такта HCLK при чтении.
1111 1110: 254 такта HCLK при записи, 256 тактов HCLK при чтении.
1111 1111: Резерв.

— Биты 15:9 **ATTWAITx[7:0]:** Время ожидания памяти атрибутов X.

При чтении и записи памяти атрибутов PC Card/NAND Flash определяют минимальное число тактов HCLK (+1) выдачи команд (NWE, NOE) для сокета X. Длительность команд может расширяться сигналом ожидания (NWAIT) после заданного значения HCLK:

0000 0000: Резерв

0000 0001: 2 такта HCLK (+ такт ожидания, вставленный снятием NWAIT)

1111 1110: 255 тактов HCLK (+ такт ожидания, вставленный снятием NWAIT)

1111 1111: Резерв.

— Биты 8:0 **ATTSETx[7:0]:** Время установки памяти атрибутов X.

При чтении и записи памяти атрибутов PC Card/NAND Flash определяют число тактов HCLK (+1) установки адреса перед подачей команд (NWE, NOE) для сокета X:

0000 0000: 1 такт HCLK

1111 1110: 255 тактов HCLK

1111 1111: Резерв

Регистр 4 времянки пространства Вв./Выв. (FSMC_PIO4)

Смещение адреса: 0xA000 0000 + 0xB0

По сбросу: 0xFCFCFCFC

Регистр чтения/записи `FSMC_PIO4` содержит данные времянки доступа к пространству I/O 16-бит PC Card/CompactFlash.

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

— Биты 31:24 **IOHIZx[7:0]:** Время состояния HiZ шины данных I/O X.

Число тактов HCLK удержания состояния HiZ на шине данных после старта записи в пространство I/O (сокет X) PC Card. Работает только при записи:

0000 0000: 0 тактов HCLK

1111 1111: 255 тактов HCLK (по умолчанию после сброса)

— Биты 23:16 **IOHOLDx[7:0]**: Время удержания I/O X.

При чтении/записи памяти I/O PC Card определяют число тактов (HCLK+2) удержания адреса (и данных при записи) после снятия команды (NWE, NOE).

0000 0000: Резерв.

0000 0001: 1 такт HCLK.

1111 1111: 255 тактов HCLK (по умолчанию после сброса).

— Биты 15:9 **IOWAITx[7:0]:** Время ожидания I/O X.

При чтении и записи памяти I/O PC Card определяют минимальное число тактов HCLK (+1) выдачи команд (SMNWE, SMNOE) для сокета X. Длительность команд может расширяться сигналом ожидания (NWAIT) после заданного значения HCLK:

0000 0000: Резерв

0000 0001: 2 такта HCLK (+ такт ожидания, вставленный снятием NWAIT)

1111 1111: 255 тактов HCLK (+ такт ожидания, вставленный Картой снятием NWAIT)

— Биты 8:0 **IOSETx[7:0]:** Время установки I/O X.

При чтении и записи памяти I/O PC Card определяют число тактов HCLK (+1) установки адреса перед подачей команд (NWE, NOE) для сокета X:

0000 0000: 1 такт HCLK

1111 1111: 256 тактов HCLK (по умолчанию после сброса)

Регистры 2/3 результата ECC (FSMC_ECCR2/3)

Смещение адреса: 0xA000 0000 + 0x54 + 0x20 * (x – 1), x = 2 или 3

По сбросу: 0x0000 0000

Они содержат текущий код коррекции ошибки от модулей ECC. При чтении/записи данных из страницы NAND Flash по нормальному адресу они автоматически обрабатываются модулем этого банку. По концу чтения X байтов (в соответствии с полем ECCPS в регистре FSMC_PCRx) нужно прочитать число из регистра FSMC_ECCx и сравнить его с ранее сохранённым числом для тех же данных чтобы узнать о наличии ошибки и возможном её исправлении. Читать регистры FSMC_ECCRx нужно перед их очисткой снятием бита ECCEN. Для вычисления нового блока нужно установить бит ECCEN снова.

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

— Биты 31:24 **ECCx[31:0]**: Результат ECC

Таблица 135. Релевантные биты результата ECC.

ECCPS[2:0]	Размер страницы в байтах	Биты ECC
000	256	ECC[21:0]
001	512	ECC[23:0]
010	1024	ECC[25:0]
011	2048	ECC[27:0]
100	4096	ECC[29:0]
101	8192	ECC[31:0]

21.6.9. Карта регистров FSMC

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0000	FSMC_BCR1	Reserved												CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID[1:0]		MTYP[0:1]		MUXEN	MBKEN
0008	FSMC_BCR2	Reserved												CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID[1:0]		MTYP[0:1]		MUXEN	MBKEN
0010	FSMC_BCR3	Reserved												CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID[1:0]		MTYP[0:1]		MUXEN	MBKEN
0018	FSMC_BCR4	Reserved												CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID[1:0]		MTYP[0:1]		MUXEN	MBKEN
0004	FSMC_BTR1	Res.		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
000C	FSMC_BTR2	Res.		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
0014	FSMC_BTR3	Res.		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
001C	FSMC_BTR4	Res.		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
0104	FSMC_BWTR 1	Res.		ACC MOD [1:0]		Res.								BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
010C	FSMC_BWTR 2	Res.		ACC MOD [1:0]		Res.								BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			

Продолжение…

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0114	FSMC_BWTR 3	Res.		ACC MOD [1:0]		Res.								BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
011C	FSMC_BWTR 4	Res.		ACC MOD [1:0]		Res.								BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
0xA000 0060	FSMC_PCR2	Reserved												ECCPS[2:0]				TAR[2:0]				TCLR[2:0]			Res.	ECCEN		PWID[1:0]		PTYP	PBKEN	PWAITEN	Reserved
0xA000 0080	FSMC_PCR3	Reserved												ECCPS[2:0]				TAR[2:0]				TCLR[2:0]			Res.	ECCEN		PWID[1:0]		PTYP	PBKEN	PWAITEN	Reserved
0xA000 00A0	FSMC_PCR4	Reserved												ECCPS[2:0]				TAR[2:0]				TCLR[2:0]			Res.	ECCEN		PWID[1:0]		PTYP	PBKEN	PWAITEN	Reserved
0xA000 0064	FSMC_SR2	Reserved																								FEMPT		IFEN	ILEN	IREN	IFS	ILS	IRS
0xA000 0084	FSMC_SR3	Reserved																								FEMPT		IFEN	ILEN	IREN	IFS	ILS	IRS
0xA000 00A4	FSMC_SR4	Reserved																								FEMPT		IFEN	ILEN	IREN	IFS	ILS	IRS
0xA000 0068	FSMC_PMEM 2	MEMHIZ[7:0]								MEMHOLD[7:0]								MEMWAIT[7:0]								MEMSET[7:0]							
0xA000 0088	FSMC_PMEM 3	MEMHIZ[7:0]								MEMHOLD[7:0]								MEMWAIT[7:0]								MEMSET[7:0]							
0xA000 00A8	FSMC_PMEM 4	MEMHIZ[7:0]								MEMHOLD[7:0]								MEMWAIT[7:0]								MEMSET[7:0]							
0xA000 006C	FSMC_PATT2	ATTHIZ[7:0]								ATTHOLD[7:0]								ATTWAIT[7:0]								ATTSET[7:0]							
0xA000 008C	FSMC_PATT3	ATTHIZ[7:0]								ATTHOLD[7:0]								ATTWAIT[7:0]								ATTSET[7:0]							
0xA000 00AC	FSMC_PATT4	ATTHIZ[7:0]								ATTHOLD[7:0]								ATTWAIT[7:0]								ATTSET[7:0]							
0xA000 00B0	FSMC_PIO4	IOHIZ[7:0]								IOHOLD[7:0]								IOWAIT[7:0]								IOSET[7:0]							
0xA000 0074	FSMC_ECCR2	ECC[31:0]																															
0xA000 0094	FSMC_ECCR3	ECC[31:0]																															

22. Интерфейс Secure digital (SDIO)

22.1. Основные свойства SDIO

Хост-интерфейс SD/SDIO MMC card (SDIO) соединяет периферийную шину AHB и MultiMediaCards (MMCs), карты памяти SD, карты SDIO и устройства CE-ATA.

Спецификация системы MultiMediaCard от технического комитета MMCA доступна на сайте MultiMediaCard Association www.mmca.org.

Спецификация системы карт памяти SD и карт SD I/O доступна на сайте SD card Association www.sdcard.org.

Спецификация системы CE-ATA доступна на сайте CE-ATA workgroup www.ce-ata.org.

Свойства SDIO:

- Полное соответствие *MultiMediaCard System Specification Version 4.2*. Поддержка трёх режимов шины данных карты: 1-бит (по умолчанию), 4-бит и 8-бит
- Полная совместимость с предыдущими версиями MultiMediaCards
- Полное соответствие *SD Memory Card Specifications Version 2.0*
- Полное соответствие *SD I/O Card Specification Version 2.0*: Поддержка двух режимов шины данных карты: 1-бит (по умолчанию) и 4-бит
- Полная поддержка CE-ATA (полное соответствие *CE-ATA digital protocol Rev 1.1*)
- Передача данных до 48 MHz в 8-бит режиме
- Сигналы разрешения вывода команд и данных для внешних двунаправленных драйверов.

NB: SDIO не имеет SPI-совместимого режима связи.

Протокол карт памяти SD это надстройка протокола MultiMediaCard, определённого в *MultiMediaCard system specification V2.11*. Некоторые команды устройств памяти SD не поддерживаются SD I/O-только картами или блоками I/O комбинированных карт. Некоторые из этих команд не используются в устройствах SD I/O, вроде команд стирания, и соответственно не поддерживаются SDIO. Кроме того, несколько команд карт памяти SD и карт SD I/O различаются и не поддерживаются SDIO. См. *SD I/O card Specification Version 1.0*. CE-ATA поддерживаются электрическим интерфейсом MMC по протоколу, использующему существующие примитивы доступа MMC. Определения электрических сигналов даны в руководствах MMC.

Шина MultiMediaCard/SD подключает карты к контроллеру.

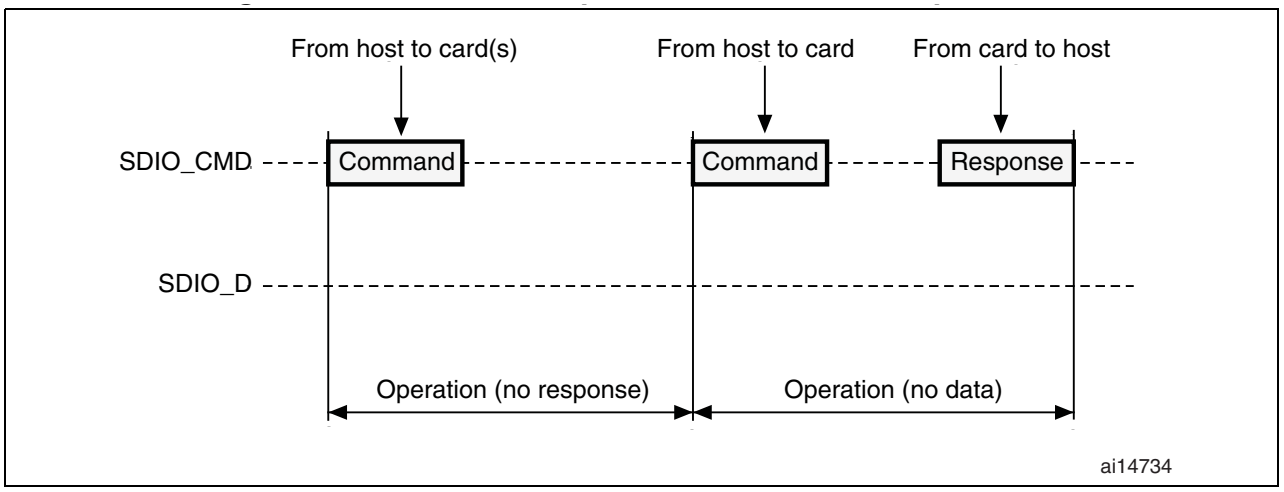
Текущая версия SDIO поддерживает только одну карту SD/SDIO/MMC4.2 одновременно и стек MMC4.1 или ранее.

22.2. Топология шины SDIO

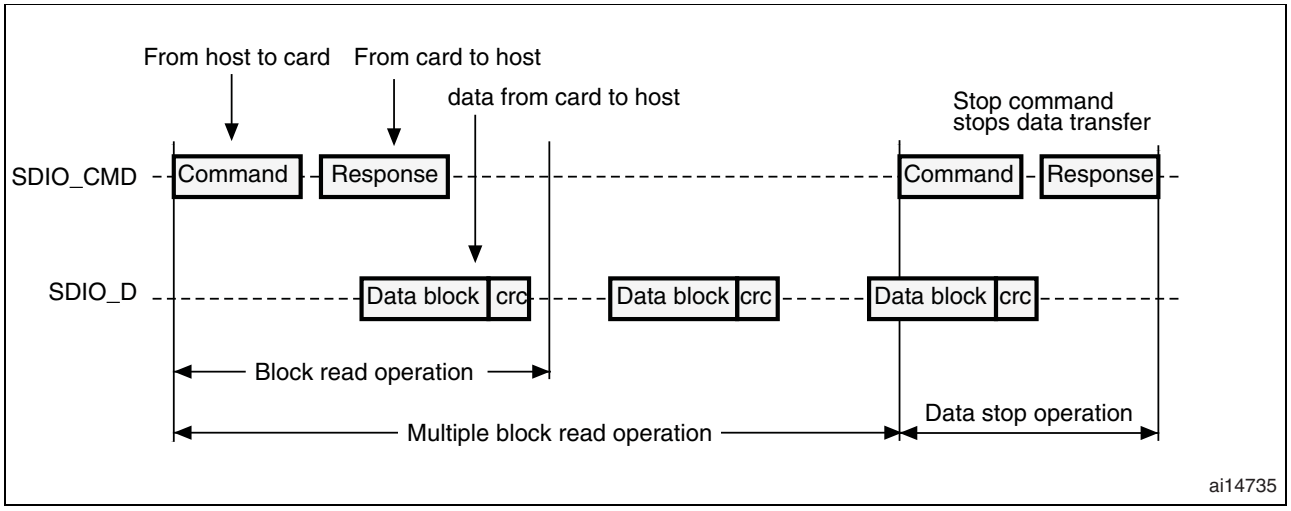
Связь по шине основана на передаче команд и данных в режиме команда/ответ. Информация передаётся внутри структур команды и ответа. Кроме того, некоторые операции имеют токены данных.

Данные карт памяти SD/SDIO передаются блоками. Данные MMC передаются блоками или потоком. Данные устройств CE-ATA передаются блоками.

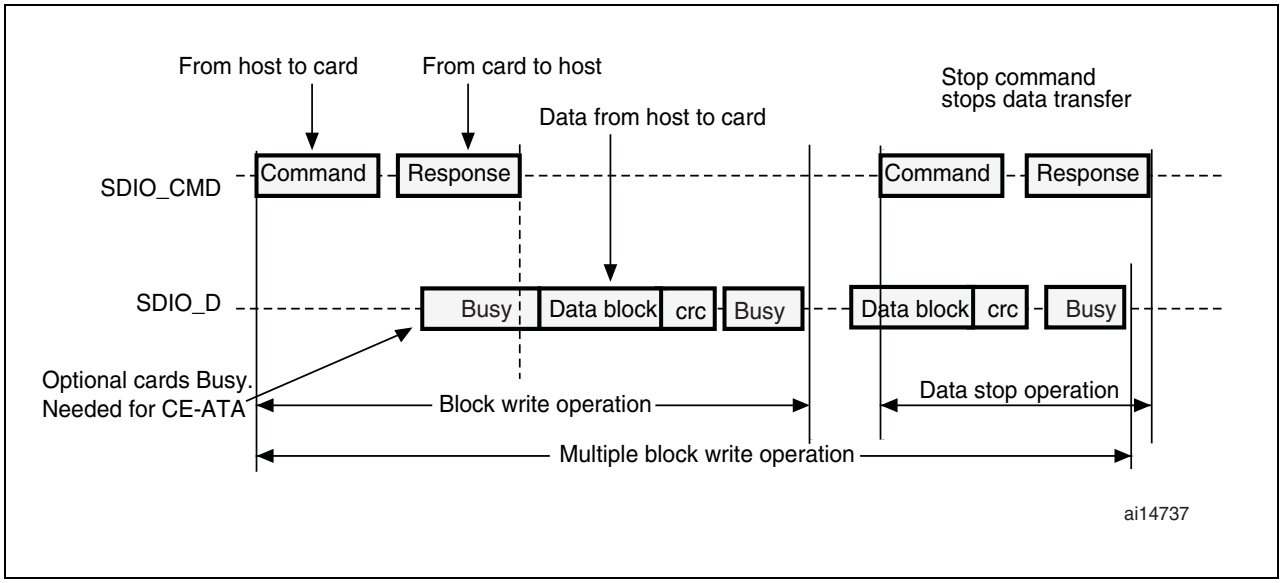
Операции "без ответа" и "без данных" SDIO.



Блоковое чтение SDIO.

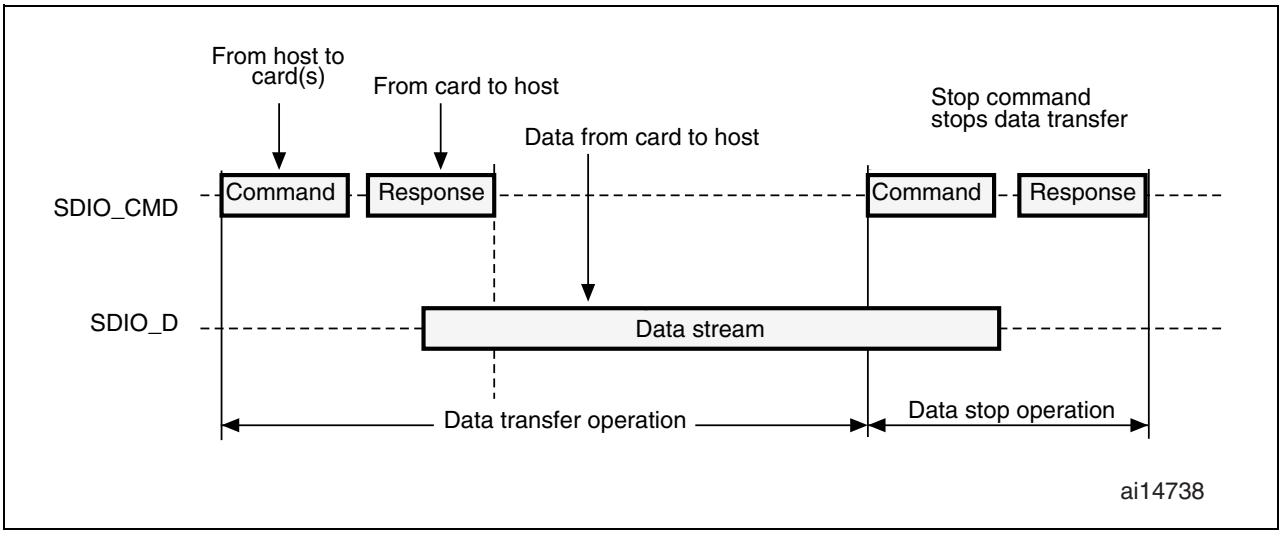


Блоковая запись SDIO.

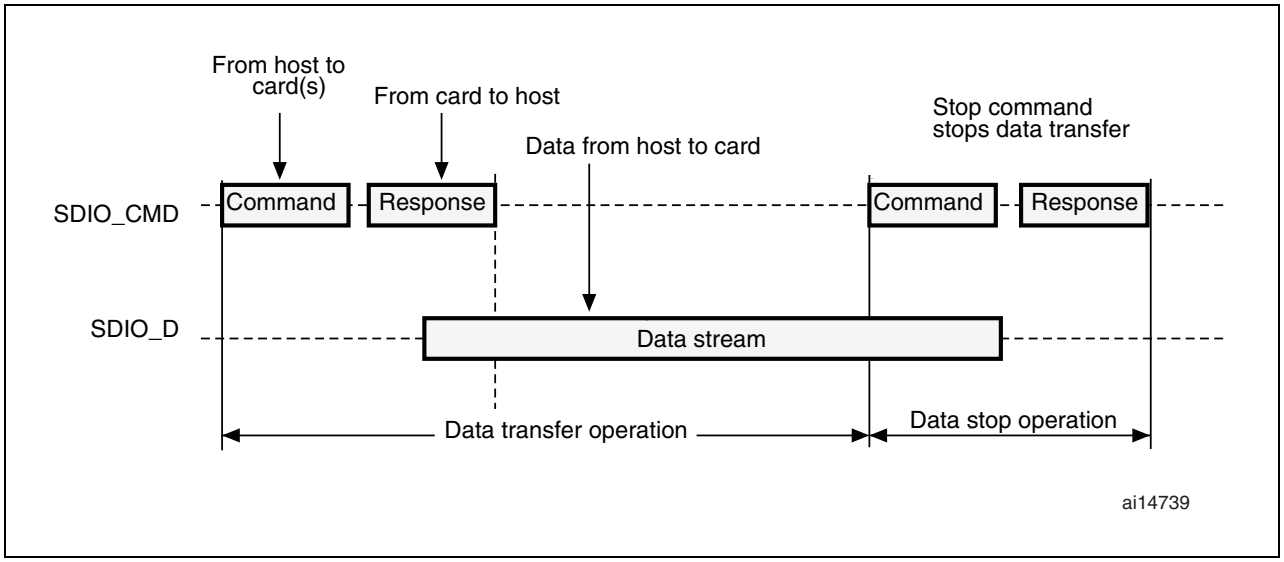


NB: При стоящем сигнале Busy (SDIO_D0 низкий) SDIO данные не посылает.

Последовательное чтение SDIO.



Последовательная запись SDIO.

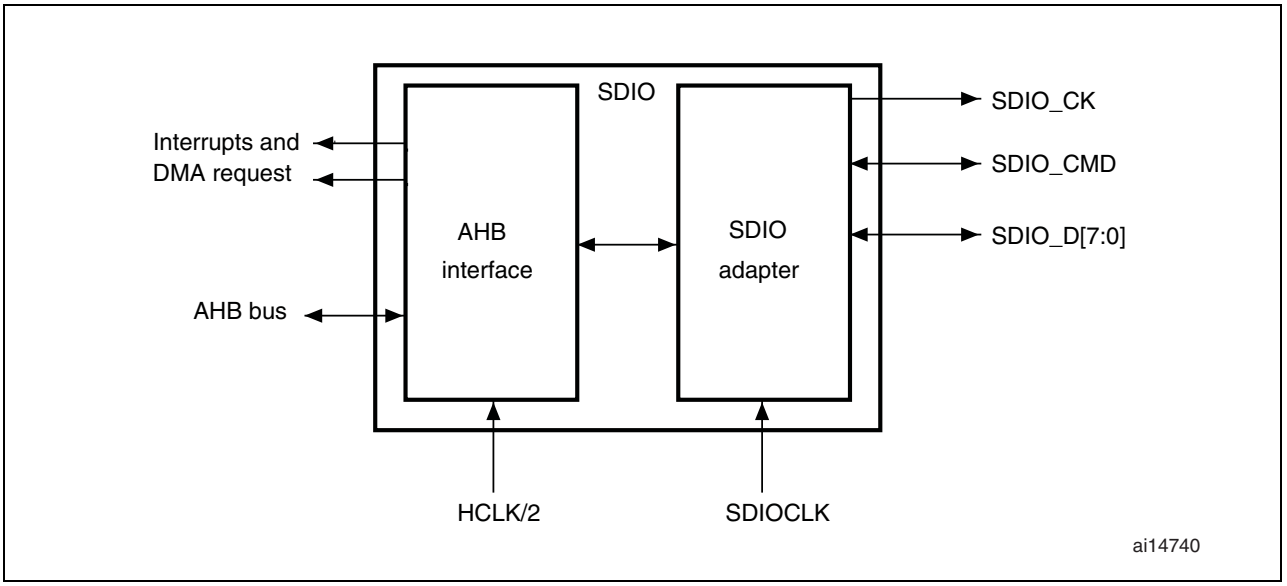


22.3. Функциональное описание SDIO

SDIO состоит из двух частей:

- Блок адаптера SDIO обеспечивает функции карт MMC/SD/SD I/O вроде тактов и передач.
- Интерфейс AHB для доступа к регистрам адаптера SDIO, выдачи прерываний и запросов DMA.

Блок-схема SDIO.



По умолчанию `SDIO_D0` используется для передачи данных. После инициализации хост может менять ширину шины данных.

С картами MultiMediaCard для передачи данных можно использовать `SDIO_D0`, `SDIO_D[3:0]` или `SDIO_D[7:0]`. MMC V3.31 и ранее поддерживают только 1 бит данных на `SDIO_D0`.

С картами SD SD I/O для передачи данных можно использовать `SDIO_D0` или `SDIO_D[3:0]`. Все линии данных работают в двухтактном режиме.

SDIO_CMD работает в двух режимах:

- Открытый сток при инициализации (только для MMCV3.31 и ранее)
- Двухтактный для передачи команд (SD/SD I/O карты MMC4.2 используют двухтактный режим и при инициализации)

SDIO_CK тактирует карту: по одному такту на один бит на линиях команд и данных. Частота может меняться от 0 MHz до 20 MHz (для MultiMediaCard V3.31), от 0 до 48 MHz для MultiMediaCard V4.0/4.2, и от 0 до 25 MHz (для SD/SD I/O карт).

SDIO использует два тактовых сигнала:

- Такты адаптера SDIO (SDIOCLK = HCLK)
- Такты шины AHB (HCLK/2)

Частоты PCLK2 и SDIO_CK должны быть:

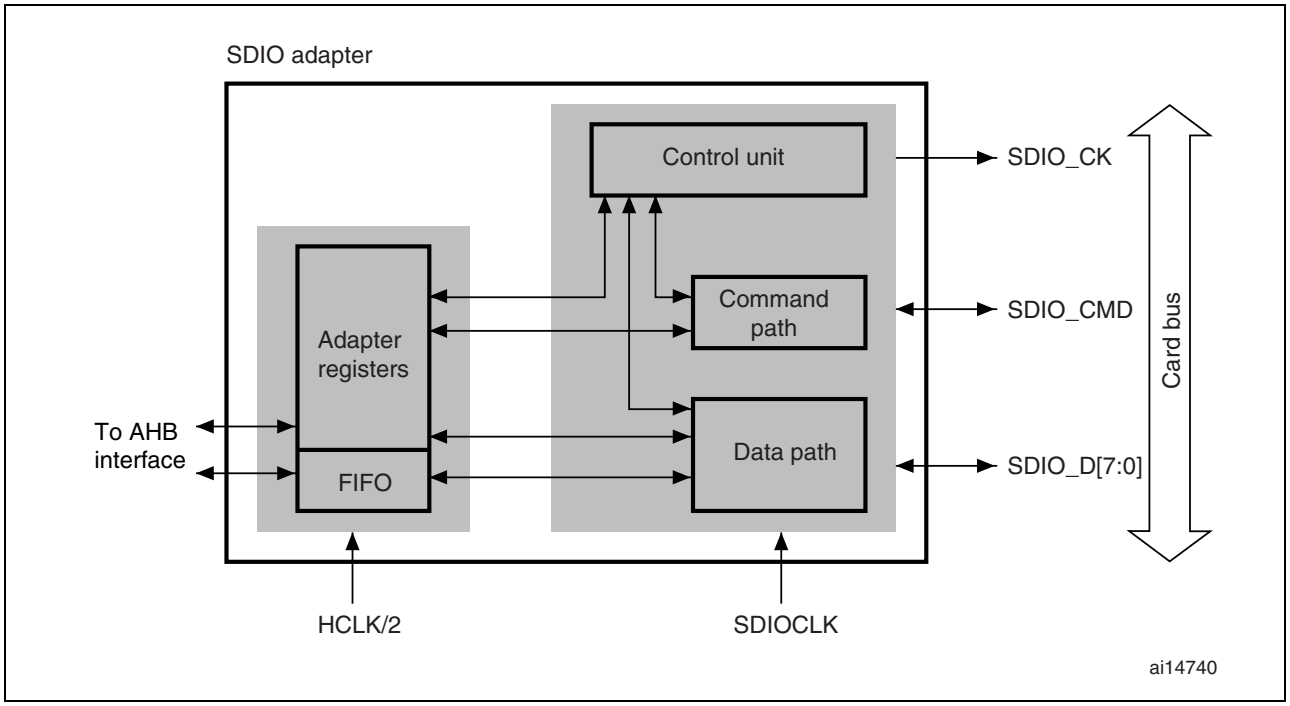
$$\text{Frequency(PCLK2)} \geq \frac{3}{8} \cdot \text{Frequency(SDIO\_CK)}$$

Таблица 137. Определения I/O SDIO.

Ножка	Направление	Описание
SDIO_CK	Вывод	Такты карт MultiMediaCard/SD/SDIO. Из хоста карте.
SDIO_CMD	Двунаправленная	Команда/ответ карты MultiMediaCard/SD/SDIO.
SDIO_D[7:0]	Двунаправленная	Данные от/для карты MultiMediaCard/SD/SDIO.

22.3.1. Адаптер SDIO

Блок-диаграмма.



Состоит из пяти под-блоков:

- Блок регистров адаптера
- Блок управления
- Путь команд
- Путь данных
- FIFO данных

NB: Регистры адаптера и FIFO используют домен тактов шины AHB (HCLK/2). Блок управления, путь команд и путь данных используют домен тактов адаптера SDIO (SDIOCLK).

Блок регистров адаптера

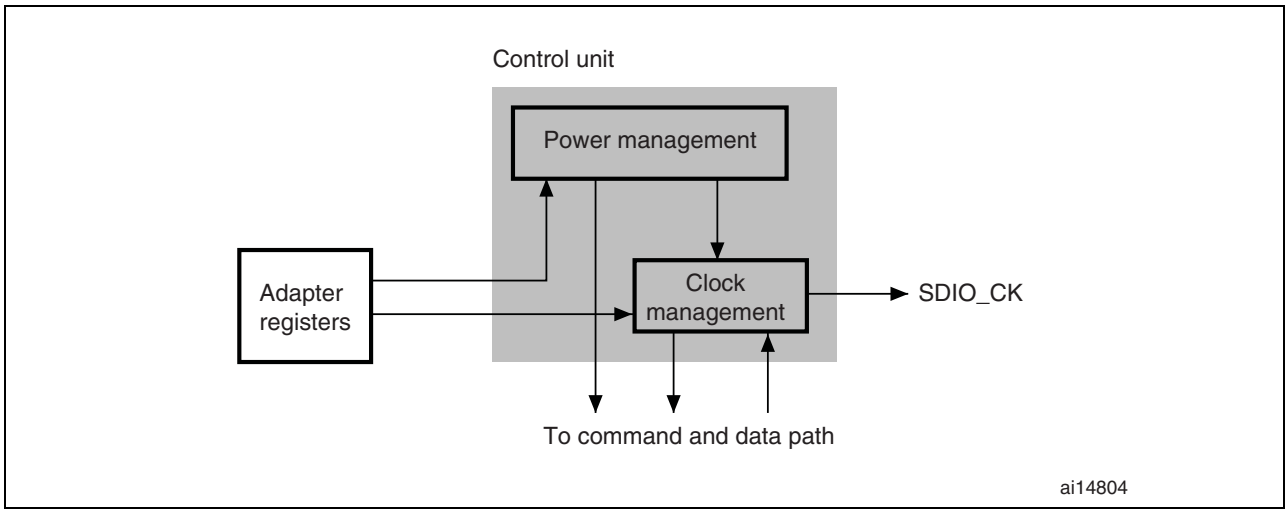
Содержит все системные регистры и генерирует сигналы очистки статических флагов карт мультимедиа по записи 1 в соответствующий бит регистра очистки SDIO.

Блок управления

Содержит управление питанием и делитель тактов для карт памяти.

Есть три фазы питания:

- Выключено
- Подъём
- Подача



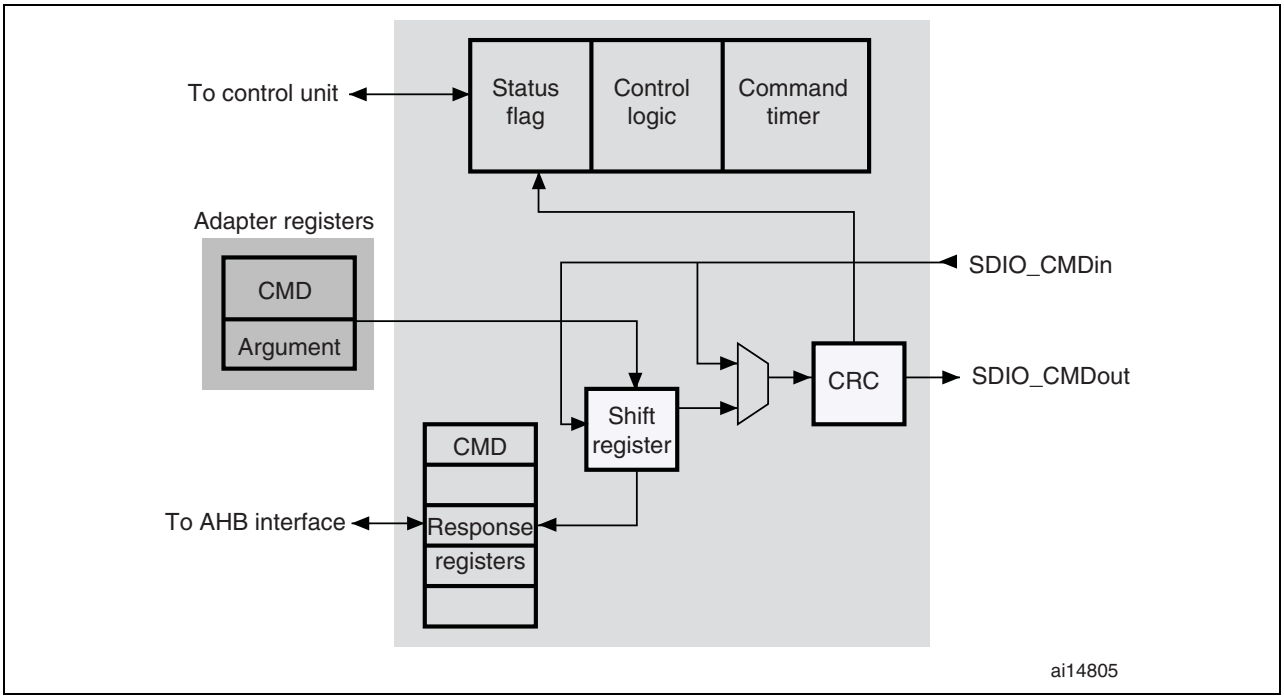
В фазах выключенного и поднимающегося питания блок управления питанием отключает выходные сигналы карты.

Блок тактирования выдаёт сигнал SDIO_CK, который можно получить делением тактов или обходом. Выход тактов неактивен:

- после сброса
- во время фаз выключенного и поднимающегося питания
- в режиме низкого энергопотребления и Простое шины карты (восемь тактов после перехода путей команд и данных в Простой)

Путь команд

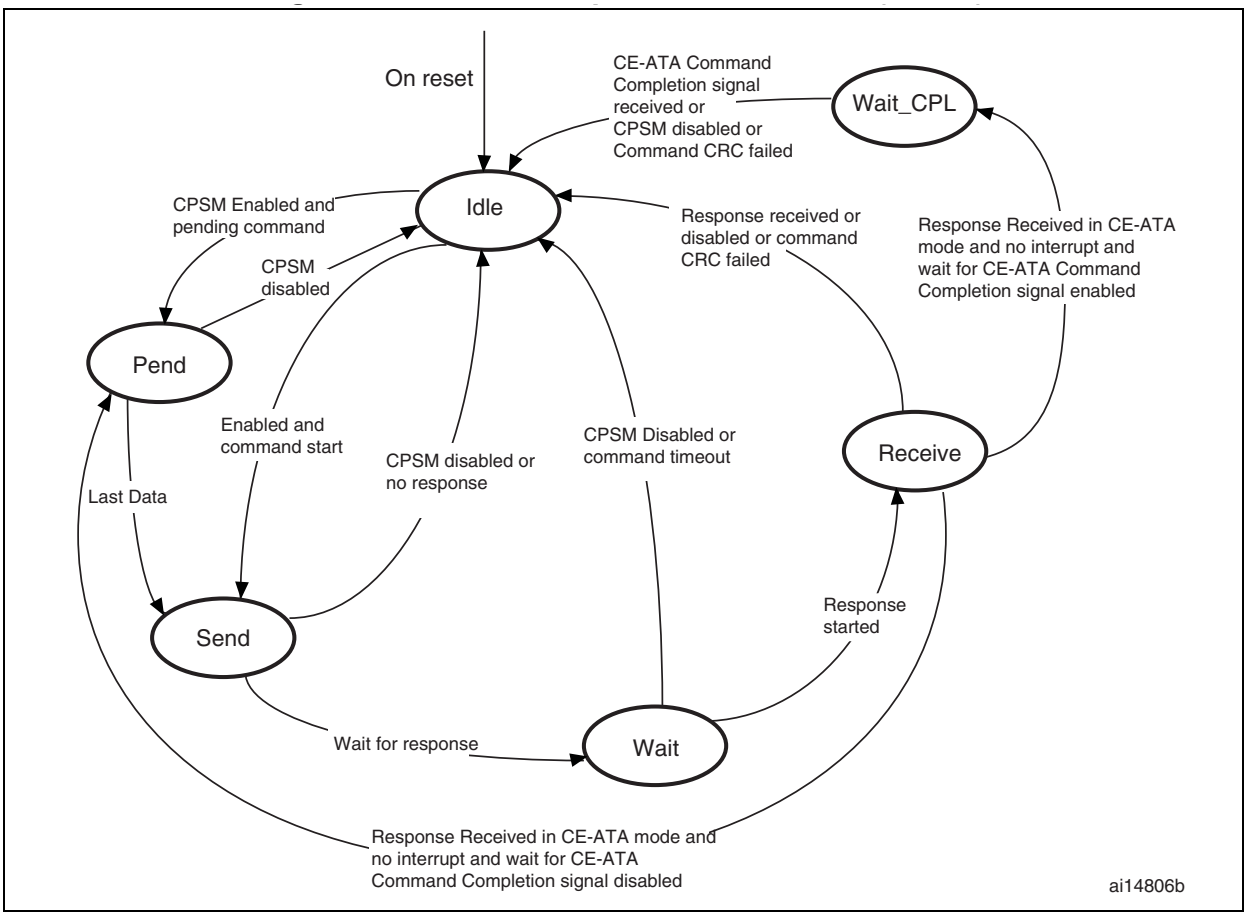
Путь команд посылает картам команды и принимает их ответы.



- Машина состояния пути машин (CPSM)

– Передача команды начинается после записи регистра и стоящем бите разрешения. Если ответ не требуется, то после передачи команды CPSM ставит флаг и переходит в Простой. Иначе она дожидается ответа. По получении ответа полученный CRC сравнивается с полученным внутри и ставится и ставится соответствующий флаг.

Машина состояния пути команд (CPSM)

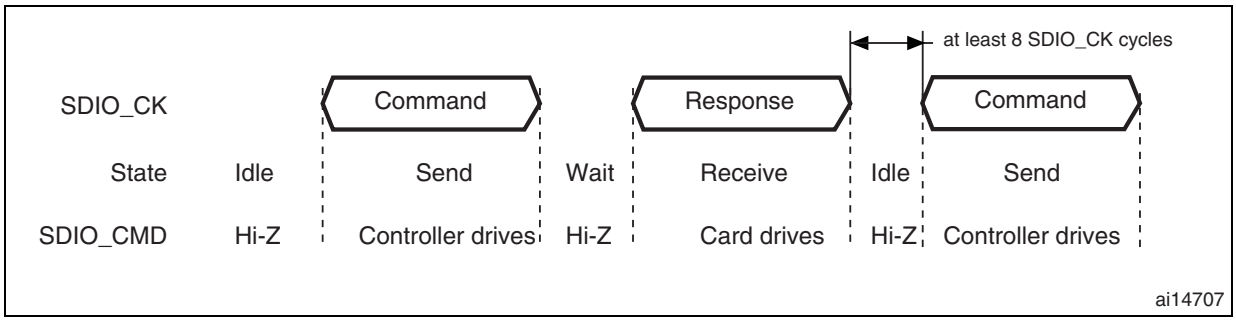


При переходе в Ожидание запускается таймер команды на 64 такта SDIO_CK. Если он истекает до перехода CPSM в Приём, то ставится флаг таймаута и CPSM переходит в Простой.

Если стоит флаг прерывания, то таймер выключается и CPSM ждёт прерывания от одной из карт. Если стоит бит удержания, то CPSM переходит в Удержание и ждёт сигнала CmdPend от пути данных. При обнаружении CmdPend CPSM переходит в состояние Посылки. Этим счётчику данных разрешается остановка передачи команд.

NB: CPSM остаётся в простое не меньше восьми периодов SDIO_CK ради удовлетворения требований NCC и NRC. NCC это минимальная задержка между двумя командами хоста, а NRC это минимальная задержка между командой хоста и ответом карты.

Передача команды.



• Формат команды

– Команда: это токен пуска операции. Она посылается хостом одной (адресная команда) или всем подключённым картам (широковещательные команды доступны для MMC V3.31 и раньше). Команды передаются последовательно по линии CMD. Они имеют длину 48 бит. Общий формат токена команд для MultiMediaCards, карт памяти SD и карт SDIO приведён в таблице 138. Команды CE-ATA это расширение команд MMC V4.2 с тем же форматом.

Путь команд работает в полудуплексном режиме, так что команды и ответы можно посылать и принимать. При CPSM не в режиме Посылки выход SDIO_CMD уходит в состояние Hi-Z. Данные на SDIO_CMD синхронны с передним фронтом SDIO_CK.

– Ответ: это токен ответа адресованной карты (или синхронно от всех подключённых карт для MMC V3.31 или раньше) на переданную хостом команду. Ответы передаются последовательно по линии CMD.

SDIO поддерживает два типа ответов, оба с контролем CRC:

- короткий 48-бит
- длинный 136-бит

NB: Если ответ не содержит CRC (ответ CMD1), то драйвер должен игнорировать сбой CRC.

Таблица 138. Формат команды.

Позиция бита	Ширина	Значение	Описание
47	1	0	Стартовый бит
46	1	1	Бит передачи
[45:40]	6	-	Индекс команды
[39:8]	32	-	Аргумент
[7:1]	7	-	CRC7
0	1	1	Концевой бит

Таблица 139. Формат короткого ответа.

Позиция бита	Ширина	Значение	Описание
47	1	0	Стартовый бит
46	1	0	Бит передачи
[45:40]	6	-	Индекс команды
[39:8]	32	-	Аргумент
[7:1]	7	-	CRC7(или 1111111)
0	1	1	Концевой бит

Таблица 140. Формат длинного ответа.

Позиция бита	Ширина	Значение	Описание
135	1	0	Стартовый бит
134	1	0	Бит передачи
[133:128]	6	111111	Резерв
[127:1]	127	-	CID или CSD (включая внутренний CRC7)
0	1	1	Концевой бит

Регистр команд содержит индекс команды (посылаемые карте шесть бит) и её тип. Они определяют необходимость ответа и его длину. Путь команд реализует флаги состояния:

Таблица 141. Флаги состояния пути команд.

Флаг	Описание
CMDREND	Нормальный CRC ответа.
CCRCFAIL	Сбойный CRC ответа.
CMDSENT	Команда послана (ответа не надо)
CTIMEOUT	Таймаут ответа.
CMDACT	Команда передаётся.

Генератор CRC вычисляет контрольную сумму всех битов перед кодом CRC. Включаются стартовый бит, бит передачи, индекс команды и её аргумент (или состояние карты). В длинном ответе контрольная сумма вычисляется для первых 120 бит `CID` или `CSD`. Стартовый бит, бит передачи и шесть резервных бит в вычислении CRC не участвуют.

Контрольная сумма CRC это 7-бит значение:

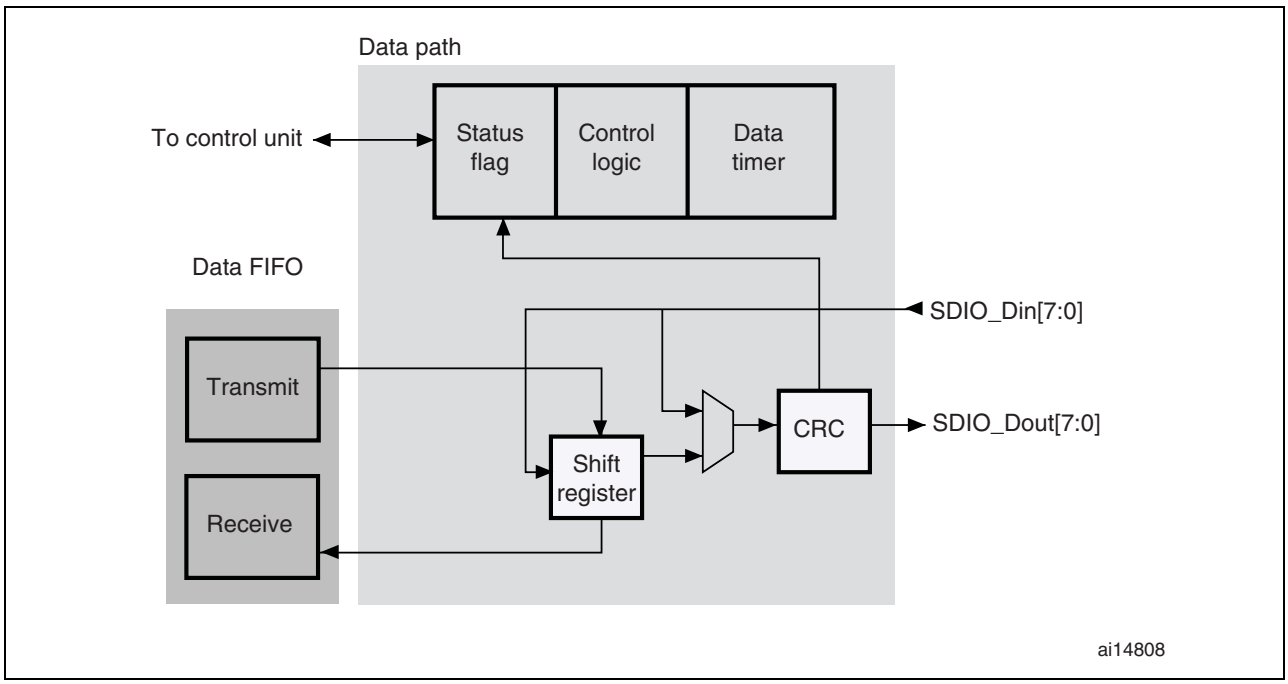
CRC[6:0] = Остаток [(M(x) * x^7) / G(x)]

G(x)=x^7 +x^3 +1

M(x) = (начальный бит) * x^{39} + ... + (последний бит перед CRC) * x^0, или

M(x) = (начальный бит) * x^{119} + ... + (последний бит перед CRC) * x^0

Путь данных.



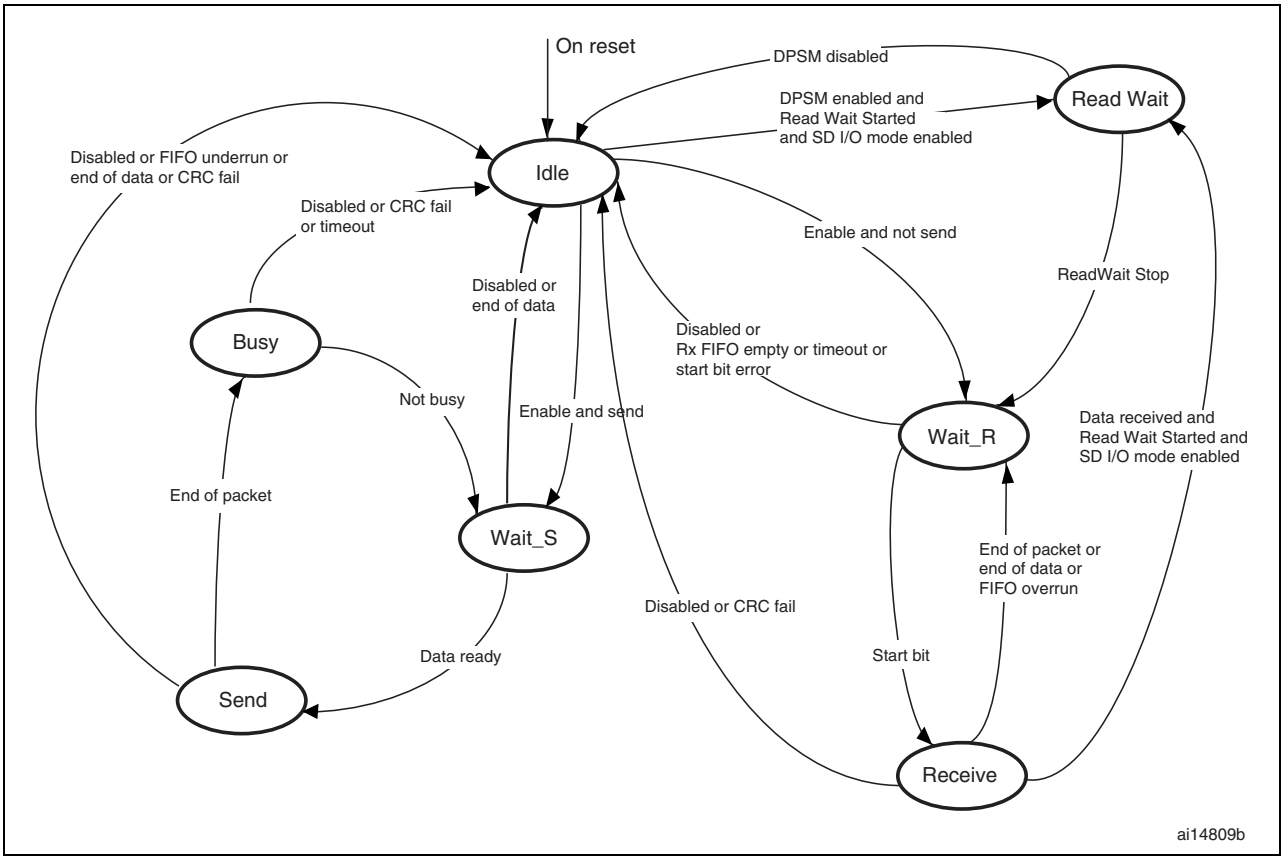
Ширина шины данных карты определяется в регистре управления тактами. При 4-бит шине данные передаются по четыре бита на такт по сигналам `SDIO_D[3:0]`. При 8-бит шине данные передаются по восемь битов на такт по сигналам `SDIO_D[7:0]`. При отключённой ширине шины данные передаются по одному биту на такт по `SDIO_D0`.

В зависимости от направления передачи машина состояния пути данных (DPSM) переходит в состояние Wait_S или Wait_R, если можно:

- Передача: DPSM идёт в Wait_S. Если в FIFO есть данные, то DPSM переходит в состояние Посылки и начинает посылать карте данные.
- Приём: DPSM идёт в Wait_R и ждёт стартового бита. При его появлении DPSM переходит в состояние Приёма и начинает принимать данные от карты.

Машина состояния пути данных (DPSM)

DPSM работает с частотой SDIO_CK. Сигналы данных на шине карты синхронны с передним фронтом SDIO_CK. DPSM имеет шесть состояний.



- **Простой**: путь данных неактивен, выходы SDIO_D[7:0] в Hi-Z. После записи регистра управления данных и установки бита разрешения DPSM пишет новое число в счётчик данных и в зависимости от направления передачи переходит в Wait_S или Wait_R.
- **Wait_R**: если счётчик данных нулевой, то при пустом FIFO DPSM идёт в Простой. Если счётчик данных не нулевой, то DPSM ждёт стартового бита на SDIO_D. При получении его до таймаута DPSM уходит в Приём и заполняет счётчик данных блока. При достижении таймаута до стартового бита или его ошибке он уходит в Простой и ставит бит таймаута.
- **Приём**: принятые последовательные данные пишутся в FIFO данных. В зависимости от бита в регистре управления режим передачи может быть блоками или потоком:
  - В режиме блоков, при исчерпании счётчика данных блока DPSM дожидается приёма CRC. При его совпадении с вычисленным DPSM идёт в Wait_R. Иначе ставится флаг сбоя CRC с уходом в Простой.
  - В режиме потока DPSM принимает данные до исчерпания счётчика. Тогда остаток данных в регистре сдвига пишется в FIFO и DPSM идёт в Wait_R.
  - При переполнении FIFO DPSM ставит флаг ошибки FIFO и идёт в Простой.
- **Wait_S**: При нулевом счётчике данных DPSM уходит в Простой. Иначе он ждёт снятия флага пустого FIFO данных и идёт в Посылку.

  NB: DPSM остаётся в Wait_S не менее двух тактов для соблюдения N_{WR}, где N_{WR} это число тактов от приёма ответа карты и стартом передачи данных от хоста.
- **Посылка**: DPSM начинает передачу данных карте. В зависимости от бита в регистре управления режим передачи может быть блоками или потоком:
  - В режиме блоков, при исчерпании счётчика данных блока DPSM посылает вычисленный CRC с битом конца и уходит в Занят.
  - В режиме потока DPSM посылает данные карте пока стоит бит разрешения и счётчик не нулевой. Затем спокойно уходит в Простой.

  При исчерпании FIFO DPSM ставит флаг ошибки FIFO и идёт в Простой.

- **Занят**: DPSM ждёт флага состояния CRC:
  - Если он появляется плохим, то уходит в Простой и ставит флаг сбоя CRC.
  - При хорошем CRC и высоком SDIO_D0 (карта не занята) идёт в Wait_S.

  При появлении таймаута в этом состоянии DPSM ставит флаг таймаута данных и уходит в Простой.

  В режимах DPSM Wait_R и Занят таймер данных включён и выдаёт ошибку таймаута данных когда:
  - При передаче данных DPSM остаётся Занятым дольше периода таймаута
  - При приёме данных не достигнут конец данных и DPSM стоит в Wait_R дольше периода таймаута.
- **Данные:** передача по линиям данных работает. Они хранятся в FIFO из 32 слов по 32 бита.

Таблица 142. Формат токена данных.

Описание	Стартовый бит	Данные	CRC16	Бит конца
Блок данных	0	-	Да	1
Поток	0	-	Нет	1

FIFO данных

Это буфер данных с блоками приёма и передачи из 32-х 32-бит слов. Тактируется от домена AHB (HCLK/2), так что сигналы в домен SDIO (SDIOCLK) ресинхронизируются.

Битами TXACT и RXACT FIFO может отключаться, включаться приём или передача.Взаимно исключающие TXACT и RXACT управляются путём данных:

• FIFO передачи:
  - При стоящем TXACT FIFO является логикой и буфером передачи
  - При стоящем RXACT FIFO является логикой и буфером приёма

В передающий FIFO данные пишутся по интерфейсу AHB при разрешении передач SDIO.

Передающий FIFO доступен по 32 последовательным адресам с указателями чтения и записи. Указатель чтения инкрементируется при загрузке путём данных своего регистра сдвига.

При выключенном FIFO все флаги состояния снимаются. Путь данных ставит TXACT во время передачи.

Таблица 143. Флаги состояния FIFO передачи.

Флаг	Описание
TXFIFOF	Все 32 слова FIFO имеют данные.
TXFIFOE	Данных в FIFO нет.
TXFIFOHE	Половина или больше слов FIFO пусты. Может стать запросом DMA.
TXDAVL	Данные в FIFO есть. Инверсия флага TXFIFOE.
TXUNDERR	Ошибка исчерпания. Снимается записью в регистр очистки SDIO.

- FIFO приёма

После получения слова данных путь данных выставляет его на на шину записи. Указатель записи инкрементируется после завершения операции. На стороне чтения слово FIFO по текущему указателю чтения выставляется на шину чтения. При выключенном FIFO чтения все флаги снимаются и указатели чтения и записи сбрасываются. Путь данных выставляет RXACT во время приёма. Доступен приёмный FIFO по 32 последовательным адресам.

Таблица 144. Флаги состояния FIFO приёма.

Флаг	Описание
RXFIFOF	Все 32 слова FIFO имеют данные.
RXFIFOE	Данных в FIFO нет.
RXFIFOHF	Половина или больше слов FIFO полны. Может стать запросом DMA.
RXDAVL	Данные в FIFO есть. Инверсия флага RXFIFOE.
RXOVERR	Ошибка переполнения. Снимается записью в регистр очистки SDIO.

22.3.2. AHB интерфейс SDIO

Интерфейс AHB выдаёт прерывания и запросы DMA, обращается к регистрам адаптера SDIO и FIFO. Состоит из пути данных, декодера регистров и логики прерываний/DMA.

Прерывания SDIO

Прерывания вызываются стоящим флагом состояния, при стоящем соответствующем бите в регистре маски.

Интерфейс SDIO/DMA: процедура передачи данных

Ниже приведена передача из хост контроллера SDIO в MMC (512 байт с использованием `CMD24` (`WRITE_BLOCK`)). FIFO SDIO заполняется данными из памяти контроллером DMA.

1. Идентифицировать карту
2. Увеличить частоту SDIO_CK
3. Выбрать карту посылкой `CMD7`
4. Сконфигурировать DMA2:
   a) Включить контроллер DMA2 и очистить удерживаемые прерывания
   b) Поставить адрес источника DMA2_Channel4 на память и адрес получателя на адрес регистра `SDIO_FIFO`
   c) Установить регистр управления DMA2_Channel4 (инкремент памяти, периферия без инкремента, ширина источника и получателя - слово)
   d) Включить DMA2_Channel4
5. Послать `CMD24` (`WRITE_BLOCK`):
   a) Записать регистр длины данных SDIO (Регистр таймера данных SDIO надо писать перед идентификацией карты)
   b) Записать в регистр аргумента SDIO адрес в памяти карты (куда писать)
   c) Установить регистр команд SDIO: "24" в `CmdIndex` (`WRITE_BLOCK`); "1" в `WaitResp` (хост SDIO карты ждёт ответа); "1" в `CPSMEN` (хост SDIO может посылать команду). Другие поля остаются в состоянии сброса.
   d) Дождаться прерывания `SDIO_STA[6]` = `CMDREND`, записать регистр управления данными SDIO: '1' в `DTEN` (хост SDIO может посылать данные); '0' в `DTDIR` (из контроллера в карту); '0' в `DTMODE` (передача блока); '1' в `DMAEN` (DMA включён); `0x9` в `DBLOCKSIZE` (512 байт). Иные поле не волнуют.
   e) Дождаться `SDIO_STA[10]` = `DBCKEND`
6. Опросить регистр состояния каналов DMA на предмет ещё включённых каналов.

22.4. Функциональное описание карты

22.4.1. Режим идентификации карт

Тут хост сбрасывает все карты на шине, определяет диапазон рабочего напряжения, идентифицирует карты и устанавливает из относительные адреса (`RCA`) на шине. Все данные передаются только по линии команд (`CMD`).

22.4.2. Сброс карт

Команда сброса GO_IDLE_STATE (CMD0) переводит MultiMediaCard и карты памяти SD в Простой. Команда IO_RW_DIRECT (CMD52) сбрасывает карты SD I/O. После подачи питания или CMD0 все драйверы шины переводятся в высокоимпедансное состояние и карты инициализируются с относительным адресом (RCA=0x0001) и установками каскадов драйверов (наименьшая скорость, наибольший ток) по умолчанию.

22.4.3. Определение рабочего напряжения

Все карты беседуют с хостом SDIO на рабочем уровне напряжения V_{DD}, определённом в регистре условий работы (OCR) карты.

Карты, хранящие идентификационный номер (CID) и специфичные данные карты (CSD) в памяти данных, могут обмениваться этой информацией только в условиях V_{DD}. При различных уровнях V_{DD} хоста SDIO и карты она не может закончить цикл идентификации и послать данные CSD. Для целей идентификации и извлечения карт, не попадающих в требуемое окно V_{DD}, придуманы специальные команды: SEND_OP_COND (CMD1), SD_APP_OP_COND (ACMD41 для памяти SD) и IO_SEND_OP_COND (CMD5 для SD I/O). Хост SDIO посылает требуемое окно V_{DD} в операндах этих команд. Карты, которые не могут этого выполнить, отключаются от шины и переходят в неактивное состояние.

Этими командами без диапазона напряжений в параметрах хост SDIO может опросить все карты для определения общего диапазона и перевода неподходящих карт в неактивное состояние. Кроме того, этим можно по требованию пользователя определить неиспользуемые карты.

22.4.4. Процесс идентификации карт

Он различен для MultiMediaCard и SD карт.

Для MultiMediaCard процесс стартует на частоте F_{od}. Выходной каскад линии SDIO_CMD стоит в режиме открытого стока, разрешая параллельную работу карт. Регистрация идёт так:

1. Активируется шина.
2. Хост SDIO широковещает SEND_OP_COND (CMD1) для приёма условий работы.
3. Ответ это проводное AND регистров условий работы.
4. Несовместимые карты уходят в неактивное состояние.
5. Хост SDIO широковещает ALL_SEND_CID (CMD2) всем активным картам.
6. Активные карты одновременно шлют свои CID номера по последовательной линии. Карты с отдельными битами CID, не совпадающими с битами на командной линии, останавливают передачу и должны ждать следующего цикла идентификации. Одна карта, успешно передавшая полный CID, идёт в состояние Идентификации.
7. Хост SDIO посылает ей SET_RELATIVE_ADDR (CMD3) с адресом. Это относительный адрес карты (RCA); он короче CID. Обозначенная карта уходит в Дежурство (Standby) и не реагирует на следующие циклы идентификации. Выходной каскад переключается из открытого стока в двухтактный режим.
8. Хост SDIO повторяет шаги 5 — 7 до получения таймаута.

У SD карт процесс стартует на частоте F_{od} и выходной каскад линии SDIO_CMD стоит в двухтактном режиме вместо режиме открытого стока. Регистрация идёт так:

1. Активируется шина.
2. Хост SDIO широковещает SD_APP_OP_COND (ACMD41).
3. Карты отвечают содержимым их регистров условий работы.
4. Несовместимые карты уходят в неактивное состояние.
5. Хост SDIO широковещает ALL_SEND_CID (CMD2) всем активным картам.
6. Карты шлют свои CID и уходят в Идентификацию.
7. Хост SDIO посылает активной карте SET_RELATIVE_ADDR (CMD3) с адресом. Это относительный адрес карты (RCA); он короче CID. Обозначенная карта уходит в Дежурство (Standby). Этой командой хост SDIO может изменять RCA карт.
8. Хост SDIO повторяет шаги 5 — 7 со всеми активными картами.

Для карт SD I/O регистрация идёт так:

1. Активируется шина.
2. Хост SDIO посылает IO_SEND_OP_COND (CMD5).
3. Карты отвечают содержимым их регистров условий работы.
4. Несовместимые карты уходят в неактивное состояние.
5. Хост SDIO посылает активной карте SET_RELATIVE_ADDR (CMD3) с адресом. Это относительный адрес карты (RCA); он короче CID. Обозначенная карта уходит в Дежурство (Standby). Этой командой хост SDIO может изменять RCA карт.

22.4.5. Запись блоками

Во время блоковой записи (CMD24 – 27) хост посылает карте, поддерживающей такой режим, один или несколько блоков данных с добавленным в конце CRC блока. Длина блока определена в WRITE_BL_LEN. При сбое CRC карта показывает его на линии SDIO_D, отбрасывает передаваемые данные и игнорирует остальные передаваемые блоки многоблоковой передачи.

Если хост использует неполные блоки с общей длиной не кратной длиной блока и невыравненные блоки не разрешены (параметр CSD WRITE_BLK_MISALIGN не стоит), то карта обнаружит ошибку перед началом первого невыравненного блока (бит ADDRESS_ERROR в регистре состояния. Также операция записи будет прервана при попытки записи в защищённую область. При этом ставится бит WP_VIOLATION.

Запись регистров CID и CSD не требует предварительной установки длины блока. Передаваемые данные также защищены CRC. Если часть регистра CSD или CID хранится в ROM, то эта часть сравнивается с соответствующими данными в приёмном буфере. При несовпадении карта сообщает от ошибке и не меняет остальную часть регистров. Некоторые карты требуют долгого и непредсказуемого времени для записи блока данных. После приёма блока и проверки CRC карта начинает запись и держит линию SDIO_D низкой вплоть до готовности приёма новой команды WRITE_BLOCK. Хост может в любое время опросить состояние карт командой SEND_STATUS (CMD13). Бит READY_FOR_DATA показывает готовность к приёму данных. Задумчивую карту можно перевести в режим Отключки (Disconnect) командой CMD7, не прерывая её записи. При этом линии SDIO_D освобождаются для других карт. При повторном выборе карты она показывает свою занятость состоянием SDIO_D (низкий при продолжающейся записи и недоступном буфере).

22.4.6. Чтение блоками

Максимальный размер блока передачи определён в CSD (READ_BL_LEN). При стоящем бите READ_BL_PARTIAL можно передавать блоки меньшего размера, чьи адреса начала и конца лежат внутри физического блока. В конце каждого блока добавляется CRC. Команда CMD17 (READ_SINGLE_BLOCK) инициирует чтение одного блока с возвращением состояние Передачи.

Команда CMD18 (READ_MULTIPLE_BLOCK) запускает передачу нескольких последовательных блоков. Командой остановки передачи хост может прервать такое чтение в любое время, независимо от типа.

Если во время множественной передачи (обоих типов) возникает ошибка (выход за границы, невыравненный адрес или внутренняя ошибка), то карта прекращает передачу и возвращается в состояние Данных. Хост должен послать команду остановки передачи. Ошибка чтения возвращается в ответ на эту команду.

После передачи предопределённого числа блоков карта выходит из состояния Данных и присылаемая команда остановки передачи является незаконной, о чём и сообщается хосту. При неразрешённой передаче невыравненных блоков ошибка выявляется в начале первого невыравненного блока (бит ошибки ADDRESS_ERROR в регистре состояния).

22.4.7. Потоковые чтение и запись (только MultiMediaCard)

В потоковом режиме данные передаются байтами без добавления CRC в конце каждого блока.

Потоковая запись (только MultiMediaCard)

WRITE_DAT_UNTIL_STOP (CMD20) пускает передачу данных от хоста SDIO карте, начиная с заданного адреса, вплоть до команды останова от хоста. Если разрешены неполные блоки (параметр CSD WRITE_BLK_MISALIGN стоит), то поток данных может начинаться и заканчиваться на любом адресе внутри адресного пространства карты, иначе только на границе блока. Поскольку количество

данных заранее определить невозможно, то CRC использовать невозможно. Если во время передачи достигнут конец диапазона памяти, то все поступающие данные выбрасываются вплоть до получения команды останова.

Максимальная частота тактов потока использует регистры карты и вычисляется по формуле:

$$\text{Maximumspeed} = \text{MIN}(\text{TRANSPEED}, \frac{(8 \times 2^{\text{writebllen}})(-\text{NSAC})}{\text{TAAC} \times \text{R2WFACTOR}})$$

- Maximumspeed = максимальная частота записи
- TRANSPEED = максимальная частота передачи данных
- writebllen = максимальный размер блока записи
- NSAC = время 2 чтения в тактах CLK
- TAAC = время 1 чтения
- R2WFACTOR = множитель скорости записи

Попытка хоста использовать частоту, большую, чем доступная карте, то она останавливает запись, ставит в регистре состояния ошибку OVERRUN и плюёт на все поступающие данные (в состоянии приёма данных) вплоть до получения команды останова. Операция записи прерывается также при записи в защищённую область. В этом случае ставится бит WP_VIOLATION.

Потоковое чтение (только MultiMediaCard)

READ_DAT_UNTIL_STOP (CMD11) запускает чтение данных карты начиная с указанного адреса вплоть до посылки хостом команды STOP_TRANSMISSION (CMD12). Команда остановки передаётся последовательно и передача данных останавливается после получения конечного бита команды. Если во время передачи достигнут конец диапазона памяти, то вплоть до получения команды остановки посылается всякая ерунда.

Максимальная частота тактов потока использует регистры карты и вычисляется по формуле:

$$\text{Maximumspeed} = \text{MIN}(\text{TRANSPEED}, \frac{(8 \times 2^{\text{readbllen}})(-\text{NSAC})}{\text{TAAC} \times \text{R2WFACTOR}})$$

- Maximumspeed = максимальная частота чтения
- TRANSPEED = максимальная частота передачи данных
- readbllen = максимальный размер блока чтения
- NSAC = время 2 чтения в тактах CLK
- TAAC = время 1 чтения
- R2WFACTOR = множитель скорости записи

Попытка хоста использовать частоту, большую, чем доступная карте, то она останавливает передачу, ставит в регистре состояния ошибку UNDERRUN и в состоянии Данных ждёт получения команды останова.

22.4.8. Стирание: групповое и сектора

Группа Стирания MultiMediaCard измеряется в блоках записи, основной единицы записи карты. Размер группы определён в CSD. Хост может стирать непрерывный диапазон групп стирания. Процесс запускается последовательностью из трёх шагов.

Сначала хост командой ERASE_GROUP_START (CMD35) указывает адрес начала диапазона, затем командой ERASE_GROUP_END (CMD36) задаёт последний адрес диапазона и наконец запускает стирание командой ERASE (CMD38). Поле адреса в команде стирания содержит адрес группы стирания в байтах. Карта выравнивает этот адрес на границу группы, игнорируя его младшие биты.

При подаче команды стирания вне последовательности в регистре состояния ставится бит ERASE_SEQ_ERROR и сбрасывается вся последовательность.

Если внутри последовательности появляется неположенная команда (кроме SEND_STATUS), то карта ставит в регистре состояния бит ERASE_RESET, сбрасывает последовательность и выполняет последнюю команду.

Попадающие в диапазон стирания защищённые блоки остаются нетронутыми и ставится бит WP_ERASE_SKIP.

Во время стирания линия SDIO_D удерживается низкой. Хост может освободить её, отменив выбор карты командой CMD7.

22.4.9. Широкая шина

Режим широкой шины (4 бита) включается и выключается командой SET_BUS_WIDTH (ACMD6). По включению питания и команде GO_IDLE_STATE (CMD0) используется 1 бит. SET_BUS_WIDTH (ACMD6) допустима только в состоянии передачи, то есть после её выбора командой SELECT/DESELECT_CARD (CMD7).

22.4.10.Управление защитой записи

Хост SDIO поддерживает три метода защиты записи:

1. Внутренняя защита (ответственность карты)
2. Механический переключатель (ответственность хоста SDIO)
3. Защита карты паролём

Внутренняя защита

Биты в CSD включают постоянную или временную защиту записи всей карты. Некоторые карты битом WP_GRP_ENABLE в CSD позволяют включать защиту групп секторов памяти и изменять её программно. Объём защищённой памяти измеряется в единицах WP_GRP_SIZE секторов из CSD. Команды SET_WRITE_PROT и CLR_WRITE_PROT управляют защитой адресованных групп. Команда SEND_WRITE_PROT подобна чтению одного блока. Карта посылает блок данных с 32 битами защиты записи, представляющими 32 защищённые группы, начиная с указанного адреса, сопроводив это 16 битами CRC. Поле адреса в команде защиты это байтовый адрес группы.

Карта игнорирует биты, младше размера группы.

Механический переключатель

Он расположен сбоку карты и доступен только хосту. Так что вся ответственность за запись возлагается на хост.

Защита паролём

Пароль, замыкающий доступ к карте, хранится в 128-бит регистре PWD, а его длина в 8-бит регистре PWD_LEN. Они энергонезависимые. Замкнутая карта отвечает и выполняет сброс инициализацию, выбор и выдаёт состояние, но без доступа к данным карты. При установленном пароле (ненулевая PWD_LEN) после подачи питания карта замыкается автоматически. Как и команды записи регистров CSD и CID, команды замыкания/размыкания доступны только в состоянии Передачи. Здесь они у них нет аргумента адреса и выбрать карту нужно заранее. Они имеют структуру и тип передачи по шине регулярной команды записи блока. Вся требуемая информация (режим установки пароля, сам пароль и замыкание/размыкание) передаётся в блоке данных. Размер блока данных команды определяется хостом до посылки команды. См. *Таблицу 158*.

Установки битов:

- ERASE: стирание. остальные биты должны быть нулевыми, посылается только бит команды
- LOCK_UNLOCK: замок карты. LOCK_UNLOCK может стоять одновременно с SET_PWD, но не с CLR_PWD
- CLR_PWD: очистка пароля
- SET_PWD: пишет пароль в память
- PWD_LEN: длина пароля в байтах
- PWD: пароль (новый или старый)

Установка пароля

1. Выбрать карту (SELECT/DESELECT_CARD, CMD7), если ещё нет.
2. Задать длину блока посылки (SET_BLOCKLEN, CMD16). Включает 8-бит режима замыкания, 8-бит PWD_LEN, число байтов нового пароля и, при замене пароля, длину старого.
3. Послать LOCK/UNLOCK (CMD42) с блоком данных и 16-бит CRC. Блок данных задаёт режим (SET_PWD = 1), длину (PWD_LEN) и сам пароль (PWD). При замене пароля PWD_LEN задаёт длину обоих паролей, а поле PWD старый и затем новый.
4. При совпадении паролей, новый пароль и его длина пишутся в поля PWD и PWD_LEN. Иначе просто ставится бит ошибки LOCK_UNLOCK_FAILED в регистре состояния.

Замкнуть карту можно сразу при записи пароля, установив бит LOCK_UNLOCK или послав команду замыкания.

Сброс пароля

1. Выбрать карту (SELECT/DESELECT_CARD, CMD7), если ещё нет.
2. Задать длину блока посылки (SET_BLOCKLEN, CMD16). Включает 8-бит режима замыкания и 8-бит PWD_LEN.
3. Послать LOCK/UNLOCK (CMD42) с блоком данных и 16-бит CRC. Блок данных задаёт режим (CLR_PWD = 1), длину (PWD_LEN) и сам пароль (PWD). Бит LOCK_UNLOCK игнорируется.
4. При совпадении паролей, поля PWD и PWD_LEN обнуляются. Иначе просто ставится бит ошибки LOCK_UNLOCK_FAILED в регистре состояния.

Блокировка карты

1. Выбрать карту (SELECT/DESELECT_CARD, CMD7), если ещё нет.
2. Задать длину блока посылки (SET_BLOCKLEN, CMD16). Включает 8-бит режима замыкания (байт 0 в *Таблице 158*.), 8-бит PWD_LEN и число байтов текущего пароля.
3. Послать LOCK/UNLOCK (CMD42) с блоком данных и 16-бит CRC. Блок данных задаёт режим (LOCK_UNLOCK = 1), длину (PWD_LEN) и сам пароль (PWD).
4. При совпадении паролей, карта замыкается и ставится бит CARD_IS_LOCKED в регистре состояния. Иначе просто ставится бит ошибки LOCK_UNLOCK_FAILED в регистре состояния.

Замкнуть карту можно сразу при записи нового пароля, дополнительно установив бит LOCK_UNLOCK режима.

При стоящем пароле (PWD_LEN не 0), карта автоматически замыкается по сбросу питания. Попытка замкнуть уже замкнутую карту или карту без пароля ставит бит ошибки LOCK_UNLOCK_FAILED в регистре состояния.

Разблокировка карты

1. Выбрать карту (SELECT/DESELECT_CARD, CMD7), если ещё нет.
2. Задать длину блока посылки (SET_BLOCKLEN, CMD16). Включает 8-бит режима замыкания (байт 0 в *Таблице 158*.), 8-бит PWD_LEN и число байтов текущего пароля.
3. Послать LOCK/UNLOCK (CMD42) с блоком данных и 16-бит CRC. Блок данных задаёт режим (LOCK_UNLOCK = 1), длину (PWD_LEN) и сам пароль (PWD).
4. При совпадении паролей, карта размыкается и снимется бит CARD_IS_LOCKED в регистре состояния. Иначе просто ставится бит ошибки LOCK_UNLOCK_FAILED в регистре состояния.

Разблокировка действует только в текущем включении питания. При стоящем пароле (PWD_LEN не 0), карта автоматически замыкается по сбросу питания. Попытка разомкнуть уже открытую карту ставит бит ошибки LOCK_UNLOCK_FAILED в регистре состояния.

Насильственное стирание

Если пароль забыт напрочь, то использовать карту можно только после полного стирания её содержимого. Для этого надо:

1. Выбрать карту (SELECT/DESELECT_CARD, CMD7), если ещё нет.
2. Задать длину блока посылки (SET_BLOCKLEN, CMD16) 1 байт для команды замыкания 8-бит карт (байт 0 в *Таблице 158*.).
3. Послать LOCK/UNLOCK (CMD42) с байтом данных режима (ERASE = 1) и 16-бит CRC. Остальные биты байта равны нулю.
4. При единственном стоящем бите ERASE в поле данных стирается вся информация карты, включая регистры PWD и PWD_LEN, и она разблокируется. Если стоят и другие биты, то просто ставится бит ошибки LOCK_UNLOCK_FAILED в регистре состояния, карта остаётся нетронутой и блокированной.

Попытка стирания открытой карты ставит бит ошибки LOCK_UNLOCK_FAILED в регистре состояния.

22.4.11.Регистр состояния карты

32-бит поле состояния карты формата R1 относится к предыдущей команде (если не сказано иное). В *Таблице 145*. используются следующие аббревиатуры:

Тип:

- E: бит ошибки
- S: бит состояния
- R: обнаружено и установлено для этого ответа
- X: обнаружено и установлено при исполнении команды. Для чтения этих битов хост SDIO должен опрашивать карту командой.

Чистка:

- A: в соответствии с текущим состоянием карты
- B: всегда относится к предыдущей команде. Приём правильной команды чистит их (с задержкой в одну команду)
- C: чистка чтением

Таблица 145. Состояние карты

Биты	Имя	Тип	Значение	Описание	Чистка
31	ADDRESS_ OUT_OF_RANGE	ER X	0= чисто 1= ошибка	Аргумент адреса в команде не годится для этой карты. Очередное чтение/запись блока или потока выходит за допустимые пределы для карты.	C
30	ADDRESS_MISALIGN	-	0= чисто 1= ошибка	Адрес первого блока не выравнен на границу физического блока карты. Очередное чтение/запись блока не выравнено на границу физического блока карты.	C
29	BLOCK_LEN_ERROR	-	0= чисто 1= ошибка	Один из аргументов команды SET_BLOCKLEN превышает допустимое значение для карты или ранее заданная длина блока недопустима для команды (например, при запрещённой записи частями, длина блока меньше длины блока карты)	C
28	ERASE_SEQ_ERROR	-	0= чисто 1= ошибка	Ошибка последовательности команд стирания.	C
27	ERASE_PARAM	EX	0= чисто 1= ошибка	Недопустимый выбор групп стирания.	C
26	WP_VIOLATION	EX	0= чисто 1= ошибка	Попытка записи в защищённый блок.	C
25	CARD_IS_LOCKED	SR	0 =открыто 1 =закрыто	Карта замкнута хостом.	A
24	LOCK_UNLOCK_ FAILED	EX	0= чисто 1= ошибка	Ошибка последовательности или пароля команд блокировки/разблокировки карты.	C
23	COM_CRC_ERROR	ER	0= чисто 1= ошибка	Сбой CRC предыдущей команды.	B
22	ILLEGAL_COMMAND	ER	0= чисто 1= ошибка	Команда недопустима с таким состоянием карты.	B
21	CARD_ECC_FAILED	EX	0= успех 1= сбой	Неисправимая ошибка ECC данных карты.	C
20	CC_ERROR	ER	0= чисто 1= ошибка	Ошибка карты, не относящаяся к командам (в стандарт нет).	C
19	ERROR	EX	0= чисто 1= ошибка	Ошибка карты, относящаяся к командам или при выполнении последней команды (в стандарт нет).	C
18	Резерв				
17	Резерв				
16	CID/CSD_OVERWRITE	EX	0= чисто 1= ошибка	Одна из ошибок: – Регистр CID уже записан – Неизменяемая часть CSD не соответствует данным карты – Попытка реверса копии или постоянно защищённых битов	C
15	WP_ERASE_SKIP	EX	0=всё 1=часть	Стирание только незащищённых частей памяти.	C

Биты	Имя	Тип	Значение	Описание	Чистка
14	CARD_ECC_DISABLED	SX	0= выкл. 1= вкл.	Команда выполнилась без внутренней коррекции данных (ECC).	A
13	ERASE_RESET	-	0= нету 1= стоит	Сброс последовательности стирания из-за неверной последовательности команд (не CMD35, CMD36, CMD38 или CMD13)	C
12:9	CURRENT_STATE	SR	0 = Idle 1 = Ready 2 = Ident 3 = Stby 4 = Tran 5 = Data 6 = Rcv 7 = Prg 8 = Dis 9 = Btst 10-15 = резерв	Состояние карты при приёме команды. Хост увидит изменение состояния на следующей команде.	B
8	READY_FOR_DATA	SR	0= не готов 1 = готов	Соответствует пустоте буфера.	-
7	SWITCH_ERROR	EX	0= чисто 1= ошибка	По команде SWITCH карта не перешла в требуемый режим	B
6	Резерв				
5	APP_CMD	SR	0 = Выкл. 1 = Вкл.	Карта подождёт ACMD или команда воспринималась как ACMD.	C
4	Резерв для карт SD I/O				
3	AKE_SEQ_ERROR	ER	0= чисто 1= ошибка	Ошибка последовательности аутентификации.	C
2	Резерв для особых команд.				
1	Резерв для тестов производителя.				
0					

22.4.12.Регистр состояния SD

Содержит собственную информацию SD карт памяти одним блоком из 512 бит. Передаётся хосту SDIO по команде ACMD13 (CMD55 и за ней CMD13). ACMD13 посылается только в режиме Передачи (карта выбрана).

В *Таблице 146*. используются следующие аббревиатуры:

Тип:

- E: бит ошибки
- S: бит состояния
- R: обнаружено и установлено для этого ответа
- X: обнаружено и установлено при исполнении команды. Для чтения этих битов хост SDIO должен опрашивать карту командой.

Чистка:

- A: в соответствии с текущим состоянием карты
- B: всегда относится к предыдущей команде. Приём правильной команды чистит их (с задержкой в одну команду)
- C: чистка чтением

Таблица 146. Состояние карты SD

Биты	Имя	Тип	Значение	Описание	Чистка
511:510	DAT_BUS_WIDTH	SR	00’= 1 бит (по умолчанию) ‘01’= резерв ‘10’= 4 бита ‘11’= резерв	Shows the currently defined databus width that was defined by SET_BUS_WIDTH command	A
509	SECURED_MODE	SR	0’= Не в режиме ’1’= В безопасном режиме	Card is in Secured Mode of operation (refer to the “SD Security Specification”).	A
508:496	Резерв.				
495:480	SD_CARD_TYPE	SR	’00xxh’= SD Memory Card (’x’= не волнуют). Сейчас есть: ’0000’= Регулярная SD RD/WR. ’0001’= SD ROM	In the future, the 8 LSBs will be used to define different variations of an SD memory card (each bit will define different SD types). The 8 MSBs will be used to define SD Cards that do not comply with current SD physical layer specification.	A
479:448	SIZE_OF_PROTECT ED_AREA	SR	Размер защищённой области (См. ниже)	(См. ниже)	A
447:440	SPEED_CLASS	SR	Класс скорости. (См. ниже)	(См. ниже)	A
439:432	PERFORMANCE_MOVE	SR	Скорость перемещения с шагом [MB/s]. (См. ниже)	(См. ниже)	A
431:428	AU_SIZE	SR	Размер AU (См. ниже)	(См. ниже)	A
427:424	Резерв.				
423:408	ERASE_SIZE	SR	Число стираемых за раз AU	(См. ниже)	A
407:402	ERASE_TIMEOUT	SR	Таймаут стирания областей из UNIT_OF_ERASE_AU	(См. ниже)	A
401:400	ERASE_OFFSET	SR	Фиксированная добавка к времени стирания.	(См. ниже)	A
399:312	Резерв.				
311:0	Резерв для производителя.				

SIZE_OF_PROTECTED_AREA

У карт стандартной плотности ёмкость защищённой области задаётся как:

Защищённая область = `SIZE_OF_PROTECTED_AREA * MULT*BLOCK_LEN`.

`SIZE_OF_PROTECTED_AREA` определяется в единицах `MULT*BLOCK_LEN`.

У карт высокой плотности ёмкость защищённой области задаётся как:

Защищённая область = `SIZE_OF_PROTECTED_AREA`

`SIZE_OF_PROTECTED_AREA` определяется в байтах.

SPEED_CLASS

Это 8-бит поле, указывающее класс скорости устройства и определяемое как $P_W/2$ (где P_W это производительность записи).

Таблица 147. Класс скорости

SPEED_CLASS	Определение
00h	Class 0
01h	Class 2
02h	Class 4
03h	Class 6
04h – FFh	Резерв

PERFORMANCE_MOVE

Это 8-бит поле показывает P_M (производительность пересылки) с шагом 1 [MB/sec]. Если карта не использует блоки записи (RU), то P_M должно рассматриваться как бесконечность.

PERFORMANCE_MOVE	Определение
00h	Не определено
01h	1 [MB/sec]
02h	02h 2 [MB/sec]
---	---
FEh	254 [MB/sec]
FFh	Бесконечность

AU_SIZE

Это 4-бит поле показывает размер AU как степень 2 от 16 KB.

AU_SIZE	Определение
00h	Not defined
01h	16 KB
02h	32 KB
03h	64 KB
04h	128 KB
05h	256 KB
06h	512 KB
07h	1 MB
08h	2 MB
09h	4 MB
Ah – Fh	Резерв

Максимальный размер AU зависит от ёмкости карты. Карта может содержать любой размер AU от размера RU до максимального AU.

Ёмкость	16-64 MB	128-256 MB	512 MB	1-32 GB
Макс. размер AU	512 KB	1 MB	2 MB	4 MB

ERASE_SIZE

Это 16-бит поле с N_{ERASE}. После стирания N_{ERASE} числа AU берётся таймаут из ERASE_TIMEOUT. Хост должен определять правильное число стираемых за одну операцию AU, чтобы правильно показывать прогресс стирания. Если поле равно 0, то таймаут не вычисляется.

ERASE_SIZE	Определение
0000h	Таймаут не вычисляется.
0001h	1 AU
0002h	2 AU
0003h	3 AU
---------	---------
FFFFh	65535 AU

ERASE_TIMEOUT

Это 6-поле содержит T_{ERASE}, значение таймаута от смещения после стирания ERASE_SIZE AU. Диапазон ERASE_TIMEOUT определяет до 63 секунд. Производитель может выбрать любую комбинацию ERASE_SIZE и ERASE_TIMEOUT. Определение ERASE_TIMEOUT определяет ERASE_SIZE.

ERASE_TIMEOUT	Определение
0	Таймаут не вычисляется.
1	1 [sec]
2	2 [sec]
3	3 [sec]
---------	---------
63	63 [sec]

ERASE_OFFSET

Это 2-бит поле содержит T_{OFFSET}. Бессмысленно при нулевых ERASE_SIZE и ERASE_TIMEOUT.

ERASE_OFFSET	Определение
0h	0 [sec]
1h	1 [sec]
2h	2 [sec]
3h	3 [sec]

22.4.13.Режим SD I/O

Прерывание SD I/O

Карта SD I/O может прерывать модуль MultiMediaCard/SD с помощью ножки 8 (SDIO_D1) SD интерфейса при работе в 4-бит режиме SD. Эта функция необязательная для карт и их функций. Прерывание SD I/O чувствительно по уровню, то есть линия прерывания должна удерживаться низкой до её распознавания о обработки модулем MultiMediaCard/SD или истечения периода прерывания. После обработки прерывания модуль MultiMediaCard/SD операцией записи снимает соответствующий бит состояния внутреннем регистре карты SD I/O. Все линии данных (SDIO_D[3:0]) должны иметь внешние резисторы подпорки. Модуль MultiMediaCard/SD передаёт уровень 8 ножки (SDIO_D/IRQ) на детектор прерывания только во время периода прерывания, всё остальное время он на неё плюёт.

Период прерывания работает при работе памяти и I/O. При работе с одним блоком и множественными передачами период прерывания определяется по разному.

Задержка и восстановление SD I/O

При работе с некоторыми многофункциональными картами модуль MMC/SD может временно приостанавливать передачу данных и освобождать шину для обслуживания более приоритетных функций. Работа возобновляется с прерванного места. Для этого модуль MMC/SD:

1. Определяет функцию, использующую линии SDIO_D [3:0]
2. Требует задержки низкоприоритетной или медленной передачи
3. Ждёт завершения операции задержки
4. Начинает высокоприоритетную передачу
5. Ждёт её завершения
6. Восстанавливает задержанную передачу

SD I/O ReadWait

Необязательная операция ReadWait (RW) (IO_RW_EXTENDED, CMD53) определена только в 1-бит и 4-бит режимах SD. Ей модуль MMC/SD говорит карте, что это чтение нескольких регистров должно приостановить передачу данных карты для посылки команд другим функциям карты SD I/O. Поддержка протокола ReadWait отмечена в регистрах карты. Времянка ReadWait основана на периоде прерывания.

22.4.14.Команды и ответы

Общие и специфичные команды

В стандарте определены два типа команд: общие (GEN_CMD) и специфичные для конкретной системы (ACMD).

Команда ACMD предваряется посылкой команды APP_CMD (CMD55) и имеет структуру регулярной команды MultiMediaCard и может иметь тот же номер CMD. Если после APP_CMD (CMD55) приходит не ACMD, то выполняется обычная команда. Например, если у карты есть определение SD_STATUS (ACMD13), и она приходит сразу после APP_CMD (CMD55), то выполняется SD_STATUS (ACMD13). Но, если после APP_CMD (CMD55) приходит CMD7, а определения для ACMD7 нет, то выполняется стандартная (SELECT/DESELECT_CARD) CMD7.

Для выполнения ACMD карты SD хост:

1. Шлёт APP_CMD (CMD55)

   Карта отвечает, что бит APP_CMD стоит и ожидается ACMD.

2. Шлёт нужную ACMD

   Карта отвечает, что бит APP_CMD стоит и принята ACMD. Если послана не ACMD, то она выполняется как обычная и бит APP_CMD остаётся чистым.

   Присланная недопустимая команда (ни ACMD, ни CMD) обрабатывается обычным путём.

Передача по шине для GEN_CMD соответствует чтению или записи одного блока (WRITE_BLOCK, CMD24 или READ_SINGLE_BLOCK,CMD17). В этом случае, аргумент определяет направление передачи, а не адрес, и блок данных имеет формат и назначение от производителя.

Перед посылкой GEN_CMD (CMD56) карта должна быть выбрана (в состоянии Передачи state). Размер блока данных определён командой SET_BLOCKLEN (CMD16). Ответ на GEN_CMD (CMD56) выдаётся в формате R1b.

Типы команд

Оба вида команд разделены на 4 типа:

- **широковещательная (BC)**: всем картам, отвечать не нужно.
- **широковещательная с ответом (BCR):** всем картам, ответы принимаются одновременно от всех карт.
- **адресная (point-to-point) (AC):** выбранной карте без передачи данных по линиям SDIO_D.
- **адресная (point-to-point) с передачей данных (ADTC):** выбранной карте с передачей данных по линиям SDIO_D.

Форматы команд

См. *Таблицу 138*.

Команды модуля MultiMediaCard/SD

Таблица 154. Блоковая запись

CMD	Тип	Аргумент	Формат ответа	Имя	Описание
CMD23	ac	[31:16] = 0 [15:0] число блоков	R1	SET_BLOCK_COUNT	Число блоков для передачи по следующей команде.
CMD24	adtc	[31:0] адрес данных	R1	WRITE_BLOCK	Пишет блок длиной из команды SET_BLOCKLEN.
CMD25	adtc	[31:0] адрес данных	R1	WRITE_MULTIPLE_BLOCK	Постоянно пишет блоки до получения команды STOP_TRANSMISSION или исчерпания заданного числа блоков.
CMD26	adtc	[31:0] заполн.	R1	PROGRAM_CID	Пишет регистр идентификации карты. Выдаётся единожды и обычно производителем.
CMD27	adtc	[31:0] заполн.	R1	PROGRAM_CSD	Пишет изменяемые биты CSD.

Таблица 155. Блоковая запись защиты (если есть у карты)

CMD	Тип	Аргумент	Формат ответа	Имя	Описание
CMD28	ac	[31:0] адрес данных	R1b	SET_WRITE_PROT	Ставит бит защиты записи адресованной группы. Свойства защиты описаны в специфичных данных (WP_GRP_SIZE).
CMD29	ac	[31:0] адрес данных	R1b	CLR_WRITE_PROT	Снимает бит защиты записи адресованной группы.
CMD30	adtc	[31:0] адрес данных защиты	R1	SEND_WRITE_PROT	Запрос на отсылку состояния битов защиты.
CMD31	Резерв				

Таблица 156. Команды стирания

CMD	Тип	Аргумент	Формат ответа	Имя	Описание
CMD32 ... CMD34	Резерв. Это команды старых версий MultiMediaCard.				
CMD35	ac	[31:0] адрес данных	R1	ERASE_GROUP_START	Ставит адрес первой группы диапазона стирания.
CMD36	ac	[31:0] адрес данных	R1	ERASE_GROUP_END	Ставит адрес последней группы диапазона стирания.
CMD37	Резерв. Это команды старых версий MultiMediaCard.				
CMD38	ac	[31:0] заполн.	R1	ERASE	Стереть выбранные блоки.

Таблица 157. Команды режима I/O

CMD	Тип	Аргумент	Формат ответа	Имя	Описание
CMD39	ac	[31:16] RCA [15:15] флаг записи [14:8] адрес регистра [7:0] данные	R4	FAST_IO	Чтение/запись 8-бит регистра карты. При стоящем флаге записи пишет в адресованный регистр данные из команды, при чтении данные из адресованного регистра выдаются в ответе R4. В стандарте MultiMediaCard этой команды нет.
CMD40	bcr	[31:0] заполн.	R5	GO_IRQ_STATE	Переключает систему в режим прерывания.
CMD41	Резерв				

Таблица 158. Замок карты

CMD	Тип	Аргумент	Формат ответа	Имя	Описание
CMD42	adtc	[31:0] заполн.	R1b	LOCK_UNLOCK	Ставит/снимает пароль или замок карты. Размер блока данных из команды SET_BLOCK_LEN.
CMD43 ... CMD54	Резерв				

Таблица 159. Специфичные команды

CMD	Тип	Аргумент	Формат ответа	Имя	Описание
CMD55	ac	[31:16] RCA [15:0] заполн.	R1	APP_CMD	Говорит карте, что следующая команда специфичная
CMD56	adtc	[31:1] заполн. [0]: RD/WR	-	-	Чтение/запись блока из/в карту для специфичных команд или общего назначения. Размер блока данных из команды SET_BLOCK_LEN.
CMD57 ... CMD59	Резерв				
CMD60 ... CMD63	Резерв для производителя				

22.5. Форматы ответа

Все ответы посылаются по линии команд SDIO_CMD MCCMD. Передача начинается с левого бита кодового слова ответа. Длина кода зависит от типа ответа.

Ответ начинается с стартового бита (всегда 0) и бита направления передачи (карта = 0). Переменное значение в таблицах обозначено как X. Все ответы, кроме R3, сопровождаются CRC. Командное слово команд завершается конечным битом (всегда 1).

Есть пять типов команд:

22.5.1. R1 (нормальный ответ)

Длина = 48 бит.

Позиция бита	Ширина	Значение	Описание
47	1	0	Бит старта
46	1	0	Бит передачи
[45:40]	6	X	Индекс команды
[39:8]	32	X	Состояние карты
[7:1]	7	X	CRC7
0	1	1	Бит конца

22.5.2. R1b

Идентичен R1 с возможной передачей сигнала занятости по линии данных. Карта может стать занятой после приёма команд, в зависимости от своего состояния перед её приёмом.

22.5.3. R2 (регистры CID, CSD)

Длина = 136 бит. Содержимое регистра CID отсылается в ответ на команды CMD2 и CMD10. Содержимое регистра CSD отсылается в ответ на CMD9. Передаются только биты [127...1] CID и CSD, резервный бит [0] заменяется битом конца ответа. Карта показывает процесс стирания, удерживая низким MCDAT. Стирание может быть долгим и карту можно отключить командой CMD7.

Позиция бита	Ширина	Значение	Описание
135	1	0	Бит старта
134	1	0	Бит передачи
[133:128]	6	111111’	Индекс команды
[127:1]	127	X	Состояние карты
0	1	1	Бит конца

22.5.4. R3 (регистр OCR)

Длина: 48 бит. Состояние регистра OCR отсылается в ответ на команду CMD1. Уровень кодирования: ограниченно окно напряжения = низкий, карта занята = низкий.

Позиция бита	Ширина	Значение	Описание
47	1	0	Бит старта
46	1	0	Бит передачи
[45:40]	6	111111’	Резерв
[39:8]	32	X	Регистр OCR
[7:1]	7	1111111’	Резерв
0	1	1	Бит конца

22.5.5. R4 (Быстрый I/O)

Длина: 48 бит. Поле аргумента содержит RCA адресованной карты, адрес регистра и его данные.

Позиция бита		Ширина	Значение	Описание
47		1	0	Бит старта
46		1	0	Бит передачи
[45:40]		6	100111’	CMD39
[39:8] Аргумент	[31:16]	16	X	RCA
	[15:8]	8	X	Адрес регистра
	[7:0]	8	X	Содержимое регистра
[7:1]		7	X	CRC7
0		1	1	Бит конца

22.5.6. R4b

Только SD I/O: карта SDIO после CMD5 ответит одним R4.

Формат:

Позиция бита	Ширина	Значение	Описание
47	1	0	Бит старта
46	1	0	Бит передачи
[45:40]	6	X	Резерв

Позиция бита		Ширина	Значение	Описание
[39:8] Аргумент	39	16	X	Карта готова
	[38:36]	3	X	Число функций I/O
	35	1	X	Память есть
	[34:32]	3	X	Заполнитель
	[31:8]	24	X	I/O ORC
[7:1]		7	X	Резерв
0		1	1	Бит конца

Команда CMD5 включает I/O часть карты. Она остаётся включённой вплоть до сброса, цикла питания или команды CMD52 с записью в I/O сброс. Причём карты только с памятью SD могут отвечать на CMD5. Они должны ответить *Память есть* = 1 and *Число функций I/O* = 0. Карты памяти SD спецификации 1.0 должны опознавать CMD5 как недопустимую команду и не отвечать на неё. Осведомлённый хост I/O пошлёт CMD5. По полученному ответу R4, если есть, он определит конфигурацию карты.

22.5.7. R5 (запрос прерывания)

Только MultiMediaCard. Длина: 48 бит. Если ответ выдаётся хостом, то поле RCA аргумента будет равно 0x0.

Позиция бита		Ширина	Значение	Описание
47		1	0	Бит старта
46		1	0	Бит передачи
[45:40]		6	101000’	CMD40
[39:8] Аргумент	[31:16]	16	X	RCA [31:16] победившей карты или хоста
	[15:0]	16	X	Не определено. Можно использовать для данных IRQ
[7:1]		7	X	CRC7
0		1	1	Бит конца

22.5.8. R6

Только SD I/O. Нормальный ответ карты памяти на CMD3.

Позиция бита		Ширина	Значение	Описание
47		1	0	Бит старта
46		1	0	Бит передачи
[45:40]		6	101000’	CMD40
[39:8] Аргумент	[31:16]	16	X	RCA [31:16] победившей карты или хоста
	[15:0]	16	X	Не определено. Можно использовать для данных IRQ
[7:1]		7	X	CRC7
0		1	1	Бит конца

У карт только I/O биты состояния [23:8] другие, они содержат:

- Бит [15] COM_CRC_ERROR
- Бит [14] ILLEGAL_COMMAND
- Бит [13] ERROR
- Бит [12:0] Резерв.

22.6. Специфичные операции SDIO I/O

Это:

- SDIO ReadWait сигналом SDIO_D2
- SDIO ReadWait остановкой тактов SDIO_CK
- SDIO задержка/восстановление (чтение и запись)
- SDIO прерывания

SDIO поддерживает эти операции только при стоящем бите SDIO_DCTRL[11], кроме задержки чтения, для которой аппаратная поддержка не нужна.

22.6.1. SDIO I/O ReadWait сигналом SDIO_D2

Интервал ReadWait можно запустить перед приёмом первого блока данных: если путь данных включён (стоит бит SDIO_DCTRL[0]), разрешены специфичные операции (бит SDIO_DCTRL[11]), ReadWait стартует (SDI0_DCTRL[10]=0 и SDI_DCTRL[8]=1) и направление от карты к SDIO (SDIO_DCTRL[1] = 1), то DPSM напрямую идёт из Простоя в ReadWait. В ReadWait DPSM уводит SDIO_D2 в 0 после 2 тактов SDIO_CK. В этом состоянии при установке бита RWSTOP (SDIO_DCTRL[9]) DPSM остаётся в Ожидании ещё на 2 такта SDIO_CK для переключения SDIO_D2 в 1 на 1 такт (в соответствии с спецификацией SDIO). Затем DPSM снова ждёт приёма данных от карты.

DPSM не стартует интервал ReadWait во время приёма блока даже при стоящем бите старта: он запустится после приёма CRC. Для запуска новой операции ReadWait бит RWSTOP надо чистить. Во время интервала ReadWait прерывания SDIO ищут на SDIO_D1.

22.6.2. SDIO ReadWait остановкой тактов SDIO_CK

Если карта SDIO не поддерживает предыдущий метод ReadWait, то его можно добиться остановкой SDIO_CK (SDIO_DCTRL должен стоять как в *Секции 22.6.1*, но SDIO_DCTRL[10] =1): DSPM останавливает SDIO_CK на два такта после конечного бита принимаемого блока и пускает его снова после установки бита старта ReadWait.

После такой остановки SDIO_CK карте можно выдавать команды. Во время интервала ReadWait прерывания SDIO ищут на SDIO_D1.

22.6.3. Задержка/ восстановление SDIO

SDIO может задержать работающую операцию записи в карту. Бит SDIO_CMD[11] показывает CPSM, что операция задержана. CPSM анализирует ответ и при получении от карты ACK он извещает DPSM, который после приёма CRC текущего блока уходит в Простой.

Аппаратура не сохраняет число оставшихся блоков для завершения задержаной операции.

Операция записи может быть задержана программно простым отключением DPSM (SDIO_DCTRL[0]=0) при получении ACK задержанной команды от карты. DPSM идёт в Простой.

Задержка чтения: DPSM ждёт в состоянии Wait_r, тогда как задерживаемая функция посылает полный пакет непосредственно перед остановкой передачи данных. Система продолжает чтение RxFIFO до опустения FIF0 и DPSM автоматически уходит в Простой.

22.6.4. Прерывания SDIO

Прерывания SDIO ищут на линии SDIO_D1 при установке бита SDIO_CMD[11].

22.7. Специфичные операции CE-ATA

Это:

- посылка CE-ATA запрещения сигнала завершения команды

- приём от CE-ATA сигнала завершения команды
- извещение CPU о завершении команды CE-ATA битом состояния/прерыванием.

SDIO использует команду CMD61 CE-ATA, если стоит SDIO_CMD[14].

22.7.1. Запрещение сигнала завершения команды

Запрещение сигнала завершения команды посылается через 8 битовых циклов после приёма **короткого** ответа если бита 'Разрешить завершение CMD' , SDIO_CMD[12], нет, а бит 'Без прерывания', SDIO_CMD[13], стоит.

CPSM Идёт в режим Удержания, пишет в регистр сдвига команды последовательность выключения "00001" и 43 в счётчик команды. Через 8 циклов триггер переводит CPSM в состояние Посылки. При достижении счётчиком команды 48 CPSM переходит в Простой не ожидая ответа.

22.7.2. Разрешение сигнала завершения команды

Если стоят биты 'Разрешить завершение CMD' SDIO_CMD[12] и 'Без прерывания' SDIO_CMD[13], то CPSM ждёт сигнала завершения команды в состоянии Waitcpl.

После приёма '0' на линии CMD, CPSM идёт в Простой. Новый команды можно присылать через 7 битовых циклов. Затем, на последние 5 циклов (вне 7) линия CMD уходит в '1' в двухтактном режиме.

22.7.3. Прерывание CE-ATA

Завершение команды отмечается битом состояния SDIO_STA[23]. Снимается он битом очистки SDIO_ICR[23].

Бит SDIO_STA[23] может генерировать прерывание на любой линии, в зависимости от маски SDIO_MASKx[23].

22.7.4. Прекращение CMD61

Если запрещения сигнала завершения команды не посылалось и нужно прервать CMD61, то надо отключить машину состояния команд. Она идёт в Простой и можно посылать CMD12. В это время Запрещение сигнала завершения команды не посылают.

22.8. Аппаратное управление (HW)

Используется для предотвращения исчерпания (режим TX) и переполнения (режим RX) FIFO.

Надо остановить SDIO_CK и стопорнуть машину состояний SDIO. Пока FIFO не может принимать или передавать данные, они останавливаются. Стопорится только машина состояний (SDIOCLK), интерфейс AHB ещё живёт и можно заполнять или опустошать FIFO.

Работа HW разрешается стоящим битом SDIO_CLKCR[14]. После сброса управление HW отключено.

22.9. Регистры SDIO

32-бит регистры доступны через AHB словами.

22.9.1. Управление питанием SDIO (SDIO_POWER)

Смещение адреса: 0x00

По сбросу: 0x0000 0000

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

— Биты 31:2 Резерв, не трогать.

— Биты 1:0 **PWRCTRL**: Управление питанием.

Текущее функциональное состояние тактов карты:

00: Выкл: тактов нет.

01: Резерв

10: Резерв, подача

11: Вкл: такты есть.

NB: Между двумя записями в этот регистр должно быть не менее семи тактов HCLK.

22.9.2. Управление тактами SDI (SDIO_CLKCR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000

31–15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	HWFC_EN	NEGEDGE	WIDBUS		BYPASS	PWRSAV	CLKEN	CLKDIV							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:2 Резерв, не трогать.

— Бит 14 **HWFC_EN**: Разрешение управления HW

0b: Нельзя

1b: Можно

Сигналы прерываний TXFIFOE и RXFIFOF при разрешённом HW описаны в *Секции 22.9.11*.

— Бит 13 **NEGEDGE:** Выбор фазы SDIO_CK

0b: SDIO_CK выдаётся по переднему фронту SDIOCLK

1b: SDIO_CK выдаётся по заднему фронту SDIOCLK

— Биты 12:11 **WIDBUS:** Ширина шины

00: По умолчанию, 1 бит: SDIO_D0

01: 4-бита: SDIO_D[3:0]

10: 8-бит: SDIO_D[7:0]

— Бит 10 **BYPASS:** Шунт делителя тактов

0: Выкл: SDIOCLK делится на CLKDIV для выдачи SDIO_CK.

1: Вкл: SDIOCLK напрямую идёт на SDIO_CK.

— Бит 9 **PWRSAV**: Экономия питания

Такты SDIO_CK при простое шины:

0: SDIO_CK включены всегда

1: SDIO_CK включены при активной шине

— Бит 8 **CLKEN:** Разрешение тактов

0: SDIO_CK выключены

1: SDIO_CK включены

— Биты 7:0 **CLKDIV:** Делитель тактов

Делитель входных (SDIOCLK) для получения выходных тактов (SDIO_CK):

Частота SDIO_CK = SDIOCLK / [CLKDIV + 2].

NB: При идентификации SD/SDIO и MultiMediaCard частота SDIO_CK должна быть меньше 400 kHz.

Повысить частоту шины можно только после назначения относительных адресов всем картам.

Между двумя записями в этот регистр должно пройти не менее 7 тактов HCLK.

Остановить SDIO_CK можно на время интервала ReadWait SD I/O карт: тогда SDIO_CLKCR не влияет на SDIO_CK.

22.9.3. Регистр аргумента SDIO (SDIO_ARG)

Смещение адреса: 0x08

По сбросу: 0x0000 0000

32-бит аргумент посылаемой карте команды.

31–16	15–0
CMDARG	
rw	rw

— Биты 31:0 **CMDARG:** Аргумент команды

Аргумент пишется в этот регистр до записи кода команды в регистр команды.

22.9.4. Регистр команды SDIO (SDIO_CMD)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000

Регистр SDIO_CMD содержит биты индекса и типа команды. Биты индекса посылаются карте, биты типа управляют машиной состояния пути команды (CPSM).

31 … 15	14	13	12	11	10	9	8	7 … 6	5 … 0
Reserved	CE-ATACMD	nIEN	ENCMDcompl	SDIOSuspend	CPSMEN	WAITPEND	WAITINT	WAITRESP	CMDINDEX
	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:15 Резерв, не трогать.

— Бит 14 **ATACMD:** Команда CE-ATA

Если ATACMD стоит, то CPSM передаёт CMD61.

— Бит 13 **nIEN:** Не-разрешение прерываний

Если равен 0, то прерывания в CE-ATA разрешены.

— Бит 12 **ENCMDcompl:** Разрешение завершения команды

Если стоит, то сигнал завершения команды разрешён.

— Бит 11 **SDIOSuspend:** Задержка SD I/O

Если стоит, то передаётся команда задержки (только карты SDIO).

— Бит 10 **CPSMEN:** Разрешение машины состояния пути команд (CPSM)

Если стоит, то CPSM разрешена.

— Бит 9 **WAITPEND:** CPSM ждёт конца передачи данных (CmdPend внутренний сигнал).

Если стоит, то CPSM ждёт конца передачи данных перед посылкой команды.

— Бит 8 **WAITINT:** CPSM ждёт запроса прерывания

Если стоит, то CPSM выключает таймаут команды и ждёт запроса прерывания.

— Биты 7:6 **WAITRESP**: Ожидание ответа

00: Без ответа, ожидается флаг CMDSENT

01: Короткий ответ, ожидается флаг CMDREND или CCRCFAIL

10: Без ответа, ожидается флаг CMDSENT

11: Длинный ответ, ожидается флаг CMDREND или CCRCFAIL

— Биты 5:0 **CMDINDEX:** Индекс посылаемой команды.

NB: Между двумя записями в этот регистр должно пройти не менее 7 тактов HCLK.

MultiMediaCards посылают два вида ответов: короткий (48 бит) и длинный (136 бит). Карты SD и SD I/O посылают только короткий ответ. Устройства CE-ATA посылают только короткий ответ. Аргумент может меняться по типу ответа: это определяется по типу команды.

22.9.5. Регистр отвечаемой команды SDIO (SDIO_RESPCMD)

Смещение адреса: 0x10

По сбросу: 0x0000 0000

Регистр SDIO_RESPCMD содержит поле индекса команды последнего принятого ответа. Если ответ не содержит поле индекса команды (длинный ответ или OCR), то содержимое RESPCMD неизвестно, хоть и должно содержать 111111b (резервное поле из ответа).

31 … 6	5 … 0
Reserved	RESPCMD
	r

— Биты 31:6 Резерв, не трогать.

— Биты 5:0 **RESPCMD:** Индекс отвечаемой команды.

Только чтение.

22.9.6. Регистры 1..4 ответа SDIO (SDIO_RESPx)

Смещение адреса: (0x10 + (4 × x)); x = 1..4

По сбросу: 0x0000 0000

Регистры SDIO_RESP1/2/3/4 содержат состояние карты из принятого ответа.

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

CARDSTATUSx:

Регистр	Короткий ответ	Длинный ответ
SDIO_RESP1	Биты[31:0]	Биты[127:96]
SDIO_RESP2	Не используется	Биты[95:64]
SDIO_RESP3	Не используется	Биты[63:32]
SDIO_RESP4	Не используется	Биты[31:1]0b

Старший значащий бит состояния принимается первым. Младший бит регистра SDIO_RESP3 всегда 0b.

22.9.7. Регистр таймера данных SDIO (SDIO_DTIMER)

Смещение адреса: 0x24

По сбросу: 0x0000 0000

32-бит регистр SDIO_DTIMER содержит период таймаута данных (DATATIME) в единицах тактов шины карты. Декрементный счётчик грузит это значение при переходе DPSM в состояние Wait_R или Занят. При обнулении счётчика ставится флаг таймаута.

NB: Регистры таймера и длины данных надо писать до записи регистра управления данными.

22.9.8. Регистр длины данных SDIO (SDIO_DLEN)

Смещение адреса: 0x28

По сбросу: 0x0000 0000

32-бит регистр SDIO_DLEN содержит число байтов передаваемых данных (DATALENGTH) в единицах тактов шины карты. Декрементный счётчик грузит это значение при старте передачи.

NB: При блоковой передаче длина передаваемых данных должна быть кратной размеру блока (см. SDIO_DCTRL). Регистры таймера и длины данных надо писать до записи регистра управления данными.

22.9.9. Регистр управления данными SDIO (SDIO_DCTRL)

Смещение адреса: 0x2C

По сбросу: 0x0000 0000

Управляет машиной состояния пути данных (DPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCKSIZE				DMAEN	DTMODE	DTDIR	DTEN
																				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:12 Резерв, не трогать.

— Бит 11 **SDIOEN:** Разрешение функций SD I/O
Если стоит, то DPSM выполняет функции SD I/O.

— Бит 10 **RWMOD:** Режим ReadWait
0: Останов ReadWait по SDIO_D2
1: Работа ReadWait по SDIO_CK

— Бит 9 **RWSTOP:** Останов ReadWait
0: ReadWait активен, если стоит RWSTART
1: Разрешение останова ReadWait при стоящем RWSTART

— Бит 8 **RWSTART:** Запуск ReadWait
Если стоит, то запускается ReadWait.

— Биты 7:4 **DBLOCKSIZE:** Размер блока данных в байтах.
Размер блока = $2^{DBLOCKSIZE}$ байт, значение 1111: резерв.

— Бит 3 **DMAEN:** Разрешение DMA
0: Нельзя. 1: Можно.

— Бит 2 **DTMODE:** Режим передачи данных
0: Блоковая передача
1: Поток или мультибайт SDIO для устройств STM32F10xxx XL-плотности. Поток для устройств STM32F10xxx высокой плотности.

— Бит 1 **DTDIR:** Направление передачи данных
0: От контроллера карте.
1: От карты контроллеру.

— Бит 0 **DTEN:** Разрешение передачи данных
Передача данных начинается при записи 1 в бит DTEN. В зависимости от бита DTDIR, DPSM переходит состояние Wait_S, Wait_R или Readwait если RW Start ставится сразу после начала передачи. После конца передачи чистить его не надо, но для начала новой передачи SDIO_DCTRL нужно обновить.

NB: Между двумя записями в этот регистр должно пройти не менее 7 тактов HCLK.

22.9.10.Счётчик данных SDIO (SDIO_DCOUNT)

Смещение адреса: 0x30

По сбросу: 0x0000 0000

При переходе DPSM из Простоя в состояние Wait_R или Wait_S, регистр SDIO_DCOUNT загружается из SDIO_DLEN. При чтении выдаёт число оставшихся байтов передачи, запись не имеет смысла. При обнулении SDIO_DCOUNT DPSM переходит простой и ставится флаг DATAEND.

31–26	25–0
Reserved	DATACOUNT
r	r

— Биты 31:25 Резерв, не трогать.

— Биты 24:0 **DATACOUNT:** Счётчик данных.

NB: Регистр должно читать при завершённом цикле передачи данных.

22.9.11.Регистр состояния SDIO (SDIO_STA)

Смещение адреса: 0x34

По сбросу: 0x0000 0000

Только читаемый регистр SDIO_STA содержит два типа флагов:

- Статические (биты [23:22,10:0]): они снимаются записью в регистр очистки прерываний (SDIO_ICR)
- Динамические (биты [21:11]): они отражают состояние нижележащей логики (например, флаги заполнения и опорожнения FIFO).

31–24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CEATAEND	SDIOIT	RXDAVL	TXDAVL	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	RXACT	TXACT	CMDACT	DBCKEND	STBITERR	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
Res.	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:24 Резерв, не трогать.

— Бит 23 **CEATAEND:** Принят сигнал CE-ATA завершения команды CMD61

— Бит 22 **SDIOIT:** Принято прерывание SDIO

— Бит 21 **RXDAVL:** В приёмном FIFO есть данные

— Бит 20 **TXDAVL:** В передающем FIFO есть данные

— Бит 19 **RXFIFOE:** Приёмный FIFO пуст

— Бит 18 **TXFIFOE:** Передающий FIFO пуст
При разрешённом HW Flow Control, TXFIFOE ставится при наличии в FIFO двух слов.

— Бит 17 **RXFIFOF:** Приёмный FIFO полон
При разрешённом HW Flow Control, RXFIFOF ставится за 2 слова до заполнения FIFO.
— Бит 16 **TXFIFOF:** Передающий FIFO полон
— Бит 15 **RXFIFOHF:** Приёмный FIFO наполовину полон, осталось не менее 8 слов
— Бит 14 **TXFIFOHE:** Передающий FIFO наполовину пуст, можно записать до 8 слов
— Бит 13 **RXACT:** Данные принимаются
— Бит 12 **TXACT:** Данные передаются
— Бит 11 **CMDACT:** Команда передаётся
— Бит 10 **DBCKEND:** Блок данных послан/принят (CRC проверен)
— Бит 9 **STBITERR:** В широком режиме шины во всех сигналах данных бит старта не найден
— Бит 8 **DATAEND:** Конец данных (SDIDCOUNT обнулился)
— Бит 7 **CMDSENT:** Команда послана (отвэт нэ нада)
— Бит 6 **CMDREND:** Ответ команды принят (CRC проверен)
— Бит 5 **RXOVERR:** Переполнение приёмного FIFO
— Бит 4 **TXUNDERR:** Исчерпание передающего FIFO
— Бит 3 **DTIMEOUT:** Таймаут данных
— Бит 2 **CTIMEOUT:** Таймаут ответа команды (64 такта SDIO_CK).
— Бит 1 **DCRCFAIL:** Блок данных послан/принят (сбой CRC)
— Бит 0 **CCRCFAIL:** Ответ команды принят (сбой CRC)

22.9.12.Регистр очистки прерываний SDIO (SDIO_ICR)

Смещение адреса: 0x38
По сбросу: 0x0000 0000

Только запись. Запись 1 в бит чистит соответствующий бит в регистре SDIO_STA. Запись 0 ничего не изменяет.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CEATAENDC	SDIOITC	Reserved											DBCKENDC	STBITERRC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
								rw	rw												rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.
— Бит 23 **CEATAENDC:** Принят сигнал CE-ATA завершения команды CMD61
— Бит 22 **SDIOITC:** Принято прерывание SDIO
— Биты 21:11 Резерв, не трогать.
— Бит 10 **DBCKENDC:** Блок данных послан/принят (CRC проверен)
— Бит 9 **STBITERRC:** В широком режиме шины во всех сигналах данных бит старта не найден
— Бит 8 **DATAENDC:** Конец данных (SDIDCOUNT обнулился)
— Бит 7 **CMDSENTC:** Команда послана (отвэт нэ нада)
— Бит 6 **CMDRENDC:** Ответ команды принят (CRC проверен)
— Бит 5 **RXOVERRC:** Переполнение приёмного FIFO
— Бит 4 **TXUNDERRC:** Исчерпание передающего FIFO
— Бит 3 **DTIMEOUTC:** Таймаут данных
— Бит 2 **CTIMEOUTC:** Таймаут ответа команды (64 такта SDIO_CK).
— Бит 1 **DCRCFAILC:** Блок данных послан/принят (сбой CRC)
— Бит 0 **CCRCFAILC:** Ответ команды принят (сбой CRC)

22.9.13.Регистр маски прерываний SDIO (SDIO_MASK)

Смещение адреса: 0x3C
По сбросу: 0x0000 0000

Запись 1 в бит разрешает прерывание по стоящему соответствующему биту в регистре SDIO_STA. Запись 0 запрещает прерывание.

31–24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CEATAENDIE	SDIOITIE	RXDAVLIE	TXDAVLIE	RXFIFOEIE	TXFIFOEIE	RXFIFOFIE	TXFIFOFIE	RXFIFOHFIE	TXFIFOHEIE	RXACTIE	TXACTIE	CMDACTIE	DBCKENDIE	STBITERRIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.
— Бит 23 **CEATAENDIE:** Принят сигнал CE-ATA завершения команды CMD61
— Бит 22 **SDIOITIE:** Принято прерывание SDIO
— Бит 21 **RXDAVLIE:** В приёмном FIFO есть данные
— Бит 20 **TXDAVLIE:** В передающем FIFO есть данные
— Бит 19 **RXFIFOEIE:** Приёмный FIFO пуст
— Бит 18 **TXFIFOEIE:** Передающий FIFO пуст
— Бит 17 **RXFIFOFIE:** Приёмный FIFO полон
— Бит 16 **TXFIFOFIE:** Передающий FIFO полон
— Бит 15 **RXFIFOHFIE:** Приёмный FIFO наполовину полон, осталось не менее 8 слов
— Бит 14 **TXFIFOHEIE:** Передающий FIFO наполовину пуст, можно записать до 8 слов
— Бит 13 **RXACTIE:** Данные принимаются
— Бит 12 **TXACTIE:** Данные передаются
— Бит 11 **CMDACTIE:** Команда передаётся
— Бит 10 **DBCKENDIE:** Блок данных послан/принят (CRC проверен)
— Бит 9 **STBITERRIE:** В широком режиме шины во всех сигналах данных бит старта не найден
— Бит 8 **DATAENDIE:** Конец данных (SDIDCOUNT обнулился)
— Бит 7 **CMDSENTIE:** Команда послана (отвэт нэ нада)
— Бит 6 **CMDRENDIE:** Ответ команды принят (CRC проверен)
— Бит 5 **RXOVERRIE:** Переполнение приёмного FIFO
— Бит 4 **TXUNDERRIE:** Исчерпание передающего FIFO
— Бит 3 **DTIMEOUTIE:** Таймаут данных
— Бит 2 **CTIMEOUTIE:** Таймаут ответа команды (64 такта SDIO_CK).
— Бит 1 **DCRCFAILIE:** Блок данных послан/принят (сбой CRC)
— Бит 0 **CCRCFAILIE:** Ответ команды принят (сбой CRC)

22.9.14.Счётчик FIFO SDIO (SDIO_FIFOCNT)

Смещение адреса: 0x48

По сбросу: 0x0000 0000

Регистр SDIO_FIFOCNT содержит количество оставшихся слов для записи или чтения из FIFO. Загружается из регистра SDIO_DLEN при установке бита DTEN в регистре SDIO_DCTRL и DPSM в режиме Простоя. Если длина данных не кратна 4, то остаток (1-3 байта) признаётся словом.

31–24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	FIFOCOUNT																							
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:24 Резерв, не трогать.
— Биты 23:0 **FIFOCOUNT:** Число оставшихся слов для записи или чтения из FIFO.

22.9.15.Регистры данных FIFO SDIO (SDIO_FIFO)

Смещение адреса: 0x80

По сбросу: 0x0000 0000

Оба FIFO (приёмный и передающий) доступны как 32 смежных регистра по 32 бита по адресу от SDIO base + 0x080 до SDIO base + 0xFC. Так CPU может обращаться к ним командами с множеством операндов.

22.9.16.Карта регистров SDIO

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SDIO_POWER	Reserved																														PWRCTRL	
0x04	SDIO_CLKCR	Reserved																	HWFC_EN	NEGEDGE	WIDBUS		BYPASS	PWRSAV	CLKEN	CLKDIV							
0x08	SDIO_ARG	CMDARG																															
0x0C	SDIO_CMD	Reserved																	CE-ATACMD	nIEN	ENCMDcompl	SDIOSuspend	CPSMEN	WAITPEND	WAITINT	WAITRESP	CMDINDEX						
0x10	SDIO_RESPC-MD	Reserved																									RESPCMD						
0x14	SDIO_RESP1	CARDSTATUS1																															
0x18	SDIO_RESP2	CARDSTATUS2																															
0x1C	SDIO_RESP3	CARDSTATUS3																															
0x20	SDIO_RESP4	CARDSTATUS4																															
0x24	SDIO_DTIM-ER	DATATIME																															
0x28	SDIO_DLEN	Reserved								DATALENGTH																							
0x2C	SDIO_DCTRL	Reserved																				SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCKSIZE			DMAEN	DTMODE	DTDIR	DTEN
0x30	SDIO_D-COUNT	Reserved								DATACOUNT																							
0x34	SDIO_STA	Reserved										CEATAEND	SDIOIT	RXDAVL	TXDAVL	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	RXACT	TXACT	CMDACT	DBCKEND	STBITERR	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
0x38	SDIO_ICR	Reserved										CEATAENDC	SDIOITC	Reserved											DBCKENDC	STBITERRC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
0x3C	SDIO_MASK	Reserved										CEATAENDIE	SDIOITIE	RXDAVLIE	TXDAVLIE	RXFIFOEIE	TXFIFOEIE	RXFIFOFIE	TXFIFOFIE	RXFIFOHFIE	TXFIFOHEIE	RXACTIE	TXACTIE	CMDACTIE	DBCKENDIE	STBITERRIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
0x48	SDIO_FIFO-CNT	Reserved								FIFOCOUNT																							
0x80	SDIO_FIFO	FIF0Data																															

23. Полноскоростной интерфейс USB

23.1. Введение в USB

Блок управления (БУ) USB это интерфейс между полноскоростной шиной USB 2.0 и шиной APB1.

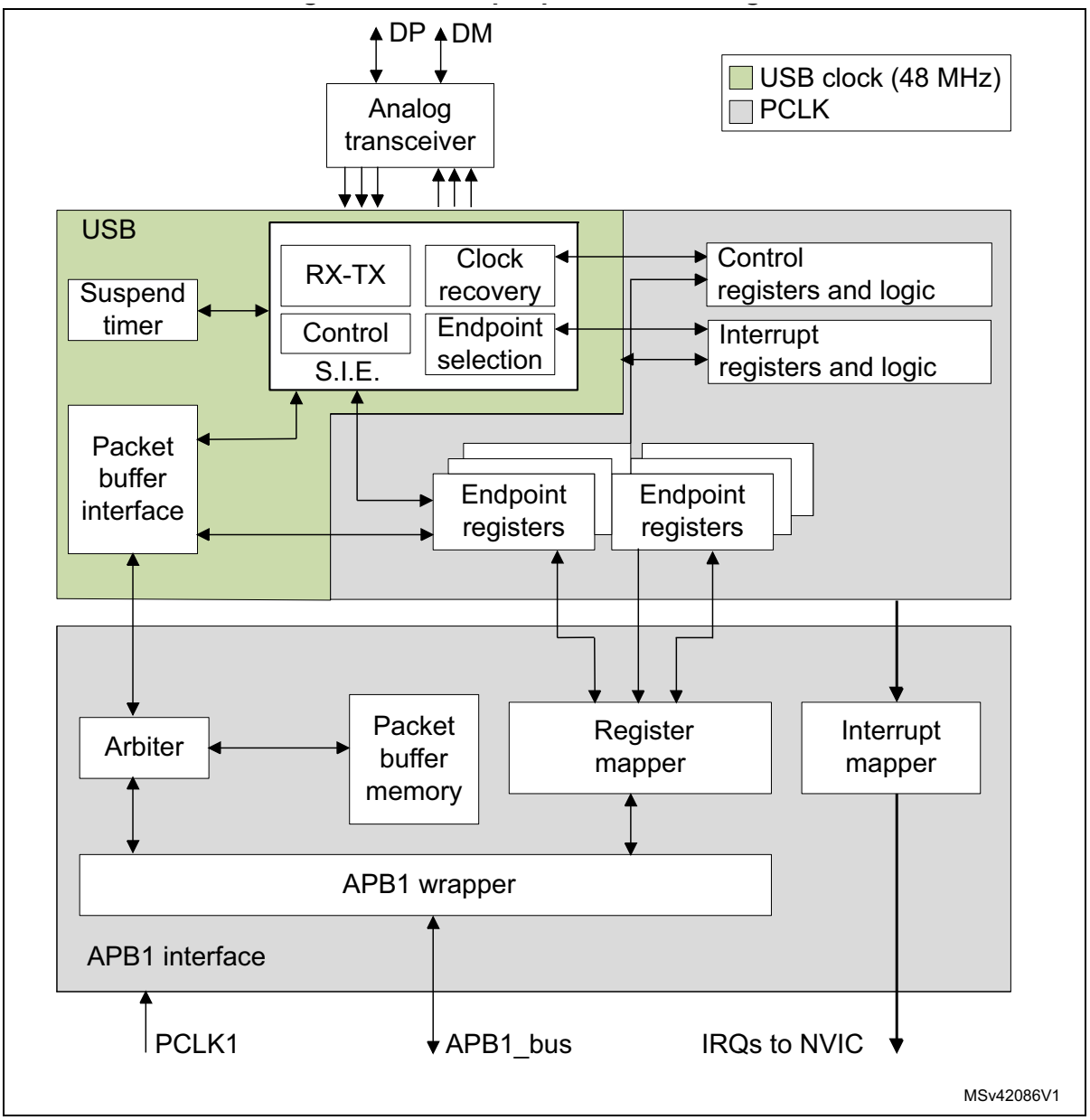
Остановка/восстановление USB останавливает тактирование устройств во имя экономии электроэнергии на благо окружающей среды.

23.2. Основные свойства USB

- Совместимость со спецификацией полноскоростной USB 2.0
- Число конечных точек от 1 до 8
- Генерация/проверка CRC, кодирование/декодирование NRZI и бит-заполнение
- Поддержка изохронных передач
- Поддержка двухбуферных пакетных/изохронных конечных точек
- Операции остановки/восстановления USB
- Генерация фиксированных тактов фреймов

В устройствах разной плотности USB и CAN совместно используют выделенные 512 байт SRAM и не могут использовать её параллельно. USB и CAN могут работать в одной системе, но не одновременно.

23.3. Функциональное описание USB



Для передачи данных между хостом PC и MCU выделяются буферы пакетов, напрямую доступные БУ USB. Размер памяти буферов должен соответствовать числу конечных точек и размеру пакетов. Для 16 однонаправленных или 8 двунаправленных конечных точек отводится до 512 байт. Форматирование передач, включая вычисление и проверку CRC выполняется аппаратно.

Каждая конечная точка связана с блоком описания буфера, показывающим адрес памяти, её размер или длину передачи. Когда БУ USB распознаёт допустимый токен пары функция/конечная

точка, он начинает запрошенную передачу сконфигурированной конечной точки. Данные в/из памяти проходят через внутренний 16-бит регистр БУ USB. После передачи всех данных, если надо, то генерируется или ожидается нужный пакет протокола, в зависимости от направления передачи.

В конце передачи выдаётся прерывание конечной точки. MCU может определить:

- Обслуживаемую конечную точку
- В случае ошибки (битовое заполнение, формат, CRC, протокол, отсутствие ACK, исчерпание/переполнение и пр.) тип прошедшей передачи.

Для изохронных и групповых передач поддерживается использование двух сменных буферов.

Через регистр управления можно поставить блок в высокоэкономичный, малопотребляющий режим (SUSPEND). В это время статическое рассеяние энергии снижается, а тактирование USB можно замедлить или остановить. Активность на входах USB пробуждает устройство асинхронно. К линии побудки можно подключить специальный источник прерывания, немедленно возобновляющий тактирование системы.

23.3.1. Описание блоков USB

БУ USB содержит следующие блоки:

- Машина последовательного интерфейса (**SIE**): Она выполняет: распознавание паттерна синхронизации, битовое заполнение, вычисление и сверку CRC, выдачу/сравнение PID и квитирование связи. Она работает с приёмопередатчиками USB и буферами пакетов в памяти. Также она генерирует сигналы протокола USB, вроде Начала фрейма (SOF), Сброса USB, Ошибки данных и т. д. и события конечных точек, типа конца передачи, правильного приёма пакета; эти сигналы используются для выдачи прерываний.
- Таймер: Он выдаёт фиксированный импульс Начала-Фрейма и обнаруживает глобальную задержку (от хоста) при отсутствии передачи более 3 ms.
- Интерфейс буфера пакетов: Он работает с буферами приёма и передачи по запросам от SIE и размещением в памяти из регистров конечных точек. Адрес инкрементируется после каждого слова до конца пакета, отслеживает число переданных байтов и предупреждает переполнение буферов.
- Регистры конечных точек: 8 регистров содержат тип точек и их текущее состояние. Один регистр может использоваться для двух однонаправленных однобуферных точек. Итого, до 16 однонаправленных однобуферных точек или до 7 двухбуферных* в любой комбинации. Например, 4 двухбуферных и 8 однонаправленных однобуферных точек.
- Регистры управления: Они содержат информацию о всём БУ USB и используются для выдачи некоторых событий USB, вроде восстановления и выключения питания.
- Регистры прерывания: Они содержат маски прерываний и запись событий. Используются для определения причины прерывания, его состояния и очистки удерживаемых прерываний.

NB: * Конечная точка 0 всегда используется для управляющих передач в однобуферном режиме.

БУ USB подключён к шине APB1 через блоки:

- Память пакетов: Эта локальная память буферов пакетов, в которых создаются структуры пакетов. Доступна программно. Содержит 256 слов по 16 бит (512 байт).
- Арбитр: Принимает запросы от шины APB1 и интерфейса USB. Конфликты разрешаются приоритетностью доступов APB1, оставляя половину полосы пропускания памяти для завершения передач USB. Реализован как дуплексный канал с временным разделением, виртуальная двухпортовая SRAM с доступом к памяти от APB1 во время передач USB. Разрешены многословные передачи APB1.
- Картирование регистров: Собирает разнообразные битовые и байтовые регистры БУ USB в набор 16-бит слов, доступных APB1.
- Упаковщик APB1: Даёт доступ APB1 к памяти и регистрам. Отображает весь БУ USB в адресное пространство APB1.
- Коммутатор прерываний: Отображает возможные события USB на три линии NVIC:
  - Низкоприоритетное прерывание USB (Channel 20): Запускается всеми событиями USB (Правильная передача, Сброс USB, и т.д.). Перед обработкой источник надо проверять.

- Высокоприоритетное прерывание USB (Channel 19): Запускается только правильной изохронной и двухбуферной передачей.
- Прерывание побудки USB (Channel 42): Запускается событием побудки их режима Приостановки USB.

23.4. Обзор программирования

23.4.1. Общие вопросы

Здесь основные задачи программ обработки наиболее общих событий USB, двухбуферных конечных точек и изохронных передач. Кроме системного сброса, все действия инициируются БУ USB.

23.4.2. Сбросы: системы и включения питания

После сброса системы и включения питания сначала нужно подать всё тактирование на БУ USB и затем снять сигналы сброса для доступа к регистрам.

Первым делом надо активировать такты макроячейки регистров и снять её сигналы сброса.

После этого нужно включить питание аналоговой части приёмопередатчиков USB битом PDWN регистра CNTR, что требует особой заботы. Перед снятием сигналов сброса нужно подождать установленное время ($t_{STARTUP}$), во время которого поведение приёмопередатчиков USB не определено. После этого можно снять сброс цифровой части USB (очисткой бита FRES в регистре CNTR). Теперь очисткой регистра ISTR устраняем удерживаемые ложные прерывания перед включением других функций макроячейки.

После сброса системы MCU должен инициализировать все требуемые регистры и таблицы описания буферов пакетов, дабы БУ USB смог правильно генерировать прерывания и передавать данные. Регистры, не относящиеся к конечным точкам, нужно инициализировать сообразно требованиям системы (разрешённые прерывания, адреса буферов пакетов и пр.). Затем продолжаем шаги как после сброса USB.

Сброс USB (прерывание RESET)

После этого события связь отключена во всех регистрах конечных точек (БУ USB не отвечает ни на один пакет). Теперь надо включить USB функцию 0 (конечная точка тоже 0) установкой бита EF регистра USB_DADDR и записать в регистр EP0R адрес буфера пакетов. Во время перебора USB хост назначает этому устройству уникальный адрес, который должен быть записан в биты ADD[6:0] регистра USB_DADDR, и конфигурирует нужные конечные точки.

После получения прерывания RESET программа должна в течении 10mS после завершения последовательности, вызвавшей прерывание, восстановить конечную точку по умолчанию USB функции 0.

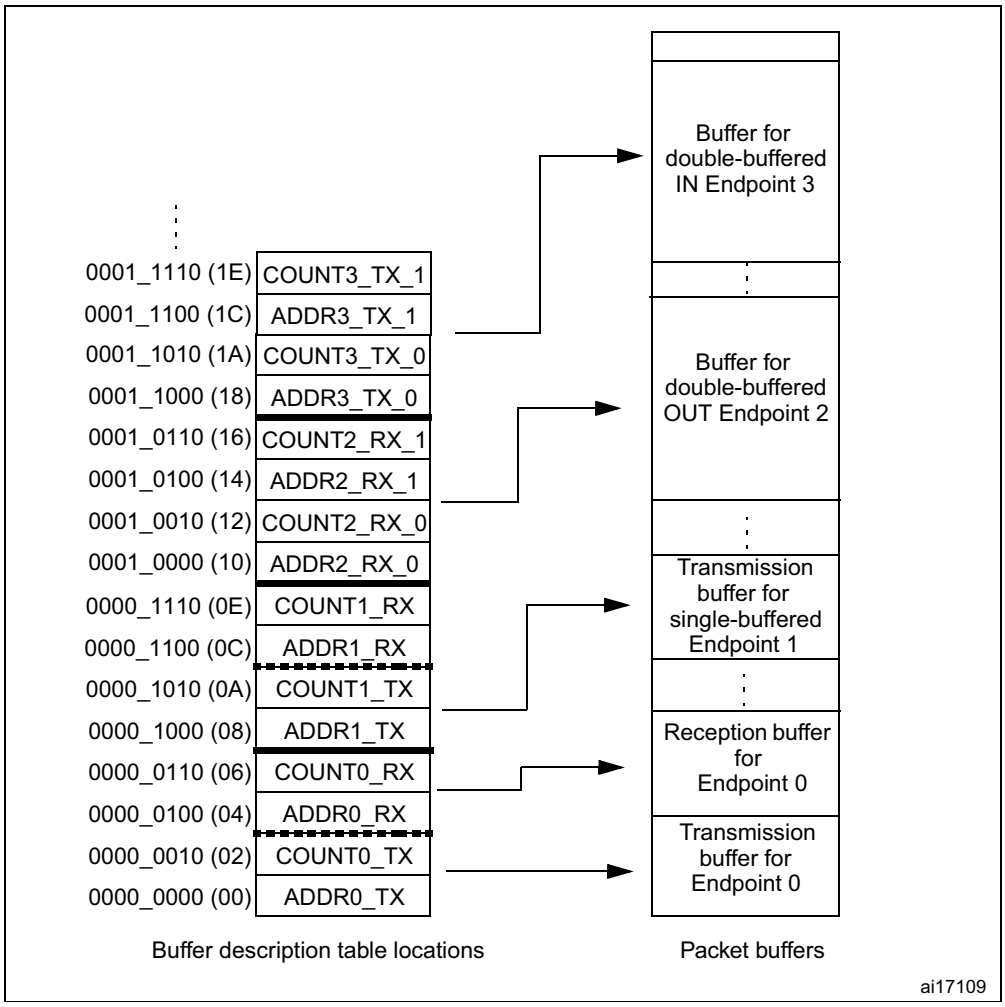
Структура и использование буфера пакетов

Двунаправленная конечная точка может принимать и передавать данные хосту. Передаваемые и принимаемые данные хранятся в своих буферах памяти. Доступ к ним идёт через блок интерфейса буферов пакетов. Логика арбитража выделяет половину цикла APB1 для доступа MCU и половину для БУ USB. Так что память пакетов работает как двухпортовая SRAM, даже при непрерывном доступе MCU. БУ USB использует свои, определённые стандартом USB (48 MHz), обычно отличающиеся от тактов шины APB1. Они могут быть и выше и ниже частоты APB1.

NB: Во избежание переполнения или исчерпания памяти пакетов частота тактов APB1 должна быть больше 8 MHz.

Конечной точке выделяются два буфера пакетов (обычно один для ввода, другой для вывода). Буферы размещаются в памяти пакетов, их адреса и размеры определены в таблице описания буферов, хранящейся тоже там же. Её адрес лежит в регистре USB_BTABLE. Каждая строка таблицы ассоциирована с регистром конечной точки и состоит из четырёх 16-бит слов, так что адрес начала таблицы должен быть выравнен на границу 8 байт (три младших бита регистра USB_BTABLE всегда равны “000”). См. *Секцию 23.4.3*. Для однонаправленных однобуферных не-изохронных конечных точек нужен только один буфер, так что свободное место доступно программе. См. *Секции 23.4.4*. и *23.4.4*.

Рис. 221. Области буферов пакетов и таблица описания буферов



Буфер пакета заполняется начиная с младшего адреса. БУ USB никогда не выходит за границы выделенной памяти буфера.

Инициализация конечной точки

Для начала заполняем регистры буферов ADDRn_TX/ADDRn_RX. Биты EP_TYPE в регистре USB_EPnR ставим в соответствии с типом точки, и, наконец, битом EP_KIND разрешаем специальные функции. На передающей стороне конечная точка включается битами STAT_TX регистра USB_EPnR при заполненном COUNTn_TX. На приёме биты STAT_RX разрешают приём, а COUNTn_RX должен содержать размер буфера, определяемый полями BL_SIZE и NUM_BLOCK. У однонаправленных точек, кроме изохронных и двухбуферных, надо ставить биты и регистры только нужного направления. После включения передачи регистр USB_EPnR и адреса со счётчиками ADDRn_TX/ADDRn_RX, COUNTn_TX/COUNTn_RX трогать нельзя, теперь это дело аппаратуры. После получения прерывания завершения операции CTR в них можно запустить новую операцию.

Пакеты IN (передача данных)

После получения пакета токена IN сравниваются присланные адрес и конечная точка. Если они совпадают с конфигурацией, то БУ USB пишет содержимое ADDRn_TX и COUNTn_TX из строки таблицы описания памяти пакетов во внутренние регистры ADDR и COUNT (они программно недоступны). Из памяти пакетов читается первое передаваемое слово и начинается передача DATA0 или DATA1 PID в соответствии с битом DTOG_TX в USB_EPnR. После завершения PID первый байт прочитанного слова уходит в регистр сдвига для передачи по шине USB. После передачи последнего байта посылается свежевычисленный CRC. Если адресованная конечная точка нерабочая, то вместо данных пересылается пакет NAK или STALL в соответствии с битами STAT_TX регистра USB_EPnR.

Внутренний регистр ADDR это указатель текущего адреса в буфере пакета, а COUNT считает число оставшихся байтов передачи. Слова из памяти пакетов передаются начиная с младшего значащего бита. Из буфера передачи читаются COUNTn_TX/2 слов, начиная с адреса ADDRn_TX. При нечётном количестве байтов в буфере используется младшая часть последнего слова.

По приёму квитка ACK от хоста регистр USB_EPnR обновляется: бит DTOG_TX перекидывается, конечная точка делается нерабочей установкой STAT_TX=10 (NAK) и ставится бит CTR_TX.

Программа сначала должна проверить биты EP_ID и DIR регистра USB_ISTR обратившейся точки. Обработка события CTR_TX начинается очисткой бита прерывания, затем программа готовит новый буфер передачи, обновляет COUNTn_TX в таблице описания и ставит STAT_TX в ‘11 (VALID) для возобновления передачи. Пока биты STAT_TX равны ‘10 (NAK), любой запрос IN к этой точке получает ответ NAK, чтобы хост USB присылал запрос снова вплоть до завершения передачи. Такая последовательность операций обязательна, дабы не терять извещения для следующей передачи IN после поставившей прерывание CTR.

Пакеты OUT и SETUP (приём данных)

Эти два токена обрабатываются БУ USB примерно одинаково, отличие обработки пакетов SETUP приведено ниже. После приёма OUT/SETUP PID рабочей конечной точки БУ USB пишет содержимое ADDRn_RX во внутренний регистр ADDR. Затем сбрасывается COUNT и внутренний 16-бит регистр BUF_COUNT, используемый для контроля переполнения буфера, заполняется на основе полей BL_SIZE и NUM_BLOCK из COUNTn_RX. БУ USB упаковывает принимаемые байты в слова (первый байт идёт младшим) и последовательно пишет их в буфер, начиная с адреса в ADDR, уменьшая BUF_COUNT и увеличивая COUNT на каждый принимаемый байт. При обнаружении конца пакета DATA проверяется CRC и, если всё в порядке, отсылается квиток ACK хосту.

При дурном CRC и других ошибках (нарушение бит-заполнения, ошибка фрейма и пр.), данные остаются в буфере пакетов, по меньшей мере данные до обнаружения ошибки, но квиток ACK не отсылается и ставится бит ERR в регистре USB_ISTR. От программы в этом случае ничего не требуется, БУ USB просто восстанавливает исходное состояние приёма и ждёт новой передачи.

Если адресована нерабочая конечная точка, то вместо ACK отсылается NAK или STALL в соответствии с битами STAT_RX в регистре USB_EPnR и данные в буфер не пишутся.

Приёмный буфер памяти принимает данные, начиная с адреса в ADDRn_RX, числом включая CRC (т.е. длина данных + 2), или до адреса защиты, определённого BL_SIZE и NUM_BLOCK. Если длина данных больше размера буфера, то возникает переполнение буфера и БУ USB вместо квитка ACK отсылает STALL, извещая хост о проблеме. Прерывание не выдаётся и передача считается сбойной.

При добром завершении передачи содержимое внутреннего регистра COUNT пишется назад в COUNTn_RX не трогая поля BL_SIZE и NUM_BLOCK, и обновляется регистр USB_EPnR: бит DTOG_RX перекидывается, конечная точка делается нерабочей установкой STAT_RX = ‘10 (NAK) и ставится бит CTR_RX. При сбойной передаче ничего этого не делается.

Программа сначала должна проверить биты EP_ID и DIR регистра USB_ISTR обратившейся точки. Обработчик события CTR_RX сначала определяет тип передачи (бит SETUP в USB_EPnR), снимает флаг прерывания и читает число принятых байтов из COUNTn_RX. После обработки принятых данных программа ставит биты STAT_RX в ‘11 (VALID) в регистре USB_EPnR, разрешая дальнейшие передачи. Пока биты STAT_RX равны ‘10 (NAK), любой запрос OUT к этой точке получает ответ NAK, чтобы хост USB присылал запрос снова вплоть до завершения передачи. Такая последовательность операций обязательна, дабы не терять извещения следующей передачи OUT после поставившей прерывание CTR.

Управляющие передачи

Управляющая передача это SETUP с необязательными последующими этапами данных одного направления, и стадией состояния (ноль байт противоположного направления). Передачи SETUP обрабатываются только управляющими конечными точками и очень похожи на передачи OUT за исключением того, что биты DTOG_TX и DTOG_RX адресованной точки ставятся в 1 и 0 соответственно, для запуска управляющей передачи, и оба STAT_TX и STAT_RX ставятся в ‘10 (NAK) чтобы программа по содержимому SETUP могла определить, что будет дальше, IN или OUT. Управляющая конечная точка отличает передачи OUT от SETUP по биту SETUP в USB_EPnR при каждом событии CTR_RX. Устройство USB может определить число и направление стадий данных интерпретируя данные стадии SETUP, и должна выдать STALL на передачу в случае ошибки. Для этого, для всех стадий, кроме последней, надо установить STALL для неиспользованного направления, так что если хост изменит направление слишком быстро, то он получит STALL на стадии состояния.

При разрешении последней стадии данных, противоположное направление передачи нужно ставить в NAK, так что, если хост немедленно переворачивает направление передачи (для выполнения стадии состояния), то он продолжает ждать завершения управляющей операции. Если она завершается успешно, то программа изменит NAK на VALID, иначе на STALL. В то же время, если стадия

состояния будет OUT, то надо ставить бит STATUS_OUT (EP_KIND в регистре USB_EPnR), чтобы передача ненулевого состояния вызывала ошибку. При обработке передачи состояния программа чистит бит STATUS_OUT и ставит STAT_RX в VALID (для приёма новой команды) и STAT_TX в NAK (для задержки стадии состояния с немедленной последующей SETUP).

Поскольку спецификация USB утверждает, что на пакет SETUP можно отвечать только ACK, то прерывая текущую команду для запуска новой, логика USB не дозволяет на токен SETUP от хоста отвечать NAK или STALL.

Приняв токен SETUP при битах STAT_RX, равных ‘01 (STALL) или ‘10 (NAK), USB принимает данные, выполняя нужные действия, и отсылает ACK. Если это точка ранее выдала необработанный запрос CTR_RX (бит CTR_RX ещё стоит), то USB отбрасывает передачи SETUP и не отвечает на них, симулируя ошибку приёма, чтобы хост передавал этот SETUP снова. Так предотвращается потеря нотификации нового SETUP при необработанном старом.

23.4.3. Двухбуферные конечные точки

При передаче больших объёмов данных наиболее приемлема групповая (bulk) модель передачи, занимающая практически всё полосу пропускания шины. Для этого занятый обработкой передачи USB на появляющуюся такую же отвечает NAK и хост упрямо повторяет новый запуск. Тут и появляется групповая передача с двумя сменными буферами. Один аппаратно пересылается, а второй программно обрабатывается.

У групповых конечных точек регистр USB_EPnR ставится на однонаправленную передачу и используются все 4 элемента строки таблицы описания буферов пакетов. Из парных битов STAT биты STAT_RX разрешают приём, биты STAT_TX разрешают передачу, незанятая пара должна стоять в ‘00 (Disabled). При необходимости иметь двухбуферную передачу в обоих направлениях используют два регистра USB_EPnR.

При такой передаче конечная точка переключается в состояние NAK только при конфликте буферов между БУ USB и программой, а не в конце каждой успешной передачи.

Используемый БУ USB буфер памяти определяется битом DTOG направления: DTOG_RX (бит 14 в USB_EPnR) для приёма, и ‘DTOG_TX (бит 6 в USB_EPnR) для передачи. Для определения того, какой буфер нужно использовать программе, в регистре USB_EPnR есть бит SW_BUF. В таблицах ниже приведено соответствие битов DTOG/SW_BUF регистра USB_EPnR направлениям групповой передачи и используемым буферам пакетов.

Таблица 169. Флаги двухбуферных операций

Флаг буфера	Конечная точка передачи	Конечная точка приёма
DTOG	DTOG_TX (USB_EPnR бит 6)	DTOG_RX (USB_EPnR бит 14)
SW_BUF	USB_EPnR бит 14	USB_EPnR бит 6

Таблица 170. Использование буферов памяти

Тип точки	DTOG	SW_BUF	Буфер пакетов БУ USB	Буфер пакетов программы
IN	0	1	ADDRn_TX_0 / COUNTn_TX_0.	ADDRn_TX_1 / COUNTn_TX_1.
	1	0	ADDRn_TX_1 / COUNTn_TX_1	ADDRn_TX_0 / COUNTn_TX_0.
	0	0	Нет (1)	ADDRn_TX_0 / COUNTn_TX_0.
	1	1	Нет (1)	ADDRn_TX_0 / COUNTn_TX_0.
OUT	0	1	ADDRn_RX_0 / COUNTn_RX_0.	ADDRn_RX_1 / COUNTn_RX_1.
	1	0	ADDRn_RX_1 / COUNTn_RX_1.	ADDRn_RX_0 / COUNTn_RX_0.
	0	0	Нет (1)	ADDRn_RX_0 / COUNTn_RX_0.
	1	1	Нет (1)	ADDRn_RX_1 / COUNTn_RX_1.

1. Конечная точка в состоянии NAK.

Двухбуферная работа активируется так:

- Делаем конечную точку групповой записью ‘00 в поле EP_TYPE регистра USB_EPnR, и
- Ставим бит EP_KIND в ‘1 (DBL_BUF), там же.

Программа инициализирует биты DTOG и SW_BUF для первой передачи. В конце каждой передачи ставится соответствующий бит CTR_RX или CTR_TX регистра USB_EPnR. Одновременно аппаратно перекидывается соответствующий бит DTOG регистра USB_EPnR. Бит DBL_BUF остаётся стоять. В отличие от общих передач и первой двухбуферной, пара битов STAT остаётся неизменной ‘11 (VALID). Однако, после приёма токена новой передачи, при обнаружении конфликта буферов (одинаковые DTOG и SW_BUF) точка переключается в ‘10 (NAK).

Программа очищает флаг прерывания по событию CTR и приступает к обработке данных законченной передачи. По её завершению, программа записью ‘1 переключает бит SW_BUF, извещая БУ USB о доступности буфера. Число передач с NAK ответом ограничено только скоростью работы программы.

Программа всегда может изменить поведение двухбуферных точек изменяя их состояние на отличное ‘11 (VALID) в паре битов STAT регистра USB_EPnR. В этом случае БУ USB использует записанное состояние.

23.4.4. Изохронные передачи

Стандарт USB полноскоростной периферии поддерживает передачи данных с точной фиксированной частотой ("Изохронные"). Типичный пример: аудио, видео-потоки и прочие, с жёсткими требованиями к частоте передачи. Если конечная точка во время фазы перечисления определена как изохронная, то хост выделяет нужную полосу пропускания фреймов и отправляет только по одному пакету IN или OUT. Повторной передачи для сбойных пакетов не предусмотрено и фазы ACK для передачи данных нет. По этой же причине изохронные передачи не поддерживают переключения передачи данных и они всегда начинаются с DATA0 PID.

Включается изохронный режим записью ‘10 в биты EP_TYPE регистра USB_EPnR; Пары битов STAT_RX/STAT_TX могут быть только ‘00 (DISABLED) и ‘11 (VALID).

Изохронные конечные точки используют двойную буферизацию.

Используемый БУ USB определяется битом DTOG (DTOG_RX для приёма, DTOG_TX для передачи).

Таблица 171. Буферы изохронных передач

Тип точки	DTOG	Буфер пакетов БУ USB	Буфер пакетов программы
IN	0	ADDRn_TX_0 / COUNTn_TX_0.	ADDRn_TX_1 / COUNTn_TX_1.
	1	ADDRn_TX_1 / COUNTn_TX_1.	ADDRn_TX_0 / COUNTn_TX_0.
OUT	0	ADDRn_RX_0 / COUNTn_RX_0.	ADDRn_RX_1 / COUNTn_RX_1.
	1	ADDRn_RX_1 / COUNTn_RX_1.	ADDRn_RX_0 / COUNTn_RX_0.

Для изохронных конечных точек регистр USB_EPnR используется для одного направления передачи. Для двунаправленных передач надо использовать два регистра USB_EPnR.

Программа инициализирует бит DTOG для первой передачи. В конце каждой передачи ставится соответствующий бит CTR_RX или CTR_TX регистра USB_EPnR. Одновременно аппаратно перекидывается соответствующий бит DTOG регистра USB_EPnR. Пара битов STAT остаётся неизменной ‘11 (VALID). Ошибки CRC и переполнение буфера при изохронных передачах OUT рассматриваются как хорошие и событие CTR_RX запускается, но ошибка CRC всегда ставит бит ERR регистра USB_ISTR.

23.4.5. События Остановки/Восстановления

Стандарт USB определяет состояние Остановки (SUSPEND) со средним потреблением от шины USB не более 2.5 mA. В режиме ?остановки? хост PC посылает уведомления всем, молчавшим на шине USB более 3mS: поскольку при нормальной работе пакет SOF должен посылаться каждую 1mS, то БУ USB обнаруживает отсутствие 3 последовательных пакетов SOF и воспринимает этот факт как запрос остановки от хоста PC и ставит бит SUSP в регистре USB_ISTR, вызывая прерывание, если разрешено. Нормальная работа остановленного устройства может возобновиться

последовательностью RESUME, запускаемой как от хоста PC, так и от периферии, но заканчивается всегда хостом PC. Остановленное БУ USB должно обнаруживать последовательность RESET, восстанавливаясь как по обычному сбросу USB.

Конкретная процедура остановки зависит от устройства, а мы даём краткое описание ответа программы на событие SUSP:

1. Ставим бит FSUSP регистра USB_CNTR. Так включается режим останова БУ USB. Теперь приём SOF отключён и повторные прерывания SUSP блокированы на время останова USB.
2. Убираем или снижаем энергопотребление оставшихся блоков системы.
3. Ставим бит LP_MODE в регистре USB_CNTR, отключая аналоговую часть USB, оставив обнаружение действий восстановления.
4. По возможности выключаем внешний генератор и PLL устройства.

При появлении события на шине USB устройство в режиме SUSPEND должно выполнить процедуру RESUME. Процесс пробуждения не должен занимать больше 10mS. Всё начинается с асинхронной очистки бита LP_MODE в регистре USB_CNTR. Использовать прерывание WKUP нужно осторожно из-за задержки восстановления системных тактов, лучше расположить процедуру восстановления сразу после остановки. Тогда восстановление системных тактов запустит процедуру восстановления. Во избежание разрядки ESD или других помех от пробуждения системы (это асинхронное событие), на линиях данных ставят аналоговый фильтр длительностью около 70ns.

Вот список действий:

1. При необходимости включаем внешний генератор и PLL устройства.
2. Чистим бит FSUSP регистра USB_CNTR.
3. Событие побудки смотрят в битах RXDP и RXDM регистра USB_FNR. Конец последовательности останова/побудки находят по комбинации “10” (состояние Простоя) упомянутых битов. Кроме того, в конце последовательности сброса ставится бит RESET регистра USB_ISTR, прерывание от которого нужно обрабатывать как обычно.

Table 172. Обнаружение события побудки

[RXDP,RXDM]	Событие побудки	Действия программы
“00”	Сброс Root	Нет
“10”	Нет (шум на шине)	Возврат в Suspend
“01”	Восстановление Root	Нет
“11”	Нельзя (шум на шине)	Возврат в Suspend

Устройство можно разбудить по событию, не связанному с протоколом USB. Это делается установкой бита RESUME регистра USB_CNTR и снятием через интервал времени от 1 mS до 15 mS (отмерить его можно прерываниями ESOF, идущими через 1mS при нормальной частоте системных тактов). После очистки бита RESUME последовательность побудки завершается хостом PC и конец его опять определяется по битам RXDP и RXDM регистра USB_FNR.

NB: Бит RESUME надо использовать только после останова БУ USB установкой бита FSUSP регистра USB_CNTR.

23.5. Регистры USB

Регистры БУ USB можно разделить на группы:

- Общие регистры: Прерывания и Управления
- Регистры конечных точек: Конфигурация и состояние конечных точек
- Таблица описания буферов: Размещение и размер памяти пакетов

Адреса регистров указаны смещением от базового адреса БУ USB 0x4000 5C00, адрес таблицы описания буферов хранится а регистре USB_BTABLE. Все регистры имеют длину 26 бит, но выравнены на границу 32-бит слов. Адреса буферов в таблице описания имеют то же выравнивание и начинаются с адреса 0x4000 6000. Регистры периферии доступны как полусловами (16-бит), так и словами (32-бит).

23.5.1. Общие регистры

Они влияют на общее поведение БУ USB, определяют режим работы, обработку прерываний, адрес устройства и дают доступ к номеру текущего фрейма, обновлённого хостом.

Регистр управления USB (USB_CNTR)

Смещение адреса: 0x40

По сбросу: 0x0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	Reserved			RESUME	FSUSP	LP_MODE	PDWN	FRES
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

В битах маски:

0: Прерывание X запрещено.

1: Разрешено прерывание X по стоящему соответствующему биту в регистре USB_ISTR.

— Бит 15 **CTRM**: Маска прерывания верной передачи

— Бит 14 **PMAOVRM**: Маска прерывания переполнения/исчерпания памяти пакетов

— Бит 13 **ERRM:** Маска прерывания ошибки ERR

— Бит 12 **WKUPM:** Маска прерывания пробуждения WKUP

— Бит 11 **SUSPM:** Маска прерывания останова SUSP

— Бит 10 **RESETM:** Маска прерывания сброса RESET

— Бит 9 **SOFM:** Маска прерывания старта фрейма SOF

— Бит 8 **ESOFM:** Маска прерывания ожидаемого старта фрейма ESOF

— Биты 7:5 Резерв, не трогать.

— Бит 4 **RESUME:** Запрос Побудки

Установка этого бита посылает сигнал Resume хосту. Он должен выдаваться не раньше 1 mS и не позже 15 mS после чего хост PC готов запустить последовательность побудки на свои конце.

— Бит 3 **FSUSP:** Принудительный станов

Программа должна ставить этот бит при получении прерывания SUSP, выдаваемого БУ USB при отсутствии активности на шине в течении 3 mS.

0: Не влияет.

1: Вход в Останов. Такты и питание аналоговых приёмопередатчиков не трогаются. При питании от шины программа после FSUSP должна поставить бит LP_MODE.

— Бит 2 **LP_MODE:** Экономный режим

В этом режиме отключается питание кроме внешнего резистора подпорки. Ставится, когда программа готова остановить все системные такты или уменьшить их частоту. Активность USB в режиме остановки (событие WKUP) асинхронно сбрасывает этот бит (можно и программно).

0: Никакой экономии.

1: Экономим.

— Бит 1 **PDWN:** Выключение питания

Полностью выключает питание всей аналоговой части БУ USB и отключает её от цифровой части.

0: Включаем.

1: Выключаем.

— Бит 0 **FRES:** Принудительный сброс

0: Снимает сброс USB.

1: Сбрасывает БУ USB прямо как сигнал RESET на шине USB. БУ USB удерживает RESET до снятия сброса. Выдаётся прерывание “USB-RESET”, если разрешено.

Регистр состояния прерываний USB (USB_ISTR)

Смещение адреса: 0x44

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMA OVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	Reserved			DIR	EP_ID[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0				r	r	r	r	r

Содержит состояние всех источников прерываний.

Биты старшей части представляют отдельные события. Они ставятся аппаратно и при стоящем соответствующем бите в регистре USB_CNTR выдаётся прерывание. Обработчик прерывания проверит каждый бит, сделает что надо и снимет обслуженный бит. Не снятый бит считается задержанным и линия прерывания остаётся высокой. Несколько поставленных битов вызывают одно прерывание.

Завершение передачи конечной точки можно обрабатывать по-разному. Бит CTR ставится аппаратно сразу после завершения передачи, вызывая общее прерывание при стоящем соответствующем бите в регистре USB_CNTR. Условие прерывания конечной точки активируется независимо от бита CTRM в регистре USB_CNTR. Оба условия прерывания остаются активными вплоть до программного снятия удерживаемого бита в соответствующем регистре USB_EPnR (бит CTR только читается). Последняя сработавшая конечная точка определяется битом направления (DIR) идентификатором EP_ID.

Относительный приоритет одновременно удерживаемых событий USB_ISTR можно задать порядком опроса битов. Очищать нужно только обработанные события. При завершении обработчика будет запрошено новое прерывание для обслуживания оставшихся событий.

Биты очищаются только программно записью 0, так что весьма полезно стирать командой записи слова и держать нестираемые биты в 1, не то прерывание, возникшее во время цикла *чтение-модификация-запись* (стирание бита), не сможет быть обработано.

Теперь подробно:

— Бит 15 **CTR**: Верная передача

Ставится аппаратно при успешном завершении передачи, конечная точка определяется по битам DIR и EP_ID. Только чтение.

— Бит 14 **PMAOVR**: Переполнение/исчерпание памяти пакетов

Ставится, если MCU не успевает вовремя ответить на запрос памяти USB. Прерывание PMAOVR не должно возникать при нормальных операциях. Сбойные не-изохронные передачи повторяются хостом и программа может подготовиться к следующей передаче. Чтение/запись, но действует только запись 0, запись 1 бессмысленна.

— Бит 13 **ERR:** Ошибка

Возникла одна из ошибок:

NANS: Ответа нет. Истёк таймаут ответа хоста.

CRC: Ошибка CRC принятого токена или данных

BST: Ошибка заполнения в PID, данных и/или CRC.

FVIO: Нарушение формата фрейма (EOP не там, не та последовательность токенов и пр.).

Программой обычно игнорируется, БУ USB и хост PC повторяют сбойную передачу понятным способом. Прерывание полезно при разработке, проверке, наладке программ и аппаратной части. Чтение/запись, но действует только запись 0, запись 1 бессмысленна.

— Бит 12 **WKUP:** Побудка

Ставится аппаратно при появлении активности на шине USB во время останова. Это событие асинхронно чистит бит LP_MODE регистра CTLR и активирует линию USB_WAKEUP, дабы известить других о начале процесса пробуждения. Чтение/запись, но действует только запись 0, запись 1 бессмысленна.

— Бит 11 **SUSP:** Запрос Останова

Ставится аппаратно если шина USB проспала 3mS (это запрос останова). Проверка этого условия включается сразу после сброса, не работает при FSUSP=1 вплоть до конца последовательности пробуждения. Чтение/запись, но действует только запись 0, запись 1 бессмысленна.

— Бит 10 **RESET:** Запрос сброса USB

Ставится когда БУ USB обнаруживает на входах сигнал USB RESET. Тогда БУ USB просто сбрасывает внутреннюю машину состояния протокола и генерирует прерывание, если разрешено битом RESETM в регистре USB_CNTR. Пока стоит, приём и передача запрещены. Регистры конфигурации не сбрасываются, нужно чистить их вручную (подтверждение обработки сброса). Адрес функции и регистры конечных точек сбрасываются событием сброса USB. Чтение/запись, но действует только запись 0, запись 1 бессмысленна.

— Бит 9 **SOF:** Старт фрейма

Ставится при появлении пакета SOF на шине USB. Можно устроить 1mS синхронизацию хосту и безопасно читать регистр USB_FNR, обновляемый приёмом пакета SOF (может пригодиться в изохронных системах). Чтение/запись, но действует только запись 0, запись 1 бессмысленна.

— Бит 8 **ESOF:** Ожидаемый старт фрейма

Ставится аппаратно, если через 1mS не пришёл пакет SOF (может быть хаб не принял его правильно). После трёх последовательных прерываний ESOF выдаётся прерывание SUSP. Бит ставится даже при поедем появлении SOF и ещё не закрытом таймере останова. Чтение/запись, но действует только запись 0, запись 1 бессмысленна.

— Биты 7:5 Резерв, не трогать.

— Бит 4 **DIR:** Направление передачи

Ставится аппаратно при успешном завершении операции, запросившей прерывание.

Если DIR=0, то в регистре USB_EPnR сработавшей конечной точки типа IN стоит бит CTR_TX.

Если DIR=1, то в регистре USB_EPnR сработавшей конечной точки типа OUT стоит бит CTR_RX или оба бита CTR_TX/CTR_RX при двух необработанных направлениях передачи.

Только чтение.

— Биты 3:0 **EP_ID[3:0]:** Идентификатор конечной точки

Аппаратно записанный номер сработавшей конечной точки. При нескольких стоящих запросах пишется более приоритетный номер. Сначала обрабатываются изохронные и двухбуферные групповые точки, затем все остальные. Из точек с одинаковыми установками первыми обрабатываются точки с меньшими номерами регистров. Программа может установить приоритеты точек, назначая им регистры. Только чтение.

Регистр номера фрейма USB (USB_FNR)

Смещение адреса: 0x48

По сбросу: 0x0xxx где x - не определено.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Бит 15 **RXDP**: Состояние линии+ данных

Может использоваться для определения события побудки в программе выхода из останова.

— Бит 14 **RXDM:** Состояние линии- данных

Может использоваться для определения события побудки в программе выхода из останова.

— Бит 13 **LCK:** Закрыто

Ставится аппаратно после приёма как минимум двух последовательных пакетов SOF после снятия сигнала сброса USB или конца последовательности побудки USB. Единожды замкнутый, таймер фреймов остаётся таковым до сброса USB или события побудки USB.

— Биты 12:11 **LSOF[1:0]**: Потерянные SOF

Число потерянных последовательных пакетов SOF при выдаче прерывания ESOF. Очищается приёмом пакета SOF.

— Биты 10:0 **FN[10:0]:** Номер фрейма

11-бит номер фрейма из последнего принятого пакета SOF. Инкрементируется хостом для каждого посылаемого фрейма и полезен для изохронных передач. Обновляется при запросе прерывания SOF.

Адрес устройства USB (USB_DADDR)

Смещение адреса: 0x4C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EF	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0
								rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:8 Резерв, не трогать.

— Бит 7 **EF:** Разрешение работы устройства USB

Ставится программно. Адрес устройства лежит в битах ADD[6:0]. Если бит равен '0, то ничего не работает, безотносительно установкам регистров USB_EPnR.

— Биты 6:0 **ADD[6:0]:** Адрес устройства

Содержат рабочий адрес устройства, назначаемый хостом PC во время фазы перечисления. Вместе с адресом конечной точки из регистра USB_EPnR сравниваются с информацией из токена USB для определения своей конечной точки.

Адрес таблицы буферов (USB_BTABLE)

Смещение адреса: 0x50

По сбросу: 0x0000

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

— Биты 15:3 **BTABLE[15:3]**: Адрес таблицы буферов в памяти пакетов
Аппаратно выравнен на границу 8 байт. См. *Структуру и использование буферов пакетов*.

— Биты 2:0 Резерв, аппаратный 0.

23.5.2. Регистры конечных точек

БУ USB поддерживает до 8 двунаправленных конечных точек. Все устройства USB должны поддерживать управляющую конечную точку с нулевым адресом (биты EA). Если нескольким конечным точкам назначен один адрес, то поведение БУ USB неописуемо. Информация конечной точки хранится в регистре USB_EPnR.

Регистрs конечной точки n USB (USB_EPnR), n=[0..7]

Смещение адреса: 0x00 до 0x1C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR_RX	DTOG_RX	STAT_RX[1:0]		SETUP	EP TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	t	t	t	rw	rw	rw	rw

Все конечные точки имеют свой регистр USB_EPnR где *n* это идентификатор точки. По сбросу USB от шины или битом FRES в регистре CTLR не сбрасываются только биты CTR_RX и CTR_TX чтобы не терять правильный пакет извещения сразу после сброса.

Циклов *чтение-модификация-запись* с этими регистрами надо избегать, поскольку некоторые биты могут аппаратно встать в период между чтением и записью и быстренько очиститься так и не обработанными. Страдающие эти биты имеют свой "инвариант", который надо использовать только когда изменять их не надо. Изменять эти регистры надо командой загрузки с значением "инварианта" в битах, изменяемых только аппаратурой.

— Бит 15 **CTR_RX:** Правильный приём
Ставится аппаратно при успешном завершении передачи OUT/SETUP, снимается программно. Если стоит бит CTRM в регистре USB_CNTR, то выдаётся общее прерывание, а запрос конечной точки уже стоит. Тип передачи, OUT или SETUP, определяют по биту SETUP. Передача, завершившаяся квитком NAK или STALL этого бита не ставит, равно как при ошибках протокола или неверном переключении данных. Чтение/запись, но действует только запись 0, запись 1 бессмысленна.

— Бит 14 **DTOG_RX:** Переключение данных, операция приёма
Для не-изохронных точек содержит ожидаемое значение бита переключения данных (0=DATA0, 1=DATA1) для следующего принимаемого пакета данных. Переключается аппаратно при посылке квитка ACK хосту USB с последующим приёмом данных с таким PID.
Для управляющих точек этот бит аппаратно снимается после приёма адресованного ей SETUP PID.
Для двухбуферных конечных точек используется и при переключении буферов пакетов (см. *Секцию 23.4.3*).
Для изохронных конечных точек используется только при переключении буферов пакетов, так как передаются только пакеты DATA0 (см. *Секцию 23.4.4*). Аппаратно переключается сразу после приёма данных, так как квитков здесь нет.
Этот бит можно переключать вручную для инициализации его значения (обязательно для не-управляющих точек) или для особого использования буферов. DTOG_RX переключается только записью 1.

— Биты 13:12 **STAT_RX[1:0]:** Биты состояния, операция приёма
См. Таблицу 173. Эти биты можно инициализировать вручную, перекидывая бит записью ‘1. После нормального приёма (CTR_RX=1) пакетов OUT или SETUP (только для управляющих точек) биты STAT_RX аппаратно ставятся в NAK, и у программы появляется время на обработку полученных данных перед подтверждением их приёма для запуска новой передачи.

Двухбуферные групповые конечные точки разбираются с этими битами по особому (см. *Секцию 23.4.3*).

У изохронных конечных точек состояние может быть только “VALID” или “DISABLED”, и аппаратно оно не изменяется. Программная установка битов STAT_RX изохронных точек в ‘STALL’ или ‘NAK’ сводит БУ USB с ума. Чтение/запись, но биты переключаются только записью ‘1.

— Бит 11 **SETUP:** Передача Setup завершена

Только чтение, ставится аппаратно при успешном приёме пакета SETUP для управляющих конечных точек. При CTR_RX=1 не изменяется, читайте на здоровье, изменится после сброса CTR_RX.

— Биты 10:9 **EP_TYPE[1:0]:** Тип конечной точки

См. Таблицу 174. Конечные точки 0 всегда управляющие, но такими могут быть и другие точки и только такие принимают передачи SETUP. Другими точками они игнорируются. На SETUP нельзя отвечать NAK или STALL. Если конечная точка определена как NAK, то БУ USB не отвечает, симулируя ошибку приёма. Если конечная точка определена как STALL, то пакет SETUP будет принят, данные записаны и выдано прерывание CTR. Передачи OUT управляющей конечной точкой принимаются как обычно.

Групповые и прерывающие конечные точки отличаются только специальными функциями, определёнными битом EP_KIND.

Изохронные точки описаны в *Секции 23.4.4*.

— Бит 8 **EP_KIND:** Вид конечной точки

Значение бита зависит от битов EP_TYPE. См. таблицу 175.

DBL_BUF: Программно разрешает двухбуферную работу групповой точки, см. *Секцию 23.4.3*.

STATUS_OUT: Ставится программно. При стоящем бите на передачи OUT с хоть одним байтом данных надо отвечать ‘STALL’ вместо ‘ACK’. Так повышают устойчивость программы к ошибкам протокола для управляющих конечных точек. Если бит STATUS_OUT сброшен, то передачи OUT могут иметь любое число байтов данных.

— Бит 7 **CTR_TX:** Правильная передача

Ставится аппаратно при успешном завершении передачи IN, снимается программно. Если стоит бит CTRM в регистре USB_CNTR, то выдаётся общее прерывание, а запрос конечной точки уже стоит. Передача, завершившаяся квитком NAK или STALL этого бита не ставит, равно как при ошибках протокола или неверном переключении данных. Чтение/запись, но действует только запись 0, запись 1 бессмысленна.

— Бит 6 **DTOG_TX:** Переключение данных, операция передачи

Для не-изохронных точек содержит ожидаемое значение бита переключения данных (0=DATA0, 1=DATA1) для следующего передаваемого пакета данных. Переключается аппаратно при приёме квитка ACK от хоста USB с последующей передачей пакета данных.

Для управляющих точек этот бит аппаратно ставится после приёма адресованного ей SETUP PID.

Для двухбуферных конечных точек используется и при переключении буферов пакетов (см. *Секцию 23.4.3*).

Для изохронных конечных точек используется только при переключении буферов пакетов, так как передаются только пакеты DATA0 (см. *Секцию 23.4.4*). Аппаратно переключается сразу после передачи данных, так как квитков здесь нет.

Этот бит можно переключать вручную для инициализации его значения (обязательно для не-управляющих точек) или для особого использования буферов. DTOG_TX переключается только записью 1.

— Биты 5:4 **STAT_TX[1:0]:** Биты состояния, операция передачи

См. Таблицу 176. Эти биты можно инициализировать вручную, перекидывая бит записью ‘1. После нормальной передачи (CTR_TX=1) пакетов IN или SETUP (только для управляющих точек) биты CTR_TX аппаратно ставятся в NAK, и БУ ждёт запуска новой передачи.

Двухбуферные групповые конечные точки разбираются с этими битами по особому (см. *Секцию 23.4.3*).

У изохронных конечных точек состояние может быть только “VALID” или “DISABLED”, и аппаратно оно не изменяется. Программная установка битов STAT_EX изохронных точек в ‘STALL’ или ‘NAK’ сводит БУ USB с ума. Чтение/запись, но биты переключаются только записью ‘1.

— Биты 3:0 **EA[3:0]:** Адрес конечной точки

Идентификатор конечной точки, пишется программно. Записывать нужно до включения точки.

Таблица 173. Статус приёма

STAT_RX[1:0]	Значение
00	**DISABLED:** все запросы приёма к этой точке игнорируются.
01	**STALL**: точка останавливается, на все запросы приёма отвечает STALL.
10	**NAK**: точка тормозится, на все запросы приёма отвечает NAK.
11	**VALID**: к приёму готовы.

Таблица 174. Тип конечной точки

EP_TYPE[1:0]	Значение
00	BULK
01	CONTROL
10	ISO
11	INTERRUPT

Таблица 175. Вид конечной точки

EP_TYPE[1:0]		Значение EP_KIND
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT
10	ISO	Не используется
11	INTERRUPT	Не используется

Таблица 176. Статус передачи

STAT_TX[1:0]	Значение
00	**DISABLED:** все запросы передачи к этой точке игнорируются..
01	**STALL**: точка останавливается, на все запросы передачи отвечает STALL.
10	**NAK**: точка тормозится, на все запросы передачи отвечает NAK.
11	**VALID**: к передаче готовы.

23.5.3. Таблица описания буферов

Строки расположенной в памяти буферов пакетов таблицы описания можно рассматривать как дополнительные регистры, задающие адреса и длины буферов данных для обмена макроячейки USB и STM32F10xxx. Доступ к памяти пакетов через мост APB идёт по адресам, выравненным на границу слова (32 бита), а не реальным, используемым БУ USB для регистра `USB_BTABLE` и строк таблицы описания.

Далее используются два вида адресов: один для программного доступа к памяти пакетов, и второй, местный, для БУ USB. Для правильного доступа к памяти пакетов из STM32F10xxx местные адреса нужно умножать на два. Память пакетов начинается с адреса 0x4000 6000. Строки таблицы описания буферов ассоциированы с регистрами `USB_EPnR`.

Адрес буфера передачи n (USB_ADDRn_TX)

Смещение адреса: `[USB_BTABLE] + n*16`

Местный адрес USB: `[USB_BTABLE] + n*8`

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRn_TX[15:1]															-
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	-

— Биты 15:1 **ADDRn_TX[15:1]:** Адрес буфера передачи

Адрес начала буфера данных конечной точки, из регистра USB_EPnR, для передачи по следующему адресованному ей токену IN.

— Бит 0

Нужно писать как "'0" из-за словного выравнивания памяти пакетов и адресов буферов.

Счётчик n байтов передачи (USB_COUNTn_TX)

Смещение адреса: `[USB_BTABLE] + n*16 + 4`

Местный адрес USB: `[USB_BTABLE] + n*8 + 2`

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

— Биты 15:10 Резерв, не трогать.

— Биты 9:0 **COUNTn_TX[9:0]:** Счётчик байтов передачи

Число байтов передачи конечной точки, из регистра USB_EPnR, для передачи по следующему адресованному ей токену IN.

Двухбуферные и изохронные конечные точки `IN` имеют два регистра `USB_COUNTn_TX`: `USB_COUNTn_TX_1` и `USB_COUNTn_TX_0`.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						COUNTn_TX_1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						COUNTn_TX_0[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Адрес буфера приёма n (USB_ADDRn_RX)

Смещение адреса: `[USB_BTABLE] + n*16 + 8`

Местный адрес USB: `[USB_BTABLE] + n*8 + 4`

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRn_RX[15:1]															-
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	-

— Биты 15:1 **ADDRn_RX[15:1]:** Адрес буфера приёма

Адрес начала буфера данных конечной точки, из регистра USB_EPnR, для приёма по следующему адресованному ей токену OUT/SETUP.

— Бит 0

Нужно писать как '0 из-за словного выравнивания памяти пакетов и адресов буферов.

Счётчик n байтов приёма (USB_COUNTn_RX)

Смещение адреса: `[USB_BTABLE] + n*16 + 12`

Местный адрес USB: `[USB_BTABLE] + n*8 + 6`

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLSIZE	NUM_BLOCK[4:0]					COUNTn_RX[9:0]									
rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r

Эта строка таблицы содержит ещё два значения, нужные для приёма пакетов. Старшие биты определяют размер приёмного буфера, для определения переполнения буфера, а младшие заполняются БУ USB числом принятых байтов. Размер приёмного буфера это часть дескриптора конечной точки и обычно определяется в процессе перечисления в соответствии с её параметром maxPacketSize.

— Бит 15 **BL_SIZE:** Размер блока
  – При BL_SIZE=0, блок равен 2 байтам, буфер занимает от 2 до 62 байт.
  – If BL_SIZE=1, блок равен 32 байтам, буфер занимает от 32 до 1024 байтов.

— Биты 14:10 **NUM_BLOCK[4:0]:** Число блоков
Число блока в выделенной памяти. См. Таблицу 177.

— Биты 9:0 **COUNTn_RX[9:0]:** Счётчик байтов приёма
Число байтов конечной точки, из регистра USB_EPnR, принятых по последнему адресованному ей токену OUT/SETUP.

Двухбуферные и изохронные конечные точки OUT имеют два регистра USB_COUNTn_RX: USB_COUNTn_RX_1 и USB_COUNTn_RX_0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE_1	NUM_BLOCK_1[4:0]					COUNTn_RX_1[9:0]									
rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLSIZE_0	NUM_BLOCK_0[4:0]					COUNTn_RX_0[9:0]									
rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r

Table 177. Память буфера

NUM_BLOCK[4:0]	Память при BL_SIZE=0	Память при BL_SIZE=1
0 (‘00000)	Недопустимо	32 байта
1 (‘00001)	2 байта	64 байта
2 (‘00010)	4 байта	96 байт
3 (‘00011)	6 байт	128 байт
...	...	...
15 (‘01111)	30 байт	512 байт
16 (‘10000)	32 байта	N/A
17 (‘10001)	34 байта	N/A
18 (‘10010)	36 байт	N/A
...	...	...
30 (‘11110)	60 байт	N/A
31 (‘11111)	62 байта	N/A

23.5.4. Карта регистров USB

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**USB_EP0R**	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**USB_EP1R**	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**USB_EP2R**	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**USB_EP3R**	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**USB_EP4R**	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**USB_EP5R**	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**USB_EP6R**	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**USB_EP7R**	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20-0x3F	Reserved																																
0x40	**USB_CNTR**	Reserved																CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	Reserved		RESUME	FSUSP	LPMODE	PDWN	FRES	
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	1	1	
0x44	**USB_ISTR**	Reserved																CTR	PMAOVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	Reserved		DIR	EP_ID[3:0]				
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	0	
0x48	**USB_FNR**	Reserved																RXDP	RXDM	LCK	LSOF [1:0]		FN[10:0]										
	Reset value																	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x
0x4C	**USB_DADDR**	Reserved																								EF	ADD[6:0]						
	Reset value																									0	0	0	0	0	0	0	0
0x50	**USB_BTABLE**	Reserved																BTABLE[15:3]															Reserved
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			

24. Локальная сеть (bxCAN)

24.1. Введение в bxCAN

Basic Extended CAN (**bxCAN**) поддерживает протоколы CAN версий 2.0A и B. Обрабатывает большое число входных сообщений не нагружая CPU и соответствует требованиям приоритетов выходных сообщений. Контроллер CAN имеет все аппаратные средства поддержки временного разделения каналов связи.

24.2. Основные свойства bxCAN

- Поддерживает протокол CAN версии 2.0 A, B Active
- Скорость до 1 Mbit/s
- Поддерживает временное разделение каналов

Передача

- Три передающих почтовых ящика
- Конфигурируемый приоритет передачи
- Метки времени при передаче SOF

Приём

- Два приёмных FIFO с тремя стадиями
- Масштабируемые банки фильтров:
  - 28 общих банков для CAN1 и CAN2 в сетевых устройствах
  - 14 банков для других устройств STM32F10xxx
- Список идентификаторов
- Конфигурируемое переполнение FIFO
- Метки времени при приёме SOF

Временное разделение каналов

- Выключение повторной передачи
- 16-бит независимый таймер
- Метка времени в последних двух байтах передачи

Управление

- Маскируемые прерывания
- Удобное размещение ящиков в адресном пространстве

Парный CAN

- CAN1: Ведущий bxCAN для связи ведомого bxCAN с 512-байт SRAM памятью
- CAN2: Ведомый bxCAN, прямой связи с SRAM памятью нет.
- Две ячейки bxCAN имеют общую 512-байт SRAM память

В устройствах разной плотности USB и CAN используют одну 512-байт SRAM память и не могут работать параллельно. USB и CAN могут работать в одной системе, но не одновременно.

24.3. Общее описание bxCAN

Сегодняшние системы CAN содержат всё увеличивающееся число узлов в сети и часто несколько сетей, объединённых через шлюзы. Число сообщений в системе значительно возросло, и к ним добавились сообщения Диагностики и Управления сетью.

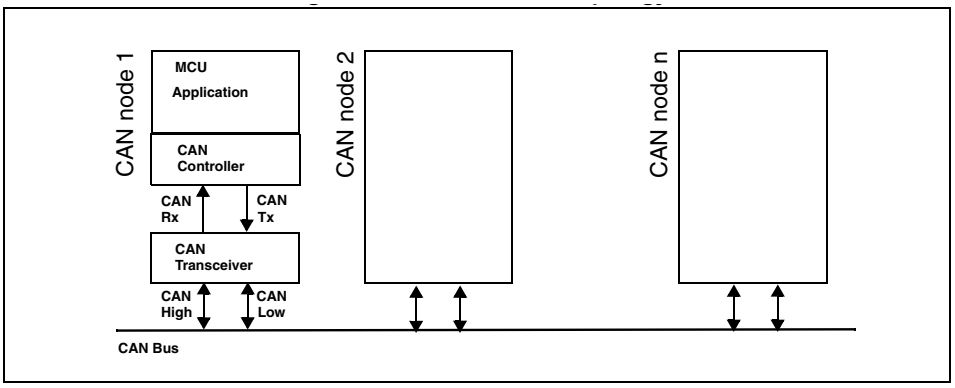
- Нужен улучшенный механизм фильтрации всех типов сообщений. Более того, прикладные задачи требуют всё больше времени CPU, так что надо сокращать затраты на приём сообщений.
- Схема приёмного FIFO отпускает CPU больше времени на прикладные задачи без потери сообщений.

Стандартный HLP (Протокол Верхнего уровня), основанный на стандартных драйверах CAN, требует эффективной связи с контроллером CAN.

24.3.1. Активное ядро CAN 2.0B

Модуль bxCAN обрабатывает приём и передачу сообщений CAN полностью автономно, полностью поддерживает стандартные (11-бит) и расширенные (29-бит) идентификаторы.

Топология сети CAN.



24.3.2. Регистры управления, состояния и конфигурации

Используются для конфигурации параметров CAN, запроса передач, обработки приёма, управления прерываниями и диагностики.

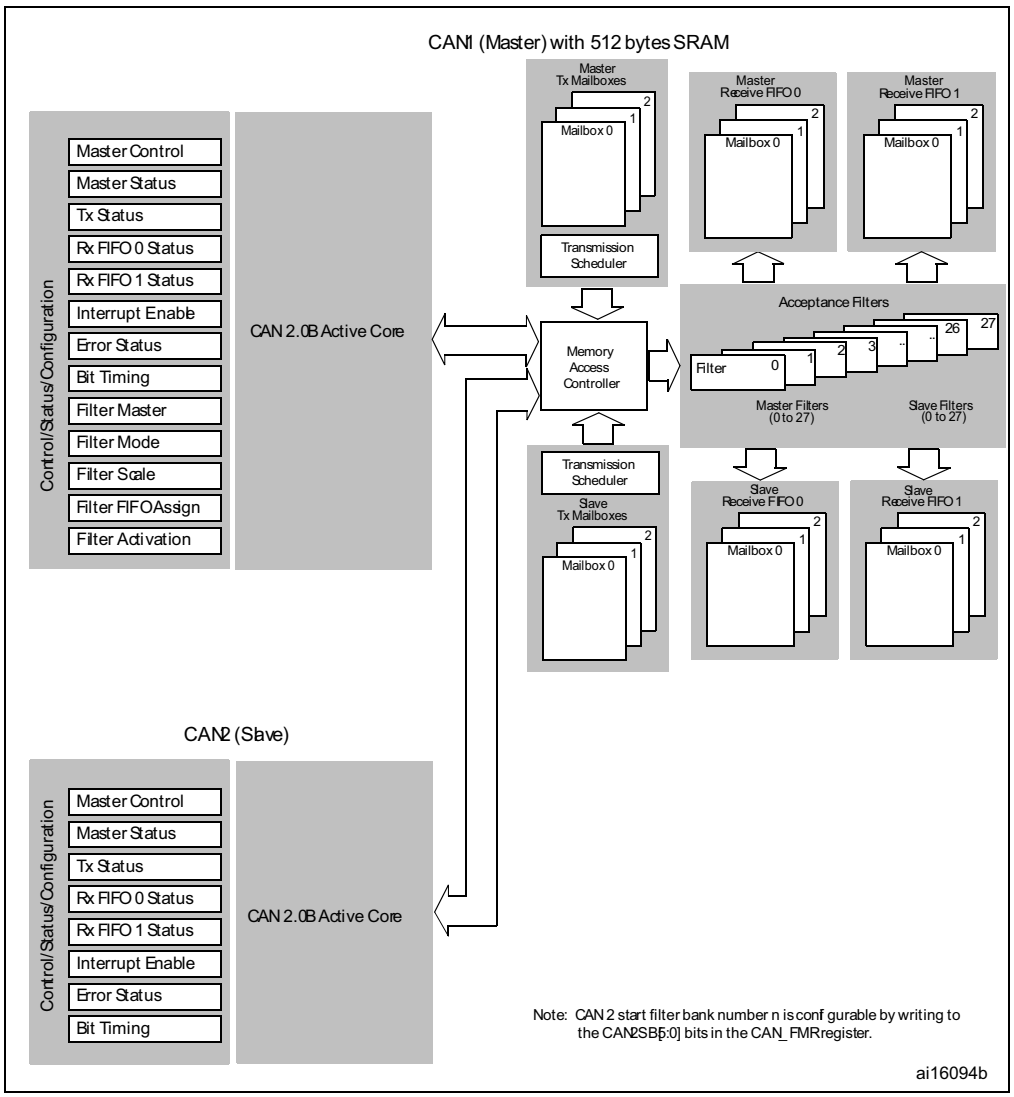
24.3.3. Почтовые ящики Tx

Программа складывает сообщения в три почтовых ящика, порядок отправки определяет Планировщик.

24.3.4. Приёмные фильтры

bxCAN сетевых устройств имеет 28 масштабируемых/конфигурируемых банков фильтров входных сообщений, отбирающих нужные и отбрасывающих чужие. В других устройствах содержится 14 банков фильтров. Аппаратура использует два приёмных FIFO на три сообщения каждый.

Блок-схема CAN



24.4. Режимы работы bxCAN

bxCAN работает в трёх режимах: **инициализация**, **нормальный** и **Сон**. После аппаратного сброса bxCAN Спит с активной внутренней подпоркой на CANTX. Программа переводит bxCAN в **инициализацию** или **Сон** установкой битов INRQ или SLEEP в регистре CAN_MCR. После установки режима bxCAN ставит биты INAK или SLAK в регистре CAN_MSR и отключает внутреннюю подпорку. В **нормальном** режиме биты INAK и SLAK чисты. Перед переходом в **нормальный** режим bxCAN всегда **синхронизируется** с шиной CAN. Для этого bxCAN ждёт простоя шины CAN, то есть на CANRX появилось 11 последовательных рецессивных битов.

24.4.1. Инициализация

Программная инициализация выполняется в режиме инициализации аппаратуры. Для этого программа ставит бит INRQ в регистре CAN_MCR и ждёт подтверждения от bxCAN (стоящий бит INAK в регистре CAN_MSR). Для выхода из инициализации программа снимает бит INRQ. bxCAN выходит из инициализации после аппаратного снятия бита INAK.

В режиме инициализации все обмены с шиной CAN остановлены и выход CANTX стоит рецессивным (высоким). Переход в инициализацию не изменяет регистры конфигурации.

Для инициализации контроллера CAN программа должна установить регистры времянки битов (CAN_BTR) и опций CAN (CAN_MCR).

Для инициализации регистров, связанных с банками фильтров CAN (режим, масштаб, назначения FIFO, активация и значения фильтров), программа должна поставить бит FINIT в CAN_FMR. Фильтры можно инициализировать и вне режима инициализации.

NB: При FINIT=1, приём CAN деактивирован. Значения фильтров можно изменить снятием битов активации (в регистре CAN_FA1R). Неиспользуемый фильтр лучше держать неактивным (соответствующий бит FACT чистый).

24.4.2. Нормальный режим

Запрос на выход в нормальный режим выдаётся очисткой бита INRQ в регистре CAN_MCR. Теперь надо синхронизироваться с шиной, дождавшись 11 последовательных рецессивных бит (Простой шины). Переключение в нормальный режим подтверждается аппаратным снятием бита INAK в регистре CAN_MSR.

Инициализация значений фильтров не зависит от режима инициализации и делается с неактивными фильтрами (чистый бит FACTx). Масштаб фильтра и конфигурация выполняются до входа в нормальный режим.

24.4.3. Режим сна

Режим сна включается программной установкой бита SLEEP в регистре CAN_MCR. Тогда останавливаются такты bxCAN, но почтовые ящики bxCAN остаются программно доступными.

При выдаче запроса на **инициализацию** установкой бита INRQ для спящего bxCAN бит SLEEP надо чистить.

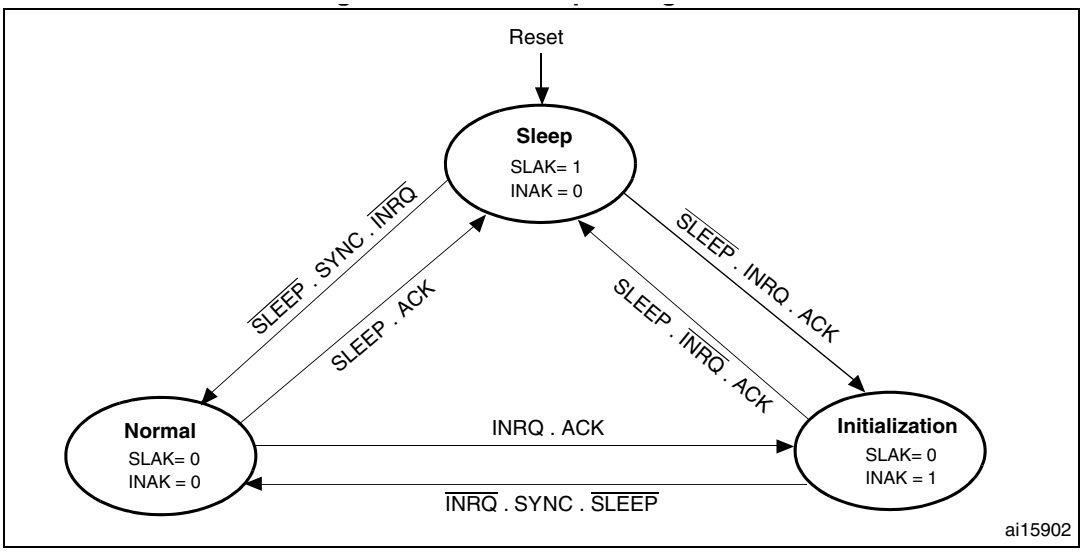
bxCAN пробуждается (выходит из Сна) программным снятием бита SLEEP или при появлении активности на шине CAN.

Если бит AWUM регистра CAN_MCR стоит, то бит SLEEP снимается аппаратно, иначе его надо чистить программно.

NB: Если прерывание побудки разрешено, (стоит бит WKUIE регистра CAN_IER), то оно выдаётся даже при обнаружении активности шины CAN.

После снятия бита SLEEP bxCAN выходит из Сна после синхронизации с шиной CAN. Бит SLAK снимается аппаратно в конце последовательности выхода has.

Режимы работы bxCAN.



1. ACK = Состояние ожидания аппаратного подтверждения запросов INAK или SLAK в регистре CAN_MSR
2. SYNC = bxCAN ждёт Простоя шины CAN (11 последовательных рецессивных бит на CANRX)

24.5. Тестовый режим

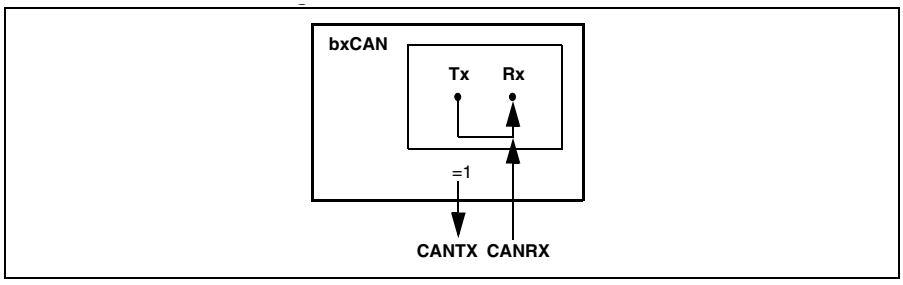
Он выбирается битами SILM и LBKM в регистре CAN_BTR. Их ставят в режиме инициализации bxCAN. После выбора тестового режима нужно отпустить bxCAN в нормальный режим снятием бита INRQ в регистре CAN_MCR.

24.5.1. Режим тишины

bxCAN уходит в Тишину при установке бита SILM в регистре CAN_BTR.

В этом режиме bxCAN может принимать верные фреймы данных и удалённые фреймы, но на шину отсылает только рецессивные биты и не может запустить передачу. Если bxCAN надо послать доминантный бит (ACK, флаг перегрузки, флаг активной ошибки), то он внутренне перенаправляется так, чтобы Ядро CAN Core отследило его, шина CAN может остаться в рецессивном состоянии. Режим тишины может использоваться для анализа передач по шине CAN, не мешая передачей доминантных битов (Биты подтверждения, Фреймы Ошибки).

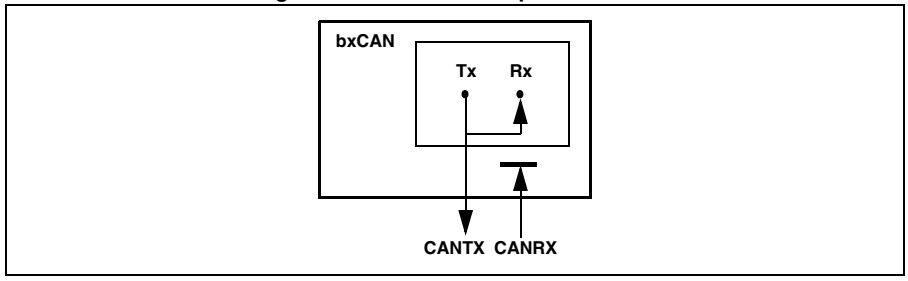
bxCAN в режиме тишины.



24.5.2. Режим перемычки (loopback)

bxCAN уходит сюда при установке бита LBKM в регистре CAN_BTR. В этом режиме bxCAN воспринимает свои передачи как входные и после фильтрации сохраняется в Приёмном почтовом ящике.

bxCAN с перемычкой.

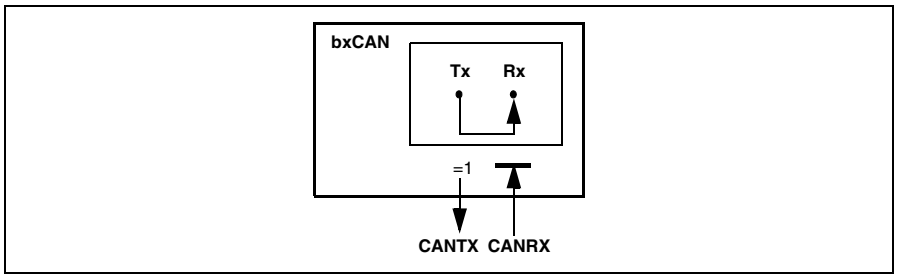


Это само-тестирование. В этом режиме Ядро CAN Core игнорирует ошибки подтверждения (нет доминантного бита в слоте подтверждения фреймов данных/удалённых фреймов). Сигнал Tx передаётся на Rx внутри, реальное значение ножки CANRX отбрасывается. Передаваемые сообщения видны на ножке CANTX.

24.5.3. Тишина + перемычка

Режимы Тишины и Перемычки можно объединить одновременной установкой битов LBKM и SILM в регистре CAN_BTR. Можно проверять себя не влияя на шину CAN. Ножка CANRX отключается от bxCAN, ножка CANTX удерживается рецессивной.

Тишина + перемычка



24.6. Режим отладки

В режиме отладки (ядро Cortex®-M3 остановлено), bxCAN может работать или стоять в зависимости от:

- Бита DBG_CAN1_STOP для CAN1 и бита DBG_CAN2_STOP для CAN2 в модуле DBG.
- Бита DBF в регистре CAN_MCR.

24.7. Функциональное описание bxCAN

24.7.1. Обработка передачи

Перед запросом передачи битом TXRQ в регистре CAN_TIxR программа должна выбрать **пустой** передающий почтовый ящик, поставить идентификатор, код длины данных (DLC) и сами данные. После выхода ящика из **пустого** состояния, программа лишается права записи в его регистры. Сразу после установки бита TXRQ уходит в состояние удержания и ждёт в очереди приоритетов. Ящик с высшим приоритетом становится **запланированным** для передачи. Передача сообщения из запланированного ящика начинается (состояние **передачи**) при Простое шины CAN. После успешной передачи ящика он снова становится **пустым**. Аппаратура обозначает успешную передачу установкой битов RQCP и TXOK в регистре CAN_TSR.

Причина сбойной передачи показывается битом ALST регистра CAN_TSR в случае Потери Арбитража, и/или битом TERR в случает ошибки передачи.

Приоритет передачи

- **По идентификатору**

  Если удерживается не один ящик, то порядок передачи определяется по идентификаторам хранящихся сообщений. Сообщение с меньшим приоритетом отправляется первым в соответствии арбитражем протокола CAN. При одинаковых идентификаторах первым планируется ящик с меньшим номером.

- **По порядку запросов передачи**

  Битом TXFP в регистре CAN_MCR передающие ящики можно превратить в FIFO. При этом порядок приоритетов задаётся порядком запросов. Весьма полезно при сегментированных передачах.

Отмена

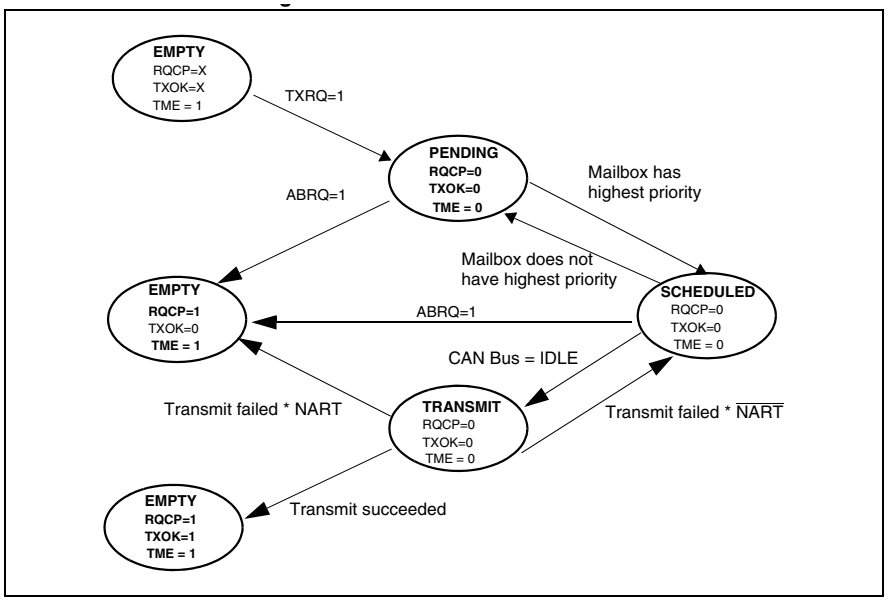
Запрос передачи может быть отменён установкой бита ABRQ в регистре CAN_TSR. В состояниях **удержания** или **планирования** ящик освобождается немедленно. Аборт запроса ящика в состоянии **передачи** может иметь два итога. При успешном завершении передачи ящик становится **пустым** с установкой бита TXOK в регистре CAN_TSR. При сбойной передаче ящик становится **планируемым,** передача прекращается и становится **пустым** с очисткой TXOK. Во всех случаях ящик становится **пустым** не раньше конца текущей передачи.

Не-автоматический повтор передачи

Так исполняются требования стандарта CAN временного разделения каналов. Режим включается установкой бита NART в регистре CAN_MCR. В этом режиме все передачи запускаются единожды.

По концу попытки запрос считается завершённым и ставится бит RQCP в регистре CAN_TSR. Результат передачи указывается в регистре CAN_TSR битами TXOK, ALST и TERR.

Состояния передающих ящиков.



24.7.2. Временное разделение каналов

В этом режиме запускается внутренний счётчик, создающий Метки Времени, хранящиеся в регистрах CAN_RDTxR/CAN_TDTxR, (для ящиков Rx и Tx). Счётчик инкрементируется по каждому биту CAN (см. *Секцию 24.7.7*.). Считывается по биту Старта Фрейма при приёме и передаче.

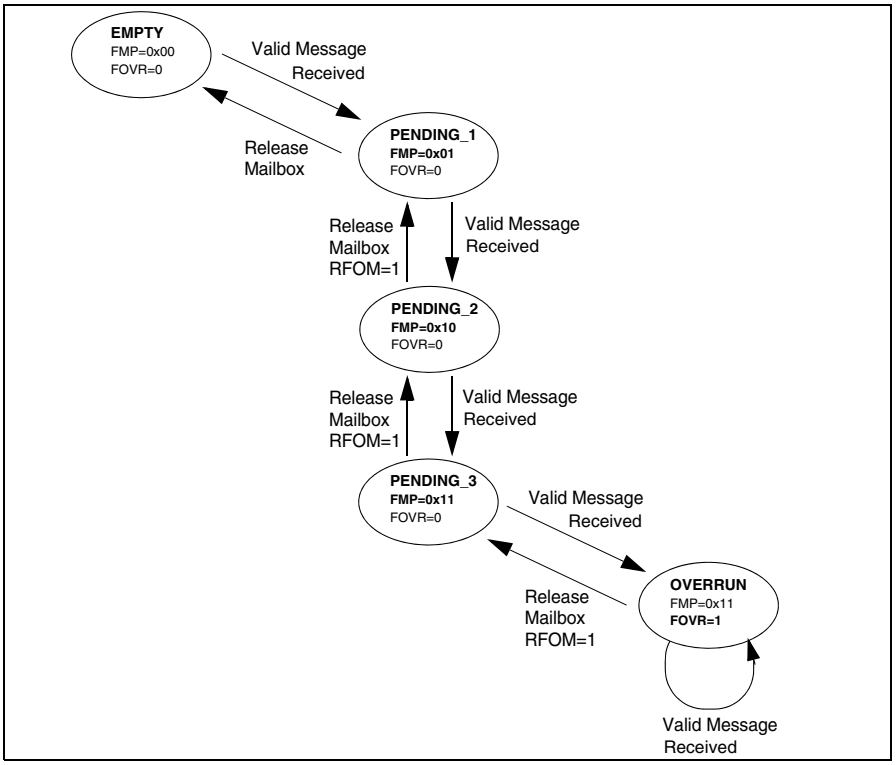
24.7.3. Обработка приёма

Сообщения принимаются в три ящика, организованных как FIFO. Он управляется аппаратно. Программа получает сообщения из выходного ящика FIFO.

Правильное сообщение

Действительным считается безошибочно принятое сообщение, прошедшее через фильтр.

Состояния приёмного FIFO.



Управление FIFO

После приёма верного сообщения **пустой** FIFO становится **задержанным_1**. Аппаратура сообщает об этом записью 01b в биты FMP[1:0] регистра CAN_RFR. Сообщение доступно в выходном ящике FIFO. Программа читает его и освобождает ящик установкой бита RFOM в регистре CAN_RFR. FIFO снова становится **пустым**. Если в то же время принято новое действительное сообщение, то FIFO остаётся **задержанным_1**, новое сообщение можно достать из выходного ящика.

Следующее верное сообщение при неосвобождённом ящике сохранится в FIFO, который перейдёт в состояние **задержанный_2** (FMP[1:0] = 10b). Запись следующего сообщения в FIFO переведёт его в состояние **задержанный_3** (FMP[1:0] = 11b). Вот тут надо бы наконец освобождать ящик установкой бита RFOM, дабы не терять сообщения. См. *Секцию 24.7.5*.

Переполнение

Следующее верное сообщение при FIFO в состоянии **задержанный_3** переведёт его в **переполнение** и одно сообщение будет потеряно. Аппаратура сообщает об этом установкой бита FOVR в регистре CAN_RFR. Теряемое сообщение определяется конфигурацией FIFO:

- Если замок FIFO отключён (бит RFLM в CAN_MCR сброшен), то заменяется самое старое сообщение в FIFO и доступны самые поздние.
- Если замок FIFO включён (бит RFLM в CAN_MCR стоит), то заменяется самое последнее сообщение в FIFO и доступны самые ранние.

Прерывания приёма

После сохранения сообщения в FIFO изменяются биты FMP[1:0] и при стоящем бите FMPIE в регистре CAN_IER выдаётся запрос прерывания.

При заполнении FIFO (записано третье сообщение) ставится бит FULL в регистре CAN_RFR и при стоящем бите FFIE в регистре CAN_IER выдаётся запрос прерывания.

При переполнении ставится бит FOVR и при стоящем бите FOVIE в регистре CAN_IER выдаётся запрос прерывания.

24.7.4. Фильтрация идентификаторов

Протокол CAN относит идентификатор к содержанию сообщения. Следовательно передатчик вещает для всех приёмников. А приёмник по идентификатору решает, нужно ли ему это сообщение. Нужное копируется в SRAM, ненужное теряется.

Нужность определяется 28 (27-0) конфигурируемыми и масштабируемыми банками фильтров. В иных устройствах их всего 14 (13-0). Каждый банк фильтров состоит из двух 32-бит регистров, CAN_FxR0 CAN_FxR1.

Масштаб ширины

Банки масштабируются независимо. В зависимости от масштаба банки имеют:

- Один 32-бит фильтр для битов STDID[10:0], EXTID[17:0], IDE и RTR.
- Два 16-бит фильтра для битов STDID[10:0], RTR, IDE и EXTID[17:15].

Более того, фильтры можно сконфигурировать в режим маски или списка идентификаторов.

Режим маски

Регистр маски определяет значимые и бестолковые биты регистра идентификатора.

Список идентификаторов

Регистры маски тоже становятся регистрами идентификаторов, удваивая их число. Сравниваются все биты идентификаторов.

Конфигурация масштаба и режима банка фильтров

Для этого есть регистр CAN_FMR, но сначала нужно очистить бит FACT регистра CAN_FAR. Масштаб фильтра определяется соответствующим битом FSCx регистра CAN_FS1R. Режим **списка** или **маски** регистров определяется битами FBMx регистра CAN_FMR.

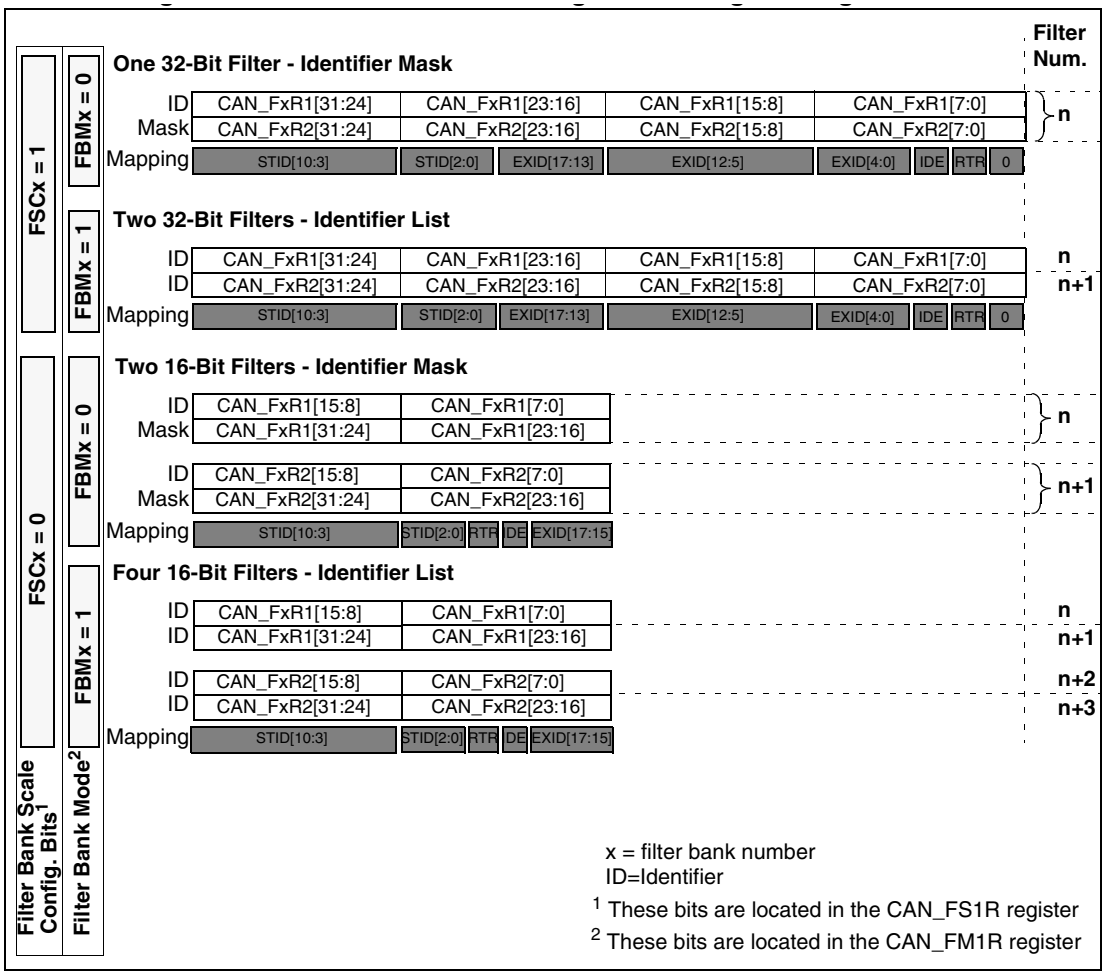
Режим маски выбирает группу идентификаторов.

Режим списка отбирает один идентификатор.

Неиспользуемые фильтры не активируются.

Фильтры внутри банка нумеруются, начиная с 0 до максимума при установленных режиме и масштабе фильтров.

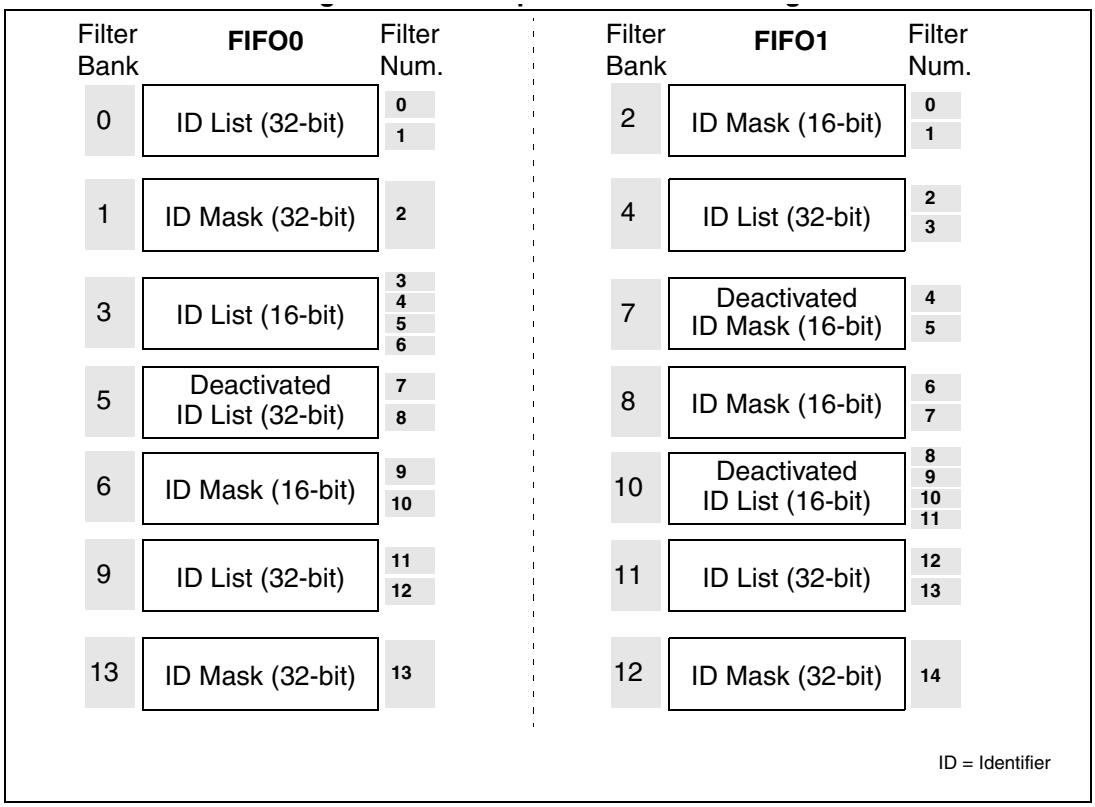
Организация регистров - масштабирование



Индекс фильтра

Вместе с записью в FIFO полученного сообщения с ним сохраняется индекс пропустившего его фильтра (`FMI`). Доступен `FMI` в регистре `CAN_RDTxR`. Это может существенно облегчить последующую обработку сообщения. Индекс вычисляется в соответствии с правилами приоритета фильтров и без учёта их активности. Кроме того, есть две независимые схемы нумерации фильтров по одному для каждого FIFO.

Пример нумерации фильтров

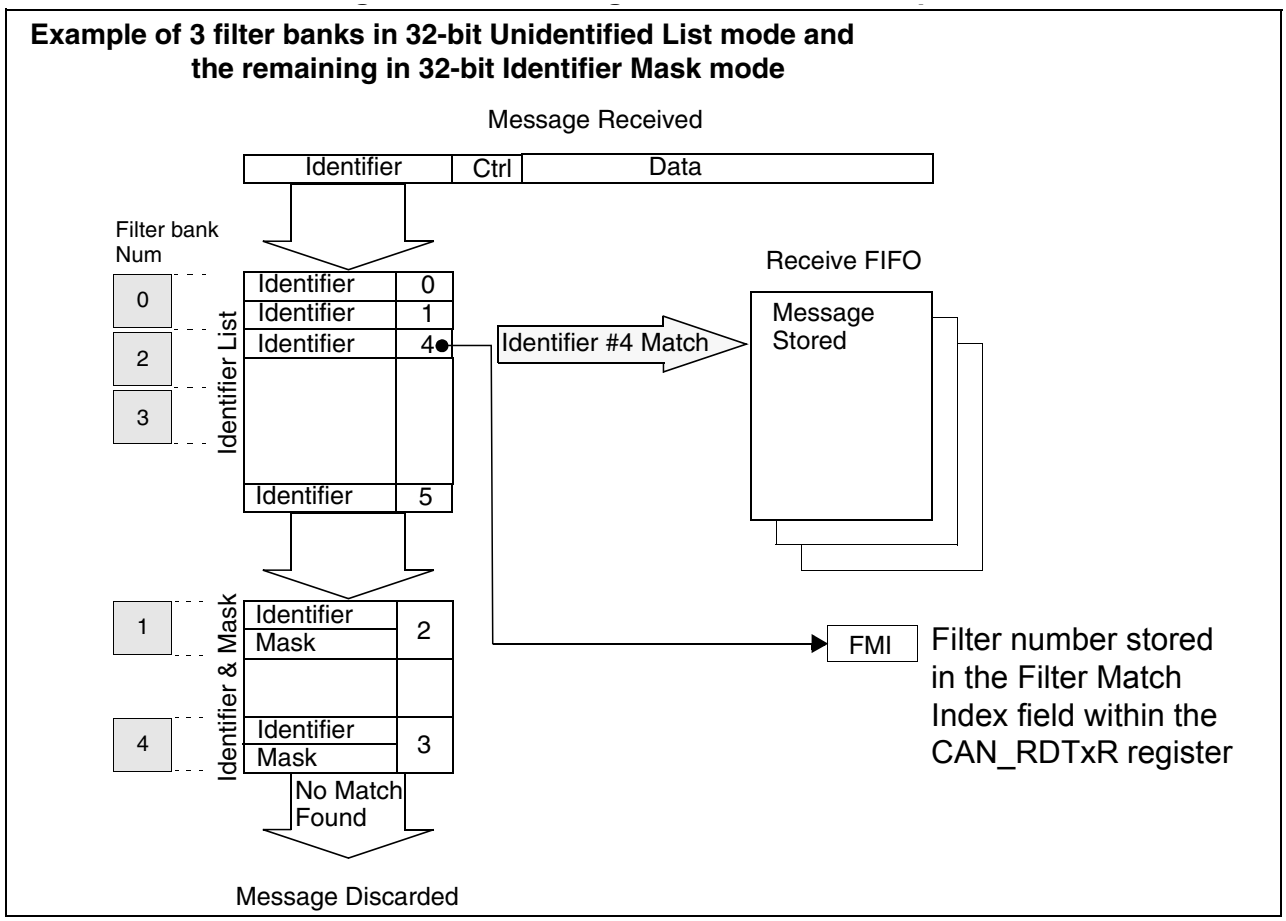


Правила приоритета фильтров

Идентификатор может пройти сразу через несколько фильтров. Тогда FMI выбирается так:

- 32-бит фильтр приоритетнее 16-бит фильтра.
- Из фильтров одинакового масштаба, Список приоритетнее Маски.
- Из фильтров одинакового масштаба и режима, приоритетнее фильтр с меньшим номером.

Пример механизма фильтрации



В этом примере в FIFO записывается FMI 4.

24.7.5. Память сообщений

Сообщения передаются через почтовые ящики, которые содержат идентификатор, данные, управление и метки времени сообщений.

Передающий ящик

Программа пишет сообщение в пустой ящик. Состояние ящика аппаратно ставится в регистр CAN_TSR.

Таблица 179. Регистры передающего ящика

Смещение от базового адреса ящика	Имя регистра
0	CAN_TIxR
4	CAN_TDTxR
8	CAN_TDLxR
12	CAN_TDHxR

Приёмный ящик

Программа может прочитать полученное сообщение через выходной ящик FIFO. После этого ящик надо освободить битом RFOM в регистре CAN_RFR, разрешая приём следующих сообщений. Индекс фильтра хранится в поле MFMI регистра CAN_RDTxR. 16-бит метка времени хранится в поле TIME[15:0] регистра CAN_RDTxR.

Таблица 180. Регистры приёмного ящика

Смещение от базового адреса ящика	Имя регистра
0	CAN_RIxR
4	CAN_RDTxR
8	CAN_RDLxR
12	CAN_RDHxR

24.7.6. Обработка ошибок

Протокол CAN возлагает это на аппаратуру с использованием Счётчика Ошибок Передачи (поле TEC в регистре CAN_ESR) и Счётчика Ошибок Приёма (поле REC в регистре CAN_ESR), которые инкрементируются и декрементируются в соответствии с условиями ошибки, см. стандарт CAN.

Читая их можно определить стабильность сети. Кроме того, текущее состояние ошибки хранится в регистре CAN_ESR. Битами регистра CAN_IER (бит ERRIE и пр.) можно разрешать выдачу прерываний ошибки.

Отключение шины

Возникает при TEC больше 255, о чём глаголет бит BOFF регистра CAN_ESR. Всё, шина недоступна для приёма/передачи.

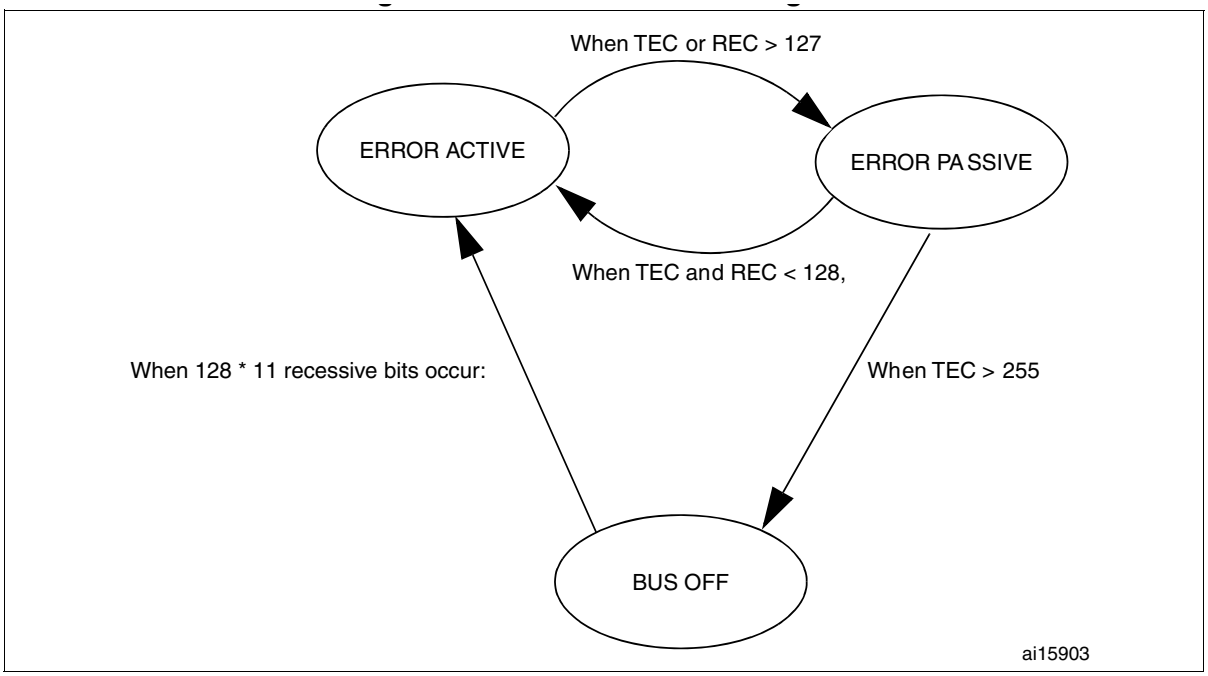
bxCAN восстанавливает работу автоматически или по запросу программы, в зависимости от бита ABOM регистра CAN_MCR (ошибка снова активна). Но в любом случае bxCAN должен дождаться последовательности восстановления (128 появлений 11-ти последовательных рецессивных бит на CANRX).

Если ABOM стоит, то bxCAN начнёт последовательность восстановления сразу после отключения шины.

Если ABOM чист, то программа должна запустить последовательность восстановления запросом включения и выхода из инициализации bxCAN.

NB: В режиме инициализации bxCAN на сигнал CANRx не глядит, и завершить последовательность восстановления не может. **Для этого bxCAN должен быть в нормальном режиме**.

Диаграмма состояний ошибки CAN



24.7.7. Времянка битов

Логика времянки бита следит за последовательной линией шины, синхронизируя точку чтения по фронту стартового бита и ресинхронизируясь по следующим фронтам.

Номинальное время бита разбивается на три сегмента:

- **Сегмент синхронизации** (SYNC_SEG): здесь ожидается изменение бита. Фиксированная длительность в один квант времени (1 x t_q).
- **Сегмент бита 1** (BS1): точка считывания. Включает PROP_SEG и PHASE_SEG1 стандарта CAN. Длительность программируемая от 1 до 16 квантов времени, но может автоматически удлиняться для компенсации положительного сдвига фазы из-за различия частот разных узлов сети.
- **Сегмент бита 2** (BS2): точка передачи или PHASE_SEG2 стандарта CAN. Длительность программируется от 1 до 8 квантов времени, может укорачиваться для компенсации отрицательного сдвига фазы.

Ширина шага ресинхронизации (SJW) определяет верхнюю границу удлинения или сокращения сегментов бита. Программируется от 1 до 4 квантов времени.

Рабочий фронт определяется как первый переход уровня шины из доминантного в рецессивный, а контроллер не передаёт рецессивного бита.

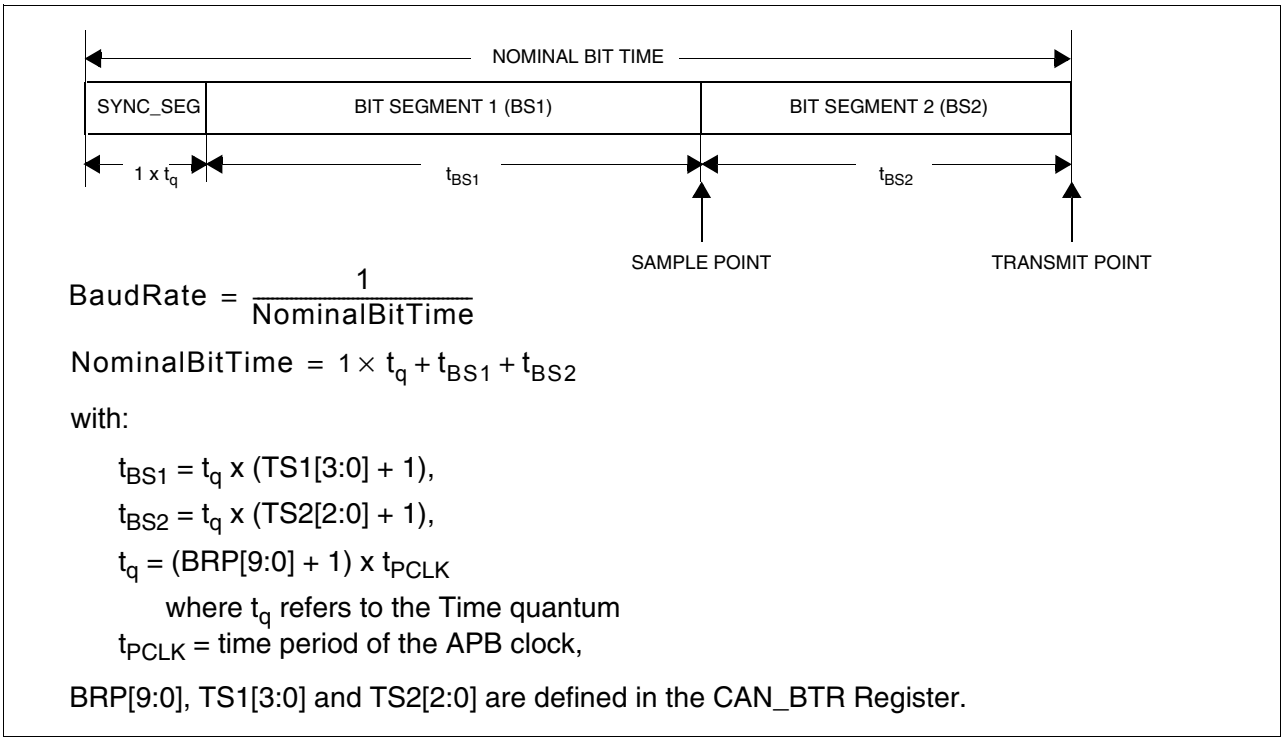
Если рабочий фронт обнаруживается во время BS1 вместо SYNC_SEG, то BS1 расширяется на SJW и точка считывания задерживается.

Если рабочий фронт обнаруживается во время BS2 вместо SYNC_SEG, то BS2 сокращается на SJW точка передачи приближается.

Регистр времянки битов (CAN_BTR) доступен для записи только в Дежурном режиме устройства.

См. стандарт ISO 11898.

Времянка бита



$$BaudRate = \frac{1}{NominalBitTime}$$

$$NominalBitTime = 1 \times t_q + t_{BS1} + t_{BS2}$$

with:

$t_{BS1} = t_q$ x (TS1[3:0] + 1),

$t_{BS2} = t_q$ x (TS2[2:0] + 1),

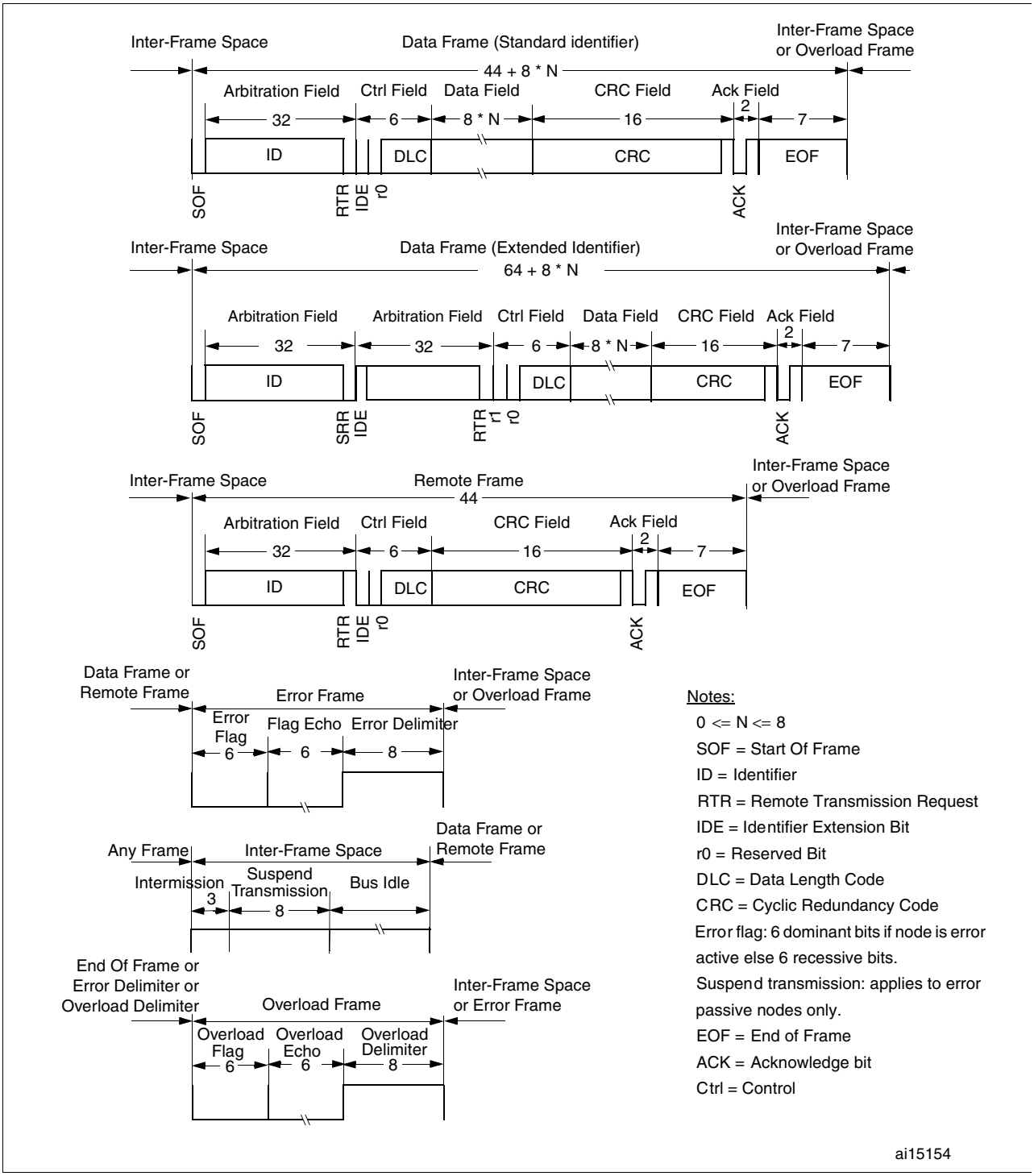
$t_q = (BRP[9:0] + 1) \times t_{PCLK}$

where t_q refers to the Time quantum

t_{PCLK} = time period of the APB clock,

BRP[9:0], TS1[3:0] and TS2[2:0] are defined in the CAN_BTR Register.

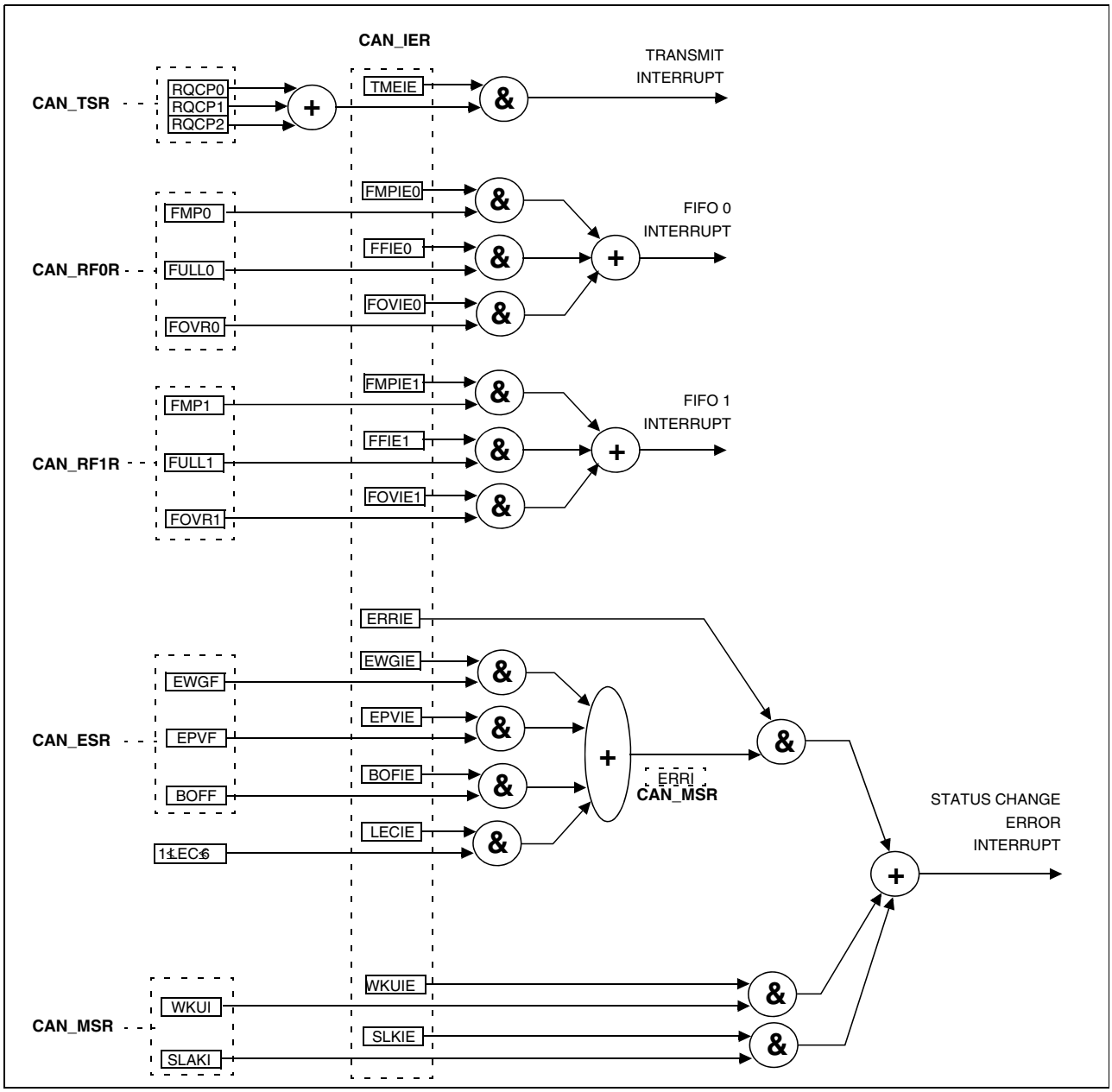
Фреймы CAN



24.8. Прерывания bxCAN

Four interrupt vectors are dedicated to bxCAN. Each interrupt source can be independently enabled or disabled by means of the CAN Interrupt Enable Register (CAN_IER).

Event flags and interrupt generation



- **Прерывание передачи** выдаётся при:

– Пустом передающем ящике 0, стоит бит RQCP0 в регистре CAN_TSR.

– Пустом передающем ящике 1, стоит бит RQCP1 в регистре CAN_TSR.

– Пустом передающем ящике 2, стоит бит RQCP2 в регистре CAN_TSR.

- **Прерывание FIFO 0** выдаётся при:

– Приёме нового сообщения, бит FMP0 в CAN_RF0R не равен ’00’.

– Заполнении FIFO0, стоит бит FULL0 в регистре CAN_RF0R.

– Переполнении FIFO0, стоит бит FOVR0 в регистре CAN_RF0R.

- **Прерывание FIFO 1** выдаётся при:

– Приёме нового сообщения, бит FMP1 в CAN_RF1R не равен ’00’.

– Заполнении FIFO1, стоит бит FULL1 в регистре CAN_RF1R.

– Переполнении FIFO1, стоит бит FOVR1 в регистре CAN_RF1R.

- **Прерывание ошибки и смены состояния** выдаётся при:

– Ошибке, см. регистр CAN_ESR.

– Пробуждении, на CANRx обнаружен SOF.

– Вход в Сон.

24.9. Регистры bxCAN

Регистры доступны словами (32 бита).

24.9.1. Защита доступа к регистрам

Неправильная установка регистров конфигурации может обрушить всю сеть CAN. Так что регистр CAN_BTR можно изменить только в режиме инициализации.

Модифицировать можно только пустой передающий ящик.

Значения фильтров можно изменять деактивируя их или установив бит FINIT. Конфигурацию фильтров (масштаб, режим и назначение FIFO) в регистрах CAN_FMxR, CAN_FSxR и CAN_FFAR можно менять только при FINIT=1 в регистре CAN_FMR.

24.9.2. Регистры управления и состояния CAN

Главный регистр управления CAN (CAN_MCR)

Смещение адреса: 0x00

По сбросу: 0x0001 0002

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET	Reserved							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
rs								rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:17 Резерв, не трогать.

— Бит 16 **DBF:** Останов при отладке

0: CAN работает

1:Приём/передача CAN стоят. Приёмные FIFO доступны/управляемы.

— Бит 15 **RESET:** Главный программный сброс bxCAN

0: Работайте.

1: Принудительный сброс bxCAN, а после сброса в Сон (биты FMP регистра CAN_MCR получают значения сброса). Бит снимается автоматически.

— Биты 14:8 Резерв, не трогать.

— Бит 7 **TTCM**: Режим временного разделения каналов

0: Выключен.

1: Включён

— Бит 6 **ABOM:** Автоматическое восстановление отключения шины

0: Восстановление по запросу программы, если отслежено 128 появлений 11 рецессивных битов и сначала программа поставила и сняла бит INRQ регистра CAN_MCR.

1: Автоматически, если отслежено 128 появлений 11 рецессивных битов.

— Бит 5 **AWUM**: Автоматическая побудка

0: Просыпаемся по запросу программы очисткой бита SLEEP регистра CAN_MCR.

1: Просыпаемся аппаратно при обнаружении сообщения CAN. Бит SLEEP регистра CAN_MCR и бит SLAK регистра CAN_MSR снимаются аппаратно.

— Бит 4 **NART**: Нет автоматическому повтору сбойной передачи!

0: Аппаратура CAN автоматически повторяет сбойную передачу.

1: Сообщение передаётся единожды, не глядя на результат (успех, ошибка или потеря арбитража).

— Бит 3 **RFLM**: Замок приёмного FIFO

0: Приёмный FIFO не замкнут. Приём в заполненный FIFO замещает предыдущее сообщение.

1: Приёмный FIFO замкнут. Сообщение, поступающее в заполненный FIFO теряется.

— Бит 2 **TXFP**: Приоритет передающего FIFO

0: Приоритет управляется идентификатором сообщения

1: Приоритет управляется порядком запросов (хронологически)

— Бит 1 **SLEEP**: Запрос режима Сна

Ставится программно чтобы CAN уснул. Уснёт сразу после завершения текущей операции CAN (передача или приём фрейма CAN).

Снимается программно для выхода из Сна.

Снимается аппаратно при обнаружении бита SOF на сигнале CANRx при стоящем бите AWUM.

Ставится после сброса - CAN стартует спящим.

— Бит 0 **INRQ**: Запрос инициализации

Программная очистка переключает аппаратуру в нормальный режим. После 11 последовательных рецессивных битов на сигнале Rx аппаратура CAN засинхронизирована и готова к обмену. Об этом сообщается аппаратным снятием бита INAK регистра CAN_MSR.

Программная установка переключает аппаратуру в режим инициализации. Переключится сразу после завершения текущей операции CAN (передача или приём фрейма CAN). Об этом сообщается аппаратной установкой бита INAK регистра CAN_MSR.

Главный регистр состояния CAN (CAN_MSR)

Смещение адреса: 0x04

По сбросу: 0x0000 0C02

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.				RX	SAMP	RXM	TXM	Reserved			SLAKI	WKUI	ERRI	SLAK	INAK
				r	r	r	r				rc_w1	rc_w1	rc_w1	r	r

— Биты 31:12 Резерв, не трогать.

— Бит 11 **RX**: Сигнал CAN Rx. Текущее состояние ножки **CAN_RX**.

— Бит 10 **SAMP**: Значение RX в последней точке считывания (принятый бит).

— Бит 9 **RXM**: Сейчас CAN принимает.

— Бит 8 **TXM**: Сейчас CAN передаёт.

— Биты 7:5 Резерв, не трогать.

— Бит 4 **SLAKI**: Прерывание подтверждения Сна

При SLKIE=1, этот бит ставится аппаратно, извещая, что bxCAN спит, и генерирует прерывание смены состояния. Снимается программно или аппаратно при очистке SLAK.

NB: Если SLKIE=0, то опрос SLAKI невозможен. В этом случае можно опросить бит SLAK.

— Бит 3 **WKUI**: Прерывание пробуждения

Этот бит ставится аппаратно, извещая, что обнаружен SOF при спящем bxCAN. Прерывание смены режима вызывается при стоящем бите WKUIE в регистре CAN_IER.

Снимается программно.

— Бит 2 **ERRI**: Прерывание ошибки

Этот бит ставится аппаратно при установке бита ошибки в регистре CAN_ESR и разрешении прерывания соответствующим битом в CAN_IER. Вызывает прерывание смены состояния при стоящем бите ERRIE в регистре CAN_IER.

Снимается программно.

— Бит 1 **SLAK**: Подтверждение Сна

Этот бит ставится аппаратно, извещая, что bxCAN спит. Это подтверждение программного запроса (ставит бит SLEEP в регистре CAN_MCR).

Снимается аппаратно при пробуждении bxCAN (для синхронизации с шиной CAN). Для синхронизации отслеживает появление 11 последовательных рецессивных битов на CAN RX.

NB: Выход из Сна запускается очисткой бита SLEEP регистра CAN_MCR.

— Бит 0 **INAK**: Подтверждение инициализации

Этот бит ставится аппаратно в ответ на программный запрос, извещая, что bxCAN инициализируется.

Снимается аппаратно при выходе bxCAN из инициализации (для синхронизации с шиной CAN). Для синхронизации отслеживает появление 11 последовательных рецессивных битов на CAN RX.

Регистр состояния передачи CAN (CAN_TSR)

Смещение адреса: 0x08

По сбросу: 0x1C00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOW2	LOW1	LOW0	TME2	TME1	TME0	CODE[1:0]		ABRQ2	Reserved			TERR2	ALST2	TXOK2	RQCP2
r	r	r	r	r	r	r	r	rs				rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRQ1	Reserved Res.			TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Reserved			TERR0	ALST0	TXOK0	RQCP0
rs				rc_w1	rc_w1	rc_w1	rc_w1	rs				rc_w1	rc_w1	rc_w1	rc_w1

— Бит 31 **LOW2**: Флаг самого низкого приоритета ящика 2
Ставится аппаратно, из удерживаемых к передаче ящиков ящик 2 имеет самый низкий приоритет.
— Бит 30 **LOW1**: Флаг самого низкого приоритета ящика 1
Ставится аппаратно, из удерживаемых к передаче ящиков ящик 1 имеет самый низкий приоритет.
— Бит 29 **LOW0**: Флаг самого низкого приоритета ящика 0
Ставится аппаратно, из удерживаемых к передаче ящиков ящик 0 имеет самый низкий приоритет.
— Бит 28 **TME2**: Передающий ящик 2 пуст
Ставится аппаратно.
— Бит 27 **TME2**: Передающий ящик 1 пуст
Ставится аппаратно.
— Бит 26 **TME2**: Передающий ящик 0 пуст
Ставится аппаратно.
— Биты 25:24 **CODE[1:0]**: Код ящиков
Если хоть один ящик пуст, то равен номеру следующего освобождаемого передающего ящика.
Если заняты все ящики, то это номер ящика с самым низким приоритетом.
— Бит 23 **ABRQ2**: Запрос аборта ящика 2
Ставится программно, снимается аппаратно при опустении ящика 2.
Установка бита не влияет на не удерживаемый к передаче ящик.
— Биты 22:20 Резерв, не трогать.
— Бит 19 **TERR2**: Ошибка передачи ящика 2
Ставится при сбойной предыдущей передаче.
— Бит 18 **ALST2**: Потеря арбитража ящика 2
Ставится при потере арбитража предыдущей передачей.
— Бит 17 **TXOK2**: Нормальная передача ящика 2
Аппаратно обновляется при каждой попытке передачи.
0: Предыдущая передача сбойная
1: Предыдущая передача успешная
— Бит 16 **RQCP2**: Запрос ящика 2 закончен
Ставится аппаратно при завершении последнего запроса (передача или аборт).
Чистится записью “1” или аппаратно при запросе передачи (установка TXRQ2 в CAN_TMID2R).
Снятие этого бита чистит все биты состояния (TXOK2, ALST2 и TERR2) ящика 2.
— Бит 15 **ABRQ1**: Запрос аборта ящика 1
Ставится программно, снимается аппаратно при опустении ящика 1.
Установка бита не влияет на не удерживаемый к передаче ящик.
— Биты 14:12 Резерв, не трогать.
— Бит 11 **TERR1**: Ошибка передачи ящика 1
Ставится при сбойной предыдущей передаче.
— Бит 10 **ALST1**: Потеря арбитража ящика 1
Ставится при потере арбитража предыдущей передачей.
— Бит 9 **TXOK1**: Нормальная передача ящика 1
Аппаратно обновляется при каждой попытке передачи.
0: Предыдущая передача сбойная
1: Предыдущая передача успешная
— Бит 8 **RQCP1**: Запрос ящика 1 закончен
Ставится аппаратно при завершении последнего запроса (передача или аборт).
Чистится записью “1” или аппаратно при запросе передачи (установка TXRQ1 в CAN_TMID2R).
Снятие этого бита чистит все биты состояния (TXOK1, ALST1 и TERR1) ящика 1.
— Бит 7 **ABRQ0**: Запрос аборта ящика 0
Ставится программно, снимается аппаратно при опустении ящика 0.
Установка бита не влияет на не удерживаемый к передаче ящик.
— Биты 6:4 Резерв, не трогать.
— Бит 3 **TERR0**: Ошибка передачи ящика 0
Ставится при сбойной предыдущей передаче.
— Бит 2 **ALST0**: Потеря арбитража ящика 0
Ставится при потере арбитража предыдущей передачей.

— Бит 1 **TXOK0**: Нормальная передача ящика 0
Аппаратно обновляется при каждой попытке передачи.
0: Предыдущая передача сбойная
1: Предыдущая передача успешная

— Бит 0 **RQCP0**: Запрос ящика 0 закончен
Ставится аппаратно при завершении последнего запроса (передача или аборт).
Чистится записью “1” или аппаратно при запросе передачи (установка TXRQ0 в CAN_TMID2R).
Снятие этого бита чистит все биты состояния (TXOK0, ALST0 и TERR0) ящика 0.

Регистр приёмного FIFO 0 CAN (CAN_RF0R)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RFOM0	FOVR0	FULL0	Res.	FMP0[1:0]	
										rs	rc_w1	rc_w1		r	r

— Биты 31:6 Резерв, не трогать.

— Бит 5 **RFOM0**: Освободи выходной ящик FIFO 0
Ставится программно. Освобождает выходной ящик непустого FIFO для приёма следующего сообщения, на пустой FIFO не влияет. Снимается аппаратно.

— Бит 4 **FOVR0**: Переполнение FIFO 0
Ставится аппаратно при приёме сообщения для заполненного FIFO. Снимается программно.

— Бит 3 **FULL0**: FIFO0 полон.
Ставится аппаратно при трёх сообщениях в FIFO. Снимается программно.

— Бит 2 Резерв, не трогать

— Биты 1:0 **FMP0[1:0]**: Счёт сообщений в FIFO 0
Увеличивается при приёме сообщения в FIFO. Уменьшается при освобождении выходного ящика установкой бита RFOM0.

Регистр приёмного FIFO 1 CAN (CAN_RF1R)

Смещение адреса: 0x10

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RFOM1	FOVR1	FULL1	Res.	FMP1[1:0]	
										rs	rc_w1	rc_w1		r	r

— Биты 31:6 Резерв, не трогать.

— Бит 5 **RFOM1**: Освободи выходной ящик FIFO 1
Ставится программно. Освобождает выходной ящик непустого FIFO для приёма следующего сообщения, на пустой FIFO не влияет. Снимается аппаратно.

— Бит 4 **FOVR1**: Переполнение FIFO 1
Ставится аппаратно при приёме сообщения для заполненного FIFO. Снимается программно.

— Бит 3 **FULL1**: FIFO 1 полон.
Ставится аппаратно при трёх сообщениях в FIFO. Снимается программно.

— Бит 2 Резерв, не трогать

— Биты 1:0 **FMP1[1:0]**: Счёт сообщений в FIFO 1
Увеличивается при приёме сообщения в FIFO. Уменьшается при освобождении выходного ящика установкой бита RFOM1.

Регистр разрешения прерываний CAN (CAN_IER)

Смещение адреса: 0x14

По сбросу: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIE	Reserved			LEC IE	BOF IE	EPV IE	EWG IE	Res.	FOV IE1	FF IE1	FMP IE1	FOV IE0	FF IE0	FMP IE0	TME IE
rw				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

— Биты 31:18 Резерв, не трогать.

— Бит 17 **SLKIE**: Разрешение прерывания Сна при стоящем бите SLAKI
0: Нельзя.
1: Можно.

— Бит 16 **WKUIE**: Разрешение прерывания побудки при стоящем бите WKUI
0: Нельзя.
1: Можно.

— Бит 15 **ERRIE**: Разрешение прерывания ошибки при наличии события в CAN_ESR
0: Нельзя.
1: Можно.

— Биты 14:12 Резерв, не трогать.

— Бит 11 **LECIE**: Разрешение прерывания по коду последней ошибки
0: Бит ERRI не ставится при аппаратной записи кода ошибки в LEC[2:0].
1: Бит ERRI ставится при аппаратной записи кода ошибки в LEC[2:0].

— Бит 10 **BOFIE**: Разрешение прерывания по отключению шины
0: Бит ERRI не ставится при стоящем BOFF.
1: Бит ERRI ставится при стоящем BOFF.

— Бит 9 **EPVIE**: Разрешение прерывания пассивной ошибки
0: Бит ERRI не ставится при стоящем EPVF.
1: Бит ERRI ставится при стоящем EPVF.

— Бит 8 **EWGIE**: Разрешение прерывания ошибки уровня предупреждения
0: Бит ERRI не ставится при стоящем EWGF.
1: Бит ERRI ставится при стоящем EWGF.

— Бит 7 Резерв, не трогать.

— Бит 6 **FOVIE1**: Разрешение прерывания переполнения FIFO при стоящем FOVR
0: Нельзя.
1: Можно.

— Бит 5 **FFIE1**: Разрешение прерывания заполнения FIFO при стоящем FULL
0: Нельзя.
1: Можно.

— Бит 4 **FMPIE1**: Разрешение прерывания непустого FIFO (FMP[1:0] ненулевые)
0: Нельзя.
1: Можно.

— Бит 3 **FOVIE0**: Разрешение прерывания переполнения FIFO при стоящем FOVR
0: Нельзя.
1: Можно.

— Бит 2 **FFIE0**: Разрешение прерывания заполнения FIFO при стоящем FULL
0: Нельзя.
1: Можно.

— Бит 1 **FMPIE0**: Разрешение прерывания непустого FIFO (FMP[1:0] ненулевые)
0: Нельзя.
1: Можно.

— Бит 0 **TMEIE**: Разрешение прерывания пустого выходного ящика (RQCPx=1)
0: Нельзя.
1: Можно.

Регистр состояния ошибки CAN (CAN_ESR)

Смещение адреса: 0x18

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REC[7:0]								TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									LEC[2:0]			Res.	BOFF	EPVF	EWGF
									rw	rw	rw		r	r	r

— Биты 31:24 **REC[7:0]**: Счётчик ошибок приёма

Реализованная часть механизма ограничения ошибок протокола CAN. При возникновении ошибки во время приёма этот счётчик увеличивается на вес ошибки от 1 до 8, определённый в стандарте CAN. После успешного приёма счётчик уменьшается на 1 или сбрасывается до 120, если был больше 128. При достижении счётчиком значения 127 наступает режим пассивной ошибки.

— Биты 23:16 **TEC[7:0]**: Младший байт 9-бит счётчика ошибок передачи

Реализованная часть механизма ограничения ошибок протокола CAN.

— Биты 15:7 Резерв, не трогать.

— Биты 6:4 **LEC[2:0]**: Код последней ошибки на шине CAN

Пишется аппаратно. Нормальная передача чистит поле.

Значение LEC[2:0] = 0b111 пишется программно.

000: Ошибки нет

001: Ошибка заполнения

010: Ошибка формы

011: Ошибка подтверждения

100: Ошибка рецессивного бита

101: Ошибка доминантного бита

110: Ошибка CRC

111: Пишется программно

— Бит 3 Резерв, не трогать.

— Биты 2 **BOFF**: Флаг отключения шины

Ставится аппаратно при переполнении TEC, больше 255.

— Биты 1 **EPVF**: Флаг пассивной ошибки

Ставится аппаратно при TEC или REC >127.

— Биты 0 **EWGF**: Флаг ошибки уровня предупреждения

Ставится аппаратно при TEC или REC≥96.

Регистр времянки бита CAN (CAN_BTR)

Смещение адреса: 0x1C

По сбросу: 0x0123 0000

Программно доступен только в режиме Инициализации bxCAN.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SILM	LBKM	Reserved				SJW[1:0]		Res.	TS2[2:0]			TS1[3:0]			
rw	rw					rw	rw		rw	rw	rw	rw	rw	rw	rw

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

— Бит 31 **SILM**: Режим тишины

0: Выключен

1: Включён

— Бит 30 **LBKM**: Режим перемычки

0: Выключен

1: Включён

— Биты 29:26 Резерв, не трогать

— Биты 25:24 **SJW[1:0]**: Ширина шага ресинхронизации

Число квантов времени аппаратуры CAN для удлинения или сокращения бита при ресинхронизации.

$t_{RJW} = t_q \times (SJW[1:0] + 1)$

— Бит 23 Резерв, не трогать.
— Биты 22:20 **TS2[2:0]**: Длина сегмента времени 2 в квантах
$t_{BS2} = t_q$ x (TS2[2:0] + 1)
— Биты 19:16 **TS1[3:0]**: Длина сегмента времени 1 в квантах
$t_{BS1} = t_q$ x (TS1[3:0] + 1)
— Биты 19:16 Резерв, не трогать
— Биты 9:0 **BRP[9:0]**: Длина кванта времени
t_q = (BRP[9:0]+1) x t_{PCLK}

24.9.3. Регистры почтовых ящиков CAN

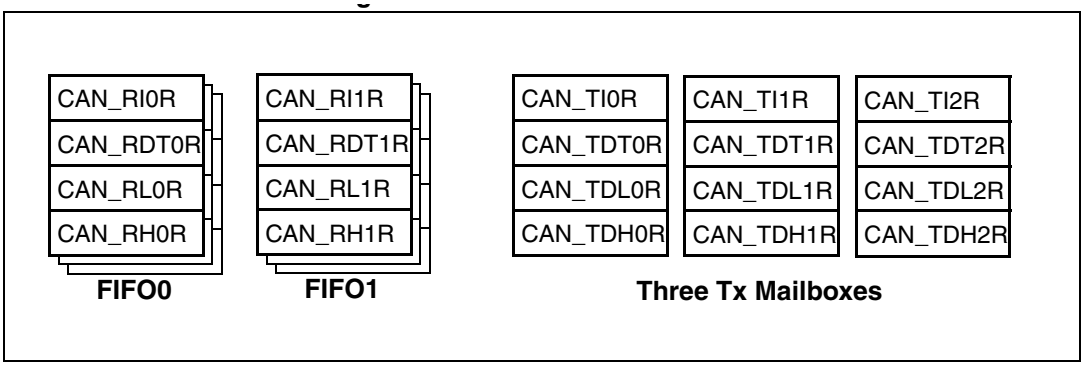
Передающие и приёмные ящики имеют одинаковые регистры, за исключением:

- Поля FMI в регистре CAN_RDTxR.
- Приёмный ящик защищён от записи всегда.
- Запись возможна только в пустой передающий ящик (стоит бит TME в регистре CAN_TSR).

Есть 3 TX ящика и 2 RX ящика. Ящик RX имеет трёхуровневый FIFO, доступно только самое старое сообщение.

Ящик состоит из 4.

Ящики RX и TX



Регистр идентификатора ящика TX (CAN_TIxR) (x=0..2)

Смещение адреса: 0x180, 0x190, 0x1A0

По сбросу: 0xXXXX XXXX (кроме бита 0, TXRQ = 0)

Регистры TX защищены от записи при сброшенном бите TMEx. Есть управление запросом передачи (бит 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]/EXID[28:18]											EXID[17:13]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXID[12:0]													IDE	RTR	TXRQ
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:21 **STID[10:0]/EXID[28:18]**: Стандартный или расширенный идентификатор
Стандартный идентификатор или старшие биты расширенного (в зависимости от бита IDE)
— Биты 20:3 **EXID[17:0]**: Младшие биты расширенного идентификатора
— Бит 2 **IDE**: Расширение идентификатора, 0 - стандартный, 1 - расширенный.
— Бит 1 **RTR**: Запрос удалённой передачи, 0 - фрейм данных, 1 - удалённый фрейм
— Бит 0 **TXRQ**: Запрос передачи ящика.
Ставится программно, снимается аппаратно при опустошении ящика.

Регистр длины данных и меток времени (CAN_TDTxR) (x=0..2)

Смещение адреса: 0x184, 0x194, 0x1A4

По сбросу: 0xXXXX XXXX

Регистр защищён от записи при непустом ящике.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							TGT	Reserved				DLC[3:0]			
							rw					rw	rw	rw	rw

— Биты 31:16 **TIME[15:0]**: Метка времени сообщения
16-бит значение таймера на момент передачи SOF.
— Биты 15:9 Резерв, не трогать
— Бит 8 **TGT**: Глобальное время передачи
Активен только в режиме временного разделения каналов, стоит бит TTCM регистра CAN_MCR.
0: Метка времени TIME[15:0] не посылается.
1: Метка времени TIME[15:0] посылается в двух последних байтах 8-байт сообщения: TIME[7:0] в байте 7 и TIME[15:8] в байте 6 данных, заменяя данные из регистра CAN_TDHxR[31:16] (DATA6[7:0] и DATA7[7:0]). DLC должно содержать число 8 для пересылки этих байтов по шине CAN.
— Биты 7:4 Резерв, не трогать
— Биты 3:0 **DLC[3:0]**: Код длины данных
Содержит число байтов данных в фрейме данных или запросе удалённого фрейма. Сообщение может содержать от 0 до 8 байтов данных, в зависимости от поля DLC.

Младший регистр данных ящика (CAN_TDLxR) (x=0..2)

Смещение адреса: 0x188, 0x198, 0x1A8
По сбросу: 0xXXXX XXXX
Регистр защищён от записи при непустом ящике.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 **DATA3[7:0]**: Байт данных 3.
— Биты 23:16 **DATA2[7:0]**: Байт данных 2.
— Биты 15:8 **DATA1[7:0]**: Байт данных 1.
— Биты 7:0 **DATA0[7:0]**: Байт данных 0.

Старший регистр данных ящика (CAN_TDLxR) (x=0..2)

Смещение адреса: 0x18C, 0x19C, 0x1AC
По сбросу: 0xXXXX XXXX
Регистр защищён от записи при непустом ящике. Байты 6 и 7 могут заменяться меткой времени.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 **DATA7[7:0]**: Байт данных 7.
— Биты 23:16 **DATA6[7:0]**: Байт данных 6.
— Биты 15:8 **DATA5[7:0]**: Байт данных 5.
— Биты 7:0 **DATA4[7:0]**: Байт данных 4.

Регистр идентификатора приёмного ящика (CAN_RIxR) (x=0..1)

Смещение адреса: 0x1B0, 0x1C0
По сбросу: 0xXXXX XXXX
Регистры RX защищены от записи.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]/EXID[28:18]											EXID[17:13]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXID[12:0]													IDE	RTR	Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

— Биты 31:21 **STID[10:0]/EXID[28:18]**: Стандартный или расширенный идентификатор
Стандартный идентификатор или старшие биты расширенного (в зависимости от бита IDE)
— Биты 20:3 **EXID[17:0]**: Младшие биты расширенного идентификатора
— Бит 2 **IDE**: Расширение идентификатора, 0 - стандартный, 1 - расширенный.
— Бит 1 **RTR**: Запрос удалённой передачи, 0 - фрейм данных, 1 - удалённый фрейм
— Бит 0 Резерв, не трогать

Регистр длины данных и меток времени (CAN_RDTxR) (x=0..2)

Смещение адреса: 0x1B4, 0x1C4
По сбросу: 0xXXXX XXXX
Регистры RX защищены от записи.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMI[7:0]								Reserved				DLC[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r

— Биты 31:16 **TIME[15:0]**: Метка времени сообщения
16-бит значение таймера на момент появления SOF.
— Биты 15:8 **FMI[7:0]**: Индекс фильтра, пропустившего сообщение.
— Биты 7:4 Резерв, не трогать
— Биты 3:0 **DLC[3:0]**: Код длины данных
Содержит число байтов данных в фрейме данных или 0 при запросе удалённого фрейма.

Младший регистр данных ящика (CAN_RDLxR) (x=0..1)

Смещение адреса: 0x1B8, 0x1C8
По сбросу: 0xXXXX XXXX
Регистры RX защищены от записи.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 **DATA3[7:0]**: Байт данных 3.
— Биты 23:16 **DATA2[7:0]**: Байт данных 2.
— Биты 15:8 **DATA1[7:0]**: Байт данных 1.
— Биты 7:0 **DATA0[7:0]**: Байт данных 0.

Старший регистр данных ящика (CAN_RDLxR) (x=0..1)

Смещение адреса: 0x1BC, 0x1CC
По сбросу: 0xXXXX XXXX
Регистры RX защищены от записи.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 **DATA7[7:0]**: Байт данных 7.
— Биты 23:16 **DATA6[7:0]**: Байт данных 6.
— Биты 15:8 **DATA5[7:0]**: Байт данных 5.
— Биты 7:0 **DATA4[7:0]**: Байт данных 4.

24.9.4. Регистры фильтров CAN

Главный регистр фильтров (CAN_FMR)

Смещение адреса: 0x200

По сбросу: 0x2A1C 0E01

Все биты ставятся и снимаются программно.

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		CAN2SB[5:0]						Reserved							FINIT
		rw	rw	rw	rw	rw	rw								rw

— Биты 31:14 Резерв, не трогать
— Биты 13:8 **CAN2SB[5:0]**: Стартовый банк CAN2
Стартовый банк ведомого CAN2 в диапазоне от 0 до 27.
NB: При CAN2SB[5:0] = 28d можно использовать фильтры CAN1.
При CAN2SB[5:0] = 0, фильтров CAN1 нет.
— Биты 7:1 Резерв, не трогать
— Бит 0 **FINIT**: Режим инициализации фильтров
0: Нормальная работа.
1: Инициализация.

Регистр режима фильтров (CAN_FM1R)

Смещение адреса: 0x204

По сбросу: 0x0000 0000

Писать можно только в режиме инициализации (FINIT=1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FBM27	FBM26	FBM25	FBM24	FBM23	FBM22	FBM21	FBM20	FBM19	FBM18	FBM17	FBM16
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FBM15	FBM14	FBM13	FBM12	FBM11	FBM10	FBM9	FBM8	FBM7	FBM6	FBM5	FBM4	FBM3	FBM2	FBM1	FBM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:28 Резерв, не трогать
— Биты 27:0 **FBM*x***: Режим фильтра
0: Два 32-бит регистра банка в режиме Маски.
1: Два 32-бит регистра банка в режиме Списка идентификаторов.
NB: Биты 27:14 доступны только в сетевых устройствах, иначе - резерв.

Регистр масштаба фильтров (CAN_FS1R)

Смещение адреса: 0x20C

По сбросу: 0x0000 0000

Писать можно только в режиме инициализации (FINIT=1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FSC27	FSC26	FSC25	FSC24	FSC23	FSC22	FSC21	FSC20	FSC19	FSC18	FSC17	FSC16
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSC15	FSC14	FSC13	FSC12	FSC11	FSC10	FSC9	FSC8	FSC7	FSC6	FSC5	FSC4	FSC3	FSC2	FSC1	FSC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:28 Резерв, не трогать

— Биты 27:0 **FSC*x***: Конфигурация масштаба фильтров 13 - 0

0: Парный 16-бит.

1: Одинарный 32-бит.

NB: Биты 27:14 доступны только в сетевых устройствах, иначе - резерв.

Регистр назначения фильтров (CAN_FFA1R)

Смещение адреса: 0x214

По сбросу: 0x0000 0000

Писать можно только в режиме инициализации (FINIT=1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FFA27	FFA26	FFA25	FFA24	FFA23	FFA22	FFA21	FFA20	FFA19	FFA18	FFA17	FFA16
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FFA15	FFA14	FFA13	FFA12	FFA11	FFA10	FFA9	FFA8	FFA7	FFA6	FFA5	FFA4	FFA3	FFA2	FFA1	FFA0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:28 Резерв, не трогать

— Биты 27:0 **FFA*x***: Назначение FIFO фильтру x.

0: FIFO 0.

1: FIFO 1.

NB: Биты 27:14 доступны только в сетевых устройствах, иначе - резерв.

Регистр активации фильтров (CAN_FFA1R)

Смещение адреса: 0x21C

По сбросу: 0x0000 0000

Писать можно только в режиме инициализации (FINIT=1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FACT27	FACT26	FACT25	FACT24	FACT23	FACT22	FACT21	FACT20	FACT19	FACT18	FACT17	FACT16
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FACT15	FACT14	FACT13	FACT12	FACT11	FACT10	FACT9	FACT8	FACT7	FACT6	FACT5	FACT4	FACT3	FACT2	FACT1	FACT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:28 Резерв, не трогать

— Биты 27:0 **FACT*x***: Активация фильтра x.

0: Не активен.

1: Активен.

NB: Биты 27:14 доступны только в сетевых устройствах, иначе - резерв.

Регистр x банка фильтров i (CAN_FiRx) (i=0..27, x=1, 2)

Смещение адреса: 0x240..0x31C

По сбросу: 0xXXXX XXXX

В сетевых устройствах есть 28 банков фильтров, i=0 .. 27, в прочих устройствах есть 14 банков фильтров i = 0 ..13. Банк состоит из двух 32-бит регистров, CAN_FiR[2:1].

Изменять этот регистр можно только при чистом бите FACTx в регистре CAN_FAxR или стоит бит FINIT регистра CAN_FMR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **FB[31:0]**: Биты фильтров

Идентификатор

Каждый бит определяет уровень соответствующего бита ожидаемого идентификатора.

0: Ожидается доминантный бит

1: Ожидается рецессивный бит

Маска

Каждый бит определяет необходимость сравнения соответствующего бита с ожидаемым идентификатором.

0: Бит не нужен

1: Сравниваем. Уровень поступающего бита должен совпадать с уровнем соответствующего бита в регистре идентификатора.

В зависимости от конфигурации масштаба и режима, функции регистров могут различаться. См. Секцию 24.7.4. и Таблицу 181.

24.9.5. Карта регистров CAN

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CAN_MCR	Reserved															DBF	RESET	Reserved							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
	Reset value																1	0								0	0	0	0	0	0	1	0
0x004	CAN_MSR	Reserved																				RX	SAMP	RXM	TXM	Res.			SLAKI	WKUI	ERRI	SLAK	INAK
	Reset value																					1	1	0	0				0	0	0	1	0
0x008	CAN_TSR	LOW[2:0]			TME[2:0]			CODE[1:0]		ABRQ2	Res.			TERR2	ALST2	TXOK2	RQCP2	ABRQ1	Res.			TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Res..			TERR0	ALST0	TXOK0	RQCP0
	Reset value	0	0	0	1	1	1	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0
0x00C	CAN_RF0R	Reserved																										RFOM0	FOVR0	FULL0	Reserved	FMP0[1:0]	
	Reset value																											0	0	0		0	0
0x010	CAN_RF1R	Reserved																										RFOM1	FOVR1	FULL1	Reserved	FMP1[1:0]	
	Reset value																											0	0	0		0	0
0x014	CAN_IER	Reserved														SLKIE	WKUIE	ERRIE	Res.			LECIE	BOFIE	EPVIE	EWGIE	Reserved	FOVIE1	FFIE1	FMPIE1	FOVIE0	FFIE0	FMPIE0	TMEIE
	Reset value															0	0	0				0	0	0	0		0	0	0	0	0	0	0
0x018	CAN_ESR	REC[7:0]								TEC[7:0]									Reserved								LEC[2:0]			Reserved	BOFF	EPVF	EWGF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0	0	0		0	0	0
0x01C	CAN_BTR	SILM	LBKM	Reserved				SJW[1:0]		Reserved	TS2[2:0]			TS1[3:0]					Reserved							BRP[9:0]							
	Reset value	0	0					0	0		0	1	0	0	0	1	1									0	0	0	0	0	0	0	0
0x020-0x17F	Reserved																																
0x180	CAN_TI0R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	TXRQ
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
0x184	CAN_TDT0R	TIME[15:0]																	Reserved							TGT	Reserved				DLC[3:0]		
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x									x					x	x	x
0x188	CAN_TDL0R	DATA3[7:0]								DATA2[7:0]									DATA1[7:0]															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18C	CAN_TDH0R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x190	CAN_TI1R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	TXRQ
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
0x194	CAN_TDT1R	TIME[15:0]																Reserved							TGT	Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
0x198	CAN_TDL1R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x19C	CAN_TDH1R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1A0	CAN_TI2R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	TXRQ
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
0x1A4	CAN_TDT2R	TIME[15:0]																Reserved							TGT	Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
0x1A8	CAN_TDL2R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1AC	CAN_TDH2R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1B0	CAN_RI0R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	Reserved
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
0x1B4	CAN_RDT0R	TIME[15:0]																FMI[7:0]								Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x
0x1B8	CAN_RDL0R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1BC	CAN_RDH0R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1C0	CAN_RI1R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	Reserved
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C4	CAN_RDT1R	TIME[15:0]																FMI[7:0]								Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x
0x1C8	CAN_RDL1R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1CC	CAN_RDH1R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1D0-0x1FF	Reserved																																
0x200	CAN_FMR	Reserved																		CAN2SB[5:0]						Reserved							FINIT
	Reset value																			0	0	1	1	1	0								1
0x204	CAN_FM1R	Reserved								FBM[27:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x208	Reserved																																
0x20C	CAN_FS1R	Reserved								FSC[27:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x210	Reserved																																
0x214	CAN_FFA1R	Reserved								FFA[27:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x218	Reserved																																
0x21C	CAN_FA1R	Reserved								FACT[27:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x220	Reserved																																
0x224-0x23F	Reserved																																
0x240	CAN_F0R1	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x244	CAN_F0R2	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x248	CAN_F1R1	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x24C	CAN_F1R2	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
. . . .	. . . .	. . . .																															
0x318	CAN_F27R1	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x31C	CAN_F27R2	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

25. Последовательный интерфейс (SPI)

25.1. Введение в SPI

В устройствах высокой-, XL-плотности и сетевых, интерфейс SPI выполняет две функции: протокол SPI или звуковой протокол I²S. По умолчанию выбран SPI. С SPI на I²S переключаются программно.

В устройствах Cat.1 и Cat.2 протокола I²S нет.

Интерфейс SPI обеспечивает полу-/полно- дуплексную синхронную последовательную связь с внешними устройствами. SPI можно сконфигурировать ведущим с подачей тактов связи (SCK) внешнему ведомому устройству. Интерфейс может работать в режиме конфигурации нескольких ведущих.

I²S это тоже синхронный последовательный интерфейс. Он может использовать четыре звуковых стандарта: I²S Philips, стандарты с MSB- и LSB-выравниванием и стандарт PCM. Он может работать ведомым и ведущим в полном дуплексе (4 ножки) или полу-дуплексе (6 ножек). При работе ведущим можно тактировать ведомого.

Внимание:

Некоторые ножки SPI3/I2S3 объединены с ножками JTAG (SPI3_NSS/I2S3_WS с JTDI и SPI3_SCK/I2S3_CK и JTDO), они не управляются контроллером IO и резервируются для JTAG (после каждого Сброса each).

Поэтому перед конфигурацией ножек SPI3/I2S3 надо отключать JTAG и при отладке использовать SWD, либо отключать оба интерфейса JTAG/SWD.

25.2. Основные свойства SPI и I²S

25.2.1. Свойства SPI

- Синхронный полный дуплекс по трём линиям
- Синхронный симплекс по двум линиям с/без двунаправленной линией данных
- 8- или 16-бит формат фрейма передачи
- Ведущий или ведомый
- Режим нескольких ведущих
- 8 предделителей скорости ведущего (f_{PCLK}/2 max.)
- Частота режима ведомого (f_{PCLK}/2 max)
- Ускоренная связь ведущего и ведомого
- Аппаратное и программное управление NSS для ведущего и ведомого: динамическая смена операций ведущий/ведомый
- Программируемая полярность и фаза тактов
- Программируемый порядок данных: первым старший или младший бит
- Отдельные флаги передачи и приёма с прерываниями
- Флаг занятости шины SPI
- Аппаратный CRC:
  - CRC можно передавать как последний байт в режиме Tx
  - Автоматическая проверка CRC для последнего принятого байта
- Флаги сбоя режима ведущего, переполнения и ошибки CRC с прерываниями
- 1-байт буфер приёма и передачи с: запросы Tx и Rx

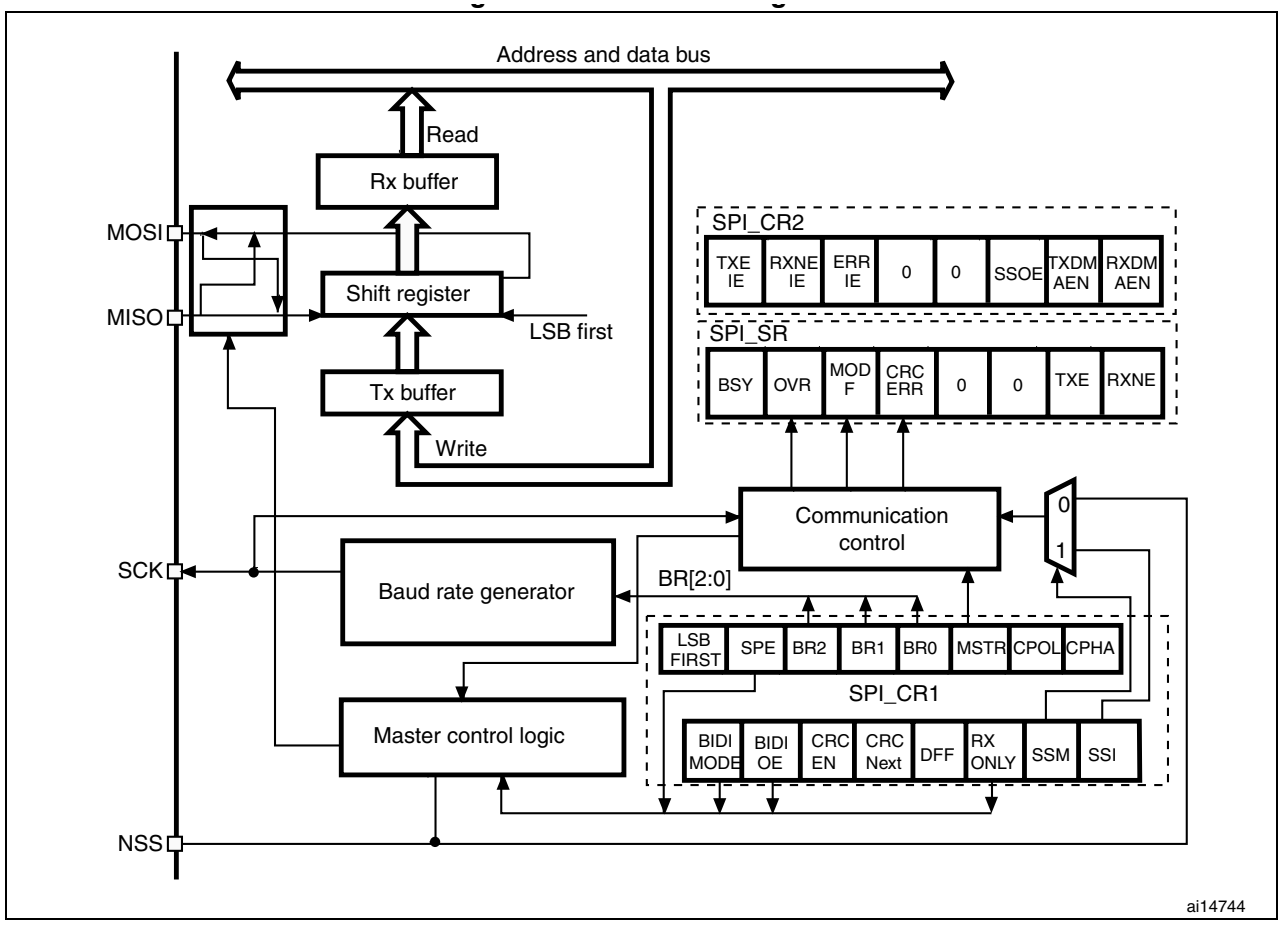
25.2.2. Свойства I²S

- Полу-дуплексная связь (только передача или приём)
- Ведущий или ведомый
- 8-бит программируемый предделитель частоты дискретизации звукового сигнала (8 - 192 kHz)
- Формат данных: 16-, 24- или 32-бит

- Фиксированный звуковым каналом пакет фрейма: 16-бит (16-бит фрейм данных) или 32-бит (16-, 24-, 32-бит фрейм данных)
- Программируемая полярность тактов (устойчивое состояние)
- Флаг исчерпания в режиме передачи ведомого и флаг переполнения в режиме приёма (ведущий и ведомый)
- 16-bit register for transmission and reception with one data register for both channel sides
- Поддерживаемые протоколы I^2S:
  - Стандарт I^2S Philips
  - MSB-выравненный (влево)
  - LSB-выравненный (вправо)
  - Стандарт PCM (с коротким или длинным фреймом синхронизации для 16-бит фрейма канала или 16-бит фрейма данных, расширенного до 32-бит фрейма канала)
- Передача начиная с MSB
- DMA передача или приём (16-бит)
- Такты ведущего могут подаваться на внешнее звуковое устройство.
  Частота фиксирована на 256 × F_S (где F_S частота дискретизации звука)
- В сетевых устройствах оба I^2S (I2S2 и I2S3) имеют отдельный PLL тактирования (PLL3).

25.3. Функциональное описание SPI

25.3.1. Общее описание

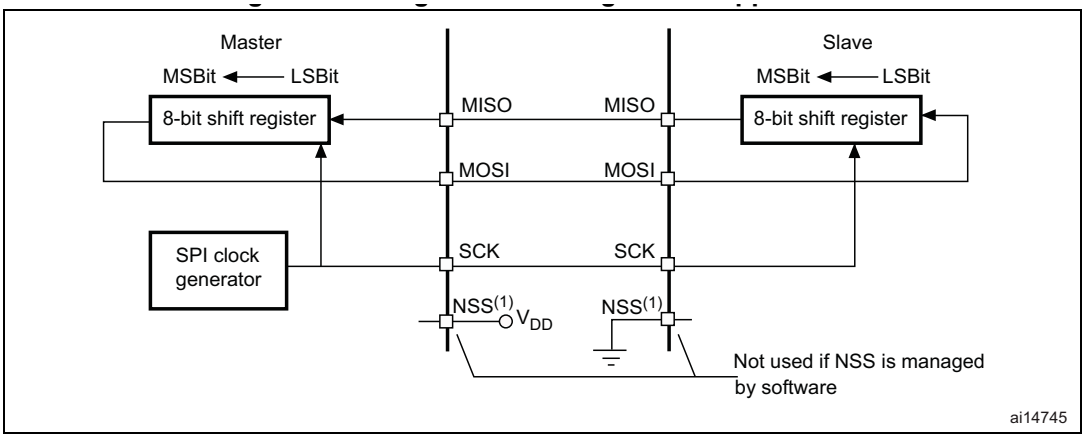


Обычно SPI подключается к внешним устройствам четырьмя ножками:

- MISO: Приём данных ведущего и передача ведомого.
- MOSI: Приём данных ведомого и передача ведущего.
- SCK: Выход тактов ведущего и вход ведомого.
- NSS: Необязательная ножка выбора ведомого. Похоже на выбор чипа, входами NSS ведомых можно управлять через стандартные IO порты. Ножку NSS можно сделать выходом ведущего

(бит SSOE) и выдавать активным низким. Теперь все подключённые сюда ножки NSS видят низкий уровень ведущего и становятся ведомыми, но конфигурировать нужно в аппаратном режиме. Если сделать NSS входом в режиме ведущего (MSTR=1 и SSOE=0), то при обнаружении на ней низкого уровня SPI уходит в ошибку режима ведущего: бит MSTR снимается и SPI переключается в ведомый.

Один ведущий / Один ведомый



1. Здесь ножка NSS это вход.

Ножки MOSI, MISO и SCK соединяются попарно. Передача инициируется ведущим. Данные по MOSI и ответы по MISO синхронизируются по SCK. Это полный дуплекс.

Выбор ведомого (NSS)

Программный или аппаратный выбор ведомого определяется битом SSM регистра SPI_CR1.

- Программное управление NSS (SSM = 1)
  Выбирается битом SSI регистра SPI_CR1. Внешняя ножка NSS освобождается.
- Аппаратное управление NSS (SSM = 0)
  Есть две конфигурации, зависящие от разрешения вывода NSS (бит SSOE регистра SPI_CR2).
  – Вывод NSS разрешён (SSM = 0, SSOE = 1)
  Только для режима ведущего. NSS ставится низким вплоть до отключения SPI.
  – Вывод NSS запрещён (SSM = 0, SSOE = 0)
  Разрешает работу нескольких ведущих устройств. У ведомых устройств ножка NSS это обычный ввод NSS: ведомый выбирается при низком входе NSS.

Фаза и полярность тактов

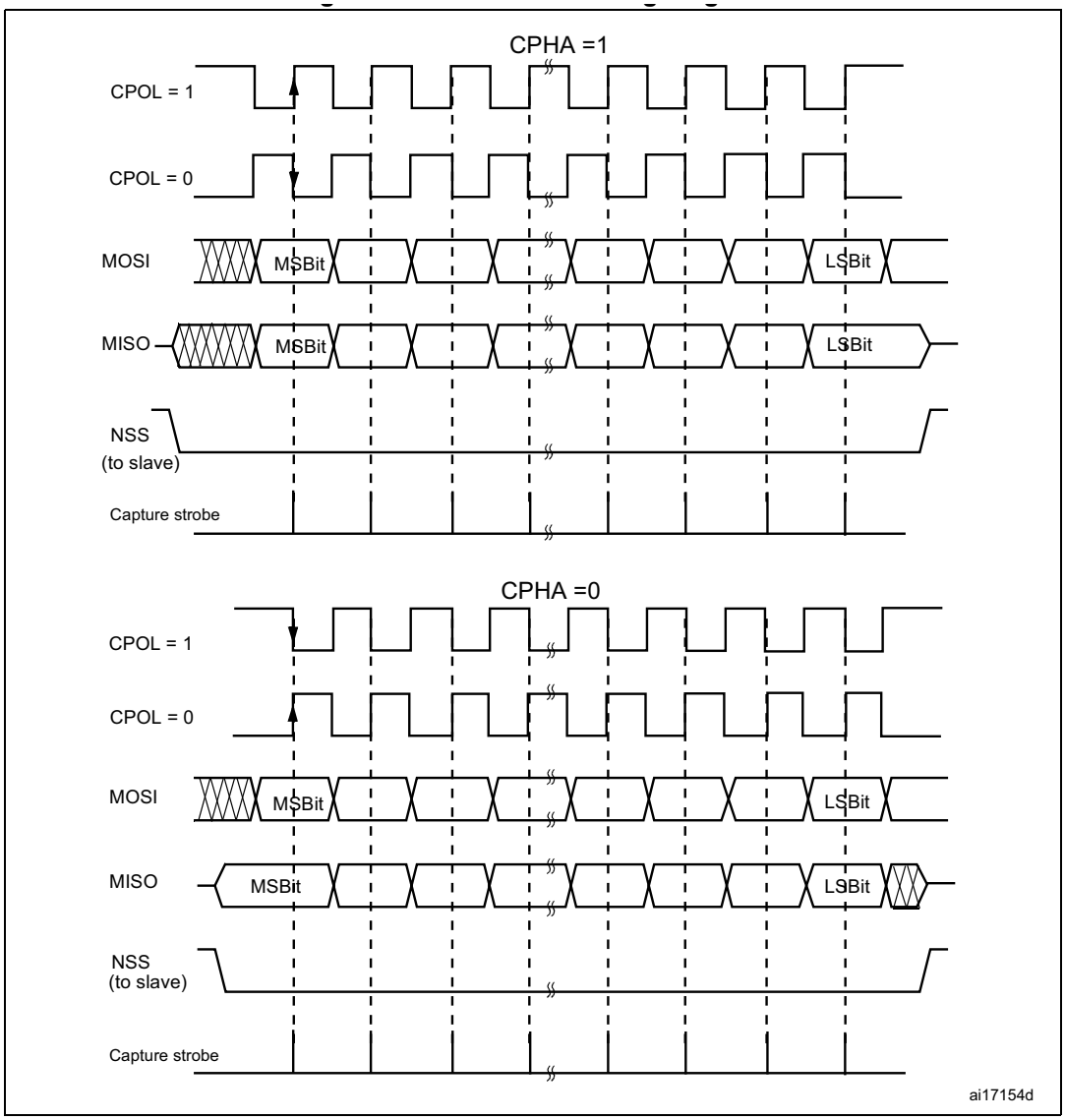
Биты CPOL и CPHA регистра SPI_CR1 определяют четыре возможных времянки. Бит CPOL (полярность) определяет значение устойчивого состояния тактового сигнала (данные не передаются). Действует и в ведущем и в ведомом режимах. Если CPOL сброшен, то SCK готовится низким, иначе - высоким.

Если бит CPHA (фаза) стоит, то передача бита стробируется вторым фронтом SCK (задним при снятом бите CPOL и передним при стоящем бите CPOL). Пришедший бит запоминается по второму тику. Если бит CPHA сброшен, то передача бита стробируется первым фронтом SCK (задним при стоящем бите CPOL и передним при снятом бите CPOL). Пришедший бит запоминается по первому же тику.

NB: Перед изменением битов CPOL/CPHA бит SPE надо снимать, отключая SPI. Ведомый и ведущий должны иметь одинаковый режим времянки.

Состояние простоя SCK должно соответствовать полярности из регистра SPI_CR1 (подпорка при CPOL=1 и подтяжка вниз при CPOL=0).

Диаграмма стробирования данных



1. Здесь бит LSBFIRST регистра SPI_CR1 сброшен.

Формат фрейма данных

Данные выдвигаются старшим или младшим битом вперёд в зависимости от бита `LSBFIRST` в регистре `SPI_CR1`. Длина фрейма данных (8- или 16-бит) выбирается битом `DFF` регистра `SPI_CR1`.

25.3.2. Ведомый SPI

Такты подаются на ножку SCK, биты `BR[2:0]` регистра `SPI_CR1` на частоту передачи не влияют.

NB: Включать ведомый SPI надо до подачи тактов. Регистр данных должен быть готов до первого фронта такта или до его конца. Ставить полярность тактов надо до включения ведущего и ведомого..

Конфигурация ведомого

1. Ставим бит `DFF` (8- или 16-бит данные)
2. Ставим биты `CPOL` и `CPHA` (такие же как для ведущего).
3. Ставим формат фрейма (первым старший или младший бит) битом `LSBFIRST`.
4. В Аппаратном режиме ножка NSS должна быть низкой во время передачи всего байта. В Программном режиме ставим бит `SSM` и снимаем бит `SSI` регистра `SPI_CR1`.
5. Чистим бит `MSTR` и ставим бит `SPE` регистра `SPI_CR1`, переведя ножки на альтернативную функцию.

В этом режиме ножка MOSI это ввод, а MISO это вывод.

Последовательность передачи

Байт данных пишется в буфер Tx параллельно.

Передача начинается по получению ведомым тактового сигнала при старшем значащем бите данных на ножке MOSI. Оставшиеся биты (7 или 15 бит) загружены в регистр сдвига. После передачи данных из буфера Tx ставится флаг TXE в регистре SPI_SR и при стоящем бите TXEIE в регистре SPI_CR2 выдаётся прерывание.

Последовательность приёма

По завершении передачи данных:

- Данные из регистра сдвига передаются в буфер Rx и ставится флаг RXNE в регистре SPI_SR
- При стоящем бите RXNEIE в регистре SPI_CR2 выдаётся прерывание.

После чтения регистра SPI_DR снимается бит RXNE и SPI возвращается в приём.

25.3.3. Ведущий SPI

Тактовый сигнал выдаётся на ножку SCK.

Конфигурация

1. Битами BR[2:0] ставим частоту тактов обмена.
2. Ставим биты CPOL и CPHA вариантов тактирования.
3. Определяем длину фрейма данных битом DFF
4. Определяем формат фрейма данных битом LSBFIRST регистра SPI_CR1.
5. Если ножка NSS нужна для ввода, то в аппаратном режиме подключаем NSS к высокому уровню на всё время передачи байта, в программном режиме ставим биты SSM и SSI в регистре SPI_CR1. Если ножка NSS нужна для вывода, то ставить надо только бит SSOE.
6. Биты MSTR и SPE должны стоять (они остаются стоять только при подключении NSS к высокому уровню).

Сейчас MOSI это вывод, а MISO это ввод.

Последовательность передачи

Начинается записью байта в буфер Tx.

Во время передачи первого бита перегружается в регистр сдвига и выдаётся на ножку MOSI побитно нужным концом вперёд. После передачи данных из буфера Tx ставится флаг TXE в регистре SPI_SR и при стоящем бите TXEIE в регистре SPI_CR2 выдаётся прерывание.

Последовательность приёма

По завершении передачи данных:

- Данные из регистра сдвига передаются в буфер Rx и ставится флаг RXNE в регистре SPI_SR
- При стоящем бите RXNEIE в регистре SPI_CR2 выдаётся прерывание.

После чтения регистра SPI_DR снимается бит RXNE и SPI возвращается в приём.

Непрерывный поток передачи можно организовать записью очередного байта в буфер Tx поле начала передачи из регистра сдвига. При этом флаг TXE должен стоять.

NB: Для отключения ведомых SPI между передачами ножку NSS нужно конфигурировать как GPIO или использовать другой GPIO и переключать программно.

25.3.4. SPI в полу-дуплексе

Есть 2 конфигурации SPI в полу-дуплексе:

- 1 тактовый и 1 двунаправленный провод данных
- 1 тактовый и 1 провод данных (приём или передача)

1 тактовый и 1 двунаправленный провод данных (BIDIMODE=1)

Включается установкой бита BIDIMODE в регистре SPI_CR1. В этом режиме такты идут по SCK, а данные по MOSI ведущего или MISO ведомого. При стоящем бите BIDIOE регистра SPI_CR1 это линия вывода, иначе это ввод.

1 тактовый и 1 провод данных (BIDIMODE=0)

В этом режиме SPI работает только на ввод или на вывод.

- "Только передача" подобна полному дуплексу (BIDIMODE=0, RXONLY=0): данные передаются по передающей ножке (MOSI ведущего или MISO ведомого), а приёмную (MISO ведущего или MOSI ведомого) можно использовать как GPIO. В этом случае просто игнорируем буфер Rx.
- В "только приёме" отключаем вывод SPI установкой бита RXONLY регистра SPI_CR1. Теперь передающая ножка (MOSI ведущего или MISO ведомого) свободна для GPIO.

Запускаем "только ввод" конфигурируя и затем включая SPI:

- У ведущего ввод начинается немедленно и останавливается при очистке бита SPE и остановке текущего приёма. Читать флаг BSY нет нужды. Он всегда стоит при работающей операции SPI.
- У ведомого SPI продолжает принимать при низком NSS (или очистке бита SSI при программном управлении) и поступающих тактах SCK.

25.3.5. Приём и передача данных

Буферы Rx и Tx

При приёме данные из регистра сдвига пишутся в буфер Rx, при передаче записываемые в буфер Tx данные поступают в регистр сдвига. Чтение регистра SPI_DR выдаёт данное из Rx, а запись в SPI_DR пишет в буфер Tx.

Стартовая последовательность ведущего

- В полном дуплексе (BIDIMODE=0 и RXONLY=0)
  - Последовательность начинается записью данных в регистр SPI_DR (буфер Tx).
  - Данные переносятся из Tx в 8-бит регистр сдвига во время передачи первого бита и затем последовательно выдвигаются на ножку MOSI.
  - В то же время приёмные данные на ножке MISO последовательно вдвигаются в 8-бит регистр сдвига, а затем параллельно переносятся в регистр SPI_DR (буфер Rx).
- При однонаправленном приёме (BIDIMODE=0 и RXONLY=1)
  - Последовательность начинается установкой SPE=1
  - Активируется только приёмник и данные на ножке MISO последовательно вдвигаются в 8-бит регистр сдвига, а затем параллельно переносятся в регистр SPI_DR (буфер Rx).
- Двунаправленный режим, передача (BIDIMODE=1 и BIDIOE=1)
  - Последовательность начинается записью данных в регистр SPI_DR (буфер Tx).
  - Данные переносятся из Tx в 8-бит регистр сдвига во время передачи первого бита и затем последовательно выдвигаются на ножку MOSI.
  - Данные не принимаются.
- Двунаправленный режим, приём (BIDIMODE=1 и BIDIOE=0)
  - Последовательность начинается установкой SPE=1 и BIDIOE=0.
  - Приёмные данные на ножке MISO последовательно вдвигаются в 8-бит регистр сдвига, а затем параллельно переносятся в регистр SPI_DR (буфер Rx).
  - Передатчик не активируется и данные на ножку MOSI не поступают.

Стартовая последовательность ведомого

- Полный дуплекс (BIDIMODE=0 и RXONLY=0)
  - Последовательность начинается когда ведомый принимает тактовый сигнал и первый бит данных на своей ножке MOSI. Остальные 7 бит идут в регистр сдвига.
  - Одновременно, при передаче первого бита данные из буфера Tx пишутся в регистр сдвига и затем выдвигаются на ножку MISO. Писать передаваемые данные нужно до запуска передачи ведущим устройством SPI.
- При однонаправленном приёме (BIDIMODE=0 и RXONLY=1)
  - Последовательность начинается когда ведомый принимает тактовый сигнал и первый бит данных на своей ножке MOSI. Остальные 7 бит идут в регистр сдвига.
  - Передатчик не активируется и данные на ножку MOSI не поступают.
- Двунаправленный режим, передача (BIDIMODE=1 и BIDIOE=1)

– Последовательность начинается когда ведомый принимает тактовый сигнал и первый бит из буфера Tx передаётся на ножку MISO.
– Данные переносятся из Tx в 8-бит регистр сдвига во время передачи первого бита и затем последовательно выдвигаются на ножку MISO. Писать передаваемые данные нужно до запуска передачи ведущим устройством SPI.
– Данные не принимаются.

• Двунаправленный режим, приём (BIDIMODE=1 и BIDIOE=0)
– Последовательность начинается когда ведомый принимает тактовый сигнал и первый бит данных на своей ножке MISO.
– Приёмные данные на ножке MISO последовательно вдвигаются в 8-бит регистр сдвига, а затем параллельно переносятся в регистр SPI_DR (буфер Rx).
– Передатчик не активируется и данные на ножку MISO не поступают.

Обработка данных передачи и приёма

Флаг TXE (буфер Tx пуст) ставится после пересылки данных из буфера Tx в регистр сдвига. При стоящем бите TXEIE в регистре SPI_CR2 выдаётся прерывание. Чистится бит TXE записью в регистр SPI_DR. Запись в Tx при снятом флаге TXE заменяет хранящиеся там данные.

Флаг RXNE (буфер Rx не пуст) ставится по последнему фронту такта считывания при передаче данных из регистра сдвига в буфер Rx. Данные можно читать из SPI_DR. При стоящем бите RXNEIE в регистре SPI_CR2 выдаётся прерывание. Чистится бит RXNE чтением регистра SPI_DR.

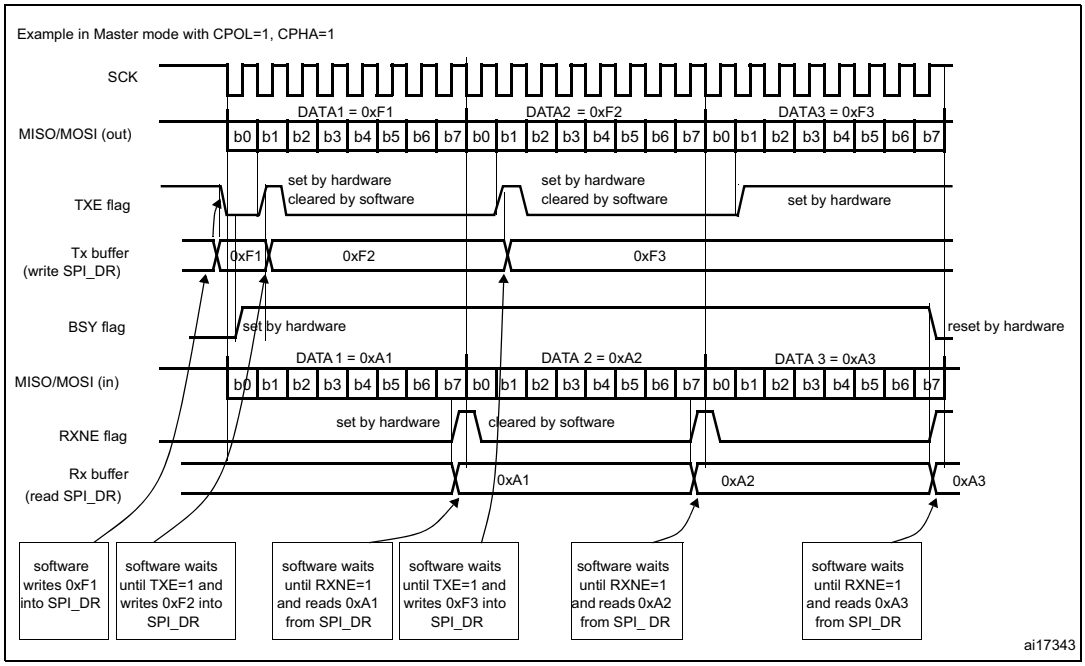
В некоторых конфигурациях для ожидания конца передачи последнего данного можно использовать флаг BSY.

При полном дуплексе BIDIMODE=0 и RXONLY=0.
Программная процедура:

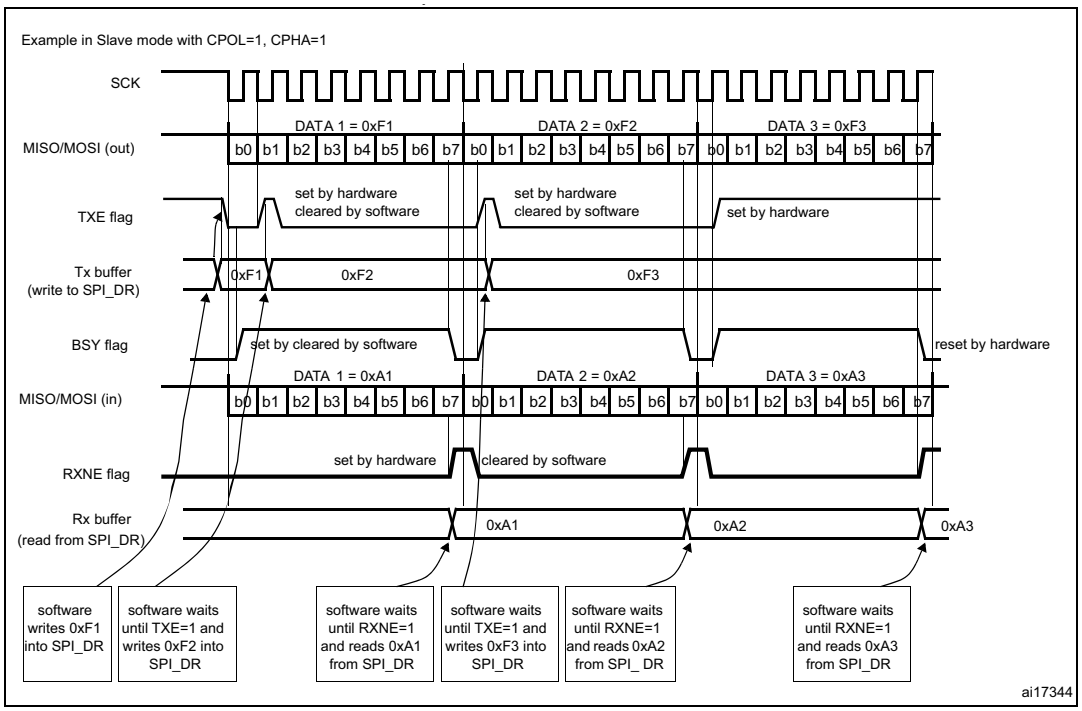
1. Включаем SPI установкой бита SPE.
2. Пишем первое передаваемое данное в регистр SPI_DR (флаг TXE снимается).
3. Ждём TXE=1 и пишем второе передаваемое данное. Теперь ждём RXNE=1 и читаем регистр SPI_DR с первым принимаемым данным (флаг RXNE снимается). Повторяем передачу/приём до получения n–1 данных.
4. Ждём RXNE=1 и читаем последнее данное.
5. Ждем TXE=1 и затем BSY=0 перед отключением SPI.

Всё это можно сделать с помощью программ прерывания по передним фронтам флагов RXNE и TXE.

Флаги TXE/RXNE/BSY ведущего/полный дуплекс (BIDIMODE=0 и RXONLY=0) при постоянной передаче



Флаги TXE/RXNE/BSY ведомого/полный дуплекс (BIDIMODE=0 и RXONLY=0) при постоянной передаче



Процедура Только передачи (BIDIMODE=0 RXONLY=0)

В этом режиме процедуру можно сократить и использовать бит BSY для ожидания конца передачи.

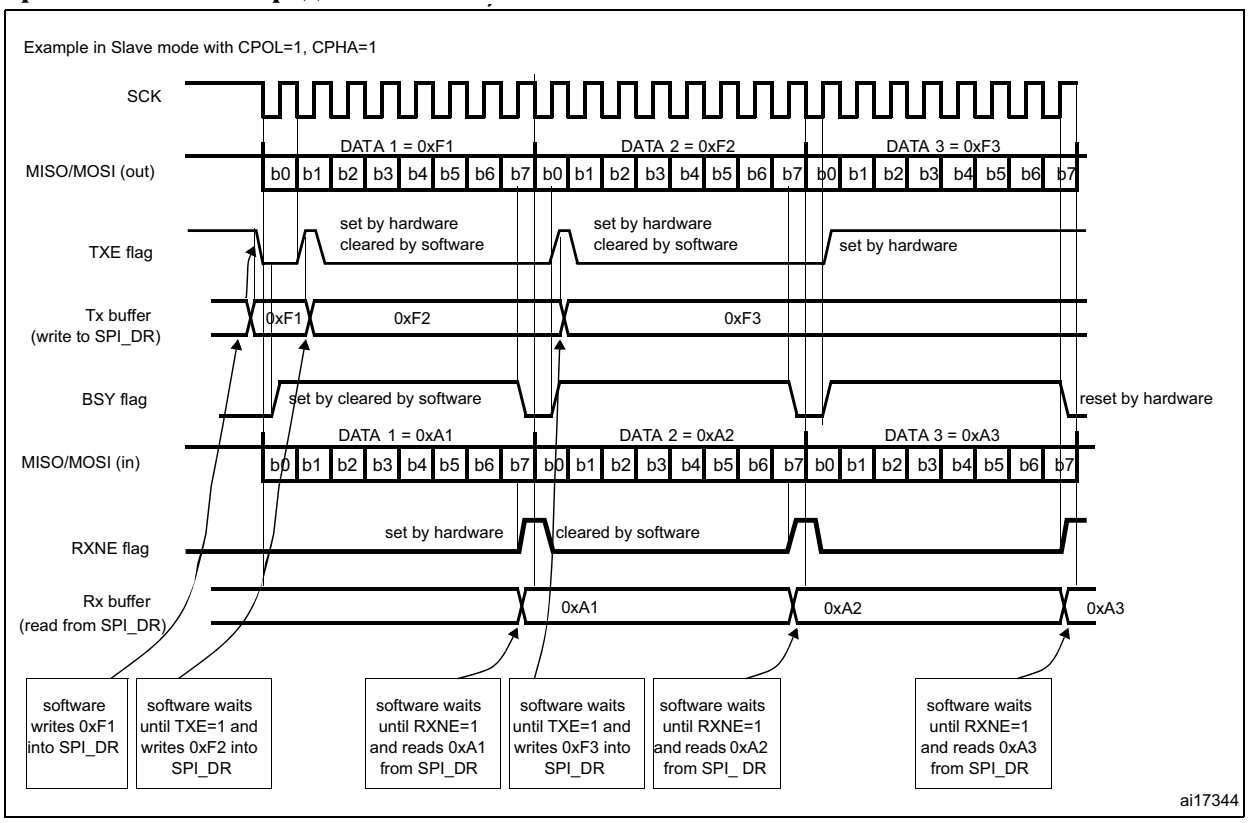
1. Включаем SPI установкой бита SPE.
2. Пишем первое посылаемое данное в регистр SPI_DR (битами TXE чистится).
3. Ждём TXE=1 и пишем следующее пересылаемое данное. Повторяем этот шаг для всех данных.
4. После записи последнего данного ждём TXE=1, и затем BSY=0 для определения конца передачи последнего данного.

Всё это можно сделать с помощью программ прерывания по переднему фронту флага TXE.

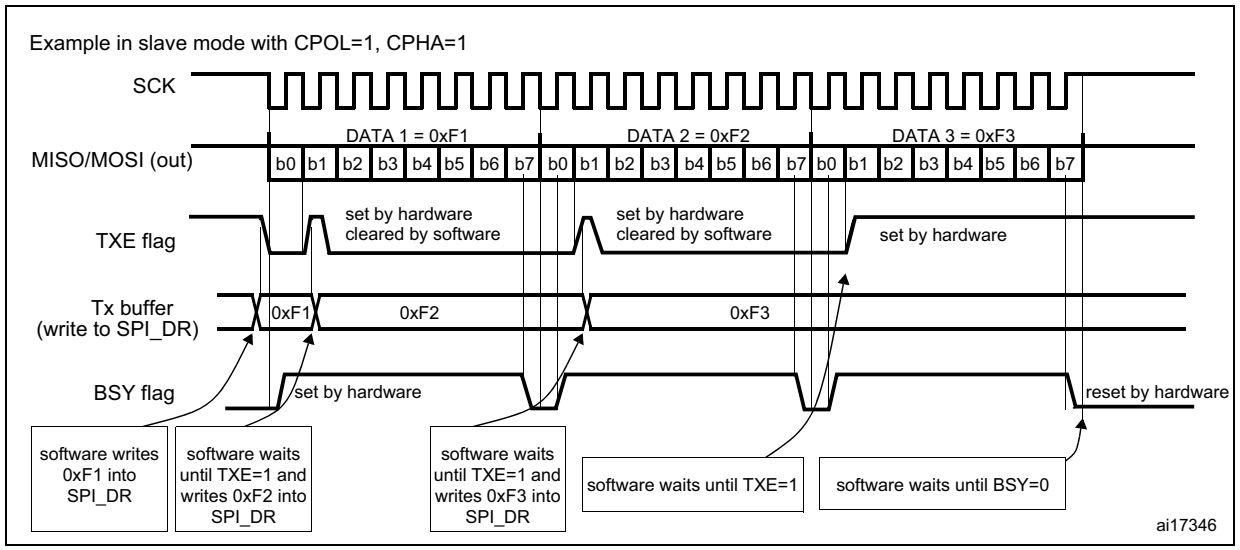
Во время прерывных передач между записью в SPI_DR и установкой бита BSY появляется задержка в 2 такта APB. Следовательно после записи последнего данного дожидаться установки бита TXE и затем снятия бита BSY нужно обязательно.

После передачи двух данных в режиме только передачи встаёт флаг OVR регистра SPI_SR, так как читать данные никто не намерен.

Флаги TXE/BSY ведущего/только передача (BIDIMODE=0 и RXONLY=0) при постоянной передаче



Флаги TXE/BSY ведомого/только передача (BIDIMODE=0 and RXONLY=0) при постоянной передаче



Процедура двунаправленной передачи (BIDIMODE=1 и BIDIOE=1)

Она подобна процедуре режима Только передачи, но перед включением SPI надо поставить биты `BIDIMODE` и `BIDIOE` регистра `SPI_CR2`.

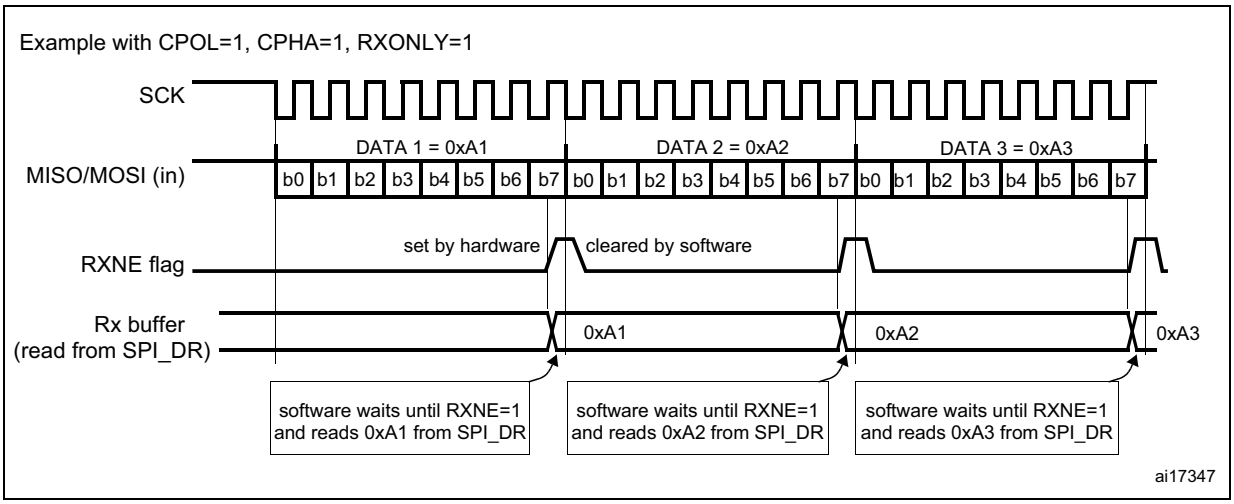
Процедура однонаправленного приёма (BIDIMODE=0 и RXONLY=1)

1. Ставим бит `RXONLY` в регистре `SPI_CR1`.
2. Включаем SPI установкой бита `SPE`:
   a) У ведущего немедленно активируется выдача тактов на SCK и приём последовательных данных вплоть до выключения SPI (`SPE`=0).
   b) У ведомого данные принимаются при низком NSS и появлении тактов SCK.
3. Ждём `RXNE`=1 и читаем регистр `SPI_DR` (бит `RXNE` снимается). Повторяем этот пункт.

Всё это можно сделать с помощью программ прерывания по переднему фронту флага `RXNE`.

После завершения всех передач SPI надо выключать.

Флаг RXNE только приём (BIDIRMODE=0 и RXONLY=1)
при постоянной передаче



Процедура двунаправленного приёма (BIDIMODE=1 и BIDIOE=0)

Она подобна процедуре режима Только приёма, но перед включением SPI надо поставить бит BIDIMODE и снять BIDIOE регистра SPI_CR2.

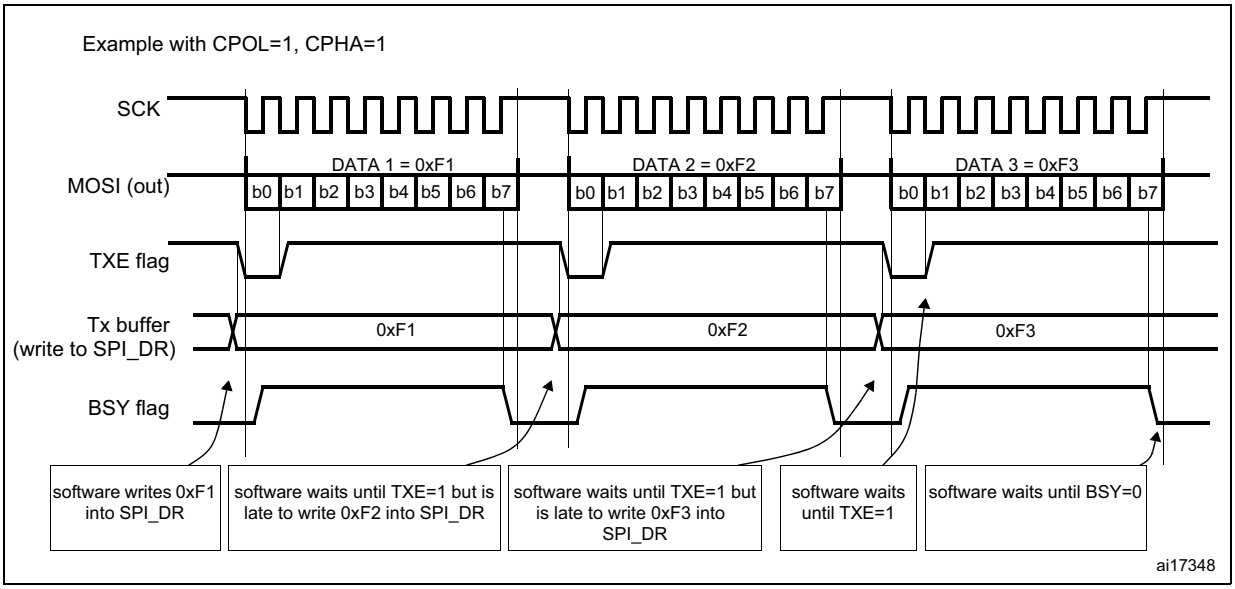
Непрерывные и прерывные передачи

Если в режиме ведущего программа успевает обнаружить каждый передний TXE (или прерывание TXE) и записать регистр SPI_DR до завершения текущей передачи, то это непрерывная передача: подача тактов SPI не прерывается и бит BSY между передачами данных не снимается. У прерывных передач бит BSY успевает сниматься.

В только приёме ведущего (RXONLY=1), связь непрерывная и BSY читается как 1.

У ведомого непрерывность определяется ведущим, но бит BSY всегда снимается хоть на один такт SPI.

Флаги TXE/BSY передача (BIDIRMODE=0 and RXONLY=0)
при прерывной передаче



25.3.6. Вычисление CRC

Для приёмных и передаваемых данных используются два отдельных калькулятора CRC. Он вычисляется по фронту считывания, в зависимости от битов CPHA и CPOL регистра SPI_CR1.

NB: Для разных данных используются два вида CRC: 8-бит (CR8) 16-бит (CRC16).

Вычисление CRC разрешается установкой бита CRCEN регистра SPI_CR1. Этим сбрасываются регистры CRC (SPI_RXCRCR и SPI_TXCRCR). При программном управлении в полном дуплексе и только передаче сразу после записи последнего данного в регистр SPI_DR надо записать бит CRCNEXT. Тогда после этой передачи будет передан регистр SPI_TXCRCR.

В только приёме и программном управлении бит CRCNEXT надо писать после второго принятого данного и CRC будет проверен.

По концу передачи CRC результат вычисления определяет флаг CRCERR в регистре SPI_SR.

При наличии данного в буфере TX значение CRC передаётся после передачи последнего байта. При передаче CRC калькулятор CRC выключается и значение регистра не изменяется.

Процедура такая:

1. Пишем значения CPOL, CPHA, LSBFIRST, BR, SSM, SSI и MSTR.
2. Пишем полином в регистр SPI_CRCPR.
3. Включаем вычисление CRC установкой бита CRCEN в регистре SPI_CR1. При этом чистятся регистры SPI_RXCRCR и SPI_TXCRCR.
4. Включаем SPI установкой бита SPE в регистре SPI_CR1.
5. Запускаем связь и поддерживаем её до предпоследнего байта.
   - При программном управлении в полном дуплексе или только передаче сразу после записи последнего байта в регистр SPI_DR, надо записать бит CRCNEXT. Тогда после этой передачи будет передан регистр SPI_TXCRCR.
   - В только приёме бит CRCNEXT надо ставить сразу после приёма предпоследнего данного, чтобы подготовить переход SPI в фазу CRC по концу приёма последнего данного. Вычисление CRC во время его передачи стопорится.
6. После передачи последнего байта или полуслова SPI уходит в фазу передачи и проверки CRC. В полном дуплексе или только приёме принятый CRC сравнивается с регистром SPI_RXCRCR. При несовпадении ставится флаг CRCERR в регистре SPI_SR и при стоящем бите ERRIE в регистре SPI_CR2 выдаётся прерывание.

В режиме ведомого включайте вычисление CRC только при стабильном такте (в устойчивом состоянии). Иначе можно вычислить не так. В действительности CRC чувствителен к входным тактам ведомого сразу после установки CRCEN, независимо от бита SPE.

Будьте внимательны с передачей CRC на высоких частотах. Число тактов CPU в фазе передачи CRC должно быть минимально возможным, так что в это время вызов функций запрещён, чтобы не терять последнее данное. В действительности бит CRCNEXT надо писать перед концом передачи/приёма последнего данного.

На высоких частотах передачи полезно использовать DMA.

У ведомых устройств в аппаратном режиме NSS, ножку NSS нужно держать низкой между фазами данных и CRC.

При разрешении функции CRC ведомых устройств, он вычисляется даже при высокой ножке NSS.

Во время переключения выбора NSS значения CRC нужно сбрасывать на ведущей и ведомой стороне.

Процедура очистки CRC такова:

1. Выключаем SPI (SPE = 0)
2. Снимаем бит CRCEN
3. Ставим бит CRCEN
4. Включаем SPI (SPE = 1)

25.3.7. Флаги состояния

Есть четыре флага состояния:

Флаг пустого буфера Tx (TXE)

Ставится аппаратно. Снимается записью в регистр SPI_DR.

Флаг занятого буфера Rx (RXNE)

Ставится аппаратно. Снимается чтением регистра SPI_DR.

Флаг BUSY

Флаг BSY ставится и снимается аппаратно при занятом и свободном уровне связи SPI соответственно. Исключение составляет двунаправленный приём ведущего (MSTR=1 и BDM=1 и BDOE=0), когда флаг BSY остаётся низким во время приёма. Полезен для обнаружения конца

передачи перед остановкой SPI (или выключения тактов устройства). Так не нарушится последняя передача. Для этого надо соблюсти описанную ниже процедуру.

Также флаг BSY полезен для исключения коллизий записи в системе с несколькими ведомыми.

Ставится в начале передачи кроме режима с (MSTR=1, BDM=1 и BDOE=0).

Снимается:

- в конце передачи (кроме непрерывных передач ведущего)
- при выключении SPI
- при сбое режима ведущего (MODF=1)

У прерывных передач флаг BSY снимается между всеми передачами.

При непрерывных передачах:

- у ведущего флаг BSY остаётся высоким во время всех передач
- у ведомого флаг BSY снимается на один такт между передачами

NB: Не используйте флаг BSY для обработки конца передачи, флаги TXE и RXNE гораздо лучше.

25.3.8. Отключение SPI

Для этого снимается бит SPE.

В некоторых конфигурациях выключение SPI с незавершённой передачей может нарушить её и исказить бит BSY. Так что нужно соблюсти процедуры:

В полном дуплексе ведущего или ведомого (BIDIMODE=0, RXONLY=0)

1. Ждём RXNE=1 для последнего данного
2. Ждём TXE=1
3. Ждём BSY=0
4. Выключаем SPI (SPE=0)

Однонаправленная только передача ведущего или ведомого (BIDIMODE=0, RXONLY=0) или двунаправленная передача (BIDIMODE=1, BIDIOE=1)

После записи последнего данного в регистр SPI_DR:

1. Ждём TXE=1
2. Ждём BSY=0
3. Выключаем SPI (SPE=0)

Однонаправленный только приём ведущего (MSTR=1, BIDIMODE=0, RXONLY=1) или двунаправленный приём (MSTR=1, BIDIMODE=1, BIDIOE=0)

Тут надо убедиться, что SPI не начал новую передачу:

1. Ждём предпоследнего появления RXNE=1 (n–1)
2. Программным циклом ждём 1 такт SPI перед его выключением (SPE=0)
3. Теперь ждём RXNE=1 перед остановом

NB: У ведущего при двунаправленном приёме (MSTR=1 and BDM=1 and BDOE=0) флаг BSY остаётся высоким во время всех передач.

Ведомый только приём (MSTR=0, BIDIMODE=0, RXONLY=1) или двунаправленный приём (MSTR=0, BIDIMODE=1, BIDOE=0)

1. Выключать SPI (SPE=1) можно когда угодно: сначала закончится текущая передача
2. Перед остановом дождитесь, BSY = 0.

25.3.9. SPI с DMA

Доступ с DMA включается отдельными битами регистра SPI_CR2 для запросов Tx и Rx буферов:

- При передаче запрос DMA выдаётся по установке TXE. Запись DMA в SPI_DR чистит его.
- При приёме запрос DMA выдаётся по установке RXNE. Чтение DMA из SPI_DR чистит его.

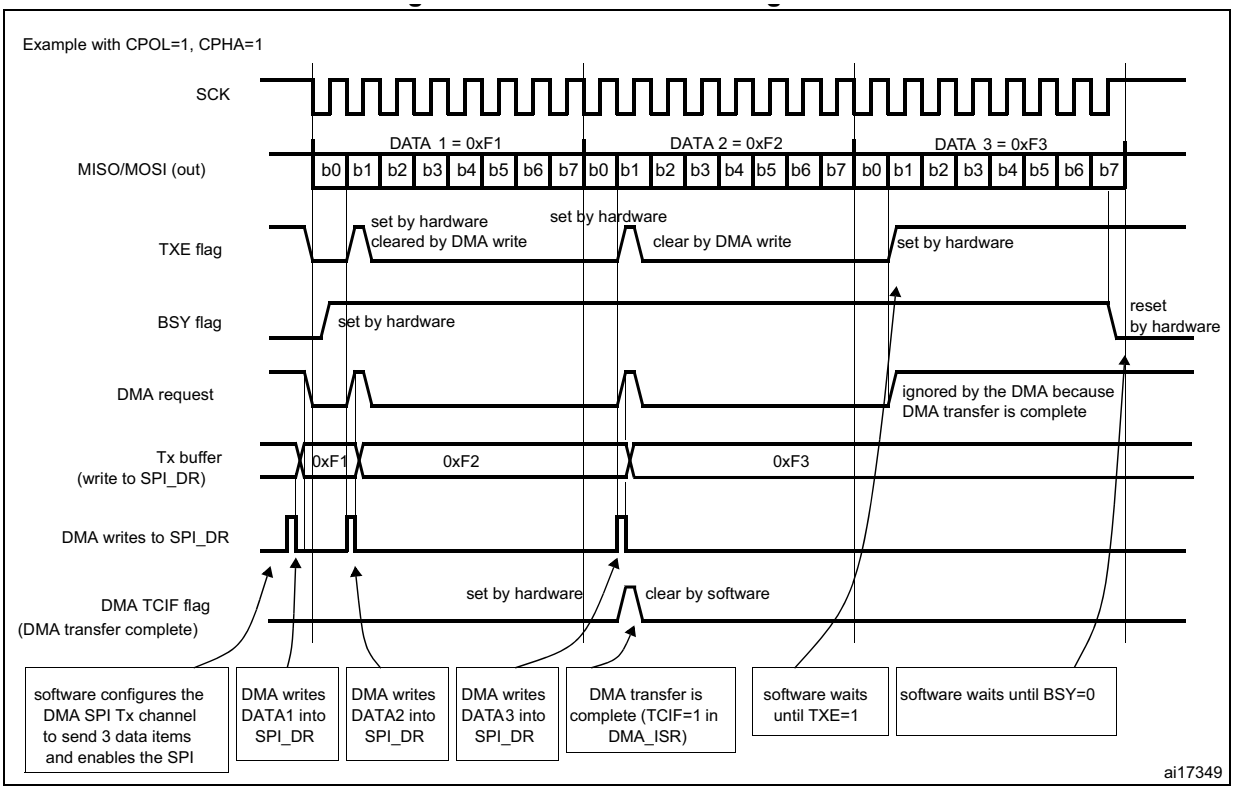
При работе только в одном направлении можно включать только один канал DMA (Tx или Rx).

При работе SPI только на передачу флаг OVR стоит, поскольку принимаемые данные не читаются.

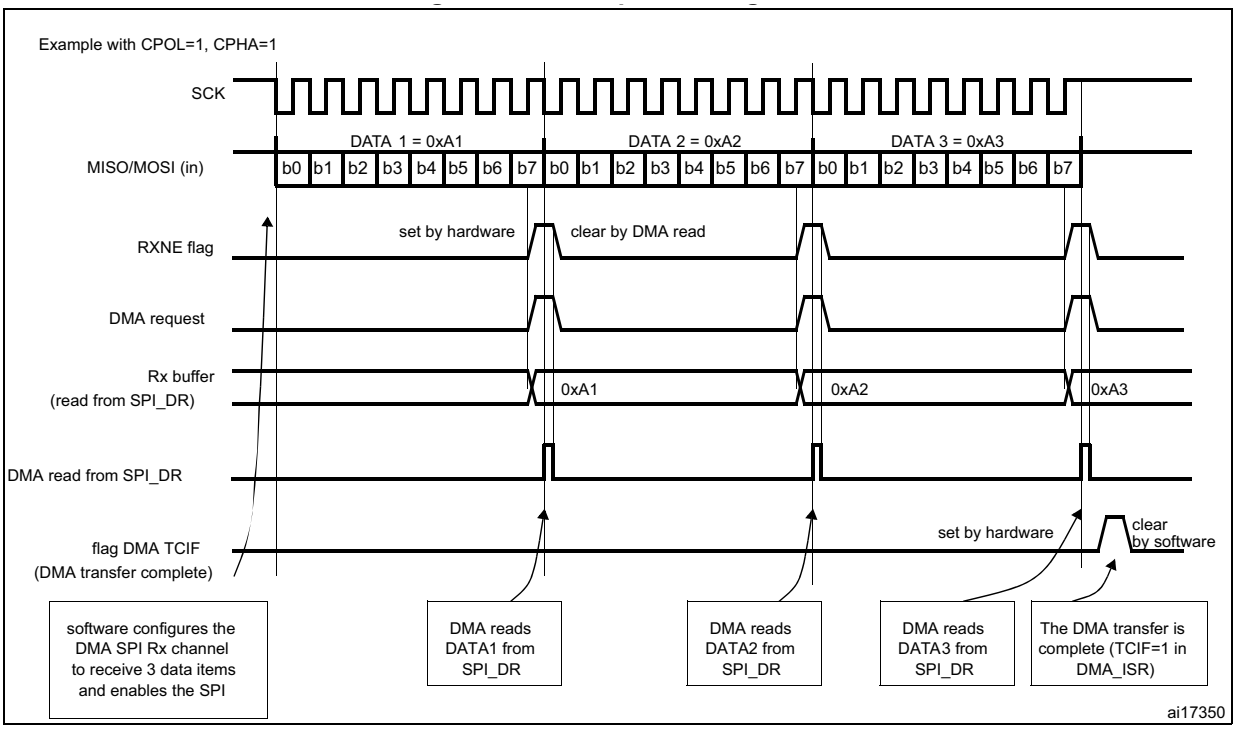
При приёме, после передачи DMA всех данных (стоит флаг TCIF регистра DMA_ISR), конец работы SPI отслеживают по флагу BSY. Нужно дождаться сначала TXE=1, затем BSY=0.

NB: Во время прерывных операций между записью регистра SPI_DR и флагом BSY появляется задержка в 2 такта APB. Так что после записи последнего данного нужно дождаться сначала TXE=1, затем BSY=0.

Передача с DMA



Приём с DMA



DMA с CRC

При DMA передачах с включённым CRC бит CRCNEXT ставить не надо, всё происходит автоматически, надо лишь прочитать полученный CRC из регистра SPI_DR чтобы снять флаг RXNE.

При ошибке передачи с CRC ставится флаг CRCERR в регистре SPI_SR.

25.3.10.Флаги ошибок

Сбой режима ведущего (MODF)

Бит MODF автоматически ставится если ножка NSS ведущего (при аппаратном управлении) силком включена в низкий уровень, или бит SSI стоит низким (при программном управлении). Сбой работает так:

- Бит MODF ставится и при стоящем бите ERRIE выдаётся прерывание.
- Чистится бит SPE, чем блокируются выходы и отключается интерфейс SPI.
- Бит MSTR снимается, теперь он ведомый.

Программно бит MODF снимается так:

1. Читаем или пишем регистр SPI_SR при стоящем бите MODF.
2. Пишем регистр SPI_CR1.

Во избежание конфликта ведомых в системе с несколькими MCU во время очистки бита MODF ножка NSS должна удерживаться высокой. После этой очистки биты SPE и MSTR можно ставить в исходное состояние, при стоящем MODF аппаратура не позволяет ставить биты SPE и MSTR.

У ведомых бит MODF не ставится. Но в системе с несколькими ведущими устройство может попасть в ведомые со стоящим MODF. Так обнаруживается конфликт ведущих.

Переполнение

Возникает при посылке ведущим данных, когда ведомый ещё не очистил бит RXNE предыдущей передачи. При переполнении:

- ставится бит OVR и при стоящем бите ERRIE выдаётся прерывание.

В этом случае новый байт не заменяет старого содержимого в регистре SPI_DR и все последующие байты теряются.

Снимается бит OVR чтением из регистра SPI_DR с последующим чтением из регистра SPI_SR.

Ошибка CRC

Флаг CRCERR в регистре SPI_SR ставится если принятое значение CRC не совпадает с содержимым регистра SPI_RXCRCR при стоящем бите CRCEN в регистре SPI_CR1.

25.3.11.Прерывания SPI

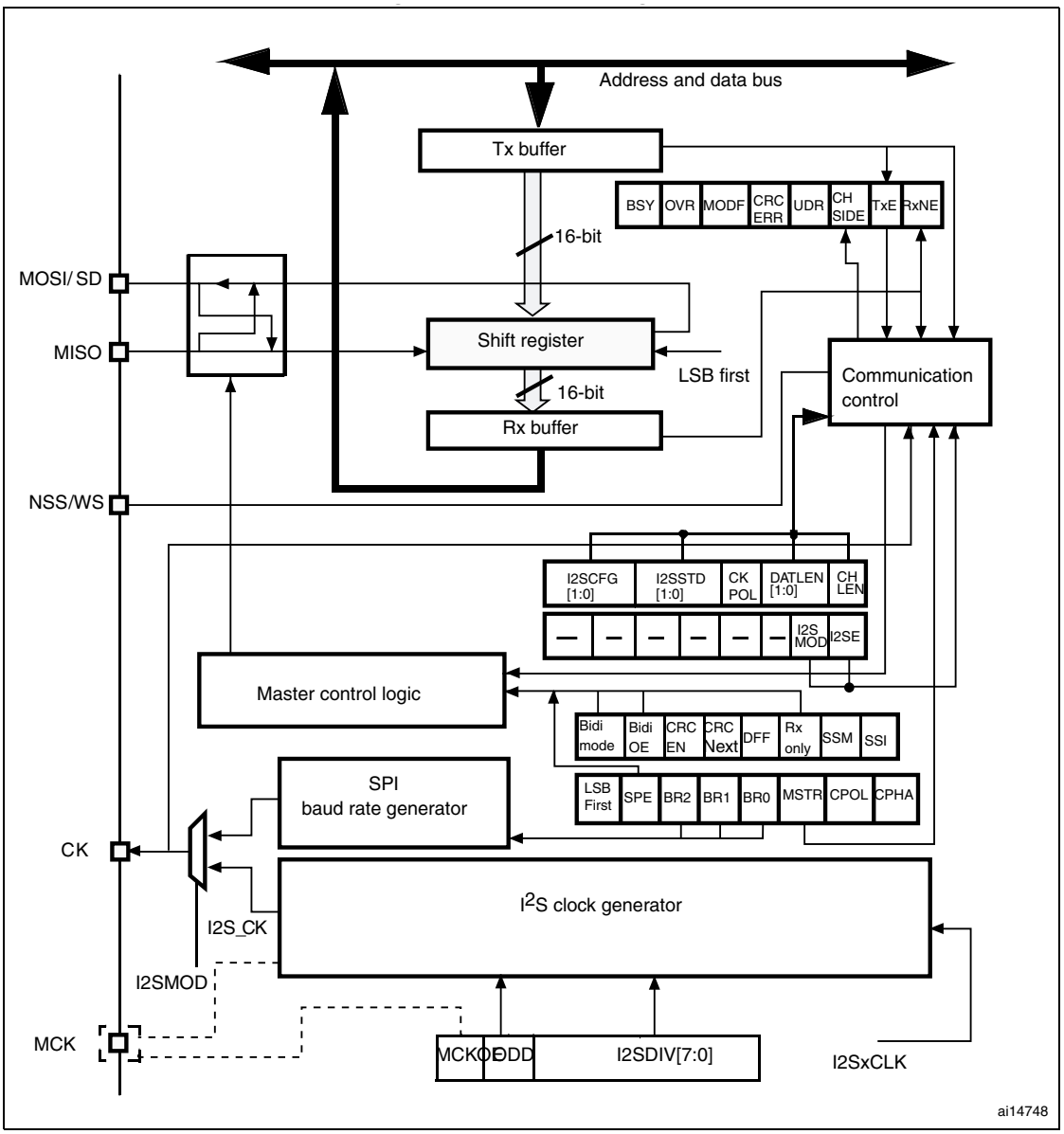
Table 182. SPI interrupt requests

Прерывание	Флаг события	Бит разрешения
Передающий буфер пуст	TXE	TXEIE
Приёмный буфер не пуст	RXNE	RXNEIE
Сбой Ведущего	MODF	ERRIE
Переполнение	OVR	
Ошибка CRC	CRCERR	

25.4. Функциональное описание I²S

Звукового интерфейса I²S в устройствах низкой и средней плотности нет, в остальных есть.

25.4.1. Общее описание I²S



Режим звукового интерфейса I^{2}S блока SPI включается установкой бита `I2SMOD` в регистре `SPI_I2SCFGR`. Используются почти те же ножки, флаги и прерывания, что и в SPI.

У I^{2}S три общие ножки с SPI:

- SD: Последовательные данные (или MOSI) для приёма или передачи двух мультиплексированных по времени каналов данных (только полу-дуплекс).
- WS: Выбор слова (или NSS) это вывод данных для ведущего или ввод данных для ведомого.
- CK: Такты линии (или SCK) это выход тактов для ведущего или вход тактов для ведомого.

Дополнительная ножка используется когда внешнему устройству нужны основные такты:

- MCK: Ведущие такты (отдельная), используется при ведущем I^{2}S (и при стоящем бите `MCKOE` в регистре `SPI_I2SPR`) для вывода дополнительных тактов частотой 256 × F_S, где F_S это частота дискретизации звука.

В режиме ведущего I^{2}S сам генерирует такты связи и ведущие такты от одного генератора. Также есть два дополнительных регистра. Один конфигурации тактов (`SPI_I2SPR`) и другой конфигурации I^{2}S (`SPI_I2SCFGR`) (стандарт звука, ведущий/ведомый, формат данных, фрейм пакета, полярность тактов и пр.).

Регистры `SPI_CR1` и CRC в режиме I^{2}S не используется. Точно также не используется бит `SSOE` регистра `SPI_CR2` и биты `MODF` и `CRCERR` регистра `SPI_SR`.

Для передачи данных в 16-бит режиме I^{2}S использует регистр данных SPI (`SPI_DR`).

25.4.2. Поддерживаемые звуковые протоколы

Для передачи звуковых данных двух мультиплексированных каналов (левого и правого) есть три линии шины и только один 16-бит регистр приёма/передачи данных. Так что писать или читать данные канала нужно сверяясь с битом CHSIDE регистра SPI_SR. Левый канал всегда передаётся первым (в протоколе PCM бит CHSIDE смысла не имеет).

Данные посылаются в форматах:

- 16-бит данные в 16-бит фрейме
- 16-бит данные в 32-bit фрейме
- 24-бит данные в 32-bit фрейме
- 32-бит данные в 32-bit фрейме

При 16-бит данных в 32-бит пакете старшие 16 бит значимые, остальные 16 бит нулевые и не требуют действий программы или запроса DMA (только одно чтение/запись).

Для 24-бит и 32-бит фреймов данных нужны два чтения/записи SPI_DR или операции DMA. Незначимые 8 бит в 24-бит фрейме аппаратно расширяются нулями до 32 бит.

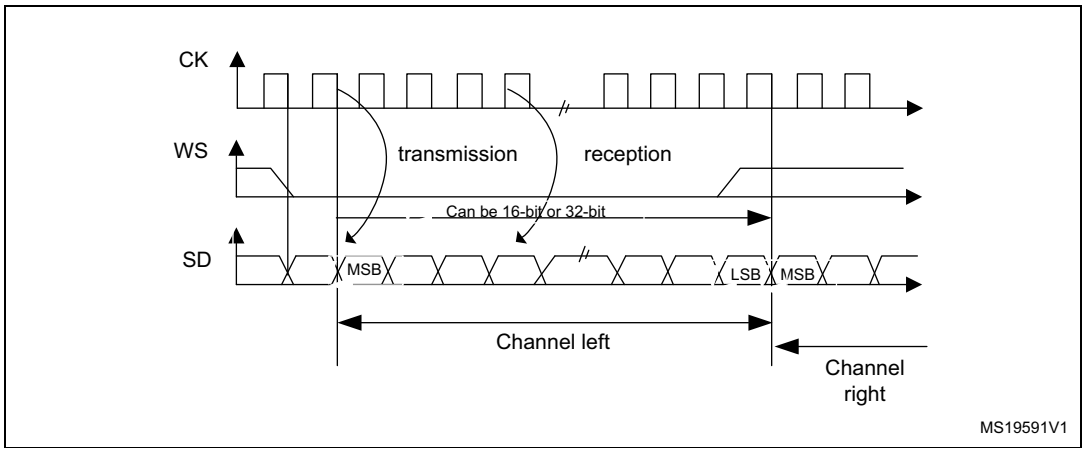
Во всех форматах данных и связи первым передаётся самый значащий бит.

Биты I2SSTD[1:0] и PCMSYNC регистра SPI_I2SCFGR определяют один из четырёх поддерживаемых I^2S звуковых стандартов:

Стандарт I^2S Philips

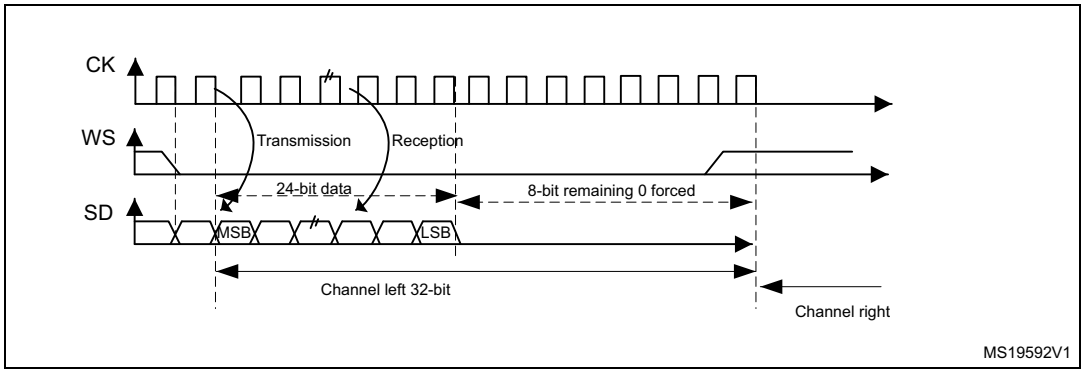
Здесь передаваемый канал указывается сигналом WS, он активируется за один такт CK перед подачей первого бита данных.

Стандарт I^2S Philips (16/32-бит полной точности, CPOL = 0)



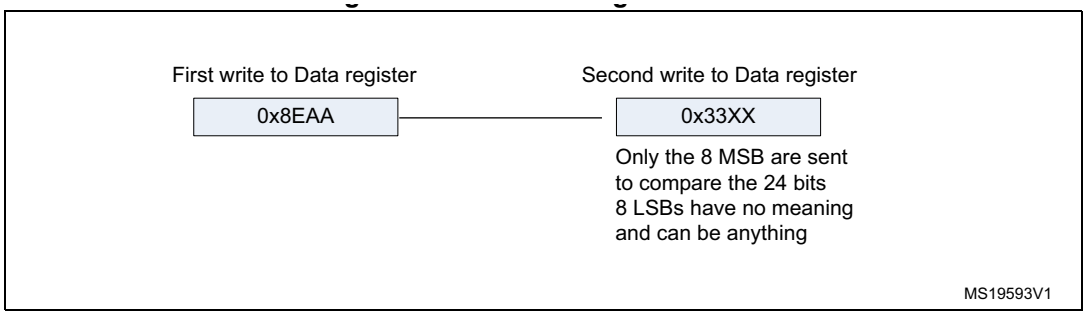
Данные фиксируются передатчиком по заднему фронту CK, а приёмником по переднему фронту. Сигнал WS считывается по заднему фронту CK.

Стандарт I^2S Philips (24-бит фрейм с CPOL = 0)

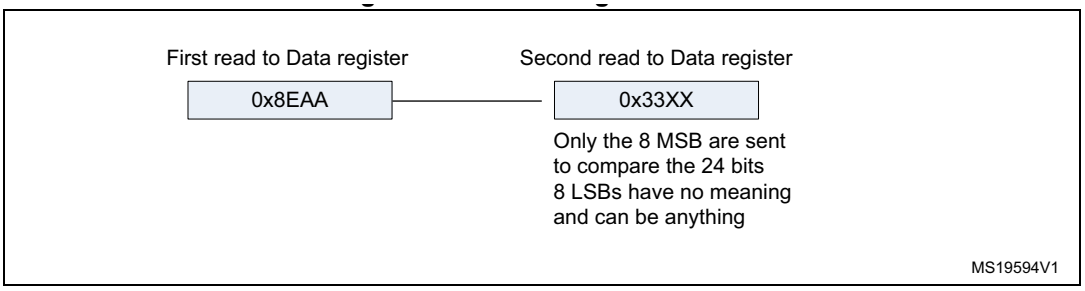


Этот режим требует двух операций чтения/записи SPI_DR.

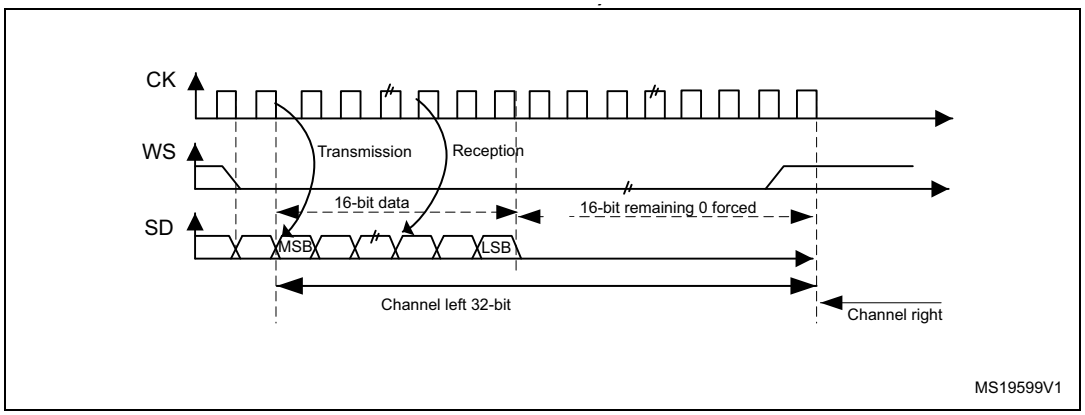
• Посылаем 0x8EAA33 (24 бита):



• Принимаем 0x8EAA33:

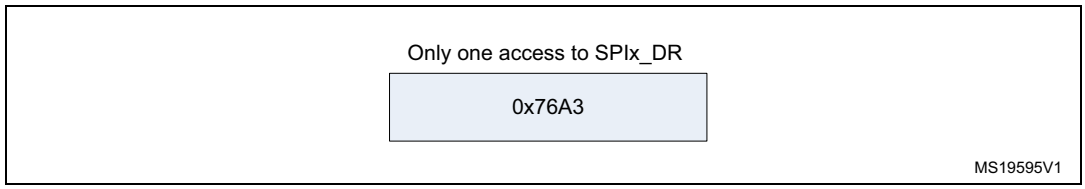


Стандарт I²S Philips (16-бит данных в 32-бит пакете с CPOL = 0)



При конфигурации I²S на 16-данные в 32-бит фрейме канала требуется только одно обращение к регистру SPI_DR. Остальные 16 бит аппаратно обнуляются.

Передаём расширенное до 32 бит число 0x76A3 (0x76A30000):



При передаче, после записи старшей части в SPI_DR, ставится флаг TXE и может вызываться прерывание для записи нового передаваемого значения. Число 0x0000 пересылается аппаратно, не затрагивая SPI_DR.

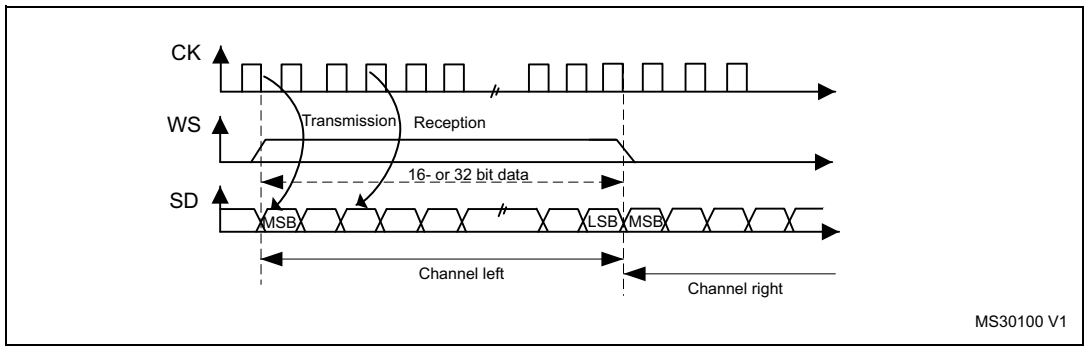
При приёме флаг RXNE ставится при получении старшего 16-бит полуслова.

Так для двух обращений к SPI_DR отводится больше времени.

Стандарт левого выравнивания

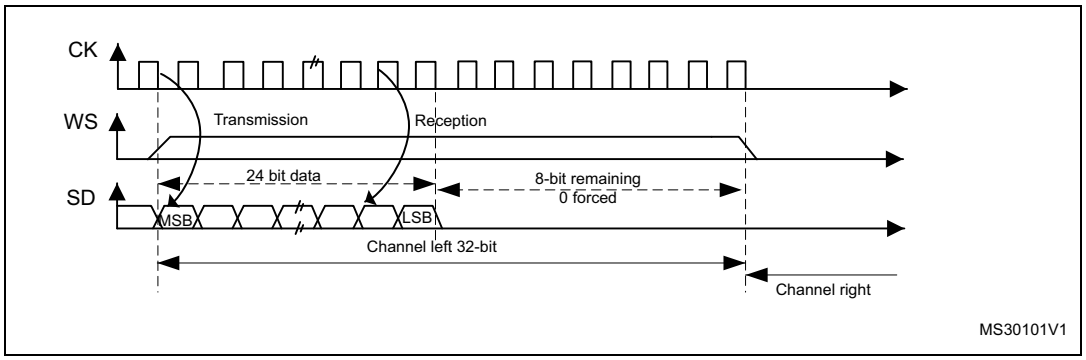
Здесь сигнал WS выдаётся одновременно с первым (старшим) битом данных.

Левое выравнивание 16- или 32-бит полной точности с CPOL = 0

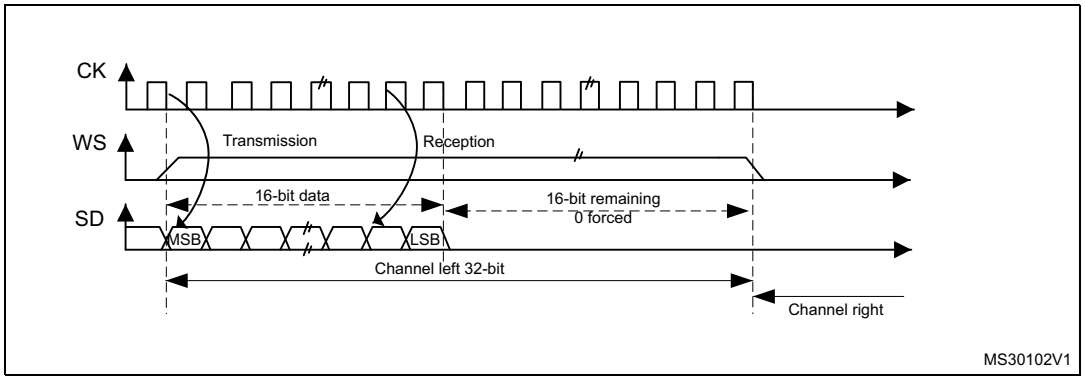


Данные фиксируются передатчиком по заднему фронту CK, а приёмником по переднему фронту.

Левое выравнивание 24-бит фрейма с CPOL = 0



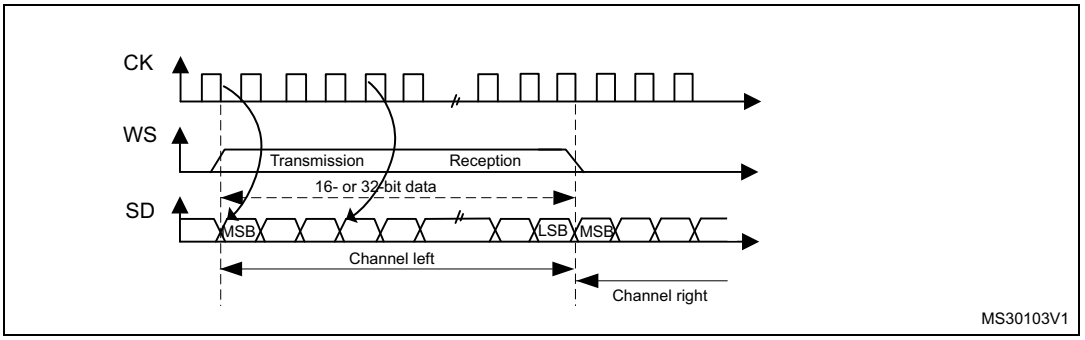
Левое выравнивание 16-бит данных в 32-бит пакете с CPOL = 0



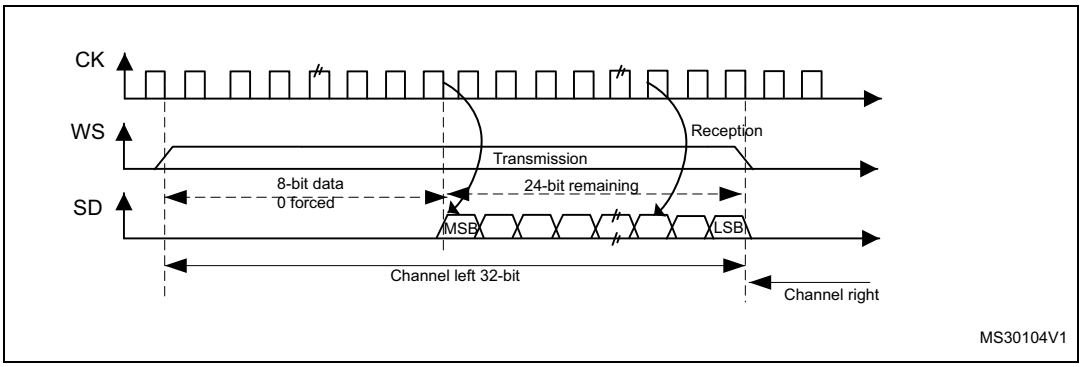
Стандарт правого выравнивания

Он подобен левому выравниванию (форматы 16-бит и 32-бит фреймов полной точности совпадают).

Правое выравнивание 16- или 32-бит полной точности с CPOL = 0

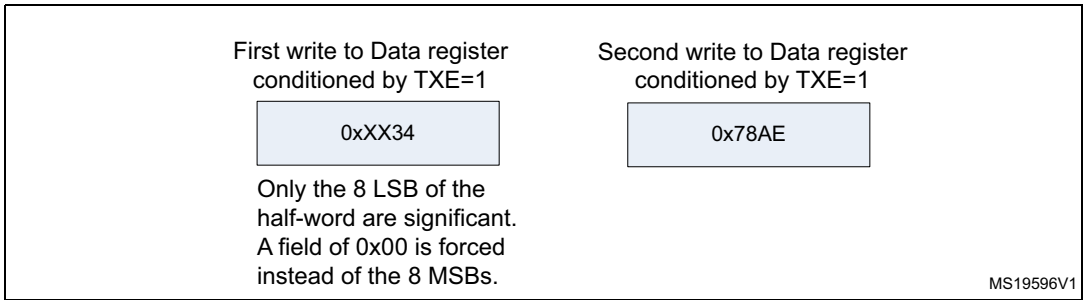


Правое выравнивание 24-бит фрейма с CPOL = 0



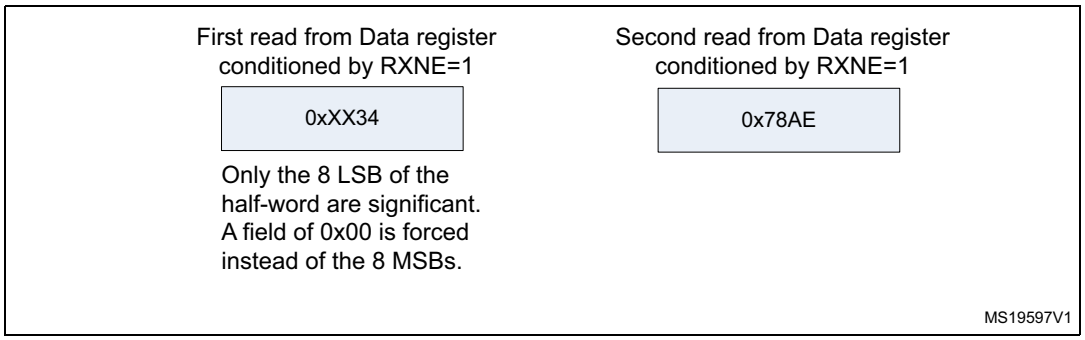
- Передаём число 0x3478AE за две записи в регистр SPI_DR:

Операции передачи 0x3478AE

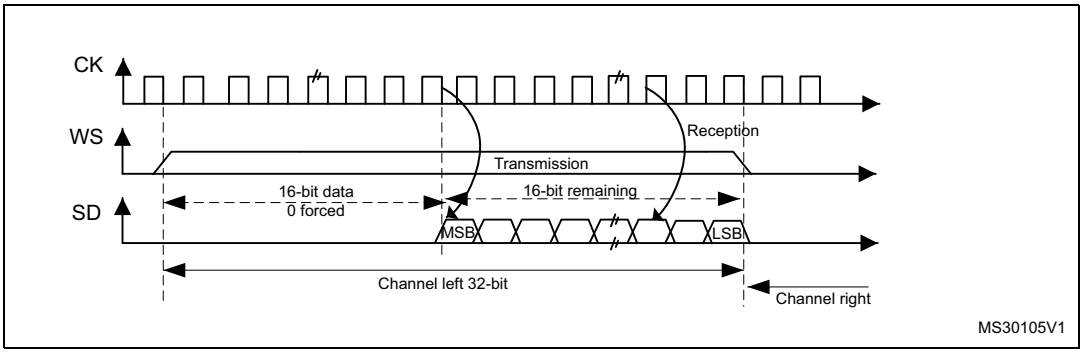


Принимаем 0x3478AE за два чтения SPI_DR по событию RXNE:

Операции приёма 0x3478AE



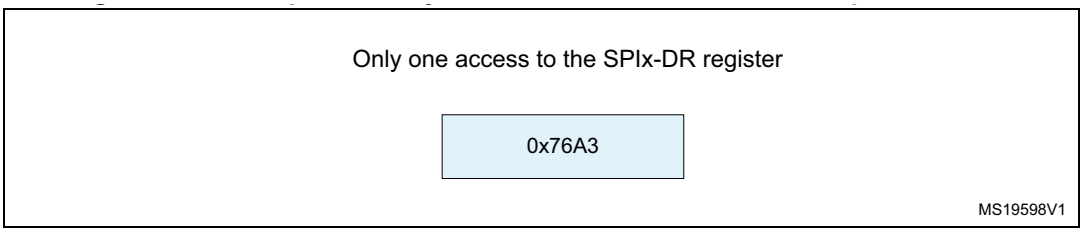
Правое выравнивание 16-бит в 32-бит фрейме с CPOL = 0



При конфигурации I^{2}S на 16-данные в 32-бит фрейме канала требуется только одно обращение к регистру SPI_DR. Остальные 16 бит аппаратно обнуляются.

Передаём расширенное до 32 бит число 0x76A3 (0x0000 76A3):

Пример правого выравнивания 16 бит в 32 бит фрейме



При передаче, по установке TXE пишутся передаваемые данные (здесь 0x76A3). Поле 0x0000 передаётся первым. Бит TXE снова ставится после передачи значимых данных (0x76A3) по SD.

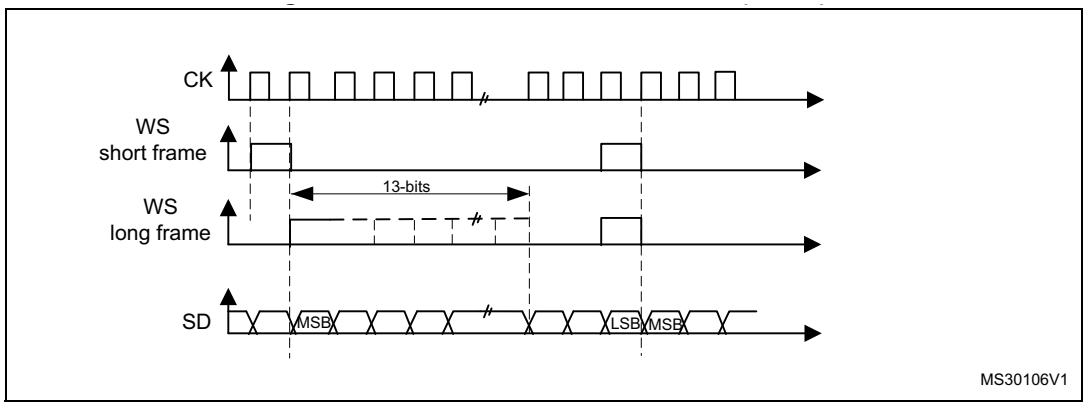
При приёме, RXNE выставляется после получения значимого полуслова (на поле 0x0000 плюём).

Так выделяется больше времени на промежуток между обращениями к SPI_DR.

Стандарт PCM

Здесь информация о канале (левый/правый) не нужна. Два режима PCM (короткий и длинный фрейм) определяются битом PCMSYNC в регистре SPI_I2SCFGR.

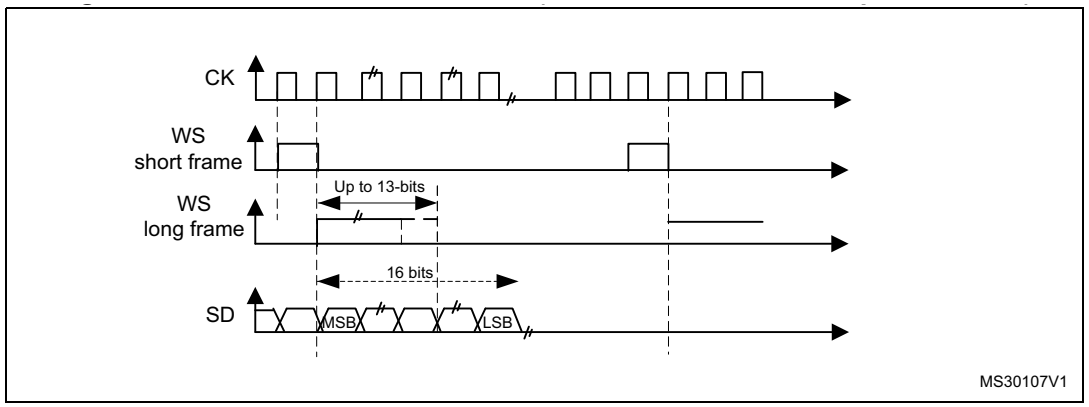
Стандартные сигналы PCM (16-бит)



При длинной синхронизации фрейма сигнал WS имеет длину 13 бит в режиме ведущего.

При короткой синхронизации фрейма сигнал WS имеет длину только 1 такт.

Стандартные сигналы PCM (16-бит данных в 32-бит фрейме)



NB: Для обоих режимов (ведущий и ведомый) и обоих видов синхронизации битов (короткий и длинный) надо указывать число битов между двумя последовательными участками данных (биты DATLEN и CHLEN в регистре SPI_I2SCFGR register).

25.4.3. Тактовый генератор

Битрейт I^2S определяет поток данных I^2S и частоту тактов I^2S.

Битрейт I^2S = число бит на канал × число каналов × частота дискретизации звука

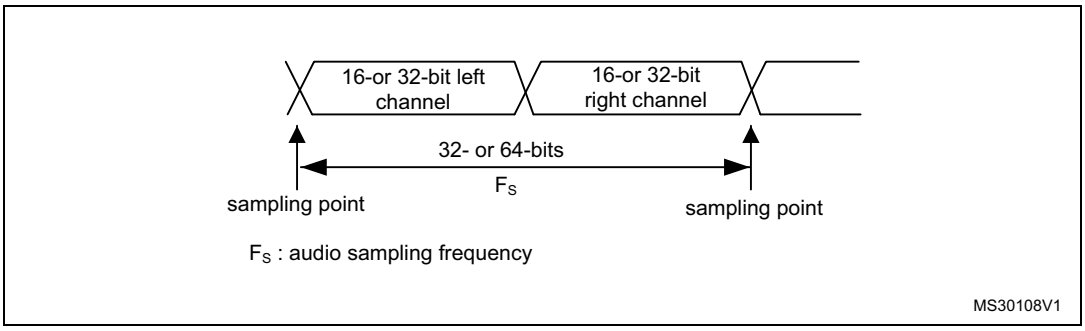
Для 16-бит звука, двух каналов битрейт I^2S вычисляется так:

Битрейт $I^2S = 16 \times 2 \times F_S$

Для 32-бит пакета он будет:

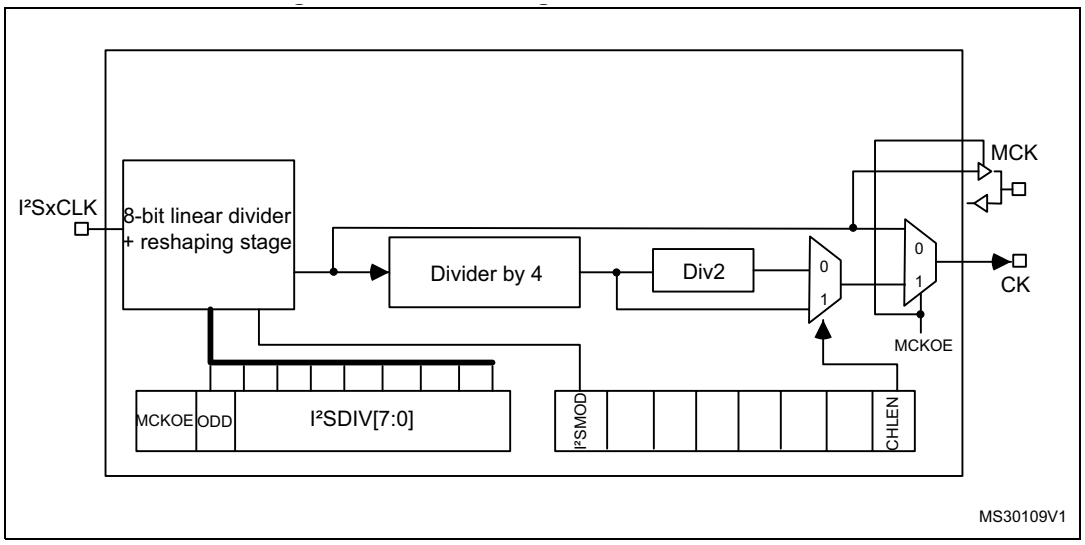
Битрейт $I^2S = 32 \times 2 \times F_S$

Определение частоты дискретизации звука



Для связи с частотой дискретизации звука в режиме ведущего надо установить линейный делитель.

Архитектура тактового генератора I²S



1. Здесь x может быть 2 или 3.

I2SxCLK это системные такты (подаются от HSI, HSE или PLL, питающих такты AHB). В сетевых устройствах источником I2SxCLK могут быть SYSCLK или PLL3 VCO (2 × PLL3CLK). Это определяется битами `I2S2SRC` и `I2S3SRC` регистра `RCC_CFGR2`.

Частота дискретизации звука может быть 96 kHz, 48 kHz, 44.1 kHz, 32 kHz, 22.05 kHz, 16 kHz, 11.025 kHz или 8 kHz (или внутри этого диапазона). Линейный делитель определяется по следующим формулам:

При выдаче ведущих тактов (стоит бит `MCKOE` в регистре `SPI_I2SPR`):

F_S = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD)*8)] при 16-бит фрейме канала

F_S = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD)*4)] при 32-бит фрейме канала

При отсутствии ведущих тактов (бит `MCKOE` чист):

F_S = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD))] при 16-бит фрейме канала

F_S = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD))] при 32-бит фрейме канала

Таблица 183. Точность звуковой частоты при стандартном 8 MHz HSE (устройства высокой и XL-плотности)

SYSCLK (MHz)	I2S_DIV		I2S_ODD		MCLK	Целевая f_S (Hz)	Реальная f_S (KHz)		Ошибка	
	16-бит	32-бит	16-бит	32-бит			16-бит	32-бит	16-бит	32-бит
72	11	6	1	0	Нет	96000	97826.09	93750	2 %	2 %
72	23	11	1	1	Нет	48000	47872.34	48913.04	0 %	2 %
72	25	13	1	0	Нет	44100	44117.65	43269.23	0 %	2 %
72	35	17	0	1	Нет	32000	32142.86	32142.86	0 %	0 %

72	51	25	0	1	Нет	22050	22058.82	22058.82	0 %	0 %
72	70	35	1	0	Нет	16000	15675.75	16071.43	0 %	0 %
72	102	51	0	0	Нет	11025	11029.41	11029.41	0 %	0 %
72	140	70	1	1	Нет	8000	8007.11	7978.72	0 %	0 %
72	2	2	0	0	Да	96000	70312.15	70312.15	27 %	27 %
72	3	3	0	0	Да	48000	46875	46875	2 %	2 %
72	3	3	0	0	Да	44100	46875	46875	6 %	6 %
72	4	4	1	1	Да	32000	31250	31250	2 %	2 %
72	6	6	1	1	Да	22050	21634.61	21634.61	2 %	2 %
72	9	9	0	0	Да	16000	15625	15625	2 %	2 %
72	13	13	0	0	Да	11025	10817.30	10817.30	2 %	2 %
72	17	17	1	1	Да	8000	8035.71	8035.71	0 %	0 %

Таблица 184. Точность звуковой частоты при стандартном 25 MHz и PLL3 (сетевые устройства)

Длина данных	PREDIV2	PLL3MUL	I2SDIV	I2SODD	MCLK	Целевая fs(Hz)	Реальная fs (KHz)	Ошибка
32	6	14	9	1	No	96000	95942.9825	0 %
16	6	14	38	0	No	48000	47971.4912	0 %
32	6	14	19	0	No	48000	47971.4912	0 %
16	8	14	31	0	No	44100	44102.823	0 %
32	8	14	15	1	No	44100	44102.823	0 %
16	5	13	63	1	No	32000	31988.189	0 %
32	8	20	30	1	No	32000	32018.443	0 %
16	8	14	62	0	No	22050	22051.4113	0 %
32	8	14	31	0	No	22050	22051.4113	0 %
16	7	20	139	1	No	16000	16001.0241	0 %
32	5	13	63	1	No	16000	15994.0945	0 %
16	8	14	124	0	No	11025	11025.7056	0 %
32	8	14	62	0	No	11025	11025.7056	0 %
16	7	10	139	1	No	8000	8000.51203	0 %
32	7	20	139	1	No	8000	8000.51203	0 %
16	5	10	2	0	Yes	96000	97656.25	2 %
32	5	10	2	0	Yes	96000	97656.25	2 %
16	7	12	3	1	Yes	48000	47831.6327	0 %
32	7	12	3	1	Yes	48000	47831.6327	0 %
16	5	9	4	0	Yes	44100	43945.3125	0 %

Длина данных	PREDIV2	PLL3MUL	I2SDIV	I2SODD	MCLK	Целевая fs(Hz)	Реальная fs (KHz)	Ошибка
32	5	9	4	0	Yes	44100	43945.3125	0 %
16	5	9	5	1	Yes	32000	31960.2273	0 %
32	5	9	5	1	Yes	32000	31960.2273	0 %
16	5	13	11	1	Yes	22050	22078.8043	0 %
32	5	13	11	1	Yes	22050	22078.8043	0 %
16	5	9	11	0	Yes	16000	15980.1136	0 %
32	5	9	11	0	Yes	16000	15980.1136	0 %
16	8	14	15	1	Yes	11025	11025.7056	0 %
32	8	14	15	1	Yes	11025	11025.7056	0 %
16	8	20	30	1	Yes	8000	8004.61066	0 %
32	8	20	30	1	Yes	8000	8004.61066	0 %

Таблица 185. Точность звуковой частоты при стандартном 14.7456 MHz и PLL3 (сетевые устройства)

Длина данных	PREDIV2	PLL3MUL	I2SDIV	I2SODD	MCLK	Целевая fs(Hz)	Реальная fs (KHz)	Ошибка
16	3	10	16	0	No	96000	96000	0 %
32	3	10	8	0	No	96000	96000	0 %
16	3	10	32	0	No	48000	48000	0 %
32	3	10	16	0	No	48000	48000	0 %
16	4	9	23	1	No	44100	44119.148	0 %
32	4	13	17	0	No	44100	44047.059	0 %
16	3	10	48	0	No	32000	32000	0 %
32	3	10	24	0	No	32000	32000	0 %
16	4	20	104	1	No	22050	22047.8469	0 %
32	4	9	32	1	No	22050	22059.5745	0 %
16	3	10	96	0	No	16000	16000	0 %
32	3	10	48	0	No	16000	16000	0 %
16	4	20	209	1	No	11025	11023.923	0 %
32	4	20	104	1	No	11025	11023.923	0 %
16	3	10	192	0	No	8000	8000	0 %
32	3	10	96	0	No	8000	8000	0 %
16	3	10	2	0	Yes	96000	96000	0 %
32	3	10	2	0	Yes	96000	96000	0 %
16	3	10	4	0	Yes	48000	48000	0 %
32	3	10	4	0	Yes	48000	48000	0 %

Длина данных	PREDIV2	PLL3MUL	I2SDIV	I2SODD	MCLK	Целевая fs(Hz)	Реальная fs (KHz)	Ошибка
16	4	20	6	1	Yes	44100	44307.6923	0 %
32	4	20	6	1	Yes	44100	44307.6923	0 %
16	3	10	6	0	Yes	32000	32000	0 %
32	3	10	6	0	Yes	32000	32000	0 %
16	4	13	8	1	Yes	22050	22023.5294	0 %
32	4	13	8	1	Yes	22050	22023.5294	0 %
16	3	10	12	0	Yes	16000	16000	0 %
32	3	10	12	0	Yes	16000	16000	0 %
16	4	13	17	0	Yes	11025	11029.7872	0 %
32	4	13	17	0	Yes	11025	11029.7872	0 %
16	3	10	24	0	Yes	8000	8000	0 %
32	3	10	24	0	Yes	8000	8000	0 %

25.4.4. Ведущий I²S

У ведущего I²S на ножку CK выдаются такты связи, равно как и сигнал выбора слова (WS), выдача ведущих тактов (MCK) разрешается битом MCKOE регистра SPI_I2SPR.

Процедура

1. Битами I2SDIV[7:0] регистра SPI_I2SPR определяем частоту тактов связи. Также определяем бит ODD регистра SPI_I2SPR.

2. Битом CKPOL определяем уровень стабильности тактов связи. Если для внешнего звукового компонента DAC/ADC нужны такты MCK, то разрешаем их битом MCKOE регистра SPI_I2SPR (значения I2SDIV и ODD зависят от наличия MCK).

3. Битом I2SMOD регистра SPI_I2SCFGR активируем функциональность I²S, битами I2SSTD[1:0] PCMSYNC выбираем стандарт I²S, длину данных битами DATLEN[1:0] и число битов на канал битом CHLEN. Включаем режим ведущего I²S и направление битами I2SCFG[1:0] регистра SPI_I2SCFGR.

4. По нужде выбираем потенциальные источники прерываний и возможности DMA в регистре SPI_CR2.

5. Ставим бит I2SE регистра SPI_I2SCFGR.

WS и CK в выходном режиме. Вывод MCK разрешается битом MCKOE регистра SPI_I2SPR.

Последовательность передачи

Начинается записью первого полуслова левого канала в буфер Tx.

Передача буфера Tx в регистр сдвига ставит флаг TXE, умоляя записать в Tx полуслово правого канала. Флаг CHSIDE указывает какой канал надо передавать следующим. Он имеет смысл при высоком флаге TXE, так как изменяется во время установки TXE.

Полный фрейм это передача данных левого и затем правого канала, по отдельности один канал передать нельзя.

Полуслово данных параллельно пишется в 16-бит регистр сдвига, откуда выдвигается на ножку MOSI/SD, начиная со старшего бита. Флаг TXE ставится каждой передачей из Tx в регистр сдвига, а ещё может выдаваться прерывание, если разрешено битом TXEIE в регистре SPI_CR2.

Для непрерывной передачи, следующее данное надо писать в регистр SPI_DR до конца текущей передачи.

Для выключения I²S очисткой бита I2SE, надо сначала дождаться TXE = 1 и BSY = 0.

Последовательность приёма

Работа подобна передаче, кроме пункта 3, где битами I2SCFG[1:0] надо поставить приём ведущего.

Независимо от длины данных и канала данные принимаются 16-бит пакетами, то есть приём отсчёта канала может занимать до двух чтений буфера Rx. После каждого приёма в буфер Rx ставится флаг RXNE и выдаётся прерывание, разрешённое битом RXNEIE регистра SPI_CR2.

Бит RXNE чистится чтением регистра SPI_DR. CHSIDE обновляется после каждого приёма, он чувствителен к сигналу WS от ячейки I^{2}S.

В случае приёма данного при ещё не прочитанном предыдущем, ставится флаг OVR. Прерывание переполнения разрешается битом ERRIE регистра SPI_CR2.

Для выключения I^{2}S нужно правильно остановить текущие передачи в соответствии с установками. В случае:

- 16-бит данных 32-бит канала (DATLEN=00 и CHLEN=1) с правым выравниванием (I2SSTD=10)
  a) Ждём предпоследнего RXNE=1(n–1)
  b) Ждём 17 тактов I^{2}S (в программном цикле)
  c) Выключаем I^{2}S (I2SE = 0)
- 16-бит данных 32-бит канала (DATLEN=00 и CHLEN=1) с левым выравниванием, I^{2}S или PCM (I2SSTD = 00, I2SSTD = 01 или I2SSTD = 11, соответственно)
  a) Ждём последнего RXNE
  b) Ждём 1 такт I^{2}S (в программном цикле)
  c) Выключаем I^{2}S (I2SE = 0)
- Для остальных комбинаций DATLEN и CHLEN, независимо от битов I2SSTD:
  a) Ждём предпоследнего RXNE=1(n–1)
  b) Ждём 1 такт I^{2}S (в программном цикле)
  c) Выключаем I^{2}S (I2SE = 0)

NB: Во время передач флаг BSY остаётся низким.

25.4.5. Ведомый I^{2}S

Здесь такты на интерфейс I^{2}S не выдаются, они и сигнал WS это входы.

Конфигурируем:

1. Битом I2SMOD регистра SPI_I2SCFGR активируем функциональность I^{2}S, битами I2SSTD[1:0] выбираем стандарт I^{2}S, длину данных битами DATLEN[1:0] и число битов на канал битом CHLEN. Включаем режим ведомого I^{2}S и направление битами I2SCFG[1:0] регистра SPI_I2SCFGR.
2. По нужде выбираем потенциальные источники прерываний и возможности DMA в регистре SPI_CR2.
3. Ставим бит I2SE регистра SPI_I2SCFGR.

Последовательность передачи

Начинается при появлении тактов от ведущего и сигнала NSS_WS запроса передачи. Ведомый должен быть уже включён и регистр данных записан.

В режимах I^{2}S, левого и правого выравнивания первым пишется левый канал. При старте данное из буфера Tx пишется в регистр сдвига. Ставится флаг TXE, чтобы в регистр данных мы записали данное правого канала.

Флаг CHSIDE показывает, какой канал надо передавать. Он зависит от сигнала WS от внешнего ведущего. То есть ведомый должен быть готов к передаче первого данного левого канала до подачи тактов. Сигнал WS выставляется для первого левого канала.

Бит I2SE надо писать не менее чем за два такта PCLK до вывода первого такта на линию CK.

Первое полуслово данных пишется в регистр сдвига вовремя передачи первого бита и затем выдвигается на ножку MOSI/SD начиная со старшего бита. Флаг TXE ставится после каждой

передачи из Tx в регистр сдвига и выдаётся прерывание, если разрешено битом TXEIE в регистре SPI_CR2.

Стоящий флаг TXE надо проверять перед попыткой записи в буфер Tx buffer.

Для непрерывных передач регистр SPI_DR надо писать до завершения текущей передачи. Если до первого фронта тактов следующей передачи регистр SPI_DR ещё не записан, то ставится флаг UDR ошибки исчерпания. Прерывание вызывается при стоящем бите ERRIE регистра SPI_CR2. В этом случае надо обязательно выключать I^2S для рестарта передачи начиная с левого канала.

I^2S выключается очисткой бита I2SE с обязательным ожиданием TXE = 1 и BSY = 0.

Последовательность приёма

Работа подобна передаче, кроме пункта 1, где битами I2SCFG[1:0] регистра SPI_I2SCFGR надо поставить приём ведомого.

Независимо от длины данных и канала данные принимаются 16-бит пакетами, то есть приём отсчёта канала может занимать до двух чтений буфера Rx. После каждого приёма в буфер Rx ставится флаг RXNE и выдаётся прерывание, разрешённое битом RXNEIE регистра SPI_CR2.

Бит RXNE чистится чтением регистра SPI_DR. CHSIDE обновляется после каждого приёма, он чувствителен к внешнему сигналу WS.

В случае приёма данного при ещё не прочитанном предыдущем, ставится флаг OVR. Прерывание переполнения разрешается битом ERRIE регистра SPI_CR2.

Выключается приём I^2S очисткой бита I2SE сразу после появления последнего RXNE = 1.

25.4.6. Флаги состояния

Их три:

Занят (BSY)

Флаг BSY ставится и снимается аппаратно, указывая состояние уровня связи I^2S.

В режиме приёма ведущего (I2SCFG = 11) флаг BSY удерживается низким.

Смотреть на флаг BSY надо перед выключением I^2S.

Очищается флаг BSY:

- при завершении передачи (кроме непрерывной передачи ведущего)
- при выключении I^2S

При непрерывных передачах:

- В режиме передачи ведущего флаг BSY удерживается высоким
- В режиме ведомого флаг BSY снимается на 1 такт I^2S между всеми передачами

NB: Не используйте флаг BSY для определения конца передачи, TXE и RXNE лучше.

Буфер Tx пуст (TXE)

Флаг TXE сбрасывается при записи в буфер Tx и у выключенного I^2S (бит I2SE снят).

Буфер RX не пуст (RXNE)

В буфере RX есть данное. Снимается при чтении регистра SPI_DR.

Флаг канала (CHSIDE)

При передаче флаг CHSIDE переключается во время установки флага TXE. Показывает номер передаваемого канала (левый/правый). В случае ошибки исчерпания от теряет смысл и I^2S надо выключать и включать для восстановления передач, начиная с левого канала.

При приёме он переключается при записи полученного данного в регистр SPI_DR. Показывает номер полученного канала. В случае ошибки (вроде OVR) от теряет смысл и I^2S надо выключать и включать для восстановления передач, начиная с левого канала.

В режиме PCM он не нужен.

Флаги ошибки OVR и UDR в регистре SPI_SR ставят бит ERRIE регистра SPI_CR2 и выдают прерывание. Прерывание снимается чтением регистра SPI_SR после очистки источника прерывания.

25.4.7. Флаги ошибок

Есть две ошибки ячейки I^{2}S.

Исчерпание (UDR)

При передаче ведомого он ставится по первому такту передачи при ещё не записанном регистре SPI_DR. Возможен при стоящем бите I2SMOD в регистре SPI_I2SCFGR. Прерывание возможно по стоящему биту ERRIE регистр SPI_CR2.

Бит UDR снимается чтением регистра SPI_SR.

Переполнение (OVR)

Ставится при получении данного с ещё не прочитанным регистром SPI_DR. Новое и все последующие данные теряются. Прерывание возможно по стоящему биту ERRIE регистр SPI_CR2.

Снимается бит OVR чтением регистра SPI_DR с последующим чтением регистра SPI_SR.

25.4.8. Прерывания I^{2}S

Прерывание	Флаг события	Бит разрешения
Передающий буфер пуст	TXE	TXEIE
Приёмный буфер не пуст	RXNE	RXNEIE
Переполнение	OVR	ERRIE
Исчерпание	UDR	

25.4.9. I^{2}S с DMA

Здесь DMA работает так же как и в SPI. Вот только контроля CRC нет.

25.5. Регистры SPI и I^{2}S

Регистры доступны полусловами (16 бит) и словами (32 бит).

25.5.1. Регистр 1 управления SPI (SPI_CR1) (в I^{2}S не используется)

Смещение адреса: 0x00

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	DFF	RX ONLY	SSM	SSI	*LSB FIRST*	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 15 **BIDIMODE:** Двунаправленный режим данных

0: Однонаправленные данные по 2 линиям

1: Двунаправленные данные по 1 линии

NB: В режиме I^{2}S не используется

— Бит 14 **BIDIOE:** Разрешение вывода двунаправленного режима

Вместе с битом BIDIMODE определяет направление двунаправленной передачи

0: Выключен (только приём)

1: Включён (только передача)

NB: В режиме I^{2}S не используется.

У ведущего используется ножка MOSI, у ведомого - MISO.

— Бит 13 **CRCEN:** Разрешение вычисления CRC

0: Нельзя

1: Нужно

NB: В режиме I^{2}S не используется. Пишется при выключенном SPI (SPE = '0').

— Бит 12 **CRCNEXT:** CRC следующий

0: Фаза данных

1: Следующая фаза CRC

NB: В режимах полного дуплекса и только передачи CRCNEXT надо ставить сразу после записи регистра SPI_DR.

В режиме только приёма CRCNEXT должен стоять после приёма предпоследнего данного.

При передачах с DMA должен быть чистым. В режиме I^2S не используется.

— Бит 11 **DFF:** Формат фрейма данных

0: 8-бит фрейм данных

1: 16-бит фрейм данных

NB: В режиме I^2S не используется. Пишется при выключенном SPI (SPE = '0').

— Бит 10 **RXONLY:** Только приём

Вместе с битом BIDIMODE определяет направление передачи в однонаправленном режиме по 2 линиям. Полезен в системе с несколькими ведомыми при недоступности одного, вывод от доступного не нарушается.

0: Полный дуплекс (приём и передача)

1: Вывод запрещён (только приём)

NB: В режиме I^2S не используется.

— Бит 9 **SSM:** Программное управление ведомым

При стоящем бите SSM ввод от ножки NSS подменяется значением бита SSI.

0: Выключено

1: Включено

NB: В режиме I^2S не используется.

— Бит 8 **SSI:** Внутренний выбор ведомого

Действует при стоящем бите SSM. Вместо ножки NSS на выбор режима выдаётся этот бит.

NB: В режиме I^2S не используется.

— Бит 7 **LSBFIRST:** Формат фрейма

0: Старший бит передаётся первым

1: Младший бит передаётся первым

NB: В режиме I^2S не используется. Во время передачи изменять нельзя.

— Бит 6 **SPE:** Включение SPI

0: Выключен

1: Включён

NB: В режиме I^2S не используется.

— Биты 5:3 **BR[2:0]:** Скорость передачи

000: $f_{PCLK}/2$

001: $f_{PCLK}/4$

010: $f_{PCLK}/8$

011: $f_{PCLK}/16$

100: $f_{PCLK}/32$

101: $f_{PCLK}/64$

110: $f_{PCLK}/128$

111: $f_{PCLK}/256$

NB: В режиме I^2S не используется. Во время передачи изменять нельзя.

— Бит 2 **MSTR:** Выбор ведущего

0: Ведомый

1: Ведущий

NB: В режиме I^2S не используется. Во время передачи изменять нельзя.

— Бит 1 **CPOL:** Полярность тактов

0: CK равен 0 в простое

1: CK равен 1 в простое

NB: В режиме I^2S не используется. Во время передачи изменять нельзя.

— Бит 0 **CPHA:** Фаза тактов

0: Фронт считывания в первом такте

1: Фронт считывания во втором такте

NB: В режиме I^2S не используется. Во время передачи изменять нельзя.

25.5.2. Регистр 2 управления SPI (SPI_CR2)

Смещение адреса: 0x04

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TXEIE	RXNEIE	ERRIE	Res.	Res.	*SSOE*	TXDMAEN	RXDMAEN
								rw	rw	rw			rw	rw	rw

— Биты 15:8 Резерв, не трогать.

— Бит 7 **TXEIE:** Разрешение прерывания пустого буфера Tx

0: Прерывание TXE запрещено

1: Прерывание TXE разрешено.

— Бит 6 **RXNEIE:** Разрешение прерывания непустого буфера RX

0: Прерывание RXNE запрещено

1: Прерывание RXNE разрешено.

— Бит 5 **ERRIE:** Разрешение прерывания ошибки

Прерывание по ошибке (CRCERR, OVR, MODF в режиме SPI и UDR, OVR в режиме I^2S).

0: Прерывание ошибки запрещено

1: Прерывание ошибки разрешено.

— Биты 4:3 Резерв, не трогать.

— Бит 2 **SSOE:** Разрешение вывода SS

0: В режиме ведущего вывод SS выключен, ячейка может работать режиме нескольких ведущих

1: В режиме ведущего вывод SS включен, ячейка не может работать режиме нескольких ведущих

NB: В режиме I^2S не используется.

— Бит 1 **TXDMAEN:** Разрешение DMA буфера Tx

При стоящем бите запрос DMA выдаётся при установке флага TXE.

0: Нельзя

1: Можно

— Бит 0 **RXDMAEN:** Разрешение DMA буфера Rx

При стоящем бите запрос DMA выдаётся при установке флага RXNE.

0: Нельзя

1: Можно

25.5.3. Регистр состояния SPI (SPI_SR)

Смещение адреса: 0x08

По сбросу: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BSY	OVR	MODF	CRC ERR	UDR	CHSIDE	TXE	RXNE
								r	r	r	rc_w0	r	r	r	r

— Биты 15:8 Резерв, не трогать.

— Бит 7 **BSY:** Занят

Ставится и снимается аппаратно.

0: SPI (или I^2S) свободен

1: SPI (или I^2S) занят связью или буфер Tx не пуст

— Бит 6 **OVR:** Переполнение

Ставится аппаратно, снимается программной последовательностью

0: Не было

1: Было

— Бит 5 **MODF:** Сбой режима

Ставится аппаратно, снимается программной последовательностью

0: Не было

1: Было

NB: В режиме I^2S не используется.

— Бит 4 **CRCERR:** Ошибка CRC

Ставится аппаратно, снимается записью 0

0: Значение CRC совпало с SPI_RXCRCR

1: Значение CRC не совпало с SPI_RXCRCR

NB: В режиме I²S не используется.

— Бит 3 **UDR:** Исчерпание

Ставится аппаратно, снимается программной последовательностью

0: Не было

1: Было

NB: В режиме I²S не используется.

— Бит 2 **CHSIDE**: Номер канала

0: Надо передать или принят левый канал

1: Надо передать или принят правый канал

NB: В режиме SPI не используется. В PCM не имеет смысла.

— Бит 1 **TXE:** Буфер Tx пуст

0: Не пуст

1: Пуст

— Бит 0 **RXNE:** Буфер Rx не пуст

0: Пуст

1: Не пуст

25.5.4. Регистр данных SPI (SPI_DR)

Смещение адреса: 0x0C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **DR[15:0]:** Регистр данных приёма или передачи

Это 2 регистра, Tx для записи и Rx для чтения, расположенных по одному адресу.

NB: Примечания для SPI:

Формат данных (8- или 16-бит) надо выбирать битом DFF регистра SPI_CR1 до включения SPI.

При 8-бит фрейме данных используется только младший байт (SPI_DR[7:0]), при приёме старший байт (SPI_DR[15:8]) аппаратно обнуляется.

При 16-бит фрейме данных используется весь регистр.

25.5.5. Регистр полинома CRC SPI (SPI_CRCPR) (в I²S не используется)

Смещение адреса: 0x10

По сбросу: 0x0007

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **CRCPOLY[15:0]:** Полином вычисления CRC

По сбросу содержит число 0007h. Можно и другой.

NB: В режиме I²S не используется.

25.5.6. Регистр RX CRC SPI (SPI_RXCRCR) (в I²S не используется)

Смещение адреса: 0x14

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 15:0 **RXCRC[15:0]:** Rx CRC

При включённом вычислении CRC биты RxCRC[15:0] содержат CRC последовательно принятых байтов. Сбрасывается записью 1 в бит CRCEN регистра SPI_CR1.

В вычислениях участвуют младшие 8 бит (стандарт CRC8) или весь регистр (стандарт CRC16) в зависимости от бита DFF регистра SPI_CR1

NB: При стоящем флаге BSY читается ерунда. В режиме I²S не используется.

25.5.7. Регистр TX CRC SPI (SPI_TXCRCR) (в I²S не используется)

Смещение адреса: 0x18

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 15:0 **TXCRC[15:0]:** Rx CRC

При включённом вычислении CRC биты TxCRC[15:0] содержат CRC последовательно переданных данных. Сбрасывается записью 1 в бит CRCEN регистра SPI_CR1.

В вычислениях участвуют младшие 8 бит (стандарт CRC8) или весь регистр (стандарт CRC16) в зависимости от бита DFF регистра SPI_CR1

NB: При стоящем флаге BSY читается ерунда. В режиме I²S не используется.

25.5.8. Регистр конфигурации SPI_I²S (SPI_I2SCFGR)

Смещение адреса: 0x1C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				I2SMOD	I2SE	I2SCFG		PCMSYNC	Res.	I2SSTD		CKPOL	DATLEN		CHLEN
				rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

— Биты 15:12 Резерв, не трогать.

— Бит 11 **I2SMOD**: Режим I²S

0: Режим SPI

1: Режим I²S

NB: Писать можно при выключенных SPI и I²S

— Бит 10 **I2SE**: Включение I²S

0:Выключен

1: Включён

NB: В режиме SPI не используется.

— Биты 9:8 **I2SCFG**: Конфигурация I²S

00: Ведомый - передача

01: Ведомый - приём

10: Ведущий - передача

11: Ведущий - приём

NB: В режиме SPI не используется. Писать можно при выключенном I²S.

— Бит 7 **PCMSYNC**: Синхронизация фрейма PCM

0: Короткая

1: Длинная

NB: В режиме SPI не используется. Имеет значение только при I2SSTD = 11 (стандарт PCM).

— Бит 6 Резерв, не трогать.

— Биты 5:4 **I2SSTD**: Стандарт I²S

00: Стандарт I²S Philips.

01: Левое выравнивание

10: Правое выравнивание

11: Стандарт PCM

NB: В режиме SPI не используется. Писать можно при выключенном I²S.

— Бит 3 **CKPOL**: Полярность стабильного такта I²S

0: Низкий

1: Высокий

NB: В режиме SPI не используется. Писать можно при выключенном I²S.

— Биты 2:1 **DATLEN**: Длина передаваемых данных

00: 16-бит

01: 24-бита

10: 32-бита

11: Нельзя

NB: В режиме SPI не используется. Писать можно при выключенном I²S.

— Бит 0 **CHLEN**: Длина канала (число бит канала)

0: 16-бит

1: 32-бита

Имеет смысл только при DATLEN = 00, иначе аппаратно равна 32-бита, независимо от этого значения.

NB: В режиме SPI не используется. Писать можно при выключенном I²S.

25.5.9. Регистр конфигурации SPI_I²S (SPI_I2SCFGR)

Смещение адреса: 0x20

По сбросу: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						MCKOE	ODD	I2SDIV							
						rw	rw	rw							

— Биты 15:10 Резерв, не трогать.

— Бит 9 **MCKOE**: Разрешение вывода ведущих тактов

0: Нельзя

1: Нужно

NB: Писать можно при выключенном I²S. Используется только в ведущем I²S.

— Бит 8 **ODD**: Нечётный предделитель

0: предделитель = I2SDIV *2

1: предделитель = (I2SDIV * 2)+1

NB: Писать можно при выключенном I²S. Используется только в ведущем I²S.

— Биты 7:0 **I2SDIV**: Линейный предделитель I²S

Значения I2SDIV [7:0] = 0 и I2SDIV [7:0] = 1 запрещены.

NB: Писать можно при выключенном I²S. Используется только в ведущем I²S.

25.5.10.Карта регистров SPI

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**SPI_CR1**	Reserved																BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	*LSBFIRST*	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**SPI_CR2**	Reserved																								TXEIE	RXNEIE	ERRIE	Reserved		*SSOE*	TXDMAEN	RXDMAEN
	Reset value																									0	0	0			0	0	0
0x08	**SPI_SR**	Reserved																								BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset value																									0	0	0	0	0	0	1	0
0x0C	**SPI_DR**	Reserved																DR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**SPI_CRCPR**	Reserved																CRCPOLY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	**SPI_RXCRCR**	Reserved																RxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**SPI_TXCRCR**	Reserved																TxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**SPI_I2SCFGR**	Reserved																				I2SMOD	I2SE	I2SCFG		PCMSYNC	Reserved	I2SSTD		CKPOL	DATLEN	CHLEN	
	Reset value																					0	0	0	0	0		0	0	0	0	0	0
0x20	**SPI_I2SPR**	Reserved																						MCKOE	ODD	I2SDIV							
	Reset value																							0	0	0	0	0	0	0	0	1	0

26. Интерфейс I²C

26.1. Введение в I²C

Шина "промеж-интегрального" (inter-integrated circuit) интерфейса поддерживает несколько ведущих устройств и имеет два режима скорости: стандартный (Sm, до 100 kHz) и быстрый (Fm, до 400 kHz).

Может использоваться в разных целях, включая вычисление и проверку CRC, SMBus (шину управления системой) и PMBus (шину управления питанием).

26.2. Основные свойства I²C

- Преобразователь протокола параллельная шина/I²C
- Несколько ведущих: один интерфейс может работать Ведущим или Ведомым
- Ведущий I²C:
  - Выдача тактов
  - Выдача сигналов Start и Stop
- Ведомый I²C:
  - Определение программируемого адреса I²C
  - Подтверждение на 2 адреса ведомых
  - Обнаружение сигнала Stop
- Формирование и определение 7-бит/10-бит адресов и Общий Вызов
- Скорость связи:
  - Стандартная (до 100 kHz)
  - Быстрая (до 400 kHz)
- Аналоговый фильтр шума
- Флаги состояния:
  - Флаг Приём/Передача
  - Передача Конец-Байта
  - I²C занят
- Флаги ошибки:
  - Потеря арбитража ведущим
  - Сбой подтверждения передачи адрес/данные
  - Обнаружение неправильного появления старта и останова
  - Переполнение/Исчерпание при выключенной растяжке тактов
- 2 вектора прерываний:
  - Успешная передача адреса/данных
  - Ошибка
- Возможная растяжка тактов
- Однобайтовый буфер с DMA
- Конфигурируемая PEC (проверка ошибки пакетов):
  - Передача PEC последним байтом
  - Проверка PEC для последнего байта
- Совместимость с SMBus 2.0:
  - задержка таймаута тактов 25 ms
  - кумулятивное расширение тактов ведущего 10 ms
  - кумулятивное расширение тактов ведомого 25 ms
  - Аппаратная проверка/ вычисление PEC с контролем `ACK`
  - Протокол разрешения адреса (ARP)
- Совместимость с PMBus

NB: Не во всех устройствах есть всё приведённое выше.

26.3. Функциональное описание I²C

Кроме приёма и передачи есть последовательно/параллельный преобразователь. Прерывания управляются программно. К шине I²C подключается ножками данных (SDA) и тактов (SCL).

26.3.1. Выбор режима

Режимов четыре: Передатчик ведомого, Приёмник ведомого, Передатчик ведущего и Приёмник ведущего.

По умолчанию работает в режиме ведомого. Автоматически переключается в ведущего при генерации START, и в ведомого при потере арбитража или появления STOP.

Связь

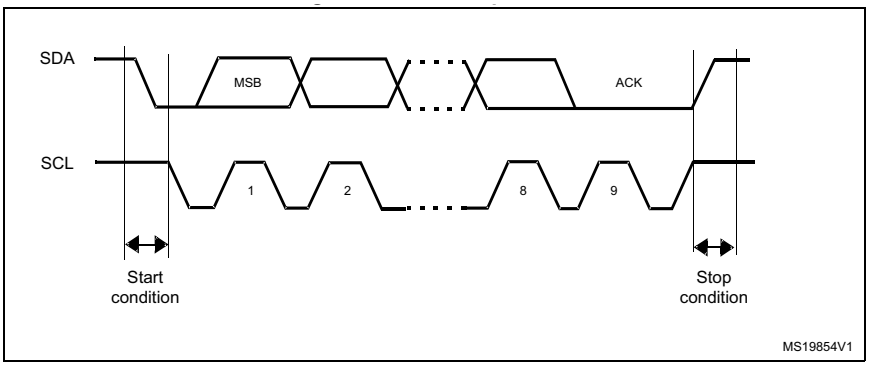
Ведущий инициирует передачу данных и выдаёт такты. Последовательная передача начинается по START и завершается по STOP. Оба события задаются программно в режиме ведущего.

Ведомый узнаёт свой адрес (7 или 10-бит) и адрес Общего Вызова. Опознавание Общего Вызова разрешается программно.

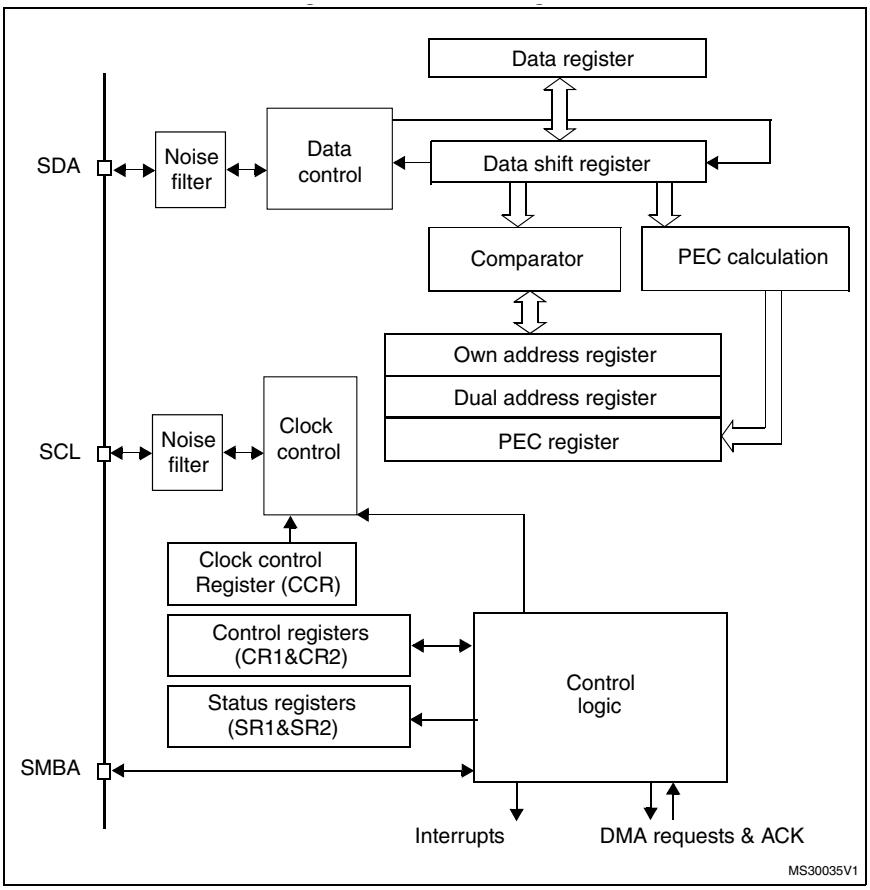
Данные и адреса передаются байтами, старший бит первым. Первым после старта одним или двумя байтами передаётся адрес. Адрес всегда передаётся в режиме ведущего.

Девятым тактом после передачи байта передатчику выдаётся подтверждение. Размер адреса и выдача подтверждения определяются программно.

Протокол шины I²C



Блок схема I²C



1. SMBA работает только при включённой SMBus.

26.3.2. Ведомый I²C

Частота входных тактов программируется в регистре I2C_CR2 и должна быть не менее:

- 2 MHz в режиме Sm
- 4 MHz в режиме Fm

По событию старта с линии SDA в регистр сдвига принимается адрес интерфейса и сравнивается с адресом в OAR1 и с OAR2 (если ENDUAL=1) или адресом Общего Вызова (при ENGC = 1).

NB: В режиме 10-бит адресации сравнение включает заголовок (11110xx0), где xx это старшие биты адреса.

Заголовок или адрес не совпали: игнорируем и ждём нового Старта.

Заголовок совпал (10-бит режим): при стоящем бите ACK выдаём подтверждение и ждём 8-бит адрес ведомого.

Адрес совпал: генерируем последовательность:

- При стоящем бите ACK выдаём подтверждение
- Аппаратно ставится бит ADDR и при стоящем бите ITEVFEN выдаётся прерывание.
- При ENDUAL=1 по биту DUALF уточняем совпавший адрес ведомого.

В 10-бит режиме после приёма адреса ведомый остаётся на Приёме, на Передачу он пойдёт после получения повторного старта с совпадающим адресом и стоящим младшим битом (11110xx1). Бит TRA показывает Приём или Передачу.

Передача ведомого

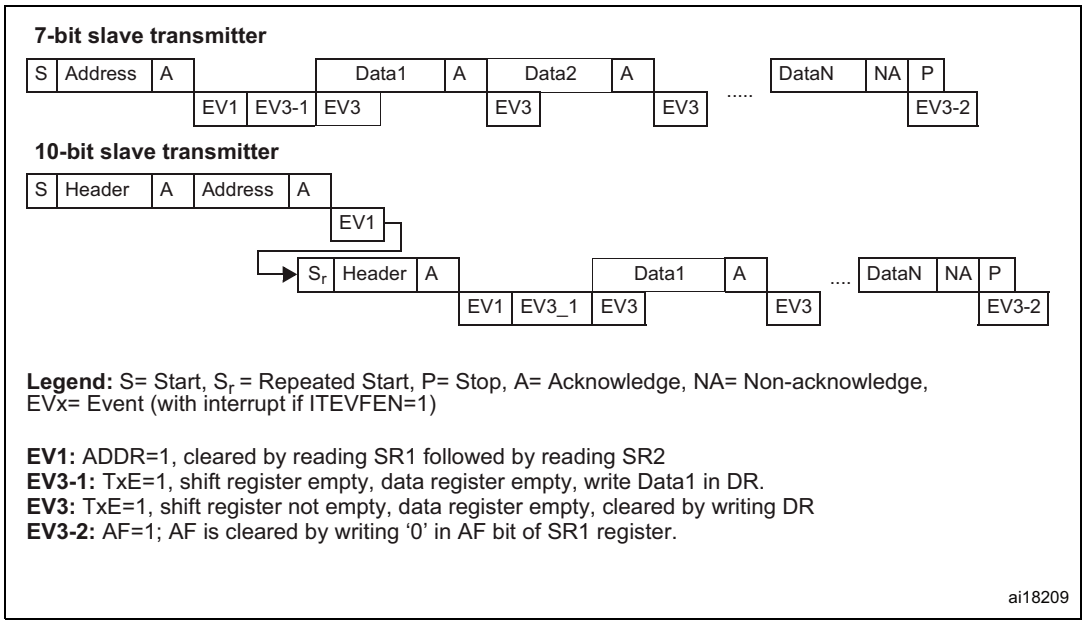
Вслед за приёмом адреса и очисткой ADDR, ведомый через регистр сдвига отсылает байты из регистра DR на линию SDA.

Ведомый растягивает низкий SCL вплоть до очистки ADDR и записи DR (EV1 EV3 в диаграмме).

После приёма подтверждения:

- Аппаратно ставится бит TxE с прерываниями при стоящих битах ITEVFEN и ITBUFEN.
- При стоящем бите TxE, когда до конца передачи в регистр I2C_DR ещё не записан следующий передаваемый байт, ставится бит BTF и растягивается низкий SCL вплоть до очистки BTF чтением I2C_SR1 с последующей записью в регистр I2C_DR.

Последовательность передачи ведомым



Legend: S= Start, S_r = Repeated Start, P= Stop, A= Acknowledge, NA= Non-acknowledge, EVx= Event (with interrupt if ITEVFEN=1)

EV1: ADDR=1, cleared by reading SR1 followed by reading SR2
EV3-1: TxE=1, shift register empty, data register empty, write Data1 in DR.
EV3: TxE=1, shift register not empty, data register empty, cleared by writing DR
EV3-2: AF=1; AF is cleared by writing '0' in AF bit of SR1 register.

ai18209

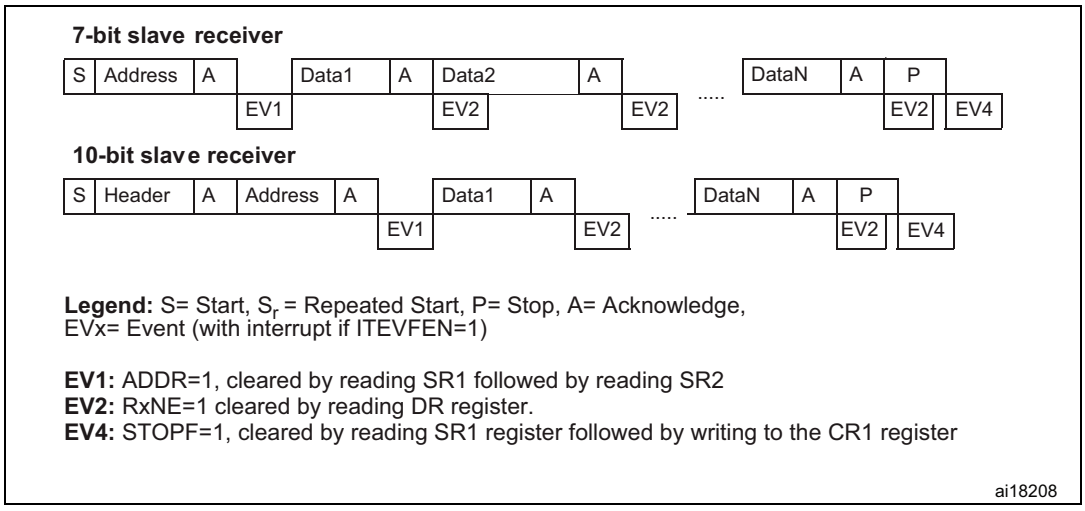
Приём ведомым

Следом за приёмом адреса после очистки ADDR ведомый из линии SDA через регистр сдвига принимает байты в регистр DR. После каждого байта:

- При стоящем бите ACK выдаётся импульс подтверждения
- Аппаратно ставится бит RxNE с прерываниями при стоящих битах ITEVFEN и ITBUFEN.

При стоящем бите RxNE, когда до конца приёма следующего байта регистр DR ещё не прочитан, ставится бит BTF и растягивается низкий SCL вплоть до очистки BTF чтением регистра I2C_SR1 с последующим чтением регистра I2C_DR.

Последовательность приёма ведомым



1. Событие EV1 растягивает низкий SCL до конца соответствующей программной последовательности.
2. Программное событие EV2 должно завершиться до конца текущей.
3. При проверке регистра SR1 надо полностью обработать и снять все стоящие флаги.

Закрытие связи ведомого

После передачи последнего байта данных ведущий выдаёт сигнал Стоп, и ведомый по стоящему биту STOPF выдаёт прерывание, если разрешено битом ITEVFEN.

Бит STOPF снимается чтением регистра SR1 с последующей записью в регистр CR1 (см. EV4).

26.3.3. Ведущий I²C

В режиме ведущего I²C инициирует передачу данных и выдаёт тактовый сигнал. Передача данных начинается сигналом Старт и завершается сигналом Стоп.

Режим ведущего включается по сигналу Старт на шине, посылаемым битом START.

Ведущий должен:

- Установить времянку входных тактов регистром I2C_CR2
- Установить регистры управления тактами
- Установить регистр времени подъёма
- Включить периферию в регистре I2C_CR1
- Поставить бит START в регистре I2C_CR1 для выдачи сигнала Старт

Частота тактов должна быть не меньше:

- 2 MHz в режиме Sm
- 4 MHz в режиме Fm

Выдача тактов SCL

Выдачу высокого и низкого уровня SCL определяют биты CCR, начиная с переднего фронта. Поскольку ведомый может затягивать такты линии SCL, то I²C проверяет вход SCL в конце времени, заданного битами TRISE после переднего фронта тактового сигнала.

- Если линия SCL низкая, то есть ведомый растягивает такты, то счётчик высокого уровня стопорится до появления высокого уровня SCL. Так обеспечивается минимальный период параметра HIGH такта SCL.
- Если линия SCL высокая, то счётчик высокого уровня продолжает считать.

В действительности, от выдачи переднего фронта SCL до его обнаружения в ответе проходит некоторое время, даже если ведомый не растягивает такты. Длительность задержки связана с временем подъёма SCL (влияние обнаружения SCL VIH), плюс задержка входного аналогового фильтра шума линии SCL, плюс задержка внутренней синхронизации SCL и тактов APB. Максимальная задержка определена в битах TRISE, так что частота SCL остаётся стабильной.

Сигнал Старт

При чистом бите BUSY установка бита START выводит сигнал Старт и переключает в режим ведущего (стоит бит MSL).

NB: В режиме ведущего установка бита START приводит к Рестарту в конце передачи текущего байта.

После посылки Старта аппаратно ставится бит SB и при стоящем бите ITEVFEN выдаётся прерывание. Затем ведущий ждёт чтения регистра SR1 с последующей записью в регистр DR адреса ведомого (см, EV5).

Передача адреса ведомого

Далее на SDA выдвигается адрес ведомого.

- При 10-бит адресации посылка заголовка вызывает следующие события:
  - Аппаратно ставится бит ADD10 и при разрешении битом ITEVFEN выдаётся прерывание. Затем ведущий ждёт чтения регистра SR1 с последующей записью второго байта адреса.
  - Аппаратно ставится бит ADDR и при разрешении битом ITEVFEN выдаётся прерывание. Затем ведущий ждёт чтения регистра SR1 с последующим чтением регистра SR2.
- При 7-бит адресации после посылки байта аппаратно ставится бит ADDR и при разрешении битом ITEVFEN выдаётся прерывание. Затем ведущий ждёт чтения регистра SR1 с последующим чтением регистра SR2.

Переход ведущего в Передачу или Приём задаёт младший бит посланного адреса ведомого.

- При 7-бит адресации:
  - Для режима Передачи младший бит адреса ведомого сброшен.
  - Для режима Приёма младший бит адреса ведомого стоит.
- При 10-бит адресации:
  - Для режима Передачи ведущий посылает заголовок (11110xx0) и затем адрес ведомого (xx это два старших бита адреса).
  - Для режима Приёма ведущий посылает заголовок (11110xx0) и затем адрес ведомого. Далее он должен послать повторный Старт с заголовком (11110xx1) (xx это два старших бита адреса).

Режим приёма или передачи ведущего показывает бит TRA.

Ведущий передатчик

Байты данных посылаются из регистра DR после передачи адреса и очистки ADDR.

Ведущий ждёт записи первого байта в I2C_DR (событие EV8_1).

После приёма импульса подтверждения аппаратно ставится бит TxE и при разрешении битами ITEVFEN и ITBUFEN выдаётся прерывание.

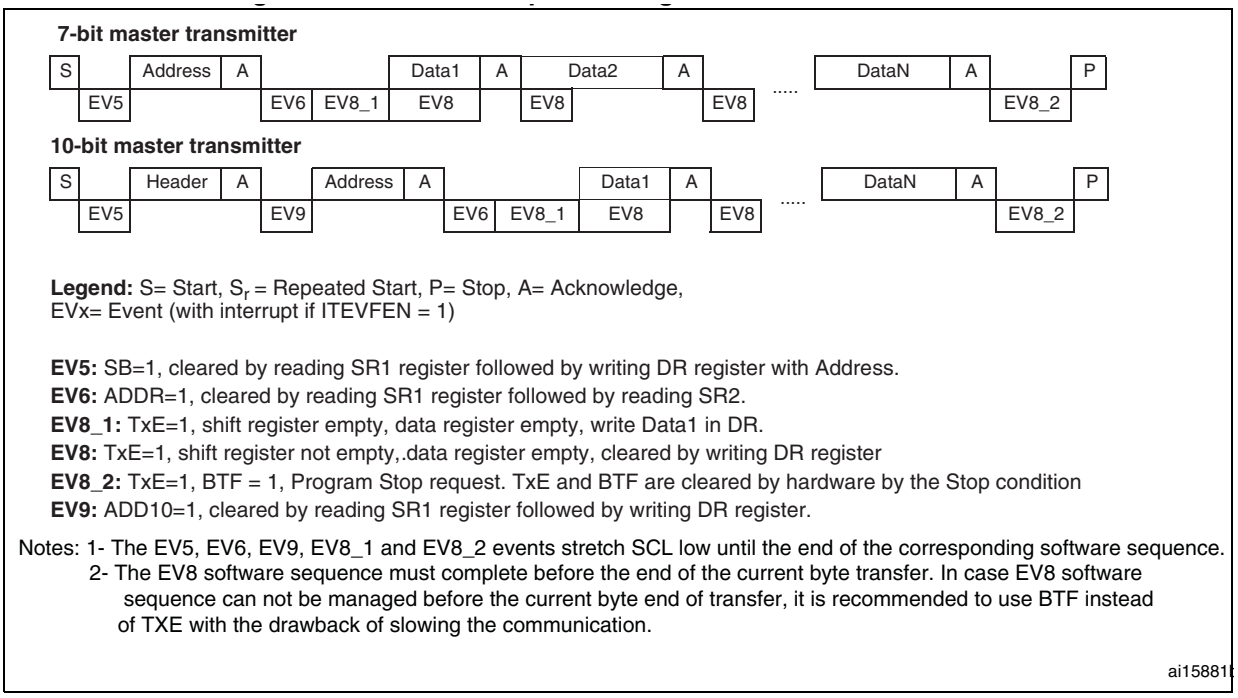
При стоящем бите TxE, когда до конца передачи текущего байта регистр DR ещё не записан, ставится бит BTF и растягивается низкий SCL вплоть до очистки BTF чтением регистра I2C_SR1 с последующей записью регистра I2C_DR.

Закрытие связи

После записи в регистр DR последнего байта, программа ставит бит STOP для выдачи сигнала Стоп (событие EV8_2). Аппарат возвращается в режим ведомого (снимается бит MSL).

NB: Бит STOP во время EV8_2 надо писать при стоящем бите TxE либо BTF.

Последовательность работы ведущего передатчика.



Ведущий приёмник

I²C переключается в ведущего после передачи адреса и очистки ADDR, теперь он принимает данные в регистр DR. После каждого байта:

1. При стоящем бите ACK выдаёт импульс подтверждения
2. При стоящем бите RxNE и разрешении битами ITEVFEN и ITBUFEN выдаёт прерывание (событие EV7).

При стоящем бите RxNE, когда до конца приёма следующего байта регистр DR ещё не прочитан, ставится бит BTF и растягивается низкий SCL вплоть до очистки BTF чтением регистра I2C_SR1 с последующим чтением регистра I2C_DR.

Закрытие связи

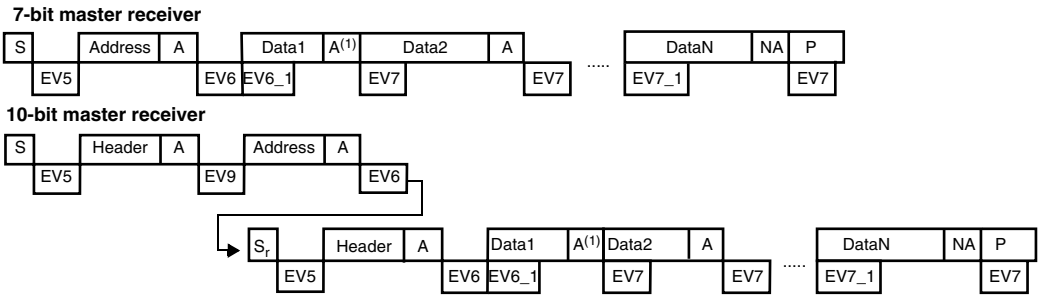
Метод 1: Используется, когда I²C работает с прерываниями высшего приоритета.

В ответ на последний принятый байт, ведущий посылает NACK и ведомый освобождает линии SCL и SDA. Теперь ведущий может сигнал Стоп/Рестарт.

1. Для посылки NACK нужно снять бит ACK сразу после приёма предпоследнего байта (после предпоследнего RxNE).
2. Для выдачи сигнала Стоп/Рестарт надо поставить бит STOP/START сразу после чтения предпоследнего байта (после предпоследнего RxNE).
3. В случае приёма одного байта, снятие подтверждения и выдача Стоп выполняются сразу после EV6 (в EV6_1, сразу после очистки ADDR).

После выдачи сигнала Стоп I²C переходит в режим ведомого (бит MSL снимается).

Метод 1. Последовательность приёма ведущим.



Legend: S= Start, S_r = Repeated Start, P= Stop, A= Acknowledge, NA= Non-acknowledge, EVx= Event (with interrupt if ITEVFEN=1)

EV5: SB=1, cleared by reading SR1 register followed by writing DR register.

EV6: ADDR=1, cleared by reading SR1 register followed by reading SR2. In 10-bit master receiver mode, this sequence should be followed by writing CR2 with START = 1.

EV6_1: no associated flag event, used for 1 byte reception only. The Acknowledge disable and Stop condition generation are made just after EV6, that is after ADDR is cleared.

EV7: RxNE=1 cleared by reading DR register.

EV7_1: RxNE=1 cleared by reading DR register, program ACK=0 and STOP request

EV9: ADD10=1, cleared by reading SR1 register followed by writing DR register.

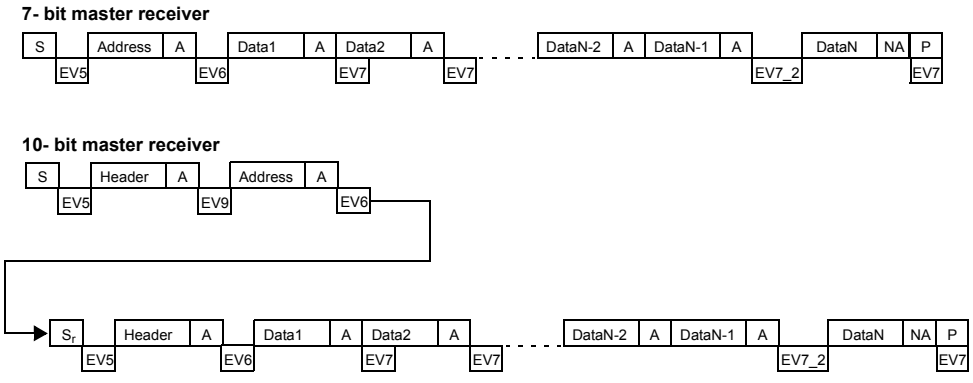
ai15882

1. Если принят один байт, то он NA.
2. События EV5, EV6 и EV9 растягивают низкий SCL до конца программной последовательности.
3. Программная последовательность EV7 должна завершиться до конца передачи текущего байта. Если это невозможно, то вместо RXNE надо использовать BTF.
4. Программные последовательности EV6_1 и EV7_1 должны завершиться до выдачи импульса ACK текущей передачи байта.

Метод 2: Используется при опросе I²C или использовании прерывания не самого высокого приоритета.

При этом, `DataN-2` сразу не читается, так что после `DataN-1` связь затягивается (стоят и `RxNE` и `BTF`). Далее, чистим бит `ACK` перед чтением `DataN-2` из `DR` чтобы он был чист перед импульсом подтверждения `DataN`. После этого, после чтения `DataN-2`, ставим бит `STOP`/ `START` и читаем `DataN-1`. После установки `RxNE` читаем `DataN`.

Метод 2: Последовательность приёма ведущим при N>2



Legend: S = Start, S_r = Repeated Start, P = Stop, A = Acknowledge, NA = Non-acknowledge, EVx = Event (with interrupt if ITEVFEN = 1)

EV5: SB=1, cleared by reading SR1 register followed by writing the DR register.

EV6: ADDR1, cleared by reading SR1 register followed by reading SR2.
In 10-bit master receiver mode, this sequence should be followed by writing CR2 with START = 1.

EV7: RxNE=1, cleared by reading DR register

EV7_2: BTF = 1, DataN-2 in DR and DataN-1 in shift register, program ACK = 0, Read DataN-2 in DR. Program STOP = 1, read DataN-1.

EV9: ADD10= 1, cleared by reading SR1 register followed by writing DR register.

1. События EV5, EV6 и EV9 растягивают низкий SCL до конца программной последовательности.
2. Программная последовательность EV7 должна завершиться до конца передачи текущего байта. Если это невозможно, то вместо RXNE надо использовать BTF.

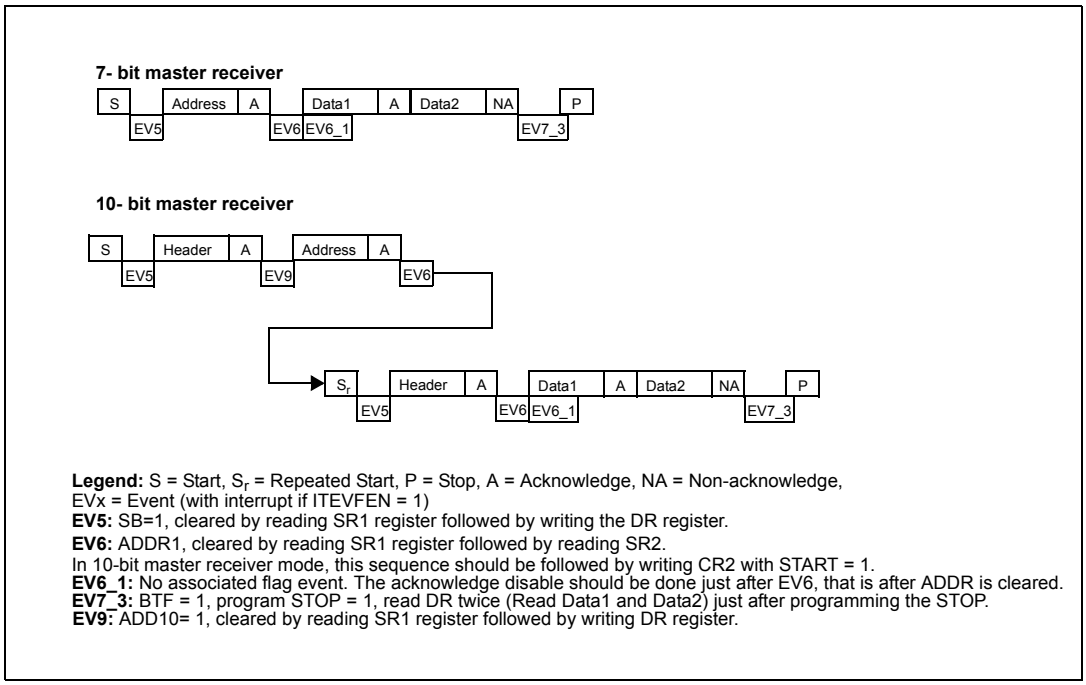
Когда остаётся прочесть 3 байта:

- RxNE = 1 => Ничего (DataN-2 не прочитан).
- DataN-1 принят
- BTF = 1 потому что заняты оба регистра: DataN-2 в DR и DataN-1 в регистре сдвига => SCL держится низким: данные по шине не принимаются.
- Чистим бит ACK
- Читаем DataN-2 из DR => этим запускается приём DataN в регистр сдвига
- DataN принят (с NACK)
- Пишем START/STOP
- Читаем DataN-1
- RxNE = 1
- Читаем DataN

Описанная процедура работает при N>2. В случае приёма одного или двух байтов надо так:

- Принимаем один байт:
  - По событию ADDR снимаем бит ACK.
  - Чистим ADDR
  - Пишем бит START/STOP.
  - Читаем байт после установки флага RxNE.
- Принимаем два байта:
  - Ставим POS и ACK
  - Ждём установку флага ADDR
  - Чистим ADDR
  - Чистим ACK
  - Ждём установку флага BTF
  - Пишем бит STOP
  - Дважды читаем DR

Метод 2: Последовательность приёма ведущим при N=2

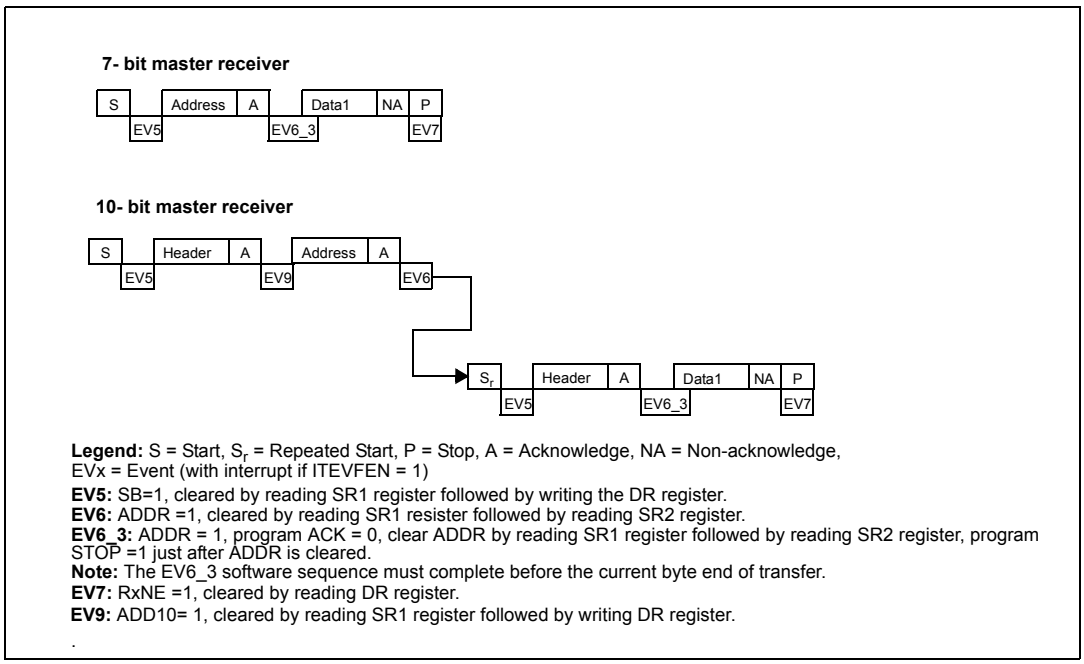


Legend: S = Start, S_r = Repeated Start, P = Stop, A = Acknowledge, NA = Non-acknowledge,
EVx = Event (with interrupt if ITEVFEN = 1)
EV5: SB=1, cleared by reading SR1 register followed by writing the DR register.
EV6: ADDR1, cleared by reading SR1 register followed by reading SR2.
In 10-bit master receiver mode, this sequence should be followed by writing CR2 with START = 1.
EV6_1: No associated flag event. The acknowledge disable should be done just after EV6, that is after ADDR is cleared.
EV7_3: BTF = 1, program STOP = 1, read DR twice (Read Data1 and Data2) just after programming the STOP.
EV9: ADD10= 1, cleared by reading SR1 register followed by writing DR register.

1. События EV5, EV6 и EV9 растягивают низкий SCL до конца программной последовательности.

2. Программная последовательность EV6_1 должна завершиться до выдачи импульса ACK текущей передачи байта.

Метод 2: Последовательность приёма ведущим при N=1



Legend: S = Start, S_r = Repeated Start, P = Stop, A = Acknowledge, NA = Non-acknowledge,
EVx = Event (with interrupt if ITEVFEN = 1)
EV5: SB=1, cleared by reading SR1 register followed by writing the DR register.
EV6: ADDR =1, cleared by reading SR1 resister followed by reading SR2 register.
EV6_3: ADDR = 1, program ACK = 0, clear ADDR by reading SR1 register followed by reading SR2 register, program STOP =1 just after ADDR is cleared.
Note: The EV6_3 software sequence must complete before the current byte end of transfer.
EV7: RxNE =1, cleared by reading DR register.
EV9: ADD10= 1, cleared by reading SR1 register followed by writing DR register.
.

1. События EV5, EV6 и EV9 растягивают низкий SCL до конца программной последовательности.

26.3.4. Ошибки

Эти ошибки могут привести к сбою связи.

Ошибка шины (BERR)

Появляется при обнаружении I²C внешнего сигнала Старт или Стоп при передаче адреса или данных. В этом случае:

- Ставится бит BERR и при разрешении битом ITERREN выдаётся прерывание
- В режиме ведомого: данные отбрасываются и линии аппаратно освобождаются:
  - при несвоевременном Старт ведомый считает это Рестартом и ждёт адрес или сигнал Стоп
  - при несвоевременном Стоп ведомый, как и при нормальном Стоп освобождает линии
- В режиме ведущего: линии не освобождаются и текущая операция не прекращается, над этим пусть думает программа.

Сбой подтверждения (AF)

Возникает при обнаружении бита не-подтверждения. В этом случае:

- ставится бит AF и при разрешении битом ITERREN выдаётся прерывание
- передатчик, получивший NACK, должен сбросить связь:
  - Ведомый: аппаратно освобождает линии
  - Ведущий: программа должна выдать Стоп или Рестарт

Потеря арбитража (ARLO)

Обнаруживается I²C. В этом случае:

- аппаратно ставится бит ARLO (прерывание зависит от разрешения битом ITERREN)
- I²C автоматически возвращается в ведомый (снимается бит MSL). При потере арбитража I²C не может подтверждать адрес ведомого вплоть до Рестарта от ведущего-победителя.
- линии аппаратно освобождаются

Переполнение/Исчерпание (OVR)

Переполнение возникает при отключённой растяжке тактов, когда принимается байт (RxNE=1) при непрочитанном регистре DR. В этом случае:

- Последний принятый байт теряется.
- Программа должна очистить бит RxNE и передатчику надо повторить передачу.

Исчерпание возникает у ведомого передатчика при отключённом растягивании тактов. В регистр DR не был записан следующий байт (TxE=1), а такт за ним пришёл. В этом случае:

- Будет послан тот же байт из регистра DR

- Программа должна выбрасывать байты, полученные при состоянии исчерпания, и следующие записаны во время низкого такта, определённого стандартом шины I²C bus.

Первый передаваемый байт должен быть записан в DR после снятия ADDR до первого переднего фронта SCL. Если это невозможно, то первый байт надо выбрасывать.

26.3.5. Линии SDA/SCL

- Если растягивание тактов разрешено:

– Передача: Если TxE=1 и BTF=1: перед передачей I²C должен удерживать низкий такт, чтобы MCU успел прочитать SR1, и затем писать байт в DR (буфер и регистр сдвига пусты).

– Приём: Если RxNE=1 и BTF=1: после приёма I²C держит линию тактов низкой, чтобы MCU успел прочитать SR1, и затем прочесть байт из DR (буфер и регистр сдвига полные).

- Если растягивание тактов в режиме ведомого запрещено:

– Ошибка переполнения при RxNE=1 и не читался DR до приёма следующего байта. Последний принятый байт теряется.

– Ошибка исчерпания при TxE=1 и ещё не писался DR для передачи, а пора передавать. Посылается тот же байт.

– Коллизия записи не управляется.

26.3.6. SMBus

Введение

Двухпроводная шина управления системой (SMBus) основана на принципах работы I²C.

В SMBus определены три типа устройств. *Ведомый* принимает и отвечает на команды. *Ведущий* выдаёт команды и такты и завершает передачи. *Хост* это специализированный ведущий для связи с CPU системы. Хост должен быть ведущим-ведомым и поддерживать протокол нотификации хоста SMBus. Хост в системе один.

Совпадения SMBus и I²C

- 2 провода шины (1 такты, 1 данные) + необязательная линия извещения SMBus
- Связь ведущий-ведомый, Такты выдаёт ведущий
- Несколько ведущих
- Формат данных SMBus подобен формату 7-бит адресации I²C.

Отличия SMBus и I²C

Таблица 188. SMBus против I²C

SMBus	I²C
Макс. частота 100 kHz	Макс. частота 400 kHz
Мин. частота 10 kHz	Не ограничена
35 ms таймаут низкого такта	Таймаута нет
Логические уровни фиксированы	Логические уровни зависят от V_{DD}
Разные типы адресации (резервный, динамический и пр.)	7-бит, 10-бит и Общий Вызов
Разные протокола шины (быстрая команда, вызов процесса и пр.)	Протокола шины нет

Использование SMBus

Через SMBus устройство выдаёт информацию производителя, номер модели/части, сохраняет своё состояние для режима задержки, извещает об ошибке, получает параметры управления и возвращает своё состояние.

Идентификация устройства

Все устройства на SMBus имеют свой уникальный адрес ведомого. Список резервных адресов ведомых выложен в спецификации SMBus версии 2.0 (*http://smbus.org/*).

Протоколы шины

SMBus поддерживает до девяти протоколов шины они реализуются программно.

Протокол разрешения адресов (ARP)

Конфликты адресов ведомых SMBus разрешаются динамически назначением новых адресов. Протокол Разрешения Адресов (ARP) имеет атрибуты:

- Назначение адресов использует стандартный механизм арбитража физического уровня SMBus
- Назначенные адреса сохраняются при включённом питании, можно и при выключенном.
- Назначенные адреса SMBus не требуют дополнительных пакетов шины по сравнению с фиксированными.
- Любой ведущий шины SMBus может выполнять нумерацию устройств.

Уникальный идентификатор устройства (UDID)

Все устройства должны иметь свой уникальный 128-бит идентификатор устройства (UDID).

Все подробности приведены в спецификации SMBus версии 2.0 (*http://smbus.org/*).

Режим извещения SMBus

Необязательная линия извещения с прерыванием SMBus (SMBA) это объединение по И сигналов устройств. SMBA используется совместно с адресом Общего Вызова. Сообщения, работающие с SMBus двухбайтовые.

Только ведомые устройства через SMBA извещают хост о желании побеседовать установкой бита ALERT в регистре I2C_CR1. Хост обрабатывает прерывание и одновременно обращается ко всем SMBA устройствам с *Адресом Ответа Извещения* (ARA равным 0001 100X). На адрес отвечают только устройства, удерживающие низким SMBA. Этому служит флаг SMBALERT регистра I2C_SR1. Хост исполняет модифицированный Приём Байта. 7-битный адрес от ведомого передатчика лежит в старших битах байта, младший бит может быть любым.

Из нескольких запрашивающих устройств с низким SMBA связь получает устройство с меньшим адресом в соответствии со стандартным арбитражем при передаче адреса ведомого. После подтверждения адреса ведомого, устройство снимает свой сигнал SMBA. Если SMBA ещё низкий, то хоста заново обращается к ARA.

Хосты без сигнала SMBA могут обращаться к ARA периодически.

Ошибка таймаута

Таймаут I²C и SMBus различается. SMBus определяет таймаут тактов TIMEOUT в 35 ms.

TLOW: SEXT это суммарное время растяжки низкого такта ведомого.

TLOW: MEXT это суммарное время растяжки низкого такта ведущего.

На ошибку таймаута TLOW указывает флаг TIMEOUT в регистре I2C_SR1.

Работа в режиме SMBus

Переход из режима I²C в режим SMBus:

- Ставим бит SMBUS регистра I2C_CR1
- Пишем биты SMBTYPE и ENARP регистра I2C_CR1

Конфигурация ведущим выполняется процедурой Старт, см. Секцию 26.3.3, иначе Секцию 26.3.2. Протоколы SMBus реализуются программно.

- SMB Device Default Address подтверждается если ENARP=1 и SMBTYPE=0
- SMB Host Header подтверждается если ENARP=1 и SMBTYPE=1
- SMB Alert Response Address подтверждается если SMBALERT=1

26.3.7. Запросы DMA

Разрешаемые запросы DMA используются только для передачи данных. Они выдаются пустым DR при передаче и полным DR при приёме. DMA инициализируют и включают до передачи данных. Бит DMAEN регистра I2C_CR2 ставят до события ADDR. При разрешённой растяжке тактов бит DMAEN

можно ставить во время события ADDR, перед очисткой флага ADDR. Запрос DMA нужно обслужить до конца текущей передачи байта. При достижении установленного числа передач DMA контроллер DMA посылает I²C сигнал EOT (Конец Передачи), чем может вызвать прерывание:

- Ведущий передатчик: обработчик прерывания EOT выключает запросы DMA и ждёт события BTF для выдачи сигнала Стоп.
- Ведущий приёмник: Если число передаваемых байтов больше или равно двум, то для предпоследнего байта контроллер DMA посылает аппаратный сигнал EOT_1. Если в регистре I2C_CR2 стоит бит LAST, то I²C автоматически посылает а NACK для следующего за EOT_1 байта. Сигнал Стоп можно выдавать из обработчика прерывания EOT.

Передача с DMA

Включается установкой бита DMAEN в регистре I2C_CR2. Данные пишутся из памяти в I2C_DR по установке бита TxE. Последовательность подключения потока DMA x (где x это номер потока) к передаче I²C такова:

1. Пишем адрес регистра I2C_DR в регистр DMA_SxPAR. Сюда по событию TxE будут писаться данные из памяти.
2. Адрес памяти пишем в регистр DMA_SxMA0R (и в DMA_SxMA1R при двух буферах). Отсюда по событию TxE будут писаться данные в I2C_DR.
3. Общее число передаваемых байт пишем в регистр DMA_SxNDTR. По каждому событию TxE оно декрементируется.
4. Приоритет потока DMA пишем в биты PL[0:1] регистра DMA_SxCR.
5. Ставим бит DIR регистра DMA_SxCR, а прерывания половины и полной передачи, как нам надо.
6. Активируем поток установкой бита EN в регистре DMA_SxCR.

При достижении установленного числа передач, контроллер DMA посылает I²C сигнал EOT/EOT_1 и выдаёт прерывание по вектору потока DMA.

NB: Не ставьте бит ITBUFEN регистра I2C_CR2 при передаче с DMA.

Приём с DMA

Включается установкой бита DMAEN в регистре I2C_CR2. Данные из I2C_DR пишутся в память по приёму байта. Последовательность подключения потока DMA x (где x это номер потока) к приёму I²C такова:

1. Пишем адрес регистра I2C_DR в регистр DMA_SxPAR. Отсюда по событию RxNE будут писаться данные в память.
2. Адрес памяти пишем в регистр DMA_SxMA0R (и в DMA_SxMA1R при двух буферах). Сюда по событию RxNE будут писаться данные из I2C_DR.
3. Общее число передаваемых байт пишем в регистр DMA_SxNDTR. По каждому событию RxNE оно декрементируется.
4. Приоритет потока DMA пишем в биты PL[0:1] регистра DMA_SxCR.
5. Чистим бит DIR регистра DMA_SxCR, а прерывания половины и полной передачи, как нам надо.
6. Активируем поток установкой бита EN в регистре DMA_SxCR.

При достижении установленного числа передач, контроллер DMA посылает I²C сигнал EOT/EOT_1 и выдаёт прерывание по вектору потока DMA.

NB: Не ставьте бит ITBUFEN регистра I2C_CR2 во время приёма с DMA.

26.3.8. Контроль ошибки пакета

Контроль ошибки пакета PEC вычисляется по полиному CRC-8 $C(x) = x^8 + x^2 + x + 1$.

- Вычисление PEC включается установкой бита ENPEC регистра I2C_CR1. PEC вычисляется для всех байтов пакета, включая адреса и биты R/W.
  - При передаче: бит передачи PEC в I2C_CR1 ставят после события TxE последнего байта.
  - При приёме: бит PEC в I2C_CR1 ставят после события RxNE последнего байта, так что при отличии следующего байта от принятого PEC приёмник пошлёт NACK. У Ведущего приёмника NACK выдаётся независимо от результата сравнения PEC. PEC должен стоять до импульса ACK приёма текущего байта.
- Флаг/прерывание PECERR лежит в регистре I2C_SR1.

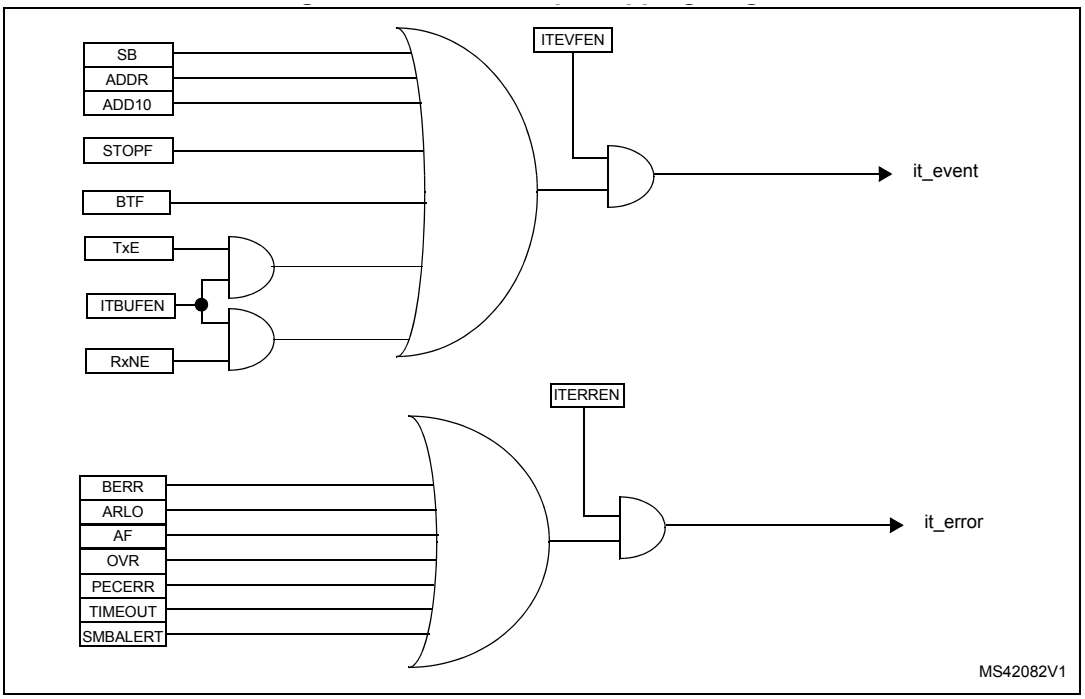
- Если разрешены и DMA и PEC:
  - При передаче: PEC автоматически посылается после получения сигнала EOT от DMA.
  - При приёме: после получения сигнала EOT_1 от DMA, следующий байт воспринимается как PEC и проверяется. Запрос DMA выдаётся после приёма PEC.
- Промежуточной передаче PEC помогает бит LAST регистра I2C_CR2, показывающий действительно последнюю передачу DMA. После последнего принятого байта DMA ведущего приёмника NACK посылается автоматически.
- Вычисление PEC разрушается потерей арбитража.

26.4. Прерывания I²C

Таблица 189. Запросы прерываний I²C

Событие	Флаг	Бит разрешения
Бит Старт послан (Ведущий)	SB	ITEVFEN
Адрес послан (Ведущий) или Адрес совпал (Ведомый)	ADDR	
10-бит заголовок послан (Ведущий)	ADD10	
Стоп принят (Ведомый)	STOPF	
Байт данных передан	BTF	
Буфер приёма не пуст	RxNE	ITEVFEN and ITBUFEN
Буфер передачи пуст	TxE	
Ошибка шины	BERR	ITERREN
Арбитраж потерян loss ()	ARLO	
Сбой подтверждения	AF	
Переполнение/Исчерпание	OVR	
Ошибка PEC	PECERR	
Ошибка Timeout/Tlow	TIMEOUT	
Предупреждение SMBus	SMBALERT	

NB:
SB, ADDR, ADD10, STOPF, BTF, RxNE и TxE логически складываются в один канал.
BERR, ARLO, AF, OVR, PECERR, TIMEOUT и SMBALERT логически складываются в другой канал.



26.5. Отладка I²C

В зависимости от битов конфигурации DBG_I2Cx_SMBUS_TIMEOUT модуля DBG таймаут SMBus может останавливаться или работать дальше.

26.6. Регистры I²C

Регистры доступны полусловами (16 бит) или словами (32 бита).

26.6.1. Регистр 1 управления I²C (I2C_CR1)

Смещение адреса: 0x00

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Res.	ALERT	PEC	POS	ACK	STOP	START	NO STRETCH	ENGC	ENPEC	ENARP	SMB TYPE	Res.	SMBUS	PE
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

— Бит 15 **SWRST**: Программный сброс

При снятии этого бита, линии и шина I²C должны быть освобождены.

0: I²C не под сбросом

1: I²C под сбросом

NB: Может использоваться для инициализации шины после ошибки или блокирования шины, например, при установке бита BUSY после глитча на шине.

— Бит 14 Резерв, не трогать.

— Бит 13 **ALERT**: Извещение SMBus

Ставится и снимается программно, аппаратно снимается при PE=0.

0: Освобождает ножку SMBA высокой. Заголовок Ответа Адреса сопровождается NACK.

1: Ставит ножку SMBA низкой. Заголовок Ответа Адреса сопровождается ACK.

— Бит 12 **PEC**: Контроль ошибки пакета

Ставится и снимается программно, аппаратно снимается после передачи PEC, сигналами Старт и Стоп и PE=0.

0: PEC не передаётся

1: PEC передаётся (в режимах Tx и Rx)

NB: Вычисление PEC разрушается потерей арбитража.

— Бит 11 **POS**: Позиция Подтверждения/PEC (для приёма данных)

Ставится и снимается программно, аппаратно снимается при PE=0.

0: Бит ACK определяет выдачу (N)ACK по приёму текущего байта в регистр сдвига. Бит PEC показывает, что в регистре сдвига лежит PEC.

1: Бит ACK определяет выдачу (N)ACK на следующий байт в регистре сдвига. Бит PEC показывает, что следующим в регистре сдвига будет PEC.

NB: Бит POS используется при приёме 2 байтов (см. Метод 2). Его надо писать до начала приёма данных. В этом случае, для выдачи NACK на второй байт, бит ACK надо снимать сразу после снятия ADDR. Для опознания второго байта как PEC, бит PEC должен стоять во время растяжки события ADDR после записи бита POS.

— Бит 10 **ACK**: Разрешение подтверждения

Ставится и снимается программно, аппаратно снимается при PE=0.

0: Подтверждение не выдаётся

1: Подтверждение выдаётся после приёма байта (совпал адрес или данные)

— Бит 9 **STOP**: Выдача сигнала Стоп

Ставится и снимается программно, аппаратно снимается при обнаружении сигнала Стоп, аппаратно ставится при обнаружении ошибки таймаута.

У ведущего:

0: Стоп не выдаётся.

1: Стоп выдаётся после передачи текущего байта или после текущей посылки Старт.

У ведомого:

0: Стоп не выдаётся.

1: Освобождаются линии SCL и SDA после передачи текущего байта.

— Бит 8 **START**: Выдача Старт

Ставится и снимается программно, аппаратно снимается при посылки Старта или PE=0.

У ведущего:

0: Старт не выдаётся

1: Повторная Выдача Старт

У ведомого:

0: Старт не выдаётся

1: Старт выдаётся при свободной шине

— Бит 7 **NOSTRETCH**: Запрет растяжки такта (Режим ведомого)

Запрещает растяжку такта ведомым при стоящем ADDR или флаге BTF до программного сброса.

0: Растягивать можно

1: Растягивать нельзя

— Бит 6 **ENGC**: Разрешение Общего Вызова

0: Запрещён. На адрес 00h выдаётся NACK.

1: Разрешён. На адрес 00h выдаётся ACK.

— Бит 5 **ENPEC:** Разрешение вычисления PEC

0: Выключено

1: Включено

— Бит 4 **ENARP**: Разрешение ARP

0: ARP выключен

1: ARP включён

Адрес устройства SMBus опознаётся при SMBTYPE=0

Адрес хоста SMBus опознаётся при SMBTYPE=1

— Бит 3 **SMBTYPE**: Тип SMBus

0: Устройство SMBus

1: Хост SMBus

— Бит 2 Резерв, не трогать.

— Бит 1 **SMBUS**: Режим SMBus

0: Режим I^2C

1: Режим SMBus

— Бит 0 **PE**: Включение

0: Выключение

1: Включение

NB: Снятие бита при работающей связи выключает устройство и сбрасывает все биты после её завершения, возврат в состояние IDLE.

У ведущего бит нельзя снимать до конца обмена.

NB: При стоящих битах STOP, START или PEC, нельзя писать в регистр I2C_CR1 вплоть до аппаратной очистки этого бита. Иначе можно получить второй запрос STOP, START или PEC.

26.6.2. Регистр 2 управления I²C (I2C_CR2)

Смещение адреса: 0x04

По сбросу: 0x0000

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]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

— Биты 15:13 Резерв, не трогать.

— Бит 12 **LAST**: Последняя передача DMA

0: Следующий EOT от DMA не последняя передача

1: Следующий EOT от DMA последняя передача

NB: Используется ведущим приёмником для разрешения выдачи NACK по последнему данному.

— Бит 11 **DMAEN**: Разрешение запросов DMA

0: Запрос DMA запрещён

1: Запрос DMA при TxE=1 или RxNE =1 разрешён

— Бит 10 **ITBUFEN**: Разрешение прерывания буфера

0: TxE = 1 или RxNE = 1 не выдают прерывания.

1: TxE = 1 или RxNE = 1 выдают прерывание Event (при любом DMAEN)

— Бит 9 **ITEVTEN**: Разрешение прерывания Event

0: Event interrupt disabled

1: Event interrupt enabled

Прерывание выдаётся при:

- SB = 1 (Ведущий)
- ADDR = 1 (Ведущий/Ведомый)
- ADD10= 1 (Ведущий)
- STOPF = 1 (Ведомый)
- BTF = 1 без события TxE или RxNE
- TxE = 1 и ITBUFEN = 1
- RxNE = 1 и ITBUFEN = 1

— Бит 8 **ITERREN**: Разрешение прерывания ошибки

0: Error interrupt disabled

1: Error interrupt enabled

Прерывание выдаётся при:

- BERR=1
- ARLO=1
- AF=1
- OVR=1
- PECERR = 1
- TIMEOUT = 1
- SMBALERT = 1

— Биты 7:6 Резерв, не трогать.

— Биты 5:0 **FREQ[5:0]**: Частота тактов периферии

В биты FREQ пишется значение частоты тактов APB (I²C подключён к APB). Поле FREQ используется для обеспечения времён установки и удержания данных в соответствии с спецификацией I²C. Минимальная частота 2 MHz, максимальная - 50 MHz (максимальная внутренняя).

0b000000: Не дозволено

0b000001: Не дозволено

0b000010: 2 MHz

…

0b110010: 50 MHz

Больше 0b100100: Не дозволено

26.6.3. Регистр 1 собственного адреса I²C (I2C_OAR1)

Смещение адреса: 0x08

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD MODE	Reserved					ADD[9:8]		ADD[7:1]							ADD0
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 15 **ADDMODE**: Режим адресации (Ведомый)
0: 7-бит ведомый
1: 10-бит ведомый

— Бит 14 Программно держится в 1

— Биты 13:10 Резерв, не трогать.

— Биты 9:8 **ADD[9:8]**: Адрес интерфейса
7-бит адрес: всё равно
10-бит адрес: биты 9:8 адреса

— Биты 7:1 **ADD[7:1]**: Биты 7:1 адреса

— Бит 0 **ADD0**: Адрес интерфейса
7-бит адрес: всё равно
10-бит адрес: бит 0 адреса

26.6.4. Регистр 2 собственного адреса I²C (I2C_OAR2)

Смещение адреса: 0x0C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ADD2[7:1]							ENDUAL
								rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:8 Резерв, не трогать.

— Биты 7:1 **ADD2[7:1]**: Биты 7:1 адреса при двойном адресе

— Бит 0 **ENDUAL**: Разрешение двух адресов
0: Только OAR1 при 7-бит адресации
1: И OAR1 и OAR2 при 7-бит адресации

26.6.5. Регистр данных I²C (I2C_DR)

Смещение адреса: 0x10

По сбросу: 0x0000

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

— Биты 15:8 Резерв, не трогать.

— Биты 7:0 **DR[7:0]**: 8-бит регистр данных
– Передача: Начинается сразу после записи байта в регистр DR. Непрерывная передача возникает при записи в DR следующего байта после начала передачи (TxE=1)
– Приём: Принятый байт копируется в DR (RxNE=1). Непрерывная передача возникает при чтении DR до завершения приёма следующего байта (RxNE=1).

NB: В режиме ведомого, адрес в DR не копируется.

Коллизия записи не разбирается (DR можно писать при TxE=0).

При событии ARLO на импульс ACK, принятый байт в DR не копируется.

26.6.6. Регистр 1 состояния I²C (I2C_SR1)

Смещение адреса: 0x14

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMB ALERT	TIME OUT	Res.	PEC ERR	OVR	AF	ARLO	BERR	TxE	RxNE	Res.	STOPF	ADD10	BTF	ADDR	SB
rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	r	r		r	r	r	r	r

— Бит 15 **SMBALERT**: Извещение SMBus

Хост SMBus:

0: Нет SMBALERT

1: На ножке есть SMBALERT

Ведомый SMBus:

0: Заголовка адреса ответа SMBALERT нет

1: Заголовок адреса ответа SMBALERT принят

Программно снимается записью 0, аппаратно снимается при PE=0.

— Бит 14 **TIMEOUT**: Ошибка Timeout или Tlow

0: Таймаута нет

1: SCL оставался низким 25 ms (Таймаут),

— Суммарное время растяжки такта ведущего больше 10 ms (Tlow:mext) или

— Суммарное время растяжки такта ведомого больше 25 ms (Tlow:sext)

– У ведомого: сбрасывается связь и аппаратно освобождаются линии

– У ведущего: аппаратно посылается Стоп

Программно снимается записью 0, аппаратно снимается при PE=0.

NB: Работает только в режиме SMBus.

— Бит 13 Резерв, не трогать.

— Бит 12 **PECERR**: Ошибка PEC на приёме

0: ошибки PEC нет: после PEC приёмник возвращает ACK (если ACK=1)

1: ошибка PEC: после PEC приёмник возвращает NACK (при любом ACK)

– Программно снимается записью 0, аппаратно снимается при PE=0.

— Бит 11 **OVR**: Переполнение/Исчерпание

0: Не было

1: Было

– Аппаратно ставится у ведомого при NOSTRETCH=1 и:

– Принят новый байт (включая импульс ACK), а регистр DR ещё не прочитан. Байт теряется.

– Надо посылать новый байт, а регистр DR ещё не записан. Посылается старый.

Программно снимается записью 0, аппаратно снимается при PE=0.

NB: Если DR пишется слишком близко к переднему фронту SCL, то посылаемое данное не определено и возникает ошибка времени удержания.

— Бит 10 **AF**: Сбой подтверждения

0: Не было

1: Было

– Аппаратно ставится при отсутствии подтверждения.

– Программно снимается записью 0, аппаратно снимается при PE=0.

— Бит 9 **ARLO**: Потеря арбитража (Ведущий)

0: Не было

1: Было

Ставится аппаратно при передаче арбитража другому ведущему

– Программно снимается записью 0, аппаратно снимается при PE=0.

После события ARLO I²C автоматически возвращается в ведомые (MSL=0).

NB: В SMBUS арбитраж данных ведомого появляется только в фазе данных и подтверждении передачи.

— Бит 8 **BERR**: Ошибка шины

0: Старт и Стоп были так, где надо

1: Старт и Стоп появились не там

– Аппаратно ставится при появлении фронта SDA при высоком SCL во время передачи байта.

– Программно снимается записью 0, аппаратно снимается при PE=0.

— Бит 7 **TxE**: Регистр данных пуст (Передача)

0: Регистр данных не пуст

1: Регистр данных пуст

– Ставится при опустении DR для передачи. В фазе адреса TxE не ставится.

– Снимается программно записью в DR или аппаратно сигналами Старт и Стоп, или при PE=0.

TxE не ставится при получении NACK и перед передачей PEC (PEC=1)

NB: TxE не снимается записью первого передаваемого байта или записью при стоящем BTF, так как регистр данных всё равно ещё пуст.

— Бит 6 **RxNE**: Регистр данных не пуст (Приём)

0: Регистр данных пуст

1: Регистр данных не пуст

– Ставится при появлении принятого данного. В фазе адреса RxNE не ставится.

– Снимается программно чтением или записью DR или аппаратно при PE=0.

RxNE не ставится при событии ARLO.

NB: RxNE не снимается чтением при стоящем BTF, так как регистр ещё полон.

— Бит 5 Резерв, не трогать.

— Бит 4 **STOPF**: Обнаружен Стоп (Ведомый)

0: Стопа нету

1: Появился Стоп

– Аппаратно ставится на при обнаружении Стоп на шине ведомого после подтверждения (ACK=1).

– Программно снимается чтением SR1 с последующей записью в CR1, аппаратно при PE=0

NB: Бит STOPF не ставится после приёма NACK.

— Бит 3 **ADD10**: 10-бит заголовок послан (Ведущий)

0: ADD10 не было.

1: Ведущий послал первый байт адреса (заголовок).

– Ставится аппаратно при посылке первого байта 10-бит адреса.

– Программно снимается чтением SR1 с последующей записью в DR второго байта адреса, аппаратно при PE=0.

NB: Бит ADD10 не ставится после приёма NACK.

— Бит 2 **BTF**: Передача байта завершена

0: Передача байта не завершена

1: Передача байта завершена

– Ставится аппаратно при NOSTRETCH=0 и:

– Приём: получен новый байт (включая импульс ACK), а DR ещё не читан (RxNE=1).

– Передача: пора посылать новый байт, а DR ещё не писан (TxE=1).

– Снимается программно чтением SR1 с последующим чтением или записью DR или аппаратно после Старт или Стоп или при стоящем PE=0.

NB: Бит BTF не ставится после приёма NACK.

Бит BTF не ставится перед передачей PEC (TRA=1 в регистре I2C_SR2 и PEC=1 в регистре I2C_CR1)

— Бит 1 **ADDR**: Адрес послан (Ведущий) / Совпал (Ведомый)

Программно снимается чтением SR1 с последующим чтением SR2, аппаратно при PE=0.

Адрес совпал (Ведомый)

0: Адрес не совпал или не принят.

1: Принятый адрес совпал.

– Аппаратно ставится при совпадении полученного адреса ведомого с регистрами OAR или Общим Вызовом, либо с адресом по умолчанию устройства SMBus или опознаны SMBus Host или SMBus Alert.

Адрес послан (Ведущий)

0: Передача адреса не завершена

1: Передача адреса завершена

– При 10-бит адресации ставится после ACK второго байта.

– При 7-бит адресации ставится после ACK байта.

NB: ADDR is not set after a NACK reception

— Бит 0 **SB**: Бит Старт (Ведущий)

Ставится при выдаче Старт, снимается чтением SR1 с записью DR, аппаратно при PE=0.

0: Старту нету

1: Выдача Старта.

26.6.7. Регистр 2 состояния I²C (I2C_SR2)

Смещение адреса: 0x18

По сбросу: 0x0000

NB: Чтение I2C_SR2 после чтения I2C_SR1 снимает флаг ADDR, даже если ADDR встал после чтения I2C_SR1. Следовательно, I2C_SR2 нужно читать только после проверки ADDR в I2C_SR1 или бит STOPF снят.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7:0]								DUALF	SMB HOST	SMBDE FAULT	GEN CALL	Res.	TRA	BUSY	MSL
r	r	r	r	r	r	r	r	r	r	r	r		r	r	r

— Биты 15:8 **PEC[7:0]** Регистр проверки PEC
Содержит внутренний PEC при ENPEC=1.

— Бит 7 **DUALF**: Флаг двойного адреса (Ведомый)
0: Принятый адрес совпал с OAR1
1: Принятый адрес совпал с OAR2
– Снимается аппаратно при Стоп или повторного Старт, или при PE=0.

— Бит 6 **SMBHOST**: Заголовок хоста SMBus (Ведомый)
0: Не было
1: Принят заголовок хоста SMBus при SMBTYPE=1 и ENARP=1.
– Снимается аппаратно при Стоп или повторного Старт, или при PE=0.

— Бит 5 **SMBDEFAULT**: Адрес устройства по умолчанию SMBus (Ведомый)
0: Не было
1: Принят при ENARP=1
– Снимается аппаратно при Стоп или повторного Старт, или при PE=0.

— Бит 4 **GENCALL**: Адрес Общего Вызова (Ведомый)
0: Не было
1: Принят при ENGC=1
– Снимается аппаратно при Стоп или повторном Старт, или при PE=0.

— Бит 3 Резерв, не трогать.

— Бит 2 **TRA**: Передатчик/Приёмник
0: Байт данных принят
1: Байт данных передан
Ставится в зависимости от бита R/W байта адреса в конце фаза адреса.
Снимается аппаратно при Стоп (STOPF=1), повторном Старт, потере арбитража (ARLO=1), при PE=0.

— Бит 1 **BUSY**: Шина занята
0: Свободен
1: Работаем
– Ставится аппаратно при низких SDA или SCL
– Снимается аппаратно при Стоп.
Обновляется даже при PE=0.

— Бит 0 **MSL**: Ведущий/Ведомый
0: Ведомый
1: Ведущий
Ставится аппаратно при SB=1.
Снимается аппаратно при Стоп, потере арбитража (ARLO=1), при PE=0.

26.6.8. Регистр управления тактами I²C (I2C_CCR)

Смещение адреса: 0x1C

По сбросу: 0x0000

NB: f_{PCLK1} должна быть не меньше 2 MHz в режиме Sm, и не меньше 4 MHz в режиме Fm. Шаг равен 10MHz для получения максимальных 400 kHz режима Fm.

Писать надо при выключенном I²C (PE = 0).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	Reserved		CCR[11:0]											
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 15 **F/S:** Режим ведущего I²C
0: режим Sm
1: режим Fm

— Бит 14 **DUTY:** Коэффициент заполнения режима Fm
0: $t_{low}/t_{high} = 2$
1: $t_{low}/t_{high} = 16/9$ (см. CCR)

— Биты 13:12 Резерв, не трогать.

— Биты 11:0 **CCR[11:0]:** Регистр управления тактами режимов Fm/Sm (Ведущий)
Задаёт такты SCL ведущего.
Режим Sm или SMBus:
$T_{high} = CCR * T_{PCLK1}$
$T_{low} = CCR * T_{PCLK1}$
Режим Fm:
При DUTY = 0:
$T_{high} = CCR * T_{PCLK1}$
$T_{low} = 2 * CCR * T_{PCLK1}$
При DUTY = 1: (для достижения 400 kHz)
$T_{high} = 9 * CCR * T_{PCLK1}$
$T_{low} = 16 * CCR * T_{PCLK1}$
Например: в режиме Sm выдаём 100 kHz SCL:
Если FREQR = 08, T_{PCLK1} = 125 ns, то в CCR надо писать 0x28 (0x28 <=> 40d x 125 ns = 5000 ns.)
NB:
Минимальное значение равно 0x04, в режиме FAST DUTY минимум равен 0x01
$t_{high} = t_{r(SCL)} + t_{w(SCLH)}$. См. описание устройства.
$t_{low} = t_{f(SCL)} + t_{w(SCLL)}$. См. описание устройства.
Скорость I²C равна $f_{SCL} \sim 1/(t_{high} + t_{low})$. Реальная частота может отличаться из-за задержки аналогового фильтра на входе.

26.6.9. Регистр TRISE I²C (I2C_TRISE)

Смещение адреса: 0x20

По сбросу: 0x0002

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

— Биты 13:6 Резерв, не трогать.

— Биты 5:0 **TRISE[5:0]**: Максимальное время подъёма в режиме Fm/Sm (Ведущий)
Это максимальное время обратной связи SCL в режиме ведущего для стабилизации частоты SCL. Должно быть взято из спецификации шины I²C плюс 1.
Например: в режиме Sm, максимальное время подъёма SCL равно 1000 ns.
Если значение битов FREQ[5:0] в регистре I2C_CR2 равно 0x08 и T_{PCLK1} = 125 ns, то TRISE[5:0] должно быть равно 09h.
(1000 ns / 125 ns = 8 + 1)
В TRISE[5:0] можно добавить значение фильтра.
Если результат не целочисленный, то в TRISE[5:0] надо писать целую часть для соблюдения параметра t_{HIGH}.
NB: TRISE[5:0] можно писать только при PE = 0.

26.6.10.Карта регистров I²C

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**I2C_CR1**	Reserved																SWRST	Reserved	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	Reserved	SMBUS	PE
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0		0	0
0x04	**I2C_CR2**	Reserved																			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Reserved	FREQ[5:0]						
	Reset value																				0	0	0	0	0		0	0	0	0	0	0	0
0x08	**I2C_OAR1**	Reserved																ADDMODE	Reserved					ADD[9:8]		ADD[7:1]							ADD0
	Reset value																	0						0	0	0	0	0	0	0	0	0	0
0x0C	**I2C_OAR2**	Reserved																								ADD2[7:1]							ENDUAL
	Reset value																									0	0	0	0	0	0	0	0
0x10	**I2C_DR**	Reserved																								DR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	**I2C_SR1**	Reserved																SMBALERT	TIMEOUT	Reserved	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Reserved	STOPF	ADD10	BTF	ADDR	SB
	Reset value																	0	0		0	0	0	0	0	0	0		0	0	0	0	0
0x18	**I2C_SR2**	Reserved																PEC[7:0]								DUALF	SMBHOST	SMBDEFAULT	GENCALL	Reserved	TRA	BUSY	MSL
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x1C	**I2C_CCR**	Reserved																F/S	DUTY	Reserved	CCR[11:0]												
	Reset value																	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**I2C_TRISE**	Reserved																										TRISE[5:0]					
	Reset value																											0	0	0	0	1	0

27. Интерфейс USART

27.1. Введение в USART

Универсальный синхронно-асинхронный приёмо-передатчик (USART) предлагает гибкие способы полно-дуплексного обмена с внешним оборудованием по стандарту NRZ в широком диапазоне скоростей с дробной частотой.

Поддерживает синхронную одностороннюю связь и полудуплекс по одному проводу. Также поддерживается LIN (коммутируемая локальная сеть), протокол Smartcard, спецификации IrDA SIR ENDEC и работа модема (CTS/RTS). Дозволена многопроцессорная связь.

Высокоскоростная связь может использовать многобуферный DMA.

27.2. Основные свойства USART

- Полный дуплекс, асинхронная связь
- Формат стандарта NRZ (Mark/Space)
- Генератор дробной скорости
  - —Общая программируемая скорость приёма и передачи до 4.5 Мбит/с

- Программируемая длина слова (8 или 9 бит)
- Программируемые биты стоп (1 или 2)
- Посылка синхронного LIN Break ведущего и приём LIN Break ведомого
  — генерация 13-бит Break и обнаружение 10/11 бит Break при USART в режиме LIN
- Вывод тактов передатчика для синхронных операций
- IrDA SIR Кодер/Декодер
  — Поддержка 3/16 бит длительности нормального режима
- Эмуляция Smartcard
  — Интерфейс Smartcard поддерживает асинхронный стандарт Smartcards ISO 7816-3
  — 0.5, 1.5 Стоп Биты для операций Smartcard operation
- Однопроводная полудуплексная связь
- DMA с несколькими буферами
  — Буферы байтов приёма/передачи в резервной SRAM с центральным DMA
- Раздельные биты включения приёма и передачи
- Флаги передачи:
  — Буфер приёма полон
  — Буфер передачи пуст
  — Флаги Конца Передачи
- Контроль чётности:
  — Бит чётности передачи
  — Проверка чётности принятого байта данных
- Четыре флага ошибок:
  — Переполнение
  — Ошибка шума
  — Ошибка фрейма
  — Ошибка чётности
- Десять источников прерываний с флагами:
  — CTS изменился
  — Обнаружен LIN break
  — Регистр данных передачи пуст
  — Передача завершена
  — Регистр данных приёма полон
  — Принят Простой линии
  — Переполнение
  — Ошибка фрейма
  — Ошибка шума
  — Ошибка чётности
- Многопроцессорная связь - уход в немоту при несовпадении адреса
- Пробуждение из немоты (по простою линии или метке адреса)
- Два режима пробуждения приёмника: Бит адреса (девятый бит), Простой линии

27.3. Функциональное описание USART

Наружу торчат три ножки, но для двунаправленной последовательной связи хватает двух: Приём данных (RX) и Передача данных (TX):

RX: Для выделения сигнала из шума используется дискретизация с повышенной частотой.

TX: При выключенном передатчике эта нога возвращается порту IO. Если включённому передатчику говорить нечего, то нога TX стоит высокой. В однопроводных режимах она используется для приёма и передачи (на уровне USART данные затем принимаются на `SW_RX`).

Данные оформляются в фреймы:

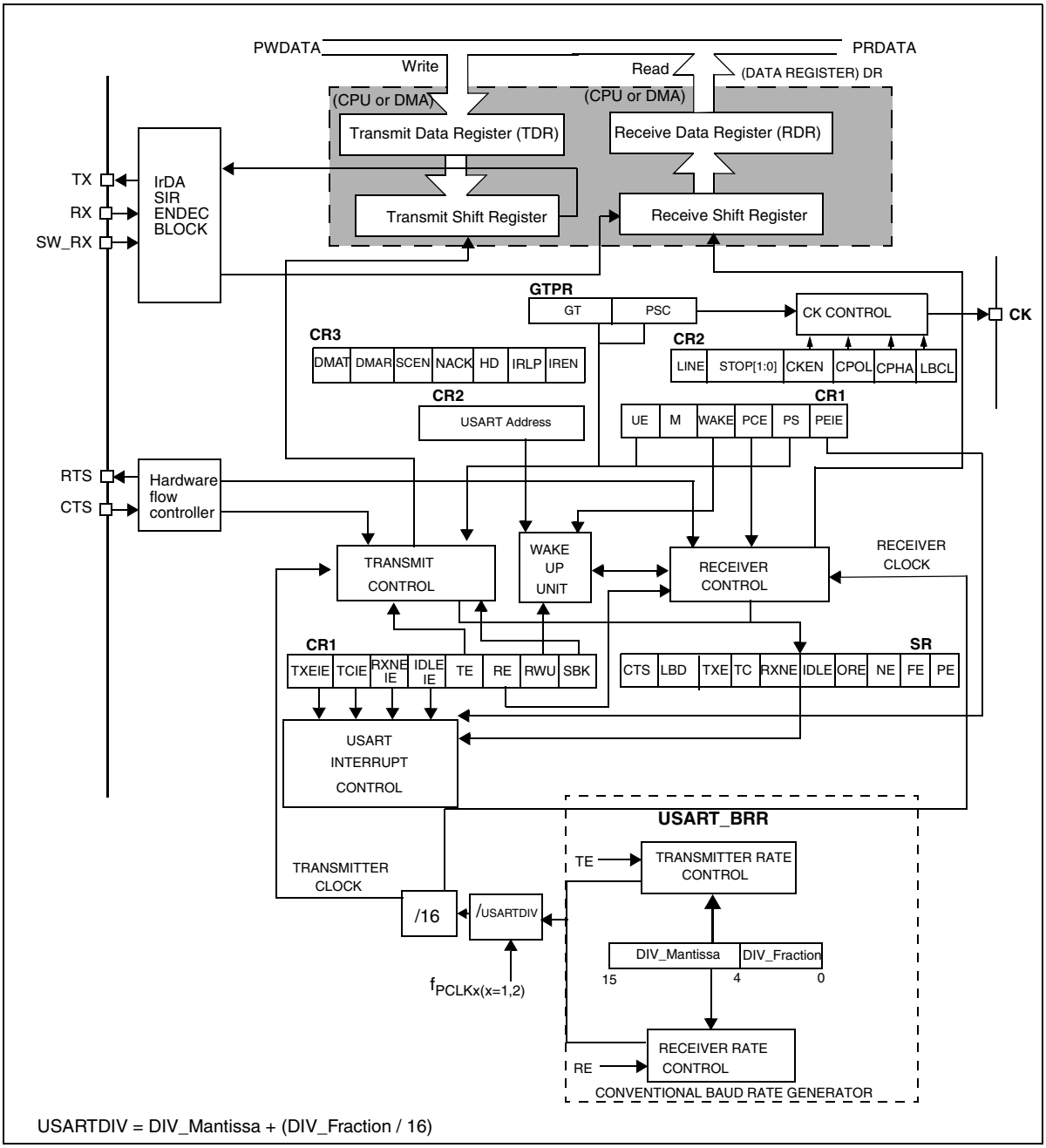
- Простой линии перед приёмом и передачей
- Стартовый бит
- Слово данных (8 или 9 бит), младшим вперёд
- 0.5,1, 1.5, 2 стоповых бита конца фрейма
- Используется дробный генератор частоты - 12-бит мантисса и 4-бит дробь
- Регистр состояния (USART_SR)
- Регистр данных (USART_DR)
- Регистр частоты (USART_BRR) - 12-бит мантисса и 4-бит дробь.
- Регистр защиты (Guardtime Register) (USART_GTPR) в режиме Smartcard.

В синхронном режиме работают ноги:

- **CK:** Такты передатчика. Соответствует режиму ведущего SPI (биты старт и стоп без импульса, необязательный импульс на последний бит данных). Параллельно можно синхронно принимать данные на RX. Фаза и полярность тактов программируются. Можно использовать в режиме Smartcard.

При аппаратном управлении используются ноги:

- **CTS:** Высоким блокирует передачу данных в конце текущей передачи
- **RTS:** Низким запрашивает передачу данных.



27.3.1. Описание символа USART

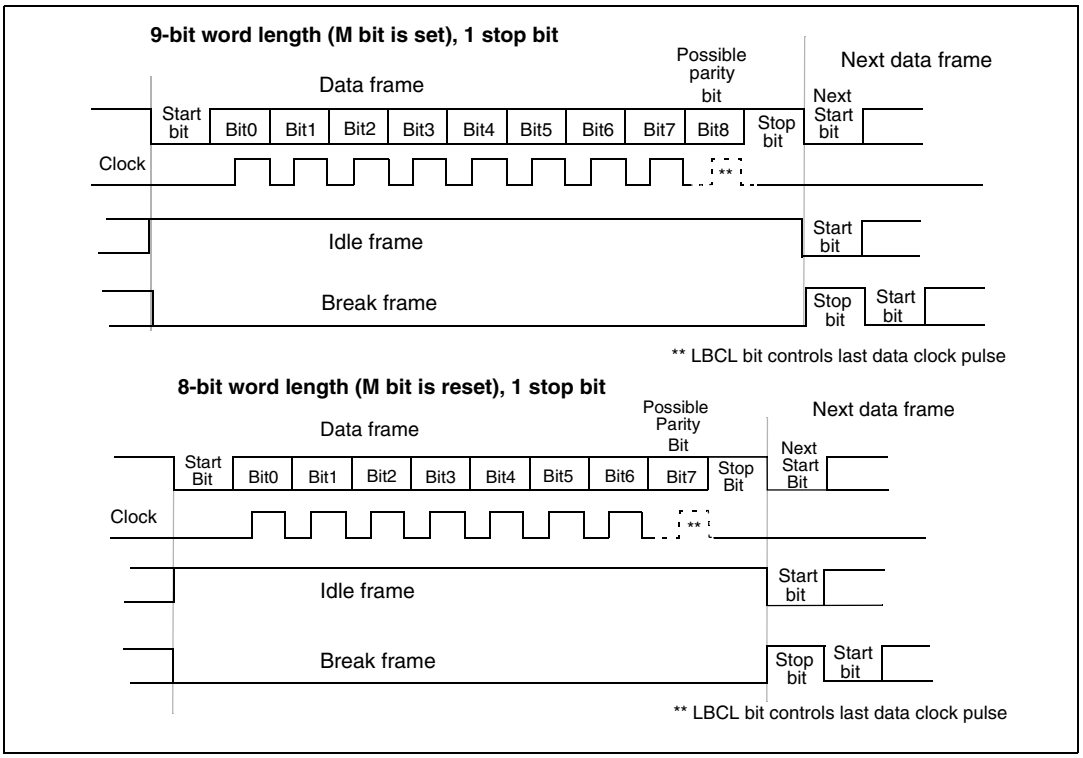
Длина слова (8 или 9 бит) выбирается битом M регистра USART_CR1. Ножка TX во время бита старта низкая, а при бите стопа - высокая.

Символ простоя это целый фрейм единиц с последующим битом старта следующего фрейма с данными (число единиц включает биты стопа).

Символ разрыва это нули во время всего периода фрейма. В конце фрейма разрыва выдаются нормальные биты стоп (логическая 1) чтобы узнать бит старта.

Такты от общего генератора подаются приёмнику и передатчику при их включении нужным битом.

Задание длины слова



27.3.2. Передатчик

Передача включается установкой бита TE, начиная выдачу битов из регистра сдвига, начиная с младшего, на ножку TX и тактовых импульсов на ножку CK.

Передача символа

В регистр сдвига данные поступают из регистра данных USART_DR (TDR), а туда с шины. Каждому символу предшествует бит старта низкого уровня. Завершается символ 0.5, 1, 1.5 или 2 битами стоп.

NB: Бит TE нельзя снимать во время передачи данных.

После установки бита TE посылается фрейм простоя.

Биты Стоп

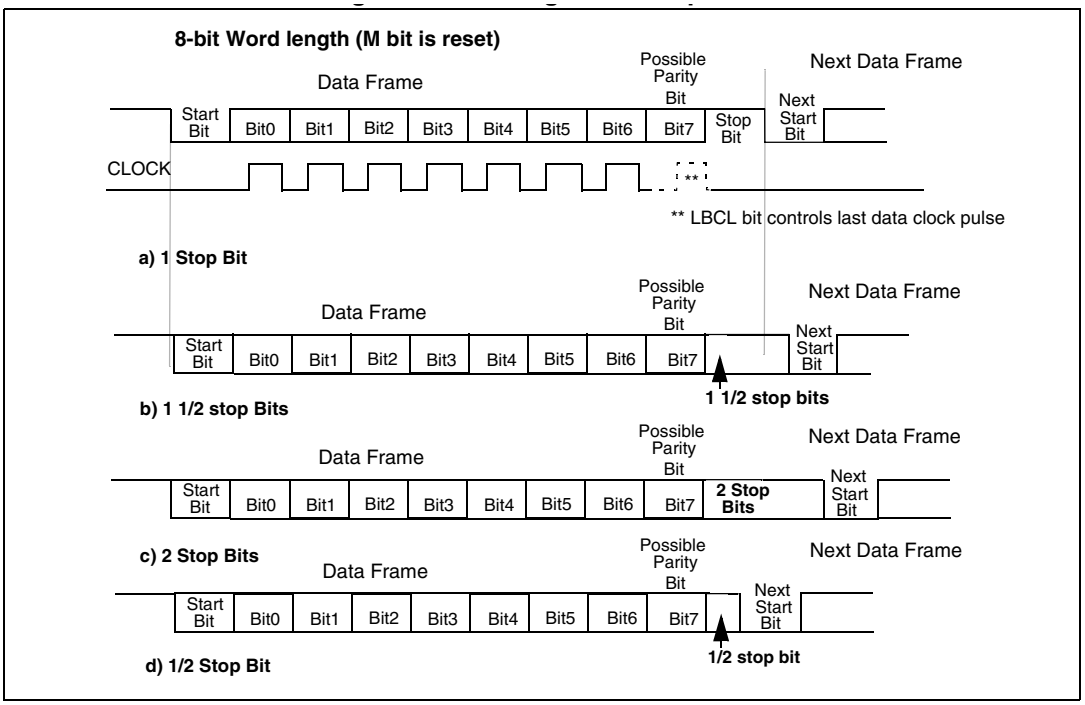
Число битов стоп задаются битами 13,12 регистра управления 2.

1. **1 stop bit:** Значение по умолчанию.
2. **2 stop bits:** Режим одного провода и режим модема USART.
3. **0.5 stop bit:** Приём данных в режиме Smartcard.
4. **1.5 stop bits:** Приём и передача данных в режиме Smartcard.

Фрейм простоя включает биты стоп.

Передача Разрыва это 10 низких битов с последующими битами стоп (при M = 0) или 11 низких битов с последующими битами стоп (when M = 1). Более длинные Разрывы передавать нельзя.

Биты Стоп



Процедура:

1. Включаем USART установкой бита `UE` в регистре `USART_CR1`.
2. Задаём длину слова битом `M` в регистре `USART_CR1`.
3. Пишем число битов стопа в регистре `USART_CR2`.
4. Выбираем DMA (`DMAT`) в регистре `USART_CR3` при многих буферах надобных. Заполняем регистры DMA.
5. Выбираем скорость регистром `USART_BRR`.
6. Ставим бит `TE` в регистре `USART_CR1` ради посылки фрейма простоя.
7. Пишем посылаемые данные в регистр `USART_DR` (бит `TXE` снимается). Повторяем для каждого посылаемого данного при использовании одного буфера.
8. После записи в `USART_DR` последнего данного, ждём сигнала конца передачи фрейма установленным битом `TC`=1.

Передача байта

Бит `TXE` снимается записью в регистр данных, ставится он аппаратно, указывая:

- Данные переданы из `TDR` в регистр сдвига и передача началась.
- Регистр `TDR` пуст.
- Надо писать в `USART_DR` следующее данное.

При стоящем бите `TXEIE` выдаётся прерывание.

Запись в `USART_DR` во время передачи пишет в регистр `TDR`.

Запись в `USART_DR` не во время передачи пишет сразу в регистр сдвига, начинает передачу и ставит бит `TXE`.

После передачи битов стопа всего фрейма и установки бита `TXE` ставится бит `TC`. При стоящем бите `TCIE` регистра `USART_CR1` выдаётся прерывание.

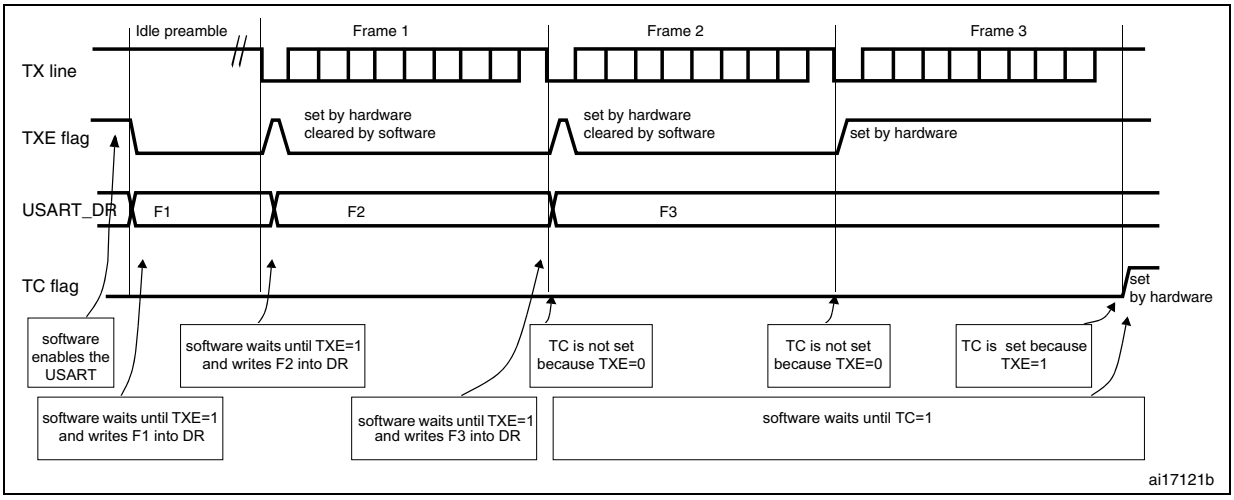
Перед выключением USART с возможным Сном MCU, после записи последнего данного в регистр `USART_DR`, надо обязательно дождаться `TC`=1.

Программно бит `TC` чистится так:

1. Читаем регистр `USART_SR`
2. Пишем в регистр `USART_DR`

Бит `TC` можно снять записью нуля, но это рекомендуется только в режиме многих буферов.

TC/TXE при передаче



Символ разрыва

Установка бита SBK передаёт символ разрыва. Длина фрейма разрыва зависит от бита M.

Стоящий бит SBK после завершения передачи текущего символа посылает на линию TX символ разрыва. Снимается SBK аппаратно во время передачи битов стоп символа разрыва. USART вставляет логическую 1 в конец фрейма разрыва, чтобы узнавать бит старта следующего фрейма.

Снятие бита SBK до начала передачи разрыва отменяет саму передачу. Для передачи двух последовательных разрывов бит SBK надо ставить после битов стопа предыдущего разрыва.

Символ простоя

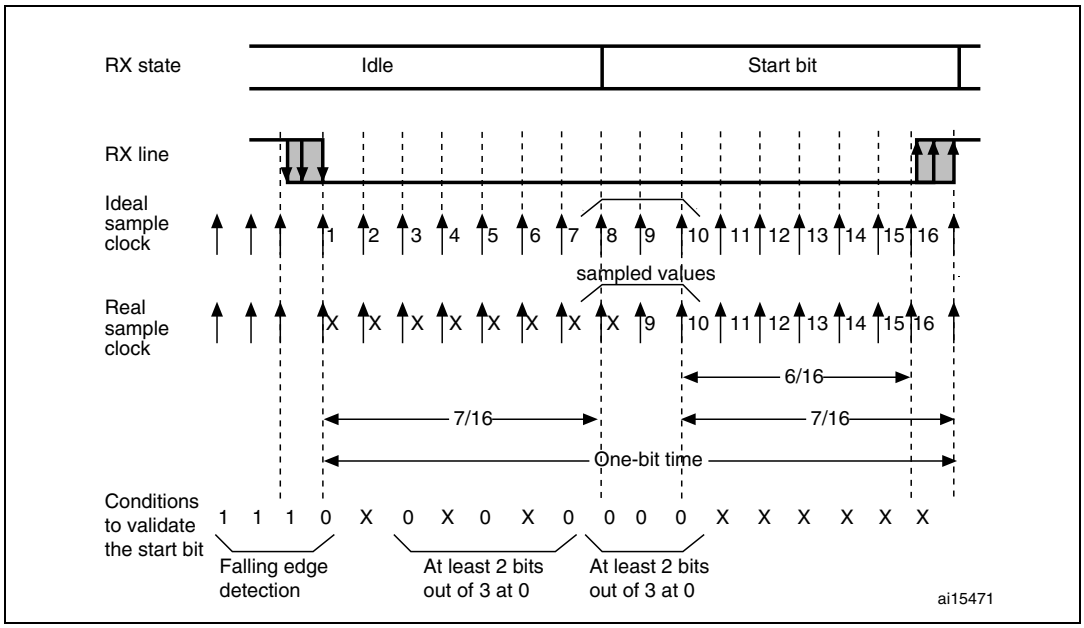
Установка бита TE понуждает USART послать фрейм простоя перед первым фреймом данных.

27.3.3. Приёмник

Длина принимаемых слов (8 или 9 битов) определяется битом M в регистре USART_CR1.

Обнаружение бита старта

USART узнаёт бит старта по последовательности битов 1 1 1 0 X 0 X 0 X 0 0 0 0.



NB: Если последовательность не завершена, то приёмник перестаёт искать бит и идёт в простой, где и ждёт заднего фронта (флаг не ставится).

Старт подтверждается (ставится бит RXNE, при RXNEIE=1 выдаётся прерывание) при 3 выборках нуля (первый ноль на 3м, 5м и 7м импульсах и второй ноль на 8м, 9м и 10м импульсах выборки).

Старт принимается (ставится бит RXNE, при RXNEIE=1 выдаётся прерывание), но ставится флаг шума NE, если для обеих выборок (на 3м, 5м и 7м импульсах и на 8м, 9м и 10м импульсах выборки) не меньше 2 из 3 раз читается 0. Если не так, то приёмник перестаёт искать бит и идёт в простой, где и ждёт заднего фронта (флаг не ставится).

Приём символа

Во время приёма вдвигаются, начиная с младшего бита, с ножки RX в регистр сдвига. Буфер USART_DR (RDR) получает данное из него.

Процедура:

1. Включаем USART установкой бита UE в регистре USART_CR1.
2. Задаём длину слова битом M в регистре USART_CR1.
3. Пишем число битов стопа в регистре USART_CR2.
4. Выбираем DMA (DMAR) в регистре USART_CR3 при многих буферах надобных. Заполняем регистры DMA.
5. Выбираем скорость регистром USART_BRR.
6. Ставим бит RE в регистре USART_CR1. Приёмник начинает искать бит старта.

Когда символ принят:

- Ставится бит RXNE. То есть, регистр сдвига переписан в RDR и его можно читать.
- При стоящем бите RXNEIE выдаётся прерывание.
- Могут встать флаги ошибки фрейма, шума и переполнения.
- При многих буферах бит RXNE ставится после каждого байта и снимается DMA.
- При одном буфере бит RXNE снимается программно чтением регистра USART_DR. Он может сниматься записью ноля. Во избежание переполнения чистить RXNE нужно до конца приёма следующего байта.

NB: Бит RE нельзя снимать во время приёма.

Символ разрыва

Обрабатывается как ошибка фрейма.

Символ простоя

Принимается как обычное данное, при стоящем бите IDLEIE выдаётся прерывание.

Переполнение

Возникает после приёма символа при стоящем RXNE. Данные в регистр RDR не переносятся.

Флаг RXNE ставится после приёма каждого байта. При появлении прерывания:

- Ставится бит ORE.
- Содержимое RDR не теряется.
- Следующим байтом перепишется регистр сдвига.
- Прерывание выдаётся при стоящем бите RXNEIE или обоих стоящих битах EIE DMAR.
- Бит ORE снимается чтением регистра USART_SR с последующим чтением USART_DR.

NB: Стоящий бит ORE означает потерю не менее 1 данного. Есть два выхода:

- При RXNE=1, можно прочитать из RDR последний достоверный байт,
- При RXNE=0, из RDR уже всё прочитали, но одновременно с приёмом нового (потерянного) байта или между чтением USART_SR и чтением USART_DR.

Ошибка шума

Для обнаружения используется способ высокочастотной выборки (исключая синхронный режим).

Дискретизация данных при обнаружении шума

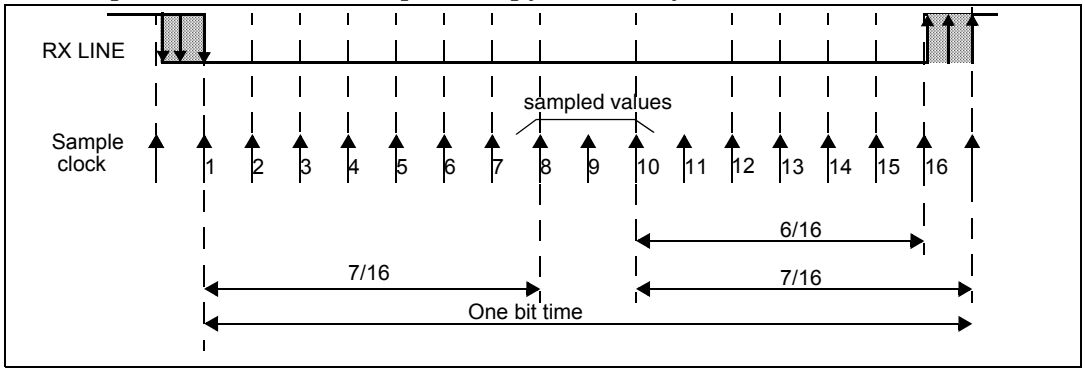


Таблица 191. Выделение шума

Считанное значение	Состояние NE	Принятый бит	Достоверность
000	0	0	Да
001	1	0	Нет
010	1	0	Нет
011	1	1	Нет
100	1	0	Нет
101	1	1	Нет
110	1	1	Нет
111	0	1	Да

При обнаружении шума в фрейме:

- По переднему фронту бита RXNE ставится бит NE.
- В регистр USART_DR пишутся непотребные данные.
- При однобайтовой передаче прерывание ошибки не выдаётся, но бит RXNE ставится. При многобуферной передаче прерывание ошибки разрешается битом EIE в регистре USART_CR3.

Бит NE снимается чтением регистра USART_SR с последующим чтением регистра USART_DR.

Ошибка фрейма

Возникает когда в нужное время не обнаруживаются биты стоп с последующей потерей синхронизации или при ужасном шуме.

Если появилась:

- Аппаратно ставится бит FE
- В регистр USART_DR пишется лажа.
- При однобайтовой передаче прерывание ошибки не выдаётся, но бит RXNE ставится. При многобуферной передаче прерывание ошибки разрешается битом EIE в регистре USART_CR3.

Бит FE снимается чтением регистра USART_SR с последующим чтением регистра USART_DR.

Биты стоп при приёме

В нормальном режиме используются 1 или 2 бита, в режиме Smartcard 0.5 или 1.5 бита.

— **0.5 бита стоп (приём в Smartcard)**: Выборки 0.5 бита стопа нет, так что ошибка фрейма и разрыв не обнаруживаются.

— **1 бит стоп**: Чтение 1 по 8, 9 и 10 импульсам выборки.

— **1.5 бита стоп (Smartcard)**: При передаче в режиме Smartcard устройство должно проверять правильность отсылаемых данных. Так что, блок приёма должен быть включён (RE =1 в регистре USART_CR1) и бит стоп проверяется на предмет обнаружения устройством ошибки чётности. В случае ошибки устройство удерживает сигнал данных низким (NACK), сообщая об ошибке фрейма. Флаг FE ставится вместе с RXNE в конце 1.5 битов стоп. Определение 1.5 битов стоп выполняется по 16, 17 и 18 импульсам выборки (1 такт связи после начала бита стоп).
1.5 бита можно разделить на 2 части: 0.5 такта без событий и 1 нормальный такт с выборкой бита стоп в его середине. См. *Секцию 27.3.11*.

— **2 бита стоп**: Считывание битов стоп выполняется по 8, 9 и 10 импульсам выборки. Ошибка фрейма определяется при первом бите стоп, при втором ошибка не проверяется. Флаг RXNE ставится по концу первого бита стоп.

27.3.4. Генерация дробной частоты

Частота приёмника и передатчика (Rx и Tx) задаётся мантиссой и дробной частью числа в регистре USARTDIV.

$$\text{Tx/ Rx baud} = \frac{f_{CK}}{(16*\text{USARTDIV})}$$

legend: f_{CK} - Input clock to the peripheral (PCLK1 for USART2, 3, 4, 5 or PCLK2 for USART1)

USARTDIV это беззнаковое число с фиксированной запятой, занесённое в регистр USART_BRR.

NB: Счётчик тактов связи обновляется из регистра USART_BRR. То есть, во время работающей связи этот регистр менять нельзя.

Как получить USARTDIV из значения регистра USART_BRR

Пример 1:

Если DIV_Mantissa = 0d27 и DIV_Fraction = 0d12 (USART_BRR = 0x1BC), то мантисса (USARTDIV) = 0d27

Дробная часть (USARTDIV) = 12/16 = 0d0.75

Так что USARTDIV = 0d27.75

Пример 2:

Пишем USARTDIV = 0d25.62

Далее:

DIV_Fraction = 16*0d0.62 = 0d9.92

Ближайшее целое 0d10 = 0xA

DIV_Mantissa = мантисса (0d25.620) = 0d25 = 0x19

Тогда, USART_BRR = 0x19A отсюда USARTDIV = 0d25.625

Пример 3:

Пишем USARTDIV = 0d50.99

Далее:

DIV_Fraction = 16*0d0.99 = 0d15.84

Ближайшее целое 0d16 = 0x10 => переполнение DIV_frac[3:0] => перенос добавляем к мантиссе DIV_Mantissa = мантисса (0d50.990 + перенос) = 0d51 = 0x33 То, USART_BRR = 0x330 отсюда USARTDIV = 0d51.000

Таблица 192. Вычисление ошибки

Скорость		f_{PCLK} = 36 MHz			f_{PCLK} = 72 MHz		
S.No	Кбит/с	Реальное	Значение в регистре скорости	% ошибки(1)	Реальное	Значение в регистре скорости	% ошибки(1)
1.	2.4	2400	937.5	0 %	2.4	1875	0 %
2.	9.6	9600	234375	0 %	9.6	468.75	0 %
3.	19.2	19.2	117.1875	0 %	19.2	234375	0 %
4.	57.6	57.6	39.0625	0 %	57.6	78125	0 %
5.	115.2	115384	19.5	0 %	115.2	39.0625	0 %
6.	230.4	230769	9.75	0 %	230769	19.5	0 %
7.	460.8	461538	4875	0 %	461538	9.75	0 %
8.	921.6	923076	2.4375	0 %	923076	4875	0 %
9.	2250	2250	1	0 %	2250	2	0 %
10.	4500	NA	NA	NA	4500	1	0 %

1. Вычисляется как (Вычисленная скорость - Желаемая скорость) / Желаемая скорость.

NB: Чем ниже частота тактов CPU, тем ниже точность скорости. С помощью этих данных можно определить верхний достижимый предел скорости.

USART1 тактируется PCLK2 (72 MHz макс.). Другие USART тактируются PCLK1 (36 MHz макс.).

27.3.5. Устойчивость приёмника к колебаниям частоты

Асинхронный приёмник USART может нормально работать только при общем отклонении тактов системы меньшем устойчивости приёмника USART. Слагаемые общего отклонения:

- DTRA: Ошибка передатчика (включает ошибку генератора передатчика)
- DQUANT: Ошибка частоты дискретизации приёмника
- DREC: Отклонение генератора приёмника
- DTCL: Отклонение линии передачи (обычно из-за асимметрии времён переднего и заднего фронтов сигнала передатчика)

DTRA + DQUANT + DREC + DTCL < устойчивости приёмника USART

Устойчивость приёмника USART зависит от:

- 10- или 11-бит длины символа, определённой битом M регистра USART_CR1
- использования дробной частоты связи

Таблица 193. Устойчивость приёмника USART при DIV_Fraction is 0

Бит M	NF это ошибка	NF не волнует
0	4 %	4375 %
1	3 %	4 %

Таблица 194. Устойчивость приёмника USART при DIV_Fraction отличном от 0

Бит M	NF это ошибка	NF не волнует
0	3 %	4 %
1	3 %	4 %

27.3.6. Многопроцессорная связь

Многопроцессорную сеть можно создать объединяя несколько USART на одной линии. Выход TX одного (ведущего) подключён к входам RX других USART (ведомых). Их выходы TX проводным логическим И объединены на входе RX ведущего.

При этом излишней нагрузки на линию можно избежать передавая сообщения только нужным адресованным устройствам, оставшиеся переключаются в немой режим. В этом режиме:

- Биты приёма не встают.
- Приёмные прерывания запрещены.
- Бит RWU в USART_CR1 стоит. RWU управляется программно и аппаратно.

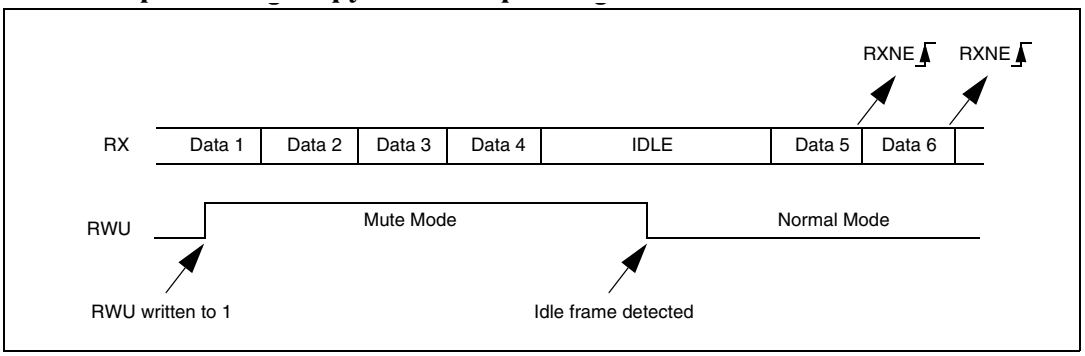
Методы входа и выхода USART из немого режима зависят от бита WAKE регистра USART_CR1:

- Обнаружение Простоя линии при сброшенном бите WAKE,
- Обнаружение Метки адреса при стоящем бите WAKE.

Обнаружение простоя линии (WAKE=0)

USART просыпается при появлении фрейма простоя. Бит RWU снимается аппаратно, но бит IDLE регистра USART_SR не ставится. RWU можно снимать программно.

Немой режим с обнаружением простоя линии



Обнаружение метки адреса (WAKE=1)

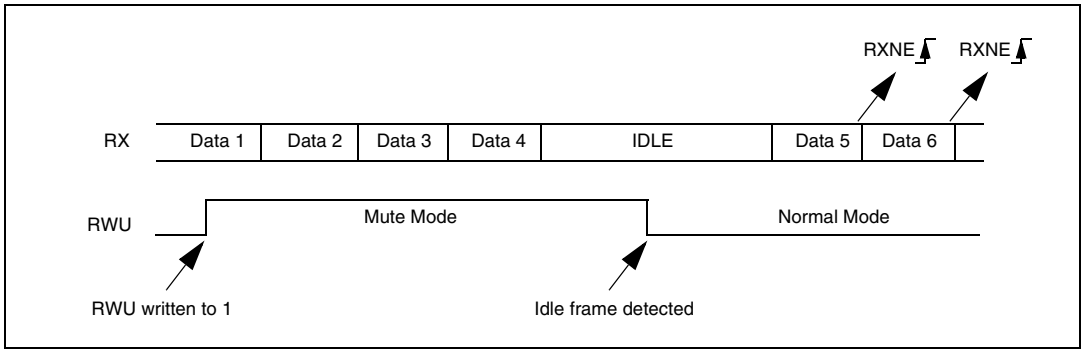
В этом режиме байт адреса распознаётся по стоящему старшему биту, младшие 4 бита это адрес устройства. Он аппаратно сравнивается с битами ADD регистра USART_CR2.

USART уходит в немой режим при несовпадении этих адресов, бит RWU ставится аппаратно, флаг RXNE не ставится, прерывание и запрос DMA не выдаются.

При совпадении адресов снимается бит RWU и дальше всё идёт нормально. Для символа адреса бит RXNE ставится, так как бит RWU уже очищен.

RWU можно писать только при пустом буфере (RXNE=0 в регистре USART_SR), иначе не пишется.

Немой режим с обнаружением метки адреса



27.3.7. Контроль чётности

Включается установкой бита PCE регистра USART_CR1.

Форматы фреймов USART зависят от бита M.

Таблица 195. Форматы фрейма(1)

Бит M	Бит PCE	Фрейм USART
0	0	I SB I 8 бит данных I STB I
0	1	I SB I 7 бит данных I PB I STB I
1	0	I SB I 9 бит данных I STB I
1	1	I SB I 8 бит данных I PB I STB I

1. Обозначения: SB: бит старта, STB: бит стопа, PB: бит чётности

При пробуждении по метке адреса бит чётности не учитывается. Бит PS определяет число передаваемых "1" с учётом 7 или 8 младших битов (зависит от бита M), включая сам бит чётности.

Чётность:

Пример: данное=00110101; 4 бита стоят => (бит PS в регистре USART_CR1 = 0).

Нечётность:

Пример: данное=00110101; 4 бита стоят => (бит PS в регистре USART_CR1 = 1).

Передача: При стоящем бите PCE регистра USART_CR1, старший передаваемый бит изменяется в зависимости бита PS для передачи чётного или нечётного числа "1". При сбое чётности ставится флаг PE регистра USART_SR. Прерывание разрешается битом PEIE регистра USART_CR1.

27.3.8. Коммутируемая локальная сеть LIN

Режим LIN включается установкой бита LINEN в регистре USART_CR2. При этом должны быть очищены биты:

- STOP[1:0], CLKEN в регистре USART_CR2
- SCEN, HDSEL и IREN в регистре USART_CR3.

Передача LIN

Для передачи ведущего LIN используется процедура из *Секции 27.3.2*. со следующими отличиями:

- Снимается бит длины слова M (8-бит).
- Ставится бит LINEN режима LIN. Теперь установка бита SBK в качестве символа разрыва посылает 13 нулей с последующей посылкой ‘1’ для обнаружения грядущего бита старт.

Приём LIN

Схема обнаружения разрыва реализована отдельно и срабатывает и при простое шины и при передаче фрейма.

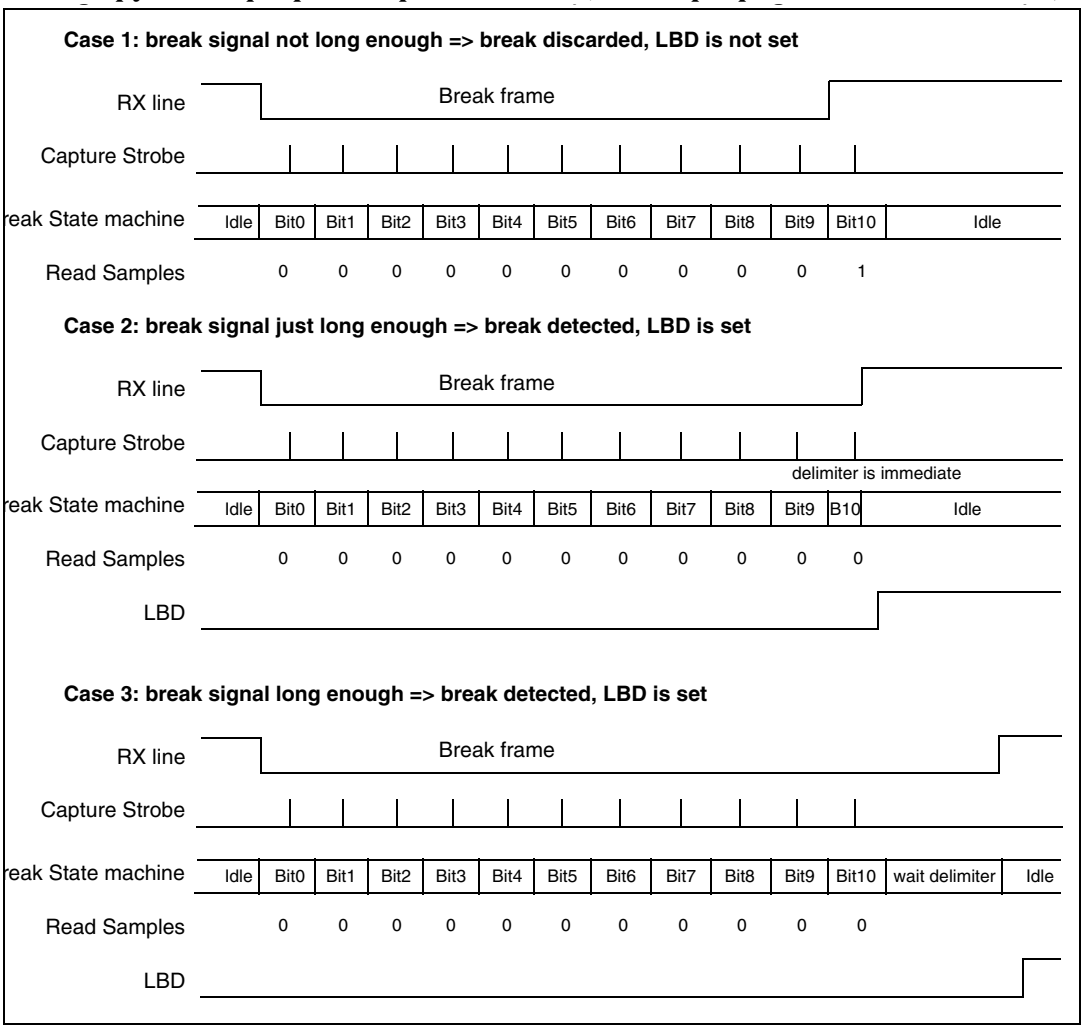
При включённом приёмнике (RE=1 в USART_CR1) схема ждёт сигнал старта на ножке RX. Метод обнаружения всё тот же. После обнаружения старта, следующие биты считываются как биты данных (по 8, 9 и 10 импульсам выборки). Если получено 10 (при LBDL = 0 в USART_CR2) или 11 (при LBDL=1 в USART_CR2) последовательных нулей с последующим разделителем, то ставится флаг LBD в регистре USART_SR. Прерывание разрешается битом LBDIE=1.

Для обнаружения разрыва линия RX должна вернуться в высокий уровень. При появлении ‘1’ перед 10 или 11 схема обнаружения разрыва сразу переключается назад в поиск бита старта.

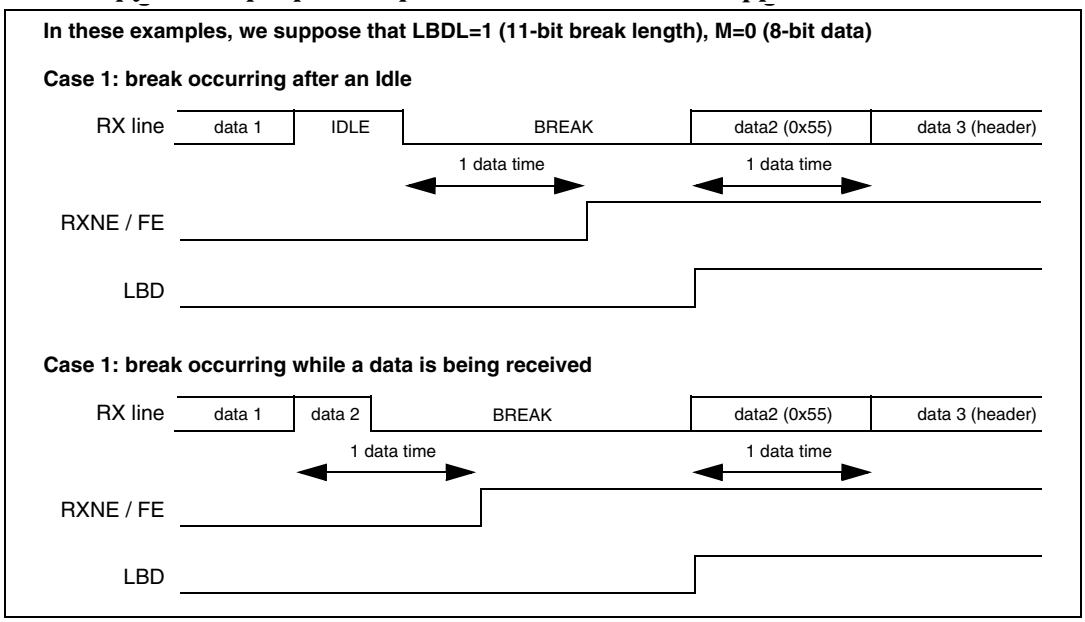
При выключенном LIN (LINEN=0), приёмник, как нормальный USART, символ разрыва не ищет.

При включённом LIN (LINEN=1), После ошибки фрейма (нулевой бит стопа, при фрейме разрыва), приёмник останавливается либо до получения схемой разрыва '1’ при незавершенном слове разрыва, либо символа разделителя при завершённом слове разрыва.

Обнаружение разрыва в режиме LIN (11-бит разрыв - бит LBDL стоит)



Обнаружение разрыва в режиме LIN и Ошибка фрейма



27.3.9. Синхронный режим USART

Включается установкой бита CLKEN в регистре USART_CR2. При этом надо снять биты:

- LINEN в регистре USART_CR2,
- SCEN, HDSEL и IREN в регистре USART_CR3.

USART может работать ведущим в двунаправленном синхронном режиме. Ножка CK выдаёт такты передатчика USART. Во время битов старта и стопа тактовые импульсы не выдаются. Бит LBCL регистра USART_CR2 разрешает выдачу тактов во время передачи последнего действительного бита данных (метка адреса). Бит CPOL в USART_CR2 задаёт полярность, а бит CPHA в USART_CR2 задаёт полярность внешних тактов.

Во время Простоя, Преамбулы и посылки разрыва внешних тактов CK нет.

Синхронный передатчик USART работает так же как асинхронный, но CK и TX синхронизируются (в соответствии с CPOL CPHA).

Работа синхронного приёмника USART отличается от асинхронного. При RE=1, данные считываются по фронту CK (см. CPOL и CPHA), а не множественной выборкой. Времена установки и удержания должны быть соответствующими (1/16 времени бита).

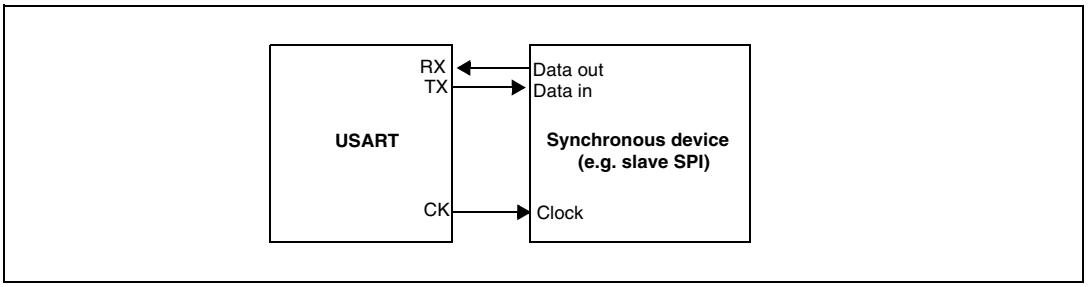
NB: Возможно принимать только данные включённого передатчика.

Ради нормальной работы тактов, биты LBCL, CPOL и CPHA пишутся только у выключенных приёмнике и передатчике (TE=RE=0), у работающих блоков их изменять нельзя.

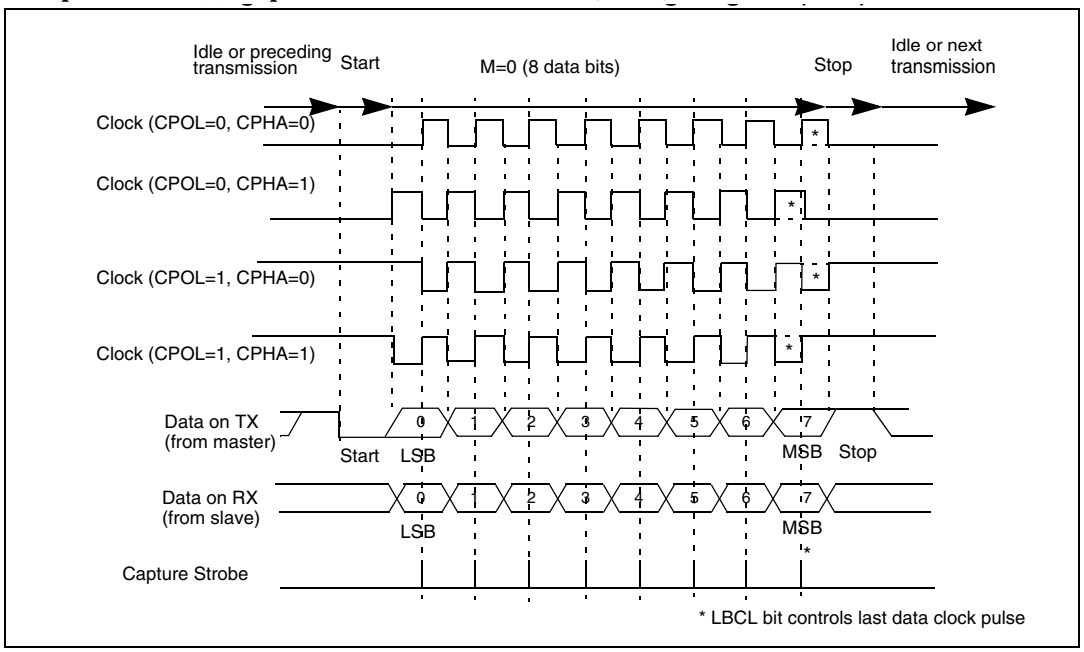
Для минимизации времён установки и удержания, биты TE и RE лучше всего ставить одной командой.

USART работает только ведущим (CK всегда вывод).

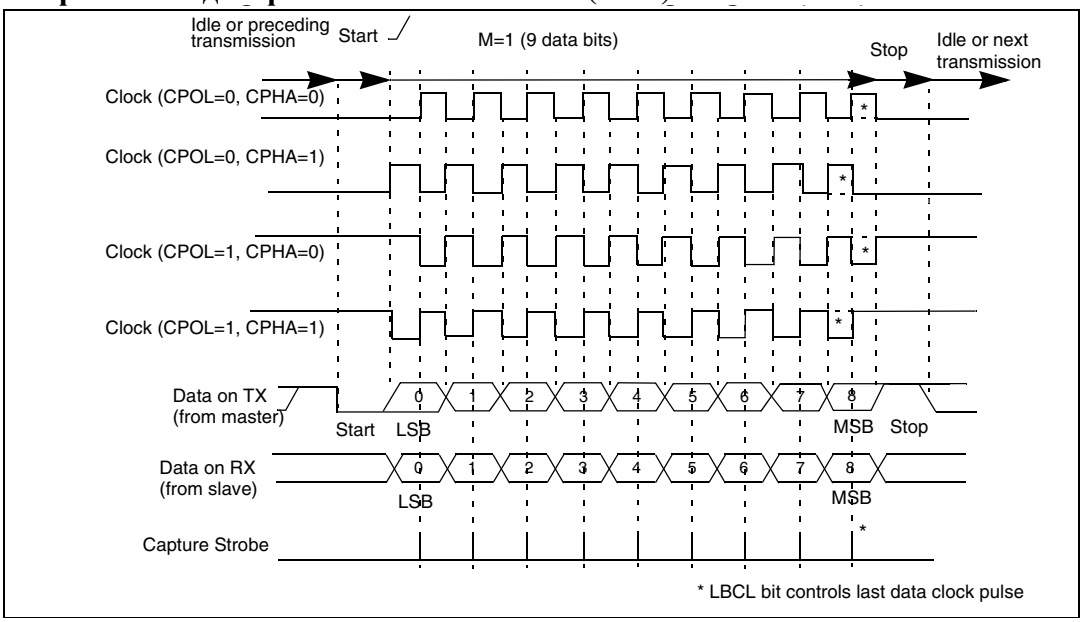
Пример синхронной передачи USART



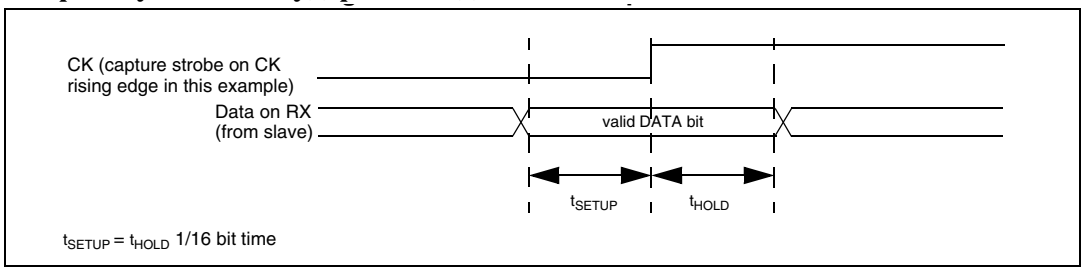
Временная диаграмма тактов USART (M=0)



Временная диаграмма тактов USART (M=1)



Время установки/удержания данных RX



В режиме Smartcard ножка CK работает по-другому.

27.3.10.Однопроводной полудуплекс

Включается установкой бита `HDSEL` в регистре `USART_CR3`. При этом должны быть сняты биты:

- `LINEN` и `CLKEN` в регистре `USART_CR2`,
- `SCEN` и `IREN` в регистре `USART_CR3`.

В этом режиме ножки TX и RX замыкаются внутри. Полудуплекс или полный дуплекс выбирается битом ‘HALF DUPLEX SEL’ (`HDSEL` в `USART_CR3`).

При стоящем HDSEL:

- RX не используется,
- Не передающая ножка TX всегда свободна, прямо как стандартный свободный или приёмный IO. То есть ножка IO для не передающей TX должна быть плавающим входом (открытый сток при выводе).

Всё остальное соответствует нормальному режиму USART. Конфликты на шине разрешаются программно. В частности, передача аппаратно не блокируется и продолжается сразу после записи регистра данных при стоящем бите TE.

27.3.11.Режим Smartcard

Включается установкой бита SCEN в регистре USART_CR3 При этом должны быть сняты биты:

- LINEN в регистре USART_CR2,
- HDSEL и IREN в регистре USART_CR3.

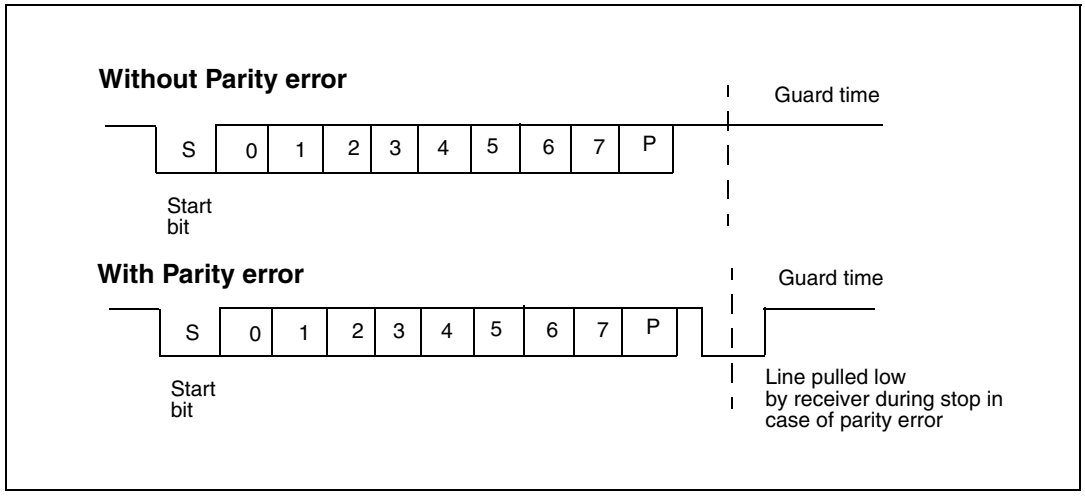
Более того, бит CLKEN можно ставить для подачи тактов на карту.

Интерфейс Smartcard определён стандартом ISO 7816-3. USART конфигурируется так:

- 8 бит и чётность: ставим M=1 и PCE=1 в регистре USART_CR1
- 1.5 бита стоп приёма и передачи (STOP='11' в регистре USART_CR2).

Можно выбрать и 0.5 бита стоп для приёма, но тогда надо переключаться.

Асинхронный протокол ISO 7816-3



Двунаправленная линия TX управляется и USART и картой, так что должна конфигурироваться с открытым стоком.

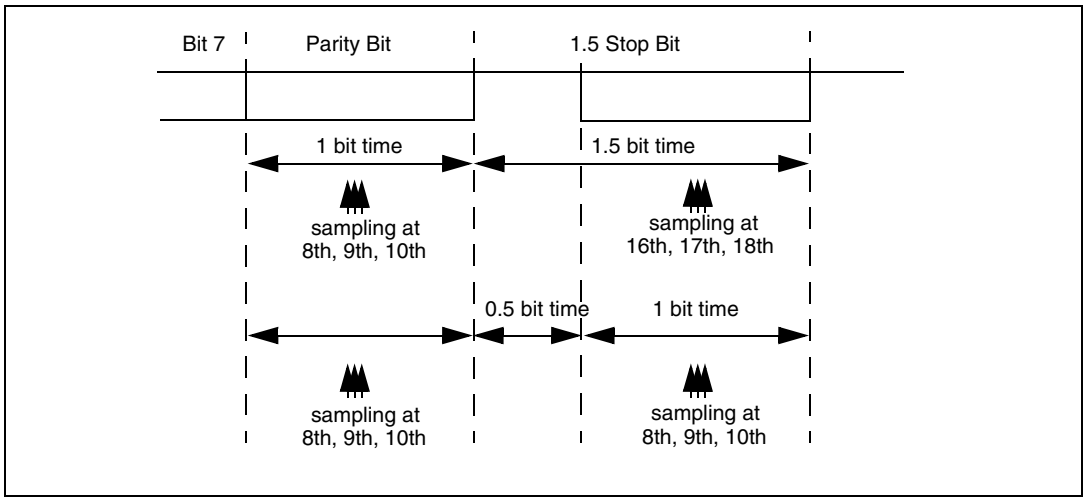
Smartcard это однопроводной полудуплексный протокол.

- Передача данных из регистра сдвига гарантированно задерживается на минимум 1/2 такта связи. При нормальной работе регистр сдвигается по следующему фронту такта связи, в режиме Smartcard эта передача ещё задерживается на 1/2 такта связи.
- При ошибке чётности приёма с 0.5 или 1.5 бита стоп после завершения приёма фрейма, линия TX уводится вниз на один такт связи (сигнал NACK), показывая карте, что данные приняты с ошибкой. На передающей стороне возникнет ошибка фрейма (у ней 1.5 бита стоп). Программа может послать данные повторно. Ошибка чётности отправляется приемником только при стоящем бите NACK.
- В режиме Smartcard установка бита TC может быть задержана относительно нормальной работы с помощью регистра защиты (Guard Time). В этом режиме опустение регистра сдвига запускает счётчик защиты, он считает до значения регистра защиты, удерживая TC низким. После этого TC встаёт.
- Режим Smartcard на снятие флага TC не влияет.
- Сигнал NACK от приёмника на передающей стороне не воспринимается как бит старта. Длительность принятого NACK может быть 1 или 2 такта связи.
- На приёмной стороне возвращаемый NACK не считается битом старта.

NB: В режиме Smartcard передача `0x00` с ошибкой фрейма не является символом разрыва, его тут нет.

NB: При переключении бита `TE` фрейм `IDLE` не передаётся.

Обнаружение ошибки чётности при 1.5 битах стоп



По ноге CK USART может выдавать карте такты. Они не связаны с передачей данных, а просто так получаются из входных тактов f_{CK} устройства делением на значение регистра. `USART_GTPR`. CK может быть от $f_{CK}/2$ до $f_{CK}/62$.

27.3.12.Блок IrDA SIR ENDEC

Режим IrDA включается установкой бита `IREN` регистра `USART_CR3`. Чистить надо биты:

- `LINEN`, `STOP` и `CLKEN` в регистре `USART_CR2`,
- `SCEN` и `HDSEL` в регистре `USART_CR3`.

Физический уровень IrDA SIR использует модуляцию RZI, логический 0 это световой импульс.

Кодер передатчика SIR получает NRZ-модулированный поток битов от USART. Для SIR ENDEC поддерживается только скорость 115.2 Кбит/с. В нормальном режиме ширина импульса равна 3/16 периода бита.

Декодер приёмника SIR демодулирует RZ поток битов от детектора излучения в NRZ поток битов для USART. Вход декодера без сигнала обычно высокий, выход кодера низкий. Бит старта на входе декодера низкий.

- IrDA это полудуплексный протокол. Если передатчик занят (USART посылает данные кодеру IrDA), то данные на линии IrDA декодером игнорируются. Если занят приёмник, то данные на TX от USART в IrDA не кодируются. При приёме данных надо избегать передачи.
- A‘0’ передаётся высоким импульсом, а ‘1’ передаётся как ‘0’. Ширина импульса в нормальном режиме равна 3/16 периода бита.
- Декодер SIR преобразует приёмный сигнал IrDA в битовый поток USART.
- Приёмная логика SIR воспринимает высокий уровень как логическую единицу, а низкий уровень как ноль.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when idle.
- Спецификация IrDA требует приёма импульса длиннее 1.41 us. Это программируемый параметр. Логика определения глитчей отфильтровывает импульсы короче 2 периодов `PSC` (`PSC` это значение предделителя IrDA в регистре `USART_GTPR`). Импульсы короче 1 периода `PSC` всегда отбрасываются, короче 2 периодов могут приниматься, длиннее 2 периодов принимаются как импульсы. При `PSC`=0 кодер/декодер IrDA не работают.
- Приёмник может работать с экономным передатчиком.
- В режиме IrDA используется один бит стопа.

Экономный режим IrDA

Передатчик

В этом режиме ширина импульса не равна 3/16 периода бита. Она равна трём тактам частоты экономного режима, обычно 1.8432 MHz (1.42 MHz < PSC < 2.12 MHz). Для этой частоты есть программируемый предделитель экономного режима.

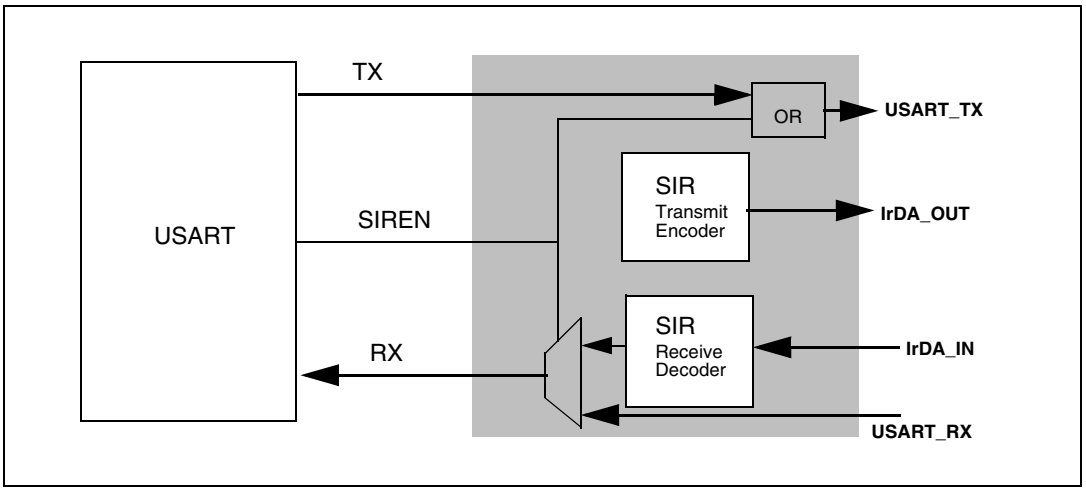
Приёмник

Это похоже на приём нормального режима. Детектор глитчей USART отбрасывает импульсы короче 1/PSC. Принимаются импульсы длиннее 2 периодов тактов IrDA экономного режима (значение PSC в USART_GTPR).

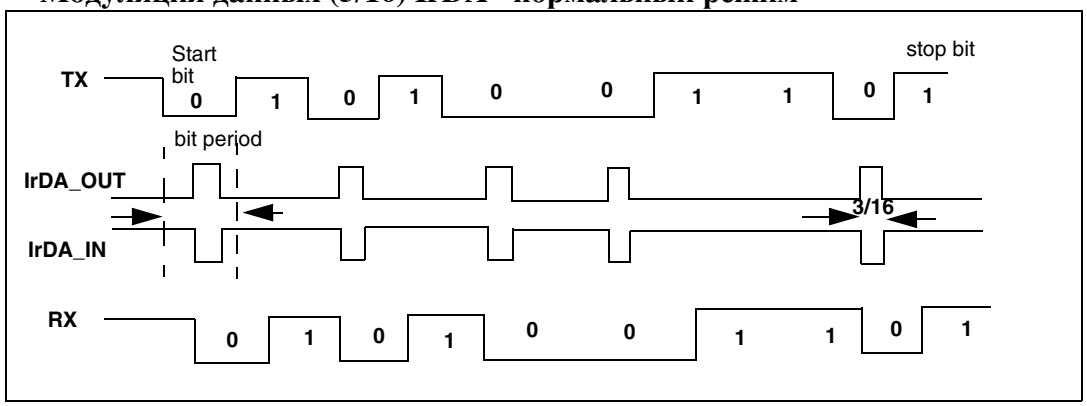
NB: Импульс длиннее 1 и короче 2 периодов PSC могут приниматься, а могут и не приниматься.

Время установки приёмника управляется программно. Физический уровень IrDA должен иметь минимальную задержку 10 ms между приёмом и передачей.

Блок-схема IrDA SIR ENDEC



Модуляция данных (3/16) IrDA - нормальный режим



27.3.13.Непрерывная связь с DMA

Для непрерывных передач нужно вовремя программно снимать биты TXE/ RXNE, а DMA снимает их сам.

Передача с DMA

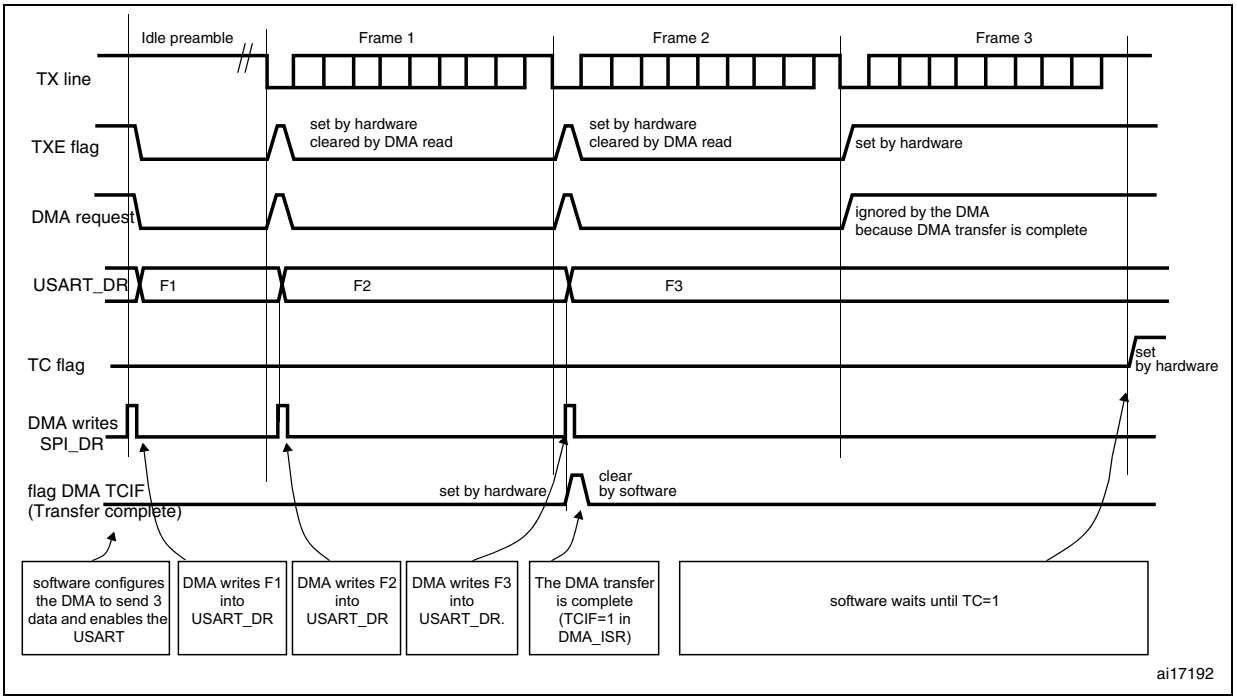
Включается установкой бита DMAT в регистре USART_CR3. Данные пишутся из SRAM в регистр USART_DR по каждой установке бита TXE. Канал DMA назначается USART процедурой:

1. Пишем адрес регистра USART_DR в регистр адреса получателя DMA.
2. Пишем адрес памяти в регистр адреса источника DMA.
3. Пишем общее число передаваемых байтоа в регистр управления DMA.
4. Ставим приоритет канала DMA
5. Определяем выдачу прерываний DMA после половины/всей передачи.
6. Снимаем бит TC регистра SR записью 0.
7. Активируем канал DMA.

После передачи заданного числа байтов контроллер DMA выдаёт прерывание по своему вектору.

После передачи DMA всех заданных байтов (флаг TCIF в регистре DMA_ISR), конец операции USART можно отследить по флагу TC. Для выключения USART или останова MCU нужно программно дождаться TC=1. Флаг TC аппаратно ставится по концу последнего фрейма передачи.

Передача с DMA



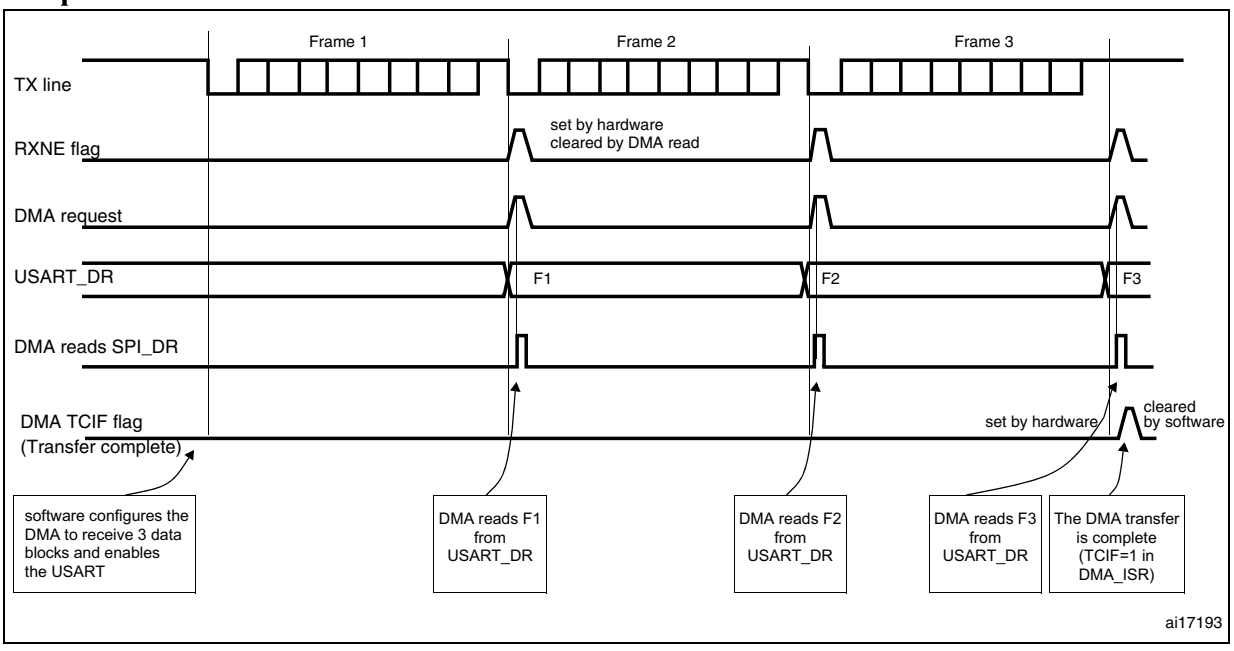
Приём с DMA

Включается установкой бита DMAR регистра USART_CR3. Данные пишутся из USART_DR в SRAM. Канал DMA назначается USART процедурой:

1. Пишем адрес регистра USART_DR в регистр адреса источника DMA.
2. Пишем адрес памяти в регистр адреса получателя DMA.
3. Пишем общее число передаваемых байтоа в регистр управления DMA.
4. Ставим приоритет канала DMA
5. Определяем выдачу прерываний DMA после половины/всей передачи.
6. Снимаем бит TC регистра SR записью 0.
7. Активируем канал DMA.

После передачи заданного числа байтов контроллер DMA выдаёт прерывание по своему вектору.

Приём с DMA



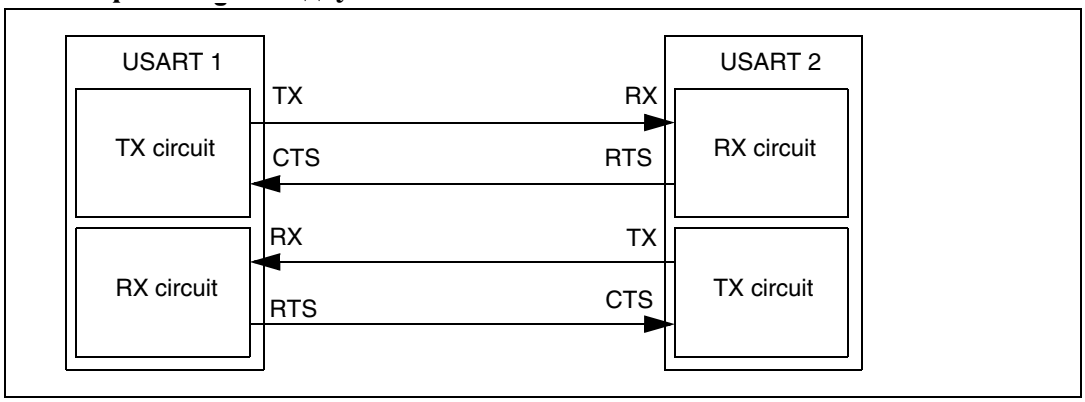
Флаги ошибок и прерывания при многобуферных передачах

При многобуферных передачах флаги ошибок ставятся после сбойного байта. Прерывание нужно разрешать. Ошибка фрейма, переполнение и флаг шума, которые при передаче одного байта ставятся вместе с RXNE, есть отдельный общий флаг разрешения прерывания ошибки (бит EIE регистра USART_CR3).

27.3.14.Аппаратное управление потоком

Для передач между двумя USART можно использовать их входы CTS и выходы RTS.

Аппаратная связь двух USART

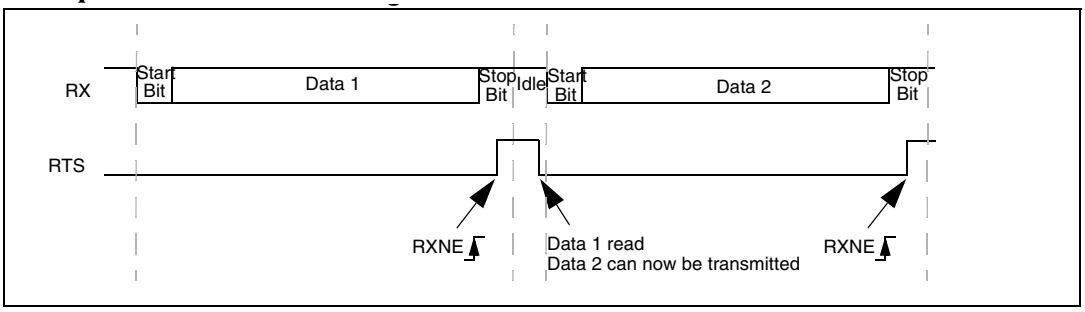


RTS и CTS включаются независимо установкой битов RTSE и CTSE регистра USART_CR3.

Управление RTS

При RTSE=1 нога RTS держится низкой во время готовности USART принять новое данное. При заполнении регистра сдвига, RTS снимается, извещая, что передатчику пора стопорить.

Управление потоком RTS

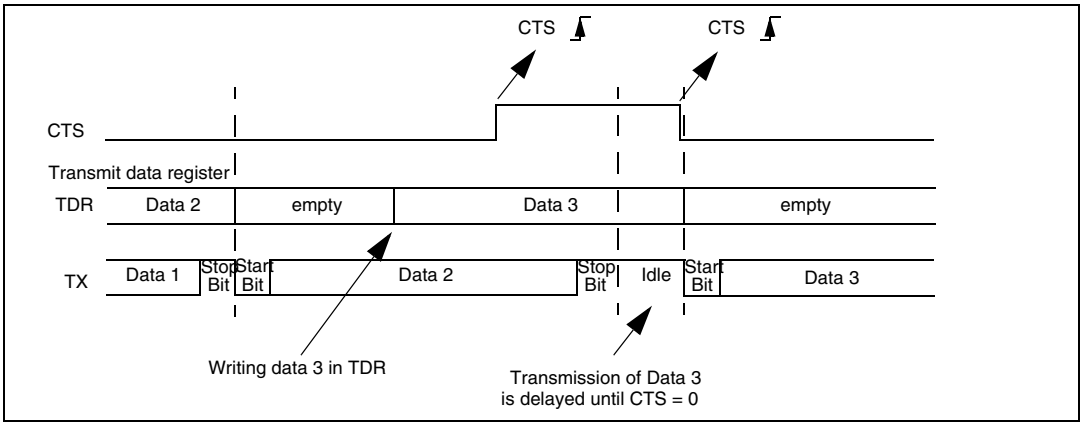


Управление CTS

При CTSE=1 передатчик перед передачей данных ждёт низкого входа CTS. Если CTS снимается во время передачи, то она прекращается.

При CTSE=1 флаг CTSIF аппаратно ставится при переключении входа CTS, показывая готовность и неготовность приёмника. Прерывание разрешается битом CTSIE регистра USART_CR3.

Управление потоком CTS



27.4. Прерывания USART

Таблица 196. Запросы прерываний USART

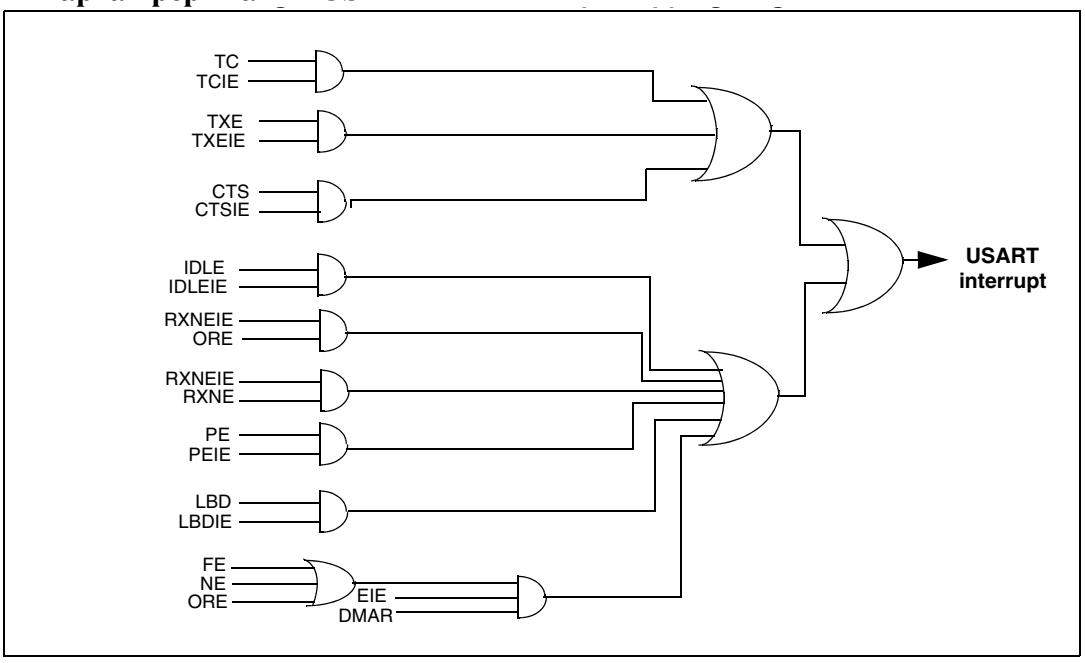
Прерывание	Флаг	Бит разрешения
Регистр данных передатчика пуст	TXE	TXEIE
Флаг CTS	CTS	CTSIE
Передача завершена	TC	TCIE
Принятые данные можно читать	RXNE	RXNEIE
Переполнение	ORE	
Простой линии	IDLE	IDLEIE
Ошибка чётности	PE	PEIE
Флаг разрыва	LBD	LBDIE
Флаг Шума, Переполнения и Ошибки фрейма при многобуферной связи	NE или ORE или FE	EIE(1)

1. Используется только при передачах с DMA.

Все прерывания USART объединены в один вектор.

- При передаче: Передача завершена, Могу Передавать или Регистр Данных Пуст.
- При приёме: Простой линии, Переполнение, Регистр Данных не Пуст, Ошибка чётности, Разрыв LIN, Флаг Шума (только многобуферный) и Ошибка Фрейма (только многобуферный).

Карта прерываний USART



27.5. Режимы USART

Таблица 197. Режимы USART(1)

Режим USART	USART1	USART2	USART3	UART4	UART5
Асинхронный	X	X	X	X	X
Аппаратное управление режимом	X	X	X	NA	NA
Многобуферная связь (DMA)	X	X	X	X	NA
Многопроцессорная связь	X	X	X	X	X
Синхронная связь	X	X	X	NA	NA
Smartcard	X	X	X	NA	NA
Полудуплекс (Однопроводной)	X	X	X	X	X
IrDA	X	X	X	X	X
LIN	X	X	X	X	X

1. X = поддерживается; NA = не применим.

27.6. Регистры USART

Регистры доступны полусловами (16-бит) и словами (32-бит).

27.6.1. Регистр состояния USART (USART_SR)

Смещение адреса: 0x00

По сбросу: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

— Биты 31:10 Резерв, не трогать.

— Бит 9 **CTS**: Флаг CTS

Ставится аппаратно при переключении входа CTS и стоящем CTSE. Снимается записью 0. Прерывание выдаётся при CTSIE=1 в регистре USART_CR3

0: Линия CTS не менялась

1: Линия CTS менялась

Бит недоступен для UART4 и UART5.

— Бит 8 **LBD**: Флаг разрыва LIN

Ставится аппаратно при обнаружении разрыва LIN. Снимается записью 0. Прерывание выдаётся при LBDIE = 1 в регистре USART_CR2.

0: Разрыва LIN не было

1: Был разрыв LIN

— Бит 7 **TXE**: Регистр данных передатчика пуст

Ставится аппаратно после передачи TDR в регистр сдвига. Прерывание выдаётся при TXEIE=1 в регистре USART_CR1. Снимается записью в USART_DR.

0: Регистр данных занят

1: Регистр данных пуст

NB: Используется при однобуферной передаче.

— Бит 6 **TC**: Передача завершена

Ставится аппаратно после завершения передачи фрейма при стоящем TXE. Прерывание выдаётся при TCIE=1 в регистре USART_CR1. Снимается чтением из USART_SR и затем записью в USART_DR. В многобуферном случае рекомендуется снимать записью '0'.

0: Передача не завершена

1: Передача завершена

— Бит 5 **RXNE**: Регистр данных приёмника не пуст

Ставится аппаратно после передачи регистра сдвига в USART_DR. Прерывание выдаётся при RXNEIE=1 в регистре USART_CR1. Снимается чтением из USART_DR. В многобуферном случае рекомендуется снимать записью '0'.

0: Данные не приняты.

1: Данные можно читать.

— Бит 4 **IDLE**: Простой линии

Ставится аппаратно при обнаружении Простоя линии. Прерывание выдаётся при IDLEIE=1 в регистре USART_CR1. Снимается чтением из USART_SR и затем чтением USART_DR.

0: Простоя не было

1: Символ простоя был

NB: Снова бит IDLE не встанет пока не встанет бит RXNE (был новый простой).

— Бит 3 **ORE**: Переполнение

Ставится аппаратно после приёма в регистр сдвига при ещё нечитаном RDR пока RXNE=1. Прерывание выдаётся при RXNEIE=1 в регистре USART_CR1. Снимается чтением из USART_SR и затем чтением USART_DR.

0: Не было

1: Переполнилось

NB: Если бит стоит, то содержимое RDR, но заменяется содержимое регистра сдвига. Прерывание выдаётся при стоящем бите EIE в многобуферном случае.

— Бит 2 **NE**: Флаг Шума

Ставится аппаратно при обнаружении шума в принятом фрейме. Снимается чтением из USART_SR и затем чтением USART_DR.

0: Не было

1: Нашумело

NB: Прерывание по этому биту выдаётся при стоящем бите EIE в многобуферном случае, а обычно прерывается по RXNE.

— Бит 1 **FE**: Ошибка фрейма

Ставится аппаратно при потере синхронизации, излишнем шуме или символе разрыва. Снимается чтением из USART_SR и затем чтением USART_DR.

0: Не было

1: Была ошибка или разрыв

NB: Если ошибка фрейма и переполнение возникают одновременно, то фрейм передаётся и ставится только бит ORE.

Прерывание по этому биту выдаётся при стоящем бите EIE в многобуферном случае, а обычно прерывается по RXNE..

— Бит 0 **PE**: Ошибка чётности

Ставится аппаратно при ошибке чётности на приёме. Снимается чтением из USART_SR и затем чтением USART_DR. Перед очисткой надо дождаться флага RXNE.

Прерывание выдаётся при PEIE = 1 в регистре USART_CR1.

0: Ошибки нет

1: Ошибка чётности

27.6.2. Регистр данных USART (USART_DR)

Смещение адреса: 0x04

По сбросу: Всякая фигня.

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							DR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:9 Резерв, не трогать.

— Биты 8:0 **DR[8:0]**: Данные приёма (RDR) и передачи (TDR)

При включённом контроле чётности старший передаваемый бит заменяется битом чётности.

27.6.3. Регистр скорости USART (USART_BRR)

NB: Соответствующий счётчик скорости останавливается при снятом TE или RE.

Смещение адреса: 0x08

По сбросу: 0x0000

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

— Биты 31:9 Резерв, не трогать.

— Биты 15:4 **DIV_Mantissa[11:0]**: мантисса USARTDIV

— Биты 3:0 **DIV_Fraction[3:0]**: дробная часть USARTDIV

27.6.4. Регистр 1 управления USART (USART_CR1)

Смещение адреса: 0x0C

По сбросу: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:14 Резерв, не трогать.

— Бит 13 **UE**: Включение USART

Ставится и снимается программно.

0: Предделитель и выходы USART отключаются после завершения передачи байта

1: USART работает

— Бит 12 **M**: Длина слова

Ставится и снимается программно.

0: 1 бит Старт, 8 битов данных, n битов Стоп

1: 1 бит Старт, 9 битов данных, n битов Стоп

NB: Нельзя изменять во время чтения или записи

— Бит 11 **WAKE**: Способ побудки

Ставится и снимается программно.

0: Простой линии

1: Метка адреса

— Бит 10 **PCE**: Включение контроля чётности

Ставится и снимается программно. При включённом контроле чётности старший передаваемый бит (9 при M=1, 8 при M=0) заменяется битом чётности. На передаваемый в момент установки байт не влияет.

0: Выключен

1: Включён

— Бит 9 **PS**: Выбор чётности

Ставится и снимается программно. На передаваемый в момент установки байт не влияет.

0: Чётность

1: Нечётность

— Бит 8 **PEIE**: Разрешение прерывания PE

Ставится и снимается программно.

0: Нельзя

1: Можно

— Бит 7 **TXEIE**: Разрешение прерывания TXE

Ставится и снимается программно.

0: Нельзя

1: Можно

— Бит 6 **TCIE**: Разрешение прерывания TC конца передачи

Ставится и снимается программно.

0: Нельзя

1: Можно

— Бит 5 **RXNEIE**: Разрешение прерывания RXNE и ORE

Ставится и снимается программно.

0: Нельзя

1: Можно

— Бит 4 **IDLEIE**: Разрешение прерывания IDLE

Ставится и снимается программно.

0: Нельзя

1: Можно

— Бит 3 **TE**: Включение передатчика

Ставится и снимается программно.

0: Выключен

1: Включён

NB:

1: Установка TE посылает на линию преамбулу (Простой линии) после текущего слова, кроме режима Smartcard.

2: При стоящем TE перед началом передачи появляется задержка на 1 бит.

— Бит 2 **RE**: Разрешение приёмника

Ставится и снимается программно.

0: Выключен

1: Включён, начинает искать бит старта

— Бит 1 **RWU**: Побудка приёмника

Задаёт немой режим приемника. Ставится и снимается программно, может сняться аппаратно при обнаружении последовательности побудки.

0: Приёмник активен

1: Приёмник нем

NB:

1: Перед установкой бита RWU USART должен сначала принять байт данных, иначе не сможет проснуться по Простою линии.

2: При побудке по Метке адреса (WAKE bit=1) бит RWU нельзя менять при стоящем с RXNE.

— Бит 0 **SBK**: Посылка символа разрыва

Ставится программно, снимается аппаратно во время бита стоп разрыва.

0: Ничего

1: Будет передан символ разрыва

27.6.5. Регистр 2 управления USART (USART_CR2)

Смещение адреса: 0x10

По сбросу: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LINEN	STOP[1:0]		CLK EN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	Res.	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

— Биты 31:15 Резерв, не трогать.

— Бит 14 **LINEN**: Включение режима LIN

Ставится и снимается программно.

0: Выключен

1: Включён, можно обнаруживать и посылать символ разрыва(13 низких битов).

— Биты 13:12 **STOP**: Число битов STOP
  00: 1 бит
  01: 0.5 бита
  10: 2 бита
  11: 1.5 бита
  The 0.5 и 1.5 бита в UART4 и UART5 нет.
— Бит 11 **CLKEN**: Включение тактов на ноге CK
  0: Выключено
  1: Включено
  Нету в UART4 и UART5.
— Бит 10 **CPOL**: Полярность тактов CK в синхронном режиме
  Работает вместе с битом CPHA
  0: Ножка CK спокойна низким вне окна передачи.
  1: Ножка CK спокойна высоким вне окна передачи.
  Нету в UART4 и UART5.
— Бит 9 **CPHA**: Фаза тактов CK в синхронном режиме
  Работает вместе с битом CPOL
  0: Фронт считывания первого данного в первом такте.
  1: Фронт считывания первого данного во втором такте.
  Нету в UART4 и UART5.
— Бит 8 **LBCL**: Тактовый импульс последнего бита в синхронном режиме
  0: На ноге CK его нет
  1: На ноге CK он есть
  Нету в UART4 и UART5.
— Бит 7 Резерв, не трогать.
— Бит 6 **LBDIE**: Разрешение прерывания разрыва LIN
  0: Нельзя
  1: Можно
— Бит 5 **LBDL**: Длина определения разрыва
  0: За 10 битов
  1: За 11 битов
— Бит 4 Резерв, не трогать.
— Биты 3:0 **ADD[3:0]**: Адрес узла USART в многопроцессорной системе

Биты CPOL, CPHA, LBCL нельзя писать при включённом передатчике.

27.6.6. Регистр 3 управления USART (USART_CR3)

Смещение адреса: 0x14

По сбросу: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:11 Резерв, не трогать.
— Бит 10 **CTSIE**: Разрешение прерывания CTS
  0: Нельзя
  1: Можно
  Нету в UART4 и UART5.
— Бит 9 **CTSE**: Разрешение аппаратного управления CTS
  0: Выключено
  1: Включено, данные передаются только при низком CTS. Выключение CTS во время передачи прекращает её. Данные, записанные в DR при снятом CTS, передаются после его установки.
  Нету в UART4 и UART5.

— Бит 8 **RTSE**: Разрешение аппаратного управления RTS
0: Выключено
1: Включено, данные запрашиваются только при пустом буфере (RTS низкий).
Нету в UART4 и UART5.
— Бит 7 **DMAT**: Разрешение использования DMA передатчика
Ставится и снимается программно.
0: Нельзя
1: Можно
Нету в UART5.
— Бит 6 **DMAR**: Разрешение использования DMA приёмника
Ставится и снимается программно.
0: Нельзя
1: Можно
Нету в UART5.
— Бит 5 **SCEN**: Разрешение режима Smartcard
0: Smartcard выключен
1: Smartcard включён
Нету в UART4 и UART5.
— Бит 4 **NACK**: Разрешение выдачи NACK Smartcard при ошибке чётности
0: Нельзя
1: Можно
Нету в UART4 и UART5.
— Бит 3 **HDSEL**: Выбор полудуплекса или полного дуплекса
0: Полный дуплекс
1: Полудуплекс
— Бит 2 **IRLP**: Экономный режим IrDA
0: Нормальный
1: Экономный
— Бит 1 **IREN**: Включение IrDA
Ставится и снимается программно.
0: IrDA выключен
1: IrDA включён
— Бит 0 **EIE**: Разрешение прерывания ошибки
0: Нельзя
1: Можно при DMAR=1 в регистре USART_CR3 или FE=1 или ORE=1 или NE=1 в USART_SR.

27.6.7. Регистр защиты и предделителя (USART_GTPR)

Смещение адреса: 0x18
По сбросу: 0x0000

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

— Биты 31:16 Резерв, не трогать.
— Биты 15:8 **GT[7:0]**: Значение защиты в тактах связи режима Smartcard
Флаг TC ставится после отсчёта данного значения. В UART4 и UART5 нету.
— Биты 7:9 PSC[7:0]: Значение предделителя
– **В экономном режиме IrDA**:
PSC[7:0] = Скорость связи IrDA
Делитель входных тактов (8 значащих бит):
00000000: Резерв, не трогать.
00000001: делит на 1
00000010: делит на 2
...

– **В нормальном режиме IrDA:** PSC должен быть равен 00000001.

– **В режиме Smartcard**:

PSC[4:0]: Значение предделителя

Входные такты делятся на удвоенное значение предделителя (5 значащих бит):

00000: Резерв, не трогать.

00001: делит на 2

00010: делит на 4

00011: делит на 6

...

NB: Биты [7:5] в режиме Smartcard не используются.

В UART4 и UART5 нету.

27.6.8. Карта регистров USART

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_SR	Reserved																						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value																							0	0	1	1	0	0	0	0	0	0
0x04	USART_DR	Reserved																						DR[8:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x08	USART_BRR	Reserved																DIV_Mantissa[15:4]												DIV_Fraction [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USART_CR1	Reserved																		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_CR2	Reserved																	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Reserved	LBDIE	LBDL	Reserved	ADD[3:0]			
	Reset value																		0	0	0	0	0	0	0		0	0		0	0	0	0
0x14	USART_CR3	Reserved																					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x18	USART_GTPR	Reserved																GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

28. Полноскоростной автономный USB (OTG_FS)

28.1. Введение в OTG_FS

Использованы сокращения:

FS	Полная скорость
LS	Низкая скорость
MAC	Контроллер доступа среды
OTG	On-the-go (здесь "*автономный*")
PFC	Контроллер FIFO пакетов
PHY	Физический уровень
USB	Универсальная последовательная шина
UTMI	USB 2.0 transceiver macrocell interface (UTMI)

OTG_FS это двухфункциональный (DRD) контроллер, поддерживающий режимы периферийного устройства и хоста USB, в соответствии с *On-The-Go Supplement to the USB 2.0 Specification*. Может работать только хостом и только устройством в соответствии с *USB 2.0 Specification*. В режиме хоста OTG_FS поддерживает полноскоростные (FS, 12 Мбит/с) и низкоскоростные (LS, 1.5 Мбит/с) передачи, а в режиме устройства только полноскоростные (FS, 12 Мбит/с) передачи. OTG_FS поддерживает и HNP и SRP. В режиме хоста нужно только внешнее питание V_{BUS}.

28.2. Основные свойства OTG_FS

Их можно разделить на три категории: общие, хоста и устройства.

28.2.1. Общие свойства OTG_FS

Они таковы:

- Интерфейс USB-IF, сертифицированный для *Universal Serial Bus Specification Rev 2.0*
- Полная поддержка (PHY) необязательного протокола OTG см. *On-The-Go Supplement Rev 1.3 specification*
- *On-The-Go Supplement Rev 1.3 specification*

— Встроенная поддержка Идентификации Устройств A-B (линия ID)

— Встроенная поддержка хостом Протокола Сообщений (HNP) и Протокола Запроса Сессии SRP

— Выключение хостом V_{BUS} для экономии батарей систем OTG

— Поддержка контроля OTG уровня V_{BUS} внутренними компараторами

— Поддержка динамического переключения режимов хост/устройство

- Программная конфигурация в режиме:

— Устройство USB FS с SRP (устройство B)

— Хост USB FS с SRP (устройство A)

— Двухфункциональный OTG FS USB

- Поддержка FS SOF и поддержки соединения LS с:

— Импульс SOF от PAD

— Импульс SOF от внутреннего таймера 2 (TIM2)

— Изменяемый период фрейма

— Прерывание конца фрейма

- Включает экономию энергии вроде останова системы во время остановки USB, выключения тактового домена ядра, управления питанием PHY и DFIFO
- Выделенная 1.25 КБ RAM с продвинутым управлением FIFO:

— Программируемое разделение пространства RAM для разных FIFO

— Все FIFO могут содержать много пакетов

— Динамическое выделение памяти

— Размеры FIFO не кратны степени двойки ради неразрывности памяти

- Гарантия максимальной полосы пропускания USB до одного фрейма (1 ms) без вмешательства системы

28.2.2. Свойства хоста OTG_FS

Это:

- Внешнее питание V_{BUS}.
- До 8 каналов хоста с динамической конфигурацией для любого типа передач USB.
- Встроенный аппаратный планировщик, имеющий:

— Аппаратную очередь для 8 прерываний плюс запросов изохронных передач

— Аппаратную очередь для 8 управляющих событий плюс запросов не-периодических групповых передач

- Управление общим RX FIFO, периодическим и не-периодическим TX FIFO.

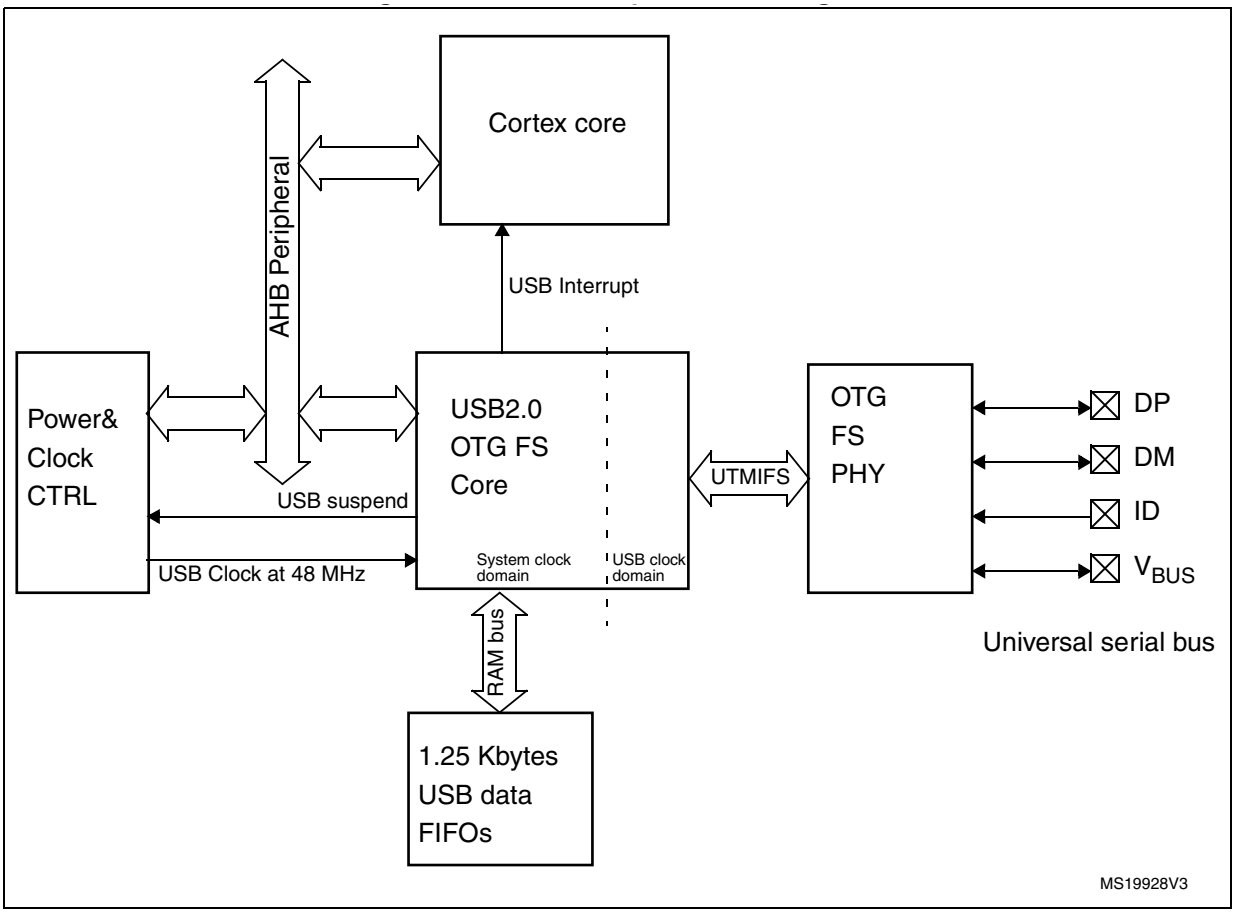
28.2.3. Свойства устройства OTG_FS

Это:

- 1 двунаправленная конечная точка 0
- 3 конечные точки (EP) IN для Групповых, Прерывающих и Изохронных передач
- 3 конечные точки OUT для Групповых, Прерывающих и Изохронных передач
- Управление общим Rx FIFO и Tx-OUT FIFO
- Управление до 4 выделенными Tx-IN FIFO (по одному для каждой активной IN EP)
- Поддержка программного отключения.

28.3. Функциональное описание OTG_FS

Блок-схема OTG_FS



28.3.1. Полноскоростное ядро OTG

USB OTG FS получает 48 MHz ±0.25% такты от контроллера сброса и тактов (RCC) через внешний кварц. Такты USB питают 48 MHz домен на полной скорости (12 Мбит/с) и должны включаться до конфигурации ядра OTG FS.

CPU и OTG FS соединены шиной AHB. Для USB OTG выделена одна линия прерываний.

CPU посылает данные через USB записью 32-бит слов по адресам регистров OTG_FS. Они автоматически идут в Tx-FIFO данных в памяти данных USB. Каждой IN EP (устройство) и каждому выходному каналу (хост) выделяется один Tx-FIFO.

CPU получает данные от USB чтением 32-бит слов из адресов регистров OTG_FS. Они автоматически идут из общего Rx-FIFO в 1.25 КБ памяти данных USB. Каждой OUT EP (устройство) и каждому входному каналу (хост) выделяется один регистр Rx-FIFO.

Уровень протокола USB использует последовательный преобразователь (SIE) и с шиной USB общается через встроенный приёмо-передатчик физического уровня (PHY).

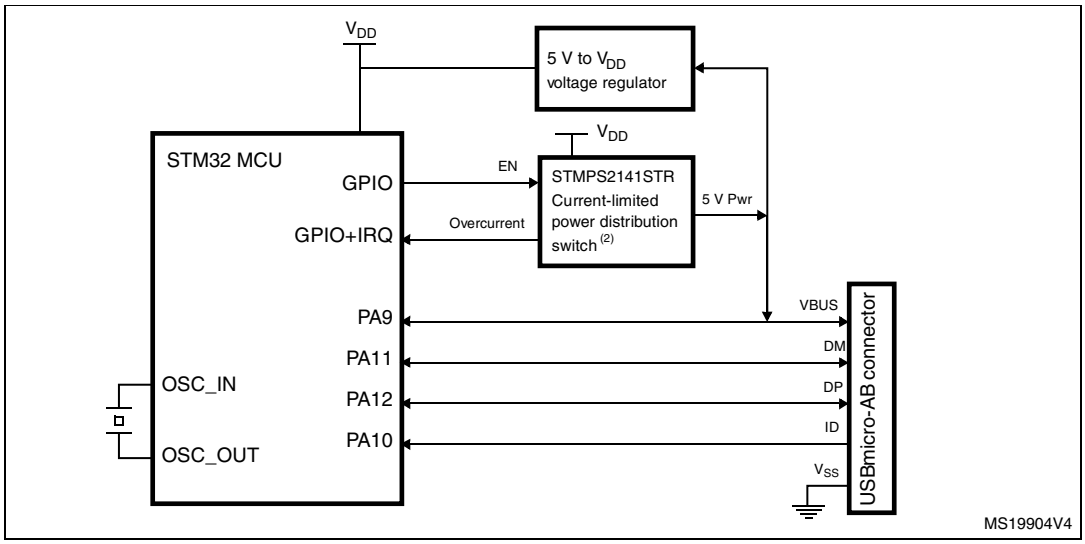
28.3.2. Полноскоростной OTG PHY

OTG PHY использует подмножество шины UTMI+ Bus (UTMIFS) и включает компоненты:

- Модуль приёмопередатчика FS/LS хоста и устройства.
- Встроенный резистор подпорки линии ID для идентификации устройств A/B.
- Встроенные резисторы подпорки и подтяжки линий DP/DM, управляемые ядром OTG_FS. В режиме устройства к V_{BUS} подключается резистор подпорки DP (полноскоростная периферия). В режиме хоста к обеим линиям DP/DM подключаются резисторы подтяжки. Все резисторы подключаются динамически в соответствии с протоколом HNP.
- Резистор подпорки DP состоит из 2 раздельно подключаемых резисторов. Динамическая подстройка подпорки DP помогает бороться с шумами сигнала Tx/Rx.
- Компараторы уровня V_{BUS} с гистерезисом для определения V_{BUS} в Норме, A-B Сессия в Норме и порогов напряжения конца сессии. Они используются в протоколе запроса сессии (SRP), определяя начало и конец сессии и следят за уровнем V_{BUS}.
- Импульсная схема заряда/разряда V_{BUS} через во время SRP (слабый привод).

28.4. Двухфункциональный OTG_FS (DRD)

Подключение устройства OTG A-B



1. Внешний регулятор напряжения только при питании устройства от V_{BUS}
2. STMPS2141STR на плате системы требуется только если надо подать V_{BUS} на внешнее устройство.

28.4.1. Определение линии ID

Режим хоста или периферии (по умолчанию) подразумевается зависящим от входной ножки ID. Состояние линии ID зависит от стороны кабеля USB, воткнутого в "разъёмаму" micro-AB.

- Если вставлена сторона B с оборванным проводом ID, то встроенный резистор подпорки создаёт высокий уровень ID и это Периферия.
- Если вставляется сторона A с заземлённым ID, то OTG_FS выдаёт прерывание изменения состояния линии ID (бит CIDSCHG в OTG_FS_GINTSTS) для инициализации программ хоста и автоматически переключается в режим хоста.

28.4.2. Двухфункциональное устройство HNP

Бит разрешения протокола HNP (бит HNPCAP в OTG_FS_GUSBCFG) позволяет ядру OTG_FS динамически переключаться из A-хоста в A-периферию и наоборот, и из B-периферии в B-хост. Текущее состояние устройства можно определить по сочетанию битов состояния ID (бит CIDSTS в OTG_FS_GOTGCTL) текущего режима (бит CMOD в OTG_FS_GINTSTS).

Программная модель HNP описана в *Секции 28.17*.

28.4.3. Двухфункциональное устройство SRP

Бит разрешения протокола SRP (бит SRPCAP в OTG_FS_GUSBCFG) разрешает ядру OTG_FS выключать питание от V_{BUS} для A-устройств, они и так всегда с ним работают.

Программная модель SRP A/B-устройств описана в *Секции 28.17*.

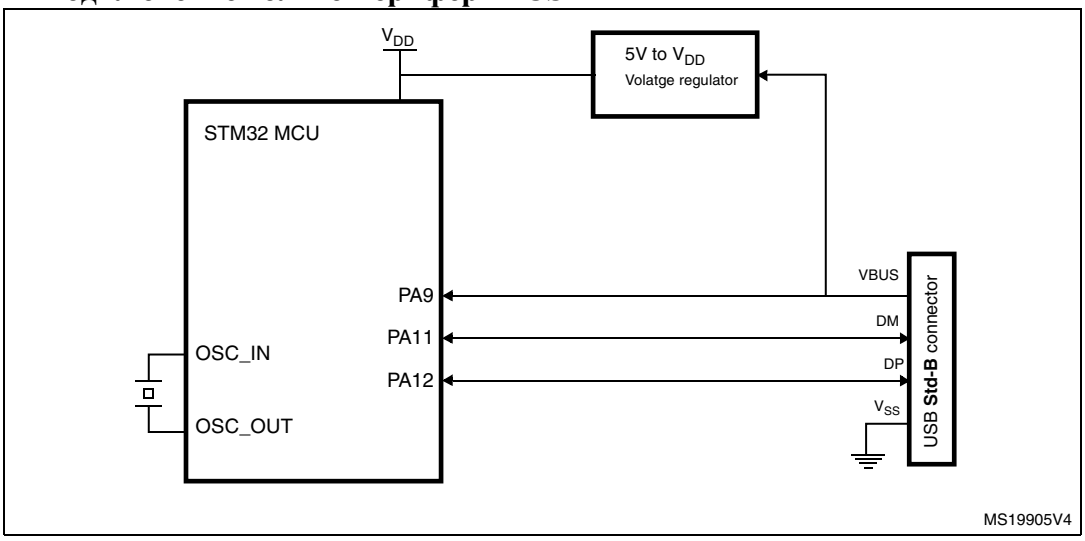
28.5. Периферийный USB

OTG_FS работает периферией USB как:

- OTG B-периферия
  - —OTG B-устройство по умолчанию, если кабель USB воткнут стороной B
- OTG A-периферия
  - —OTG A-устройство после того как HNP переключает OTG_FS в режим периферии
- B-устройство
  - —Если линия ID есть, работает и подключена к стороне B кабеля USB, и очищен бит HNPCAP в регистре OTG_FS_GUSBCFG.
- Только периферия:
  - —Стоит бит FDMOD в регистре OTG_FS_GUSBCFG. В этом случае линия ID игнорируется, даже если она есть в разъёме.

NB: При питании устройств в режимах B и только-периферии от V_{BUS}, V_{DD} надо подавать через внешний регулятор.

Подключение только-периферии USB



28.5.1. Устройство с SRP

Разрешение протокола SRP включается битом SRPCAP в регистре OTG_FS_GUSBCFG. Таким образом устройство A может выключать V_{BUS} при остановке сессии USB.

28.5.2. Состояния устройств

Запитанное

Обнаружение V_{BUS} при B-Сессии делает состояние Запитанным. OTG_FS автоматически подключает резистор подпорки сигнала DP полноскоростного хоста и выдаёт запрос прерывания сессии (бит SRQINT в регистре OTG_FS_GINTSTS).

Также проверяется рабочий уровень V_{BUS} от хоста. Если в B-сессии V_{BUS} падает, то OTG_FS автоматически отключается и выдаёт прерывание выхода из запитанного режима (бит SEDET в регистре OTG_FS_GOTGINT).

В запитанном режиме OTG_FS ждёт сигнала сброса от хоста, и всё. При появлении сброса (USBRST в OTG_FS_GINTSTS) выдаётся прерывание. По завершении сброса выдаётся прерывание конца переучёта (ENUMDNE в OTG_FS_GINTSTS) и OTG_FS идёт в состояние по умолчанию.

Программное отключение

Установка бита SDIS в регистре OTG_FS_DCTL отключает резистор подпорки DP и хост выдаёт прерывание отключения даже при реально воткнутом в порт хоста кабеле USB.

По умолчанию

В этом режиме OTG_FS ждёт от хоста команду SET_ADDRESS. По ней в поле DAD регистра OTG_FS_DCFG пишется адрес устройства, OTG_FS становится адресованным и может отвечать на запросы хоста.

Остановленное

Периферийный OTG_FS постоянно отслеживает активность USB. После 3 ms простоя USB выдаётся прерывание раннего останова (бит ESUSP в OTG_FS_GINTSTS) и через 3 ms подтверждается прерыванием останова (бит USBSUSP в OTG_FS_GINTSTS). Автоматически ставится бит состояния (SUSPSTS в OTG_FS_DSTS) и OTG_FS стопорится.

Выйти из останова можно самостоятельно установкой бита удалённой побудки (RWUSIG в OTG_FS_DCTL) и его снятием после задержки от 1 до 15 ms.

При появлении сигнала побудки от хоста выдаётся прерывание (бит WKUPINT в регистре OTG_FS_GINTSTS) и автоматически снимается бит останова устройства.

28.5.3. Конечные точки устройств

Ядро OTG_FS обслуживает следующие конечные точки USB:

• Управляющая точка 0:

— Двунаправленная, только для управляющих сообщений.

— Отдельные наборы регистров для входных и выходных передач.

— Собственные регистры управления (OTG_FS_DIEPCTL0/OTG_FS_DOEPCTL0), конфигурации передач (OTG_FS_DIEPTSIZ0/OTG_FS_DIEPTSIZ0) и состояния/прерываний (OTG_FS_DIEPINT0/OTG_FS_DOEPINT0) с несколько отличными от других точек битами.

• 3 конечных точки IN

— Поддерживают изохронные, групповые и прерывающие передачи.

— Собственные регистры управления (OTG_FS_DIEPCTL*x*), конфигурации передач (OTG_FS_DIEPTSIZ*x*) и состояния/прерываний (OTG_FS_DIEPINT*x*).

— Регистр маски общих прерываний (включая EP0) Устройства IN (OTG_FS_DIEPMSK).

— Поддержка прерывания незавершённых изохронных передач IN (бит IISOIXFR в регистре OTG_FS_GINTSTS). Оно выставляется во время прерывания конца периодического фрейма (OTG_FS_GINTSTS/EOPF).

• 3 конечных точки OUT

— Поддерживают изохронные, групповые и прерывающие передачи.

— Собственные регистры управления (OTG_FS_DOEPCTL*x*), конфигурации передач (OTG_FS_DOEPTSIZ*x*) и состояния/прерываний (OTG_FS_DOEPINT*x*).

— Регистр маски общих прерываний (включая EP0) Устройства Out (OTG_FS_DOEPMSK).

— Поддержка прерывания незавершённых изохронных передач OUT (бит INCOMPISOOUT в OTG_FS_GINTSTS). Оно выставляется во время прерывания конца периодического фрейма (OTG_FS_GINTSTS/EOPF).

Управление конечной точкой

• Через регистры управления (DIEPCTLx/DOEPCTLx) можно:

— Включить/выключить точку

— Активировать точку в текущей конфигурации

— Установить тип передачи USB (изохронная, групповая, прерывающая)

— Задать размер пакета

— Задать номер Tx-FIFO для точки IN

— Задать ожидаемый или передаваемый DATA0/DATA1 PID (только групповые/прерывающие)

— Задать чётный/нечётный фрейм изохронных передач

— Установкой бита NAK всегда выдавать хосту не-ответ независимо от состояния FIFO

— Установкой бита STALL всегда стопорить токены хоста этой точке

— Установить режим SNOOP точки OUT, чтобы не проверяла CRC принятых данных.

Передачи конечной точки

Размер и состояние передачи хранятся в регистрах DIEPTSIZx/DOEPTSIZx. Писать в него нужно до запуска конечной точки. После запуска точки регистр доступен только по чтению, пишет в него OTG FS.

Можно установить параметры:

- Размер передачи в байтах
- Число пакетов всей передачи

Состояние/прерывания конечной точки

События со стороны USB и AHB хранятся в регистрах прерываний (DIEPINTx/DOPEPINTx). Читать их надо при стоящих битах прерываний (бит OEPINT в OTG_FS_GINTSTS для точки OUT и бит IEPINT в OTG_FS_GINTSTS для точки IN). Перед чтением этих регистров надо получить номер прервавшей конечной точки из общего регистра прерываний (OTG_FS_DAINT). Снятие бита в этих регистрах чистит соответствующие биты регистров DAINT и GINTSTS.

Ядро периферии проверяет и выдаёт прерывания для:

- Завершение передачи для сторон AHB и USB
- Фаза SETUP завершена (только управляющие OUT)
- FIFO передачи наполовину или полностью пуст (точки IN)
- Хосту передан ответ NAK (только изохронные IN)
- Принят токен IN при пустом Tx-FIFO (только групповые IN и прерывающие IN)
- Принят токен OUT для ещё не включённой точки
- Ошибка перекрёстного шума
- Программа выключила точку
- Программа включила NAK точки (только изохронные IN)
- Принята череда более 3 пакетов SETUP (только управляющая OUT)
- Таймаут (только управляющие IN)
- Отброшен изохронный пакет OUT без выдачи прерывания.

28.6. Хост USB

OTG_FS работает USB хостом как:

• OTG A-хост

– состояние A-устройства по умолчанию когда вставлена сторона A кабеля USB

• OTG B-host

– B-устройство после HNP переключения в режим хоста

• A-устройство

– Если линия ID есть, работает и подключена к A-стороне кабеля USB, и снят бит HNPCAP в регистре OTG_FS_GUSBCFG. К линиям DP/DM автоматически подключаются встроенные резисторы подтяжки.

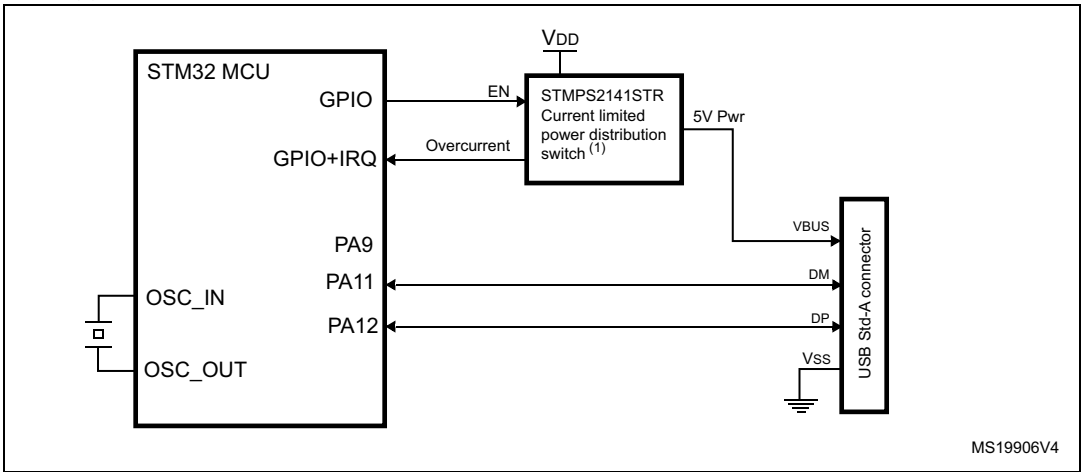
• Только хост

– Включается битом FHMOD в регистре OTG_FS_GUSBCFG. В этом случае линия ID игнорируется даже при её наличии в разъёме USB. К линиям DP/DM автоматически подключаются встроенные резисторы подтяжки.

NB: Встроенный источник 5V V_{BUS} не поддерживается, на линию надо подавать внешнее напряжение. Внешнюю подзарядку можно подавать через выход GPIO. Это требуется для конфигураций OTG A-хост, A-устройства и только-хост.

Уровень V_{BUS} при работе USB проверяется на входе V_{BUS}. Сигнал перегрузки источника подзарядки V_{BUS} можно подавать на вход GPIO для выдачи прерывания. Прерывание перегрузки должно немедленно выключать V_{BUS}.

Подключение только-хост USB



1. STMPS2141STR нужна только при питании устройств от V_{BUS}. При наличии 5V питания на плате можно использовать перемычку.

28.6.1. Хост с SRP

Поддержка SRP включается битом SRPCAP в регистре OTG_FS_GUSBCFG. Теперь можно выключать питание V_{BUS} при остановке сессии USB.

28.6.2. Состояния хоста USB

Питание порта хоста

Встроенное питание -5V V_{BUS} не поддерживается. Стало быть нужно использовать внешнее. Внешнюю подзарядку можно подавать через выходы GPIO. При питании V_{BUS} от GPIO надо ставить бит PPWR в регистре OTG_FS_HPRT.

V_{BUS} в норме

При включённом HNP или SRP ножку контроля V_{BUS} (PA9) надо подключать к V_{BUS}. Непредвиденное падение напряжения V_{BUS} ниже порога (4.25 V) запускается прерывание конца сессии (бит SEDET в OTG_FS_GOTGINT). Затем надо отключить V_{BUS} и снять бит питания порта.

При выключенных HNP SRP ножку контроля V_{BUS} (PA9) к V_{BUS} подключать не надо.

Сигнал перегрузки источника подзарядки V_{BUS} можно подавать на вход GPIO для выдачи прерывания. Прерывание перегрузки должно немедленно выключать V_{BUS} и снять бит питания порта.

Обнаружение хостом подключения периферии

При включённых SRP или HNP ядро OTG_FS обнаруживает подключение периферии USB или устройства B только при рабочем уровне питания (V_{BUS} выше 4.75 V). При подключении внешнего устройства B ядро OTG_FS выдаёт прерывание установкой бита PCDET в регистре OTG_FS_HPRT.

При выключенных HNP и SRP периферия USB и устройства B обнаруживаются сразу после подключения. Ядро OTG_FS выдаёт прерывание установкой бита PCDET в регистре OTG_FS_HPRT.

Обнаружение хостом отключения периферии

Прерывание выдаётся установкой бита DISCINT в регистре OTG_FS_GINTSTS.

Переучёт хоста

После обнаружения подключённого устройства хост должен начать переучёт, посылая новому устройству сброс USB и команды конфигурации.

Перед посылкой сброса USB надо дождаться прерывания конца антидребезга (бит DBCDNE в OTG_FS_GOTGINT), появляющегося при подключении резисторов к DP (FS) или DM (LS).

Сигнал сброса USB подаётся снятием бита PRST в регистре OTG_FS_HPRT минимум на 10 ms и максимум на 20 ms. Считать это время надо осторожно.

После сброса USB выдаётся прерывание изменения состояния включения (бит PENCHNG в OTG_FS_HPRT). Теперь можно узнать скорость подключённого устройства (бит PSPD в OTG_FS_HPRT) и начать выдавать SOF (FS) или Живи (Keep alive) (LS) и посылать команды конфигурации.

Остановка хоста

Активность USB останавливают установкой бита PSUSP в регистре OTG_FS_HPRT. OTG_FS прекращает посылку SOFs и идёт в режим останова. Выйти из него можно по инициативе устройства. При этом выдаётся прерывание побудки (бит WKUPINT в регистре OTG_FS_GINTSTS), автоматически ставится бит восстановления (бит PRES в OTG_FS_HPRT) и на USB выдаётся сигнал восстановления. Через некоторое время бит восстановления снимается и и возобновляется выдача SOF.

При побудке по инициативе хоста автоматически ставится бит восстановления через некоторое время он снимается и и возобновляется выдача SOF.

28.6.3. Каналы хоста

Ядро OTG_FS обслуживает 8 каналов хоста. Хост не может одновременно поддерживать больше 8 запросов передач. Если их больше, то контроллер хоста (HCD) должен перераспределить каналы после их освобождения завершением передачи.

Канал хоста поддерживает IN/OUT и любой тип периодических/не-периодических передач. Канал имеет собственные регистры управления (HCCHAR*x*), конфигурации (HCTSIZ*x*) состояния/прерываний (HCINT*x*) со связанным регистром маски (HCINTMSK*x*).

Управление каналом хоста

- Регистр HCCHAR*x* позволяет:

— Включать/выключать канал.

— Ставить скорость FS/LS целевой периферии USB

— Задавать адрес целевой периферии USB

— Задавать номер конечной точки целевой периферии USB

— Задавать направление передач IN/OUT

— Задавать тип передач USB (управляющая, групповая, прерывающая, изохронная)

— Задавать максимальный размер пакета (MPS)

— Задавать периодические передачи для выполнения во время чётных/нечётных фреймов

Передачи канала хоста

Длина и состояние передач содержатся в регистре HCTSIZ*x*. Писать в него надо до запуска канала. После запуска регистр доступен только по чтению.

- Можно задавать:

— размер передачи в байтах

— число пакетов всей передачи

— начальный PID данных

Состояние/прерывания канала хоста

События со стороны USB и AHB отмечаются в регистре HCINT*x*. Читать его нужно при стоящем бите прерывания (бит HCINT в регистре OTG_FS_GINTSTS). Перед этим нужно прочесть номер прервавшего канала из регистра HCAINT. Снимаются биты прерывания очисткой соответствующих битов в регистрах HAINT и GINTSTS. Маска прерываний есть в регистре OTG_FS_HCINTMSK*x*.

- Ядро хоста проверяет следующие события прерываний:

— Завершение передачи на сторонах AHB и USB

— Остановка канала по концу передачи, ошибке передачи USB или команде выключения

— Передающий FIFO наполовину или полностью пуст (точки IN)

— Принят ответ ACK

— Принят ответ NAK

— Принят ответ STALL

— Ошибка передачи USB по ошибке CRC, таймауту, битовому заполнению, сбойному EOP

— Ошибка перекрёстных шумов

— Переполнение фрейма

— Ошибка переключения данных

28.6.4. Планировщик хоста

По началу фрейма хост сначала выполняет одновременно затребованные периодические (изохронные и прерывающие) передачи, а затем не-периодические (управляющие и групповые). Это делается с помощью очередей запросов (по одной для каждого вида передач). Каждая очередь имеет до 8 элементов. Элемент очереди содержит номер IN или OUT канала и информацию к его обработке через USB. Порядок запросов в очереди определяет последовательность пересылок по USB.

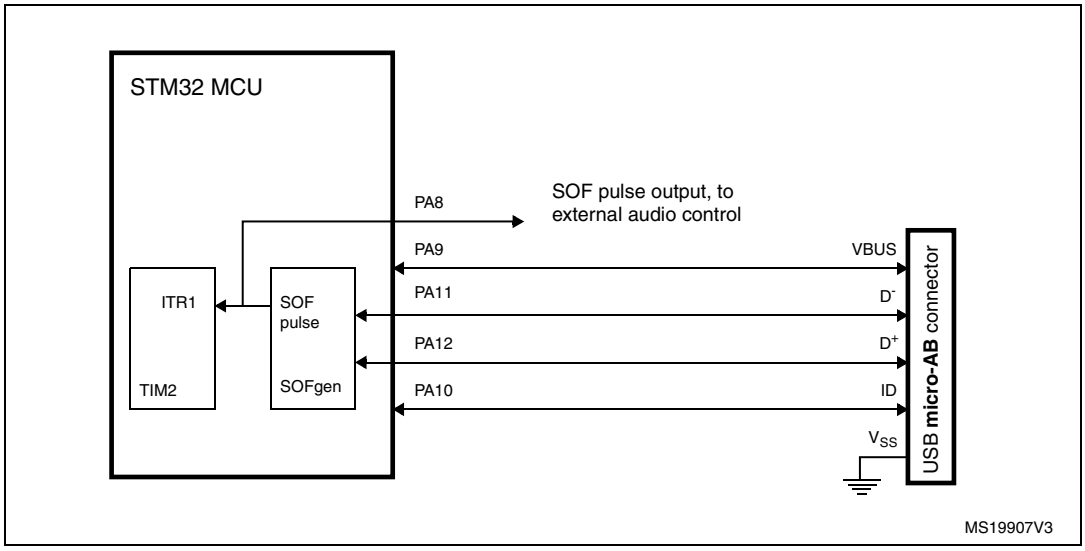
Если для текущего фрейма ещё запланирована изохронная или прерывающая передача, то в его конце выдаётся прерывание незавершённой периодической передачи (IPXFR в OTG_FS_GINTSTS). Управляет очередями запросов только ядро OTG HS. Регистры FIFO и состояния периодической (HPTXSTS) и не-периодической (HNPTXSTS) очередями доступны только по чтению. Они содержат:

- Число свободных строк (до 8)
- Доступное место в Tx-FIFO (передачи OUT)
- Токен IN/OUT, номер канала и состояние.

Перед отправкой запроса в очередь нужно убедиться в наличии там свободного места проверив биты PTXQSAV в регистре OTG_FS_HNPTXSTS или NPTQXSAV в регистре OTG_FS_HNPTXSTS.

28.7. Выдача SOF

Подключение SOF



Ядро OTG FS помогает отслеживать и конфигурировать размещение SOF в хосте и периферии, равно как и выдачу импульсов SOF. Это полезно при подключении звуковой периферии для синхронизации изохронного потока между устройством и PC.

28.7.1. SOF хоста

В режиме хоста число тактов PHY между двумя последовательными токенами SOF (FS) или Keep-alive (LS) пишутся в регистр интервала фреймов (HFIR). Прерывание выдаётся по каждому началу фрейма (бит SOF в OTH_FS_GINTSTS). Номер текущего фрейма и остаток времени до следующего отслеживается в регистре HFNUM.

Выдаваемый по токену SOF импульс, длительностью 12 системных тактов можно направлять на выходную ножку SOF с помощью бита SOFOUTEN в регистре OTG_FS_GCCFG. Также импульс SOF внутренне подключён к входу запуска TIM2.

28.7.2. SOF устройства

В режиме устройства при получении токена SOF (бит SOF в OTH_FS_GINTSTS) выдаётся прерывание. Номер фрейма можно почитать в регистре состояния устройства (FNSOF в регистре OTG_FS_DSTS). Выдаваемый по токену SOF импульс, длительностью 12 системных тактов можно направлять на выходную ножку SOF с помощью бита SOFOUTEN в регистре OTG_FS_GCCFG. Также импульс SOF внутренне подключён к входу запуска TIM2.

Время выдачи прерывания конца периодического фрейма (GINTSTS/EOPF) определено как 80%, 85%, 90% или 95% ожидаемого времени фрейма в регистре конфигурации устройства (PFIVL в OTG_FS_DCFG). Так можно определить конец всей изохронной передачи.

28.8. Варианты питания

Потребление энергии ядром OTG PHY определяется тремя битами регистра конфигурации ядра:

- Выключение PHY (GCCFG/PWRDWN)
- Включение компараторов V_{BUS} устройства A (GCCFG/VBUSASEN).
- Включение компараторов V_{BUS} устройства B (GCCFG/VBUSASEN)
- Остановка тактов PHY (бит STPPCLK в OTG_FS_PCGCCTL).
- Запрет HCLK (GATEHCLK в OTG_FS_PCGCCTL).
- Стоп системы USB.

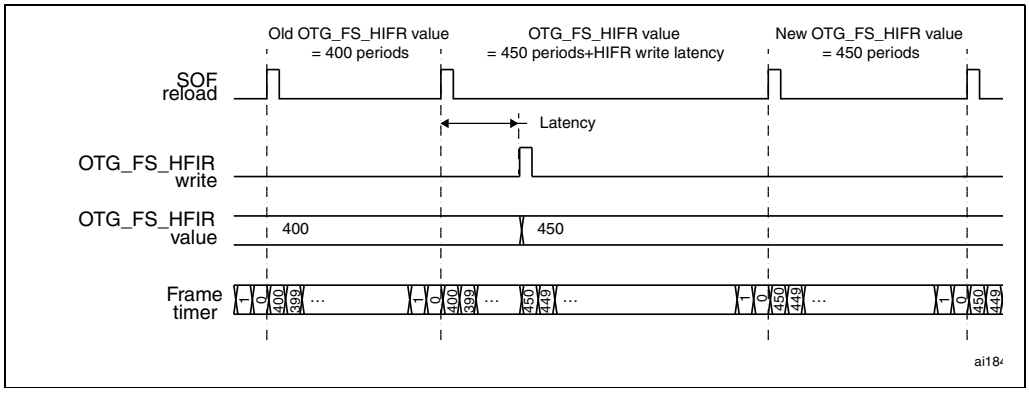
Во благо экономии, FIFO данных USB тактируются только при доступе из ядра OTG_FS.

28.9. Динамическое обновление регистра OTG_FS_HFIR

Ядро USB в режиме хоста может подстраивать время выдачи фреймов микро-SOF для синхронизации внешнего устройства.

Если регистр OTG_HS_HFIR изменился внутри текущего фрейма микро-SOF, то изменение периода SOF вступит в силу в следующем фрейме.

Динамическое обновление OTG_FS_HFIR

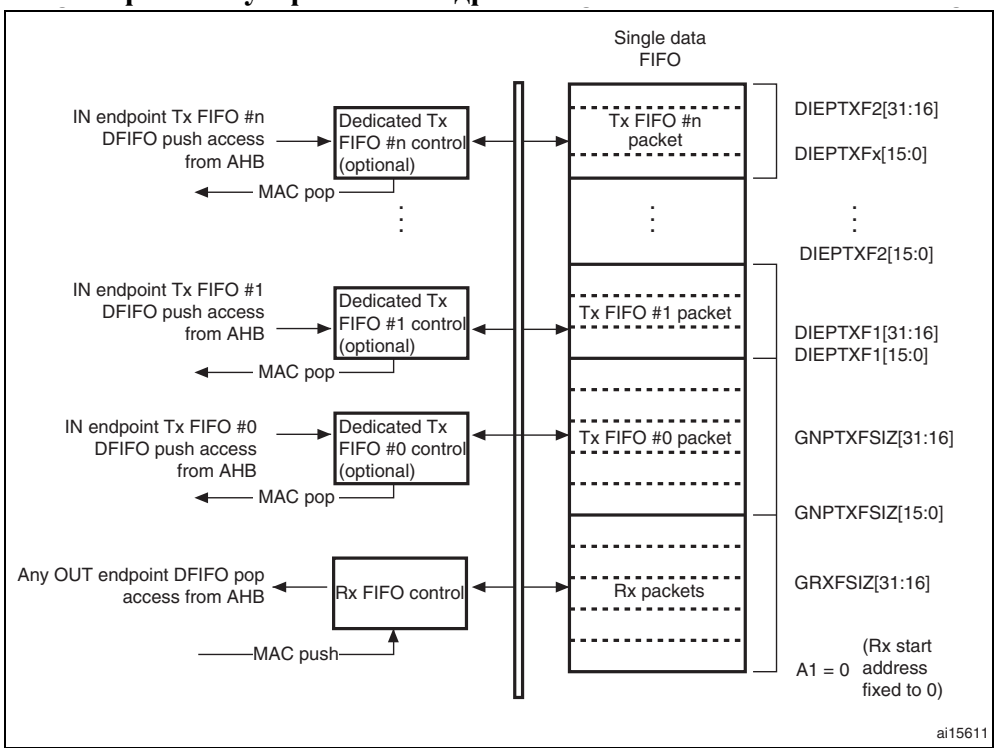


28.10. FIFO данных USB

Система USB имеет 1.25 КБ выделенного RAM для одного Rx-FIFO и нескольких Tx-FIFO. Число и использование FIFO зависит от режима устройства. В режиме периферии каждой активной точке IN выделяется дополнительный Tx-FIFO. Размер FIFO определяется программно.

28.11. Архитектура FIFO устройства

FIFO режима устройства и адреса AHB



28.11.1. Rx FIFO устройства

Он принимает данные в доступное место Rx-FIFO от всех точек OUT устройства. Состояние принятого (номер точки OUT, счётчик байт, PID данных и их достоверность) хранится вместе с данными. При исчерпании места в FIFO передача от хоста получает NACK и выдаётся прерывание адресованной конечной точки. Размер приёмного FIFO задаётся в регистре GRXFSIZ.

Выгоды одного FIFO приёма:

- Все точки OUT используют один буфер RAM (общий FIFO)
- Токены OUT могут поступать в любой последовательности

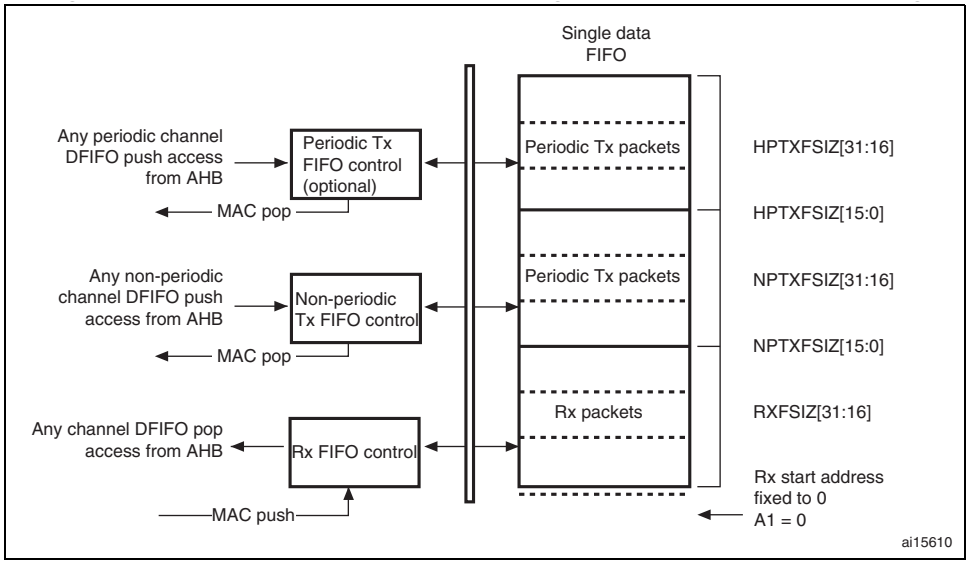
Если Rx-FIFO не пуст (бит RXFLVL в OTG_FS_GINTSTS) то программа получает прерывания. Информация пакета есть в регистре состояния пакета (GRXSTSP), данные читают по соответствующему адресу FIFO.

28.11.2. Tx FIFO устройства

Для каждой точки IN выделяется свой FIFO. Размеры FIFO пишутся в регистры размера FIFO, (OTG_FS_TX0FSIZ) для точки IN-0 и (DIEPTXFx) для точек IN-x.

28.12. Архитектура FIFO хоста

FIFO режима хоста и адреса AHB



28.12.1. Rx FIFO хоста

Он принимает данные в доступное место Rx-FIFO все периодические и не-периодические передачи от всех точек IN устройства. Состояние принятого (номер точки OUT, счётчик байт, PID данных и их достоверность) хранится вместе с данными. Размер приёмного FIFO задаётся в регистре GRXFSIZ.

Выгоды одного FIFO приёма:

- Все точки OUT используют один буфер RAM (общий FIFO)
- Запись любой последовательности токенов IN определяется программой хоста

Если Rx-FIFO не пуст, то программа получает прерывания. Информация пакета есть в регистре состояния пакета, данные читают по соответствующему адресу FIFO.

28.12.2. Tx FIFO хоста

Для периодических (изохронных и прерывающих) и не-периодических (управляющих и групповых) передач OUT используются отдельные FIFO. В них хранятся передаваемые данные. Размер периодического/не-периодического Tx FIFO определяется каждый в своём регистре (HPTXFSIZ/HNPTXFSIZ).

В начале фрейма сначала обрабатывается очередь периодических Tx FIFO, затем не-периодических.

Передающие FIFO работают так:

- Для периодических (не-периодических) каналов OUT используется один буфер RAM
- Последовательность заполнения передающего FIFO определяется программой

Выдача прерываний полупустого или пустого периодического Tx FIFO (PTXFE в OTG_FS_GINTSTS) задаётся в регистре конфигурации AHB (PTXFELVL в OTG_FS_GAHBCFG). Запись передаваемых данных возможна только при наличии свободного места в Tx FIFO и очереди запросов (регистр HPTXSTS).

Выдача прерываний полупустого или пустого не-периодического Tx FIFO (NPTXFE в регистре OTG_FS_GINTSTS) задаётся в регистре конфигурации AHB (TXFELVL в OTG_FS_GAHBCFG). Запись передаваемых данных возможна только при наличии свободного места в Tx FIFO и очереди запросов (регистр HNPTXSTS).

28.13. Размещение FIFO в RAM

28.13.1. Режим устройства

Память для приёмного FIFO: надо выделить память для 10 пакетов SETUP, ядро к этой памяти не суётся. Один участок выделяется для Глобального OUT NAK. Состояние принятого пакета пишется вместе с данными.

Поэтому для приёма пакетов надо выделять минимум (Максимальный_ Размер_Пакета/ 4) + 1. При нескольких изохронных точках надо выделить не меньше двух таких участков. Обычно выделяют два таких участка.

Вместе с последним пакетом передачи конечной точки в FIFO пихается состояние завершения передачи. Для неё нужен один участок на точку OUT.

Память для передающего FIFO: минимальный размер FIFO равен максимальному размеру пакета точки IN.

NB: Больший размер FIFO точки IN повышает производительность USB.

28.13.2. Режим хоста

Память для приёмного FIFO

Состояние принятого пакета пишется вместе с данными.

Поэтому для приёма пакетов надо выделять минимум (Максимальный_ Размер_Пакета/ 4) + 1. При нескольких изохронных точках надо выделить не меньше двух таких участков. Обычно выделяют два таких участка.

Вместе с последним пакетом передачи конечной точки в FIFO пихается состояние завершения передачи. Для неё нужен один участок на канал.

Память для передающего FIFO

Минимальный размер не-периодического передающего FIFO равен самому большому размеру пакетов не-периодических каналов OUT. Обычно выделяют два максимальных размера пакета.

Минимальный размер периодического передающего FIFO равен самому большому размеру пакетов периодических каналов OUT. Если есть изохронные точки OUT, то максимальный размер этого канала надо удвоить.

NB: Больший размер не-периодического передающего FIFO повышает производительность USB.

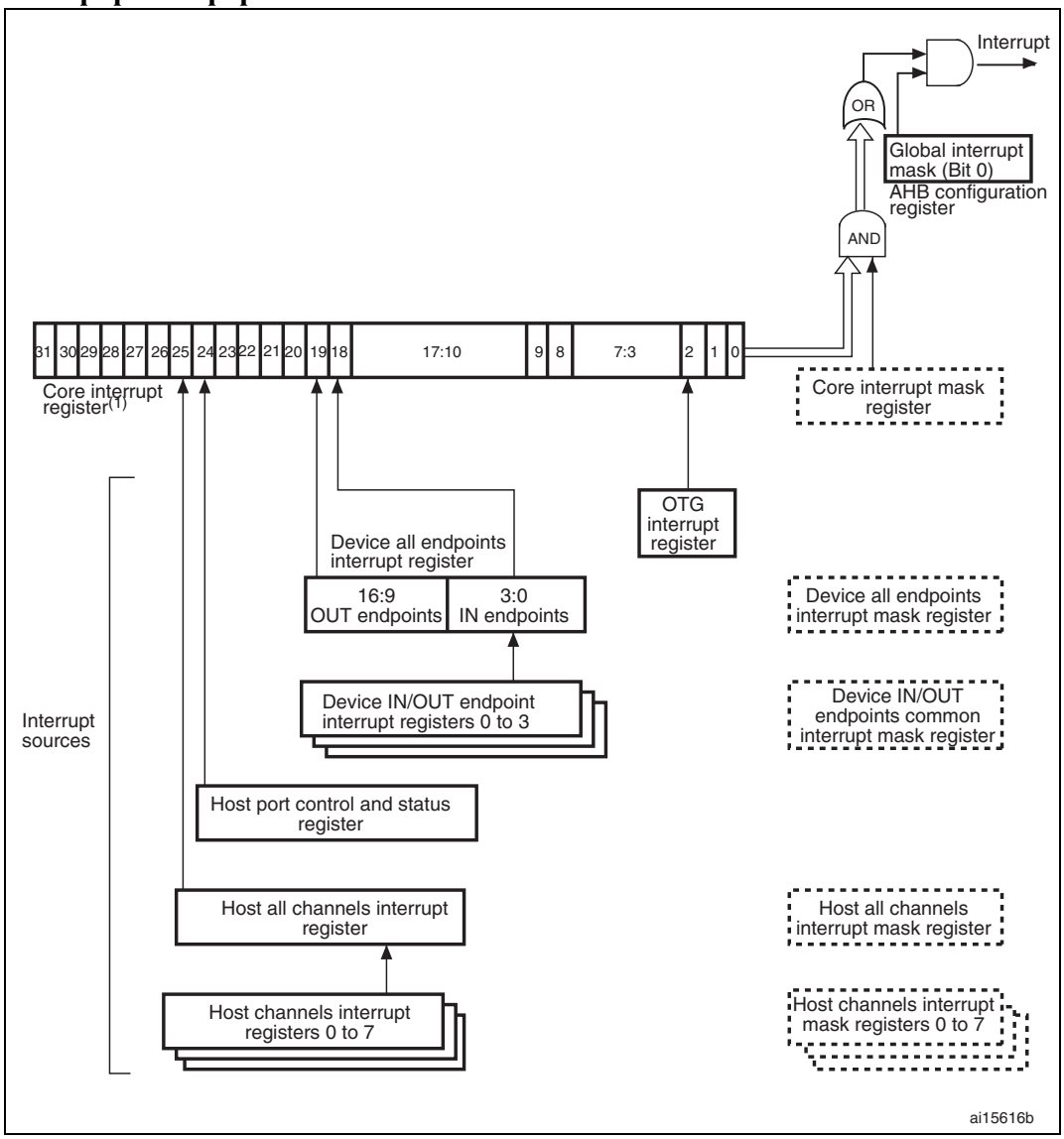
28.14. Производительность системы с USB

Наилучшая производительность USB и системы достигается благодаря большим буферам RAM, гибким размерам FIFO, быстрому 32-бит доступу к FIFO через регистры AHB и развитому механизму управления FIFO. Можно достичь максимальной скорости USB. Качества таковы:

- Программе очень удобно:
  - —Накапливать много данных для передачи по USB
  - —Собирать много данных в приёмный FIFO
- Ядру USB удобно:
  - —Аппаратно планировать передачу большого числа данных по USB
  - —Иметь много места для размещения данных из USB

28.15.Прерывания OTG_FS

Иерархия прерываний



28.16. Регистры управления и состояния OTG_FS

Контроллер OTG_FS управляется через регистры состояния и управления (CSR) по шине AHB. Это 32-бит регистры, выравненные на границу 32-бит, доступные 32-бит словами.

CSR разделяются на:

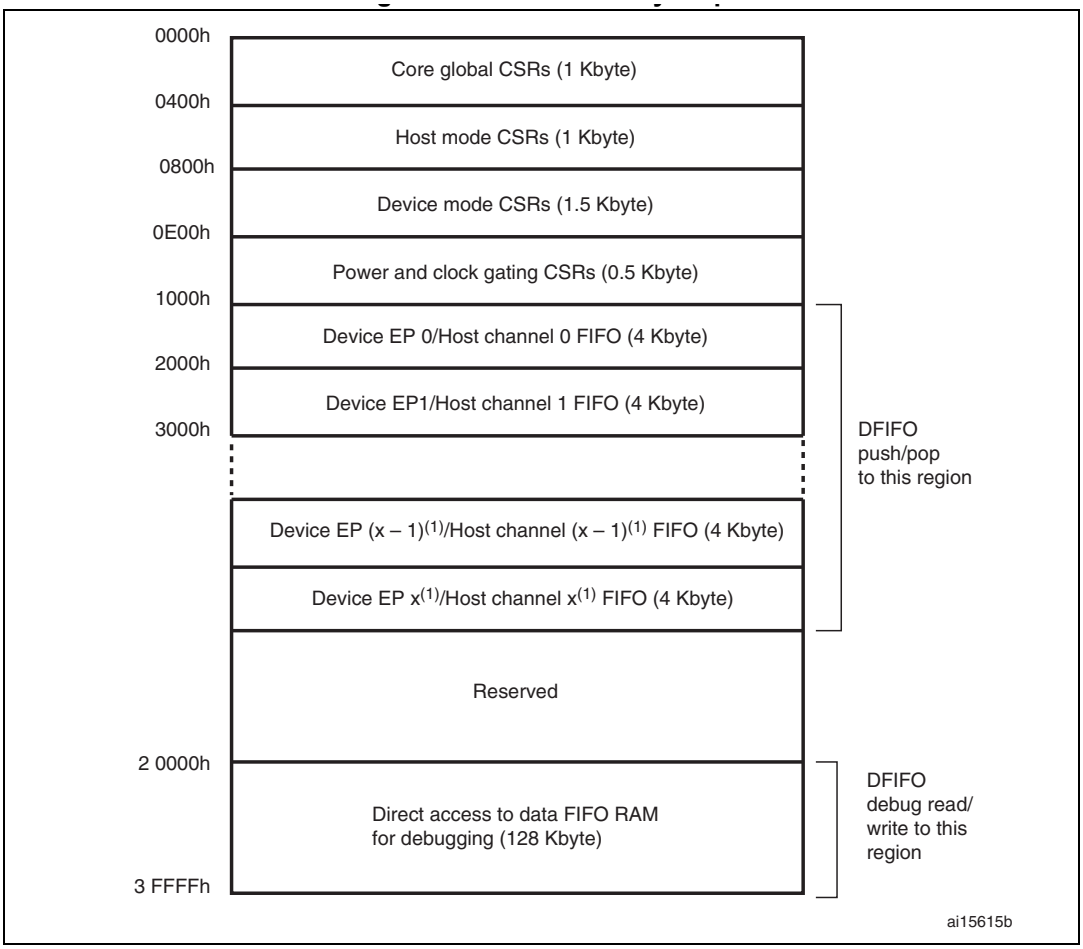
- Общие регистры ядра
- Регистры режима хоста
- Общие регистры хоста
- CSR портов хоста
- Регистры каналов хоста
- Регистры режима устройства
- Общие регистры устройства
- Регистры конечных точек устройства
- Регистры управления питанием и тактами
- Регистры доступа к FIFO данных (DFIFO)

Одновременно в режимах хоста и ядра доступны только Общие регистры ядра, Регистры управления питанием и тактами, Регистры доступа к FIFO данных и CSR портов хоста. Работая в одном режиме, контроллер OTG_FS не должен обращаться к регистрам другого режима. Это вызовет прерывание несовпадения режима (бит `MMIS` в регистре `OTG_FS_GINTSTS`). При переключении режимов регистры нужно переписать как при выходе из сброса.

28.16.1. Карта памяти CSR

Регистры режимов хоста и устройства занимают разные адреса. Все регистры сделаны в домене тактов AHB.

Карта памяти CSR



1. x=3 у устройства и x=7 у хоста.

Карта общих CSR

Эти регистры доступны в режимах хоста и устройства.

Таблица 199. Общие CSR

Имя	Смещение адреса	Имя регистра
OTG_FS_GOTGCTL	0x000	Регистр управления и состояния OTG_FS (OTG_FS_GOTGCTL)
OTG_FS_GOTGINT	0x004	Регистр прерываний OTG_FS (OTG_FS_GOTGINT)
OTG_FS_GAHBCFG	0x008	Регистр конфигурации AHB OTG_FS (OTG_FS_GAHBCFG)
OTG_FS_GUSBCFG	0x00C	Регистр конфигурации USB OTG_FS (OTG_FS_GUSBCFG)
OTG_FS_GRSTCTL	0x010	Регистр сброса OTG_FS (OTG_FS_GRSTCTL)
OTG_FS_GINTSTS	0x014	Регистр прерываний ядра OTG_FS (OTG_FS_GINTSTS)
OTG_FS_GINTMSK	0x018	Регистр маски прерываний OTG_FS (OTG_FS_GINTMSK)
OTG_FS_GRXSTSR	0x01C	Регистр состояния приёма отладки OTG_FS/ Регистр состояния и чтения OTG (OTG_FS_GRXSTSR/OTG_FS_GRXSTSP)
OTG_FS_GRXSTSP	0x020	
OTG_FS_GRXFSIZ	0x024	Регистр размера приёмного FIFO OTG_FS (OTG_FS_GRXFSIZ)
OTG_FS_HNPTXFSIZ/ OTG_FS_DIEPTXF0(1)	0x028	Регистр размера FIFO не-периодической передачи хоста OTG_FS (OTG_FS_HNPTXFSIZ)/размера FIFO передачи конечной точки 0 (OTG_FS_DIEPTXF0)

Имя	Смещение адреса	Имя регистра
OTG_FS_HNPTXSTS	0x02C	Регистр FIFO не-периодической передачи /состояния очереди OTG_FS (OTG_FS_HNPTXSTS)
OTG_FS_GCCFG	0x038	Общий регистр конфигурации ядра OTG_FS (OTG_FS_GCCFG)
OTG_FS_CID	0x03C	Регистр ID ядра OTG_FS (OTG_FS_CID)
OTG_FS_HPTXFSIZ	0x100	Регистр размера FIFO периодических передач OTG_FS (OTG_FS_HPTXFSIZ)
OTG_FS_DIEPTXFx	0x104 0x124 ... 0x138	Регистр размера FIFO передач конечной точки IN устройства OTG_FS (OTG_FS_DIEPTXFx) (x = 1..3, где x это номер FIFO)

1. Общее правило: OTG_FS_HNPTXFSIZ в режиме хоста, OTG_FS_DIEPTXF0 в режиме устройства.

Карта CSR хоста

Эти регистры писать при каждом входе в режим хоста.

Таблица 200. CSR режима хоста

Имя	Смещение адреса	Имя регистра
OTG_FS_HCFG	0x400	Регистр конфигурации хоста OTG_FS (OTG_FS_HCFG)
OTG_FS_HFIR	0x404	Регистр интервала фрейма хоста OTG_FS (OTG_FS_HFIR)
OTG_FS_HFNUM	0x408	Регистр номера фрейма/остатка времени хоста OTG_FS (OTG_FS_HFNUM)
OTG_FS_HPTXSTS	0x410	Регистр FIFO периодических передач/состояния очереди хоста OTG_FS (OTG_FS_HPTXSTS)
OTG_FS_HAINT	0x414	Регистр прерываний всех каналов хоста OTG_FS (OTG_FS_HAINT)
OTG_FS_HAINTMSK	0x418	Регистр маски прерываний всех каналов хоста OTG_FS (OTG_FS_HAINTMSK)
OTG_FS_HPRT	0x440	Регистр управления и состояния хоста OTG_FS (OTG_FS_HPRT)
OTG_FS_HCCHARx	0x500 0x520 ... 0x6E0	Регистр характеристик канала x хоста OTG_FS (OTG_FS_HCCHARx) (x = 0..7, где x = Номер канала)
OTG_FS_HCINTx	0x508	Регистр прерываний канала x хоста OTG_FS(OTG_FS_HCINTx) (x = 0..7, где x = Номер канала)
OTG_FS_HCINTMSKx	0x50C	Регистр маски прерываний канала x хоста OTG_FS Host (OTG_FS_HCINTMSKx) (x = 0..7, где x = Номер канала)
OTG_FS_HCTSIZx	0x510	Регистр размера передачи канала x хоста OTG_FS (OTG_FS_HCTSIZx) (x = 0..7, где x = Номер канала)

Карта CSR устройства

Эти регистры писать при каждом входе в режим устройства.

Таблица 201. CSR режима устройства

Имя	Смещение адреса	Имя регистра
OTG_FS_DCFG	0x800	Регистр конфигурации устройства OTG_FS (OTG_FS_DCFG)
OTG_FS_DCTL	0x804	Регистр управления устройства OTG_FS (OTG_FS_DCTL)
OTG_FS_DSTS	0x808	Регистр состояния устройства OTG_FS (OTG_FS_DSTS)
OTG_FS_DIEPMSK	0x810	Общий регистр маски прерывания конечных точек IN устройства OTG_FS (OTG_FS_DIEPMSK)
OTG_FS_DOEPMSK	0x814	Общий регистр маски прерывания конечных точек OUT устройства OTG_FS (OTG_FS_DOEPMSK)
OTG_FS_DAINT	0x818	Регистр прерываний всех конечных точек устройства OTG_FS (OTG_FS_DAINT)

Имя	Смещение адреса	Имя регистра
OTG_FS_DAINTMSK	0x81C	Регистр маски прерываний всех конечных точек устройства OTG_FS (OTG_FS_DAINTMSK)
OTG_FS_DVBUSDIS	0x828	Регистр времени разряда V_{BUS} устройства OTG_FS (OTG_FS_DVBUSDIS)
OTG_FS_DVBUSPULSE	0x82C	Регистр времени импульса V_{BUS} устройства OTG_FS (OTG_FS_DVBUSPULSE)
OTG_FS_DIEPEMPMSK	0x834	Регистр маски пустого FIFO конечных точек IN устройства OTG_FS(OTG_FS_DIEPEMPMSK)
OTG_FS_DIEPCTL0	0x900	Регистр управления IN конечной точки 0 устройства OTG_FS (OTG_FS_DIEPCTL0)
OTG_FS_DIEPCTLx	0x920 0x940 ... 0xAE0	Регистр управления конечной точки x устройства OTG (OTG_FS_DIEPCTLx) (x = 1..3, где x = Номер конечной точки)
OTG_FS_DIEPINTx	0x908	Регистр прерываний конечной точки x устройства OTG_FS (OTG_FS_DIEPINTx) (x = 0..3, где x = Номер конечной точки)
OTG_FS_DIEPTSIZ0	0x910	Регистр размера передачи IN конечной точки 0 OTG_FS (OTG_FS_DIEPTSIZ0)
OTG_FS_DTXFSTSx	0x918	Регистр состояния FIFO передачи конечной точки IN OTG_FS (OTG_FS_DTXFSTSx) (x = 0..3, где x = Номер конечной точки)
OTG_FS_DIEPTSIZx	0x930 0x950 ... 0xAF0	Регистр размера передачи OUT конечной точки x устройства OTG_FS (OTG_FS_DOEPTSIZx) (x = 1..3, где x = Номер конечной точки)
OTG_FS_DOEPCTL0	0xB00	Регистр управления OUT конечной точки 0 устройства OTG_FS (OTG_FS_DOEPCTL0)
OTG_FS_DOEPCTLx	0xB20 0xB40 ... 0xCC0 0xCE0 0xCFD	Регистр OTG управления конечной точки x устройства (OTG_FS_DIEPCTLx) (x = 1..3, где x = Номер конечной точки)
OTG_FS_DOEPINTx	0xB08	Регистр прерываний конечной точки x устройства OTG_FS (OTG_FS_DOEPINTx) (x = 0..3, где x = Номер конечной точки)
OTG_FS_DOEPTSIZx	0xB10	Регистр размера передачи OUT конечной точки x устройства OTG_FS (OTG_FS_DOEPTSIZx) (x = 1..3, где x = Номер конечной точки)

Карта регистров доступа к FIFO данных (DFIFO)

Регистры доступны в обоих режимах хоста и устройства. У каналов хоста IN, FIFO доступен по чтению. У каналов хоста OUT, FIFO доступен по записи.

Таблица 202. Карта регистров доступа к FIFO данных (DFIFO)

Секция регистров доступа к FIFO	Диапазон адресов	Доступ
Device IN Endpoint 0/Host OUT Channel 0: DFIFO Write Access Device OUT Endpoint 0/Host IN Channel 0: DFIFO Read Access	0x1000–0x1FFC	w r
Device IN Endpoint 1/Host OUT Channel 1: DFIFO Write Access Device OUT Endpoint 1/Host IN Channel 1: DFIFO Read Access	0x2000–0x2FFC	w r
...	...	...
Device IN Endpoint x(1)/Host OUT Channel x(1): DFIFO Write Access Device OUT Endpoint x(1)/Host IN Channel x(1): DFIFO Read Access	0xX000–0xXFFC	w r

1. x=3 у устройства и x=7 у хоста.

Карта CSR питания и тактов

Это отдельный регистр, доступный в режимах хоста и устройства.

Таблица 203. Регистр управления питанием и тактами

Название	Имя	Смещение адреса: 0xE00–0xFFF
Power and clock gating control register	PCGCR	0xE00-0xE04
Reserved		0xE05–0xFFF

28.16.2. Общие регистры OTG_FS

Доступны в режимах хоста и устройства, при переключении режимов переписывать не надо.

Регистр управления и состояния (OTG_FS_GOTGCTL)

Смещение адреса: 0x000

По сбросу: 0x0000 0800

31–20	19	18	17	16	15–12	11	10	9	8	7–2	1	0
Reserved	BSVLD	ASVLD	DBCT	CIDSTS	Reserved	DHNPEN	HSHNPEN	HNPRQ	HNGSCS	Reserved	SRQ	SRQSCS
	r	r	r	r		rw	rw	rw	r		rw	r

— Биты 31:20 Резерв, не трогать.

— Бит 19 **BSVLD:** B-сессия в норме

0: B-сессия сбойная.

1: B-сессия достойная.

В режиме OTG можно использовать для определения факта подключения устройства.

NB: Доступен только в режиме устройства.

— Бит 18 **ASVLD:** A-сессия в норме

0: A-сессия сбойная

1: A-сессия достойная

NB: Доступен только в режиме хоста

— Бит 17 **DBCT:** Длинное/короткое время антидребезга обнаруженного подключения

0: Длинное, физическое подключение (100 ms + 2.5 µs)

1: Короткое, программное подключение (2.5 µs)

NB: Доступен только в режиме хоста

— Бит 16 **CIDSTS**: Состояние ID подключения

0: Контроллер OTG_FS в режиме A-устройства

1: Контроллер OTG_FS в режиме B-устройства

NB: Доступен в обоих режимах.

— Биты 15:12 Резерв, не трогать.

— Бит 11 **DHNPEN**: Разрешение HNP устройства

Успешно принята команда **SetFeature.SetHNPEnable** от хоста USB.

0: HNP не разрешён

1: HNP разрешён

NB: Доступен только в режиме устройства.

— Бит 10 **HSHNPEN**: Хост установил HNP устройству

Программно ставится после разрешения HNP на устройстве (командой **SetFeature.SetHNPEnable**).

0: Неудача

1: Успех

NB: Доступен только в режиме хоста

— Бит 9 **HNPRQ**: Выдача запроса HNP

Ставится программно. Программно можно снять записью 0 после установки бита HNSSCHG в регистре OTG_FS_GOTGINT. Ядро снимает этот бит после очистки бита HNSSCHG.

0: Запроса HNP нет

1: Запрос HNP

NB: Доступен только в режиме устройства.

— Бит 8 **HNGSCS**: Успешная беседа с Хостом

Ставится ядром при успешной беседе с хостом. Снимается ядром при стоящем запросе HNP (HNPRQ).

0: Не договорились

1: Успешно побеседовали

NB: Доступен только в режиме устройства.

— Биты 7:2 Резерв, не трогать.

— Бит 1 **SRQ**: Программный запрос сессии на USB
Ставится программой. Программно может сниматься записью 0 при изменении состояния беседы с хостом (стоит бит HNSSCHG в регистре OTG_FS_GOTGINT). Ядро снимает этот бит при чистом бите HNSSCHG. При использовании полноскоростного USB 1.1 перед запросом сессии надо дождаться разрядки V_{BUS} до 0.2 V после снятия бита BSVLD в OTG_FS_GOTGCTL. это время приведено в документации производителя.
0: Ничего не просили
1: Просим сессию
NB: Доступен только в режиме устройства.

— Бит 0 **SRQSCS**: Успешный запрос сессии
Ставится ядром.
0: Сбойный
1: Успех
NB: Доступен только в режиме устройства.

Регистр прерываний (OTG_FS_GOTGINT)

Смещение адреса: 0x004

По сбросу: 0x0000 0000

Флаги ставятся при прерываниях OTG, снимаются записью 1 в нужный бит.

31 … 20	19	18	17	16 … 10	9	8	7 … 3	2	1	0
Reserved	DBCDNE	ADTOCHG	HNGDET	Reserved	HNSSCHG	SRSSCHG	Reserved	SEDET	Res.	
	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1		rc_w1		

— Биты 31:20 Резерв, не трогать.

— Бит 19 **DBCDNE**: Антидребезг завершён
Ставится ядром после подключения устройства. Теперь можно выдавать сброс USB. Бит работает только при стоящем бите HNPCAP или SRPCAP в регистре OTG_FS_GUSBCFG.
NB: Доступен только в режиме хоста

— Бит 18 **ADTOCHG**: Таймаут A-устройства
Ставится ядром после таймаута ожидания подключения B-устройства.
NB: Доступен в обоих режимах.

— Бит 17 **HNGDET**: Начало беседы хоста
Ставится ядром после обнаружения запроса беседы с хостом на USB.
NB: Доступен в обоих режимах.

— Биты 16:10 Резерв, не трогать.

— Бит 9 **HNSSCHG**: Состояние беседы хоста изменилось
Ставится ядром после изменения состояния бита HNGSCS в регистре OTG_FS_GOTGCTL.
NB: Доступен в обоих режимах.

— Бит 8 **SRSSCHG**: Состояние запроса сессии изменилось
Ставится ядром после изменения состояния бита SRQSCS в регистре OTG_FS_GOTGCTL.
NB: Доступен в обоих режимах.

— Биты 7:3 Резерв, не трогать.

— Бит 2 **SEDET**: Конец сессии
Ставится ядром после падения $V_{BUS} < 0.8$ V. Конец сессии B-устройства.

— Биты 1:0 Резерв, не трогать.

Регистр конфигурации AHB (OTG_FS_GAHBCFG)

Смещение адреса: 0x008

По сбросу: 0x0000 0000

Пишется после включения питания или смены режима до запуска любой передачи по AHB или USB. Параметры относятся в основном к AHB. После начальной записи изменять его нельзя.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							PTXFELVL	TXFELVL	Reserved						GINTMSK
																							rw	rw							rw

— Биты 31:9 Резерв, не трогать.

— Бит 8 **PTXFELVL**: Уровень пустоты Периодического TxFIFO

Выдача прерывания пустого периодического TxFIFO (бит PTXFE в OTG_FS_GINTSTS) при:

0: Полупустом периодическом TxFIFO

1: Совсем пустом периодическом TxFIFO

NB: Доступен только в режиме хоста

— Бит 7 **TXFELVL**: Уровень пустоты TxFIFO не-периодического/точек IN

В режиме устройства: выдача прерывания пустого TxFIFO точек IN (TXFE в OTG_FS_DIEPINTx) при:

0: Полупустом TxFIFO

1: Совсем пустом TxFIFO

В режиме хоста: Выдача прерывания пустого не-периодического TxFIFO (бит NPTXFE в регистре OTG_FS_GINTSTS) при:

0: Полупустом не-периодическом TxFIFO

1: Совсем пустом не-периодическом TxFIFO

— Биты 31:9 Резерв, не трогать.

— Бит 0 **GINTMSK**: Маска Общего прерывания

Маска собранных воедино источников прерываний. Регистры состояния прерываний всё равно изменяются независимо от этого бита.

0: Прерывать нельзя.

1: Прерывать можно.

NB: Доступен в обоих режимах.

Регистр конфигурации USB (OTG_FS_GUSBCFG)

Смещение адреса: 0x00C

По сбросу: 0x0000 0A00

Пишется после включения питания или смены режима до запуска любой передачи по AHB или USB. Параметры относятся к USB и USB-PHY. После начальной записи изменять его нельзя.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTXPKT	FDMOD	FHMOD	Reserved															TRDT				HNPCAP	SRPCAP	Res.	PHYSEL	Reserved				TOCAL		
rw	rw	rw																rw				r/rw	r/rw		wo					rw		

— Бит 31 **CTXPKT**: Нарушенный пакет Tx

Только для отладки, не ставьте этот бит ни за что и никогда.

NB: Доступен в обоих режимах.

— Бит 30 **FDMOD**: Принудительный режим устройства

Запись 1 переключает в режим устройства независимо от входной ножки ID.

0: Нормальный режим

1: Принудительный режим устройства

Изменения наступают через 25 ms после установки этого бита.

NB: Доступен в обоих режимах.

— Бит 29 **FHMOD**: Принудительный режим хоста

Запись 1 переключает в режим хоста независимо от входной ножки ID.

0: Нормальный режим

1: Принудительный режим хоста

Изменения наступают через 25 ms после установки этого бита.

NB: Доступен в обоих режимах.

— Биты 28:14 Резерв, не трогать.

— Биты 13:10 **TRDT**: Время обратной связи USB в тактах PHY

Они ставятся в соответствии с Таблицей 204, в зависимости от частоты AHB.

NB: Доступен только в режиме устройства.

— Бит 9 **HNPCAP**: Разрешение HNP

0: HNP запрещён.

1: HNP Разрешён.

NB: Доступен в обоих режимах.

— Бит 8 **SRPCAP**: Разрешение SRP

0: SRP запрещён.

1: SRP Разрешён.

NB: Доступен в обоих режимах.

— Бит 7 Резерв, не трогать.

— Бит 6 **PHYSEL**: Выбор полноскоростного трансивера

Всегда 1, только запись.

— Биты 5:3 Резерв, не трогать.

— Биты 2:0 **TOCAL**: Калибровка таймаута FS

Число тактов PHY, добавляемых к стандартной задержке (от 16 до 18 времён бита) между передачами пакетов для компенсации работы PHY от разных производителей. Определяется на основе скорости переучёта. На каждый такт PHY добавляется 0.25 времени бита.

Таблица 204. Значения TRDT

Диапазон частот AHB (MHz)		Мин. значение TRDT
Min.	Max	
14.2	15	0xF
15	16	0xE
16	17.2	0xD
17.2	18.5	0xC
18.5	20	0xB
20	21.8	0xA
21.8	24	0x9
24	27.5	0x8
27.5	32	0x7
32	—	0x6

Регистр сброса (OTG_FS_GRSTCTL)

Смещение адреса: 0x010

По сбросу: 0x0000 0A00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHBIDL	Reserved																				TXFNUM					TXFFLSH	RXFFLSH	Reserved	FCRST	HSRST	CSRST
r																					rw					rs	rs		rs	rs	rs

— Бит 31 **AHBIDL**: Простой ведущего AHB

Машина состояний ведущего AHB в Простое.

NB: Доступен в обоих режимах.

— Биты 30:11 Резерв, не трогать.

— Биты 10:6 **TXFNUM**: Номер сливаемого TxFIFO

FIFO сливается установкой TxFIFO Flush. Пока бит TxFIFO Flush стоит, изменять это поле нельзя.

00000:

– Не-периодический TxFIFO в режиме хоста

– TxFIFO 0 в режиме устройства

00001:
– Периодический TxFIFO в режиме хоста
– TxFIFO 1 в режиме устройства
00010: TxFIFO 2 в режиме устройства
...
00101: TxFIFO 15 в режиме устройства
10000: Все передающие FIFO в обоих режимах.
NB: Доступен в обоих режимах.

— Бит 5 **TXFFLSH**: Слив TxFIFO
Команда на слив одного или всех FIFO, но не в середине передач с FIFO.
Отсутствие передач проверяется так:
Для чтения из TxFIFO—Было прерывание "NAK в действии"
Для записи в TxFIFO—Стоит бит AHBIDL в регистре OTG_FS_GRSTCTL.
NB: Доступен в обоих режимах.

— Бит 4 **RXFFLSH**: Слив RxFIFO
Команда на слив всего RxFIFO, но не в середине передач с RxFIFO.
Перед записью в этот бит надо убедиться, что RxFIFO ни читается ни пишется.
Перед запуском других операций надо дождаться снятия этого бита. Это требует 8 тактов (самый медленный из PHY и AHB).
NB: Доступен в обоих режимах.

— Бит 3 Резерв, не трогать.

— Бит 2 **FCRST**: Сброс счётчика фреймов хоста
Следующий посылаемый SOF будет иметь номер фрейма 0.
NB: Доступен только в режиме хоста.

— Бит 1 **HSRST**: Программный сброс HCLK
Слив управляющей логики конвейеров домена тактов AHB после завершения передач AHB.
Управляющие биты CSR машин состояния домена AHB снимаются.
Маски прерываний машин состояния домена AHB снимаются.
Биты состояния прерываний машин состояния домена AHB не снимаются.
Очистка может занять несколько тактов.
NB: Доступен в обоих режимах.

— Бит 0 **CSRST**: Программный сброс ядра
Сбрасывает домены HCLK и PCLK:
Чистит прерывания и все биты регистров CSR за исключением битов:
– RSTPDMODL в OTG_FS_PCGCCTL
– GAYEHCLK в OTG_FS_PCGCCTL
– PWRCLMP в OTG_FS_PCGCCTL
– STPPCLK в OTG_FS_PCGCCTL
– FSLSPCS в OTG_FS_HCFG
– DSPD в OTG_FS_DCFG
Все машины состояния модуля (кроме блока ведомого AHB) сбрасываются в Простой, передающие и приёмный FIFO сливаются.
Передачи ведущего AHB завершаются после конца последней фазы данных. Передачи по USB завершаются немедленно.
Бит можно писать когда угодно. Снимается сам через несколько тактов. После снятия бита перед доступом к PHY надо подождать не меньше 3 тактов PHY (задержка синхронизации) и дождаться установки бита 31 в этом регистре (Простой ведущего AHB).
Обычно программный сброс используется при разработке программ и динамическом изменении битов выбора PHY, после которого надо сбросить и домен PHY.
NB: Доступен в обоих режимах.

Регистр прерываний ядра (OTG_FS_GINTSTS)

Смещение адреса: 0x014

По сбросу: 0x0400 0020

Некоторые биты работают только в режиме хоста, иные только в режиме устройства. В регистре отмечен текущий режим. Изменяемые биты состояния прерванный снимаются записью 1.

Биты состояния прерываний FIFO только читаются, они изменяются автоматически при обращении к FIFO.

Во избежание излишних прерываний чистить регистр OTG_FS_GINTSTS при инициализации надо до снятия маски запрета прерываний.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WKUINT	SRQINT	DISCINT	CIDSCHG	Reserved	PTXFE	HCINT	HPRTINT	Reserved		IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Reserved		EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Reserved		GONAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
rc_w1					r	r	r	Res.		rc_w1		r	r			rc_w1								r	r	r	r	rc_w1	r	rc_w1	r

— Бит 31 **WKUPINT**: Прерывание удалённой побудки

В режиме устройства ставится при появлении на шине USB события восстановления.

В режиме хоста ставится при появлении на шине USB события удалённой побудки.

NB: Доступен в обоих режимах.

— Бит 30 **SRQINT**: Прерывание запроса/новой сессии

В режиме хоста ставится при появлении на шине USB запроса сессии от устройства.

В режиме устройства ставится при появлении рабочего уровня V_{BUS} для B-устройства.

NB: Доступен в обоих режимах.

— Бит 29 **DISCINT**: Прерывание отключения (вытыкания) внешнего устройства

NB: Доступен в обоих режимах.

— Бит 28 **CIDSCHG**: Состояние линии ID разъёма изменилось

NB: Доступен в обоих режимах.

— Бит 27 Резерв, не трогать.

— Бит 26 **PTXFE**: Периодический TxFIFO пуст

Ставится если в периодическом передающем FIFO есть куда писать. Полу- или полное опустение задаётся битом PTXFELVL в регистре OTG_FS_GAHBCFG.

NB: Доступен только в режиме хоста.

— Бит 25 **HCINT**: Прерывание каналов хоста

В режиме хоста ставится ядром при наличии запроса какого-либо канала хоста. Номер запросившего канала читается из регистра OTG_FS_HAINT, причина запроса читается из регистра OTG_FS_HCINTx. Снимается очисткой бита запроса в регистре OTG_FS_HCINTx.

NB: Доступен только в режиме хоста.

— Бит 24 **HPRTINT**: Прерывание изменения состояния порта Хоста

Событие запроса читается из регистра OTG_FS_HPRT. Снимается очисткой бита запроса в регистре OTG_FS_HPRT.

NB: Доступен только в режиме хоста.

— Биты 23:22 Резерв, не трогать.

— Бит 21 **IPXFR**: Незавершённая периодическая передача

В режиме хоста ядро извещает о незавершённой периодической передаче для текущего фрейма.

INCOMPISOOUT: Незавершённая изохронная передача OUT

В режиме устройства ядро извещает о незавершённой изохронной передаче OUT для текущего фрейма. Выставляется вместе с битом конца периодического фрейма (EOPF) в этом регистре.

— Бит 20 **IISOIXFR**: Незавершённая изохронная передача IN

В режиме устройства ядро извещает о незавершённой изохронной передаче IN для текущего фрейма. Выставляется вместе с битом конца периодического фрейма (EOPF) в этом регистре.

NB: Доступен только в режиме устройства.

— Бит 19 **OEPINT**: Прерывание конечной точки OUT

В режиме устройства ставится ядром при наличии запроса какой-либо конечной точки OUT. Номер запросившей точки читается из регистра OTG_FS_DAINT, причина запроса читается из регистра OTG_FS_DOEPINTx. Снимается очисткой бита запроса в регистре OTG_FS_DOEPINTx.

NB: Доступен только в режиме устройства.

— Бит 18 **IEPINT**: Прерывание конечной точки IN

В режиме устройства ставится ядром при наличии запроса какой-либо конечной точки IN. Номер запросившей точки читается из регистра OTG_FS_DAINT, причина запроса читается из регистра OTG_FS_DIEPINTx. Снимается очисткой бита запроса в регистре OTG_FS_DIEPINTx.

NB: Доступен только в режиме устройства.

— Биты 17:16 Резерв, не трогать.

— Бит 15 **EOPF**: Прерывание конца периодического фрейма

Возникает при истечении интервала периодического фрейма (поле PFIVL в OTG_FS_DCFG) в текущем фрейме.

NB: Доступен только в режиме устройства.

— Бит 14 **ISOODRP**: Прерывание выброса изохронного пакета OUT

Ставится ядром при нехватке места в RxFIFO для изохронного пакета OUT.

NB: Доступен только в режиме устройства.

— Бит 13 **ENUMDNE**: Переучёт завершён

После этого в регистре OTG_FS_DSTS лежит скорость переучёта.

NB: Доступен только в режиме устройства.

— Бит 12 **USBRST**: Сигнал сброса на шине USB

NB: Доступен только в режиме устройства.

— Бит 11 **USBSUSP**: Сигнал Приостановки на шине USB

NB: Доступен только в режиме устройства.

— Бит 10 **ESUSP**: Ранняя приостановка

На шине USB не было активности в течении 3 ms.

NB: Доступен только в режиме устройства.

— Биты 9:8 Резерв, не трогать.

— Бит 7 **GONAKEFF**: Глобальный OUT NAK работает

Бит SGONAK в регистре OTG_FS_DCTL сработал. Очищается установкой бита CGONAK в регистре OTG_FS_DCTL.

NB: Доступен только в режиме устройства.

— Бит 6 **GINAKEFF**: Глобальный NAK не-периодических IN работает

Бит SGINAK в регистре OTG_FS_DCTL сработал. Очищается установкой бита CGINAK в регистре OTG_FS_DCTL.

Это прерывание не означает, что по шине USB будет посылаться NAK, бит STALL старше.

NB: Доступен только в режиме устройства.

— Бит 5 **NPTXFE**: Не-периодический TxFIFO пуст

Ставится если в не-периодическом передающем FIFO есть куда писать. Полу- или полное опустение задаётся битом TXFELVL в регистре OTG_FS_GAHBCFG.

NB: Доступен только в режиме устройства.

— Бит 4 **RXFLVL**: RxFIFO не пуст

Из RxFIFO можно прочесть хоть один пакет.

NB: Доступен в обоих режимах.

— Бит 3 **SOF**: Старт фрейма

В режиме хоста ядро извещает что на шину USB послан SOF (FS) или Keep-Alive (LS). Снимается записью 1 в этот бит.

В режиме устройства ядро извещает, что на USB появился токен. Номер текущего фрейма читают в регистре Состояния устройства. Прерывание проявляется только при работе с FS.

NB: Доступен в обоих режимах.

— Бит 2 **OTGINT**: Прерывание OTG

Ядро извещает о событии протокола OTG. Само событие выявляют по регистру OTG_FS_GOTGINT. Бит снимается очисткой бита события в регистре OTG_FS_GOTGIN.

NB: Доступен в обоих режимах.

— Бит 1 **MMIS**: Прерывание несовпадения режима

Ядро ставит этот бит при попытке доступа:

- К регистрам режима хоста из режима устройства
- К регистрам режима устройства из режима хоста

NB: Доступен в обоих режимах.

— Бит 0 **CMOD**: Текущий режим

0: Режим устройства

1: Режим хоста

NB: Доступен в обоих режимах.

Регистр маски прерываний (OTG_FS_GINTMSK)

Смещение адреса: 0x018

По сбросу: 0x0000 0000

Изменение регистра маски на регистр запросов прерывание (OTG_FS_GINTSTS) не влияет.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUIM	SRQIM	DISCINT	CIDSCHGM	Reserved	PTXFEM	HCIM	PRTIM	Reserved		IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	EPMISM	Reserved	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Reserved		GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Reserved
rw	rw	rw	rw		rw	rw	r			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	

Состояние битов:

0: Нельзя

1: Можно

— Бит 31 **WUIM**: Маска прерывания побудки
NB: Доступен в обоих режимах.

— Бит 30 **SRQIM**: Маска прерывания запроса/новой сессии
NB: Доступен в обоих режимах.

— Бит 29 **DISCINT**: Маска прерывания отключения устройства
NB: Доступен только в режиме устройства.

— Бит 28 **CIDSCHGM**: Маска прерывания изменения ID
NB: Доступен в обоих режимах.

— Бит 27 Резерв, не трогать.

— Бит 26 **PTXFEM**: Маска прерывания пустого периодического TxFIFO
NB: Доступен только в режиме хоста.

— Бит 25 **HCIM**: Маска прерывания каналов хоста
NB: Доступен только в режиме хоста.

— Бит 24 **PRTIM**: Маска прерывания порта хоста
NB: Доступен только в режиме хоста.

— Биты 23:22 Резерв, не трогать.

— Бит 21 **IPXFRM**: Маска прерывания незавершённой периодической передачи
NB: Доступен только в режиме хоста.
IISOOXFRM: Маска прерывания незавершённой изохронной передачи OUT
NB: Доступен только в режиме устройства.

— Бит 20 **IISOIXFRM**: Маска прерывания незавершённой изохронной передачи IN
NB: Доступен только в режиме устройства.

— Бит 19 **OEPINT:** Маска прерывания конечной точки OUT
NB: Доступен только в режиме устройства.

— Бит 18 **IEPINT**: Маска прерывания конечной точки IN
NB: Доступен только в режиме устройства.

— Бит 17 **EPMISM**: Маска прерывания несовпадения конечной точки
NB: Доступен только в режиме устройства.

— Бит 16 Резерв, не трогать.

— Бит 15 **EOPFM**: Маска прерывания конца периодического фрейма
NB: Доступен только в режиме устройства.

— Бит 14 **ISOODRPM**: Маска прерывания выброса изохронного пакета OUT
NB: Доступен только в режиме устройства.

— Бит 13 **ENUMDNEM**: Маска прерывания конца переучёта
NB: Доступен только в режиме устройства.

— Бит 12 **USBRST**: Маска прерывания сброса USB
NB: Доступен только в режиме устройства.

— Бит 11 **USBSUSPM**: Маска прерывания приостановки USB
NB: Доступен только в режиме устройства.
— Бит 10 **ESUSPM**: Маска прерывания ранней приостановки
NB: Доступен только в режиме устройства.
— Биты 9:8 Резерв, не трогать.
— Бит 7 **GONAKEFFM**: Маска прерывания Global OUT NAK effective
NB: Доступен только в режиме устройства.
— Бит 6 **GINAKEFFM**: Маска прерывания Global non-periodic IN NAK effective
NB: Доступен только в режиме устройства.
— Бит 5 **NPTXFEM**: Маска прерывания пустого не-периодического TxFIFO
NB: Доступен только в режиме хоста.
— Бит 4 **RXFLVLM**: Маска прерывания непустого RxFIFO
NB: Доступен в обоих режимах.
— Бит 3 **SOFM**: Маска прерывания старта фрейма
NB: Доступен в обоих режимах.
— Бит 2 **OTGINT**: Маска прерывания OTG
NB: Доступен в обоих режимах.
— Бит 1 **MMISM**: Маска прерывания несовпадения режима
NB: Доступен в обоих режимах.
— Бит 0 Резерв, не трогать.

Регистры Состояния приёма отладки/Состояния приёма и извлечения (OTG_FS_GRXSTSR/OTG_FS_GRXSTSP)

Смещение адреса регистра состояния приёма: 0x01C

Смещение адреса регистра извлечения: 0x020

По сбросу: 0x0000 0000

Чтение Состояния приёма отладки возвращает верхушку приёмного FIFO. Чтение Состояния приёма и извлечения дополнительно извлекает данные верхней строки RxFIFO.

The receive status contents must be interpreted differently in host and device modes. The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the Receive Status FIFO when the Receive FIFO non-empty bit of the Core interrupt register (`RXFLVL` bit in `OTG_FS_GINTSTS`) is asserted.

Режим хоста

31 … 21	20 … 17	16 … 15	14 … 4	3 … 0
Reserved	PKTSTS	DPID	BCNT	CHNUM
	r	r	r	r

— Биты 31:21 Резерв, не трогать.
— Биты 20:17 **PKTSTS**: Статус принятого пакета
0010: Принят пакет данных IN
0011: Передача IN завершена (с прерыванием)
0101: Ошибка переключения данных (с прерыванием)
0111: Канал остановлен (с прерыванием)
Иные: Резерв
— Биты 16:15 **DPID**: PID данных принятого пакета
00: DATA0
10: DATA1
01: DATA2
11: MDATA
— Биты 14:4 **BCNT**: Счётчик байтов принятого пакета IN
— Биты 3:0 **CHNUM**: Номер канала текущего принятого пакета

Режим устройства

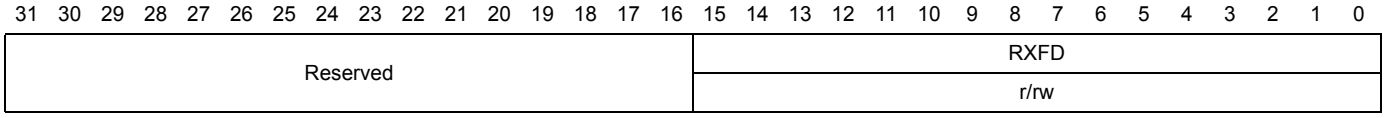
31 … 25	24 … 21	20 … 17	16 … 15	14 … 4	3 … 0
Reserved	FRMNUM	PKTSTS	DPID	BCNT	EPNUM
	r	r	r	r	r

— Биты 31:25 Резерв, не трогать.
— Биты 24:21 **FRMNUM**: Номер фрейма
Младшие 4 бита номера фрейма принятого пакета. Только при поддержке изохронных точек OUT.
— Биты 20:17 **PKTSTS**: Статус принятого пакета
0001: Глобальный OUT NAK (с прерыванием)
0010: Принят пакет данных OUT
0011: Передача OUT завершена (с прерыванием)
0100: Передача SETUP завершена (с прерыванием)
0110: Принят пакет данных SETUP
Others: Reserved
— Биты 16:15 **DPID**: PID данных принятого пакета OUT
00: DATA0
10: DATA1
01: DATA2
11: MDATA
— Биты 14:4 **BCNT**: Счётчик байтов принятого пакета
— Биты 3:0 **EPNUM**: Номер конечной точки текущего принятого пакета

Регистр размера приёмного FIFO (OTG_FS_GRXFSIZ)

Смещение адреса: 0x024

По сбросу: 0x0000 0200

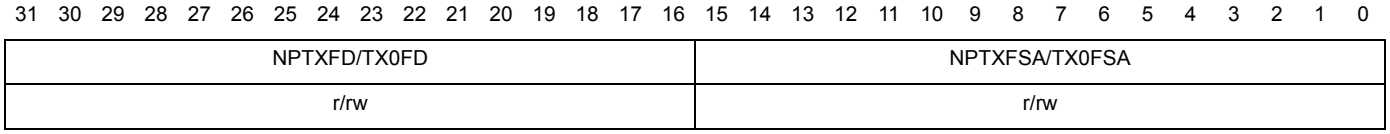


— Биты 31:16 Резерв, не трогать.
— Биты 15:0 **RXFD**: Глубина RxFIFO в 32-бит словах
Минимальное значение 16
Максимальное значение 256
По сбросу питания пишется значение наибольшей глубины Rx FIFO данных.

Регистр размера передающего не-периодического FIFO хоста (OTG_FS_HNPTXFSIZ)
Размер передающего FIFO точки 0 (OTG_FS_DIEPTXF0)

Смещение адреса: 0x028

По сбросу: 0x0000 0200



Режим хоста

— Биты 31:16 **NPTXFD**: Глубина не-периодического TxFIFO в 32-бит словах.
Минимальное значение 16
Максимальное значение 256
— Биты 15:0 **NPTXFSA**: Адрес начала RAM не-периодических передач

Режим устройства

— Биты 31:16 **TX0FD**: Глубина TxFIFO конечной точки 0 в 32-бит словах.
Минимальное значение 16
Максимальное значение 256
— Биты 15:0 **TX0FSA**: Адрес начала RAM передач конечной точки 0

Регистр не-периодического передающего FIFO/состояния очереди (OTG_FS_HNPTXSTS)

Смещение адреса: 0x02C

По сбросу: 0x0008 0200

Содержит информацию о свободном месте в TxFIFO.

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

— Бит 31 Резерв, не трогать.

— Биты 30:24 **NPTXQTOP**: Верх очереди не-периодических запросов передачи

Биты 30:27: Номер Канала/Конечной точки

Биты 26:25: Статус передачи

– 00: Токен IN/OUT

– 01: Пакет нулевой длины (устройство IN/хост OUT)

– 11: Команда остановки канала

Бит 24: Окончание (последняя строка этого Канала/Конечной точки)

— Биты 23:16 **NPTQXSAV**: Доступное место в очереди не-периодических запросов передачи

В режиме хоста очередь хранит запросы и IN и OUT, в режиме устройства только запросы. IN.

00: Очередь запросов не-периодических передач заполнена

01: 1 свободная строка

10: 2 свободных строки

bxn: *n* свободных строк ($0 \le n \le 8$)

Иные: Резерв

— Биты 15:0 **NPTXFSAV**: Доступное место очереди TxFIFO в 32-бит словах

00: Непериодический TxFIFO полон

01: 1 свободное слово

10: 2 свободных слова

0xn: *n* свободных слов ($0 \le n \le 256$)

Иные: Резерв

Общий регистр конфигурации ядра (OTG_FS_GCCFG)

Смещение адреса: 0x038

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											SOFOUTEN	VBUSBSEN	VBUSASEN	Reserved	.PWRDWN	Reserved															
											rw	rw	rw		rw																

— Биты 31:21 Резерв, не трогать.

— Бит 20 **SOFOUTEN**: Разрешение выдачи SOF на PAD

0: Нельзя

1: Можно

— Бит 19 **VBUSBSEN:** Разрешение проверки V_{BUS} устройства “B”

0: Нельзя

1: Можно

— Бит 18 **VBUSASEN:** Разрешение проверки V_{BUS} устройства “A”

0: Нельзя

1: Можно

— Бит 17 Резерв, не трогать.

— Бит 16 **PWRDWN**: Выключение питания трансивера

0: Питание выключено

1: Трансивер включён

— Биты 15:0 Резерв, не трогать.

Регистр ID ядра (OTG_FS_CID)

Смещение адреса: 0x03C

По сбросу: 0x0000 1100

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

— Биты 31:21 **PRODUCT_ID**: Программируемый ID продукта

Регистр размера периодического передающего FIFO хоста (OTG_FS_HPTXFSIZ)

Смещение адреса: 0x100

По сбросу: 0x0200 0600

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

— Биты 31:16 **PTXFD**: Глубина периодического TxFIFO хоста в 32-бит словах
Минимальное значение 16

— Биты 15:0 **PTXSA**: Адрес начала периодического TxFIFO хоста
По сбросу питания содержит сумму наибольших глубин Rx FIFO данных и не-периодического Tx FIFO данных.

Регистр размера передающего FIFO конечной точки (OTG_FS_DIEPTXFx) (x = 1..3, где x это номер FIFO)

Смещение адреса: 0x104 + (FIFO_number – 1) × 0x04

По сбросу: 0x0200 0400

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

— Биты 31:16 **INEPTXFD**: Глубина TxFIFO конечной точки N в 32-бит словах
Минимальное значение 16
По сбросу питания это значение определено как наибольшая глубина FIFO точки IN.

— Биты 15:0 **INEPTXSA**: Адрес начала FIFOx передачи конечной точки IN
Должен быть выравнен на границу 32-бит слова.

28.16.3. Регистры режима хоста

В режиме устройства их содержимое не определено.

Регистр конфигурации хоста (OTG_FS_HCFG)

Смещение адреса: 0x400

По сбросу: 0x0000 0000

Конфигурация ядра по сбросу питания. После инициализации хоста изменять нельзя.

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

— Биты 31:16 Резерв, не трогать.

— Бит 2 **FSLSS**: Поддержка только FS и LS
Определяет скорость переучёта. Пишется единожды.
1: Только FS/LS, даже для устройств с поддержкой HS (только чтение)

— Биты 1:0 **FSLSPCS**: Выбор тактов PHY FS/LS

В режиме FS

01: PHY на частоте 48 MHz
Иные: Резерв

В режиме LS

00: Резерв

01: PHY на частоте 48 MHz

10: PHY на частоте 6 MHz

11: Резерв

NB: Ставить надо после события подключения устройства в соответствии с его частотой (после изменения бита надо делать программный сброс).

Регистр интервала фреймов хоста (OTG_FS_HFIR)

Смещение адреса: 0x404

По сбросу: 0x0000 EA60

Значение интервала фреймов для текущей скорости переучёта.

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

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **FRIVL**: Интервал фреймов

Программное определение интервала между двумя последовательными токенами SOF (FS) или Keep-Alive (LS) в тактах PHY. Писать можно только после установки бита PENA в регистре OTG_FS_HPRT. Если не задано, то ядро вычисляет его по значению поля FSLSPCS в регистре OTG_FS_HCFG. Писать надо единожды при начальной конфигурации.

1 ms × (частоту тактов PHY)

Регистр номера/остатка времени фрейма хоста (OTG_FS_HFNUM)

Смещение адреса: 0x408

По сбросу: 0x0000 3FFF

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

— Биты 31:16 **FTREM**: Остаток времени фрейма в тактах PHY

Декрементируется по каждому такту PHY. Перегружается из регистра интервала при достижении 0 и передаче нового SOF.

— Биты 15:0 **FRNUM**: Номер фрейма

Инкрементируется по новому SOF, обнуляется при достижении 0x3FFF.

Регистр состояния периодических передающего FIFO/очереди (OTG_FS_HPTXSTS)

Смещение адреса: 0x410

По сбросу: 0x0008 0100

Только чтение. Информация о свободном месте в TxFIFO и очереди периодических передач.

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

— Биты 31:24 **PTXQTOP**: Верх очереди запросов периодических передач

Используется при отладке.

Бит 31: Чётный/Нечётный фрейм

0: Чётный

1: Нечётный

Биты 30:27: Номер Канала/Конечной точки

Биты 26:25: Тип

00: IN/OUT

01: Пакет нулевой длины

11: Команда выключения канала

Бит 24: Последняя строка этого Канала/Конечной точки

— Биты 23:16 **PTXQSAV**: Свободное место в очереди периодических передач
Очередь содержит и IN и OUT запросы.
00: Очередь полна
01: 1 свободная строка
10: 2 свободных строки
bxn: n свободных строк (0 ≤ n ≤ 8)
Others: Резерв
— Биты 15:0 **PTXFSAVL**: Свободное место в TxFIFO периодических передач в 32-би словах
0000: TxFIFO полон
0001: 1 свободное слово
0010: 2 свободных слова
bxn: n свободных слов (0 ≤ n ≤ PTXFD)
Others: Резерв

Регистр всех прерываний хоста (OTG_FS_HAINT)

Смещение адреса: 0x414

По сбросу: 0x0000 0000

Отражает запросы прерываний каналов. Прерывание вызывается по общему биту HCINT в регистре OTG_FS_GINTSTS при разрешении в регистре маски всех каналов. Биты ставятся и снимаются в соответствии с битами регистров прерывания соответствующего канала x.

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	HAINT															
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:24 Резерв, не трогать.
— Биты 31:24 **HAINT**: Прерывания каналов
По одному биту на канал: бит 0 для канала 0, бит 15 для канала 15

Регистр маски всех прерываний хоста (OTG_FS_HAINTMSK)

Смещение адреса: 0x418

По сбросу: 0x0000 0000

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	HAINTM															
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.
— Биты 31:24 **HAINTM**: Маска прерывания каналов
0: Нельзя
1: Можно
По одному биту на канал: бит 0 для канала 0, бит 15 для канала 15

Регистр управления и состояния портов хоста (OTG_FS_HPRT)

Смещение адреса: 0x440

По сбросу: 0x0000 0000

Сейчас хост OTG поддерживает только один порт. Регистр содержит информацию о сбросе USB, включении, остановке, восстановлении, подключении и поверке. Прерывания вызываются по биту HPRTINT в регистре OTG_FS_GINTSTS. Чтение этого регистра снимает бит запроса прерывания. Биты rc_w1 снимаются записью 1.

31–19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PSPD		PTCTL				PPWR	PLSTS		Reserved	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS
	r	r	rw	rw	rw	rw	rw	r	r		rw	rs	rw	rc_w1	r	rc_w1	rc_w0	rc_w1	r

— Биты 31:19 Резерв, не трогать.

— Биты 18:17 **PSPD**: Скорость подключённого к порту устройства
  01: FS
  10: LS
  11: Резерв
— Биты 16:13 **PTCTL**: Тест порта
  0000: Режим теста выключен
  0001: Режим Test_J
  0010: Режим Test_K
  0011: Режим Test_SE0_NAK
  0100: Режим Test_Packet
  0101: Режим Test_Force_Enable
  Others: Резерв
— Бит 12 **PPWR**: Питание порта
  0: Выключено
  1: Включено
— Биты 11:10 **PLSTS**: Текущее состояние линий порта
  Бит 10: Логический уровень OTG_FS_FS_DP
  Бит 11: Логический уровень f OTG_FS_FS_DM
— Бит 9 Резерв, не трогать.
— Бит 8 **PRST**: Сброс порта

Ставится программно, снимается программно после завершения последовательности сброса.
  0: Сбросу нету
  1: Сброс есть

Ставится на период не менее 10 ms перед запуском сброса порта и ещё ждём 10 ms перед снятием этого бита. В стандарте USB этого нет.

— Бит 7 **PSUSP**: Приостановка порта

Ставится программно. Ядро просто перестаёт посылать SOF. Для остановки тактов PHY надо ставить свой бит остановки. При чтении выдаёт текущее состояние порта.

Снимается ядром после установки битов WKUINT или DISCINT в регистре OTG_FS_GINTSTS.
  0: Порт работает
  1: Порт стоит

— Бит 6 **PRES**: Восстановление порта

Программно подаёт на порт сигнал восстановления. При программном снятии бита снимается и сигнал. При обнаружении побудки (бит WKUINT в OTG_FS_GINTSTS) ядро выдаёт сигнал и снимает его после обнаружении отключения. Чтение выдаёт текущее состояние сигнала.
  0: Сигнала восстановления нет
  1: Сигнал есть

— Бит 5 **POCCHNG**: Сигнал перегрузки порта по току изменился
— Бит 4 **POCA**: Перегрузка порта по току
  0: Нету
  1: Есть
— Бит 3 **PENCHNG**: Включение порта изменилось
— Бит 2 **PENA**: Включение порта

Порт включается ядром в конце последовательности сброса, выключается при перегрузке, отключении или программном снятии этого бита. Программно бит не ставится. Прерывания не вызывает.
  0: Выключен
  1: Включён

— Бит 1 **PCDET**: Порт подключён

Ставится ядром при подключении устройства, прерывание вызывается битом HPRTINT в регистре OTG_FS_GINTSTS). Снимается записью 1.

— Бит 0 **PCSTS**: Состояние подключения порта
  0: Устройства нет
  1: Устройство есть

Регистр характеристик канала-x (OTG_FS_HCCHARx)
(x = 0..7, где x = Номер_канала)

Смещение адреса: 0x500 + (Номер_канала × 0x20)

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **CHENA**: Включение канала

Ставится программно, снимается хостом OTG.

0: Выключен

1: Включён

— Бит 30 **CHDIS**: Выключение канала

Программная установка бита останавливает приём/передачу данных не дожидаясь их завершения. Далее надо дождаться подтверждения выключения прерыванием.

— Бит 29 **ODDFRM**: Нечётный фрейм

Программно задаёт передачу чётного/нечётного фрейма периодической (изохронной или прерывающей) передачи.

0: Чётный фрейм

1: Нечётный фрейм

— Биты 28:22 **DAD**: Адрес устройства

— Биты 21:20 **MCNT**: Многосчётчик

Число передач для этого фрейма периодической конечной точки. Для не-периодических точек не используется.

00: Резерв. Результат непредсказуем

01: 1 передача

10: 2 передачи на фрейм

11: 3 передачи на фрейм

— Биты 19:18 **EPTYP**: Тип конечной точки

00: Управляющая

01: Изохронная

10: Групповая

11: Прерывающая

— Бит 17 **LSDEV**: Низкоскоростное устройство

— Бит 16 Резерв. не трогать.

— Бит 15 **EPDIR**: Направление конечной точки

0: OUT

1: IN

— Биты 14:11 **EPNUM**: Номер конечной точки

— Биты 10:0 **MPSIZ**: Максимальный размер пакета

Регистр прерываний канала-x (OTG_FS_HCINTx)
(x = 0..7, где x = Номер_канала)

Смещение адреса: 0x508 + (Номер_канала × 0x20)

По сбросу: 0x0000 0000

Читать регистр надо при стоящем бите HCINT в регистре OTG_FS_GINTSTS, но сначала надо узнать номер прервавшего канала из регистра OTG_FS_HAINT. Снятие бита в этом регистре чистит соответствующие биты в регистрах OTG_FS_HAINT и OTG_FS_GINTSTS.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
																					rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

— Биты 31:11 Резерв. не трогать.

— Бит 10 **DTERR**: Ошибка переключения данных
— Бит 9 **FRMOR**: Переполнение фрейма
— Бит 8 **BBERR**: Ошибка перекрёстных шумов
— Бит 7 **TXERR**: Ошибка передачи
Сбой CRC
Таймаут
Ошибка бита заполнения
Дрянной EOP
— Бит 6 Резерв. не трогать.
— Бит 5 **ACK**: Прерывание принятого/переданного ACK
— Бит 4 **NAK**: Прерывание принятого ответа NAK
— Бит 3 **STALL**: Прерывание принятого ответа STALL
— Бит 2 Резерв. не трогать.
— Бит 1 **CHH**: Канал остановлен
Indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application.
— Бит 0 **XFRC**: Передача завершена без ошибок

Регистр маски прерываний канала-x (OTG_FS_HCINTMSKx) (x = 0..7, где x = Номер_канала)

Смещение адреса: 0x508 + (Номер_канала × 0x20)

По сбросу: 0x0000 0000

31 … 11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Маска прерывания события: 0- нельзя, 1 - можно.
— Биты 31:11 Резерв. не трогать.
— Бит 10 **DTERRM**: Ошибка переключения данных
— Бит 9 **FRMORM**: Переполнение фрейма
— Бит 8 **BBERRM**: Ошибка перекрытых шумов
— Бит 7 **TXERRM**: Ошибка передачи
— Бит 6 **NYET**: Ответ принят
— Бит 5 **ACKM**: Прерывание принятого/переданного ACK
— Бит 4 **NAKM**: Прерывание принятого ответа NAK
— Бит 3 **STALLM**: Прерывание принятого ответа STALL
— Бит 2 Резерв. не трогать.
— Бит 1 **CHHM**: Канал остановлен
— Бит 0 **XFRCM**: Передача завершена

Регистр размера передачи канала канала-x (OTG_FS_HCTSIZx) (x = 0..7, (x = 0..7, где x = Номер_канала)

Смещение адреса: 0x510 + (Номер_канала × 0x20)

По сбросу: 0x0000 0000

31	30 … 29	28 … 19	18 … 0
Reserved	DPID	PKTCNT	XFRSIZ
rw	rw rw	rw rw rw rw rw rw rw rw rw rw	rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw

— Бит 31 Резерв. не трогать.
— Биты 30:29 **DPID:** PID данных начальной передачи
Программно задаёт PID данных начальной передачи. Для остальных передач управляется хостом.
00: DATA0
01: DATA2

10: DATA1

11: MDATA (non-control)/SETUP (control)

— Биты 28:19 **PKTCNT**: Счётчик пакетов

Программно пишется ожидаемое число передаваемых (OUT) или принимаемых (IN) пакетов передачи.

Декрементируется хостом после каждой успешной передачи пакета OUT/IN. При достижении 0 выдаётся прерывание завершения.

— Биты 18:0 **XFRSIZ**: Размер передачи

Для OUT это число байтов данных передачи хоста.

Для IN это размер приёмного буфера. Должен быть кратен максимальному размеру пакета IN (периодического и не-периодического).

28.16.4. Регистры режима устройства

Регистр конфигурации устройства (OTG_FS_DCFG)

Смещение адреса: 0x800

По сбросу: 0x0220 0000

Конфигурация ядра по сбросу питания, некоторых команд или переучёта. После инициализации регистр изменять нельзя.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			PFIVL		DAD							Reserved	NZLSOHSK		DSPD	
																			rw		rw	rw	rw	rw	rw	rw	rw		rw		rw	rw

— Биты 31:13 Резерв. не трогать.

— Биты 12:11 **PFIVL**: Интервал периодического фрейма

Время выдачи прерывания конца периодического фрейма до полного завершения его передач.

00: 80% интервала фрейма

01: 85% интервала фрейма

10: 90% интервала фрейма

11: 95% интервала фрейма

— Биты 10:4 **DAD**: Адрес устройства

Пишется после каждой команды **SetAddress**.

— Бит 3 Резерв. не трогать.

— Бит 2 **NZLSOHSK**: Ответ на посылку OUT не-нулевой длины фазы статуса

0: Отдать принятый пакет OUT программе (нулевой или не-нулевой длины) и послать ответ на основе битов NAK и STALL конечной точки в регистре управления устройства.

1: На передачу статуса OUT не-нулевой длины ответить STALL, пакет программе не отдавать.

— Биты 1:0 **DSPD**: Скорость устройства

Скорость, показываемая устройством во время переучёта, но реальная скорость определяется хостом USB после завершения "чик-чирик" последовательности.

00: Резерв

01: Резерв

10: Резерв

11: FS (USB 1.1 Частота 48 MHz)

Регистр управления устройства (OTG_FS_DCTL)

Смещение адреса: 0x804

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
																				rw	w	w	w	w	rw	rw	rw	r	r	rw	rw

— Биты 31:12 Резерв. не трогать.

— Бит 11 **POPRGDNE**: Побудка включения завершена
Ставится программно после конфигурации регистра.
— Бит 10 **CGONAK**: Очистка глобального OUT NAK записью 1 в этот бит.
— Бит 9 **SGONAK**: Установка глобального OUT NAK
Выдаёт ответ NAK по всем конечным точкам OUT.
Ставится программно только при снятом бите GONAKEFF в регистре OTG_FS_GINTSTS.
— Бит 8 **CGINAK**: Очистка глобального IN NAK записью 1 в этот бит.
— Бит 7 **SGINAK**: Установка глобального IN NAK
Выдаёт ответ NAK по всем не-периодическим конечным точкам IN.
Ставится программно только при снятом бите GINAKEFF в регистре OTG_FS_GINTSTS.
— Биты 6:4 **TCTL**: Управление тестом
000: Выключен
001: Режим Test_J
010: Режим Test_K
011: Режим Test_SE0_NAK
100: Режим Test_Packet
101: Режим Test_Force_Enable
Иные: Резерв
— Бит 3 **GONSTS**: Состояние глобального OUT NAK
0: Ответ посылается на основе заполнения RxFIFO и состояния битов NAK и STALL.
1: Данные в RxFIFO не пишутся, независимо от заполнения. На все пакеты, кроме SETUP, отсылается NAK. Изохронные пакеты OUT пропадают втуне.
— Бит 2 **GINSTS**: Состояние глобального IN NAK
0: Ответ отсылается на основе наличия данных в передающем FIFO.
1: По всем не-периодическим точкам IN отсылается ответ NAK, независимо от данных в FIFO.
— Бит 1 **SDIS**: Программное отключение
При стоящем бите хост в упор не видит подключённого устройства и устройство не принимает сигналы по USB. При очистке этого бита после программного отключения хосту посылается событие подключения. После переподключения устройства хост начинает его переучёт.
0: Нормальная работа.
1: Ядро выдаёт хосту событие подключения.
— Бит 0 **RWUSIG**: Удалённое пробуждение
Установка бита посылает хосту сигнал удалённой побудки. Очищать бит надо в интервале от 1 ms до 15 ms после установки.

Таблица 205. Минимальная длительность программного отключения

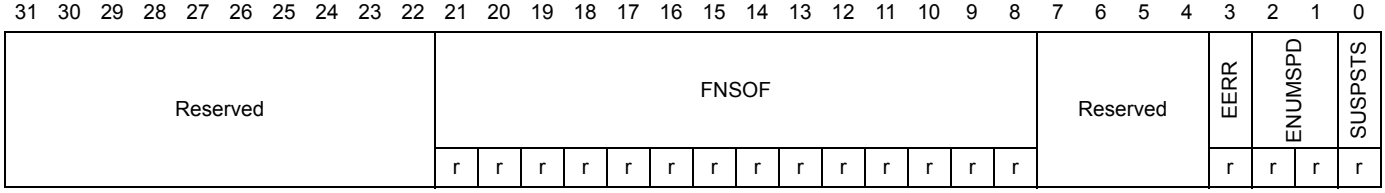
Рабочая скорость	Состояние устройства	Минимальная длительность
FS	Остановка	1 ms + 2.5 µs
FS	Простой	2.5 µs
FS	Работа	2.5 µs

Регистр состояния устройства (OTG_FS_DSTS)

Смещение адреса: 0x808

По сбросу: 0x0000 0010

Состояние событий ядра со стороны USB. Читать надо по прерываниям от регистра OTG_FS_DAINT).



— Биты 31:22 Резерв. не трогать.
— Биты 21:8 **FNSOF**: Номер фрейма принятого SOF

— Биты 7:4 Резерв. не трогать.

— Бит 3 **EERR**: Непредсказуемая ошибка

Ставится ядром. Контроллер OTG_FS уходит в Останов и выдаётся прерывание раннего останова (бит ESUSP в OTG_FS_GINTSTS). Восстановить работу можно только через программное отключение.

— Биты 2:1 **ENUMSPD**: Скорость после переучёта

01: Резерв

10: Резерв

11: FS (частота PHY равна 48 MHz)

Иные: Резерв

— Бит 0 **SUSPSTS**: Состояние приостановки

В режиме устройства отражает сигнал Останова на шине USB. Ядро выходит из Останова при:

– Активности на линиях данных USB

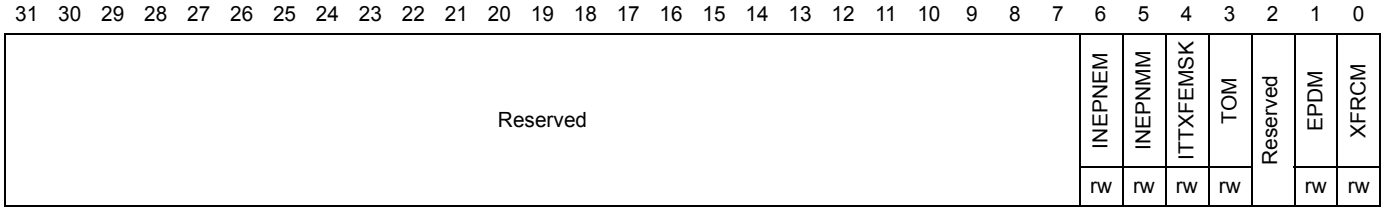
– Установке бита удалённой побудки RWUSIG в регистре OTG_FS_DCTL.

Общий регистр маски прерываний конечных точек IN устройства (OTG_FS_DIEPMSK)

Смещение адреса: 0x810

По сбросу: 0x0000 0000

Относится к выдаче прерываний всех регистров OTG_FS_DIEPINTx. По умолчанию всё замаскировано.



Маска прерывания: 0 - Нельзя, 1 - Можно.

— Биты 31:7 Резерв. не трогать.

— Бит 6 **INEPNEM**: У конечной точки IN действует NAK

— Бит 5 **INEPNMM**: Принят токен IN с несовпадающей EP

— Бит 4 **ITTXFEMSK**: Принят токен IN при пустом TxFIFO

— Бит 3 **TOM**: Таймаут (Не-изохронные конечные точки)

— Бит 2 Резерв. не трогать.

— Бит 1 **EPDM**: Конечная точка выключена

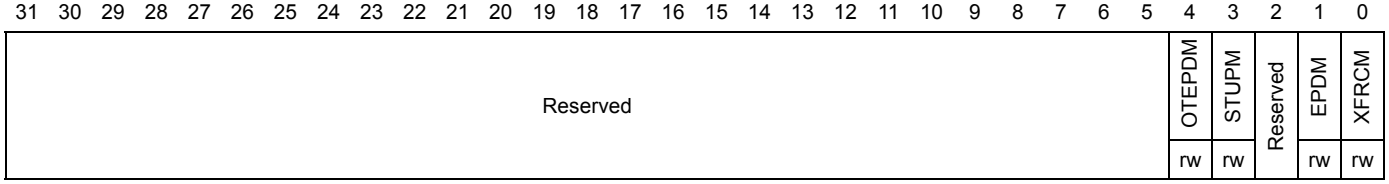
— Бит 0 **XFRCM**: Передача завершена

Общий регистр маски прерываний конечных точек OUT устройства (OTG_FS_DOEPMSK)

Смещение адреса: 0x814

По сбросу: 0x0000 0000

Относится к выдаче прерываний всех регистров OTG_FS_DOEPINTx По умолчанию всё замаскировано.



Маска прерывания: 0 - Нельзя, 1 - Можно.

— Биты 31:5 Резерв. не трогать.

— Бит 4 **OTEPDM**: Принят токен OUT выключенной управляющей точки OUT.

— Бит 3 **STUPM**: Фаза SETUP управляющей точки завершена.

— Бит 2 Резерв. не трогать.

— Бит 1 **EPDM**: Конечная точка выключена

— Бит 0 **XFRCM**: Передача завершена

Регистр прерываний всех конечных точек устройства (OTG_FS_DAINT)

Смещение адреса: 0x818

По сбросу: 0x0000 0000

Регистр OTG_FS_DAINT содержит прерывания всех конечных OUT и IN точек устройства. Прерывания разрешаются битами OEPINT и IEPINT в регистре OTG_FS_GINTSTS. Для двунаправленных точек используются два бита. Биты в этом регистре ставятся и снимаются соответствующими битами в регистрах OTG_FS_DIEPINTx/OTG_FS_DOEPINTx.

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

— Биты 31:16 **OEPINT**: Прерывания конечных точек OUT

Для точек OUT:

Бит 16 для OUT конечной точки 0, бит 19 для OUT конечной точки 3

— Биты 15:0 **IEPINT**: Прерывания конечных точек IN

Для точек IN:

Бит 0 для IN конечной точки 0, бит 3 для IN конечной точки 3.

Регистр маски прерываний всех конечных точек устройства (OTG_FS_DAINTMSK)

Смещение адреса: 0x81C

По сбросу: 0x0000 0000

Маскирует прерывания по регистру OTG_FS_DAINT.

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

Маска прерывания: 0 - Нельзя, 1 - Можно.

— Биты 31:16 **OEPINTM**: Прерывания конечных точек OUT

Для точек OUT:

Бит 16 для OUT конечной точки 0, бит 19 для OUT конечной точки 3

— Биты 15:0 **IEPINTM**: Прерывания конечных точек IN

Для точек IN:

Бит 0 для IN конечной точки 0, бит 3 для IN конечной точки 3.

Регистр времени разряда V_{BUS} устройства (OTG_FS_DVBUSDIS)

Смещение адреса: 0x828

По сбросу: 0x0000 17D7

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

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **VBUSDT**: Время разряда V_{BUS} устройства после импульса во время SRP.

Вычисляется в тактах PHY / 1 024. Зависит от нагрузки V_{BUS}, можно подстраивать.

Регистр времени импульса V_{BUS} устройства (OTG_FS_DVBUSPULSE)

Смещение адреса: 0x82C

По сбросу: 0x0000 05B8

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

— Биты 31:12 Резерв, не трогать.

— Биты 11:0 **DVBUSP:** Время импульса V_{BUS} устройства во время SRP.

Вычисляется в тактах PHY / 1 024.

Регистр маски прерываний пустого FIFO конечных точек IN (OTG_FS_DIEPEMPMSK)

Смещение адреса: 0x834

По сбросу: 0x0000 0000

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

Маска прерывания: 0 - Нельзя, 1 - Можно.

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **INEPTXFEM**: TxFIFO конечной точки IN пуст

Маскируют биты прерываний по TXFE регистров OTG_FS_DIEPINTx.

Бит 0 для IN конечной точки 0, бит 3 для IN конечной точки 3.

Регистр управления конечной точки IN 0 (OTG_FS_DIEPCTL0)

Смещение адреса: 0x900

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPENA	EPDIS	Reserved		SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	Reserved	USBAEP	Reserved													MPSIZ	
r	r			w	w	rw	rw	rw	rw	rs		r	r	r		r														rw	rw

— Бит 31 **EPENA**: Включение конечной точки

Установка бита запускает передачу данных конечной точки 0.

Снимается ядром перед выдачей прерываний:

– Точка выключена

– Передача завершена

— Бит 30 **EPDIS**: Выключение конечной точки

Установка бита останавливает даже незавершённую передачу данных конечной точки 0. Далее надо дождаться прерывания выключения точки. Снимается ядром перед выдачей прерывания. Останавливать можно только включённую конечную точку.

— Биты 29:28 Резерв, не трогать.

— Бит 27 **SNAK**: Установка ответа NAK

Программно задаёт выдачу NAK этой точки. Ядро может ставить его после приёма пакета SETUP.

— Бит 26 **CNAK**: Очистка NAK записью в этот бит

— Биты 25:22 **TXFNUM**: Номер TxFIFO точки IN 0.

— Бит 21 **STALL**: Ответ STALL

Ставится программно, снимается ядром после приёма пакета SETUP. Имеет приоритет перед битами NAK, Глобального IN NAK и Глобального OUT NAK.

— Бит 20 Резерв, не трогать.

— Биты 19:18 **EPTYP**: Тип конечной точки, ’00’ для управляющей точки.

— Бит 17 **NAKSTS**: Состояние NAK

0: Не-NAK ответы на основании состояния FIFO

1: Ответы NAK.

При стоящем бите ядро останавливает передачи данных даже при их наличии в TxFIFO. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 16 Резерв, не трогать.

— Бит 15 **USBAEP**: Активная конечная точка USB

Всегда стоит, точка 0 активна во всех конфигурациях и интерфейсах.

— Биты 14:2 Резерв, не трогать.

— Биты 1:0 **MPSIZ**: Максимальный размер пакета

00: 64 байта

01: 32 байта

10: 16 байт

11: 8 байт

Регистр управления конечной точки-x устройства (OTG_FS_DIEPCTLx)
(x = 1..3, где x = Номер_конечной_точки)

Смещение адреса: 0x900 + (Endpoint_number × 0x20)

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
rs	rs	w	w	w	w	rw	rw	rw	rw	rw/ rs		rw	rw	r	r	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **EPENA**: Включение конечной точки

Установка бита запускает передачу данных конечной точки.

Снимается ядром перед выдачей прерываний:

– Фаза SETUP завершена

– Точка выключена

– Передача завершена

— Бит 30 **EPDIS**: Выключение конечной точки

Установка бита останавливает даже незавершённую передачу данных конечной точки. Далее надо дождаться прерывания выключения точки. Снимается ядром перед выдачей прерывания. Останавливать можно только включённую конечную точку.

— Бит 29 **SODDFRM**: Нечётный фрейм изохронной передачи IN и OUT (поле EONUM).

— Бит 28 **SD0PID:** Установить PID DATA0 прерывающих/групповых точек IN (поле DPID).

SEVNFRM: Чётный фрейм изохронной передачи IN (поле EONUM)

— Бит 27 **SNAK**: Установить NAK

Программно задаёт выдачу NAK этой точки. Ядро может ставить его после завершения передачи точки OUT или приёма пакета SETUP.

— Бит 26 **CNAK:** Очистить бит NAK

— Биты 25:22 **TXFNUM**: Номер TxFIFO конечной точки IN.

Активные точки IN должны иметь разные FIFO.

— Бит 21 **STALL**: Ответ STALL

Для не-управляющих, не-изохронных точек IN (доступ rw).

Программно останавливает передачи от хоста точке. Имеет приоритет перед битами NAK, Глобального IN NAK и Глобального OUT NAK. Снимается только программно.

Для управляющих точек (доступ rs).

Ставится программно, снимается ядром после приёма пакета SETUP. Имеет приоритет перед битами NAK, Глобального IN NAK и Глобального OUT NAK. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 20 Резерв, не трогать.

— Биты 19:18 **EPTYP**: Тип конечной точки

00: Управляющая

01: Изохронная

10: Групповая

11: Прерывающая

— Бит 17 **NAKSTS**: Состояние NAK

0: Не-NAK ответы на основании состояния FIFO

1: Ответы NAK.

При стоящем бите ядро останавливает передачи данных даже при их наличии в TxFIFO. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

При стоящем бите:

Для не-изохронных точек IN: Ядро останавливает передачи по точке IN даже при данных в TxFIFO.

Для изохронных точек IN: Ядро посылает пакет данных нулевой длины даже при данных в TxFIFO. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 16 **EONUM**: Чётный/Нечётный фрейм изохронных точек IN.
Устанавливается записью в биты SEVNFRM и SODDFRM этого регистра.
0: Чётный
1: Нечётный
DPID: PID первого пакета данных прерывающей/групповой конечной точки
Устанавливается записью в бит SD0PID:
0: DATA0
1: DATA1

— Бит 15 **USBAEP**: Точка USB активна
По сбросу USB ядро снимает этот бит для всех точек, кроме EP 0. Ставить надо после завершения исполнения команд **SetConfiguration** и **SetInterface**.

— Биты 14:11 Резерв, не трогать.

— Биты 10:0 **MPSIZ**: Максимальный размер пакета в байтах

Регистр управления конечной точкой OUT 0 (OTG_FS_DOEPCTL0)

Смещение адреса: 0xB00

По сбросу: 0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPENA	EPDIS	Reserved		SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	Reserved	USBAEP	Reserved													MPSIZ	
w	r			w	w					rs	rw	r	r	r		r														r	r

— Бит 31 **EPENA**: Включить конечную точку
Установка бита запускает передачу данных конечной точки 0.
Снимается ядром перед выдачей прерываний:
– Фаза SETUP завершена
– Точка выключена
– Передача завершена

— Бит 30 **EPDIS**: Выключение конечной точки
Программно не выключается.

— Биты 29:28 Резерв, не трогать.

— Бит 27 **SNAK**: Установка NAK
Программно задаёт выдачу NAK этой точки. Ядро может ставить его после завершения передачи точки OUT или приёма пакета SETUP.

— Бит 26 **CNAK**: Очистка NAK

— Биты 25:22 Резерв, не трогать.

— Бит 21 **STALL**: Ответ STALL
Ставится программно, снимается ядром после приёма пакета SETUP. Имеет приоритет перед битами NAK и Глобального OUT NAK. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 20 **SNPM**: Режим Snoop (Подглядывания)
В этом режиме ядро не проверяет правильность пакетов OUT перед записью в память.

— Биты 19:18 **PTYP**: Тип конечной точки, ноль для управляющей.

— Бит 17 **NAKSTS**: Состояние NAK
0: Не-NAK ответы на основании состояния FIFO
1: Ответы NAK.
При стоящем бите ядро останавливает приём данных даже при наличии места в RxFIFO. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 16 Резерв, не трогать.

— Бит 15 **USBAEP**: Активная конечная точка, всегда равен 1.

— Биты 14:2 Резерв, не трогать.

— Бит 1:0 **MPSIZ**: Максимальный размер пакета точки OUT
00: 64 байта
01: 32 байта
10: 16 байт
11: 8 байт

Регистр управления конечной точки-x (OTG_FS_DOEPCTLx)
(x = 1..3, где x = Номер_конечной_точки)

Смещение адреса точки OUT: 0xB00 + (Endpoint_number × 0x20)

По сбросу: 0x0000 0000

31	30	29	28	27	26	25 24 23 22	21	20	19 18	17	16	15	14 13 12 11	10 9 8 7 6 5 4 3 2 1 0
EPENA	EPDIS	SODDFRM/SD1PID	SD0PID/SEVNFRM	SNAK	CNAK	Reserved	Stall	SNPM	EPTYP	NAKSTS	EONUM/DPID	USBAEP	Reserved	MPSIZ
rs	rs	w	w	w	w		rw/ rs	rw	rw	r	r	rw		rw

— Бит 31 **EPENA**: Включение конечной точки IN и OUT.

Установка бита запускает передачу данных конечной точки.

Снимается ядром перед выдачей прерываний:

– Фаза SETUP завершена

– Точка выключена

– Передача завершена

— Бит 30 **EPDIS**: Выключение конечной точки

Установка бита останавливает даже незавершённую передачу/приём данных конечной точки. Далее надо дождаться прерывания выключения точки. Снимается ядром перед выдачей прерывания. Останавливать можно только включённую конечную точку.

— Бит 29 **SD1PID**: Установить PID DATA1 (DPID) прерывающих/групповых точек IN и OUT.

SODDFRM: Установить нечётный фрейм изохронных точек IN и OUT (поле EONUM)

— Бит 28 **SD0PID**: Установить PID DATA0 (DPID) прерывающих/групповых точек OUT

SEVNFRM: Установить чётный фрейм изохронных точек OUT (поле EONUM)

— Бит 27 **SNAK**: Установка NAK

Программно задаёт выдачу NAK этой точки. Ядро может ставить его после завершения передачи точки OUT или приёма пакета SETUP.

— Бит 26 **CNAK**: Очистка NAK

— Биты 25:22 Резерв, не трогать.

— Бит 21 **STALL**: Ответ STALL

Для не-управляющих, не-изохронных точек IN (доступ rw).

Программно останавливает все передачи от хоста точке. Имеет приоритет перед битами NAK, Глобального IN NAK и Глобального OUT NAK. Снимается только программно.

Для управляющих точек (доступ rs).

Ставится программно, снимается ядром после приёма пакета SETUP. Имеет приоритет перед битами NAK, Глобального IN NAK и Глобального OUT NAK. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 20 **SNPM**: Режим Snoop (Подглядывания)

В этом режиме ядро не проверяет правильность пакетов OUT перед записью в память.

— Биты 19:18 **EPTYP**: Тип конечной точки

00: Управляющая

01: Изохронная

10: Групповая

11: Прерывающая

— Бит 17 **NAKSTS**: Состояние NAK

0: Не-NAK ответы на основании состояния FIFO

1: Ответы NAK.

При стоящем бите ядро останавливает приём данных даже при наличии места в RxFIFO. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 16 **EONUM**: Чётный/Нечётный фрейм изохронных точек IN.

Устанавливается записью в биты SEVNFRM и SODDFRM этого регистра.

0: Чётный

1: Нечётный

DPID: PID первого пакета данных прерывающей/групповой конечной точки

Устанавливается записью в бит SD0PID:

0: DATA0

1: DATA1

— Бит 15 **USBAEP**: Активная конечная точка USB

По сбросу USB ядро снимает этот бит для всех точек, кроме EP 0. Ставить надо после завершения исполнения команд **SetConfiguration** и **SetInterface**.

— Биты 14:11 Резерв, не трогать.

— Биты 10:0 **MPSIZ**: Максимальный размер пакета в байтах

Регистр прерываний конечной точки-x (OTG_FS_DIEPINTx) (x = 0..3, где x = Номер_конечной_точки)

Смещение адреса: 0x908 + (Endpoint_number × 0x20)

По сбросу: 0x0000 0080

Читать можно только при стоящем бите IEPINT в OTG_FS_GINTSTS, но сначала надо узнать номер точки по регистру OTG_FS_DAINT. Снятие бита в этом регистре чистит соответствующие биты в регистрах OTG_FS_DAINT и OTG_FS_GINTSTS.

31:8	7	6	5	4	3	2	1	0
Reserved	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	r	rc_w1/rw		rc_w1	rc_w1		rc_w1	rc_w1

— Биты 31:8 Резерв, не трогать.

— Бит 7 **TXFE**: Передающий FIFO полу- или полностью пуст

Уровень пустоты TxFIFO задаётся битом TXFELVL в регистре OTG_FS_GAHBCFG.

— Бит 6 **INEPNE**: NAK конечной точки IN в действии

Снимается записью бита CNAK в регистре OTG_FS_DIEPCTLx. Установка бита вызывает прерывание. По шине USB не обязательно пойдёт NAK, бит STALL приоритетнее.

— Бит 5 Резерв, не трогать.

— Бит 4 **ITTXFE**: Принят токен IN при пустом не-периодическом TxFIFO

Выставляется прерывание этой точки.

— Бит 3 **TOC**: Таймаут управляющей точки IN.

— Бит 2 Резерв, не трогать.

— Бит 1 **EPDISD**: Прерывание выключенной конечной точки

— Бит 0 **XFRC**: Прерывание конца передачи по AHB или USB.

Регистр прерываний конечной точки-x (OTG_FS_DOEPINTx) (x = 0..3, где x = Номер_конечной_точки)

Смещение адреса: 0xB08 + (Endpoint_number × 0x20)

По сбросу: 0x0000 0080

Читать можно только при стоящем бите OEPINT в регистре OTG_FS_GINTSTS, но сначала надо узнать номер точки по регистру OTG_FS_DAINT. Снятие бита в этом регистре чистит соответствующие биты в регистрах OTG_FS_DAINT и OTG_FS_GINTSTS.

31:7	6	5	4	3	2	1	0
Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	rc_w1/rw		rc_w1	rc_w1		rc_w1	rc_w1

— Биты 31:7 Резерв, не трогать.

— Бит 6 **B2BSTUP**: Принята череда (больше 3) пакетов SETUP управляющей точки OUT.

— Бит 5 Резерв, не трогать.

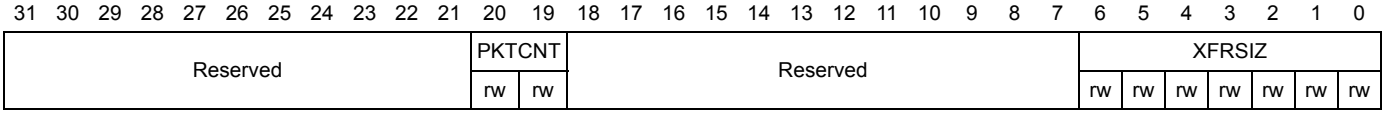
— Бит 4 **OTEPDIS**: Принят пакет OUT выключенной управляющей точки (с прерыванием)

— Бит 3 **STUP**: Фаза SETUP управляющей точки OUT завершена (с прерыванием)

— Бит 2 Резерв, не трогать.
— Бит 1 **EPDISD**: Прерывание выключения конечной точки
— Бит 0 **XFRC**: Прерывание конца передачи по AHB или USB.

Регистр размера передачи конечной точки IN 0 (OTG_FS_DIEPTSIZ0)

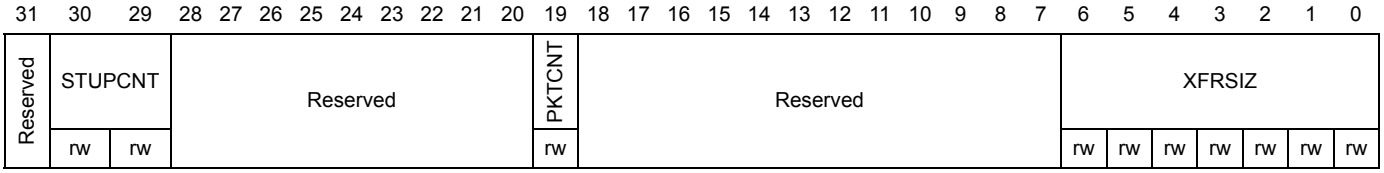
Смещение адреса: 0x910
По сбросу: 0x0000 0000
Писать надо до включения точки 0. После этого можно только читать.



— Биты 31:21 Резерв, не трогать.
— Биты 20:19 **PKTCNT**: Счётчик общего числа пакетов передачи
Декрементируется при каждом чтении из TxFIFO.
— Биты 18:7 Резерв, не трогать.
— Биты 6:0 **XFRSIZ**: Размер передачи в байтах (с прерыванием после исчерпания)
Уменьшается при каждой записи в TxFIFO.

Регистр размера передачи конечной точки OUT 0 (OTG_FS_DOEPTSIZ0)

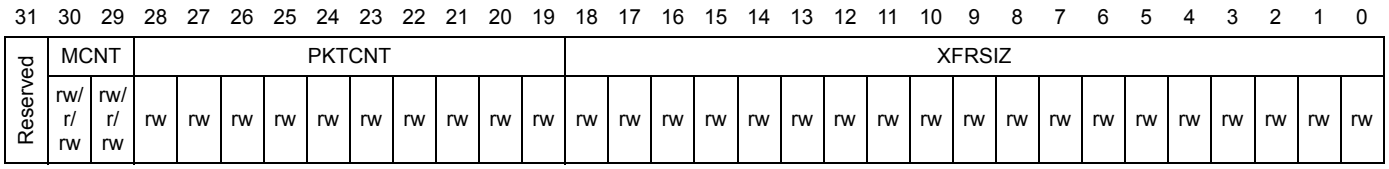
Смещение адреса: 0xB10
По сбросу: 0x0000 0000
Писать надо до включения точки 0. После этого можно только читать.



— Бит 31 Резерв, не трогать.
— Биты 30:29 **STUPCNT**: Счётчик череды пакетов SETUP
01: 1 пакет
10: 2 пакета
11: 3 пакета
— Биты 28:20 Резерв, не трогать.
— Бит 19 **PKTCNT**: Счётчик пакетов
Обнуляется при записи в RxFIFO.
— Биты 18:7 Резерв, не трогать.
— Биты 6:0 **XFRSIZ**: Размер передачи в байтах (с прерыванием после исчерпания)
Уменьшается при каждом чтении из RxFIFO.

Регистр размера передачи конечной точки-x (OTG_FS_DIEPTSIZx) (x = 1..3, где x = Номер_конечной_точки)

Смещение адреса: 0xB10 + (Endpoint_number × 0x20)
По сбросу: 0x0000 0000
Писать надо до включения точки 0. После этого можно только читать.



— Бит 31 Резерв, не трогать.
— Биты 30:29 **MCNT**: Многосчётчик пакетов в периодическом фрейме IN.
01: 1 пакет
10: 2 пакета
11: 3 пакета

— Биты 28:19 **PKTCNT**: Счётчик общего числа пакетов передачи
Декрементируется при каждом чтении из TxFIFO.
— Биты 18:0 **XFRSIZ**: Размер передачи в байтах
Уменьшается при каждой записи в TxFIFO.

Регистр состояния FIFO конечной точки IN (OTG_FS_DTXFSTSx) (x = 0..3, где x = Номер_конечной_точки)

Смещение адреса точки IN: 0x918 + (Endpoint_number × 0x20)

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

— Биты 31:16 Резерв, не трогать.
— Биты 15:0 **INEPTFSAV**: Доступное место в TxFIFO точки IN в 32-бит словах
0x0: TxFIFO полон
0x1: 1 слово
0x2: 2 слова
0xn: n слов
Иное: Резерв

Регистр размера передачи конечной точки-x OUT (OTG_FS_DOEPTSIZx) (x = 1..3, где x = Номер_конечной_точки)

Смещение адреса: 0xB10 + (Endpoint_number × 0x20)

По сбросу: 0x0000 0000

Писать надо до включения точки 0. После этого можно только читать.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	rw/rw	rw/rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 Резерв, не трогать.
— Биты 30:29 **RXDPID**: PID принятых данных изохронной точки OUT.
00: DATA0
01: DATA2
10: DATA1
11: MDATA
STUPCNT: Максимальное число пакетов SETUP управляющей точки OUT.
01: 1 пакет
10: 2 пакета
11: 3 пакета
— Биты 28:19 **PKTCNT**: Счётчик общего числа пакетов передачи
Декрементируется при каждой записи в RxFIFO.
— Биты 18:0 **XFRSIZ**: Размер передачи в байтах (с прерыванием после исчерпания)
Уменьшается при каждом чтении из RxFIFO.

28.16.5. Регистр питания и тактов OTG_FS (OTG_FS_PCGCCTL)

Смещение адреса: 0xE00

По сбросу: 0x0000 0000

Доступен в обоих режимах.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											PHYSUSP	Reserved		GATEHCLK	STPPCLK
																											rw			rw	rw

— Биты 31:5 Резерв, не трогать.
— Бит 4 **PHYSUSP**: PHY остановлен
— Биты 3:2 Резерв, не трогать.
— Бит 1 **GATEHCLK**: Разрешение подачи HCLK
Ставится и снимается программно.
— Бит 0 **STPPCLK**: Остановка тактов PHY
Ставится и снимается программно.

28.16.6. Карта регистров OTG_FS

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**OTG_FS_GOTGCTL**	Reserved												BSVLD	ASVLD	DBCT	CIDSTS	Reserved				DHNPEN	HSHNPEN	HNPRQ	HNGSCS	Reserved						SRQ	SRQSCS
	Reset value													0	0	0	1					0	0	0	0							0	0
0x004	**OTG_FS_GOTGINT**	Reserved												DBCDNE	ADTOCHG	HNGDET	Reserved							HNSSCHG	SRSSCHG	Reserved					SEDET	Res.	
	Reset value													0	0	0								0	0						0		
0x008	**OTG_FS_GAHBCFG**	Reserved																							PTXFELVL	TXFELVL	Reserved						GINTMSK
	Reset value																								0	0							0
0x00C	**OTG_FS_GUSBCFG**	CTXPKT	FDMOD	FHMOD	Reserved									TRDT									HNPCAP	SRPCAP	Reserved	PHYSEL	Reserved				TOCAL		
	Reset value													0	1	0	1						0	0		1					0	0	0
0x010	**OTG_FS_GRSTCTL**	AHBIDL	Reserved																			TXFNUM							RXFFLSH	Reserved	FCRST	HSRST	CSRST
	Reset value	1																				0	0	0		0			0	0	0	0	0
0x014	**OTG_FS_GINTSTS**	WKUINT	SRQINT	DISCINT	CIDSCHG	Reserved	PTXFE	HCINT	HPRTINT	Reserved		IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Reserved		EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Reserved		GONAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
	Reset value	0	0	0	0		1	0	0			0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0
0x018	**OTG_FS_GINTMSK**	WUIM	SRQIM	DISCINT	CIDSCHGM	Reserved	PTXFEM	HCIM	PRTIM	Reserved		IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	EPMISM	Reserved	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Reserved		GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Reserved
	Reset value	0	0	0	0		0	0	0			0	0	0	0	0		0	0	0	0	0	0			0	0	0	0	0	0	0	
0x01C	**OTG_FS_GRXSTSR (host mode)**	Reserved												PKTSTS				DPID				BCNT									CHNUM		
	Reset value													0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	OTG_FS_GRXSTSR (Device mode)	Reserved							FRMNUM					PKTSTS				DPID				BCNT									EPNUM		
	Reset value								0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x020	**OTG_FS_GRXSTSR (host mode)**	Reserved											PKTSTS				DPID		BCNT												CHNUM		
	Reset value												0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	OTG_FS_GRXSTSPR (Device mode)	Reserved							FRMNUM				PKTSTS				DPID		BCNT												EPNUM		
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	**OTG_FS_GRXFSIZ**	Reserved																	RXFD														
	Reset value																		0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x028	**OTG_FS_HNPTXFSIZ/ OTG_FS_DIEPTXF0**	NPTXFD/TX0FD																	NPTXFSA/TX0FSA														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x02C	**OTG_FS_HNPTXSTS**	Res.	NPTXQTOP							NPTQXSAV									NPTXFSAV														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0		0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x038	**OTG_FS_GCCFG**	Reserved											SOFOUTEN	VBUSBSEN	VBUSASEN	Reserved	.PWRDWN		Reserved															
	Reset value												0	0	0		0																
0x03C	**OTG_FS_CID**	PRODUCT_ID																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0x100	**OTG_FS_HPTXFSIZ**	PTXFSIZ																	PTXSA														
	Reset value	0	0	0	0	0	1	1	1	0	1	1	0	1	0	0	0		0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0
0x104	**OTG_FS_DIEPTXF1**	INEPTXFD																	INEPTXSA														
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0		0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x108	**OTG_FS_DIEPTXF2**	INEPTXFD																	INEPTXSA														
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0		0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x10C	**OTG_FS_DIEPTXF3**	INEPTXFD																	INEPTXSA														
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0		0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x400	**OTG_FS_HCFG**	Reserved																														FSLSS	FSLSPCS
	Reset value																															0	0	0
0x404	**OTG_FS_HFIR**	Reserved																	FRIVL														
	Reset value																		1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
0x408	**OTG_FS_HFNUM**	FTREM																	FRNUM														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x410	**OTG_FS_HPTXSTS**	PTXQTOP								PTXQSAV									PTXFSAVL														
	Reset value	0	0	0	0	0	0	0	0	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
0x414	**OTG_FS_HAINT**	Reserved																	HAINT														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x418	**OTG_FS_HAINTMSK**	Reserved																HAINTM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x440	**OTG_FS_HPRT**	Reserved													PSPD	PTCTL					PPWR	PLSTS		Reserved	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x500	**OTG_FS_HCCHAR0**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x520	**OTG_FS_HCCHAR1**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x540	**OTG_FS_HCCHAR2**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x560	**OTG_FS_HCCHAR3**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x580	**OTG_FS_HCCHAR4**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5A0	**OTG_FS_HCCHAR5**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C0	**OTG_FS_HCCHAR6**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5E0	**OTG_FS_HCCHAR7**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x508	**OTG_FS_HCINT0**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x528	**OTG_FS_HCINT1**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x548	**OTG_FS_HCINT2**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x568	**OTG_FS_HCINT3**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x588	**OTG_FS_HCINT4**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x5A8	**OTG_FS_HCINT5**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x5C8	**OTG_FS_HCINT6**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x5E8	**OTG_FS_HCINT7**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x50C	**OTG_FS_HCINTMSK0**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x52C	**OTG_FS_HCINTMSK1**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x54C	**OTG_FS_HCINTMSK2**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x56C	**OTG_FS_HCINTMSK3**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x58C	**OTG_FS_HCINTMSK4**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x5AC	**OTG_FS_HCINTMSK5**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5CC	**OTG_FS_HCINTMSK6**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x5EC	**OTG_FS_HCINTMSK7**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x510	**OTG_FS_HCTSIZ0**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x530	**OTG_FS_HCTSIZ1**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x550	**OTG_FS_HCTSIZ2**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x570	**OTG_FS_HCTSIZ3**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x590	**OTG_FS_HCTSIZ4**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5B0	**OTG_FS_HCTSIZ5**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5D0	**OTG_FS_HCTSIZ6**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5F0	**OTG_FS_HCTSIZ7**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x800	**OTG_FS_DCFG**	Reserved																			PFIVL		DAD							Reserved	NZLSOHSK	DSPD	
	Reset value																				0	0	0	0	0	0	0	0	0		0	0	0
0x804	**OTG_FS_DCTL**	Reserved																				POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x808	**OTG_FS_DSTS**	Reserved										FNSOF														Reserved				EERR	ENUMSPD		SUSPSTS
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x810	**OTG_FS_DIEPMSK**	Reserved																									INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM
	Reset value																										0	0	0	0		0	0
0x814	**OTG_FS_DOEPMSK**	Reserved																											OTEPDM	STUPM	Reserved	EPDM	XFRCM
	Reset value																												0	0		0	0
0x818	**OTG_FS_DAINT**	OEPINT																IEPINT															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x81C	**OTG_FS_DAINTMSK**	OEPM																IEPM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x828	**OTG_FS_DVBUSDIS**	Reserved																VBUSDT															
	Reset value																	0	0	0	1	0	1	1	1	1	1	0	1	0	1	1	1
0x82C	**OTG_FS_DVBUSPULSE**	Reserved																				DVBUSP											
	Reset value																					0	1	0	1	1	0	1	1	1	0	0	0
0x834	**OTG_FS_DIEPEMPMSK**	Reserved																INEPTXFEM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x900	**OTG_FS_DIEPCTL0**	EPENA	EPDIS	Reserved			SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	Reserved	USBAEP	Reserved													MPSIZ
	Reset value	0	0				0	0	0	0	0	0	0		0	0	0		1														0	0
0x918	**TG_FS_DTXFSTS0**	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x920	**OTG_FS_DIEPCTL1**	EPENA	EPDIS	SODDFRM/SD1PID	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM					Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved													MPSIZ	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0														0	0
0x938	**TG_FS_DTXFSTS1**	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x940	**OTG_FS_DIEPCTL2**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM					Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved													MPSIZ	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0														0	0
0x958	**TG_FS_DTXFSTS2**	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x960	**OTG_FS_DIEPCTL3**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x978	**TG_FS_DTXFSTS3**	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0xB00	**OTG_FS_DOEPCTL0**	EPENA	EPDIS	Reserved		SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	Reserved	USBAEP	Reserved													MPSIZ	
	Reset value	0	0			0	0					0	0	0	0	0		1														0	0
0xB20	**OTG_FS_DOEPCTL1**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB40	**OTG_FS_DOEPCTL2**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB60	**OTG_FS_DOEPCTL3**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x908	**OTG_FS_DIEPINT0**	Reserved																								TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																									1	0		0	0		0	0
0x928	**OTG_FS_DIEPINT1**	Reserved																								TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																									1	0		0	0		0	0
0x948	**OTG_FS_DIEPINT2**	Reserved																								TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																									1	0		0	0		0	0
0x968	**OTG_FS_DIEPINT3**	Reserved																								TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																									1	0		0	0		0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB08	**OTG_FS_DOEPINT0**	Reserved																								Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																										0		0	0		0	0
0xB28	**OTG_FS_DOEPINT1**	Reserved																								Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																										0		0	0		0	0
0xB48	**OTG_FS_DOEPINT2**	Reserved																								Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																										0		0	0		0	0
0xB68	**OTG_FS_DOEPINT3**	Reserved																								Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																										0		0	0		0	0
0x910	**OTG_FS_DIEPTSIZ0**	Reserved												PKTCNT	Reserved												XFRSIZ						
	Reset value													0	0												0	0	0	0	0	0	0
0x930	**OTG_FS_DIEPTSIZ1**	Reserved	MCNT		PKTCNT											XFRSIZ																	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x950	**OTG_FS_DIEPTSIZ2**	Reserved	MCNT		PKTCNT											XFRSIZ																	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x970	**OTG_FS_DIEPTSIZ3**	Reserved	MCNT		PKTCNT											XFRSIZ																	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB10	**OTG_FS_DOEPTSIZ0**	Reserved	STUPCNT		Reserved									PKTCNT	Reserved												XFRSIZ						
	Reset value		0	0										0													0	0	0	0	0	0	0
0xB30	**OTG_FS_DOEPTSIZ1**	Reserved	RXDPID/STUPCNT		PKTCNT											XFRSIZ																	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB50	**OTG_FS_DOEPTSIZ2**	Reserved	RXDPID/STUPCNT		PKTCNT											XFRSIZ																	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB70	**OTG_FS_DOEPTSIZ3**	Reserved	RXDPID/STUPCNT		PKTCNT											XFRSIZ																	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	Reserved	PHYSUSP	Reserved	GATEHCLK	STPPCLK
0xE00	**OTG_FS_PCGCCTL**					
	Reset value					

28.17. Программная модель OTG_FS

28.17.1. Инициализация ядра

Сторона подключения кабеля (A или B) и режим контроллера показывается битом CMOD в регистре OTG_FS_GINTSTS).

Независимо от режимов хоста или устройства сначала пишем глобальные регистры:

1. В регистре OTG_FS_GAHBCFG:
   - Бит маски глобального прерывания GINTMSK = 1
   - RxFIFO не пуст (бит RXFLVL в OTG_FS_GINTSTS)
   - Уровень пустоты периодического TxFIFO
2. В регистре OTG_FS_GUSBCFG:
   - Бит разрешения HNP
   - Бит разрешения SRP
   - Поле калибровки таймаута FS
   - Поле времени обратной связи USB
3. Разрешаем маски в регистре OTG_FS_GINTMSK:
   - Маска прерывания OTG
   - Маска прерывания несовпадения режимов
4. По биту CMOD в OTG_FS_GINTSTS определяем режим контроллера OTG_FS (хост или устройство).

28.17.2. Инициализация хоста

Исполняем следующие шаги:

1. Демаскируем HPRTINT в регистре OTG_FS_GINTMSK
2. В регистре OTG_FS_HCFG ставим полную скорость
3. Битом PPWR в OTG_FS_HPRT включаем V_{BUS} на USB.
4. Ждём подключения устройства к хосту по прерыванию PCDET в OTG_FS_HPRT0.
5. Запускаем сброс битом PRST в регистре OTG_FS_HPRT.
6. Не меньше 10 ms ждём его завершения.
7. Снимаем бит PRST в регистре OTG_FS_HPRT.
8. Ждём прерывания PENCHNG в регистре OTG_FS_HPRT.
9. Узнаём скорость переучёта по биту PSPD в регистре OTG_FS_HPRT.
10. В регистр HFIR пишем нужное число для 1 такта PHY.
11. В соответствии с полученной скоростью устройства пишем поле FSLSPCS регистра OTG_FS_HCFG. Если оно изменилось, то надо сбросить порт.
12. В регистр OTG_FS_GRXFSIZ пишем размер приёмного FIFO.
13. В регистр OTG_FS_HNPTXFSIZ пишем размер и адрес начала не-периодического передающего FIFO.
14. В регистр OTG_FS_HNPTXFSIZ пишем размер и адрес начала периодического передающего FIFO.

Для работы с устройствами нужно инициализировать хоть один канал.

28.17.3. Инициализация устройства

После сброса по включению питания или изменении режима на устройство надо:

1. В регистре OTG_FS_DCFG определить:
   - Скорость устройства
   - Состояние ответа OUT ненулевой длины.
2. В регистре OTG_FS_GINTMSK демаскировать прерывания:
   - Сброс USB
   - Переучёт завершён
   - Ранний останов

- Останов USB
- SOF

3. Битом VBUSBSEN в OTG_FS_GCCFG разрешить контроль V_{BUS} устройства “B” и резистор подпорки линии DP.
4. Дождаться прерывания USBRST в регистре OTG_FS_GINTSTS. После получения этого прерывания сброс длится ещё 10 ms.

Ждём прерывания ENUMDNE в регистре OTG_FS_GINTSTS. Это конец сброса USB. Теперь по регистру OTG_FS_DSTS определяем скорость переучёта и инициализируем конечные точки.

Всё, устройство готово принимать пакеты SOF и выдавать управляющие передачи по точке 0.

28.17.4. Программная модель хоста

Инициализация канала

Выполняем шаги:

1. В регистре OTG_FS_GINTMSK демаскируем:
2. Прерывания канала
   - Пустого не-периодического передающего FIFO передач OUT или
   - Полупустого не-периодического передающего FIFO передач OUT.
3. В регистре OTG_FS_GINTMSK демаскируем прерывания выбранных каналов.
4. В регистре OTG_FS_GINTMSK демаскируем прерывания нужных условий передач.
5. В регистры OTG_FS_HCTSIZx пишем общий размер передач и ожидаемое число пакетов, включая короткие. В поле PID пишем начальный PID (первой для OUT или ожидаемой от IN).
6. В регистре OTG_FS_HCCHARx выбранного канала пишем характеристики устройства, вроде типа, скорости, направления и т.д.

Канал можно включать только при готовности к передаче или приёму данных.

Остановка канала

Выключить канал можно установкой битов CHDIS и CHENA регистра OTG_FS_HCCHARx. Хост сливает стоящие запросы, не прерывая запущенные передачи, и выдаёт прерывание остановки канала. Перед выделением канала для другой работы надо дождаться прерывания CHH в OTG_FS_HCINTx.

Перед выключением канала нужно убедиться в наличии хоть одной свободной строки в соответствующей очереди запросов (периодической или не-периодической). Запросы в очереди можно слить установкой бита CHDIS и очисткой бита CHENA регистра OTG_FS_HCCHARx.

Канал выключают при:

1. Прерываниях STALL, TXERR, BBERR или DTERR регистра OTG_FS_HCINTx. При этом такой же канал должен уметь принимать прерывания DTERR, NAK, DATA, TXERR.
2. Прерывании DISCINT регистра OTG_FS_GINTSTS. (выключаются все каналы).
3. При прекращении передач до их нормального завершения.

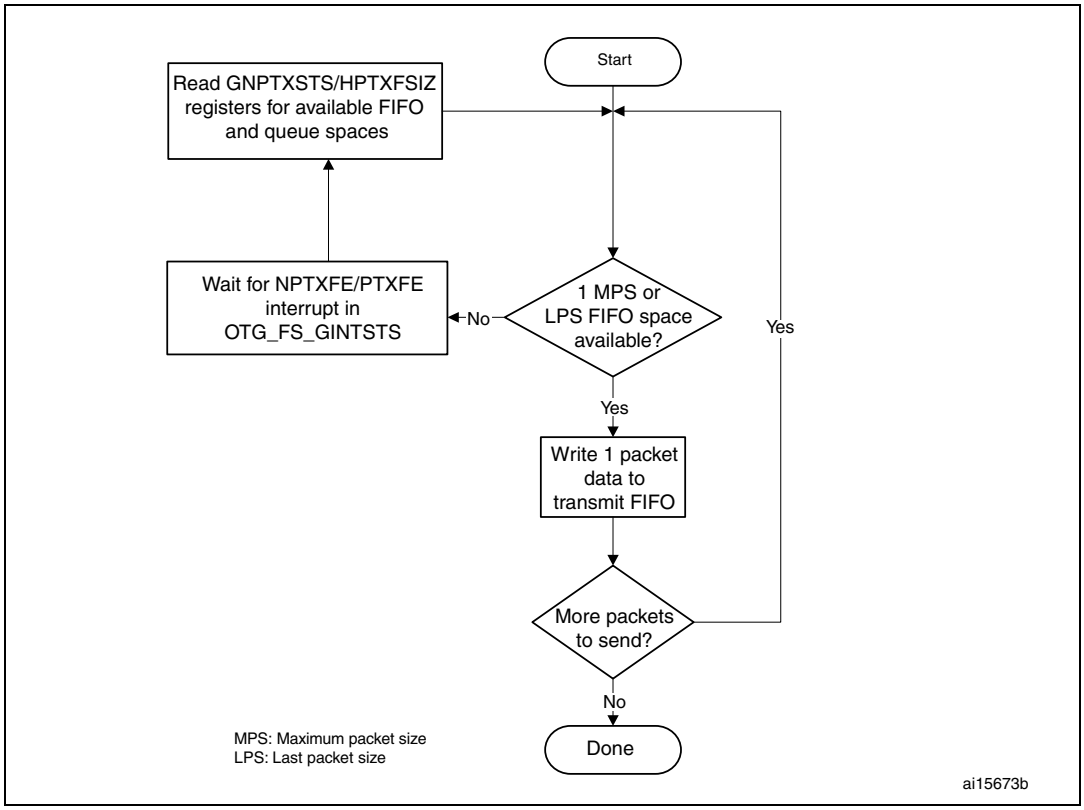
Модель работы

Всё делается с уже инициализированным каналом.

- **Запись в передающий FIFO**

Одновременно с записью последнего слова пакета, хост OTG_FS автоматически пишет запрос OUT в периодическую/не-периодическую очередь. Писать надо только при наличии свободного места в передающем FIFO. В FIFO пишут только словами. При некратной длине пакета он дополняется. Хост OTG_FS определяет конкретную длину пакета на основе установленных максимальной длины пакета и размера передачи.

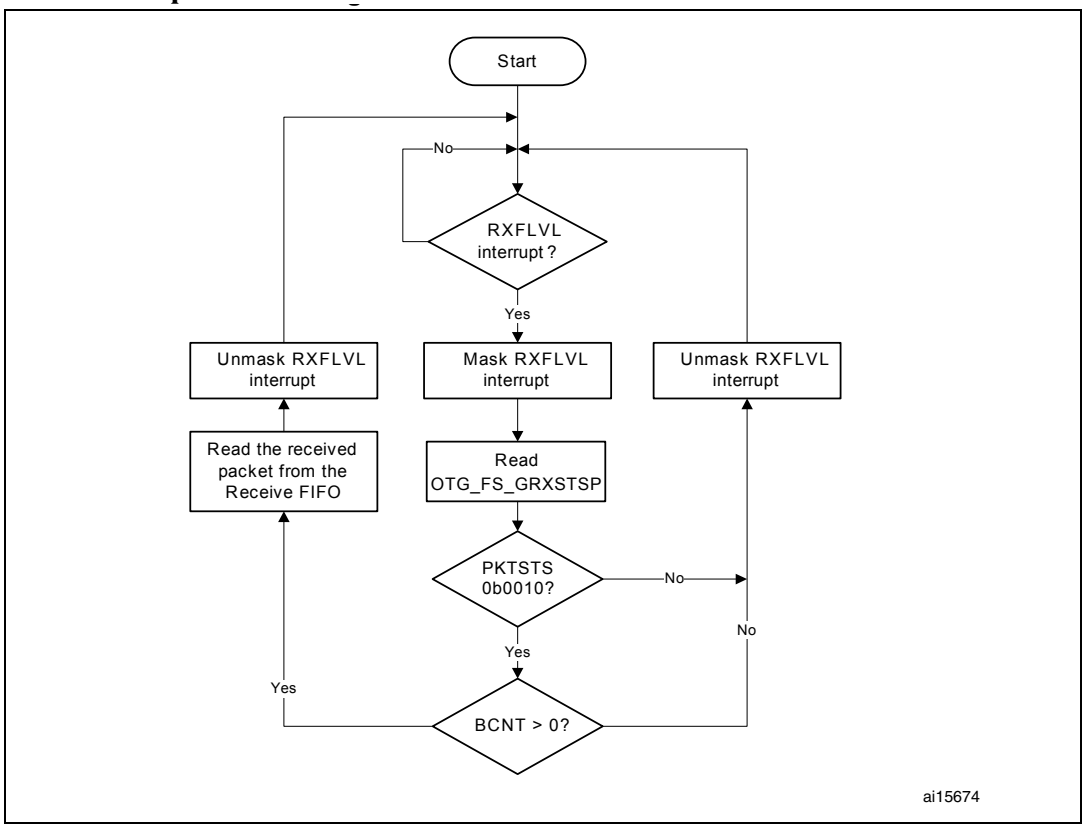
Запись в передающий FIFO



• Чтение приёмного FIFO

Пакеты, отличные от пакетов данных IN (bx0010) должны игнорироваться.

Чтение приёмного FIFO



• Групповые и управляющие передачи OUT/SETUP

На рисунке ниже передаются два пакета OUT и один пакет SETUP. Полагаем, что:

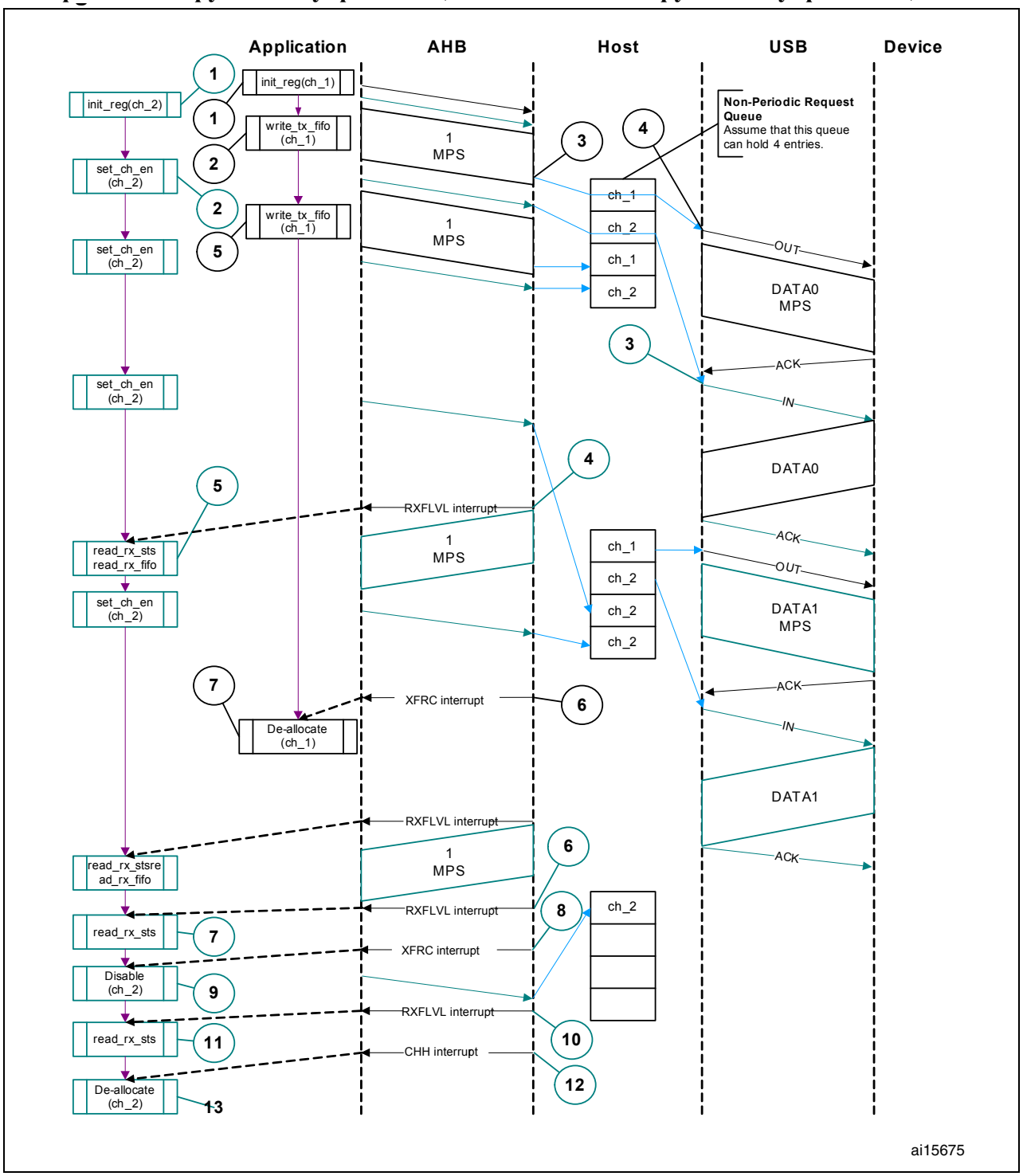
- Передаются два максимальных пакета (размер = 1024 байта).
- Передающий не-периодический FIFO может хранить два пакета (128 байт для FS).
- Глубина очереди не-периодических запросов = 4.

- **Нормальные групповые и управляющие операции OUT/SETUP**

Последовательность действий для канала 1 такова:

a) Инициализируем канал 1
b) Пишем первый пакет канала 1
c) Вместе с записью последнего слова хост пишет не-периодический запрос в очередь
d) После появления запроса в очереди ядро пытается послать токен OUT в текущем фрейме
e) Пишем второй (последний) пакет канала 1
f) После успешного завершения последней передачи выдаётся прерывание `XFRC`
g) По прерыванию `XFRC` освобождает канал для других передач
h) Обрабатываем не-ACK ответы

Нормальные групповые/управляющие OUT/SETUP и групповые/управляющие IN



• **Обработчик прерываний групповых/управляющих передач OUT/SETUP и IN**

a) Групповые/Управляющие OUT/SETUP

Наличие свободного места в FIFO и в очереди запросов определяют по прерыванию `NPTXFE` регистра `OTG_FS_GINTSTS`.

```
Unmask (NAK/TXERR/STALL/XFRC)
if (XFRC) {
    Reset Error Count
    Mask ACK
    De-allocate Channel
    }
else if (STALL) {
    Transfer Done = 1
    Unmask CHH
    Disable Channel
    }
else if (NAK or TXERR ) {
    Rewind Buffer Pointers
    Unmask CHH
    Disable Channel
    if (TXERR)   {
        Increment Error Count
        Unmask ACK
        }
    else {
        Reset Error Count
        }
    }
else if (CHH){
    Mask CHH
    if (Transfer Done or (Error_count == 3)) {
        De-allocate Channel
        }
    else  {
        Re-initialize Channel
        }
    }
else if (ACK) {
    Reset Error Count
    Mask ACK
    }
```

b) Групповые/Управляющие IN

Запросы выдаются при наличии свободного места в очереди.

```
Unmask (TXERR/XFRC/BBERR/STALL/DTERR)
if (XFRC){
    Reset Error Count
    Unmask CHH
    Disable Channel
    Reset Error Count
    Mask ACK
    }
else if (TXERR or BBERR or STALL) {
    Unmask CHH
    Disable Channel
    if (TXERR) {
        Increment Error Count
        Unmask ACK
        }
    }
else if (CHH) {
    Mask CHH
    if (Transfer Done or (Error_count == 3)) {
```

```
        De-allocate Channel
        }
    else {
        Re-initialize Channel
        }
    }
else if (ACK) {
    Reset Error Count
    Mask ACK
    }
else if (DTERR) {
    Reset Error Count
    }
```

- **Групповые и управляющие передачи IN**

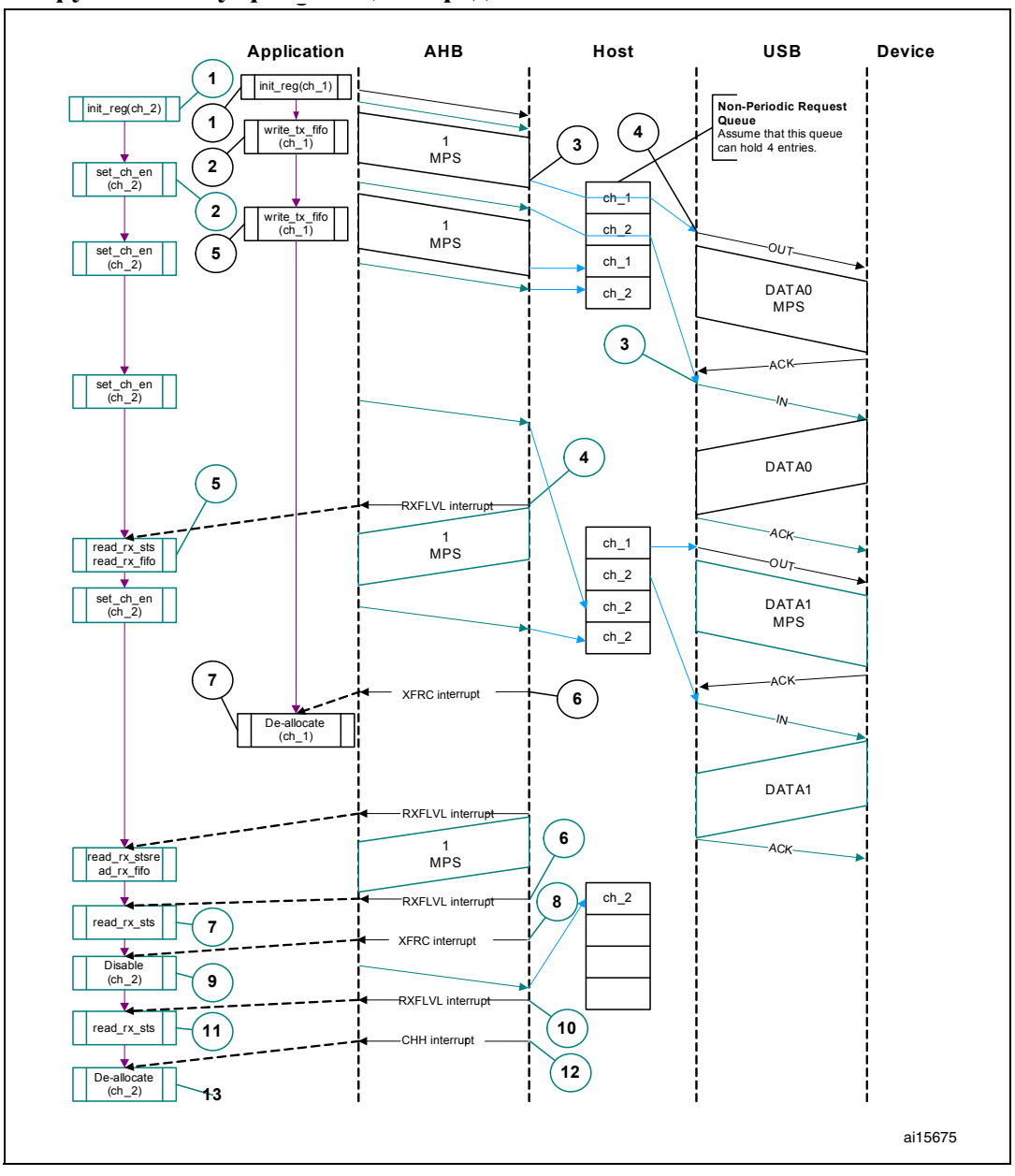
На рисунке ниже принимаем групповые или управляющие передачи IN по каналу 2. Полагаем, что:

– Принимаются два максимальных пакета (размер = 1024 байта).

– Приёмный FIFO может хранить хоть один максимальный пакет и два слова состояния на пакет (72 байта для FS).

– Глубина не-периодической очереди = 4.

Групповые и управляющие передачи IN



Последовательность операций такова:

a) Инициализируем канал 2.

b) Ставим бит CHENA в HCCHAR2 для записи запроса IN в не-периодическую очередь.

c) Ядро пробует послать токен IN после завершения текущей передачи OUT.

d) После приёма пакета в FIFO ядро выдаёт прерывание RXFLVL.

e) Приняв прерывание RXFLVL маскируем его, читаем число принятых байтов из состояния принятого пакета и читаем FIFO. Затем демаскируем прерывание RXFLVL.

f) Ядро выдаёт прерывание RXFLVL по каждой строке состояния конца приёма в FIFO.

g) Принятые не-IN пакеты данных (PKTSTS в GRXSTSR ≠ 0b0010) игнорируются.

h) Ядро выдаёт прерывание XFRC по каждому чтению состояния принятого пакета.

i) По прерыванию XFRC выключаем канал и стопорим запись дальнейших запросов в OTG_FS_HCCHAR2. Ядро пишет запрос выключения канала в не-периодическую очередь сразу после записи в регистр OTG_FS_HCCHAR2.

j) Ядро выдаёт прерывание RXFLVL сразу записи состояния останова в FIFO.

k) Читаем и игнорируем состояния приёма пакета.

l) Ядро выдаёт прерывание CHH сразу после извлечения состояния останова из FIFO.

m) По прерыванию CHH отдаём канал для новых операций.

n) Обрабатываем не-ACK ответы

Управляющие передачи

Управляющие передачи Setup, Data и Status передаются раздельно. Передача OUT Setup, Data или Status подобна групповой передаче OUT. Передача IN Data или Status подобна групповой передаче IN. Для всех трёх передач в поле EPTYP регистра OTG_FS_HCCHAR1 пишут "Управляющая". Для передач Setup в поле PID регистра OTG_FS_HCTSIZ1 пишут SETUP.

Прерывающие передачи OUT

Они показаны на рисунке ниже. Полагаем, что:

– Передаётся 1 пакет в каждом фрейме (до 1 максимального размера), начиная с нечётного фрейма (размер = 1024 байта)

– Периодический передающий FIFO может хранить один пакет (1 KB)

– Глубина периодической очереди равна = 4

Последовательность операций такова:

a) Инициализируем и включаем канал 1. Бит ODDFRM в OTG_FS_HCCHAR1 должен стоять.

b) Пишем первый пакет канала 1.

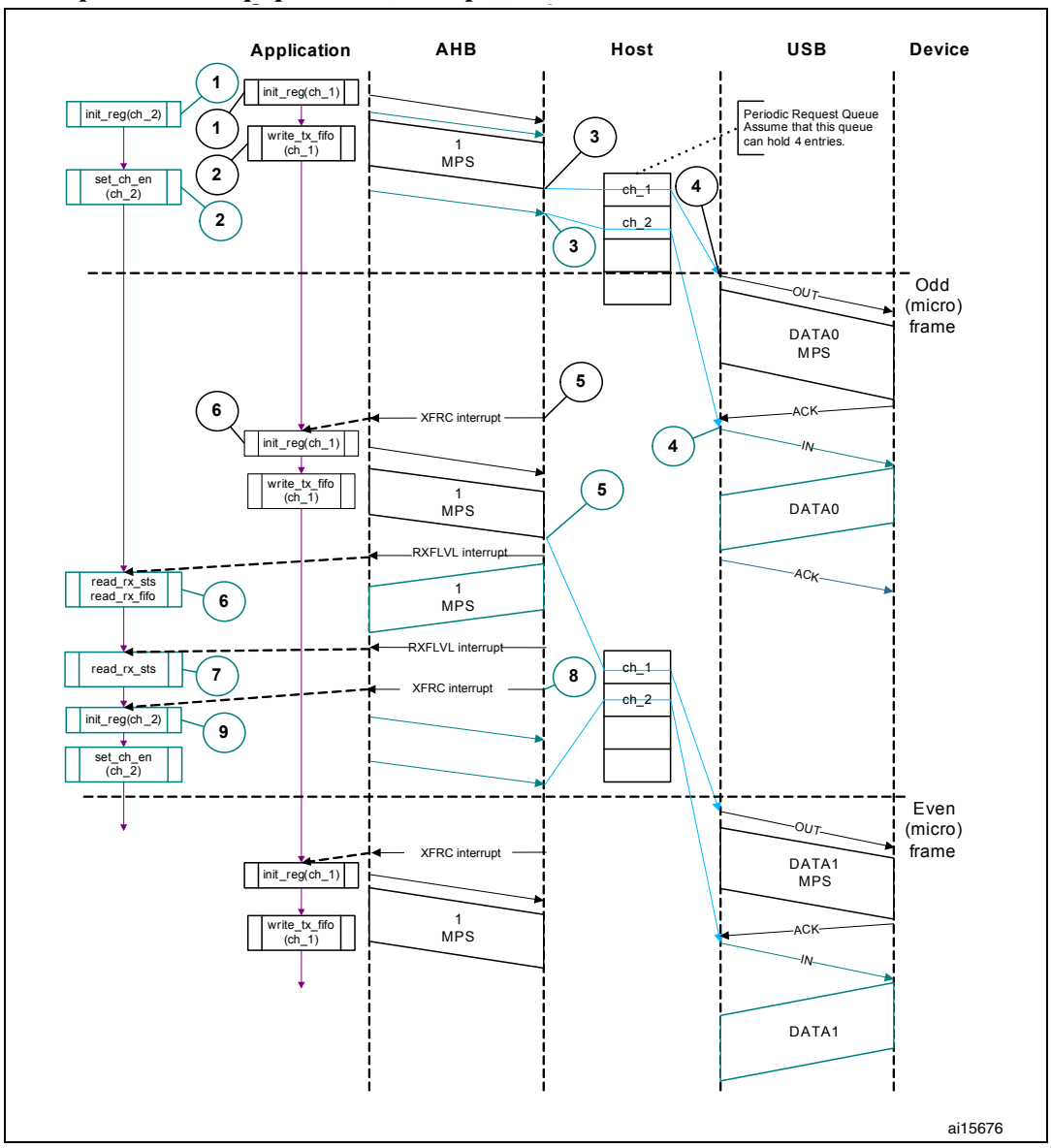
c) Вместе с записью последнего слова каждого пакета хост OTG_FS пишет запрос в периодическую очередь.

d) Хост OTG_FS пытается послать токен OUT в следующем (нечётном) фрейме.

e) Прерывание XFRC выдаётся после успешной передачи последнего пакета.

f) После прерывания XFRC инициализируем канал для следующей передачи.

Нормальные прерывающие передачи OUT/IN



• **Программа обработки прерываний для прерывающих передач OUT/IN**

а) Прерывающие OUT

Место в передающем FIFO определяется по прерыванию `NPTXFE` в `OTG_FS_GINTSTS`.

```
Unmask (NAK/TXERR/STALL/XFRC/FRMOR)
if (XFRC) {
    Reset Error Count
    Mask ACK
    De-allocate Channel
    }
else
    if (STALL or FRMOR) {
        Mask ACK
        Unmask CHH
        Disable Channel
        if (STALL) {
                Transfer Done = 1
                }
        }
    else
        if (NAK or TXERR) {
            Rewind Buffer Pointers
            Reset Error Count
            Mask ACK
            Unmask CHH
            Disable Channel
```

```
        }
    else
        if (CHH) {
            Mask CHH
            if (Transfer Done or (Error_count == 3)) {
                De-allocate Channel
                }
            else  {
                Re-initialize Channel (in next b_interval - 1 Frame)
                }
            }
    else
            if (ACK)  {
            Reset Error Count
            Mask ACK
            }
```

b) Прерывающие IN

```
Unmask (NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)
if (XFRC) {
    Reset Error Count
    Mask ACK
    if (OTG_FS_HCTSIZx.PKTCNT == 0 {
        De-allocate Channel
        }
    else {
        Transfer Done = 1
        Unmask CHH
        Disable Channel
        }
    }
else
    if (STALL or FRMOR or NAK or DTERR or BBERR)  {
        Mask ACK
        Unmask CHH
        Disable Channel
        if (STALL or BBERR)  {
                Reset Error Count
                Transfer Done = 1
                }
        else
            if (!FRMOR) {
                Reset Error Count
                }
        }
else
    if (TXERR)  {
        Increment Error Count
        Unmask ACK
        Unmask CHH
        Disable Channel
        }
    else
        if (CHH)  {
            Mask CHH
            if (Transfer Done or (Error_count == 3))  {
                De-allocate Channel
                }
            else
                Re-initialize Channel (in next b_interval - 1 /Frame)
            }
        }
    else
        if (ACK)  {
            Reset Error Count
            Mask ACK  }
```

- **Прерывающие передачи IN**

Полагаем, что:

- Принимаем один пакет максимальной длины в каждом фрейме, начиная с нечётного (размер = 1024 байта).
- Приёмный FIFO может хранить один максимальный пакет и два слова состояния на один пакет (1031 байт).
- Глубина периодической очереди = 4.

- **Нормальная прерывающая операция IN**

Последовательность операций такова:

a) Инициализируем канал 2. Бит ODDFRM в OTG_FS_HCCHAR2 должен стоять.
b) Ставим бит CHENA в OTG_FS_HCCHAR2 для записи запроса IN в периодическую очередь.
c) Хост пишет запрос IN в очередь по каждой установке бита CHENA в OTG_FS_HCCHAR2.
d) Хост OTG_FS пробует послать токен IN в следующем (нечётном) фрейме.
e) После приёма пакета IN в FIFO, хост OTG_FS выдаёт прерывание RXFLVL.
f) По прерыванию RXFLVL в слове состояния узнаём число принятых байтов и читаем FIFO. Перед чтением FIFO прерывание RXFLVL надо маскировать и открывать его после извлечения всего пакета.
g) После получения в FIFO строки статуса выдаётся прерывание RXFLVL. Принятые не-IN пакеты данных (PKTSTS в GRXSTSR ≠ 0b0010) надо читать и игнорировать.
h) После чтения строки статуса выдаётся прерывание XFRC.
i) По прерыванию XFRC читаем поле PKTCNT регистра OTG_FS_HCTSIZ2. Если бит PKTCNT в OTG_FS_HCTSIZ2 стоит, то выключаем канал, иначе инициализируем канал для следующей передачи. При этом бит ODDFRM в OTG_FS_HCCHAR2 надо сбросить.

- **Изохронные передачи OUT**

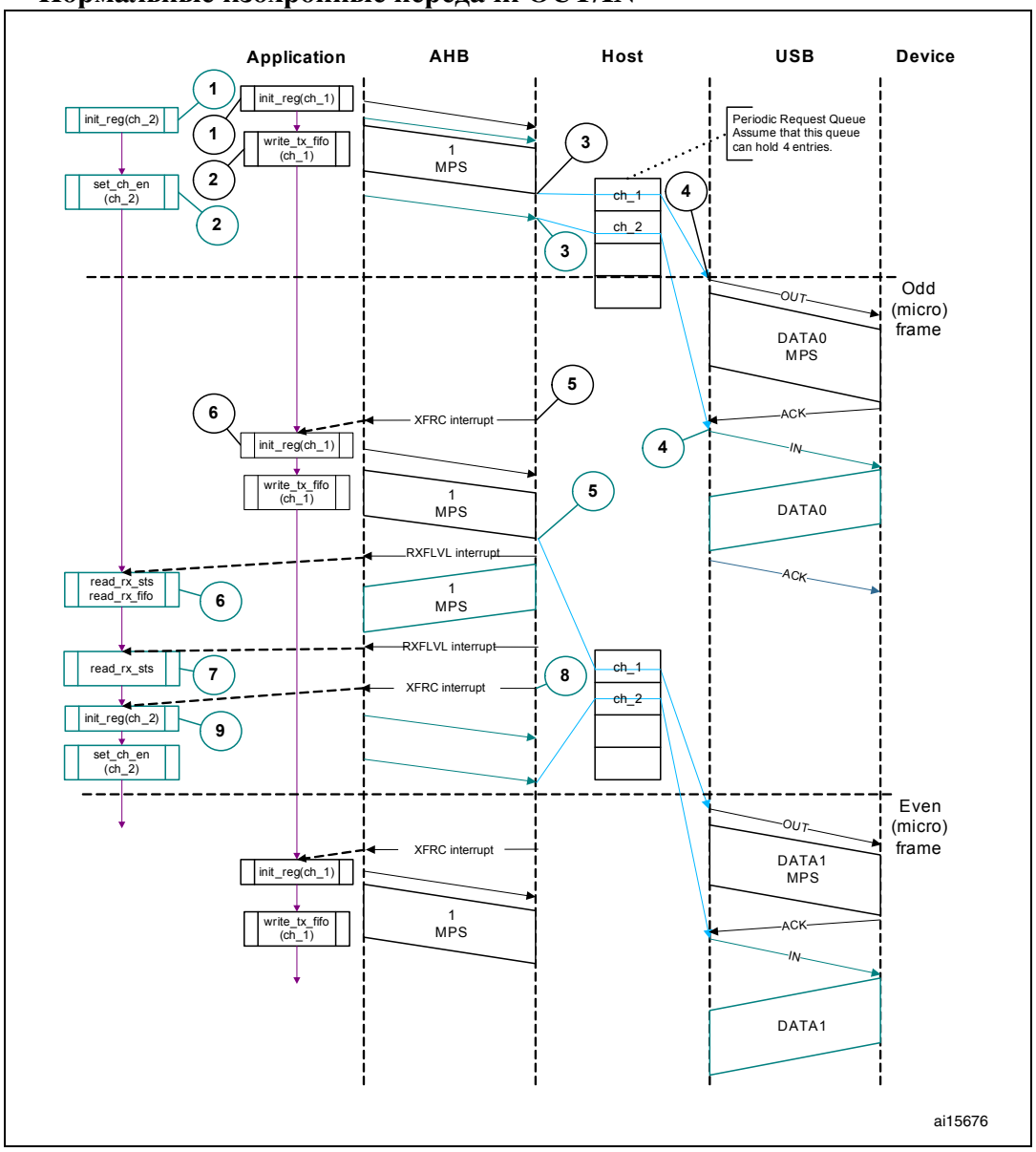
Пример показан на рисунке ниже. Полагаем, что:

- Посылается один пакет максимальной длины в каждом фрейме, начиная с нечётного (размер = 1024 байта).
- Периодический передающий FIFO может содержать один пакет (1 KB).
- Глубина периодической очереди = 4.

Последовательность операций такова:

a) Инициализируем канал 1. Бит ODDFRM в OTG_FS_HCCHAR1 должен стоять.
b) Пишем первый пакет канала 1.
c) Вместе с записью последнего слова пакета хост OTG_FS пишет запрос в периодическую очередь.
d) Хост OTG_FS пробует послать токен OUT в следующем (нечётном) фрейме.
e) Прерывание XFRC выдаётся после успешной передачи последнего пакета.
f) После него инициализируем канал для следующей передачи.
g) Обрабатываем не-ACK ответы.

Нормальные изохронные передачи OUT/IN



- **Программа обработки прерываний для изохронных передач OUT/IN**

Изохронные OUT

```
Unmask (FRMOR/XFRC)
if (XFRC) {
    De-allocate Channel
    }
else
    if (FRMOR)  {
        Unmask CHH
        Disable Channel
        }
    else
        if (CHH)  {
            Mask CHH
            De-allocate Channel
            }
```

Изохронные IN

```
Unmask (TXERR/XFRC/FRMOR/BBERR)
if (XFRC or FRMOR)  {
    if (XFRC and (OTG_FS_HCTSIZx.PKTCNT == 0))  {
        Reset Error Count
        De-allocate Channel
        }
    else  {
        Unmask CHH
        Disable Channel
```

```
    }
}
else
    if (TXERR or BBERR) {
        Increment Error Count
        Unmask CHH
        Disable Channel
        }
    else
        if (CHH) {
            Mask CHH
            if (Transfer Done or (Error_count == 3)) {
                De-allocate Channel
                }
            else {
                Re-initialize Channel
                }
            }
```

- **Изохронные передачи IN**

Полагаем, что:

- Принимаем один пакет максимальной длины в каждом фрейме, начиная с нечётного (размер = 1024 байта).
- Приёмный FIFO может содержать 1 максимальный пакет и два слова статуса на каждый пакет (1031 байт).
- Глубина периодической очереди = 4.

Последовательность операций такова:

a) Инициализируем канал 2. Бит ODDFRM в OTG_FS_HCCHAR2 должен стоять.
b) Ставим CHENA в OTG_FS_HCCHAR2 для записи запроса IN в периодическую очередь.
c) Запрос IN пишется в очередь по каждой установке бита CHENA в OTG_FS_HCCHAR2.
d) Хост пробует послать токен IN в следующем нечётном фрейме.
e) После приёма пакета IN в FIFO выдаётся прерывание RXFLVL.
f) По нему из состояния принятого пакета берём число полученных байтов и читаем FIFO. Перед чтением FIFO маскируем прерывание RXFLVL и открываем его после извлечения всего пакета.
g) Прерывание RXFLVL выдаётся после приёма строки состояния пакета в FIFO. Строки состояния не-IN пакетов данных (бит PKTSTS и OTG_FS_GRXSTSR ≠ 0b0010) читаются и игнорируются.
h) Прерывание XFRC выдаётся после чтения из FIFO строки состояния пакета.
i) По нему читаем поле PKTCNT регистра OTG_FS_HCTSIZ2. Если PKTCNT≠ 0, то выключаем канал, иначе инициализируем его для следующей передачи. В это время нужно сбросит бит ODDFRM в регистре OTG_FS_HCCHAR2.

- **Выбор глубины очереди**

Глубина периодической и не-периодической очередей должна соответствовать числу одновременно работающих соответствующих конечных точек.

Чем глубже не-периодической очереди (с соответствующим размером FIFO), тем чаще возможны не-периодические передачи.

Глубокая периодическая очередь позволяет планировать периодические передачи. Если глубина меньше запланированного числа периодических передач в микрофрейме, то возникает переполнение фрейма.

- **Обработка ошибки перекрёстных шумов**

Две типа ошибки: пакета и порта.

Ошибка пакета появляется когда устройство посылает данных больше, чем максимальный размер пакета канала. Ошибка порта появляется когда ядро продолжает принимать данные от устройства после EOF2 (конец фрейма 2, очень близкого к SOF).

Когда контроллер OTG_FS обнаруживает ошибку пакета, он перестаёт писать данные в буфер Rx и ждёт конца пакета (EOP). После обнаружения EOP он сливает уже записанные в Rx данные и выдаёт прерывание ошибки перекрёстных шумов.

Когда контроллер OTG_FS обнаруживает ошибку порта, он сливает RxFIFO и выключает порт. Выдаётся прерывание выключения порта (HPRTINT в OTG_FS_GINTSTS, PENCHNG в OTG_FS_HPRT). Программа по биту POCA in OTG_FS_HPRT, определяет не было ли это вызвано перегрузкой порта и выполняет программный сброс порта. После ошибки порта токены больше не посылаются.

28.17.5. Программная модель устройства

Инициализация по сбросу USB

1. Ставим бит NAK для всех конечных точек OUT
   - SNAK = 1 в OTG_FS_DOEPCTLx
2. Демаскируем прерывания:
   - INEP0 = 1 в OTG_FS_DAINTMSK (точки 0 IN)
   - OUTEP0 = 1 в OTG_FS_DAINTMSK (точки 0 OUT)
   - STUP = 1 в DOEPMSK
   - XFRC = 1 в DOEPMSK
   - XFRC = 1 в DIEPMSK
   - TOC = 1 в DIEPMSK
3. Определяем RAM для всех FIFO данных
   - В регистре OTG_FS_GRXFSIZ обеспечиваем возможность принимать управляющие OUT и SETUP. Если контроль порога не включён, то как минимум размер должен быть равен 1 максимальному пакету точки 0 + 2 слова (для состояния управляющего пакета данных OUT) + 10 слов (для пакетов SETUP).
   - В регистре OTG_FS_TX0FSIZ обеспечиваем возможность (в зависимости от номера выбранного FIFO) передавать управляющие данные IN. Как минимум размер должен быть равен 1 максимальному пакету конечной точки 0.
4. Пишем регистры конечной точки 0 OUT для приёма пакетов SETUP
   - STUPCNT = 3 в OTG_FS_DOEPTSIZ0 (для приёма череды 3 пакетов SETUP)

Всё, пакеты SETUP можно принимать.

Инициализация конечной точки по завершению переучёта

1. По прерыванию ENUMDNE в регистре OTG_FS_GINTSTS, из регистра OTG_FS_DSTS выясняем скорость переучёта.
2. В поле MPSIZ регистра OTG_FS_DIEPCTL0 пишем максимальный размер пакета точки 0. Он зависит от скорости переучёта.

Отныне устройство может принимать пакеты SOF и передавать по точке 0.

Инициализация конечной точки по команде SetAddress

После приёма команды **SetAddress** в пакете SETUP:

1. В регистр OTG_FS_DCFG пишем полученный в команде **SetAddress** адрес
2. Вынуждаем ядро послать пакет состояния IN.

Инициализация конечной точки по команде SetConfiguration/SetInterface

После приёма команд **SetConfiguration** или **SetInterface** в пакете SETUP:

1. По команде **SetConfiguration** в регистрах конечных точек пишем новую конфигурацию.
2. По команде **SetInterface** пишем регистры затронутых командой конечных точек.

3. Некоторые точки в новой конфигурации и установках становятся нерабочими. Деактивируем такие точки.
4. В регистре the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the OTG_FS_DAINTMSK демаскируем прерывания активных точек и маскируем прерывания неактивных конечных точек.
5. Определяем RAM для всех FIFO.
6. Вынуждаем ядро послать пакет состояния IN.

Отныне устройство может принимать и передавать любые пакеты данных.

Активация конечной точки

Новый тип конечной точки задаём так:

1. В регистре OTG_FS_DIEPCTLx (для IN или двунаправленных точек) или регистре OTG_FS_DOEPCTLx (для OUT или двунаправленных точек) пишем поля:
   - Максимальный размер пакета
   - Активная точка USB = 1
   - Начальное значение переключения данных точки (прерывающие и групповые)
   - Тип точки
   - Номер TxFIFO

Теперь точка активирована и ядро начинает рассматривать получаемые точкой токены и отсылать нужные ответы.

Деактивация конечной точки

Делаем так:

1. Снимаем бит активной точки USB в В регистре OTG_FS_DIEPCTLx (для IN или двунаправленных точек) или регистре OTG_FS_DOEPCTLx (для OUT или двунаправленных точек).

Теперь ядро игнорирует направленные этой точке токены, что приводит к таймауту USB.

NB: Программа должна снять биты NPTXFEM и RXFLVLM в регистре OTG_FS_GINTMSK.

28.17.6. Модель работы

Передачи SETUP и OUT данных

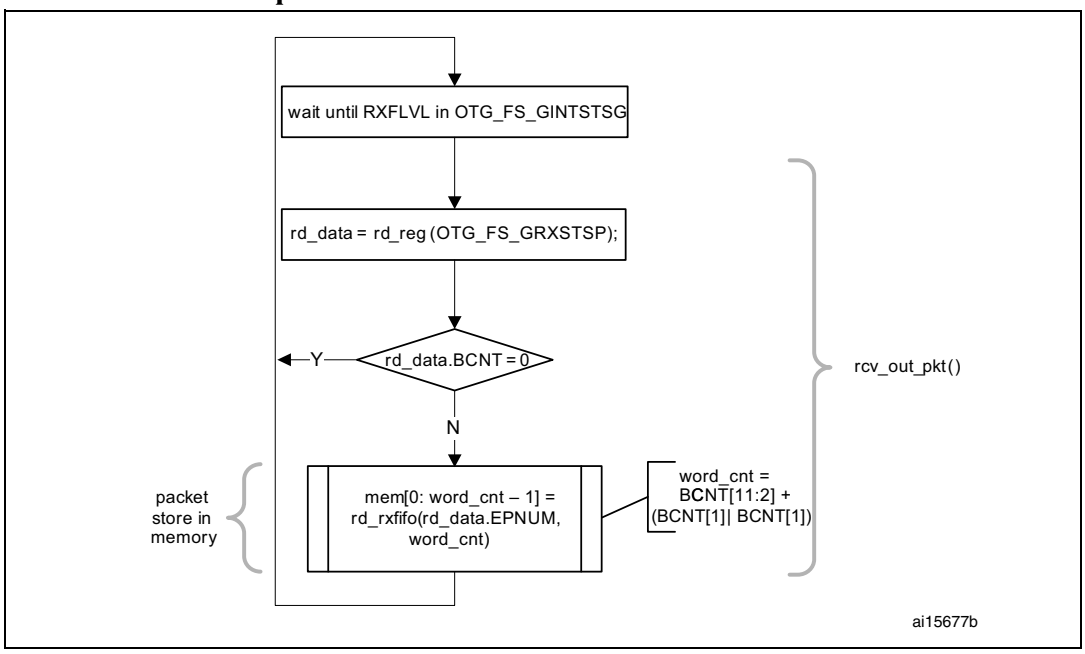
• **Чтение данных пакета**

1. По прерыванию RXFLVL регистра OTG_FS_GINTSTS читаем регистр состояния приёма (OTG_FS_GRXSTSP).
2. Записью RXFLVL = 0 (в OTG_FS_GINTMSK) маскируем прерывание RXFLVL регистра OTG_FS_GINTSTS вплоть до извлечения пакета из FIFO.
3. Если счётчик байтов пакета не равен 0, то извлекаем байты из FIFO в память.
4. Статус принятого в FIFO пакета может показывать:
   a) Действует Глобальный OUT NAK:
   PKTSTS = Глобальный OUT NAK, BCNT = 0x000, EPNUM = Всё Равно (0x0),
   DPID = Всё Равно (0b00).
   b) Пакет SETUP:
   PKTSTS = SETUP, BCNT = 0x008, EPNUM = Номер управляющей EP, DPID = D0.
   c) Конец фазы SETUP:
   PKTSTS = Конец фазы SETUP, BCNT = 0x0, EPNUM = Номер управляющей EP,
   DPID = Всё Равно (0b00).
   После чтения этой строки из FIFO выдаётся прерывание SETUP этой конечной точки.
   d) Пакет данных OUT:
   PKTSTS = DataOUT, BCNT = Размер пакета (0 ≤ BCNT ≤ 1 024), EPNUM = Номер точки,
   DPID = PID данных.

e) Передача данных завершена:
PKTSTS = Передача данных завершена, BCNT = 0x0, EPNUM = Номер точки,
DPID = Всё Равно (0b00).
После чтения этой строки из FIFO выдаётся прерывание Конца передачи этой точки.

5. После извлечения данных пакета из FIFO надо демаскировать прерывание RXFLVL в регистре (OTG_FS_GINTSTS).
6. Шаги 1–5 повторяются по каждому прерыванию RXFLVL в OTG_FS_GINTSTS. Чтение пустого приёмного FIFO сводит ядро с ума.

Чтение пакета приёмного FIFO



- **Передачи SETUP**

Программные требования

1. При ненулевом поле STUPCNT в регистре OTG_FS_DOEPTSIZx принимаемые пакеты SETUP точки OUT пишутся в FIFO независимо от состояния NAK и бита EPENA в регистре OTG_FS_DOEPCTLx. Поле STUPCNT декрементируется после каждого приёма пакета SETUP. При неверном значении поля STUPCNT пакеты принимаются и поле декрементируется, но определить число присланных пакетов невозможно.
   - STUPCNT = 3 в OTG_FS_DOEPTSIZx
2. В приёмном FIFO данных управляющей точки всегда надо отводить место для приёма трёх пакетов SETUP.
   - Резервируется 10 слов. Три слова для первого пакета SETUP, 1 слово для статуса Конца фазы SETUP и 6 слов для ещё двух пакетов SETUP всех управляющих точек.
   - В 3 словах пакета SETUP хранятся 8 байтов данных и 4 байта статуса.
   - Это место используется ядром только для данных SETUP.
3. Из FIFO надо читать 2 слова пакета SETUP.
4. Слово статуса Конца фазы SETUP надо выбрасывать.

Внутренний ход данных

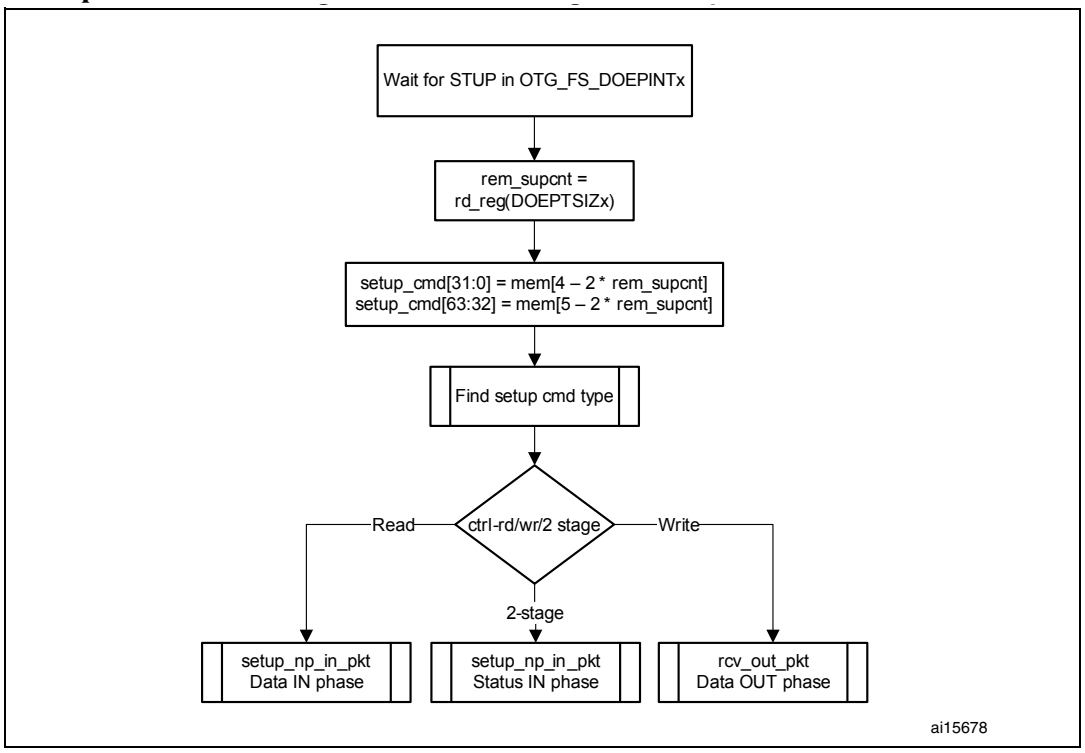
1. Принимаемые пакеты SETUP пишутся в FIFO без проверки доступного места и независимо от состояния NAK и бита STALL.
   - В точках IN/OUT, принявших пакет SETUP, ядро ставит биты IN NAK и OUT NAK.
2. По каждому пакету SETUP декрементируется поле STUPCNT и в FIFO пишутся 3 слова:
   - В первом слове информация для ядра
   - Во втором слове первые 4 байта команды SETUP
   - В третьем слове последние 4 байта команды SETUP
3. При изменении фазы SETUP на IN/OUT данных ядро пишет в FIFO слово Конца фазы SETUP.
4. На стороне AHB пакеты SETUP опустошаются программой.

5. По чтению из FIFO слова Конца фазы SETUP выдаётся прерывание STUP OTG_FS_DOEPINTx.
   - Ядро снимает бит разрешения управляющей точки OUT.

Программная последовательность

1. В регистр OTG_FS_DOEPTSIZx пишем STUPCNT = 3
2. Ждём прерывания RXFLVL (OTG_FS_GINTSTS) и читаем данные пакета из FIFO.
3. Прерывание STUP (OTG_FS_DOEPINTx) отмечает завершение передачи данных SETUP.
   - По нему из регистра OTG_FS_DOEPTSIZx узнаём число принятых пакетов SETUP и обрабатываем последний.

Обработка пакета SETUP



Обработка череды более 3 пакетов SETUP

Обычно при ошибке пакета SETUP хост не посылает подряд более трёх пакетов SETUP одной точке, но при нормальной работе это не исключено. В этом случае контроллер OTG_FS выдаёт прерывание B2BSTUP в регистре OTG_FS_DOEPINTx.

- **Установка Глобального OUT NAK**

Внутренний ход данных

1. При установке бита SGONAK в регистре OTG_FS_DCTL в FIFO пишутся только пакеты SETUP, не-изохронные пакеты OUT получают ответ NAK, изохронные пакеты OUT игнорируются.
2. В приёмный пишется слово Глобального OUT NAK. Места должно хватать.
3. При его чтении из FIFO выдаётся прерывание GONAKEFF в регистре OTG_FS_GINTSTS.
4. Теперь можно снять бит SGONAK в регистре OTG_FS_DCTL.

Программная последовательность

1. Пишем SGONAK = 1 в регистре OTG_FS_DCTL
2. Ждём прерывания GONAKEFF в регистре OTG_FS_GINTSTS.
3. Между установкой бита SGONAK в OTG_FS_DCTL и прерыванием GONAKEFF (OTG_FS_GINTSTS) нормальные пакеты OUT принимаются.
4. Это прерывание можно временно маскировать записью GINAKEFFM = 0 в регистре OTG_FS_GINTMSK
5. Из режима Глобального OUT NAK выходят снятием бита SGONAK в регистре OTG_FS_DCTL. Заодно очищается прерывание GONAKEFF (OTG_FS_GINTSTS).
6. Если оно было ранее маскировано, то демаскируем его.

- **Выключение точки OUT**

Программная последовательность

1. Сначала включаем Глобальный OUT NAK. SGONAK = 1 в OTG_FS_DCTL
2. Ждём прерывания GONAKEFF (OTG_FS_GINTSTS)
3. Выключаем точку OUT записью:
   - EPDIS = 1 в OTG_FS_DOEPCTLx
   - SNAK = 1 в OTG_FS_DOEPCTLx
4. Ждём прерывания EPDIS (OTG_FS_DOEPINTx), автоматически снимаются биты EPDIS = 0 и EPENA = 0 в регистре OTG_FS_DOEPCTLx
5. Чтобы принимать данные по другим точка OUT выходим из режима Глобального OUT NAK снятием бита SGONAK в регистре OTG_FS_DCTL.

- **Общие не-изохронные передачи данных OUT**

Программная последовательность

1. Для начала размещаем приёмный буфер в памяти.
2. Размер передачи точки должен быть кратен максимальному размеру пакета точки, выравненному на границу слова.
   - размер передачи[EPNUM] = $n \times$ (MPSIZ[EPNUM] + 4 – (MPSIZ[EPNUM] mod 4))
   - счётчик пакетов[EPNUM] = n
   - n>0
3. Размер переданных данных вычисляется по регистру размера передачи.
   - Принято данных = начальный размер – остаток в регистре
   - Число пакетов с принятыми данными = начальное число – остаток в регистре

Внутренний ход данных

1. Сначала в регистрах точки ставим размер передачи и счётчик пакетов, снимаем бит NAK и включаем точку на приём.
2. После снятия бита NAK ядро начинает писать данные в свободное место FIFO. Вместе с данными в FIFO пишется статус пакета и декрементируется счётчик пакетов.
   - Пакеты OUT с ошибкой CRC сливаются из приёмного FIFO автоматически.
   - Повторно посылаемые чередой не-изохронные пакеты OUT той же точке с тем же PID от хоста, который не воспринимает ответ ACK, отбрасываются ядром и счётчик пакетов не декрементируется.
   - При нехватке места в FIFO изохронные и не-изохронные пакеты данных игнорируются, на не-изохронные токены OUT выдаётся ответ NAK.
   - Во всех трёх случаях счётчик пакетов не декрементируется.
3. При обнулении счётчика пакетов или по приёму короткого пакета ставится бит NAK конечной точки и изохронные и не-изохронные пакеты данных игнорируются, на не-изохронные токены OUT выдаётся ответ NAK.
4. Из приёмного FIFO данные извлекается пакетами, по одному пакету за раз.
5. Размер передачи конечной точки уменьшается на размер пакета после его извлечения из FIFO.
6. Строка конца передачи данных OUT пишется в приёмный FIFO после:
   - Обнулении размера передачи и счётчика пакетов
   - Последний принятый в FIFO пакет OUT был коротким (0 ≤ размер < максимума)
7. При извлечении этой строки из FIFO выдаётся прерывание и точка выключается.

Программная последовательность

1. В регистр OTG_FS_DOEPTSIZx пишем размер передачи и счётчик пакетов.
2. В регистре OTG_FS_DOEPCTLx пишем характеристки точки и ставим биты EPENA и CNAK.
3. Ждём прерывания RXFLVL (в OTG_FS_GINTSTS) и извлекаем данные из FIFO.
   - Этот шаг может повторяться несколько раз в зависимости от размера передачи.
4. Прерывание XFRC (OTG_FS_DOEPINTx) отмечает конец не-изохронной передачи данных OUT.
5. По регистру OTG_FS_DOEPTSIZx определяем длину принятых данных.

• Общие изохронные передачи данных OUT

Программные требования

1. Требования такие же, как и для не-изохронных передач OUT.
2. Размер передачи и счётчик пакетов всегда должны быть равны ожидаемому числу пакетов максимальной длины в одном фрейме и не больше. Изохронные передачи OUT не могут занимать более 1 фрейма.
3. Данные и статус изохронных передач OUT надо прочесть из FIFO до конца периодического фрейма (прерывание EOPF в регистре OTG_FS_GINTSTS).
4. Для приёма следующего фрейма изохронную точку OUT надо включить после EOPF (OTG_FS_GINTSTS) и до SOF (OTG_FS_GINTSTS).

Внутренний ход данных

1. Он немного отличается от хода данных не-изохронных точек OUT.
2. При включении изохронной точки OUT установкой бита EPENA и очисткой бита NAK, бит Чётного/Нечётного фрейма должен быть установлен правильно. Ядро принимает изохронные данные отдельным фреймом только при:
   - EONUM (в OTG_FS_DOEPCTLx) = SOFFN[0] (в OTG_FS_DSTS)
3. После чтения данных и статуса пакета из FIFO в поле RXDPID регистра OTG_FS_DOEPTSIZx пишется PID данных последнего принятого пакета.

Программная последовательность

1. В регистр OTG_FS_DOEPTSIZx пишем размер передачи и счётчик пакетов.
2. В регистре OTG_FS_DOEPTSIZx ставим характеристики точки и пишем EPENA = 1, CNAK=1 и EONUM = (0: Чётный/1: Нечётный)
3. Ждём прерывания RXFLVL (в OTG_FS_GINTSTS) и извлекаем данные пакета из FIFO
   - Этот шаг может повторяться несколько раз в зависимости от размера передачи.
4. Прерывание XFRC (OTG_FS_DOEPINTx) отмечает конец изохронной передачи данных OUT. При этом данные не обязательно будут достоверными.
5. Для изохронных передач OUT это прерывание выдаётся не всегда, лучше использовать прерывание IISOOXFRM в регистре OTG_FS_GINTSTS.
6. По регистру OTG_FS_DOEPTSIZx определяем длину принятых данных. Данные достоверны когда:
   - RXDPID = D0 (в OTG_FS_DOEPTSIZx) и число пакетов с данными = 1
   - RXDPID = D1 (в OTG_FS_DOEPTSIZx) и число пакетов с данными = 2
   - RXDPID = D2 (в OTG_FS_DOEPTSIZx) и число пакетов с данными = 3

   Число пакетов с данными = Начальное число пакетов – Конечное число пакетов.

   Недостоверные пакеты можно выбрасывать.

• Незавершённые изохронные передачи данных OUT

Внутренний ход данных

1. У изохронных точек OUT прерывание XFRC (в OTG_FS_DOEPINTx) не выдаётся когда:
   - Приёмный FIFO не вмещает весь пакет данных ISO OUT и он отбрасывается,
   - Возникла ошибка CRC принятого пакета,
   - Принятый токен OUT нарушен,
   - Программа не успевает читать данные из FIFO.
2. При появлении конца периодического фрейма до завершения передач всех изохронных точек OUT ядро выдаёт прерывание IISOOXFRM в регистре OTG_FS_GINTSTS, говоря, что не для всех точек выдавалось прерывание XFRC (в OTG_FS_DOEPINTx). Точка с незавершённой передачей остаётся включённой, но передачи по ней нет.

Программная последовательность

1. Прерывание IISOOXFRM (OTG_FS_GINTSTS) означает, что есть хоть одна изохронная точка OUT с незавершенной передачей.

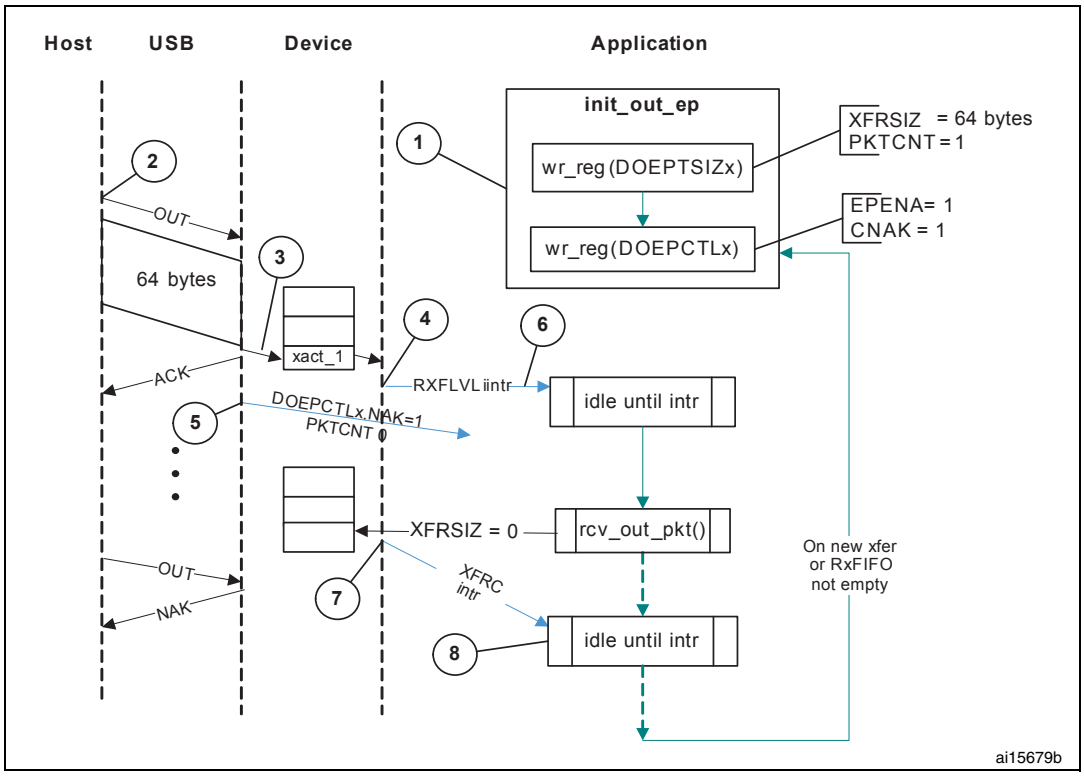
2. Если оно возникло из-за не прочитанного FIFO, то из FIFO надо извлечь все данные и статус. При этом может выдаться прерывание XFRC (OTG_FS_DOEPINTx). В этом случае надо снова включить точку для приёма изохронных данных OUT в следующем фрейме.
3. По прерыванию IISOOXFRM (OTG_FS_GINTSTS) в регистрах OTG_FS_DOEPCTLx надо определить конечные точки с незавершёнными передачами. При этом:
   - Бит EONUM (в OTG_FS_DOEPCTLx) = SOFFN[0] (в OTG_FS_DSTS)
   - EPENA = 1 (в OTG_FS_DOEPCTLx)
4. Предыдущий шаг надо выполнить до прерывания SOF (в OTG_FS_GINTSTS), чтобы номер текущего фрейма не изменился.
5. У изохронных точек OUT с незавершёнными передачами надо отбросить данные и выключить точку установкой бита EPDIS в регистре OTG_FS_DOEPCTLx.
6. Ждём прерывания EPDIS (в OTG_FS_DOEPINTx) и включаем точку для следующего фрейма.
   - Из-за задержки выключения точки следующий фрейм может быть не получен.

• Остановка не-изохронной точки OUT

1. Ставим ядро в режим Глобального OUT NAK.
2. Выключаем точку
   - Вместо установки бита SNAK в OTG_FS_DOEPCTL, ставим STALL = 1 (OTG_FS_DOEPCTL). Бит STALL старше бита а NAK.
3. При готовности отмены ответов STALL снимаем бит STALL в OTG_FS_DOEPCTLx.
4. При изменении бита STALL по командам **SetFeature.Endpoint Halt** или **ClearFeature.Endpoint Halt** сделать это надо до фазы передачи Статуса управляющей конечной точки.

Примеры

• Групповая точка OUT



После команд **SetConfiguration/SetInterface** надо инициировать все конечные точки OUT установкой CNAK = 1 и EPENA = 1 (в OTG_FS_DOEPCTLx) и записью подходящих XFRSIZ и PKTCNT в регистрах OTG_FS_DOEPTSIZx.

1. Хост пробует передавать токен OUT по конечной точке.
2. По нему ядро пишет пакет в доступное место RxFIFO.
3. По завершению записи ядро выдаёт прерывание RXFLVL (в OTG_FS_GINTSTS).
4. После приёма PKTCNT пакетов ядро ставит точке бит NAK для задержки приёма.
5. Программа обрабатывает прерывание и читает данные из RxFIFO.

6. По прочтению всех данных (равного XFRSIZ), ядро выдаёт прерывание XFRC в регистре OTG_FS_DOEPINTx.
7. Программа обрабатывает конец передачи.

Передачи данных IN

- **Запись пакета**

1. Работать можно по прерываниям или опросом.
   - При опросе программа по регистру OTG_FS_DTXFSTSx следит за наличием свободного места в передающем FIFO данных.
   - В режиме прерываний программа ждёт прерывания TXFE (в OTG_FS_DIEPINTx) и по регистру OTG_FS_DTXFSTSx определяет наличие места в FIFO данных для пакета ненулевой длины.
   - Для пакета нулевой длины места в FIFO не требуется.
2. Перед записью в FIFO данных надо записать регистр управления точки (OTG_FS_DIEPCTLx) командой чтение-модификация-запись во избежание изменения регистра (исключая установку бита включения точки).

При наличии места в передающем FIFO можно писать несколько пакетов для одной точки. Для периодических точек IN надо писать по одному пакету на микрофрейм. Новый пакет можно писать только после подтверждения завершения предыдущей передачи.

- **Установка NAK точки IN**

Внутренний ход данных

1. При установке IN NAK точки ядро прекращает передачи независимо от наличия места в наличии данных в передающем FIFO точки.
2. Не-изохронные токены IN получают ответ NAK
   - Изохронные токены IN получают ответ с нулевой длиной данных
3. В ответ на бит SNAK в OTG_FS_DIEPCTLx ядро выдаёт прерывание INEPNE (действует NAK точки IN) в OTG_FS_DIEPINTx.
4. Снимается режим IN NAK установкой бита CNAK в регистре OTG_FS_DIEPCTLx.

Программная последовательность

1. Для остановки передач по точке IN программа пишет SNAK = 1 в OTG_FS_DIEPCTLx
2. Ждём подтверждения остановки по прерыванию INEPNE в OTG_FS_DIEPINTx.
3. В промежутке между установкой бита NAK и прерыванием INEPNE ядро может передавать данные IN.
4. Программа маскировать это прерывание записью INEPNEM = 0 в регистр DIEPMSK
5. Из режима NAK точки выходят снятием бита NAKSTS в OTG_FS_DIEPCTLx (запись CNAK = 1 in OTG_FS_DIEPCTLx). Этим также чистится прерывание INEPNE (в OTG_FS_DIEPINTx).
6. Ранее маскированное прерывание демаскируют записью INEPNEM = 1 in DIEPMSK.

- **Выключение точки IN**

Программная последовательность

1. Программа перестаёт писать данные для выключаемой точки IN.
2. Ставим режим NAK записью SNAK = 1 в OTG_FS_DIEPCTLx.
3. Ждём прерывания INEPNE в OTG_FS_DIEPINTx.
4. В регистр OTG_FS_DIEPCTLx пишем EPDIS = 1 и SNAK = 1.
5. Прерывание EPDISD в OTG_FS_DIEPINTx подтверждает выключение точки. Вместе с этим ядро снимает биты EPENA и EPDIS в регистре OTG_FS_DIEPCTLx.
6. У периодических IN EP число переданных данных узнают по регистру OTG_FS_DIEPTSIZx.
7. Программа должна слить данные из передающего FIFO точки записью номера точки в TXFNUM, и TXFFLSH = 1 регистра OTG_FS_GRSTCTL

Конец слива определяют по снятию ядром бита TXFFLSH опросом регистра OTG_FS_GRSTCTL. Новые данные можно передавать после включения точки.

- **Общие не-периодические передачи данных IN**

Программные требования

1. Сначала надо убедиться, что все передаваемые данные лежат в одном буфере.
2. Для передач IN размер передачи включает несколько пакетов максимальной длины и один короткий пакет, передаваемый последним.
   - Для передачи нескольких пакетов:
     Размер передачи[EPNUM] = *x* × MPSIZ[EPNUM] + sp
     Если (sp > 0), то Счётчик пакетов[EPNUM] = x + 1. Иначе, Счётчик пакетов[EPNUM] = *x*
   - Для передачи пакета без данных:
     Размер передачи[EPNUM] = 0
     Счётчик пакетов[EPNUM] = 1
   - Для передачи нескольких максимальных пакетов и пакета без данных передача разбивается на две части: пакеты с данными и отдельно пакет без данных.
     Первая передача: Размер передачи[EPNUM] = *x* × MPSIZ[EPNUM]; Счётчик пакетов = *n*;
     Вторая передача: Размер передачи[EPNUM] = 0; Счётчик пакетов = 1;
3. После включения точки на передачу ядро изменяет регистр размера передачи. В конце передачи IN по нему определяют сколько данных было передано из FIFO.
4. В FIFO извлечено = Начальный размер передачи – Остаток размера передачи
   - Передано по USB = (Начальный счётчик пакетов – Остаток счётчика пакетов) × MPSIZ[EPNUM]
   - Ещё передавать по USB = (Начальный размер передачи – Передано по USB)

Внутренний ход данных

1. В регистрах точки надо записать размер передачи и счётчик пакетов и включить точку на передачу.
2. Конечно же, данные надо записать в FIFO точки.
3. После записи пакета в FIFO размер передачи уменьшается на размер пакета. Данные пишутся в FIFO до обнуления размера передачи. После записи данных в FIFO инкрементируется “счётчик пакетов в FIFO” (Это 3-бит счётчик, управляемый ядром для каждого FIFO). Для пакетов без данных в FIFO ставится особый флаг, данные не пишутся.
4. Данные из FIFO передаются ядром по токену IN. По каждому ответу ACK не-изохронного пакета IN декрементируется счётчик пакетов вплоть до его обнуления. При таймауте счётчик пакетов не декрементируется.
5. Пакет нулевой длины высылается по токену IN и счётчик пакетов декрементируется.
6. По приёму токена IN при пустом FIFO и нулевом счётчике пакетов ядро выдаёт прерывание ITTXFE, извещая, что бит не стоит NAK. Не-изохронные точки отсылают ответ NAK.
7. Ядро само отматывает указатели FIFO и прерывание таймаута не выдаётся.
8. При нулевых размере передачи и счётчике пакетов выдаётся прерывание конца передачи (XFRC) и точка выключается.

Программная последовательность

1. В регистр OTG_FS_DIEPTSIZx пишем размер передачи и счётчик пакетов.
2. В регистр OTG_FS_DIEPCTLx пишем характеристики точки и ставим биты CNAK и EPENA.
3. При передаче пакета без данных надо опрашивать регистр OTG_FS_DTXFSTSx для определения наличия места в FIFO, или использовать TXFE (в OTG_FS_DIEPINTx).

- **Общие периодические передачи данных IN**

Программные требования

1. Они совпадают с пунктами 1, 2, 3 и 4 *Общих не-периодических передач данных IN* с некоторым изменением пункта 2.
   - Передавать можно только несколько максимальных пакетов данных с возможным коротким пакетом с данными в конце передачи. При этом:
     Размер передачи[EPNUM] = x × MPSIZ[EPNUM] + sp (где $x \geq 0$, и $0 \leq sp < MPSIZ[EPNUM]$)
     Если (sp > 0), то Счётчик пакетов[EPNUM] = *x* + 1 Иначе, Счётчик пакетов[EPNUM] = *x*;
     MCNT[EPNUM] = Счётчик пакетов[EPNUM]

- Пакеты без данных передаются отдельно. При этом:
- Размер передачи[EPNUM] = 0
  Счётчик пакетов[EPNUM] = 1; MCNT[EPNUM] = Счётчик пакетов[EPNUM]

2. Программа может планировать передачу только по одному фрейму за раз.
   - (MCNT – 1) × MPSIZ ≤ XFERSIZ ≤ MCNT × MPSIZ
   - PKTCNT = MCNT (in OTG_FS_DIEPTSIZx)
   - Если XFERSIZ < MCNT × MPSIZ, то последний пакет короткий.
   - **NB**: MCNT, PKTCNT и XFERSIZ лежат в OTG_FS_DIEPTSIZx, MPSIZ лежит в OTG_FS_DIEPCTLx.
3. В FIFO должны быть записаны все передаваемые данные до приёма токена IN. При нехватке в FIFO хоть одного слова он будет считаться пустым. Тогда:
   - Изохронная точка IN передаст пакет без данных
   - Прерывающая точка IN передаст ответ NAK.

Внутренний ход данных

1. В регистрах точки пишем размер передачи и счётчик пакетов и включаем точку на передачу.
2. Также пишем в FIFO точки передаваемые данные.
3. При записи очередного пакета в FIFO размер передачи уменьшается на размер пакета. Донные пишутся до обнуления размера передачи.
4. По приёму пакета IN периодическая точка передаёт данные из FIFO. Если там нет полного пакета, то ядро выдаёт прерывание пустого TxFIFO точки.
   - Изохронная точка IN передаст пакет без данных
   - Прерывающая точка IN передаст ответ NAK.
5. Счётчик пакетов точки декрементируется когда:
   - Изохронная точка IN передаст пакет без данных
   - Прерывающая точка IN передаст ответ NAK.
   - Обнулятся размер передачи и счётчик пакетов, будет выдано прерывание завершения передачи и точка выключится.
6. Если по "Интервалу периодического фрейма" (PFIVL в OTG_FS_DCFG) ядро найдёт непустые FIFO изохронных точек IN для текущего фрейма, то оно выдаст прерывание IISOIXFR в OTG_FS_GINTSTS.

Программная последовательность

1. В регистр OTG_FS_DIEPCTLx пишем характеристики точки и ставим биты CNAK EPENA.
2. Пишем передаваемые данные следующего фрейма в FIFO.
3. Прерывание ITTXFE (в OTG_FS_DIEPINTx) покажет, что в FIFO записаны ещё не все данные.
4. Если прерывающая точка включена, то игнорируем прерывание, иначе включаем её, чтобы она передала данные по следующему токену IN.
5. Прерывание XFRC (в OTG_FS_DIEPINTx) без прерывания ITTXFE покажет успешное завершение изохронной передачи IN. При этом размер передачи и счётчик пакетов в регистре OTG_FS_DIEPTSIZx должны быть нулевыми.
6. Прерывание XFRC (в OTG_FS_DIEPINTx) независимо от наличия прерывания ITTXFE interrupt (in OTG_FS_DIEPINTx) покажет успешное завершение прерывающей передачи IN. При этом размер передачи и счётчик пакетов в регистре OTG_FS_DIEPTSIZx должны быть нулевыми.
7. Прерывание IISOIXFR в OTG_FS_GINTSTS без упомянутых прерываний покажет, что ядро не приняло хоть один периодический токен IN в текущем фрейме.

- **Незавершённые изохронные передачи данных IN**

Внутренний ход данных

1. Изохронная передача IN считается незавершённой когда:
   а) Ядро принимает нарушенный токен IN по какой-либо изохронной точке. В этом случае выдаётся прерывание IISOIXFR в OTG_FS_GINTSTS.

b) Программа не успевает записать в FIFO все передаваемые данные до получения токена IN. В этом случае выдаётся прерывание пустого TxFIFO в OTG_FS_DIEPINTx. Его можно игнорировать, поскольку оно приведёт к прерыванию IISOIXFR в OTG_FS_GINTSTS в конце периодического фрейма. В ответ на токен IN ядро пошлёт пакет без данных.

2. Запись данных в передающий FIFO надо немедленно остановить.
3. Ставим бит NAK и бит выключения точки.
4. Ядро выключает точку, снимает бит выключения и выдаёт прерывание Выключения точки.

Программная последовательность

1. Прерывание пустого TxFIFO изохронной точки можно игнорировать, поскольку оно приведёт к прерыванию IISOIXFR в OTG_FS_GINTSTS.
2. Оно извещает о незавершённой передаче на какой-либо изохронной точке IN.
3. Точка с незавершённой передачей определяется чтением Управляющих регистров всех изохронных точек IN.
4. Останавливаем запись в периодические передающие FIFO этих точек.
5. Выключаем точку записью SNAK = 1 и EPDIS = 1 в OTG_FS_DIEPCTLx
6. Выключение точки подтверждается соответствующим прерыванием в регистре OTG_FS_DIEPINTx.
   - В этот момент надо слить данные из FIFO с помощью регистра OTG_FS_GRSTCTL или переписать их включением точки для новой передачи в следующем микрофрейме.

- **Остановка не-изохронной точки IN**

Программная последовательность

1. Выключаем точку IN и затем ставим бит STALL в регистре OTG_FS_DIEPCTLx
   - Бит STALL всегда старше бита NAK
2. Прерывание Выключения точки (в OTG_FS_DIEPCTLx) извещает об этом.
3. Сливаем ассоциированный FIFO. В случае не-периодической точки надо повторно включить остальные не-периодические точки.
4. При готовности отмены ответов STALL для точки снимаем бит STALL в OTG_FS_DIEPCTLx.
5. При изменении бита STALL по командам **SetFeature.Endpoint Halt** или **ClearFeature.Endpoint Halt** изменяем его до фазы Статуса управляющей конечной точки.

Особый случай: остановка управляющей точки OUT

Если хост посылает токенов IN/OUT больше, чем предусмотрено а пакете SETUP, то их надо остановить. В этом случае, в фазе данных, после передачи предусмотренных пакетом SETUP данных надо разрешить прерывания ITTXFE в OTG_FS_DIEPINTx и OTEPDIS в OTG_FS_DOEPINTx. Затем при обработке прерываний ставим STALL точки и чистим прерывание.

28.17.7. Худшее время ответа

Это ответ на токен, следующий за изохронным OUT при работе OTG_FS устройством. Время ответа зависит от частоты тактов AHB (7 тактов PHY при равных частотах тактов PHY и AHB, и меньше при более высокой частоте AHB).

Регистры ядра лежат в домене AHB и оно не принимает новых токенов до обновления регистров, а изохронные передачи OUT не требуют ответа, и следующий групповой/прерывающий токен может получить ответ NAK. Изохронные токены отбрасываются и выдаётся прерывание IISOIXFR.

Хост воспринимает отсутствие ответа на токен SETUP как таймаут и повторяет его передачу.

Выбор значения TRDT в OTG_FS_GUSBCFG

Поле TRDT (OTG_FS_GUSBCFG) это число тактов PHY, отведённое MAC для получения статуса FIFO и чтения первого данного из блока PFC после получения токена IN. Это включает задержку синхронизации тактов PHY и AHB (в худшем случае 5 тактов).

Информация о получении MAC токена IN синхронизируется с частотой PFC (такты AHB). Затем PFC читает данные из SPRAM и пишет их в буфер с двойным тактированием (глубиной 4), откуда их читает MAC.

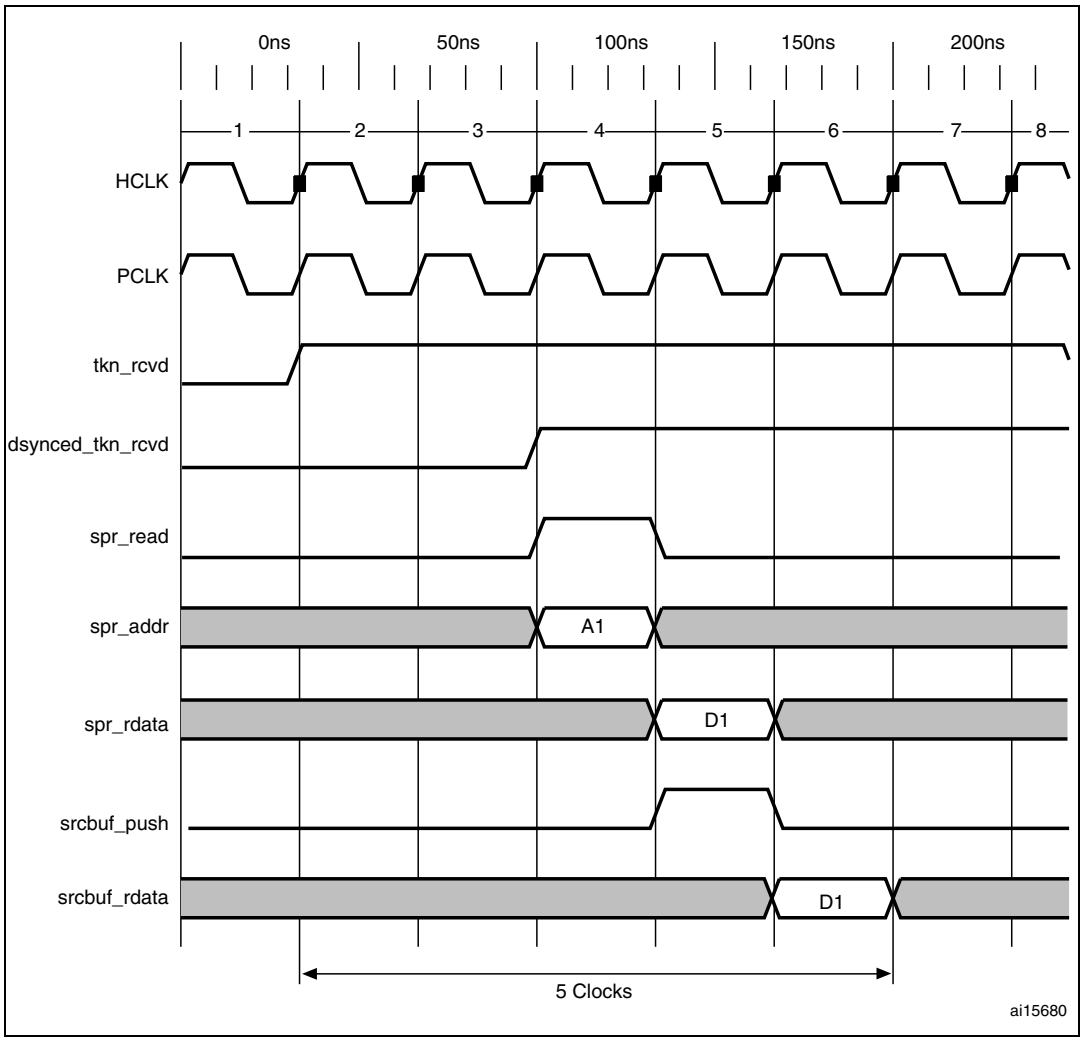
Если частота AHB выше частоты PHY, то можно использовать меньшие значения TRDT (в OTG_FS_GUSBCFG).

На рисунке ниже есть сигналы:

- tkn_rcvd: Сигнал PFC от MAC о получении токена
- dynced_tkn_rcvd: Двухчастотный tkn_rcvd, от PCLK в домен HCLK
- spr_read: Чтение в SPRAM
- spr_addr: Адрес в SPRAM
- spr_rdata: Чтение данных из SPRAM
- srcbuf_push: Запись в буфер
- srcbuf_rdata: Чтение данных из буфера. Данные видны для MAC

См. *Таблицу 204: Значения TRDT*.

Максимальные значения TRDT

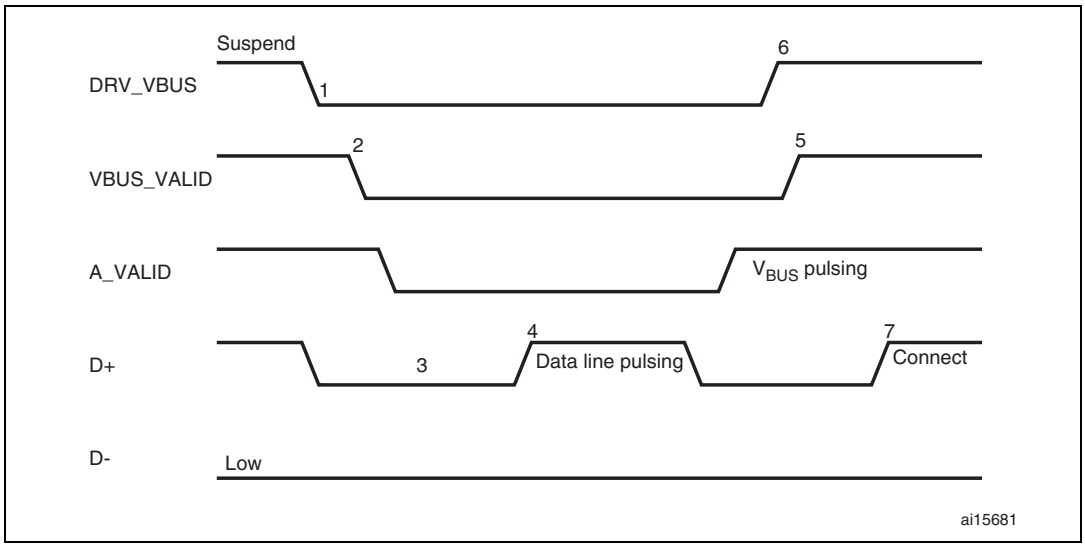


28.17.8. Программная модель OTG

Контроллер OTG_FS поддерживает протоколы HNP и SRP. Если ядро подключено к концу “A”, то именуется A-устройством, если к концу “B” то B-устройством. В режиме хоста контролер OTG_FS выключает V_{BUS}. Протокол SRP позволяет B-устройству попросить A-устройство включить V_{BUS}. Устройство выдаёт импульсы по линии данных и V_{BUS}, но при SRP хост обнаруживает импульс только на линии данных или V_{BUS}. Протокол HNP позволяет B-устройству переговорить и стать хостом. По переговорам после HNP B-устройство приостанавливает шину и оборачивается в периферию.

Протокол запроса сессии A-устройства

Программа ставит бит разрешения SRP в регистре Конфигурации Ядра и контроллер становится A-устройством.

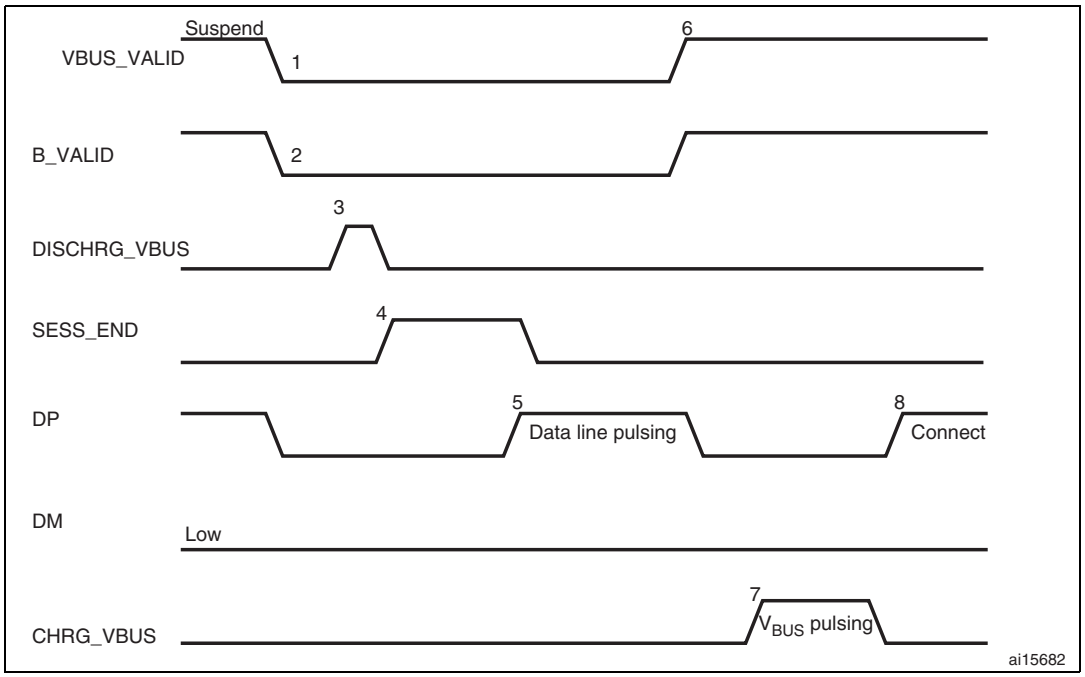


DRV_VBUS = Сигнал PHY подачи V_{BUS}
VBUS_VALID = Сигнал рабочего V_{BUS} от PHY
A_VALID = Сигнал рабочего уровня V_{BUS} от A-периферии к PHY
D+ = Линия данных плюс
D- = Линия данных минус

1. Программа останавливает порт и выключает питание простаивающей шины установкой битов остановки порта и выключения питания в регистре управления хоста.
2. PHY показывает выключение питания снятием сигнала VBUS_VALID.
3. Устройство должно обнаружить SE0 не менее чем за 2 ms до пуска SRP при выключенном V_{BUS}.
4. Для запуска SRP устройство подключает свой резистор подпорки линии данных на время от 5 до 10 ms. Контроллер OTG_FS обнаруживает этот импульс.
5. Устройство поднимает V_{BUS} выше порога (2.0 V минимум). Контроллер OTG_FS ставит бит запроса сессии (SRQINT в OTG_FS_GINTSTS) и выдаёт прерывание.
6. По прерыванию программа включает питание установкой бита в регистре управления хоста. PHY показывает включение питания выдачей сигнала VBUS_VALID.
7. USB запитан, устройство подключено, процесс SRP завершён.

Протокол запроса сессии B-устройства

Программа ставит бит разрешения SRP в регистре Конфигурации Ядра и контроллер становится B-устройством. Можно запросить сессию от хоста.



VBUS_VALID = Сигнал рабочего V_{BUS} от PHY

B_VALID = B-peripheral valid session to PHY
DISCHRG_VBUS = Сигнал разрядки для PHY
SESS_END = Сигнал конца сессии для PHY
CHRG_VBUS = Сигнал зарядки V_{BUS} для PHY

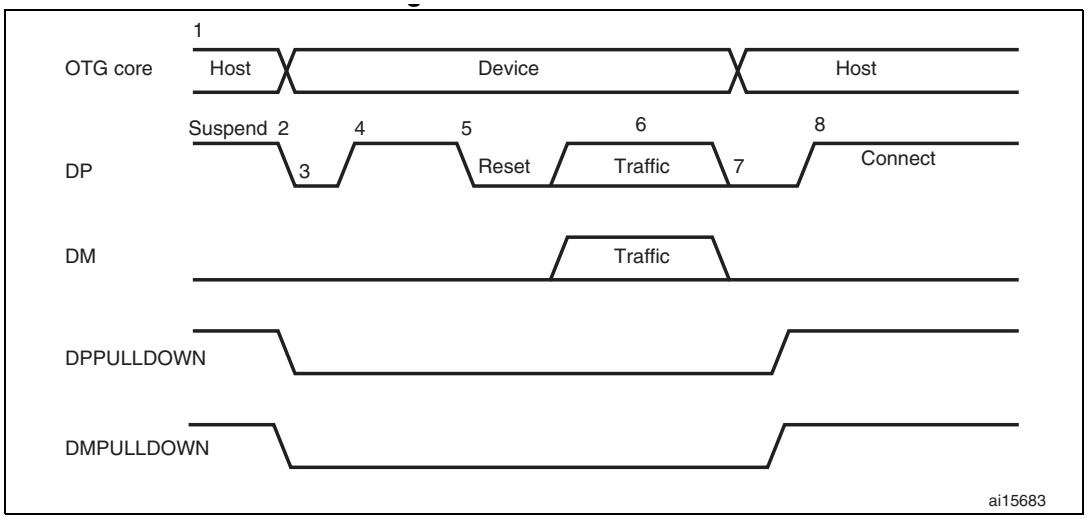
DP = Линия данных плюс

DM = Линия данных минус

1. При простое шины хост останавливает порт и выключает питание. Через 3 ms простоя контроллер OTG_FS ставит бит ранней остановки в регистре прерываний ядра. Затем он там же ставит бит остановки USB и подаёт PHY сигнал разрядки V_{BUS}.
2. PHY выдаёт устройству конец сессии. Это исходное состояние SRP. Контроллеру OTG_FS нужно 2 ms SE0 перед запуском SRP.
   Для полноскоростных трансиверов USB 1.1, после снятия BSVLD (в OTG_FS_GOTGCTL) программе надо дождаться разряда V_{BUS} до 0.2 V. Время указывает производитель.
3. Ядро USB OTG говорит PHY ускорить разряд V_{BUS}.
4. Программа пускает SRP записью бита запроса а регистре Управления OTG. Контроллер OTG_FS выдаёт импульс на линии данных с последующим импульсом на V_{BUS}.
5. Хост обнаруживает SRP по одному из импульсов и включает V_{BUS}. PHY показывает устройству включение V_{BUS}.
6. Контроллер OTG_FS выдаёт импульс V_{BUS}.
   Хост начинает новую сессию включением V_{BUS}. Контроллер OTG_FS выдаёт прерывание изменения состояния запроса сессии в регистре состояния прерываний OTG. Программа бит успешной сессии в регистре Управления OTG.
7. USB запитан, контроллер OTG_FS подключен, процесс SRP завершён.

Протокол беседы хоста A-устройства (HNP)

Протокол HNP переключает хост USB из A-устройства в B-устройство. Программа ставит бит разрешения HNP в регистре конфигурации ядра USB и контроллер становится A-устройством.



DPPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY.
DMPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

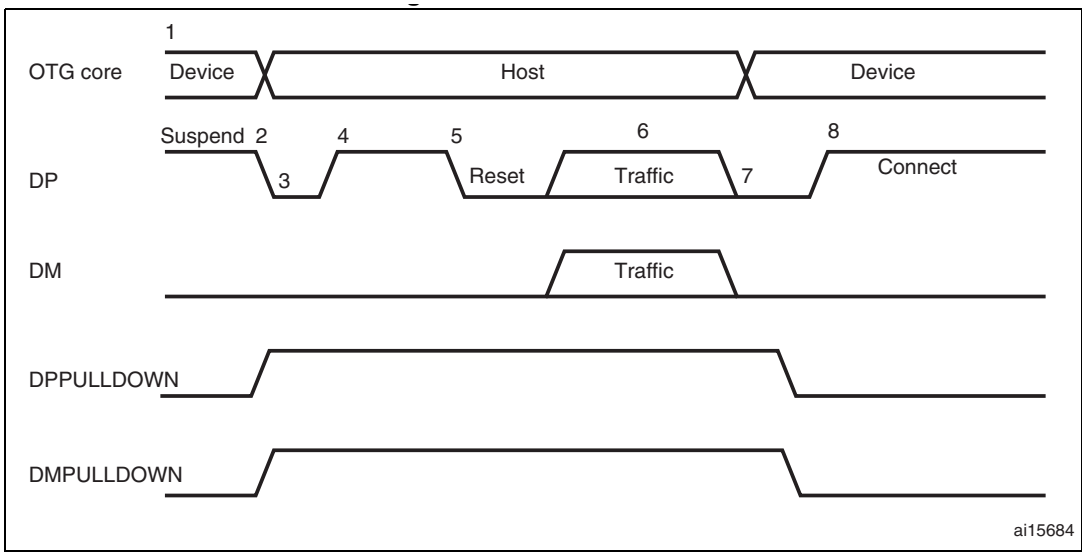
1. The OTG_FS controller sends the B-device a SetFeature b_hnp_enable descriptor to enable HNP support. The B-device's ACK response indicates that the B-device supports HNP. The application must set host Set HNP Enable bit in the OTG Control and status register to indicate to the OTG_FS controller that the B-device supports HNP.
2. When it has finished using the bus, the application suspends by writing the Port suspend bit in the host port control and status register.
   When the B-device observes a USB suspend, it disconnects, indicating the initial condition for HNP. The B-device initiates HNP only when it must switch to the host role; otherwise, the bus continues to be suspended.
   The OTG_FS controller sets the host negotiation detected interrupt in the OTG interrupt status

register, indicating the start of HNP.
The OTG_FS controller deasserts the DM pull down and DM pull down in the PHY to indicate a device role. The PHY enables the OTG_FS_DP pull-up resistor to indicate a connect for B-device. The application must read the current mode bit in the OTG Control and status register to determine device mode operation.

4. The B-device detects the connection, issues a USB reset, and enumerates the OTG_FS controller for data traffic.
5. The B-device continues the host role, initiating traffic, and suspends the bus when done.
The OTG_FS controller sets the early suspend bit in the Core interrupt register after 3 ms of bus idleness. Following this, the OTG_FS controller sets the USB Suspend bit in the Core interrupt register.
6. In Negotiated mode, the OTG_FS controller detects the suspend, disconnects, and switches back to the host role. The OTG_FS controller asserts the DM pull down and DM pull down in the PHY to indicate its assumption of the host role.
7. The OTG_FS controller sets the Connector ID status change interrupt in the OTG Interrupt Status register. The application must read the connector ID status in the OTG Control and Status register to determine the OTG_FS controller operation as an A- device. This indicates the completion of HNP to the application. The application must read the Current mode bit in the OTG control and status register to determine host mode operation.
8. The B-device connects, completing the HNP process.

HNP B-устройства

HNP switches the USB host role from B-device to A-device. The application must set the HNP-capable bit in the Core USB configuration register to enable the OTG_FS controller to perform HNP as a B-device.



DPPULLDOWN = сигнал от ядра к PHY включения/выключения резистора подтяжки линии DP.
DMPULLDOWN = сигнал от ядра к PHY t включения/выключения резистора подтяжки линии DM.

1. A-устройство посылает дескриптор **SetFeature b_hnp_enable** для включения поддержки HNP. ACK от контроллера OTG_FS подтверждает сей факт. Программа ставит бит разрешения HNP устройства в регистре управления OTG извещая о поддержке HNP.
Программа ставит бит запроса HNP в регистре управления OTG извещая контроллер о необходимости запуска HNP.
2. По завершению работы шины A-устройство останавливает порт в регистре Управления портом хоста. После 3 ms простоя шины контроллер OTG_FS ставит бит прерывания Раннего останова в регистре прерываний ядра. Затем контроллер OTG_FS ставит бит останова USB в регистре прерываний ядра.
Контроллер OTG_FS отключается и A-устройство обнаруживает на шине SE0, показывающий HNP. Контроллер OTG_FS включает подтяжку DP и DM в PHY, подтверждая роль хоста. A-устройство в ответ включает свой резистор подтяжки OTG_FS_DP в течении 3 ms обнаружения SE0. Контроллер OTG_FS понимает это как подключение.
Контроллер OTG_FS ставит прерывание изменения состояния беседы хоста в регистре

состояния прерываний OTG, показывая статус HNP. Программа определяет его по биту в регистре Управления OTG. Текущий режим выясняют в регистре OTG_FS_GINTSTS.

3. Программа ставит бит сброса PRST в OTG_FS_HPRT и контроллер OTG_FS выдаёт сброс USB и выполняет переучёт A-устройства.
4. По завершению работы хостом контроллер OTG_FS останавливает шину записью бита останова порта в регистр Управления портом хоста.
5. В режиме Беседы, когда A-устройство обнаруживает останов, оно отключается и переключается в хост. Контроллер OTG_FS отключает резисторы подтверждая роль устройства.
6. Текущий режим читают в регистре OTG_FS_GINTSTS.
7. Контроллер OTG_FS подключается, завершая процесс HNP.

29. Ethernet (ETH): контроллер доступа к среде (MAC) с контроллером DMA

29.1. Введение в Ethernet

Поддерживаются два промышленных стандарта интерфейса с внешним физическим уровнем (PHY): медиа-независимый интерфейс по умолчанию (MII) спецификации IEEE 802.3 и упрощённый медиа-независимый интерфейс (RMII).

Ethernet совместим со следующими стандартами:

- IEEE 802.3-2002 для Ethernet MAC
- IEEE 1588-2002 для синхронизации тактов точных сетей
- AMBA 2.0 для портов AHB Master/Slave
- Спецификация RMII от консорциума RMII

29.2. Основные свойства Ethernet

Свойства ядра MAC

- Поддерживает 10/100 Мбит/с передачу данных по внешнему интерфейсу PHY
- Совместимо с IEEE 802.3 MII интерфейсом связи с внешним Fast Ethernet PHY
- Поддерживает операции полу- и полного дуплекса
  - Поддерживает протокол CSMA/CD полу-дуплекса
  - Поддерживает управление потоком IEEE 802.3x полного дуплекса
  - Необязательная передача программе принятых фреймов паузы в полном дуплексе
  - Поддерживает сеть с обратным давлением в полу-дуплексе
  - Автоматическая передача кадра паузы нулевых квантов по снятию ввода управления потоком в полном дуплексе
- Вставка Преамбулы и Старта-фрейма (SFD) при Передаче и удаление при Приёме
- Автоматическое вычисление CRC и заполнителя для фрейма
- Возможное удаление Заполнителя/CRC на приёме
- Программируемая длина фрейма до 16 KB
- Программируемый промежуток между фреймами (40-96 бит шагами по 8)
- Поддержка режимов фильтрации адресов:
  - До четырёх 48-бит фильтров полного адреса (DA) с масками для каждого байта
  - До трёх 48-бит сравнений адреса SA с масками для каждого байта
  - 64-бит Хэш фильтр (опция) для широковещательных и полных (DA) адресов
  - Опция передачи всех широковещательных фреймов
  - Беспорядочный режим передачи всех фреймов без фильтрации для контроля сети
  - Передача всех поступающих пакетов с отчётом о состоянии
- Раздельное 32-бит состояние принимаемых и передаваемых пакетов

- Поддержка определения тэга IEEE 802.1Q VLAN в принимаемых фреймах
- Раздельные интерфейсы программы для передачи, приёма и управления
- Поддержка обязательной статистики сети с счётчиками RMON/MIB (RFC2819/RFC2665)
- Интерфейс MDIO для конфигурации и управления устройствами PHY
- Обнаружение фреймов побудки LAN и фреймов AMD Magic Packet™
- Приём дополнительной контрольной суммы принятых пакетов IPv4 и TCP внутри фреймов Ethernet
- Улучшенная проверка контрольной суммы заголовка IPv4 и контрольных сумм TCP, UDP или ICMP внутри дейтаграмм IPv4 и IPv6
- Поддержка меток времени фреймов Ethernet согласно IEEE 1588-2002. 64-бит метки времени в статусе каждого фрейма приёма и передачи
- Два набора FIFO: 2-KB Передающий FIFO с программируемым порогом и 2-KB и Приёмный FIFO с изменяемым порогом (по умолчанию 64 байта)
- Сохраняемые в приёмном FIFO после EOF векторы статуса приёма
- Опция фильтрации сбойных фреймов без участия программы
- Опция выдачи малоразмерных хороших фреймов
- Поддержка статистики выдачей импульсов для отброшенных и сбойных фреймов в FIFO приёма
- Поддержка механизма временного хранения (Store and Forward) передач ядру MAC
- Автоматическая выдача ядру MAC сигнала фрейма PAUSE или обратного давления на основе порога FIFO приёма
- Автоматическая повторная передача фреймов при коллизии
- Отбрасывание фреймов при запаздывании, чрезмерных коллизиях, излишней задержке и исчерпании
- Программное управление сливом TxFIFO
- Вычисление и вставка контрольных сумм заголовков IPv4 и TCP, UDP или ICMP внутри передаваемых фреймов в режиме временного хранения
- Поддержка внутренней перемычки (loopback) в MII для отладки

29.2.1. Свойства DMA

- Поддержка всех типов пакетов ведомого AHB
- Программный выбор типа пакета (фиксированный или неопределённый) ведущего AHB.
- Опция выбора пакетов с выравненными адресами из порта ведущего AHB
- Оптимизация пакетно-ориентированных передач DMA с разделителями фреймов
- Байтовое выравнивание адресов буферов данных
- Двухбуферная (кольцевая) или Цепная (связанным списком) связка дескрипторов
- Архитектура дескрипторов, позволяющая передачу больших блоков данных без участия CPU;
- Один дескриптор может передавать до 8 KB данных
- Исчерпывающий отчёт о нормальных и сбойных передачах
- Раздельный программируемый размер пакетов у машин DMA приёма и передачи
- Программируемые опции прерываний
- Прерывания по концу фреймов Приёма/Передачи
- Кольцевой или приоритетный арбитраж для машин Приёма и Передачи
- Режимы Старт/Стоп
- Текущий указатель буфера Tx/Rx как регистр состояния
- Текущий указатель Дескрипторов Tx/Rx как регистр состояния

29.2.2. Свойства PTP

- Метки времени принимаемых и передаваемых фреймов
- Методы грубой и точной коррекции

- Выдаёт прерывание если системное время превышает целевое
- Импульс секунд

29.3. Ножки Ethernet

Таблица 207. Конфигурация ножек Ethernet

Сигналы MAC	Основной MII	Сменный MII	Основной RMII	Сменный RMII	Нога	Конфигурация ножки
ETH_MDC	MDC	-	MDC	-	PC1	Двухтактный выход AF (50 MHz)
ETH_MII_TXD2	TXD2	-	-	-	PC2	Двухтактный выход AF (50 MHz)
ETH_MII_TX_CLK	TX_CLK	-	-	-	PC3	Плавающий вход (по сбросу)
ETH_MII_CRS	CRS	-	-	-	PA0	Плавающий вход (по сбросу)
ETH_MII_RX_CLK ETH_RMII_REF_CLK	RX_CLK	-	REF_CLK	-	PA1	Плавающий вход (по сбросу)
ETH_MDIO	MDIO	-	MDIO	-	PA2	Двухтактный выход AF (50 MHz)
ETH_MII_COL	COL	-	-	-	PA3	Плавающий вход (по сбросу)
ETH_MII_RX_DV ETH_RMII_CRS_DV	RX_DV	-	CRS_DV	-	PA7	Плавающий вход (по сбросу)
ETH_MII_RXD0 ETH_RMII_RXD0	RXD0	-	RXD0	-	PC4	Плавающий вход (по сбросу)
ETH_MII_RXD1 ETH_RMII_RXD1	RXD1	-	RXD1	-	PC5	Плавающий вход (по сбросу)
ETH_MII_RXD2	RXD2	-	-	-	PB0	Плавающий вход (по сбросу)
ETH_MII_RXD3	RXD3	-	-	-	PB1	Плавающий вход (по сбросу)
ETH_MII_RX_ER	RX_ER	-	-	-	PB10	Плавающий вход (по сбросу)
ETH_MII_TX_EN ETH_RMII_TX_EN	TX_EN	-	TX_EN	-	PB11	Двухтактный выход AF (50 MHz)
ETH_MII_TXD0 ETH_RMII_TXD0	TXD0	-	TXD0	-	PB12	Двухтактный выход AF (50 MHz)
ETH_MII_TXD1 ETH_RMII_TXD1	TXD1	-	TXD1	-	PB13	Двухтактный выход AF (50 MHz)
ETH_PPS_OUT	PPS_OUT	-	PPS_OUT	-	PB5	Двухтактный выход AF (50 MHz)
ETH_MII_TXD3	TXD3	-	-	-	PB8	Двухтактный выход AF (50 MHz)
ETH_RMII_CRS_DV	-	RX_DV	-	CRS_DV	PD8	Плавающий вход (по сбросу)
ETH_MII_RXD0 ETH_RMII_RXD0	-	RXD0	-	RXD0	PD9	Плавающий вход (по сбросу)
ETH_MII_RXD1 ETH_RMII_RXD1	-	RXD1	-	RXD1	PD10	Плавающий вход (по сбросу)
ETH_MII_RXD2	-	RXD2	-	-	PD11	Плавающий вход (по сбросу)
ETH_MII_RXD3	-	RXD3	-	-	PD12	Плавающий вход (по сбросу)

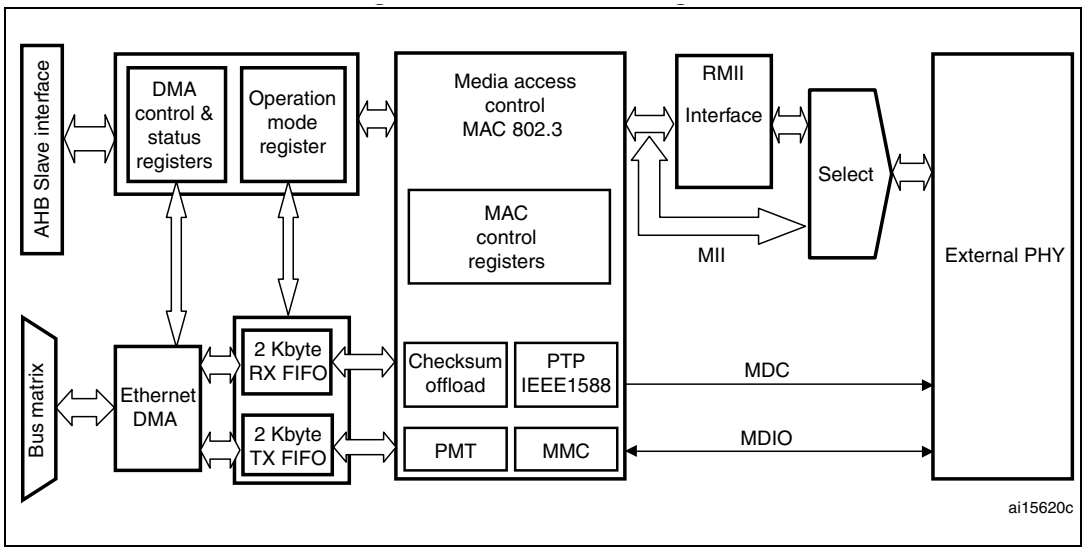
29.4. Функциональное описание Ethernet: SMI, MII и RMII

Блок состоит из MAC 802.3 (контроллер доступа среды) с контроллером DMA. Он поддерживает интерфейсы MII и RMII. Также есть SMI для связи с внешним PHY.

Контроллер DMA работает с ядром и памятью ведущим и ведомым по шине AHB. Ведущим при передаче данных и ведомым при доступе к регистрам (CSR).

NB: Частота AHB должна быть не меньше 25 MHz при работе с Ethernet.

Блок-схема ETH



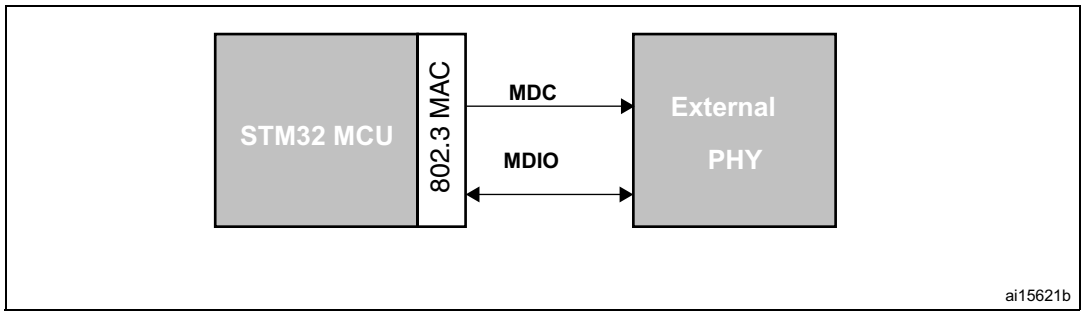
29.4.1. Интерфейс управления станцией: SMI

Обеспечивает доступ к регистрам PHY по 2 линиям тактов (MDC) и данных (MDIO). Поддерживаются до 32 PHY. Единовременно доступен только один регистр PHY.

Обе линии реализованы как альтернативные функции I/O:

- MDC: Такты передачи данных частотой до 2.5 MHz. Минимальная длительность состояний MDC равна 160 ns, минимальный период 400 ns. В состоянии простоя удерживается низким.
- MDIO: Линия данных, синхронна с MDC.

Сигналы интерфейса SMI



Формат фрейма SMI

Биты фрейма передаются начиная с левого.

Таблица 208. Формат фрейма управления

Поля фрейма								
	Преамбула (32 бита)	Старт	Операция	PADDR	RADDR	TA	Данные (16 бит)	Простой
Чтение	1... 1	01	10	ppppp	rrrrr	Z0	dddddddddddddddd	Z
Запись	1... 1	01	01	ppppp	rrrrr	10	dddddddddddddddd	Z

Фрейм состоит из восьми полей:

- **Преамбула**: все передачи начинаются с 32 единиц на MDIO за 32 такта MDC. Это синхронизация с PHY.
- **Старт**: Это биты <01>.
- **Операция**: Это тип операции (чтение или запись).
- **PADDR**: Это 5 бит адреса PHY. Старший бит передаётся первым.
- **RADDR**: Это 5 бит адреса регистра PHY. Старший бит передаётся первым.
- **TA**: Это 2-бит разделитель между полями RADDR и DATA. При чтении контроллер MAC ставит линию MDIO в Z-состояние на 2 бита TA. PHY ставит линию MDIO в Z-состояние на первый бит

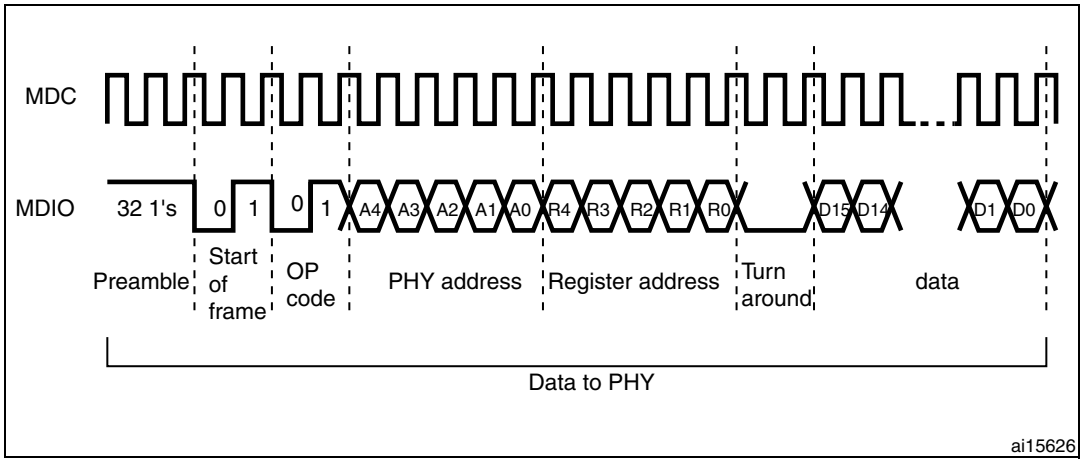
TA и в 0 во втором бите TA. При записи контроллер MAC выдаёт бита <10>. PHY держит линию в Z-состоянии 2 бита TA.

- **Данные**: Это 16 бит регистра ETH_MIID, начиная со старшего.
- **Простой**: Ключи линии MDIO стоят в Z-состоянии, резисторы подпорки PHY держат единицу.

Операция записи SMI

После программной установки битов MII Запись и Занят (ETH_MACMIIAR) SMI передаёт в регистры PHY адрес PHY, адрес регистра PHY и данные из регистра ETH_MACMIIDR. При работающей передаче регистры адреса и данных изменить нельзя (стоит бит Занят), он снимется после завершения операции.

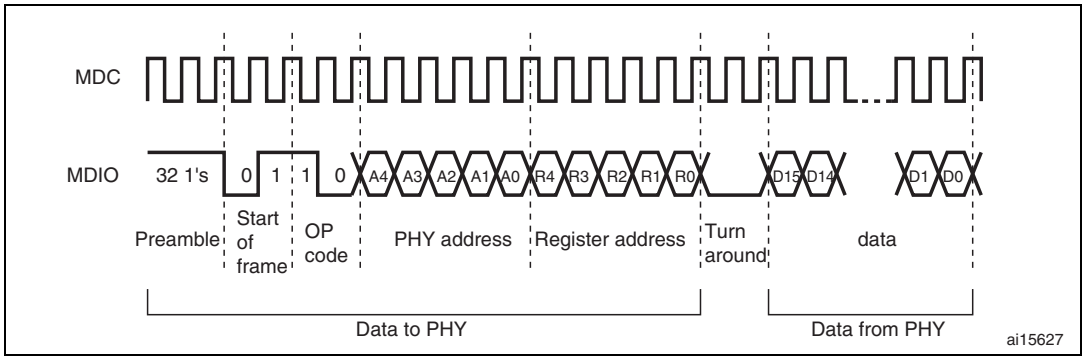
Времянка MDIO при записи



Операция чтения SMI

Запускается установкой бита Занят MII (в ETH_MACMIIAR) при снятом бите Запись. SMI передаёт в регистры PHY адрес PHY и адрес регистра PHY. Т При работающей передаче регистры адреса и данных изменить нельзя (стоит бит Занят). После завершения операции чтения SMI снимет бит Занят и заполнит регистр данных MII принятыми от PHY данными.

Времянка MDIO при чтении



Выбор тактов SMI

MAC запускает операцию Чтения/Записи. Такты SMI получаются делением тактов AHB в соответствии с полем в регистре адреса MII.

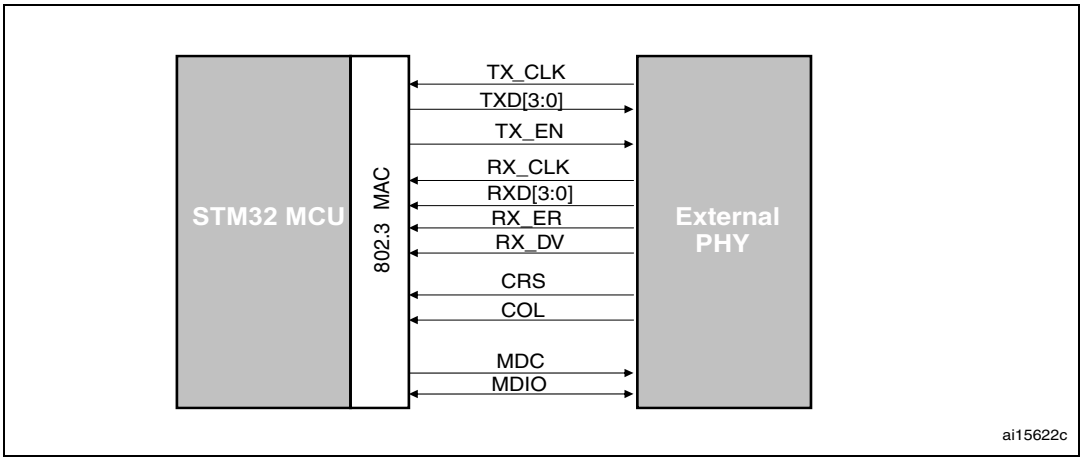
Таблица 209. Диапазон частот

Выбор	Частота HCLK	Частота MDC
000	60-72 MHz	Такты AHB / 42
001	Резерв	—
010	20-35 MHz	Такты AHB / 16
011	35-60 MHz	Такты AHB / 26
0100, 0101, 0110, 0111	Резерв	—

29.4.2. Независимый от среды интерфейс: MII

Определяет соединение подуровня MAC и PHY при передачах на 10 Мбит/с и 100 Мбит/с.

Сигналы интерфейса MII.



- **MII_TX_CLK**: Тактовый сигнал для TX данных. Частота 2.5 MHz для 10 Мбит/с до 25 MHz для 100 Мбит/с.
- **MII_RX_CLK**: Тактовый сигнал для RX данных. Частота 2.5 MHz для 10 Мбит/с до 25 MHz для 100 Мбит/с.
- **MII_TX_EN**: Разрешение передачи. MAC выставляет полубайты передачи. Ставится синхронно (MII_TX_CLK) с выдачей первого полубайта преамбулы и держится до конца передачи.
- **MII_TXD[3:0]**: 4 бита передаваемых данных от подуровня MAC. MII_TXD[0] это младший бит, MII_TXD[3] - старший. Без стоящего MII_TX_EN на PHY не влияет.
- **MII_CRS**: Несущая есть. Выставляется PHY при наличии приёма или передачи. Им же и снимается. В случае коллизии должен стоять. С тактами TX и RX не синхронизируется. В полном дуплексе подуровню MAC не нужен.
- **MII_COL**: Коллизия. Ставится PHY в середине выборки и должен стоять все время существования коллизии. С тактами TX и RX не синхронизируется. В полном дуплексе подуровню MAC не нужен.
- **MII_RXD[3:0]**: 4 бита принимаемых от PHY данных, первый полубайт ставится синхронно с сигналом MII_RX_DV. MII_RXD[0] это младший бит, MII_RXD[3] - старший. При снятом MII_RX_EN и стоящем MII_RX_ER используются для передачи спецданных от PHY (см. *Таблицу 211*).
- **MII_RX_DV**: Разрешение приёма. Ставится PHY синхронно (MII_RX_CLK) с первым полубайтом фрейма и снимается после передачи последнего до начала следующего такта. Должен включать фрейм, начинаясь не позднее поля SFD.
- **MII_RX_ER**: Ошибка приёма. Ставится не менее чем на один период MII_RX_CLK для извещения MAC об ошибке в фрейме. Смысл определяется вместе с состоянием MII_RX_DV (см. *Таблицу 211*).

Таблица 210. Сигналы интерфейса TX

MII_TX_EN	MII_TXD[3:0]	Описание
0	0000 до 1111	Нормальный интер-фрейм
1	0000 до 1111	Нормальная передача данных

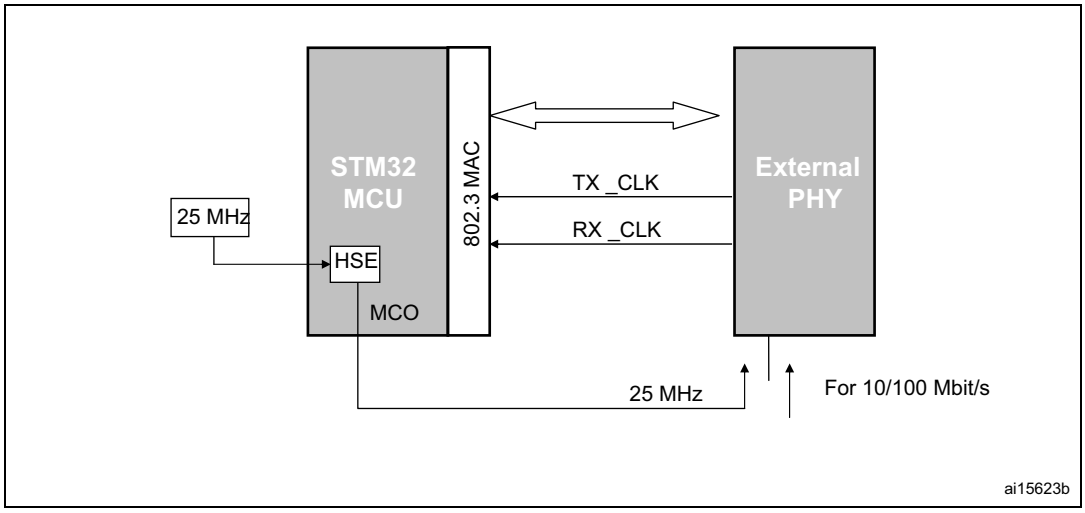
Таблица 211. Сигналы интерфейса RX

MII_RX_DV	MII_RX_ER	MII_RXD[3:0]	Описание
0	0	0000 до 1111	Нормальный интер-фрейм
0	1	0	Нормальный интер-фрейм
0	1	0001 до 1101	Резерв
0	1	1110	Дурная несущая
0	1	1111	Резерв
1	0	0000 до 1111	Нормальный приём данных
1	1	0000 до 1111	Приём данных с ошибками

Источники тактов MII

Для выдачи сигналов TX_CLK и RX_CLK на внешний PHY надо подать внешние 25 MHz. Вместо внешнего 25 MHz кварца можно подать его с ножки MCO микроконтроллера STM32F10xxx. В этом случае надо правильно сконфигурировать PLL с внешним 25 MHz кварцем.

Источники тактов MII



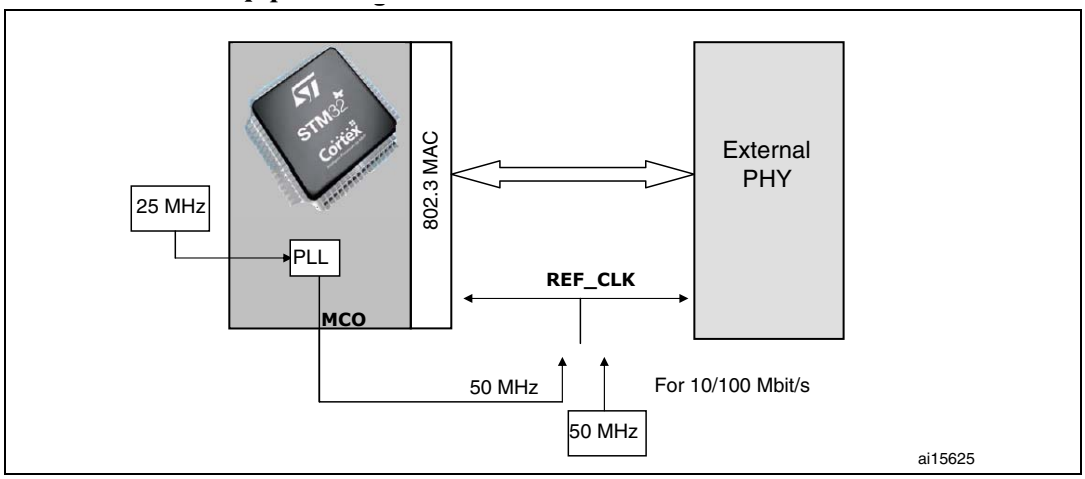
29.4.3. Сокращённый независимый от среды интерфейс: RMII

Он уменьшает число ножек соединения микроконтроллера и PHY с 16 до 7.

RMII стоит между MAC и PHY для трансляции интерфейса MII в RMII. Характеристики такие:

- Поддержка скоростей 10-Mbit/s и 100-Mbit/s
- Опорная частота удваивается до 50 MHz
- Опорная частота подаётся и на MAC и на внешний Ethernet PHY
- Обеспечивает независимые 2-бит (дибит) приём и передачу

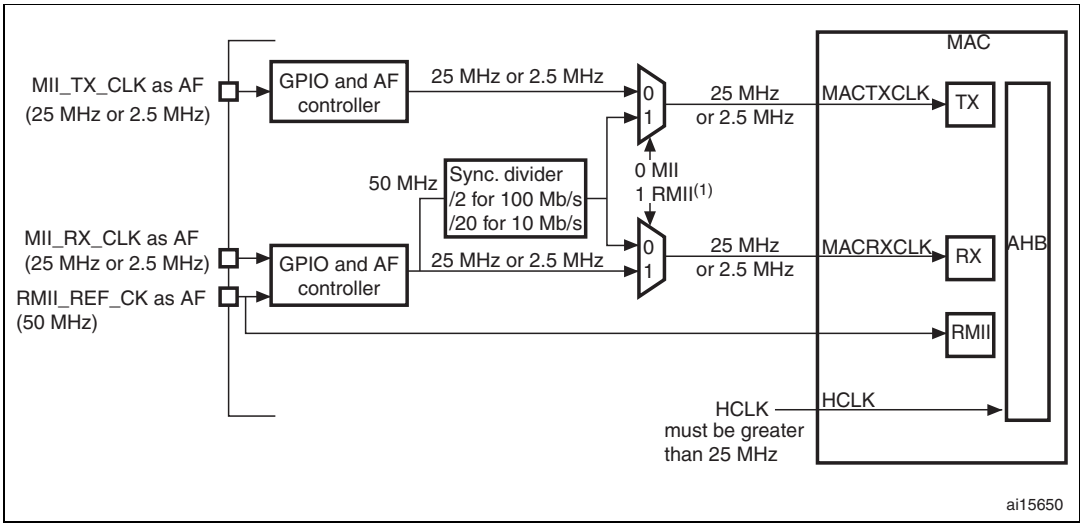
Сигналы интерфейса RMII.



29.4.4. Выбор режима MII/RMII

Режим определяется битом `MII_RMII_SEL` в регистре `AFIO_MAPR`. Писать бит надо при сброшенном контроллере или до подачи тактов.

Схема внутреннего тактирования MII/RMII



Сигналы, RMII_REF_CK и MII_RX_CLK мультиплексируются на одну ножку GPIO.

29.5. Функциональное описание Ethernet: MAC 802.3

Блок MAC реализует подуровень LAN CSMA/CD 10 Mbit/s и 100 Mbit/s узкополосных и широкополосных систем. Поддерживаются полу- и полный дуплекс. Коллизии обнаруживаются только в полудуплексе. Поддерживается подуровень управляющих фреймов MAC.

Он выполняет следующие функции:

- Инкапсуляция данных (передача и приём)
  - Разделение границ и синхронизация фреймов
  - Адресация (источник и получатель)
  - Обнаружение ошибок
- Управление доступом к среде
  - Срединное размещение (предотвращение коллизий)
  - Разрешение конфликтов (обработка коллизий)

Два режима работы подуровня MAC:

- Полудуплекс: доступ к среде по алгоритмам CSMA/CD.
- Полный дуплекс: одновременные приём и передача (CSMA/CD не нужны) когда:

- Физическая среда поддерживает одновременные приём и передачу
- К LAN подключены только 2
- Обе станции работают в полном дуплексе

29.5.1. Формат фрейма MAC 802.3

CSMA/CD MAC определяет два формата фреймов:

- Базовый формат фрейма MAC
- Фрейм MAC с тегами (расширение базового формата)

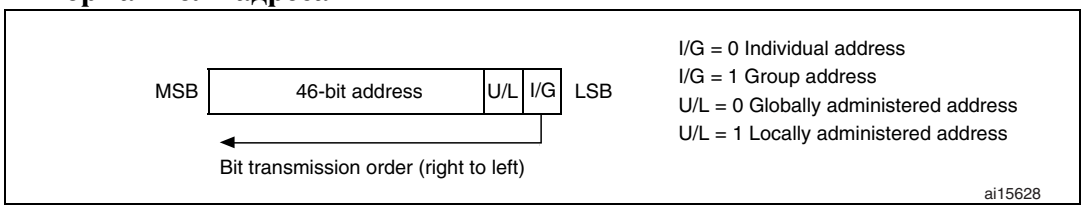
Фреймы содержат следующие поля:

- Преамбула: 7-байтовое поле синхронизации (схема PLS)
  Шестнадцатиричное: 55-55-55-55-55-55-55
  Биты: 01010101 01010101 01010101 01010101 01010101 01010101 01010101 (справа налево)
- Старт фрейма (SFD): 1-байтовое поле.
  Шестнадцатиричное: D5
  Биты: 11010101 (справа налево)
- 6-байтовые поля адресов источника и получателя:
  - Каждый адрес по 48 бит длиной
  - Младший бит адреса получателя (I/G) определяет индивидуальный (I/G = 0) или групповой (I/G = 1) адрес. он может определять все, несколько или отсутствующие станции в LAN. Младший бит адреса источника всегда равен 0.
  - Второй бит (U/L) отличает локальное (U/L = 1) или глобальное (U/L = 0) администрирование адресов. У широковещательных адресов этот бит всегда равен 1.
  - Все байты передаются младшим битом вперёд.

Есть два типа адресов назначения:

- Индивидуальный адрес отдельной станции в сети.
- Групповой адрес двух типов:
  - Групповой адрес для нескольких станций в сети.
  - Широковещательный адрес (все 1 в поле адреса получателя) для всех станций в сети.

Формат поля адреса.



- Префикс QTag: 4-байт поле между полями Адреса Источника и Длины/Типа Клиента MAC. Это расширение базового формата (там его нет). Состоит из:
  - 2-байтовая константа Длины/Типа, совместимая с интерпретацией Типа (больше 0x0600) равная значению 802.1Q Tag Protocol Type (0x8100). Отличает теговые фреймы MAC.
  - 2-байтовое поле Тега, состоящее из: 3- бит приоритета пользователя, бита канонического индикатора формата (CFI) и 12-бит идентификатора VLAN.
- Длина/Тип Клиента MAC: 2-байтовое поле с взаимоисключающими значениями:
  - Если оно меньше или равно maxValidFrame (0d1500), то это число байтов данных клиента MAC в последующем поле данных фрейма 802.3 (интерпретация длины).
  - Если оно больше или равно MinTypeValue (0d1536 или 0x0600), то это тип протокола клиента MAC (интерпретация типа) фрейма Ethernet.

  Независимо от интерпретации поля длины/типа, если длина поля данных меньше требуемого для нормальной работы протокола, то после поля данных перед полем FCS (последовательность проверки фрейма) добавляется расширитель (PAD). Поле длины/типа передаётся начиная со старшего байта.

  Значения поля длины/типа между maxValidLength и minTypeValue (исключая границы) подуровнем MAC могут не передаваться.

- Поля данных и PAD: Любые данные. Общий размер:
  - Максимальная длина = 1500 байт
  - Минимальная длина бестегового фрейма MAC = 46 байт
  - Минимальная длина тегового фрейма MAC = 42 байта

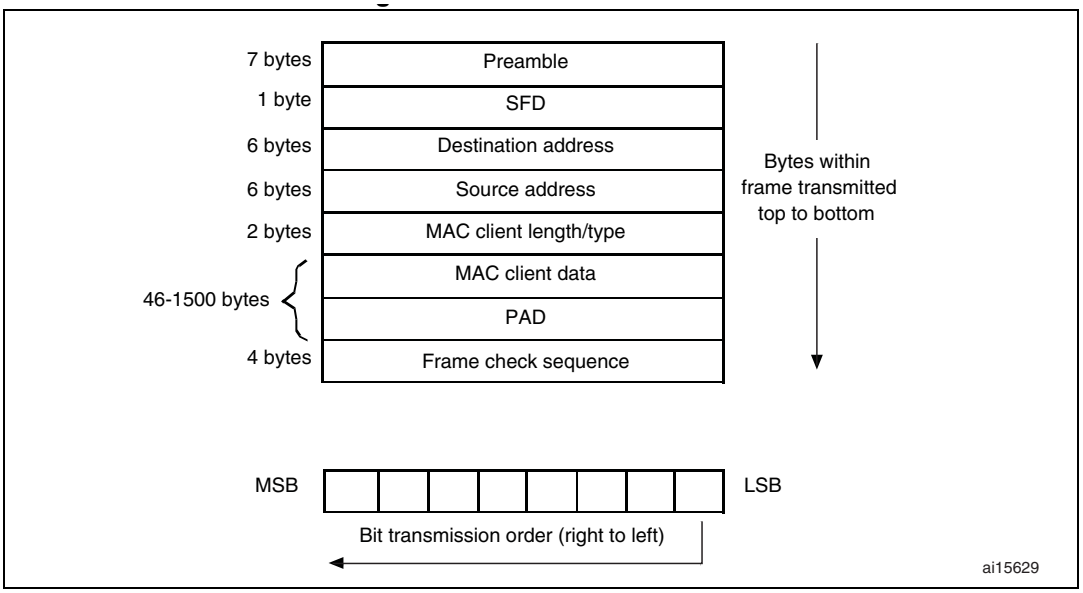
  PAD добавляется для получения минимальной общей длины.
- Последовательность проверки фрейма: 4-байтовое поле с CRC. В вычислении CRC участвуют все поля, кроме преамбулы и SFD. Полином такой:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

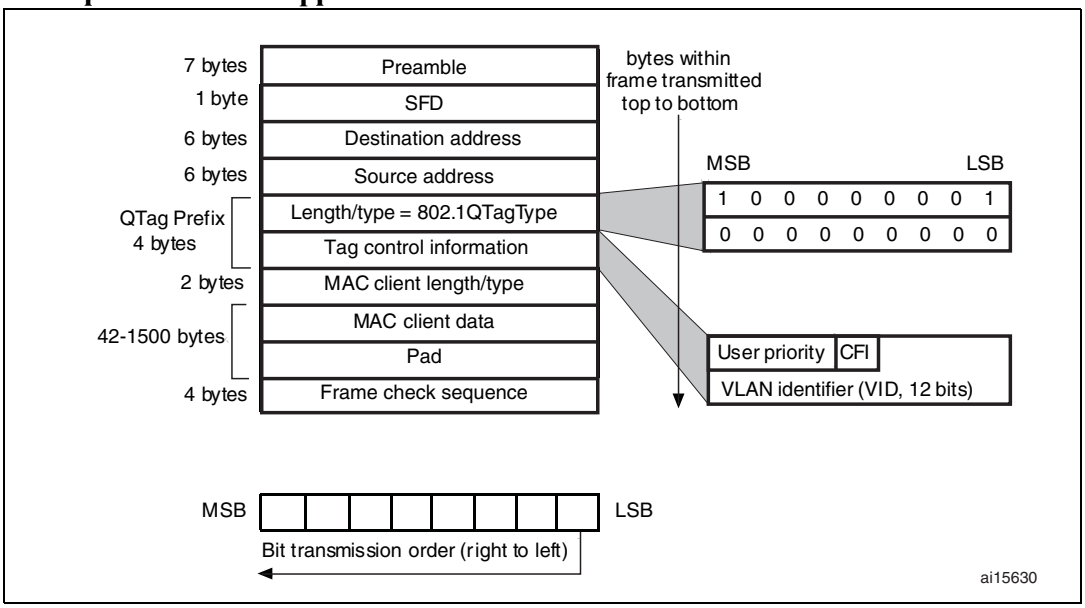
CRC фрейма вычисляется так:

- Первые 2 бита фрейма комплементируются
- n-битов фрейма это коэффициенты полинома $M(x)$ степени $(n-1)$. Первый бит адреса получателя соответствует x^{n-1} и последний бит поля данных соответствует x^0
- $M(x)$ умножается на x^{32} и делится на $G(x)$, выдавая остаток $R(x)$ степени ≤31
- Коэффициенты $R(x)$ рассматриваются как 32-бит последовательность
- CRC получается комплементом этой последовательности
- 32-бит CRC помещается в последовательности проверки фрейма. x^{32} передаётся первым

Формат фрейма MAC.



Формат тегового фрейма MAC.



Все байты фрейма, кроме поля FCS, передаются младшим битом вперёд. Недостойный фрейм MAC определяется так:

- Длина фрейма не соответствует указанной в поле длины/типа. Если указан тип, то должна быть соответствующая ему длина
- Не целое число байтов (лишние биты)
- Вычисленный CRC не равен переданному в FCS

29.5.2. Передача фрейма MAC

Фреймы из системной памяти в FIFO передаются через DMA, оттуда их выталкивает ядро MAC. После передачи конца фрейма состояние передачи возвращается DMA. Глубина передающего FIFO равна 2 КБ. При передаче данных DMA использует уровень заполнения FIFO.

При появлении SOF, MAC принимает данные и начинает передачу в MII. Время передачи зависит от задержки IFG, времени передачи преамбулы/SFD и задержки подтверждения полу-дуплекса. После получения EOF ядро MAC прекращает передачу и возвращает DMA статус передачи. При появлении коллизии (в полу-дуплексе) ядро MAC отмечает это в статусе и отбрасывает дальнейшие данные до получения SOF. Этот фрейм надо передать повторно начиная с SOF. Если передаваемые данные не поступают вовремя, то MAC выдаёт исчерпание. Если новый SOF поступает до получения EOF предыдущего фрейма, то SOF игнорируется и новый фрейм рассматривается как продолжение предыдущего.

Есть два режима выдачи данных ядру MAC:

- В режиме Порога данные передаются ядру MAC после заполнения FIFO выше порога, заданного битами `TTC` в регистре `ETH_DMABMR`, или при получении EOF.
- В режиме Накопления (Store-and-forward) данные передаются ядру MAC только после записи в FIFO всего фрейма. Если TxFIFO меньше передаваемого фрейма, то данные передаются ядру MAC при почти полном TxFIFO.

Передающий FIFO сливается установкой бита `FTF` в регистре `ETH_DMAOMR`. По окончанию слива бит аппаратно снимается и FIFO возвращается в исходное состояние. Если бит `FTF` ставится во время передачи данных ядру MAC, то передача останавливается и FIFO становится пустым. Возникает Исчерпание и DMA передаётся соответствующий статус.

Автоматическое формирование CRC и расширения

Если число байтов для фрейма (DA+SA+LT+Data) меньше 60, то в данные добавляются нули до получения 46 байт. Это действие MAC можно запретить. К передаваемым данным в поле FCS добавляется CRC, что тоже можно запретить. Если расширение фрейма разрешено, то обязательно добавится и CRC. CRC вычисляется по полиному:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Протокол передачи

Он соответствует спецификации IEEE 802.3/802.3z, это:

- генерация преамбулы и SFD
- генерация джем-последовательности (jam pattern) в полу-дуплексе
- управление длительностью трёпа (Jabber timeout)
- управление потоком в полу-дуплексе (обратное давление)
- генерация статуса передачи фрейма
- логика меток времени в соответствии с IEEE 1588

При запросе передачи нового фрейма MAC перед данными посылает преамбулу и SFD. Преамбула это 7 байтов `0b10101010`, а SFD это 1 байт `0b10101011`. Окно коллизии определено как 1 слот (время 512 бит для 10/100 Mbit/s Ethernet).

В режиме MII, при появлении коллизии во время передачи от начала фрейма до конца поля CRC, MAC высылает по MII 32-бит джем-последовательность `0x5555 5555` для извещения других станций о возникновении коллизии. Если коллизия возникла при передаче преамбулы, то MAC прекращает передачу преамбулы и SFD и высылает джем.

Таймер трёпа отслеживает, чтобы передача фрейма не превышала 2048 байт. В полу-дуплексе MAC использует механизм отсрочки (обратного давления). По запросу остановки приёма MAC в

любом месте приёма отсылает 32 байта Джема, разрешая передачу. Возникает коллизия и удалённая станция стопорится. Программа запрашивает управление передачей установкой бита BPA в регистре ETH_MACFCR. Запрошенный фрейм планируется и передаётся даже при активном обратном давлении. Оно сохраняется активным длительное время (более 16 последовательных коллизий) и удалённая станция стопорит передачу по причине излишних коллизий. В метках времени отмечается момент передачи SFD на шину MII.

Планирование передачи

В полу-дуплексе MAC управляет промежутками между передаваемыми фреймами по усечённому двоичному экспоненциальному алгоритму возврата. Он разрешает передачу нового фрейма после задержек IFG (биты IFG в регистре ETH_MACCR) и возврата. Если передаваемые фреймы появляются раньше отведённого времени IFG, то MII перед запуском передачи ждёт сигнала разрешения от MAC. Счётчик IFG запускается при отключении несущей от MII. В полном дуплексе MAC разрешает передачу по истечении времени IFG. В полу-дуплексе и при IFG равном времени 96 бит MAC следует правилу задержки из спецификации IEEE 802.3. MAC сбрасывает свой счётчик IFG если несущая появляется в течение первых двух третей интервала IFG (время 64-бит). Если несущая появляется во время последней трети интервала IFG, то счётчик не останавливается и передача разрешается после истечения всего интервала IFG.

Управление потоком передачи

В полном дуплексе при стоящем бите TFE в регистре ETH_MACFCR MAC создаёт фреймы Паузы с добавленным CRC и выдаёт их при установке бита FCB в регистре ETH_MACFCR или при заполненном FIFO приёма.

- По установке бита FCB в ETH_MACFCR MAC выдаёт один фрейм Паузы длительностью, заданной в битах PT в регистре ETH_MACFCR. Для окончания или расширения паузы надо выдать новый фрейм с изменённой длительностью паузы.
- При запросе передачи в заполненный FIFO MAC выдаёт фрейм Паузы длительностью, определённой в регистре ETH_MACFCR. Если FIFO остаётся заполненным в течении заданного числа слотов времени (биты PLT в ETH_MACFCR) до истечения этой паузы, то выдаётся новый фрейм паузы, иначе MAC передаёт фрейм с нулевым временем паузы, извещая о готовности принимать новые фреймы.

Передача одного пакета

Общая последовательность такова:

1. Если системе есть что передавать, то контроллер DMA переносит данные из памяти в FIFO вплоть до конца фрейма.
2. При достижении уровня порога или полном пакете в FIFO контроллер DMA начинает передавать данные фрейма ядру MAC. И так продолжается до выдачи из FIFO полного пакета. По завершению фрейма контроллер DMA получает извещение от MAC.

Передача—два пакета в буфере

1. Поскольку перед возвращением дескриптора хосту DMA должен обновить его статус, то в FIFO можно держать два фрейма. DMA переносит второй фрейм в FIFO только при стоящем бите OSF, иначе следующий фрейм извлекается из памяти только после завершения обработки фрейма MAC и освобождения дескрипторов контроллером DMA.
2. При стоящем бите OSF DMA начинает переносить второй фрейм в FIFO сразу после записи первого и не ждёт изменения статуса. Сразу после передачи первого пакета MAC выдаёт DMA статус передачи и, если второй пакет уже в FIFO, то вторая передача должна дождаться статуса первого пакета.

Повторная передача при коллизии в полу-дуплексе

В случае коллизии MAC должен показать это выдачей статуса даже до получения конца фрейма. Затем разрешается повторная передача и фрейм выдаётся из FIFO. После выдачи ядру MAC более 96 байт контроллер FIFO освобождает место и отдаёт его DMA. То есть при превышении порога или коллизии опоздания повторная передача невозможна.

Слив передающего FIFO

Выполняется битом 20 Регистра режима работы. TxFIFO и соответствующие указатели сбрасывается в начальное состояние даже при незавершённой выдаче данных ядру MAC, возникает исчерпание и передача прекращается. В статусе фрейма отмечаются и исчерпание и слив (биты 13 и 1

в TDES0). Во время слива данные в FIFO не поступают. На каждый слитый фрейм (включая незавершённые) выдаются слова статуса со стоящим битом (TDES0 13) для полностью слитых фреймов. Операция слива завершается после приёма слов состояния всех слитых фреймов. Бит слива снимается. Можно принимать начинающиеся с SOF новые фреймы.

Слово статуса передачи

Оно передаётся программе после того как ядро MAC завершит приём или передачу фрейма по сети. Это биты[23:0] в TDES0. Если разрешены метки времени, то вместе со словом состояния выдаётся и 46-битная метка времени.

Контрольная сумма передачи

Протоколы TCP и UDP используют передачу контрольных сумм по сети. Стало быть и контроллер занимается этим.

NB: Контрольная сумма для TCP, UDP и ICMP вычисляется для всего фрейма и вставляется в нужное поле заголовка. Это разрешается для передающего FIFO только в режиме Накопления (Store-and-forward) установкой бита TSF в регистре ETH_ETH_DMAOMR.

Tx FIFO должен уметь хранить передаваемый фрейм целиком, иначе в любом режиме обрабатывается только контрольная сумма заголовка фрейма IPv4.

Два режима вычисления и вставки контрольной суммы определяются битами CIC в TDES1.

- **Контрольная сумма заголовка IP**

В дейтаграммах IPv4 16-бит контрольная сумма лежит в 11-м и 12-м байтах. Она проверяется когда поле Типа равно 0x0800, а поле Версии равно 0x4. Во входящем фрейме она вычисляется заново и пишется в это поле даже при ошибке CRC. Заголовки IPv6 контрольную сумму не хранят. Результат проверки указывается в бите 16 регистра состояния передачи. От ставится если поля Типа и Версии не те или длина фрейма не соответствует заявленной в заголовке. То есть:

a) В дейтаграммах IPv4:
- Принятый тип Ethernet равен 0x0800, а Версия не равна 0x4
- Поле длины заголовка IPv4 содержит число меньше 0x5 (20 байт)
- Общая длина фрейма меньше заявленной в заголовке

b) В дейтаграммах IPv6:
- Принятый тип Ethernet равен 0x86DD, а Версия не равна 0x6
- Принят конец фрейма, а заголовок IPv6 (40 байт) или заголовок расширения ещё не закончился.

- **Контрольная сумма TCP/UDP/ICMP**

Контрольная сумма TCP/UDP/ICMP обслуживает заголовки IPv4 и IPv6 (включая заголовки расширения) и определяет, относятся ли вложенные данные к TCP, UDP или ICMP.

NB:

a) В не-TCP, -UDP или -ICMP/ICMPv6 фреймах контрольная сумма не проверяется и ничего не изменяется.

b) Фрагментированные фреймы IP (IPv4 или IPv6), фреймы IP с обеспечением безопасности (вроде заголовков аутентификации или вложенными данными безопасности) и фреймы IPv6 с заголовками маршрута на контрольную сумму не проверяются.

Контрольная сумма данных TCP, UDP или ICMP вычисляется и вставляется в заголовок двумя методами:
- Первый, псевдо-заголовки TCP, UDP и ICMPv6 не учитываются, они уже есть в поле контрольной суммы принятого фрейма. Принятая контрольная сумма учитывается и заменяется новым значением.
- Второй, поле контрольной суммы игнорируется, данные TCP, UDP и псевдо-заголовков ICMPv6 учитываются и результат пишется в поле контрольной суммы.

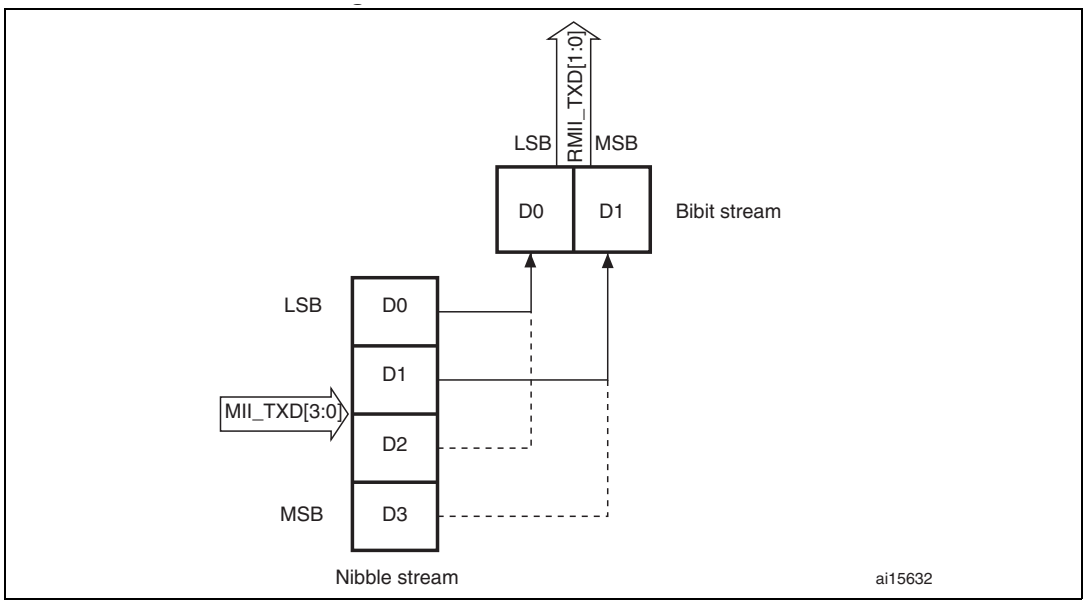
NB: Поскольку в пакетах ICMP-по-IPv4 псевдо-заголовков нет, то поле контрольной суммы пакета ICMP всегда должно быть равно 0x0000, иначе пакет получит неверную контрольную сумму.

Ошибка контрольной суммы ставится в статусе передачи когда:
- в режиме Накопления на передачу подаётся пакет без Конца фрейма в FIFO
- принятый пакет короче заявленного в принятом заголовке IP.

Если пакет длиннее заявленного, то это байты заполнителя. Они игнорируются и ошибки не возникает. При ошибке первого типа заголовки t TCP, UDP и ICMP не изменяются. При ошибке второго типа новая контрольная сумма всё же записывается.

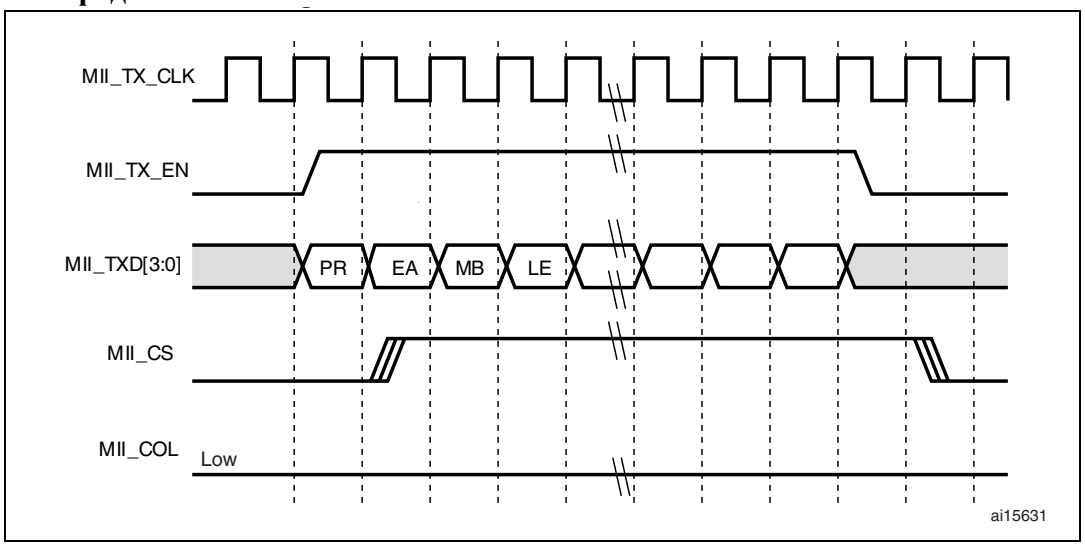
Порядок передачи битов MII/RMII

Полубайты от MII к RMII передаются парами, сначала младшие (D1 и D0) и затем D2 и D3.

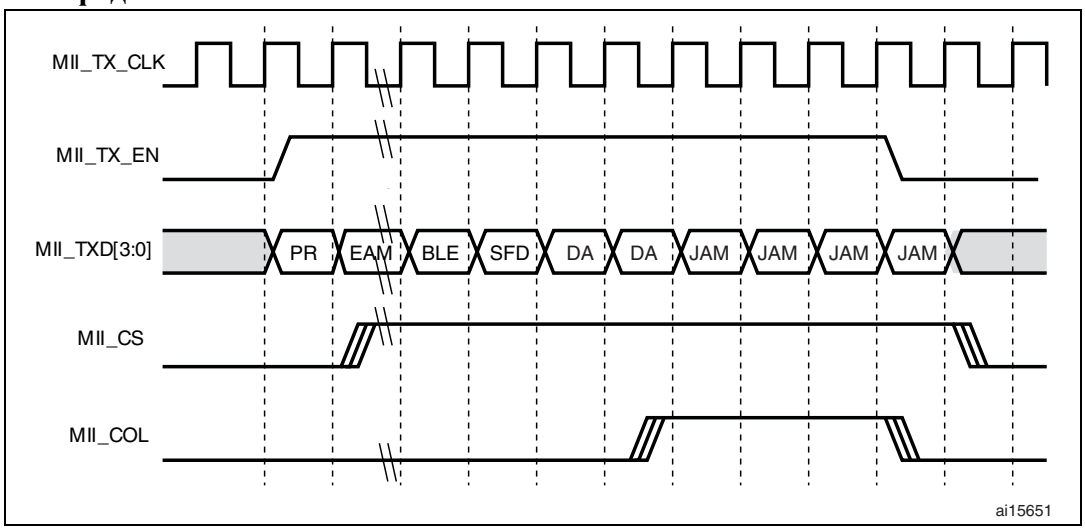


Времянки передач MII/RMII

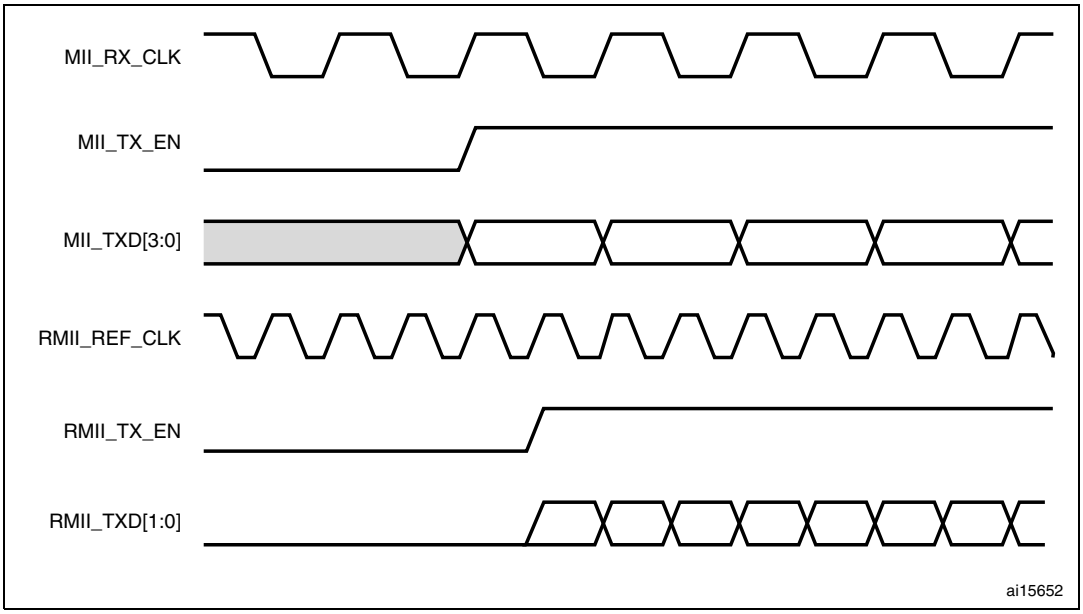
Передача без коллизий



Передача с коллизиями



Передача фрейма в режимах MII и RMII



29.5.3. Приём фрейма MAC

MAC пихает принятые фреймы в RxFIFO. DMA может начать передачи из FIFO в память после получения извещения о достижении порога заполнения (RTC в регистре ETH_DMAOMR).

В режиме по умолчанию DMA получает уведомление о доступности данных при записи в FIFO полного пакета данных или при достижении 64 байт (биты RTC в ETH_DMAOMR). Он передаёт данные из FIFO вплоть до конца пакета. После передачи EOF контроллер DMA получает слово статуса. В память могут попадать и некоторые сбойные фреймы.

В режиме Накопления из RxFIFO (бит RSF в ETH_DMAOMR) извлекаются только завершённые фреймы. Фреймы со сбоями выбрасываются (если это дозволено ядру).

Приём запускается при появлении SFD на MII. Преамбула и SFD отсекаются ядром. Поля заголовков проверяются фильтрами, поле FCS используется для проверки CRC фрейма. Фреймы, не прошедшие фильтры адреса, выбрасываются.

Протокол приёма

Отбрасываются преамбула и SFD принятого фрейма. По получению SFD, MAC пишет данные фрейма в FIFO, начиная с адреса получателя. Метки времени IEEE 1588, если разрешены, отмечают системное время появления SFD на MII. Программе они подаются независимо от результата фильтрации пакетов.

Если значение поля длины/типа меньше 0x600 и разрешена авто-стрижка CRC/расширителя, то MAC отсылает в RxFIFO ровно указанное число байтов, безвозвратно теряя остальные (включая поле FCS).

Если значение поля длины/типа больше или равно 0x600, то MAC отсылает в RxFIFO все полученные байты, независимо от разрешения авто-стрижки. Сторожевой таймер MAC включён по умолчанию, то есть, фреймы длиннее 2048 байт (DA + SA + LT + Data + pad + FCS) подвергаются обрезанию. Это выключается битом WD в регистре конфигурации MAC. Фреймы длиннее 16 KB обрезаются в любом случае и выдаётся статус таймаута сторожевого таймера.

Приём CRC: автоматическое удаление CRC и расширения

32-бит CRC полей от адреса назначения до FCS вычисляется по полиному:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

Для вычисления CRC, MAC принимает весь фрейм, независимо от разрешения авто-стрижки.

Контрольная сумма приёма

Приём контрольной суммы фреймов IPv4 и IPv6 разрешается установкой бита IPCO в регистре ETH_MACCR. Фреймы IPv4 и IPv6 различают по значениям 0x0800 и 0x86DD поля типа фрейма. Это относится и к фреймам с тегами VLAN. Вычисленная контрольная сумма заголовка IPv4 сравнивается с принятой. Бит ошибки заголовка ставится при любом несоответствии типа данных, версии заголовка IP и числе принятых данных, меньше указанного в заголовке IPv4 (или меньше 20 байтов для заголовков IPv4 и IPv6). Принятая контрольная сумма различает данные TCP, UDP и

ICMP в принятой дейтаграмме IP (IPv4 и IPv6). Проверка включает байты псевдо-заголовка TCP/UDP/ICMPv6. По результату ставится бит ошибки контрольной суммы в слове статуса. Также он ставится при несовпадении длины данных TCP, UDP и ICMP с заявленной в заголовке IP. Фрагментированные фреймы IP (IPv4 или IPv6), фреймы IP с обеспечением безопасности (вроде заголовков аутентификации или вложенными данными безопасности) и фреймы IPv6 с заголовками маршрута на контрольную сумму не проверяются. Факт проверки контрольной суммы отмечается в статусе приёма. В этой конфигурации байты контрольной суммы к принятым фреймам Ethernet не добавляются.

Таблица 212. Статусы фрейма

Бит 18: Фрейм Ethernet	Бит 27: Ошибка контрольной суммы заголовка	Бит 28: Ошибка контрольной суммы данных	Статус фрейма
0	0	0	Фрейм IEEE 802.3 (Поле длины меньше 0x0600).
1	0	0	Фрейм IPv4/IPv6 с ошибкой контрольной суммы.
1	0	1	Фрейм IPv4/IPv6 с ошибкой контрольной суммы данных
1	1	0	Фрейм IPv4/IPv6 с ошибкой контрольной суммы заголовка.
1	1	1	Фрейм IPv4/IPv6 с ошибкой контрольной суммы заголовка и данных.
0	0	1	Фрейм IPv4/IPv6 с ошибкой контрольной суммы заголовка без проверки данных.
0	1	1	Фрейм не типа IPv4 или IPv6 (без проверки)
0	1	0	Резерв

Контроллер фрейма приёма

Бит RA в регистре фильтра фреймов включает фильтрацию по адресам источника/получателя. При динамическом изменении параметров фильтра и сбое фильтра (DA-SA остаток фрейма отбрасывается и в слове статуса приёма отмечается причина сбоя. В режиме выключения питания Ethernet выбрасываются все принятые фреймы.

Управление приёмом

В полном дуплексе MAC может обнаруживать фреймы Паузы и приостанавливать передачу фреймов на указанное время. Это разрешается битом RFCE регистра ETH_MACFCR. При разрешённом управлении потоком приёма из него выявляются управляющие фреймы с множественным адресом назначения (0x0180 C200 0001). Бит PCF в регистре ETH_MACFFR разрешает передавать их программе.

В управляющем фрейме MAC различает поля типа, операции и таймера паузы. Если счётчик принятых байтов управляющего фрейма без ошибки CRC равен 64, то передатчик MAC приостанавливает выдачу фреймов на принятое число времени слота (64 байта режимов 10/100 Mbit/s). По фрейму паузы с нулевым временем текущая пауза обрывается.

Если тип управляющего фрейма не равен 0x8808), код не равен 0x00001 и длина не равна 64 байта, то пауза не генерируется. Фрейм паузы с множественным адресом получателя проходит фильтр.

При стоящем бите UPDF в регистре ETH_MACFCR фрейм паузы с уникальным адресом получателя проверяется по регистру 0 адреса MAC. Биты PCF регистра ETH_MACFFR определяют дополнительную фильтрацию адресов.

Приём нескольких фреймов

Поскольку статус следует сразу за данными, то FIFO может хранить любое число фреймов, вплоть до заполнения.

Обработка ошибок

Если RxFIFO заполняется до приёма EOF от MAC, то в статусе объявляется переполнение, выбрасывается весь фрейм и инкрементируется счётчик переполнений (регистр ETH_DMAMFBOCR). Биты FEF и FUGF в регистре ETH_DMAOMR позволяют RxFIFO отфильтровывать сбойные и маломерные фреймы.

В режиме Накопления (Store-and-forward) RxFIFO может отсевать все сбойные фреймы.

В режиме по умолчанию (Cut-through) RxFIFO может отсевать завершённые сбойные фреймы, незавершённые должен сливать DMA.

Слово статуса приёма

Закончив приём фрейма Ethernet, MAC передаёт DMA слово статуса приёма в битах[31:0] RDES0.

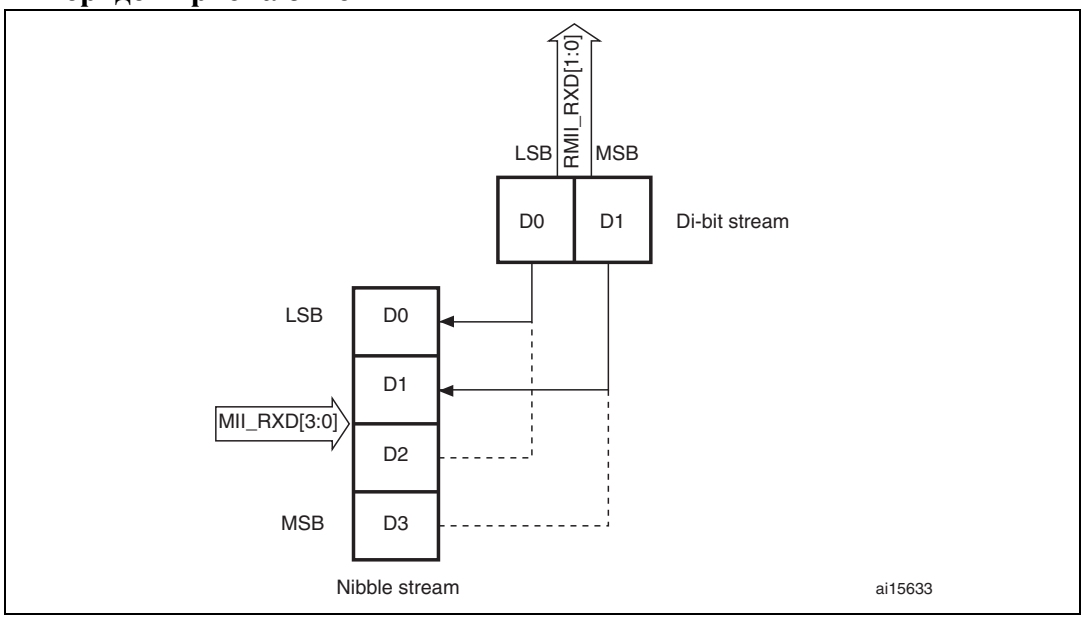
Длина приёма

В коммутаторах сети MAC обменивается с системой полными фреймами. Так что следить за длиной входящих фреймов для передачи их на выход нужно на уровне приложения. Фреймы нулевой длины попадают в RxFIFO при переполнении.

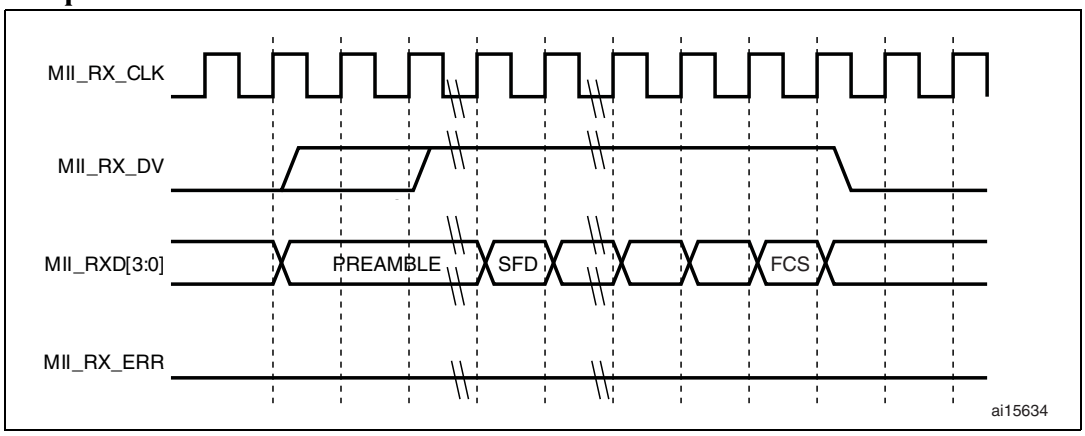
Порядок приёма битов MII/RMII

Полубайты из RMII в MII поступают парами, начиная с D0 и D1, затем D2 и D3)

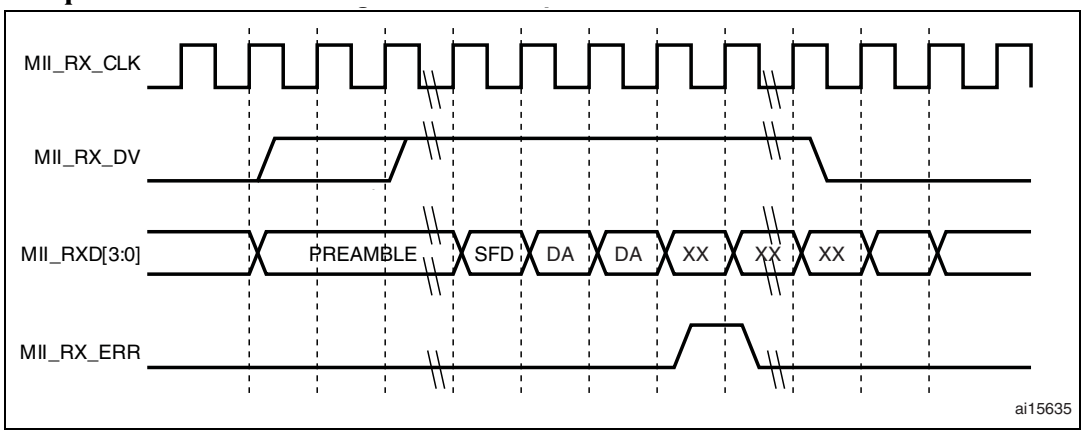
Порядок приёма битов



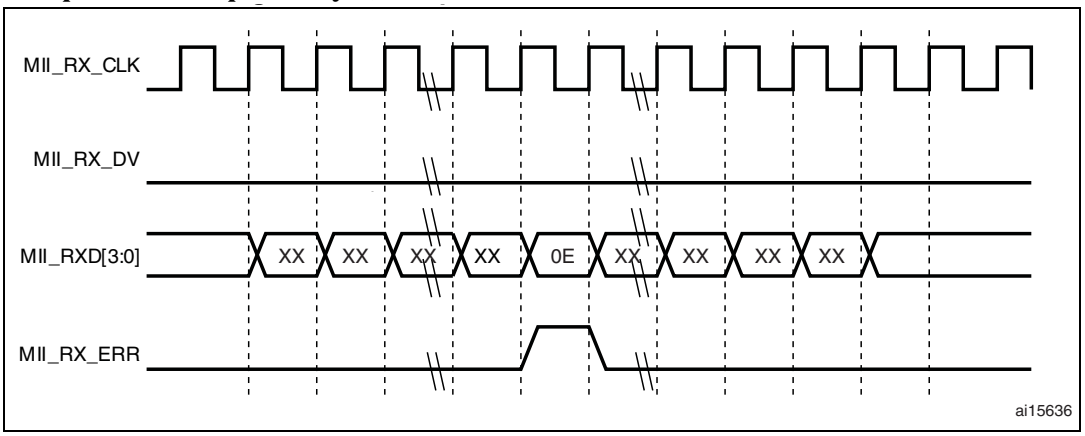
Приём без ошибок.



Приём с ошибками



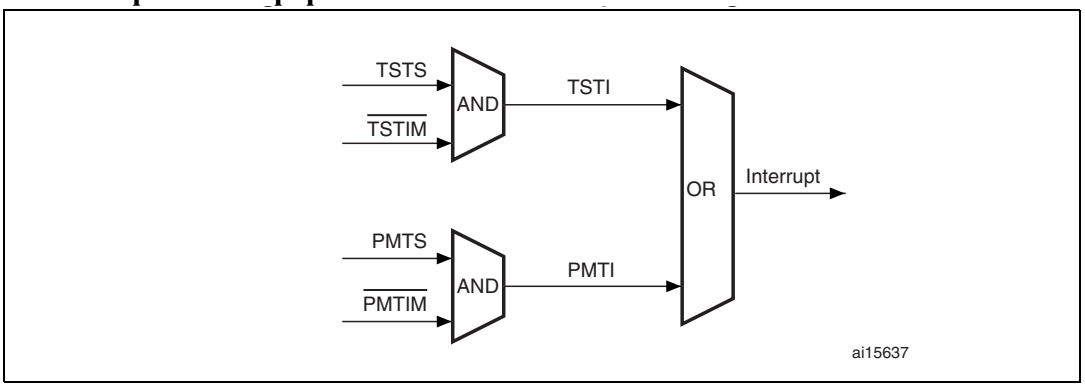
Приём с потерей несущей



29.5.4. Прерывания MAC

Источники прерываний отмечаются в регистре ETH_MACSR. Прерывания по ним разрешаются в регистре маски прерываний. Очищаются биты прерываний чтением соответствующих регистров состояния и других регистров источников.

Маскирование прерываний MAC



29.5.5. Фильтры MAC

Фильтры адреса

Все полученные фреймы фильтруются по адресу источника и назначения и им ставится статус. Параметры фильтрации устанавливаются программой. При фильтрации используется физический адрес (MAC) и Хэш-таблица.

Фильтр уникального адреса назначения

Тут используется до 4 MAC адресов. При снятом бите HU в регистре фильтра фрейма сравниваются все 48 бит полученного адреса. MacAddr0 разрешён всегда, адреса MacAddr1–MacAddr3 имеют свои биты разрешения. Так фильтруются групповые адреса. При стоящем бите HU для фильтрации используется 64-бит Хэш-таблица. Для этого в качестве её индекса используются 6 старших битов CRC. Значение 000000 соответствует биту 0, а 111111 выбирает бит 63 регистра Хэш-таблицы. Если выбранный бит стоит, то фрейм проходит сквозь фильтр.

Фильтр множественного адреса назначения

При стоящем бите PAM в регистре фильтра фреймов MAC пропускает все фреймы с множественными адресами. Если он сброшен, то фильтрация определяется битом HM этого регистра. В режиме полной фильтрации сравниваются регистры 1-3 адреса получателя MAC. Фильтрация групповых адресов поддерживается. При хэш-фильтрации используется 64-бит Хэш-таблица. Для этого в качестве её индекса используются 6 старших битов CRC. Значение 000000 соответствует биту 0, а 111111 выбирает бит 63 регистра Хэш-таблицы. Если выбранный бит стоит, то фрейм проходит сквозь фильтр.

Фильтр Хэш или полного адреса

Стоящий бит HPF в регистре фильтра позволяет пропускать фреймы, прошедшие один из фильтров DA (полного адреса или хэш) в зависимости от битов HU и HM bits. Если бит HPF сброшен, то работает только один фильтр.

Фильтр широковещательного адреса

При стоящем бите BFD в регистре фильтра все фреймы с широковещательными адресами теряются.

Фильтр уникального адреса источника

По умолчанию поле SA сравнивается с регистрами SA. Установка бита 30 в регистре DA ([1:3]) превращает его в SA. Групповая фильтрация SA поддерживается. При стоящем бите SAF в регистре фильтра фреймы, не прошедшие фильтр, отбрасываются, иначе им просто отмечается этот статус.

Инверсная фильтрация

Биты DAIF и SAIF позволяют инвертировать условие фильтрации фреймов с уникальными и множественными адресами.

Таблица 213. Фильтрация адреса назначения

Тип фрейма	PM	HPF	HU	DAIF	HM	PAM	DB	Что делает фильтр DA
Широковещательный	1	X	X	X	X	X	X	Пропускает
	0	X	X	X	X	X	0	Пропускает
	0	X	X	X	X	X	1	Выбрасывает
Уникальный	1	X	X	X	X	X	X	Пропускает все фреймы
	0	X	0	0	X	X	X	Пропускает по полному/групповому сравнению
	0	X	0	1	X	X	X	Выбрасывает по полному/групповому сравнению
	0	0	1	0	X	X	X	Пропускает по хэш сравнению
	0	0	1	1	X	X	X	Выбрасывает по хэш сравнению
	0	1	1	0	X	X	X	Пропускает по хэш или полному/групповом сравнению
	0	1	1	1	X	X	X	Выбрасывает по хэш или полному/групповом сравнению
Множественный	1	X	X	X	X	X	X	Пропускает все фреймы
	X	X	X	X	X	1	X	Пропускает все фреймы
	0	X	X	0	0	0	X	Пропускает по полному/групповому сравнению, теряет фреймы PAUSE при PCF = 0x
	0	0	X	0	1	0	X	Пропускает по хэш сравнению, теряет фреймы PAUSE при PCF = 0x
	0	1	X	0	1	0	X	Пропускает по хэш или полному/групповом сравнению, теряет фреймы PAUSE при PCF = 0x
	0	X	X	1	0	0	X	Выбрасывает по полному/групповом сравнению, теряет фреймы PAUSE при PCF = 0x
	0	0	X	1	1	0	X	Выбрасывает по хэш сравнению, теряет фреймы PAUSE при PCF = 0x
	0	1	X	1	1	0	X	Выбрасывает по хэш или полному/групповом сравнению, теряет фреймы PAUSE при PCF = 0x

Таблица 214. Фильтрация адреса источника

Тип фрейма	PM	SAIF	SAF	Что делает фильтр SA
Unicast	1	X	X	Пропускает все фреймы
	0	0	0	Пропускает статус по полному/групповом сравнению, фреймы не теряет
	0	1	0	Выбрасывает статус по полному/групповом сравнению, фреймы не теряет
	0	0	1	Пропускает по полному/групповом сравнению
	0	1	1	Выбрасывает по полному/групповом сравнению

29.5.6. Перемычка MAC

Подача передаваемых фреймов на приёмник разрешается битом Loopback в регистре ETH_MACCR.

29.5.7. Счётчики работы MAC: MMC

Это сбор статистики по передаваемым и принимаемым фреймам. Включает также 32-бит регистр управления, два 32-бит регистра общих прерываний (приём и передача) и два 32-регистра маски этих прерываний. На приёме учитываются только фреймы, прошедшие фильтр. В числе отбросов фреймы-коротышы учитываются.

Хорошо переданные и принятые фреймы

У хорошей передачи не появились ошибки:

+ Таймаут Трёпа
+ Нет/Потеря несущей
+ Поздняя коллизия
+ Исчерпание фрейма
+ Чрезмерная отсрочка
+ Излишние коллизии

У хорошего приёма не было ошибок:

+ Ошибка CRC
+ Фрейм-коротыш (короче 64 байт)
+ Ошибка выравнивания (только 10/100 Mbit/s)
+ Ошибка длины (только фреймы без типа)
+ Вне размера (только фреймы без типа, длиннее максимума)
+ Ошибка ввода MII_RXER

Максимальная длина фрейма:

+ Бестеговый фрейм = 1518
+ Фрейм VLAN = 1522

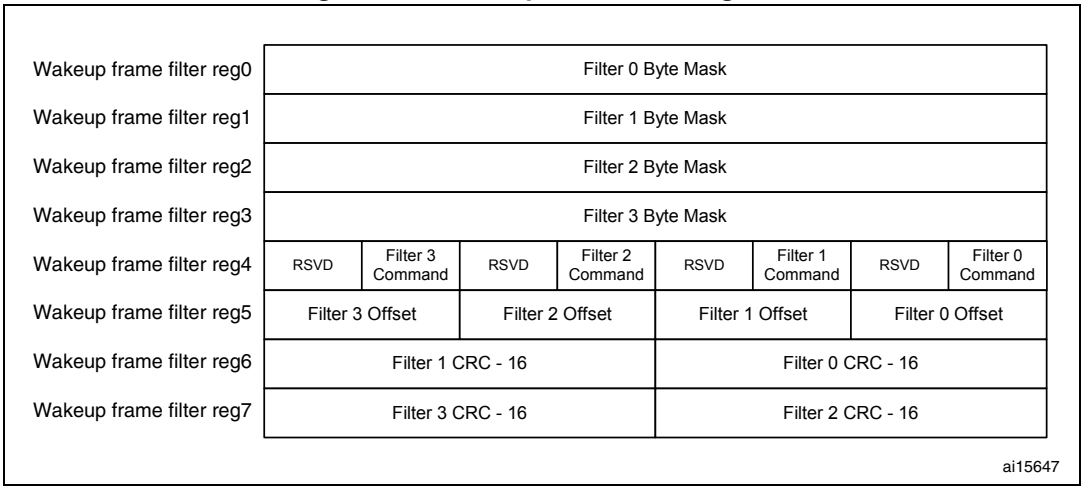
29.5.8. Управление питанием: PMT

Это приём Magic Packet и фреймов удалённой побудки с выдачей прерывания. PMT включается битами WFE и MPE в регистре ETH_MACPMTCSR. При включённом PMT все принятые фреймы отбрасываются. MAC выходит из выключенного питания по приёму Magic Packet или фрейма удалённой побудки при включённом распознавании.

Регистры фильтра фрейма побудки

Их 8 штук. Читают и пишут их последовательно за 8 обращений. При каждом смещается точка доступа к регистрам.

Фильтр фрейма побудки



- **Маска байта фильтра i**
  Определяет байт, проверяемый фильтром **i** (0, 1, 2 и 3) при определении фрейма побудки. MSB должен быть нулевым. Биты **j** [30:0] это маска байта. Если бит **j[номер байта]** стоит, то байт **Смещение фильтра i + j** обрабатывается блоком CRC, иначе игнорируется.
- **Команда фильтра i**
  Здесь 4 бита. Бит 3 это тип адреса получателя, 1 - множественный, 0 - уникальный. Биты 2 и 1 - резерв. Бит 0 это включение фильтра **i**, 1- Вкл.
- **Смещение фильтра i**
  Смещение проверяемого байта от начала фрейма для фильтра i. Минимальное значение равно 12, (13-й байт).
- **CRC-16 фильтра i**
  Нужное значение CRC_16 для проверяемого фрейма.

Обнаружение фрейма удалённой побудки

Выход MAC из режима сна по фрейму побудки разрешается установкой бита 2 в регистре ETH_MACPMTCSR. Программа пишет все 8 регистров фильтра application разрешает удалённую побудку. PMT имеет 4 фильтра для разных фреймов. Если фрейм проходит фильтры команды и CRC-16, то это побудка. Фрейм побудки проверяется на ошибки: длины, FCS, dribble bit, MII, коллизии и на фрейм-коротыш. Длинный фрейм (больше 512 байт) с нужным значением CRC считается побудкой. Обнаружение обновляется в регистре ETH_MACPMTCSR по каждому фрейму побудки. PMT может выдавать прерывание побудки.

Обнаружение Magic packet от фирмы AMD

Адрес получателя у Magic Packet может быть уникальным и широковещательным. Пакет сначала проходит фильтр адреса и затем проверяется на формат данных Wake-on-LAN. Это неразрывная последовательность 6 байтов единиц и 16 повторений адреса MAC. Обнаружение Magic Packet разрешается установкой бита 1 в регистре ETH_MACPMTCSR. PMT постоянно проверяет все адресованные сюда фреймы на последовательность Magic Packet. Она может начинаться в любом месте фрейма за адресами получателя и источника. Пакеты с множественным адресом тоже принимаются. Если адрес MAC узла равен 0x0011 2233 4455, то ищется последовательность:

Адрес назначения адрес источника ..... FFFF FFFF FFFF
0011 2233 4455 0011 2233 4455 0011 2233 4455 0011 2233 4455
0011 2233 4455 0011 2233 4455 0011 2233 4455 0011 2233 4455
0011 2233 4455 0011 2233 4455 0011 2233 4455 0011 2233 4455
0011 2233 4455 0011 2233 4455 0011 2233 4455 0011 2233 4455
...CRC

Обнаружение обновляется в регистре ETH_MACPMTCSR по каждому Magic Packet. PMT может выдавать прерывание побудки.

Выключение питания системы

При разрешённой EXTI line 19 блок PMT может обнаруживать фреймы в режиме Stop. При этом машина состояний приёмника должна быть включена стоящим битом RE, а машину состояний передатчика надо выключить очисткой бита TE в регистре ETH_MACCR. Ethernet DMA надо выключить очисткой битов ST и SR в регистре ETH_DMAOMR.

Рекомендуется следующая последовательность выключения и побудки:

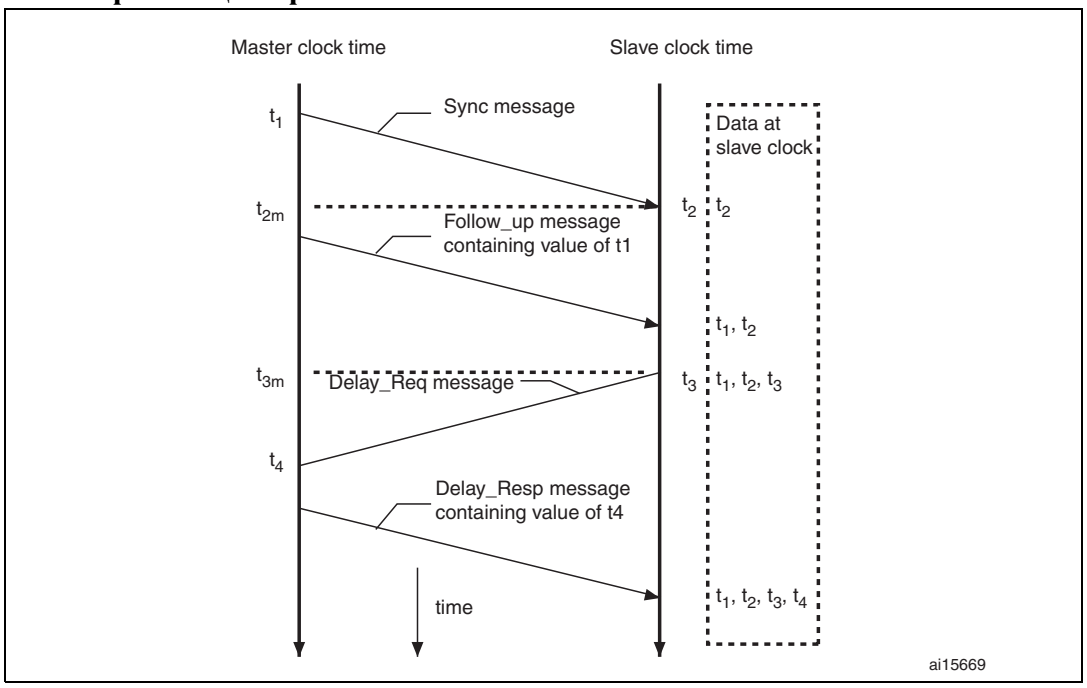
1. Выключаем передающий DMA и ждём конца передачи по прерыванию в ETH_DMASR.
2. Выключаем передатчик и приёмник MAC очисткой битов RE и TE в регистре ETH_MACCR.
3. Ждём пока приёмный DMA очистит RxFIFO.
4. Выключаем приёмный DMA.
5. Ставим и разрешаем EXTI line 19 на выдачу события или прерывания.
6. При прерывании, обработчик ETH_WKUP_IRQ должен снимать бит удержания EXTI line 19.
7. Включаем обнаружение побудки установкой бита MFE/ WFE в регистре ETH_MACPMTCSR.
8. Включаем экономный режим MAC установкой бита PD в регистре ETH_MACPMTCSR.
9. Включаем приёмник MAC установкой бита RE в регистре ETH_MACCR.
10. Идём в режим Stop
11. По приёму фрейма побудки блок Ethernet выйдет из режима экономии.

12. Читаем ETH_MACPMTCSR для очистки флага события, включаем машину состояний передатчика и приём и передачу DMA.
13. Конфигурируем системные такты: включаем HSE и ставим частоту.

29.5.9. Протокол точного времени (IEEE1588 PTP)

В системе выделяются ведущий и ведомые узлы.

Синхронизация времени сети



1. Ведущий выдаёт широковещательные сообщения PTP **Sync** с информацией о своём опорном времени всем узлам. Момент выхода из ведущей системы это t_1. Ethernet порты снимают его в момент появления на входе MII.
2. Ведомый принимает сообщение **Sync**, отмечая точный момент прихода по своему времени t_2.
3. Далее ведущий посылает ведомому сообщение **Follow_up**, содержащее t_1 чтобы знали.
4. Ведомый отсылает ведущему сообщение **Delay_Req**, отметив момент выхода из MII, t_3.
5. Ведущий принимает его, отмечая точный момент его его прихода t_4.
6. Ведущий отсылает ведомому время t_4 в сообщении **Delay_Resp**.
7. По значениям t_1, t_2, t_3 и t_4 ведомый синхронизирует своё опорное время с опорным временем ведущего.

В большинстве случаев протокол реализуется поверх уровня UDP, но нужна аппаратная поддержка точных отметок времени входа и выхода пакетов из MII.

Источник опорного времени

Спецификация IEEE 1588 определяет 64-бит формат отметок времени (старшие 32 бита это секунды, младшие 32 бита - наносекунды).

Для учёта опорного времени (Системное Время) и меток времени PTP использует внешний источник с частотой большей или равной разрешению счётчика меток времени. Точность синхронизации времён ведущего и ведомого должна быть около 100 ns. Она зависит от входной частоты, смещения генератора и частоты процедуры синхронизации. Такты Tx и Rx синхронизированы с доменом опорной частоты PTP и точность меток времени равна 1 периоду опорных тактов. Нарушив разрешение, мы добавим в метки времени ещё пол-периода.

Передача фреймов с PTP

Метки времени хранят момент появления SFD на MII. Необходимость снятия метки времени отмечается в дескрипторе передачи фрейма. Метки времени возвращаются программе вместе со словом статуса передачи в дескрипторе. 64-бит метка времени пишется в поля TDES2 и TDES3, TDES2 это 32 младших бита метки.

Приём фреймов с PTP

Если метки времени разрешены, то они отмечают конец приёма любого входного фрейма на MII. Они возвращаются программе вместе со словом статуса приёма в дескрипторе. 64-бит метка времени пишется в поля RDES2 и RDES3, RDES2 это 32 младших бита метки.

Методы коррекции Системного Времени

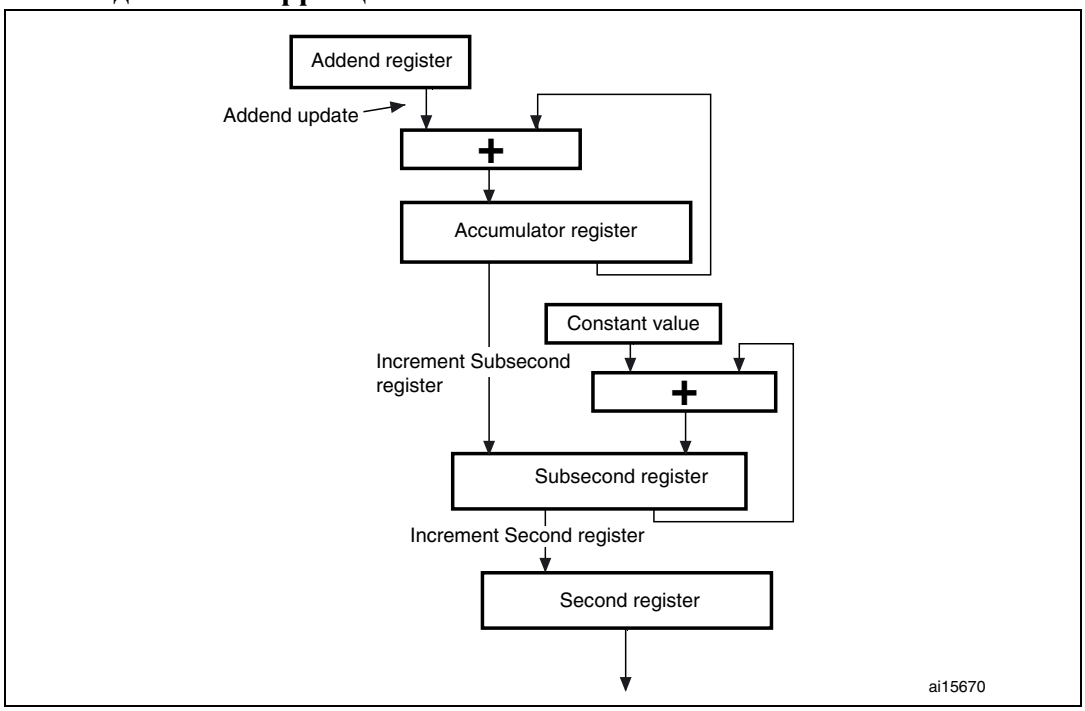
Их два: Грубый и Тонкий.

При грубом методе в регистр обновления метки времени пишется начальное значение или смещение. При инициализации этот регистр пишется в счётчик системного времени, при коррекции это значение добавляется или вычитается из счётчика.

При тонком методе отклонение частоты тактов ведомого от ведущего определяется за некоторый период времени, тогда как при грубом методе за один такт. Это такое сглаживание изменений.

При этом методе 32-бит аккумулятор накапливает содержимое 32-бит регистра слагаемого (Addend). Арифметический перенос аккумулятора используется для инкремента счётчика системного времени. Здесь аккумулятор служит высокоточным умножителем или делителем частоты.

Метод тонкой коррекции



Логика обновления системного времени требует частоты тактов 50 MHz для достижения 20 ns точности. Стало быть, если HCLK равна 66 MHz, то отношение частот FreqDivisionRatio равно 66 MHz/50 MHz = 1.32. Отсюда, слагаемое в регистре равно $2^{32}/1.32$ (0xC1F0 7C1F).

Например, если опорная частота уплывает до 65 MHz, то отношение 65/50 равно 1.3 и слагаемое становится равным $2^{32}/1.30$ (0xC4EC 4EC4). Если частота уходит в 67 MHz, То слагаемое становится равным 0xBF0 B7672. При нулевом отклонении в регистр слагаемого пишется 0xC1F0 7C1F ($2^{32}/1.32$).

На рисунке выше для инкремента суб-секундного регистра используется константа 0d43. Так достигается 20 ns точность системного времени.

Программа обновляет регистр слагаемого на основании сообщений **Sync**. Сначала в регистр слагаемого пишем значение компенсации частот:

FreqCompensationValue(0) = 2^{32}/FreqDivisionRatio

Если MasterToSlaveDelay (Задержка Ведущий-Ведомый) изначально полагается такой же как у последовательных сообщений **Sync**, то через несколько циклов **Sync** её можно определить точней и синхронизироваться с ведущим по алгоритму:

- В момент времени MasterSyncTime(n) ведущий посылает сообщение **Sync**. Ведомый получает его в момент SlaveClockTime(n) и вычисляет время ведущего:
  MasterClockTime(n) = MasterSyncTime(n) + MasterToSlaveDelay(n)

- Значение счётчика ведущего в текущем цикле **Sync** вычисляем как:
  MasterClockCount(n) = MasterClockTime(n) – MasterClockTime(n-1) (полагая, MasterToSlaveDelay одинаков для циклов **Sync** за номерами n и n – 1)
- Значение счётчика ведомого в текущем цикле **Sync** вычисляем как:
  SlaveClockCount(n) = SlaveClockTime(n) – SlaveClockTime(n – 1)
- Разность обоих счётчиков текущего цикла **Sync** равна:
  ClockDiffCount(n) = MasterClockCount(n) – SlaveClockCount(n)
- Множитель масштаба тактов ведомого равен:
  FreqScaleFactor(n) = (MasterClockCount(n) + ClockDiffCount(n)) / SlaveClockCount(n)
- Значение компенсации в регистр слагаемого равно:
  FreqCompensationValue(n) = FreqScaleFactor(n) × FreqCompensationValue(n – 1)

В теории это может сработать в течении одного цикла **Sync**, но в реальности задержки в работающей сети непредсказуемы.

Это алгоритм само-настройки: если начальные установки тактов ведомого относительно ведущего неверны, то через какое-то число циклов **Sync** всё пойдёт путём.

Программные шаги запуска выдачи меток времени

Это включается установкой бита 0 регистра `ETH__PTPTSCR`. Но после этого надо инициализировать счётчик меток времени, вот тогда и запустится. Последовательность такая:

1. Маскируем прерывание запуска меток времени битом 9 в регистре `MACIMR`.
2. Битом 0 в регистре меток разрешаем их.
3. Пишем регистр инкремента суб-секунд на основе частоты тактов PTP.
4. Если используем тонкую коррекцию, то пишем регистр слагаемого и ставим бит 5 регистра управления метками 5 (обновление регистра слагаемого).
5. Ждём его снятия (опросом).
6. Режим тонкой коррекции можно включить битом 1 в регистре управления метками.
7. В старший и младший регистры обновления меток времени пишем нужное значение времени.
8. Ставим бит 2 регистра управления (инициируем метки времени).
9. Счётчик меток времени стартует после записи в регистр обновления меток времени.
10. Включаем передатчик и приёмник MAC.

NB: Если метки времени отключаются снятием бита 0 регистра `ETH__PTPTSCR`, то для нового запуска надо повторить все упомянутые шаги.

Программные шаги Грубой коррекции системного времени

Это:

1. В старший и младший регистры обновления меток пишем смещение (плюс или минус).
2. Ставим бит `TSSTU` в регистре управления метками.
3. Обновление регистров отмечается снятием этого бита.

Программные шаги Тонкой коррекции системного времени

Это:

1. По упомянутому выше алгоритму вычисляем скорость, с которой надо увеличивать или уменьшать инкремент системного времени.
2. Обновляем метки времени.
3. Ждём достижения системным временем нужного значения, можно по прерыванию запуска меток времени.
4. В старший и младший регистры целевого времени пишем нужное значение. Снимаем маску прерываний меток времени очистки бита 9 в регистре `ETH_MACIMR`.
5. Ставим бит `TSARU` в регистре управления метками времени.
6. По этому прерыванию читаем регистр `ETH_MACSR`.
7. Переписываем в регистр слагаемого старое значение и снова ставим бит 5 в `ETH_TPTSCR`.

Запуск TIM2 от сигнала PTP

Когда системное время достигает целевого значения MAC выдаёт прерывание запуска, что вносит известную задержку и неопределённость в время выполнения команд.

Дабы избежать неопределённости, вместо прерывания используют выходной сигнал запуска от PTP. Он внутренне подключён к входу запуска TIM2 со всеми его возможностями. Так его синхронизируют с системным временем PTP. Неопределённость исчезает, поскольку такты таймера (PCLK1: TIM2 APB1) и PTP (HCLK) синхронизированы.

Сигнал подключается битом 29 регистра AFIO_MAPR.

Секундный сигнал от PTP

Бит 30 регистра AFIO_MAPR разрешает выдачу посекундного сигнала (PPS) длительностью 125ms на внешний вывод. Он поможет с помощью осциллоскопа проверить и уточнить синхронизацию узлов всей сети.

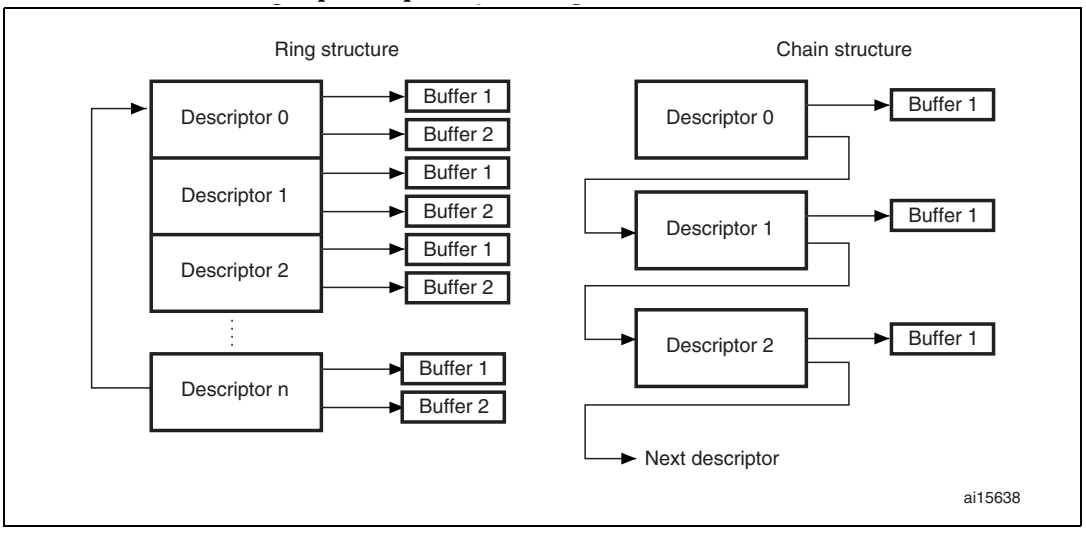
29.6. Функциональное описание Ethernet: Контроллер DMA

DMA имеет независимые машины приёма и передачи и пространство CSR. Для обмена с памятью используются TxFIFO и RxFIFO. Для передачи пакетов данных используются дескрипторы. Активно используются прерывания. Для работы DMA использует через две структуры данных:

- Регистры управления и состояния (CSR)
- Списки дескрипторов и буферы данных.

Дескрипторы лежат в памяти и работают как указатели буферов. Есть два списка дескрипторов: приёма и передачи. Базовые адреса списков пишутся в 3й и 4й регистры DMA. Список дескрипторов связанный (явно или неявно). Последний дескриптор может указывать на первый, образуя кольцо. Явная сцепка дескрипторов выполняется во втором адресе приёмных и передающих дескрипторов (RDES1[14] и TDES0[20]). Список дескрипторов лежит в физической памяти хоста. Один дескриптор может указывать максимум на два буфера. Это позволяет использовать не соприкасающиеся буферы. Буфер данных лежит в физической памяти и содержит часть фрейма, один целый фрейм, но не более. Буферы содержат только данные. Статус буфера отображается в дескрипторе. Цепочки данных относится к фреймам, занимающим несколько буферов, но один дескриптор не может охватывать несколько фреймов. По концу фрейма DMA переходит к следующему буферу. Цепочки данных можно разрешать и запрещать.

Кольцо и цепь дескрипторов



29.6.1. Инициализация передачи с DMA

Вот так:

1. В регистре ETH_DMABMR ставим параметры доступа к шине STM32F107xx.
2. В регистре ETH_DMAIER маскируем ненужные прерывания.
3. Программа создаёт списки дескрипторов прима и передачи и пишет их адреса начала в регистры ETH_DMARDLAR and ETH_DMATDLAR.
4. В регистрах MAC номер 1, 2 и 3 задаём параметры фильтрации.

5. В регистре ETH_MACCR задаём и включаем режимы передачи и приёма. Биты PS и DM ставятся на основе результатов переговоров (чтение из PHY).
6. В регистре ETH_DMAOMR битами 13 и 1 стартуем передачу и приём.
7. Машины передачи и приёма достают дескрипторы из списков и начинают работать независимо друг от друга.

29.6.2. Пакетный доступ хоста к шине

Если бит FB в регистре ETH_DMABMR стоит, то DMA обменивается с памятью пакетами фиксированной длины. Длина пакета ограничена значением поля PBL в регистре ETH_DMABMR. К дескрипторам всегда обращаются максимально возможными пакетами по 16 байт или PBL.

Tx DMA начинает передачу только если в TxFIFO хватает места для всего пакета или остатка для конца фрейма. Если интерфейс AHB поддерживает пакеты фиксированной длины, то выбирается лучшая комбинация передач INCR4, INCR8, INCR16 и SINGLE, иначе используются INCR (неопределённой длины) SINGLE.

Rx DMA начинает передачу только если в RxFIFO есть данные нужной длины или появился конец фрейма. Если интерфейс AHB поддерживает пакеты фиксированной длины, то выбирается лучшая комбинация передач INCR4, INCR8, INCR16 и SINGLE. Передачи, короче фиксированного пакета, дополняются пустыми передачами до нужной длины.

Если бит FB регистра ETH_DMABMR снят, то используются передачи INCR и SINGLE.

Если интерфейс AHB поставлен на передачи с выравненным адресом, то обе машины DMA смотрят, чтобы первая передача была равна или меньше значения PBL. Так все последующие передачи выравниваются на границу PBL. При PBL > 16 DMA выравнивает адреса на границу 16, поскольку AHB не поддерживает больше чем INCR16.

29.6.3. Выравнивание буфера данных хоста

Адрес начала буферов может начинаться с любого байта, но DMA передаёт данные в соответствии с шириной шины, заполняя ненужные линии пустыми данными. Обычно так бывает в начале или конце фрейма Ethernet.

- Пример чтения буфера:
  Адрес передающего буфера равен 0x0000 0FF2, передаётся 15 байт. DMA прочитает 5 слов с адреса 0x0000 0FF0, но в TxFIFO первые 2 байта и последние 3 не попадут. Кроме конца фрейма DMA всегда пишет в TxFIFO словами.
- Пример записи в буфер:
  Адрес приёмного буфера равен 0x0000 0FF2, передаётся 16 байт. DMA запишет пять 32-бит слов данных по адресу 0x0000 0FF0. Первые 2 байта первой передачи и последние 2 байта третьей передачи будут пустыми.

29.6.4. Вычисление размера буфера

В дескрипторах передающий DMA изменяет только статус (xDES0), размерами занимается программа. DMA передаёт ядру MAC точно указанное в поле размера буфера TDES1 число байт. Если дескриптор отмечен как первый (стоит бит FS в TDES0), то DMA отмечает первую передачу из буфера как начало фрейма. Если дескриптор отмечен как последний (бит LS в TDES0), то DMA отмечает последнюю передачу из буфера как конец фрейма.

Приёмный DMA передаёт данные в буфер вплоть до его заполнения или конца фрейма. Если дескриптор не отмечен как последний (бит LS в RDES0), то его буфер(ы) полный и число данных равно размеру буфера, и минус смещение указателя буфера при стоящем бите FS. Если указатель буфера выравнен на ширину шины, то смещение равно нулю. Если дескриптор отмечен как последний, то буфер может быть не полным. Число данных в этом буфере определяется вычитанием из длины фрейма (биты FL в RDES0[29:16]) суммы длин предыдущих буферов. Приёмный DMA всегда начинает передачу нового фрейма с новым дескриптором.

NB: Адрес начала приёмного буфера надо всегда выравнивать на ширину системной шины чтобы избежать выхода за его границы из-за требований DMA.

29.6.5. Арбитр DMA

Есть два типа арбитража доступа приёмного и передающего каналов DMA к шине AHB: круговой и приоритетный. При круговом (снят бит DA в ETH_DMABMR) выделяет шину в зависимости от бита PM в ETH_DMABMR. Если бит DA стоит, то приёмный DMA всегда приоритетнее передающего.

29.6.6. Извещение DMA об ошибке

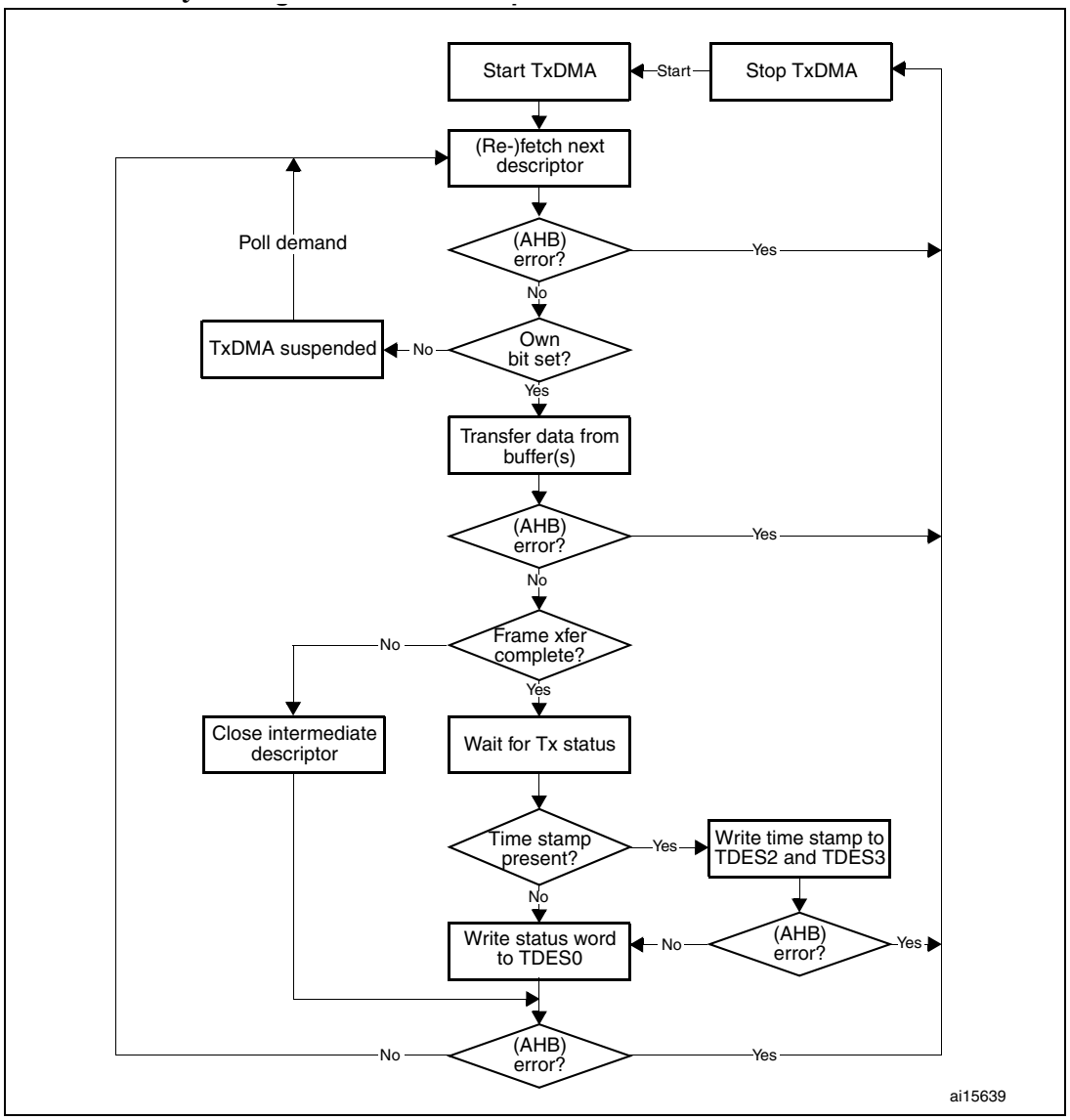
Если при передаче DMA ведомый сообщил об ошибке, то DMA останавливает все передачи и ставит биты ошибки и фатальной ошибки шины в регистре состояния (ETH_DMASR). Возобновить работу можно только после программного или аппаратного сброса и инициализации DMA.

29.6.7. Конфигурация TxDMA

Работа TxDMA: режим по умолчанию (не-OSF)

1. Пользователь оформляет дескриптор (TDES0-TDES3) и его буфер(ы) с передаваемым фреймом и ставит бит OWN (TDES0[31]).
2. По установке бита ST (ETH_DMAOMR[13]) DMA идёт в режим Run.
3. В том режиме DMA ищет в списке дескрипторов передаваемые фреймы. При появлении ошибки или дескриптора, принадлежащего CPU, передача останавливается и ставятся биты недоступности буфера (ETH_DMASR[2]) и Нормального прерывания (ETH_DMASR[16]). Машина переходит к шагу 9.
4. Из принадлежащего DMA дескриптора (стоит TDES0[31]) извлекается адрес буфера данных.
5. DMA передаёт данные из памяти.
6. Если фрейм лежит в буферах нескольких дескрипторов, то DMA закрывает промежуточный дескриптор и выбирает следующий. Шаги 3, 4 и 5 повторяются до конца фрейма.
7. По концу передачи фрейма в TDES0 пишется статус передачи, а в TDES2 и TDES3 может быть записана метка времени. Бит OWN снимается и дескриптор передаётся CPU.
8. По концу передачи фрейма со стоящем бите Прерывания по концу передачи (TDES1[31]) в последнем дескрипторе запрашивается прерывание Передачи (ETH_DMASR [0]). DMA возвращается к шагу 3.
9. В режиме Останова DMA, после получения запроса на просмотр передачи очереди пытается получить дескриптор передачи и вернуться к шагу 3. Прерывание исчерпания очищается.

Режим по умолчанию TxDMA

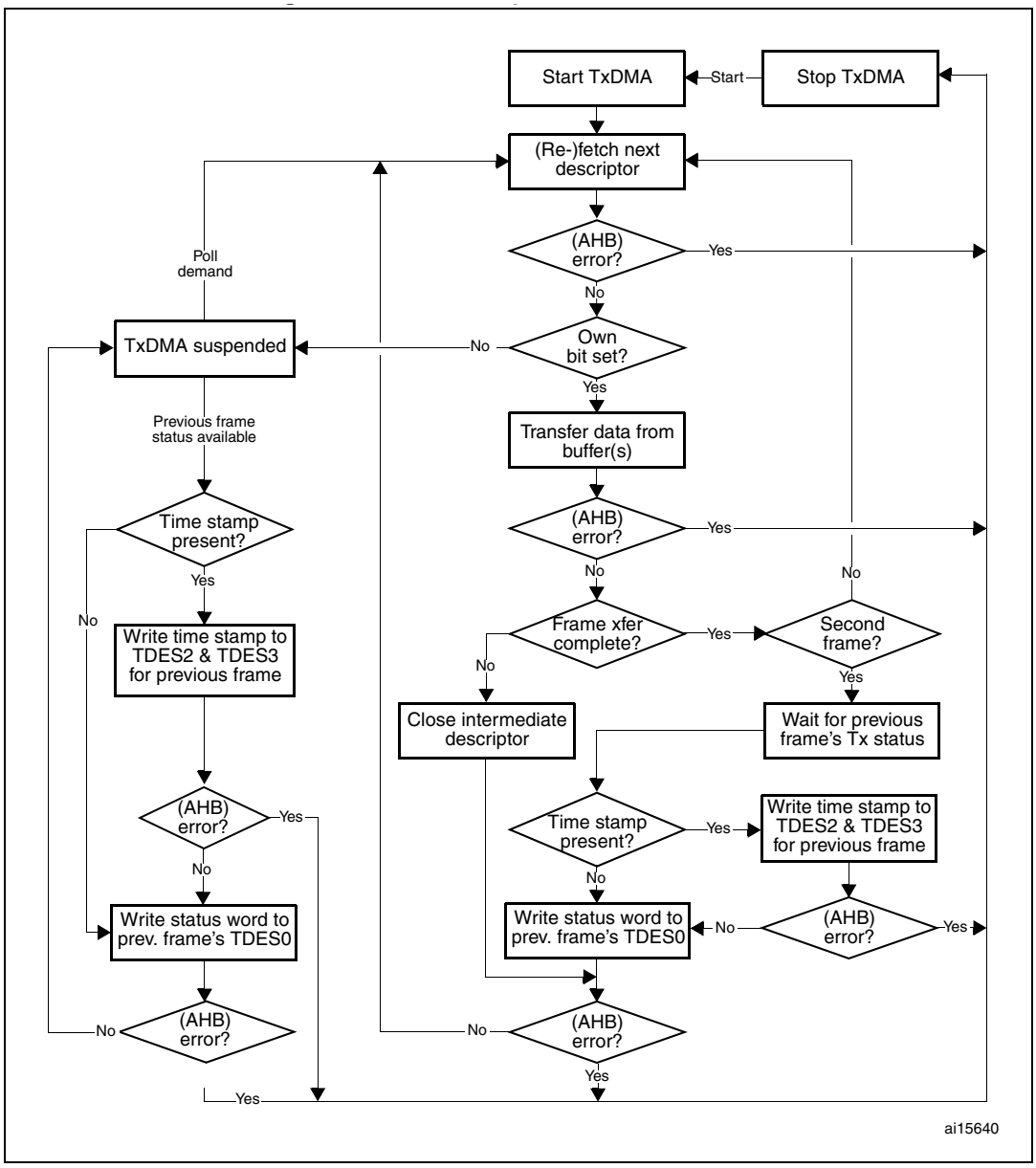


Работа TxDMA: режим OSF

При стоящем бите OSF в регистре ETH_DMAOMR[2] DMA может одновременно иметь два фрейма без закрытия дескриптора статуса первого. После завершения передачи первого фрейма DMA ищет в списке дескрипторов второй. Если он нормальный, то передаётся до записи статуса первого. Последовательность такая:

1. DMA выполняет шаги 1–6 режима по умолчанию TxDMA.
2. Затем DMA извлекает следующий дескриптор без закрытия первого.
3. Если DMA владеет этим дескриптором, то он извлекает адрес буфера, иначе идет на шаг 7.
4. DMA передаёт фрейм из памяти вплоть до его конца, закрывая промежуточные дескрипторы, если фрейм занимает несколько дескрипторов.
5. DMA ждёт статуса передачи и метку времени предыдущего фрейма, если надо, пишет метку времени в TDES2 и TDES3, статус передачи и снимает бит OWN в TDES0, закрывая дескриптор.
6. DMA ставит прерывание Передачи (если можно), извлекает следующий дескриптор и, если предыдущий статус нормальный, идёт к шагу 3. Иначе DMA идёт в Останов (шаг 7).
7. Если в режиме Останова DMA принимает запоздавшие статус и метку времени, то он пишет метку в TDES2 и TDES3, статус в соответствующий TDES0, ставит соответствующие прерывания и возвращается в Останов.
8. DMA выходит из Останова в Run (шаг 1 или 2 в зависимость от хранимого статуса) только по требованию Опроса передачи (регистр ETH_DMATPDR).

Режим OSF в TxDMA



Обработка фрейма передачи

Данные в буфере для DMA не должны содержать преамбулу, поля DA, SA, и Тип/Длина включаются всегда, а если выключено автоматическое добавление CRC и расширителя, то и эти поля и FCS. Фреймы могут охватывать цепочку буферов. Разделяются фреймы первым (TDES0[28]) и последним дескрипторами (TDES0[29]). Если установлены оба, то это единственный дескриптор. У промежуточных дескрипторов очищены и TDES0[28] и TDES0[29]. У последнего стоит TDES0[29]. После передачи последнего буфера, DMA возвращает в слово статуса дескриптора 0 (TDES0) статус из дескриптора последнего сегмента передачи. Если стоит Прерывание по завершению (TDES0[30]), то ставится Прерывание передачи (ETH_DMASR [0]), выбирается следующий дескриптор и процесс повторяется. Реальная передача начинается после заполнения TxFIFO до заданного порога (ETH_DMAOMR[16:14]) или полном фрейме в FIFO. Ещё есть опция режима Накопления (Store-and-Forward, ETH_DMAOMR[21]). По завершению передачи DMA дескриптор освобождается (снимается бит OWN TDES0[31]).

Остановка опроса передачи

Это происходит когда:

- DMA обнаруживает дескриптор, принадлежащий CPU (TDES0[31]=0), ставится флаг недоступности буфера (ETH_DMASR[2]). Программа должна вернуть дескриптор в собственность DMA и выдать требование Опроса.

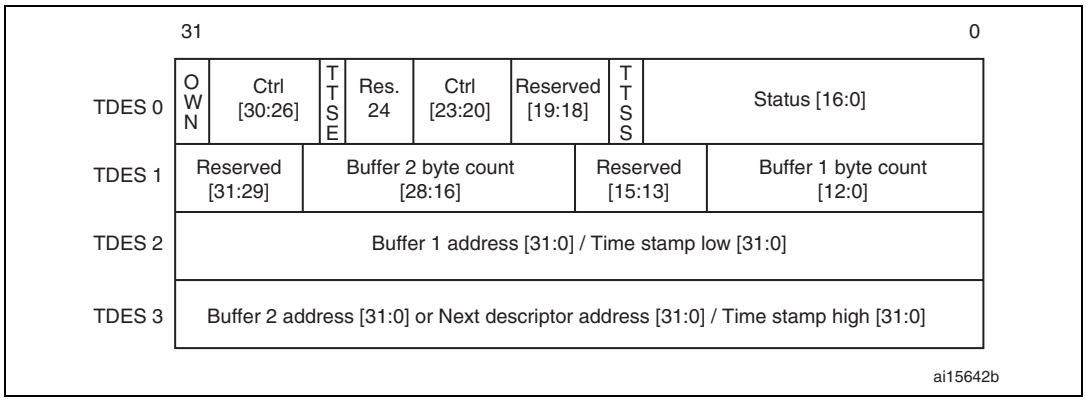
- Передача прекращена по исчерпанию буфера, ставится нужный бит в TDES0. Ставятся биты Ненормального прерывания (ETH_DMASR [15]) и Исчерпания (ETH_DMASR[5]), в дескриптор 0 пишется причина прерывания.

Если DMA уходит в Останов по первому условию, то ставятся биты Нормального прерывания (ETH_DMASR [16]) и Недоступного буфера (ETH_DMASR[2]). В обоих случаях позиция в списке передачи сохраняется на следующем за последним дескриптором, закрытым DMA. После снятия причины останова надо программно выдать требование Опроса передачи.

Дескрипторы TxDMA

Это четыре 32-бит слова.

Дескриптор передачи



- **TDES0: Дескриптор передачи Слово 0**

При инициализации дескриптора надо писать управляющие биты [30:26]+[23:20] и бит OWN [31]. Возвращая дескриптор, DMA снимает все управляющие биты и бит OWN и пишет только статус.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	IC	LS	FS	DC	DP	TTSE	Res	CIC		TER	TCH	Res.		TTSS	IHE
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	JT	FF	IPE	LCA	NC	LCO	EC	VF	CC				ED	UF	DB
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **OWN:** Бит владения
1 - дескриптором владеет DMA, 0 - CPU. DMA снимает этот бит по завершению передачи или по полному прочтению буфера. Бит владения первого дескриптора фрейма надо ставить последним.

— Бит 30 **IC:** Прерывание по завершению
Если стоит, то по завершению передачи фрейма ставится бит прерывания передачи (Регистр 5[0]).

— Бит 29 **LS:** Последний сегмент фрейма

— Бит 28 **FS:** Первый сегмент фрейма

— Бит 27 **DC:** Запрет добавления CRC в конце фрейма
Действует только для первого сегмента фрейма (TDES0[28]=1).

— Бит 26 **DP:** Запрет добавления расширения в фрейм, короче 64 байт
Если бит снят, то DMA автоматически добавляет расширитель и CRC в конец фрейма-коротыша, независимо от состояния бита DC (TDES0[27])bit. Действует только для первого сегмента фрейма (TDES0[28]=1).

— Бит 25 **TTSE:** Разрешение меток времени передачи
Если стоит одновременно с битом TSE(ETH_PTPTSCR бит 0), то в дескриптор передачи пишется метка времени. Действует только для первого сегмента фрейма (TDES0[28]=1).

— Бит 24 Резерв, не трогать.

— Биты 23:22 **CIC:** Управление вставкой контрольной суммы
Аппаратное вычисление и вставка CRC:
00: Запрещено
01: Можно только в заголовках IP
10: Можно в заголовках IP и данных, кроме псевдо-заголовков
11: Можно в заголовках IP, данных, и псевдо-заголовках.

— Бит 21 **TER:** Конец кольца передачи

Указывает, что достигнут конец списка дескрипторов. DMA возвращает адрес начала списка, создавая кольцо.

— Бит 20 **TCH:** Второй адрес в цепочке

Указывает, что второй адрес это адрес следующего дескриптора, а не второго буфера. При этом значение TBS2 (TDES1[28:16]) безразлично. Бит TDES0[21] старше,чем TDES0[20].

— Биты 19:18 Резерв, не трогать.

— Бит 17 **TTSS:** Метка времени передачи фрейма захвачена

Если стоит, то в TDES2 и TDES3 содержат метку времени. Действует только для последнего сегмента фрейма (TDES0[29]=1).

— Бит 16 **IHE:** Ошибка заголовка IP

Длина в заголовке пакета IPv4 не совпала с числом принятых от программы байтов. В фреймах IPv6 длина главного заголовка не равна 40 байтов. Поле длины/типа фреймов IPv4 и IPv6 должно соответствовать версии заголовка IP пакета. У фреймов IPv4 поле длины заголовка меньше 0x5.

— Бит 15 **ES:** Индикатор ошибок

Это логическое OR битов:

TDES0[14]: Таймаут трёпа
TDES0[13]: Слив фрейма
TDES0[11]: Потеря несущей
TDES0[10]: Нет несущей
TDES0[9]: Поздняя коллизия
TDES0[8]: Излишек коллизий
TDES0[2]: Излишняя отсрочка
TDES0[1]: Исчерпание
TDES0[16]: Ошибка заголовка IP
TDES0[12]: Ошибка данных IP

— Бит 14 **JT:** Таймаут трёпа передатчика MAC

Работает при снятом бите JD регистра конфигурации MAC.

— Бит 13 **FF:** Фрейм слит DMA/MTL по команде от CPU

— Бит 12 **IPE:** Ошибка данных IP

Длина данных в заголовке IPv4 или IPv6 не совпала в с длиной пакета TCP, UDP или ICMP в дейтаграмме IP.

— Бит 11 **LCA:** Потеря несущей при передаче фрейма

Сигнал MII_CRS был неактивен в течении одного или более тактов передачи. Действителен только при передаче фрейма без коллизий в полу-дуплексе.

— Бит 10 **NC:** No carrier

При передаче сигнала Carrier Sense (Есть несущая) от PHY не было.

— Бит 9 **LCO:** Поздняя коллизия

Коллизия появилась позже окна коллизий (время 64 байтов, включая преамбулу, в режиме MII). Недействителен при стоящем бите Исчерпания.

— Бит 8 **EC:** Излишек коллизий

Передача прекращена после 16 коллизий при попытках передать текущий фрейм. Если бит RD (Запрет повторных попыток) в регистре конфигурации MAC стоит, то этот бит ставится после первой же коллизии передачи.

— Бит 7 **VF:** Передан фрейм VLAN

— Биты 6:3 **CC:** Счётчик коллизий передачи

При стоящем бите TDES0[8] это брехня.

— Бит 2 **ED:** Излишняя отсрочка

Передача прекращена из-за задержки больше 24 288 времён бита. Проверка включается битом DC в регистре управления MAC.

— Бит 1 **UF:** Исчерпание передающего буфера

Данные из памяти не успели поступить в буфер и он опустел во время передачи фрейма. Машина уходит в Останов, одновременно ставятся Исчерпание (Регистр 5[5]) и прерывание Передачи (Регистр 5[0]).

— Бит 0 **DB:** Отсрочка

MAC отложил передачу из-за несущей. Работает только в полу-дуплексе.

- **TDES1: Дескриптор передачи Слово 1**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			TBS2													Reserved			TBS1												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:29 Резерв, не трогать.

— Биты 28:16 **TBS2:** Размер передающего буфера 2 в байтах
Недействителен при стоящем TDES0[20].

— Биты 15:13 Резерв, не трогать.

— Биты 12:0 **TBS1:** Размер передающего буфера 1 в байтах
Если 0, то DMA использует буфер 2 или следующий дескриптор в зависимости от TCH (TDES0[20]).

- **TDES2: Дескриптор передачи Слово 2**

Указатель адреса первого буфера данных дескриптора или данные метки времени.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBAP1/TBAP/TTSL																															
rw																															

— Биты 31:0 **TBAP1:** Указатель адреса буфера 1/Младшее слово метки времени
TBAP: В момент передачи дескриптора DMA (установка бита OWN в TDES0) это физический адрес буфера 1. Ограничений по выравниванию нет.
TTSL: Если разрешено битом TTSE в TDES0 последнего сегмента, то перед снятием бита OWN в TDES0, DMA пишет сюда младшие 32 бита метки времени.

- **TDES3: Дескриптор передачи Слово 3**

Указатель адреса данных второго буфера дескриптора или данные метки времени.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBAP2/TBAP2/TTSH																															
rw																															

— Биты 31:0 **TBAP2:** Указатель адреса буфера 2 (Адрес следующего дескриптора)/Старшее слово метки времени
TBAP2: Если используется кольцо дескрипторов, то в момент передачи дескриптора DMA (установка бита OWN в TDES0) это физический адрес буфера 2. Если второй буфер в цепочке (TDES1[20]=1), то это адрес следующего дескриптора, при этом указатель должен выравнен на ширину шины (Младшие биты игнорируются внутри.)
TTSH: Если разрешено битом TTSE в TDES0 последнего сегмента, то перед снятием бита OWN в TDES0, DMA пишет сюда старшие 32 бита метки времени.

29.6.8. Конфигурация Rx DMA

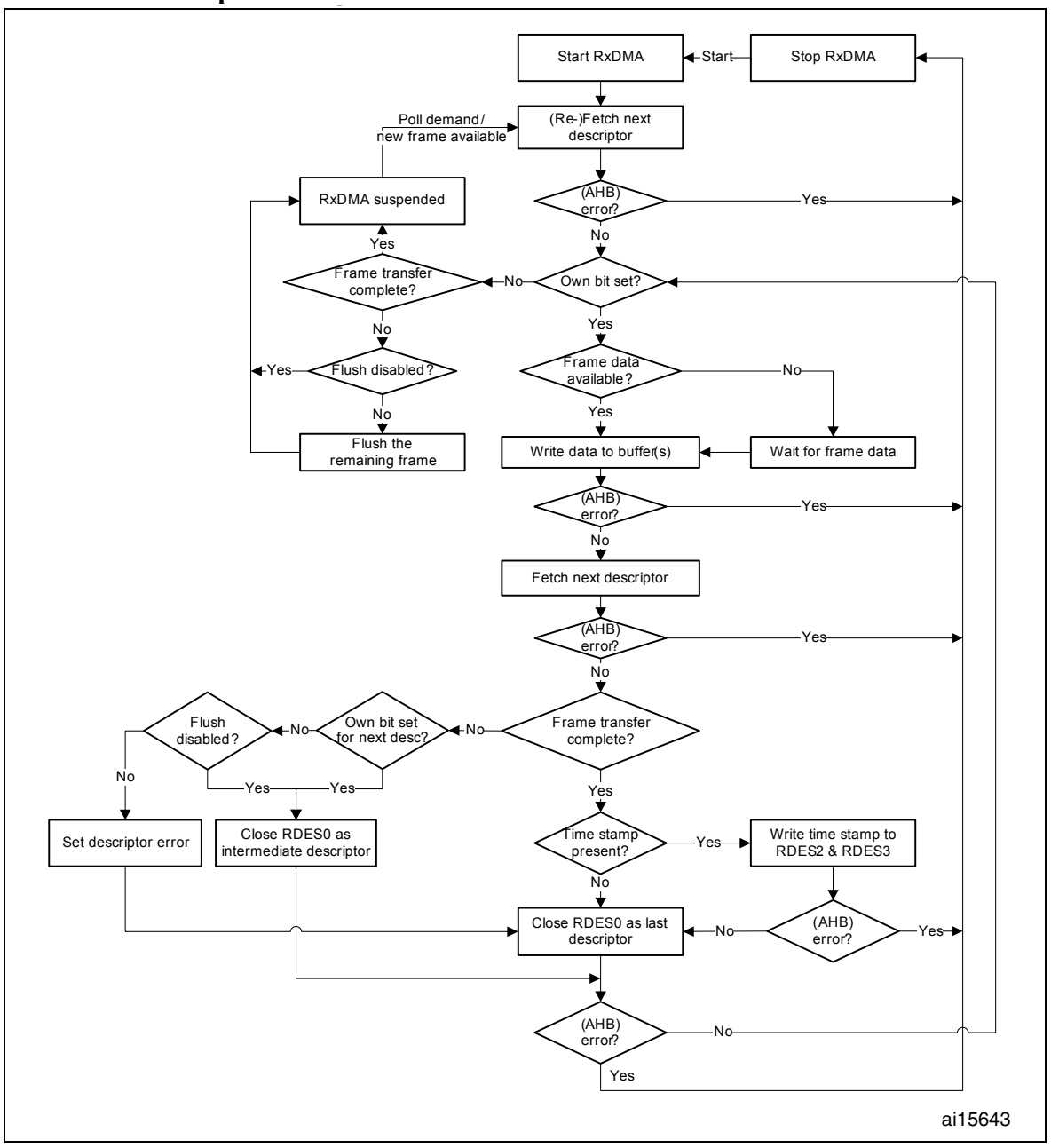
Последовательность такая:

1. CPU оформляет дескрипторы (RDES0-RDES3) и ставит бит OWN (RDES0[31]).
2. По установке бита SR (ETH_DMAOMR[1]) DMA идёт в режим Run. Теперь DMA опрашивает список дескрипторов в поисках свободного. Если извлечённый дескриптор принадлежит CPU, то DMA идёт в Останов и к шагу 9.
3. DMA достаёт из дескрипторов адреса буферов данных.
4. Входные фреймы размещаются в буферы данных дескриптора.
5. При заполнении буфера или при завершении фрейма машина достаёт следующий дескриптор.
6. Если передача текущего фрейма завершена, то DMA идёт к шагу 7. Если следующий дескриптор не принадлежит DMA, а фрейм ещё не завершён (не принят EOF), то DMA ставит бит ошибки дескриптора в RDES0 (если не выключен слив). DMA закрывает текущий дескриптор (снимает бит OWN) и делает его промежуточным, снимая бит LS в RDES1, и идёт на шаг 8. Если следующий дескриптор принадлежит DMA, но текущий фрейм ещё не закончился, то DMA закрывает текущий дескриптор как промежуточный и возвращается к шагу 4.
7. Если метки времени разрешены, то DMA пишет её в RDES2 и RDES3 текущего дескриптора, затем пишет статус в RDES0, снимает бит OWN и ставит бит LS.

8. Машина проверяет бит OWN последнего дескриптора. Если он принадлежит DMA, то идёт на шаг 4 и ждёт следующего фрейма, иначе ставит бит недоступности буфера (ETH_DMASR[7]) и уходит в Останов (шаг 9).
9. Перед Остановом частичные фреймы сливаются из FIFO (бит 24 регистра ETH_DMAOMR).
10. DMA выходит из Останова по требованию опроса или появлению в FIFO начала следующего фрейма. Он идёт к шагу 2 и повторно извлекает следующий дескриптор.

DMA подтверждает приём статуса после записи метки времени и самого статуса в дескриптор. Если в CSR метки времени разрешены, а действительной метки фрейма нет (например, FIFO заполнился до её записи), то DMA пишет в RDES2 и RDES3 другие.

Работа DMA приёма



Получение дескриптора приёма

DMA пытается заранее раздобыть дополнительный дескриптор приёма когда:

- При входе и режим Run бит Start/Stop (ETH_DMAOMR[1]) стоит.
- Буфер данных текущего дескриптора полон, а фрейм ещё не закончился.
- Контроллер завершил приём фрейма, а текущий дескриптор ещё не закрыт.
- Буфер принадлежит CPU (RDES0[31] = 0) и принят новый фрейм.
- Получено требование опроса.

Приём фрейма

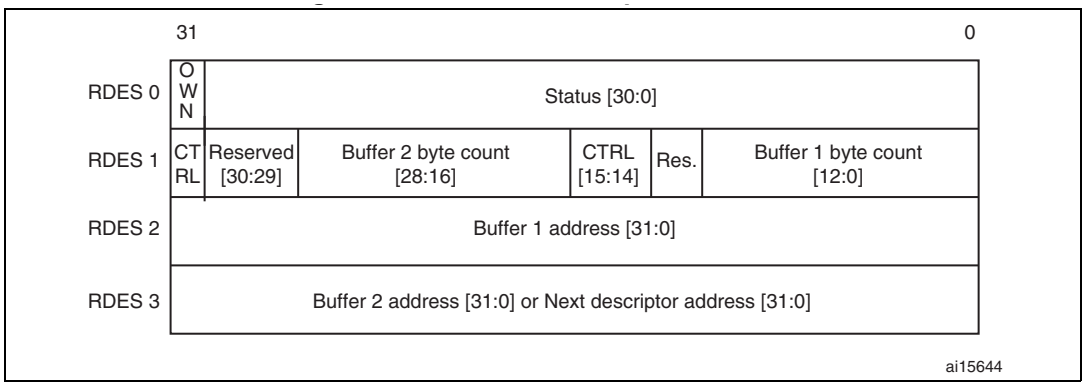
MAC передаёт фрейм в память только если он прошёл фильтр адреса и размером больше или равен установленному порогу RxFIFO, или в режиме Накопления в FIFO записан полный фрейм. Если бит Принять всё (ETH_MACFFR[31]) снят, то фреймы, не прошедшие фильтр адреса, выбрасываются. Фреймы, короче 64 байт из-за коллизии или преждевременного завершения, из RxFIFO можно вычищать. Передача данных фрейма в буфер текущего дескриптора начинается после достижения порога заполнения RxFIFO. Если DMA готов передавать данные в память, то для разделения фреймов ставится бит первого дескриптора (RDES0[9]). Дескрипторы освобождаются по снятию бита OWN (RDES0[31]), заполнению буфера данных или по записи последнего сегмента. Если фрейм лежит в одном дескрипторе, то стоят биты последнего (RDES0[8] и первого (RDES0[9]) дескриптора. DMA извлекает следующий дескриптор, в предыдущем дескрипторе фрейма ставит бит последнего дескриптора (RDES0[8]) и освобождает биты статуса RDES0. Наконец, DMA ставит бит прерывания приёма (ETH_DMASR [6]). Процесс повторяется до обнаружения дескриптора, принадлежащего CPU. При этом ставится бит недоступности буфера (ETH_DMASR[7]) и всё идёт в Останов. Позиция в списке приёма сохраняется.

Режим Останова приёма

Если в режиме Останова появляется новый фрейм, то DMA снова извлекает текущий дескриптор из памяти. Если он принадлежит DMA, то машина идёт в режим Run и начинает приём фрейма. Иначе DMA выбрасывает верхний фрейм из RxFIFO и наращивает счётчик выброшенных фреймов. Если в RxFIFO лежит более одного фрейма, то процесс повторяется. Выброс или слив верхнего фрейма из RxFIFO можно запретить установкой бита DFRF в регистре режима работы DMA. В таком случае ставится бит недоступности буфера и машина идёт в Останов.

Дескрипторы Rx DMA

Это структура из четырёх 32-бит слов.



- **RDES0: Дескриптор приёма Слово 0**

31	30	29:16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OWN	AFM	FL	ES	DE	SAF	LE	OE	VLAN	FS	LS	IPHCE	LCO	FT	RWT	RE	DE	CE	PCE
rw																		

— Бит 31 **OWN:** Бит принадлежности дескриптора
0 - CPU, 1- DMA. DMA снимает этот бит по завершению приёма фрейма или заполнению буфера.

— Бит 30 **AFM:** Фрейм не прошёл фильтр DA.

— Биты 29:16 **FL:** Длина фрейма в байтах в памяти (включая CRC).
Действителен в последнем дескрипторе (стоит RDES0[8]) при снятой ошибке **DE** (RDES0[14]).

— Бит 15 **ES:** Ошибка, логическое OR битов:
RDES0[1]: Ошибка CRC
RDES0[3]: Ошибка приёма
RDES0[4]: Таймаут сторожевого таймера
RDES0[6]: Поздняя коллизия
RDES0[7]: Фрейм-гигант (Не работает при ошибке контрольной суммы IPV4 (RDES0[7].)
RDES0[11]: Переполнение буфера
RDES0[14]: Ошибка последнего дескриптора (RDES0[8]=1).

— Бит 14 **DE:** Ошибка последнего дескриптора (RDES0[8]=1)

Не вместившийся в буфер фрейм усечён, следующий дескриптор не принадлежит DMA.

— Бит 13 **SAF:** Фрейм не прошёл фильтр SA.

— Бит 12 **LE:** Ошибка длины

Длина принятого фрейма не совпала с полем Типа/Длины, бит действителен при (RDES0[5]=0).

— Бит 11 **OE:** Переполнение входного буфера.

— Бит 10 **VLAN:** Тег фрейма VLAN.

— Бит 9 **FS:** Первый дескриптор

Если адрес первого буфера равен 0, то берётся адрес второго буфера. Если и он равен 0, то берётся следующий дескриптор.

— Бит 8 **LS:** Последний дескриптор фрейма.

— Бит 7 **IPHCE:** Ошибка контрольной суммы заголовка IPv4 или IPv6

Поле Типа не соответствует Версии заголовка IP, ошибка контрольной суммы заголовка IPv4, фрейм короче значения, указанного в заголовке IP.

— Бит 6 **LCO:** Позднее появление коллизии в полу-дуплексе.

— Бит 5 **FT:** Тип фрейма

0 - Фрейм IEEE802.3

1 - Фрейм Ethernet (поле LT больше или равно 0x0600).

Бит недействителен для фреймов короче 14 байт.

— Бит 4 **RWT:** Таймаут сторожевого таймера приёма, фрейм усечён.

— Бит 3 **RE:** Ошибка приёма, RX_ERR появился при стоящем RX_DV.

— Бит 2 **DE:** Ошибка дробности (Dribble)

Нечётное число полубайтов в режиме MII.

— Бит 1 **CE:** Ошибка CRC последнего дескриптора (RDES0[8]=1)

— Бит 0 **PCE**: Ошибка CRC данных

Вычисленная CRC пакетов TCP, UDP или ICMP не совпала с заявленной или длина принятых данных не совпала с заявленной в дейтаграммах IPv4 или IPv6 фрейма Ethernet.

Бит 5: FT	Бит 7: IPHCE	Бит 0: PCE	Статус фрейма
0	0	0	Фрейм IEEE 802.3 (Поле длины меньше 0x0600.)
1	0	0	Фрейм IPv4/IPv6, ошибки CRC нет
1	0	1	Фрейм IPv4/IPv6, ошибка CRC данных
1	1	0	Фрейм IPv4/IPv6, ошибка CRC заголовка IP
1	1	1	Фрейм IPv4/IPv6, ошибки CRC данных и заголовка IP
0	0	1	Фрейм IPv4/IPv6 ошибки CRC заголовка IP, данные не проверялись
0	1	1	Фрейм ни IPv4, ни IPv6 (CRC не проверялась.)
0	1	0	Резерв

- **RDES1: Дескриптор приёма Слово 1**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIC	RBS2		RBS2													RER	RCH	Reserved	RBS												
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **DIC:** Запрет прерывания завершения по биту RS (CSR5[6]).

— Биты 30:29 Резерв, не трогать.

— Биты 28:16 **RBS2:** Размер приёмного буфера 2 в байтах при RDES1[14]=0

Должен быть кратен 4, 8 или 16, в зависимости от ширины шины (32, 64 или 128), даже при невыравненном адресе буфера. Иначе поведение непредсказуемо.

— Бит 15 **RER:** Конец кольца приёма

Последний дескриптор в списке, DMA возвращает адрес начала списка, создавая кольцо.

— Бит 14 **RCH:** Второй адрес в цепочке

Второй адрес в дескрипторе это адрес следующего дескриптора, а не второго буфера. Значение поля RBS2 (RDES1[28:16]) не волнует. RDES1[15] старше RDES1[14].

— Бит 13 Резерв, не трогать.

— Биты 12:0 **RBS1:** Размер приёмного буфера 1 в байтах

Должен быть кратен 4, 8 или 16, в зависимости от ширины шины (32, 64 или 128), даже при невыравненном адресе буфера. Иначе поведение непредсказуемо. Если равен 0, то DMA использует адрес буфера 2 или следующий дескриптор, в зависимости от бита RCH.

- **RDES2: Дескриптор приёма Слово 2**

Указатель адреса первого буфера данных дескриптора или данные метки времени.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBP1 / RTSL																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **RBAP1 / RTSL:** Указатель адреса буфера 1/Младшее слово метки времени

RBAP1: В момент передачи дескриптора DMA (установка бита OWN в TDES0) это физический адрес буфера 1. Ограничений по выравниванию нет, за исключением передачи начала фрейма, когда DMA пишет в память с нулевыми битами RDES2[3/2/1:0], смещая реальные данные.

RTSL: Если это последний фрейм и метки времени разрешены, то перед снятием бита OWN в TDES0, DMA пишет сюда младшие 32 бита метки времени.

- **RDES3: Дескриптор приёма Слово 3**

Указатель адреса данных второго буфера дескриптора, адрес следующего дескриптора или данные метки времени.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBP2 / RTSH																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **RBAP2/RTSH:** Указатель адреса буфера 2 (Адрес следующего дескриптора)/Старшее слово метки времени

TTSH: Если разрешено битом TTSE в TDES0 последнего сегмента, то перед снятием бита OWN в TDES0, DMA пишет сюда старшие 32 бита метки времени.

RBAP2: Если используется кольцо дескрипторов, то в момент передачи дескриптора DMA (установка бита OWN в RDES0) это физический адрес буфера 2.
Если бит цепочки (RDES1[24]) стоит, то это адрес следующего дескриптора. Он должен быть выравнен на границу ширины шины (RDES3[3,2,1:0] = 0, для ширины шины 128, 64 или 32. Младшие биты игнорируются внутри.)
Если бит цепочки (RDES1[24]) сброшен, то ограничений по выравниванию нет, за исключением передачи начала фрейма, когда DMA пишет в память с нулевыми битами RDES2[3/2/1:0], смещая реальные данные.

RTSH: Если это последний фрейм и метки времени разрешены, то перед снятием бита OWN в TDES0, DMA пишет сюда старшие 32 бита метки времени.

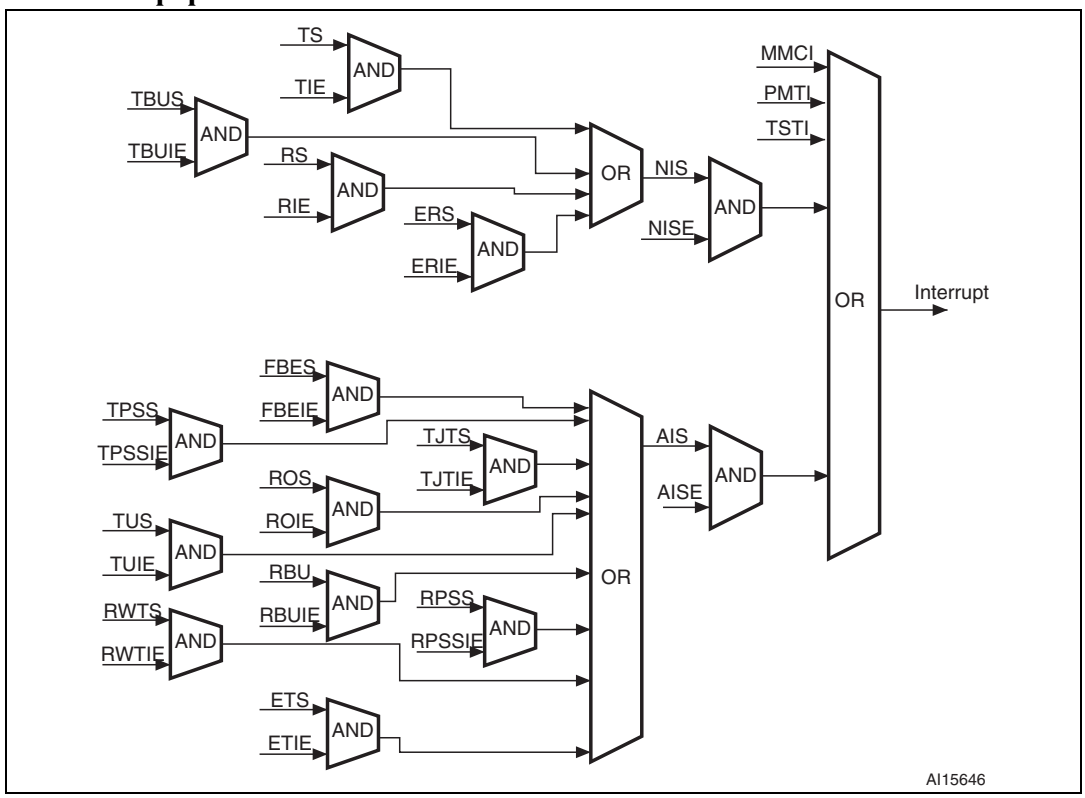
29.6.9. Прерывания DMA

Все источники прерываний отмечены своим битом регистра ETH_DMASR. Прерывания по ним разрешаются в регистре ETH_DMAIER.

Есть Нормальные и Ненормальные прерывания. Снимаются прерывания записью 1 в соответствующий бит. Общий бит прерывания снимается после очистки битов всех его источников прерываний. Ядро MAC вызывает прерывания только с битами TSTS или PMTS в ETH_DMASR.

Для всех одновременно стоящих запросов выдаётся только одно прерывание. Обработчик прерывания должен в регистре ETH_DMASR найти все причины прерывания и обслужить их. Если при выходе из разработчика сняты не все биты источников прерывания (или возникло новое), то вызывается новое прерывание.

Схема прерываний



29.7. Прерывания Ethernet

Контроллер Ethernet имеет два вектора прерываний: один для обычной работы, второй для события побудки по фрейму Побудки или Magic Packet, если оно подключено к EXTI line19.

Если разрешены и прерывание MAC PMT по событию побудки и прерывание по переднему фронту на EXTI Line19, то выдаются оба.

Если в сторожевой таймер (регистр ETH_DMARSWTR) что-то записано, то он активируется после завершения передачи фрейма в память без записи статуса (не разрешено в дескрипторе RDES1[31]). При достижении заданного значения ставится бит RS (ETH_DMASR) и, если разрешено битом RIE в регистре ETH_DMAIER, выдаётся прерывание. Если установка бита RS разрешена дескриптором, то по завершению приёма фрейма таймер отключается.

NB: Чтение регистра управления PMT автоматически снимает флаги приёма фрейма Побудки и Magic Packet. Но, поскольку регистры этих флагов лежат в домене CLK_RX, то может появиться задержка их появления (особо длинная в режиме 10 Mbit mode). Из-за этого при может возникнуть ложное прерывание даже после чтения PMT_CSR. Поэтому, обработчик должен выходить из прерывания только после снятия обоих битов приёма фрейма Побудки и Magic Packet.

29.8. Описание регистров Ethernet

Они доступны байтами (8-бит), полу-словами (16-бит) и словами (32-бита).

29.8.1. Описание регистров MAC

Регистр конфигурации Ethernet MAC (ETH_MACCR)

Смещение адреса: 0x0000

По сбросу: 0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WD	JD	Reserved		IFG			CSD	Reserved	FES	ROD	LM	DM	IPCO	RD	Reserved	APCS	BL		DC	TE	RE	Reserved	
								rw	rw			rw			rw		rw	rw	rw	rw	rw	rw		rw	rw		rw	rw	rw		

— Биты 31:24 Резерв, не трогать.

— Бит 23 **WD:** Выключение сторожевого таймера приёма

0 - MAC принимает не более 2 048 байт, остальные усекаются.

1- Таймер выключен, MAC принимает фреймы до 16 384 байт.

— Бит 22 **JD:** Выключение сторожевого таймера трёпа передачи

0 - MAC передаёт не более 2 048 байт, остальные усекаются.

1- Таймер выключен, MAC передаёт фреймы до 16 384 байт.

— Биты 21:20 Резерв, не трогать.

— Биты 19:17 **IFG:** Промежуток между фреймами при передаче

000: Время 96 бит

001: Время 88 бит

010: Время 80 бит

....

111: Время 40 бит

NB: В полудуплексе IFG может быть равным только 0b100. Меньшие значения не рассматриваются.

— Бит 16 **CSD:** Выключение датчика несущей в полудуплексе

1 - Передатчик MAC игнорирует сигнал MII CRS. Ошибки потери и отсутствия несущей не выдаются.

0 - Эти ошибки выдаются и передача прекращается.

— Бит 15 Резерв, не трогать.

— Бит 14 **FES:** Скорость Fast Ethernet (MII)

0: 10 Mbit/s

1: 100 Mbit/s

— Бит 13 **ROD:** Выключение приёма своего в полудуплексе

0 - MAC принимает все фреймы, появляющиеся на PHY при передаче.

1 - MAC выключает приём фреймов.

Этот бит в дуплексе не работает.

— Бит 12 **LM:** Режим перемычки на MII

Нужны входные такты MII (RX_CLK), поскольку такты передачи внутри не перемыкаются.

— Бит 11 **DM:** Режим дуплекса

— Бит 10 **IPCO:** Проверка контрольной суммы IPv4

1 - проверяется CRC заголовков TCP/UDP/ICMP в данных пакетов IPv4.

0 - CRC заголовков TCP/UDP/ICMP не проверяется, биты статуса PCE и IPHCE всегда чистые.

— Бит 9 **RD:** Запрет повторной передачи в полу-дуплексе

1 - Только 1 попытка передачи. При появлении коллизии на MII, MAC прекращает передачу и пишет в статус ошибку Излишка коллизий.

0 - Число попыток передачи определено в BL.

— Бит 8 Резерв, не трогать.

— Бит 7 **APCS:** Автоматическое удаление Pad/CRC

1 - MAC удаляет поля Pad/FCS из принятых фреймов короче 1 501 байта. Фреймы длиннее 1 500 байт передаются в память как есть.

0 - Все принимаемые фреймы передаются в память как есть.

— Биты 6:5 **BL:** Предел задержки ответа в полу-дуплексе

Это целое число слотов (r) задержки (время 4 096 битов для 1000 Mbit/s и 512 битов для 10/100 Mbit/s), которые MAC ждёт до начала повторной передачи после коллизии.

00: k = min (*n*, 10)

01: k = min (*n*, 8)

10: k = min (*n*, 4)

11: k = min (*n*, 1),

где *n* = попытка передачи, *r* - случайное число в диапазоне $0 \le r < 2^k$

— Бит 4 **DC:** Проверка отсрочки в полу-дуплексе

При отсрочке больше 24 288 времён бита в режиме 10/100- Mbit/s, MAC выдаёт статус фрейма со стоящим битом Излишней задержки. Задержка начинается с момента готовности передачи при активном сигнале CRS (датчик несущей) на MII. Время отсрочки не накапливается. Если передатчик задерживается на 10 000 времён бита, то он передаёт, конфликтует, и после завершения ответа сбрасывает таймер отсрочки и начинает снова. При снятом бите MAC ждёт снятия сигнала CRS.

— Бит 3 **TE:** Включение передатчика

Снимается после завершения передачи текущего фрейма на MII.

— Бит 2 **RE:** Включение приёмника

Снимается после завершения приёма текущего фрейма из MII.

— Биты 1:0 Резерв, не трогать.

Регистр фильтра фрейма Ethernet MAC (ETH_MACFFR)

Смещение адреса: 0x0004

По сбросу: 0x0000 0000

Определяет первый уровень фильтрации адресов принятых фреймов.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RA	Reserved																				HPF	SAF	SAIF	PCF		BFD	PAM	DAIF	HM	HU	PM
rw																					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **RA:** Принимать всё

1 - MAC принимает все фреймы, но результат фильтрации SA/DA отражается в слове статуса.

0 - MAC пропускает только фреймы, прошедшие фильтры SA/DA.

— Биты 30:11 Резерв, не трогать.

— Бит 10 **HPF:** Фильтрация хэш или полного адреса

1 - Если стоит бит HM или HU, то выполняется либо полная фильтрация адреса, либо хэш.

0 - Если стоит бит HM или HU, то фильтруется по хэш-таблице.

— Бит 9 **SAF:** Фильтрация адреса источника

Поле SA фрейма сравнивается с разрешёнными регистрами SA. При совпадении в слове статуса ставится бит SA Match.

1 - Отказ зависит от бита SAIF, фреймы, получившие отказ, выбрасываются.

0 - MAC передаёт принятый фрейм программе, в статусе передаётся бит SA Match.

— Бит 8 **SAIF:** Инверсия фильтра адреса источника

1 - Отказ получают фреймы, чей адрес SA есть в регистрах SA.

0 - Отказ получают фреймы, чьего адреса SA нет в регистрах SA.

— Биты 7:6 **PCF:** Пропускать управляющие фреймы, включая PAUSE

Обработка фреймов PAUSE зависит только от RFCE в регистре управления потоком[2].

00 или 01: MAC отбрасывает все управляющие фреймы

10: MAC пропускает все управляющие фреймы, независимо от фильтрации

11: MAC пропускает управляющие фреймы, прошедшие фильтр адреса.

— Бит 5 **BFD:** Запрет широковещательных фреймов

1 - Широковещательные фреймы получают отказ.

0 - Широковещательные фреймы пропускаются.

— Бит 4 **PAM:** Пропускать все многоадресные фреймы

1 - Пропускаются все многоадресные (первый бит в DA стоит) фреймы.

0 - Фильтрация зависит от бита HM.

— Бит 3 **DAIF:** Инверсная фильтрация DA одноадресных и многоадресных фреймов

0 - нормальная фильтрация DA адресов.

1 - Инверсная.

— Бит 2 **HM:** Многоадресная хэш-фильтрация

1 - Фильтрация многоадресных фреймов по хэш-таблице.

0 - Фильтрация многоадресных фреймов по полному адресу DA

— Бит 1 **HU:** Одноадресная хэш-фильтрация

1 - Фильтрация одноадресных фреймов по хэш-таблице.

0 - Фильтрация одноадресных фреймов по полному адресу DA

— Бит 0 **PM:** Беспорядочный режим

Если стоит, то пропускаются все фреймы, независимо от фильтрации, биты статуса SA/DA нулевые.

Старший регистр хэш-таблицы Ethernet MAC (ETH_MACHTHR)

Смещение адреса: 0x0008

По сбросу: 0x0000 0000

Для фильтрации групповых адресов используется 64-хэш-таблица в двух регистрах. Индексом к ней служат 6 старших битов CRC полученного фрейма. Старший бит это регистр, а остальные 5 это номер бита в регистре. Хэш 0b0 0000 определяет бит 0 регистра 0, а хэш 0b1 1111 бит 31 регистра 1.

Например, DA фрейма равен 0x1F52 419C B6AF, значение хэш равно 0x2C. Тогда проверяется бит [12] регистра `HTH`, при хэш, равном 0x07 проверяется бит [7] регистра `HTL`.

Фрейм принимается если проверяемый бит равен 1, иначе отвергается.

При стоящем бите PAM в регистре ETH_MACFFR пропускаются все фреймы с групповыми адресами, независимо от результата фильтрации.

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

— Биты 31:0 **HTH:** Старшие 32 бита хэш-таблицы.

Младший регистр хэш-таблицы Ethernet MAC (ETH_MACHTLR)

Смещение адреса: 0x000C

По сбросу: 0x0000 0000

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

— Биты 31:0 **HTL:** 32 бита хэш-таблицы.

Регистр адреса MII Ethernet MAC (ETH_MACMIIAR)

Смещение адреса: 0x0010

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PA					MR					Reserved	CR			MW	MB
																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rc_w1

— Биты 31:16 Резерв, не трогать.

— Биты 15:11 **PA:** Адрес устройства PHY из 32 возможных.

— Биты 10:6 **MR:** Регистр MII устройства PHY.

— Бит 5 Резерв, не трогать.

— Биты 4:2 **CR:** Диапазон частот HCLK для получения тактов MDC

000 60-72 MHz HCLK/42

001 Резерв -

010 20-35 MHz HCLK/16

011 35-60 MHz HCLK/26

100, 101, 110, 111 Резерв -

— Бит 1 **MW:** Запись MII

1 - Операция записи PHY через регистр данных MII.

0 - Операция чтения PHY в регистр данных MII.

— Бит 0 **MB:** MII занят (или Запуск операции)

Писать в регистры ETH_MACMIIAR и ETH_MACMIIDR можно только при снятом бите. Установка бита запускает работу PHY. Снимает аппаратно.

Регистр данных MII Ethernet MAC (ETH_MACMIIDR)

Смещение адреса: 0x0014

По сбросу: 0x0000 0000

Промежуточный регистр между программой и регистром PHY, заданным в ETH_MACMIIAR.

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

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **MD:** 16-бит данные MII из PHY после операции чтения и для операции записи PHY.

Регистр управления потоком Ethernet MAC (ETH_MACFCR)

Смещение адреса: 0x0018

По сбросу: 0x0000 0000

Управляет приёмом и выдачей фреймов Паузы. Запись со стоящим битом FCB выдаёт фрейм паузы. Он остаётся стоять всё время передачи по кабелю, писать в регистр можно только при снятом бите.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PT																Reserved								ZQPD	Reserved	PLT		UPFD	RFCE	TFCE	FCB/ BPA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw									rw		rw	rw	rw	rw	rw	rc_w1 /rw

— Биты 31:16 **PT:** Значение времени паузы в передаваемом фрейме

Если время сконфигурировано с двойной синхронизацией в домене тактов MII, то писать в этот регистр можно только после не менее 4 тактов домена назначения.

— Биты 15:8 Резерв, не трогать.

— Бит 7 **ZQPD:** Запрет выдачи фреймов Zero-quanta паузы

— Бит 6 Резерв, не трогать.

— Биты 5:4 **PLT:** Нижний порог паузы

Это время таймера паузы, после которого автоматически передаётся новый фрейм паузы. Должен быть меньше времени паузы в битах[31:16]. Например, если PT = 100H (256 времён слота), и PLT = 01, то второй фрейм PAUSE будет выдан, если можно, через 228 (256 – 28) времён слота после первого.

00 - Время паузы минус 4 времени слота

01 - Время паузы минус 28 времён слота

10 - Время паузы минус 144 времени слота

11 - Время паузы минус 256 времён слота

Время слота это время передачи 512 битов (64 байта) по MII.

— Бит 3 **UPFD:** Обнаружение одноадресного фрейма паузы

1 - Кроме фреймов паузы с уникальным групповым адресом обнаруживаются фреймы паузы с адресом, лежащим в регистрах ETH_MACA0HR и ETH_MACA0LR.

0 - Только фреймы с уникальным групповым адресом.

— Бит 2 **RFCE:** Разрешение приёма фреймов Паузы с прекращением передач

— Бит 1 **TFCE:** Разрешение передачи фреймов Паузы

В полном дуплексе разрешает выдачу фреймов паузы.

В полу-дуплексе разрешает операции обратного давления.

— Бит 0 **FCB/BPA**: Управление потоком занято (Запуск)/Включение обратного давления

В полном дуплексеThis bit initiates a Pause Control frame in Full-duplex mode and activates the back pressure function in Half-duplex mode if TFCE bit is set.

В полном дуплексе запускает выдачу фрейма паузы. Стоит всё время передачи фрейма паузы, снимается аппаратно. Писать в регистр можно только при снятом бите.

В полу-дуплексе, если одновременно стоит бит TFCE, то ядро MAC выставляет обратное давление. При этом по принятию ядром MAC нового фрейма передатчик начинает передавать JAM-паттерн, вызывая коллизию. В полном дуплексе BPA автоматически отключается.

Регистр тега VLAN Ethernet MAC (ETH_MACVLANTR)

Смещение адреса: 0x001C

По сбросу: 0x0000 0000

Фреймы VLAN имеют в поле Типа/Длины число 0x8100, и тег VLAN в следующих двух байтах. Появление такого фрейма ставит бит VLAN в слове статуса. Допустимая длина фрейма увеличивается с 1518 байт до 1522 байт.

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

— Биты 31:17 Резерв, не трогать.

— Бит 16 **VLANTC:**12-бит сравнение тега VLAN

1 - Для сравнения и фильтрации используются биты [11:0] идентификатора VLAN.

0 - Сравниваются 16 бит идентификатора VLAN принятого фрейма.

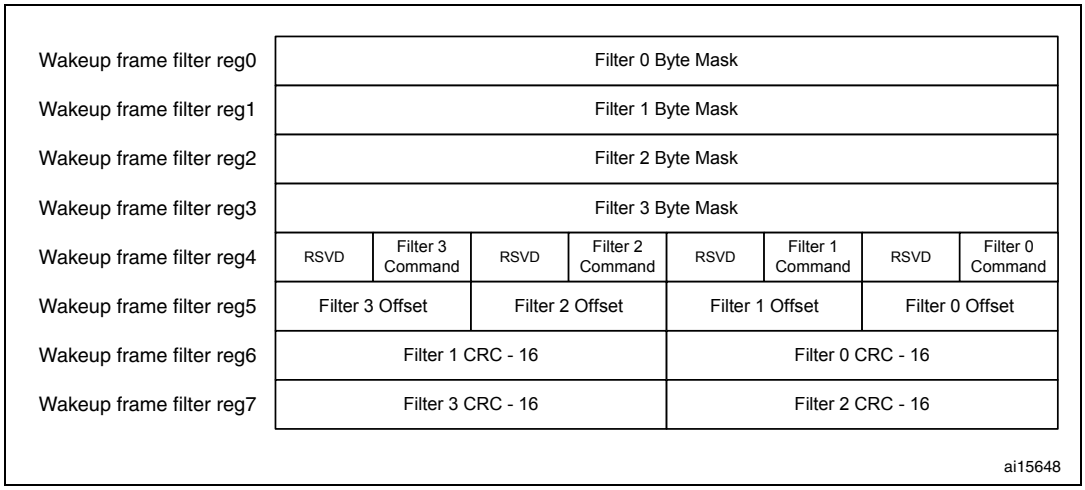
— Биты 15:0 **VLANTI:** Идентификатор тега VLAN фреймов приёма
Биты[15:13] это приоритет пользователя, бит[12] это индикатор канонического формата (CFI), биты[11:0] это поле идентификатора VLAN (VID). Если идентификатор нулевой, то он не проверяется и фреймами VLAN объявляются все фреймы с полем типа, равным 0x8100.

Регистр фильтра фреймов удалённой побудки Ethernet MAC (ETH_MACRWUFFR)

Смещение адреса: 0x0028

По сбросу: 0x0000 0000

Это адрес доступа к восьми регистрам фильтра фреймов удалённой побудки по восьми последовательным операциям чтения или по восьми последовательным операциям записи. Этот регистр содержит старшие 16 бит 7-го адреса MAC.



Регистр управления и состояния PMT Ethernet MAC (ETH_MACPMTCSR)

Смещение адреса: 0x002C

По сбросу: 0x0000 0000

Определяет и управляет событиями побудки.

31	30:10	9	8:7	6	5	4:3	2	1	0
WFFRPR	Reserved	GU	Reserved	WFR	MPR	Reserved	WFE	MPE	PD
rs	Res.	rw		rc_r	rc_r		rw	rw	rs

— Бит 31 **WFFRPR:** Сброс указателя регистров фильтра фреймов побудки
Обнуляет указатель, снимается сам через 1 такт.
— Биты 30:10 Резерв, не трогать.
— Бит 9 **GU:** Разрешение опознания фрейма побудки в любом одноадресном пакете.
— Биты 8:7 Резерв, не трогать.
— Бит 6 **WFR:** Принят фрейм побудки
Ставится аппаратно, снимается чтением этого регистра.
— Бит 5 **MPR:** Принят Magic packet
Ставится аппаратно, снимается чтением этого регистра.
— Биты 4:3 Резерв, не трогать.
— Бит 2 **WFE:** Разрешение выдачи события PMT по фрейму побудки
— Бит 1 **MPE:** Разрешение выдачи события PMT по Magic Packet
— Бит 0 **PD:** Выключение питания
Все принимаемые фреймы теряются. Снимается автоматически по приёму Magic Packet или фрейма побудки. Ставить этот бит нужно только при стоящих битах WFE или MPE.

Регистр состояния прерываний Ethernet MAC (ETH_MACSR)

Смещение адреса: 0x0038

По сбросу: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TSTS	Reserved		MMCTS	MMCRS	MMCS	PMTS	Reserved		
						rc_r			r	r	r	r			

— Биты 15:10 Резерв, не трогать.

— Бит 9 **TSTS:** Статус запуска по времени

Ставится когда системное время достигает значения, заложенного в регистрах Целевого времени. Снимается чтением регистра ETH_PTPTSSR.

— Биты 8:7 Резерв, не трогать.

— Бит 6 **MMCTS:** Статус передачи MMC

Ставится по появлению прерываний в регистре ETH_MMCTIR. Снимается после очистки всех прерываний в регистре ETH_MMCTIR.

— Бит 5 **MMCRS:** Статус приёма MMC

Ставится по появлению прерываний в регистре ETH_MMCRIR. Снимается после очистки всех прерываний в регистре ETH_MMCRIR.

— Бит 4 **MMCS:** Статус MMC

Ставится при любом стоящем бите 6:5. Снимается при очистке обоих.

— Бит 3 **PMTS:** Статус PMT

Ставится по приёму Magic packet или фрейма Побудки в режиме выключения питания (см. биты 5 и 6 в регистре ETH_MACPMTCSR). Снимается при очистке обоих битов чтением регистра ETH_MACPMTCSR.

— Биты 2:0 Резерв, не трогать.

Регистр маски прерываний Ethernet MAC (ETH_MACIMR)

Смещение адреса: 0x003C

По сбросу: 0x0000 0000

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

— Биты 15:10 Резерв, не трогать.

— Бит 9 **TSTIM:** Запрет прерывания запуска по времени.

— Биты 8:4 Резерв, не трогать.

— Бит 3 **PMTIM:** Запрет прерывания PMT по биту PMTS в регистре ETH_MACSR.

— Биты 2:0 Резерв, не трогать.

Старший регистр адреса 0 Ethernet MAC (ETH_MACA0HR)

Смещение адреса: 0x0040

По сбросу: 0x8000 FFFF

NB: Байты DA передаются по MII начиная с младшего.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MO	Reserved															MACA0H															
1																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **MO:** Всегда 1.

— Биты 30:16 Резерв, не трогать.

— Биты 15:0 **MACA0H:** Биты[47:32] MAC адреса 0 станции

Младший регистр адреса 0 Ethernet MAC (ETH_MACA0LR)

Смещение адреса: 0x0044

По сбросу: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACA0L																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **MACA0L:** Биты[31:0] MAC адреса 0 станции.

Старший регистр адреса 1 Ethernet MAC (ETH_MACA1HR)

Смещение адреса: 0x0044

По сбросу: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AE	SA	MBC						Reserved								MACA1H															
rw	rw	rw	rw	rw	rw	rw	rw									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **AE:** Разрешение участия MAC адреса 1 в полной фильтрации адреса.

— Бит 30 **SA:** Это адрес источника

1 - MAC адрес 1 [47:0] это адрес источника.

0 - MAC адрес 1 [47:0] это адрес получателя.

— Биты 29:24 **MBC:** Маска байтов сравнения

Если бит стоит, то соответствующий байт принятых DA/SA в сравнении с MAC адресом 1 из регистров не участвует. Соответствие битов байтам:

– Бит 29: ETH_MACA1HR [15:8]

– Бит 28: ETH_MACA1HR [7:0]

– Бит 27: ETH_MACA1LR [31:24]

...

– Бит 24: ETH_MACA1LR [7:0]

— Биты 23:16 Резерв, не трогать.

— Биты 15:0 **MACA1H:** Биты[47:32] MAC адреса 1 станции.

Младший регистр адреса 1 Ethernet MAC (ETH_MACA1LR)

Смещение адреса: 0x004C

По сбросу: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACA1L																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **MACA1L:** Биты[31:0] MAC адреса 1 станции (надо писать при инициализации).

Старший регистр адреса 2 Ethernet MAC (ETH_MACA2HR)

Смещение адреса: 0x0050

По сбросу: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AE	SA	MBC						Reserved								MACA2H															
rw	rw	rw	rw	rw	rw	rw	rw									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **AE:** Разрешение участия MAC адреса 2 в полной фильтрации адреса.

— Бит 30 **SA:** Это адрес источника

1 - MAC адрес 2 [47:0] это адрес источника.

0 - MAC адрес 2 [47:0] это адрес получателя.

— Биты 29:24 **MBC:** Маска байтов сравнения

Если бит стоит, то соответствующий байт принятых DA/SA в сравнении с MAC адресом 2 из регистров не участвует. Соответствие битов байтам:

– Бит 29: ETH_MACA2HR [15:8]

– Бит 28: ETH_MACA2HR [7:0]

– Бит 27: ETH_MACA2LR [31:24]

...

– Бит 24: ETH_MACA2LR [7:0]

— Биты 23:16 Резерв, не трогать.

— Биты 15:0 **MACA2H:** Биты[47:32] MAC адреса 2 станции.

Младший регистр адреса 2 Ethernet MAC (ETH_MACA2LR)

Смещение адреса: 0x0054

По сбросу: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACA2L																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **MACA2L:** Биты[31:0] MAC адреса 2 станции (надо писать при инициализации).

Старший регистр адреса 3 Ethernet MAC (ETH_MACA3HR)

Смещение адреса: 0x0058

По сбросу: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AE	SA	MBC						Reserved								MACA3H															
rw	rw	rw	rw	rw	rw	rw	rw									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **AE:** Разрешение участия MAC адреса 3 в полной фильтрации адреса.

— Бит 30 **SA:** Это адрес источника

1 - MAC адрес 3 [47:0] это адрес источника.

0 - MAC адрес 3 [47:0] это адрес получателя.

— Биты 29:24 **MBC:** Маска байтов сравнения

Если бит стоит, то соответствующий байт принятых DA/SA в сравнении с MAC адресом 2 из регистров не участвует. Соответствие битов байтам:

– Бит 29: ETH_MACA3HR [15:8]

– Бит 28: ETH_MACA3HR [7:0]

– Бит 27: ETH_MACA3LR [31:24]

...

– Бит 24: ETH_MACA3LR [7:0]

— Биты 23:16 Резерв, не трогать.

— Биты 15:0 **MACA3H:** Биты[47:32] MAC адреса 3 станции.

Младший регистр адреса 3 Ethernet MAC (ETH_MACA3LR)

Смещение адреса: 0x005C

По сбросу: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACA3L																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **MACA3L:** Биты[31:0] MAC адреса 3 станции (надо писать при инициализации).

29.8.2. Описание регистров MMC

Регистр управления Ethernet MMC (ETH_MMCCR)

Смещение адреса: 0x0100

По сбросу: 0x0000 0000

Определяет режим работы счётчиков MMC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												MCF	ROR	CSR	CR
																												rw	rw	rw	rw

— Биты 31:4 Резерв, не трогать.

— Бит 3 **MCF:** Заморозка счётчиков MMC в текущем положении

Если какой-либо счётчик читается при стоящем бите ROR, то он сбрасывается и в этом режиме.

— Бит 2 **ROR:** Сброс счётчика после прочтения младшего байта

— Бит 1 **CSR:** Запрет возврата счётчиков к нулю при достижении максимального значения

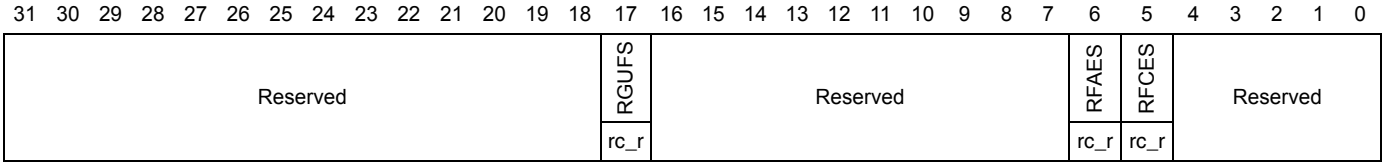
— Бит 0 **CR:** Сброс всех счётчиков, снимается через 1 такт.

Регистр прерываний счётчиков приёма Ethernet MMC (ETH_MMCRIR)

Смещение адреса: 0x0104

По сбросу: 0x0000 0000

Запрос прерывания от счётчика ставится при достижении половины максимального значения (ставится старший бит). Снимается после чтения соответствующего счётчика.



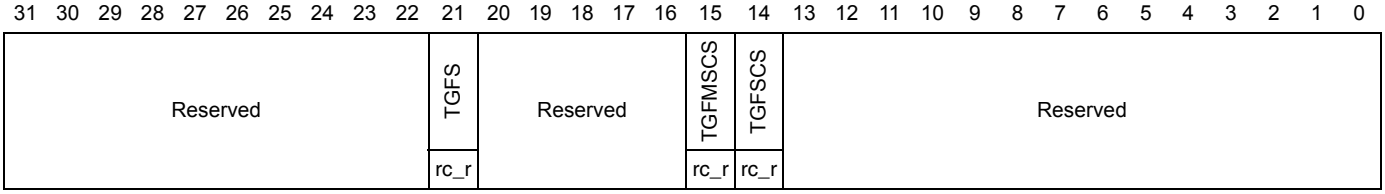
- Биты 31:18 — Резерв, не трогать.
- Бит 17 — **RGUFS:** Приём хороших одноадресных фреймов.
- Биты 16:7 — Резерв, не трогать.
- Бит 6 — **RFAES:** Приём фреймов с ошибкой выравнивания.
- Бит 5 — **RFCES:** Приём фреймов с ошибкой CRC.
- Биты 4:0 — Резерв, не трогать.

Регистр прерываний счётчиков передачи Ethernet MMC (ETH_MMCTIR)

Смещение адреса: 0x0108

По сбросу: 0x0000 0000

Запрос прерывания от счётчика ставится при достижении половины максимального значения (ставится старший бит). Снимается после чтения соответствующего счётчика.



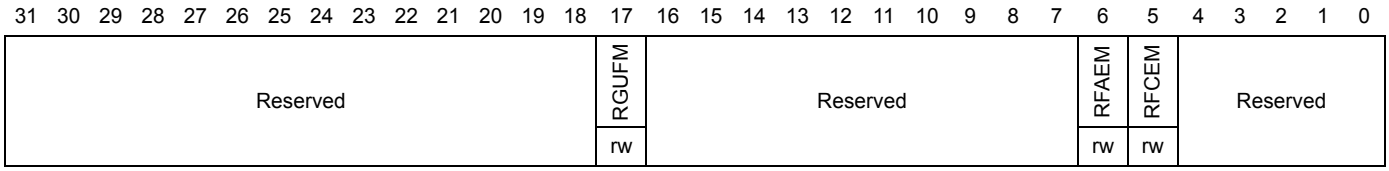
- Биты 31:22 — Резерв, не трогать.
- Бит 21 — **TGFS:** Передача хороших фреймов.
- Биты 20:16 — Резерв, не трогать.
- Бит 15 — **TGFMSCS:** Передача хороших фреймов с более чем одной коллизией.
- Бит 14 — **TGFSCS:** Передача хороших фреймов с одной коллизией.
- Биты 13:0 — Резерв, не трогать.

Регистр маски прерываний счётчиков приёма Ethernet MMC (ETH_MMCRIMR)

Смещение адреса: 0x010C

По сбросу: 0x0000 0000

Стоящий бит маски запрещает прерывание.



- Биты 31:18 — Резерв, не трогать.
- Бит 17 — **RGUFM:** Приём хороших одноадресных фреймов.
- Биты 16:7 — Резерв, не трогать.
- Бит 6 — **RFAEM:** Приём фреймов с ошибкой выравнивания.
- Бит 5 — **RFCEM:** Приём фреймов с ошибкой CRC.
- Биты 4:0 — Резерв, не трогать.

Регистр маски прерываний счётчиков передачи Ethernet MMC (ETH_MMCTIMR)

Смещение адреса: 0x0110

По сбросу: 0x0000 0000

Стоящий бит запрещает соответствующее прерывание.

31:22	21	20:16	15	14	13:0
Reserved	TGFM	Reserved	TGFMSCM	TGFSCM	Reserved
	rw		rw	rw	

— Биты 31:22 Резерв, не трогать.
— Бит 21 **TGFM:** Передача хороших фреймов.
— Биты 20:16 Резерв, не трогать.
— Бит 15 **TGFMSCM:** Передача хороших фреймов с более чем одной коллизией.
— Бит 14 **TGFSCM:** Передача хороших фреймов с одной коллизией.
— Биты 13:0 Резерв, не трогать.

Регистр числа успешно переданных фреймов после одной коллизии Ethernet MMC (ETH_MMCTGFSCCR)

Смещение адреса: 0x014C

По сбросу: 0x0000 0000

Значение **TGFSCC** действительно в полу-дуплексе.

31:0
TGFSCC
r

Регистр числа успешно переданных фреймов после нескольких коллизий Ethernet MMC (ETH_MMCTGFMSCCR)

Смещение адреса: 0x0150

По сбросу: 0x0000 0000

Значение **TGFMSCC** действительно в полу-дуплексе.

31:0
TGFMSCC
r

Регистр числа успешно переданных фреймов Ethernet MMC (ETH_MMCTGFCR)

Смещение адреса: 0x0168

По сбросу: 0x0000 0000

This register contains the number of good frames transmitted.

31:0
TGFC
r

— Биты 31:0 **TGFC:** Transmitted good frames counter

Регистр числа принятых фреймов с ошибкой CRC Ethernet MMC (ETH_MMCRFCECR)

Смещение адреса: 0x0194

По сбросу: 0x0000 0000

RFCEC это число фреймов, принятых с ошибкой CRC.

31:0
RFCEC
r

Регистр числа принятых фреймов с ошибкой выравнивания Ethernet MMC (ETH_MMCRFAECR)

Смещение адреса: 0x0198

По сбросу: 0x0000 0000

RFAEC это число фреймов, принятых с ошибкой выравнивания (дробный полубайт).

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

Регистр числа принятых одноадресных фреймов Ethernet MMC (ETH_MMCRGUFCR)

Смещение адреса: 0x01C4

По сбросу: 0x0000 0000

RGUFC это число хорошо принятых одноадресных фреймов.

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

29.8.3. Регистры меток времени IEEE 1588

Регистр управления меток времени Ethernet PTP (ETH_PTPTSCR)

Смещение адреса: 0x0700

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										TTSARU	TSITE	TSSTU	TSSTI	TSFCU	TSE
																										rw	rw	rw	rw	rw	rw

— Биты 31:6 Резерв, не трогать.

— Бит 5 **TSARU:** Обновление регистра слагаемого при тонкой коррекции

Ставить можно только обнулённый бит.

— Бит 4 **TSITE:** Разрешение прерывания запуска по времени, после прерывания снимается.

— Бит 3 **TSSTU:** Обновление системного времени

Системное время изменяется на значение регистров обновления метки времени. Биты TSSTU и TSSTI должны быть чистыми. По завершению обновления бит снимается.

— Бит 2 **TSSTI:** Инициализация системного времени

В системное время пишется на значение регистров обновления метки времени. Ставить можно только обнулённый бит. После записи бит снимается.

— Бит 1 **TSFCU:** Грубая или тонкая коррекция

1 - Нужна Тонкая коррекция.

0 - Грубая коррекция.

— Бит 0 **TSE:** Разрешение передачи и приёма меток времени

После установки этого бита надо всегда инициализировать системное время.

Регистр суб-секундного инкремента Ethernet PTP (ETH_PTPSSIR)

Смещение адреса: 0x0704

По сбросу: 0x0000 0000

При Грубой коррекции (бит TSFCU в ETH_PTPTSCR) это значение добавляется в системное время по каждому такту HCLK. При Тонкой коррекции добавляется в системное время по переполнению Накопителя.

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

— Биты 31:8 Резерв, не трогать.

— Биты 7:0 **STSSI:** Суб-секундный инкремент системного времени

Старший регистр метки времени Ethernet PTP (ETH_PTPTSHR)

Смещение адреса: 0x0708

По сбросу: 0x0000 0000

STS это текущее значение секунд системного времени, обновляется само и постоянно.

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

Младший регистр метки времени Ethernet PTP (ETH_PTPTSLR)

Смещение адреса: 0x070C

По сбросу: 0x0000 0000

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

— Бит 31 **STPNS:** Знак системного времени, плюс или минус

Обычно всегда сброшен (плюс).

— Биты 30:0 **STSS:** Суб-секунды системного времени с точностью 0.46 ns.

Старший регистр обновления метки времени Ethernet PTP (ETH_PTPTSHUR)

Смещение адреса: 0x0710

По сбросу: 0x0000 0000

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

— Биты 31:0 **TSUS:** Секунды обновления системного времени

Используется для записи, сложения или вычитания из системного времени. Писать надо до установки битов `TSSTI` или `TSSTU` в регистре управления.

Младший регистр обновления метки времени Ethernet PTP (ETH_PTPTSLUR)

Смещение адреса: 0x0714

По сбросу: 0x0000 0000

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

— Бит 31 **TSUPNS:** Знак обновления, плюс (0) или минус (1)

При стоящем бите TSSTI (инициализация системного времени) должен быть нулевым. Если он стоит при стоящем бите TSSTU, то значение обновления вычитается из системного времени, иначе, добавляется.

— Биты 30:0 **TSUSS:** Суб-секунды обновления системного времени с точностью 0.46 ns.

Писать надо до установки битов `TSSTI` или `TSSTU` в регистре управления.

Регистр слагаемого метки времени Ethernet PTP (ETH_PTPTSAR)

Смещение адреса: 0x0718

По сбросу: 0x0000 0000

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

— Биты 31:0 **TSA:** Слагаемое времени

Используется только при Тонкой коррекции (бит TSFCU в ETH_PTPTSCR). По каждому такту добавляется в регистр накопителя, системное время обновляется при переполнении Накопителя.

Старший регистр целевого времени Ethernet PTP (ETH_PTPTTHR)

Смещение адреса: 0x071C

По сбросу: 0x0000 0000

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

— Биты 31:0 **TTSH:** Секунды целевого времени

При достижении системным временем значения целевого выдаётся прерывание (бит TSARU в ETH_PTPTSCR).

Младший регистр целевого времени Ethernet PTP (ETH_PTPTTLR)

Смещение адреса: 0x0720

По сбросу: 0x0000 0000

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

— Биты 31:0 **TTSL:** Наносекунды целевого времени (знаковое)

При достижении системным временем значения целевого выдаётся прерывание (бит TSARU в ETH_PTPTSCR).

29.8.4. Описание регистров DMA

Регистр режима шины Ethernet DMA (ETH_DMABMR)

Смещение адреса: 0x1000

По сбросу: 0x0002 0101

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						AAB	FPM	USP	RDP						FB	PM		PBL						Reserved	DSL					DA	SR
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rs

— Биты 31:26 Резерв, не трогать.

— Бит 25 **AAB:** Выравнивание адреса частей

Если этот бит стоит вместе с битом FB, то AHB выдаёт все пакеты, выравненными по младшим битам адреса начала. Если бит FB равен 0, то первый пакет не выравнивается, а последующие таки да..

— Бит 24 **FPM:** Режим 4xPBL

Если бит стоит, то значение PBL (биты [22:17] и [13:8]) умножается на 4. И DMA передаёт данные за максимум 4, 8, 16, 32, 64 или 128 частей, в зависимости от значения PBL.

— Бит 23 **USP:** Использовать отдельный PBL

1 - RxDMA как PBL использует биты[22:17], а TxDMA биты[13:8].

0 - RxDMA и TxDMA для PBL используют биты[13:8].

— Биты 22:17 **RDP:** RxDMA PBL

Максимальное число частей одной передачи RxDMA. Допустимое значение RDP равно 1, 2, 4, 8, 16, или 32. Иное приводит к неадекватному поведению.

Работает только при стоящем бите USP.

— Бит 16 **FB:** Фиксированный пакет

1 - В начале нормальных передач AHB использует только SINGLE, INCR4, INCR8 или INCR16 операции.

0 - SINGLE и INCR.

— Биты 15:14 PM: Отношение приоритетов Rx:Tx при чистом бите DA

00: 1:1

01: 2:1

10: 3:1

11: 4:1

— Биты 13:8 **PBL:** Программируемая длина пакета

Максимальное число частей одной передачи DMA.

Допустимое значение PBL равно 1, 2, 4, 8, 16, или 32. Иное приводит к неадекватному поведению. При стоящем бите USP используется только TxDMA.

Ограничения PBL такие:

- Максимально допустимое значение частей (PBL) ограничено размером Tx FIFO и Rx FIFO.
- FIFO вынужденно поддерживает максимальное число частей, равное половине глубины FIFO.
- Если PBL общий для приёма и передачи, то надо учитывать меньшую глубину RxFIFO и TxFIFO.
- Недопустимое значение PBL сводит систему с ума.

— Бит 7 Резерв, не трогать.

— Биты 6:2 **DSL:** Длина пропуска дескриптора

Это число пропускаемых слов между двумя несцеплёнными дескрипторами от конца первого до начала второго. Если DSL равен нулю, то это неразрывная таблица дескрипторов в режиме кольца.

— Бит 1 **DA:** Арбитраж DMA

0: Круговой с отношением приоритетов Rx:Tx в битах [15:14]

1: Rx приоритетнее Tx

— Бит 0 **SR:** Программный сброс всех подсистем MAC DMA

Автоматически снимается после завершения сброса всех доменов тактирования. В регистры ядра можно писать только при нулевом бите.

Регистр требования опроса списка дескрипторов передачи Ethernet DMA (ETH_DMATPDR)

Смещение адреса: 0x1004

По сбросу: 0x0000 0000

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

— Биты 31:0 **TPD**: Требование опроса передачи

После записи любого числа DMA проверяет текущий дескриптор (адрес в регистре ETH_DMACHTDR) на доступность. Если он принадлежит CPU, то передача возвращается в Останов и ставится бит 2 регистра ETH_DMASR, иначе передача возобновляется.

Регистр требования опроса списка дескрипторов приёма Ethernet DMA (ETH_DMARPDR)

Смещение адреса: 0x1008

По сбросу: 0x0000 0000

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

— Биты 31:0 **RPD:** Требование опроса приёма

После записи любого числа DMA проверяет текущий дескриптор (адрес в регистре ETH_DMACHRDR) на доступность. Если он принадлежит CPU, то передача возвращается в Останов и ставится бит 7 регистра ETH_DMASR, иначе передача возобновляется.

Регистр базового адреса списка дескрипторов приёма Ethernet DMA (ETH_DMARDLAR)

Смещение адреса: 0x100C

По сбросу: 0x0000 0000

Список дескрипторов лежит в памяти и должен быть выравнен на границу слова. DMA сам выравнивает адрес на границу ширины шины, обнуляя младшие биты (биты[1/2/3:0] для 32/64/128-бит ширины). Писать в регистр можно только при остановленном приёме.

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

— Биты 31:0 **SRL:** Начало списка приёма

Регистр базового адреса списка дескрипторов передачи Ethernet DMA (ETH_DMATDLAR)

Смещение адреса: 0x1010

По сбросу: 0x0000 0000

Список дескрипторов лежит в памяти и должен быть выравнен на границу слова. DMA сам выравнивает адрес на границу ширины шины, обнуляя младшие биты (биты[1/2/3:0] для 32/64/128-бит ширины). Писать в регистр можно только при остановленном приёме.

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

— Биты 31:0 **STL:** Начало списка передачи

Регистр состояния Ethernet DMA (ETH_DMASR)

Смещение адреса: 0x1014

По сбросу: 0x0000 0000

Нерезервированные биты из ETH_DMASR[16:0] снимаются записью 1 в них. Прерывания по ним маскируются в регистре ETH_DMAIER.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		TSTS	PMTS	MMCS	Reserved	EBS			TPS			RPS			NIS	AIS	ERS	FBES	Reserved		ETS	RWTS	RPSS	RBUS	RS	TUS	ROS	TJTS	TBUS	TPSS	TS
		r	r	r		r	r	r	r	r	r	r	r	r	rc-w1	rc-w1	rc-w1	rc-w1			rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1

— Биты 31:30 Резерв, не трогать.

— Бит 29 **TSTS:** Статус прерывания запуска по времени

Снимается чтением регистра состояния ядра MAC, очищая источник (бит 9).

— Бит 28 **PMTS:** Статус прерывания PMT

Снимается чтением соответствующих регистров ядра MAC, очищая источники.

— Бит 27 **MMCS:** Статус прерывания MMC

Снимается чтением соответствующих регистров ядра MAC, очищая источники.

— Бит 26 Резерв, не трогать.

— Биты 25:23 **EBS:** Биты причины фатальной ошибки шины (ETH_DMASR[13])

Бит 23: 1 - Ошибка передачи TxDMA, 0 - Ошибка передачи RxDMA

Бит 24: 1 - Ошибка при чтении, 0 - Ошибка при записи

Бит 25: 1 - Ошибка при доступе к дескриптору, 0 - Ошибка при доступе к буферу данных

— Биты 22:20 **TPS:** Состояние процесса передачи (TxDMA FSM)

000: Стоит; Была команда Сброса или Останова передачи

001: Работа; Выборка дескриптора передачи

010: Работа; Ждёт статус

011: Работа; Пишет данные из буфера памяти в TxFIFO

100, 101: Резерв

110: Приостановка; Недоступен дескриптор передачи или Исчерпание буфера

111: Работа; Закрывает дескриптор передачи

— Биты 19:17 **RPS:** Состояние процесса приёма (RxDMA FSM)

000: Стоит: Была команда Сброса или Останова приёма

001: Работа: Выборка дескриптора приёма

010: Резерв

011: Работа: Ждёт приёмного пакет

100: Приостановка: Недоступен дескриптор приёма

101: Работа: Закрывает дескриптор приёма

110: Резерв

111: Работа: Передаёт данные из RxFIFO в память

— Бит 16 **NIS:** Общее нормальное прерывание

Это логическое OR запросов прерываний, разрешённых в регистре ETH_DMAIER:

- ETH_DMASR [0]: прерывание Передачи
- ETH_DMASR [2]: Буфер передачи недоступен
- ETH_DMASR [6]: прерывание Приёма
- ETH_DMASR [14]: прерывание Раннего приёма

Бит снимается записью 1 после снятия бита причины прерывания.

— Бит 15 **AIS:** Общее ненормальное прерывание

Это логическое OR запросов прерываний, разрешённых в регистре ETH_DMAIER:

- ETH_DMASR [1]: Передача остановлена
- ETH_DMASR [3]: Таймаут трёпа
- ETH_DMASR [4]: Переполнение RxFIFO
- ETH_DMASR [5]: Исчерпание передачи
- ETH_DMASR [7]: Буфер приёма недоступен
- ETH_DMASR [8]: Приём остановлен
- ETH_DMASR [9]: Таймаут сторожевого таймера приёма
- ETH_DMASR [10]: прерывание Ранней передачи
- ETH_DMASR [13]: Фатальная ошибка шины

Бит снимается записью 1 после снятия бита причины прерывания.

— Бит 14 **ERS:** Ранний приём

DMA заполнил первый буфер данных пакета. Прерывание приёма ETH_DMASR[6] автоматически снимает этот бит.

— Бит 13 **FBES:** Фатальная ошибка шины

Причина ошибки указана в битах [25:23]. Соответствующая машина DMA прекращает доступ к шине.

— Биты 12:11 Резерв, не трогать.

— Бит 10 **ETS:** Ранняя передача

Передаваемый фрейм полностью записан в TxFIFO.

— Бит 9 **RWTS:** Таймаут сторожевого таймера приёма

Длина принятого фрейма больше 2 048 байт.

— Бит 8 **RPSS:** Приём остановлен

— Бит 7 **RBUS:** Буфер приёма недоступен

Следующий дескриптор принадлежит CPU, а предыдущий принадлежал DMA. Приём приостановлен. Он восстановится после того как CPU изменит принадлежность дескриптора выдаст и требование опроса приёма или будет принят следующий распознанный фрейм.

— Бит 6 **RS:** Фрейм принят

Статус приёма уже записан в дескриптор. Машина остаётся в Работе.

— Бит 5 **TUS:** Исчерпание передачи

Передача приостановлена, ставится бит исчерпания TDES0[1].

— Бит 4 **ROS:** Переполнение буфера приёма

Если программе передана часть фрейма, то ставится бит переполнения RDES0[11].

— Бит 3 **TJTS:** Таймаут трёпа

Допустимое время активной передачи исчерпано, машина Остановлена, ставится флаг TDES0[14].

— Бит 2 **TBUS:** Буфер передачи недоступен

Следующий дескриптор принадлежит CPU. Передача приостановлена. Причина лежит в битах[22:20]. Передача восстановится после того как CPU изменит принадлежность дескриптора выдаст и требование опроса передачи.

— Бит 1 **TPSS:** Передача остеновлена.

— Бит 0 **TS:** Конец передачи фрейма, ставится TDES1[31].

Регистр режима работы Ethernet DMA (ETH_DMAOMR)

Смещение адреса: 0x1018

По сбросу: 0x0000 0000

При инициализации DMA этот регистр пишут последним

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					DTCEFD	RSF	DFRF	Reserved		TSF	FTF	Reserved			TTC			ST	Reserved					FEF	FUGF	Reserved	RTC		OSF	SR	Reserved
					rw	rw	rw			rw	rs				rw	rw	rw	rw						rw	rw		rw	rw	rw	rw	

— Биты 31:27 Резерв, не трогать.

— Бит 26 **DTCEFD:** Запрет выброса фреймов с ошибкой контрольной суммы только у вложенных пакетов TCP/IP.

— Бит 25 **RSF:** Режим Накопления (Store-and-forward) при приёме

1 - Фреймы читаются из RxFIFO после приёма всего фрейма, игнорируя биты RTC.

0 - RxFIFO работает в режиме Cut-through, порог начала выдачи задан в битах RTC.

— Бит 24 **DFRF:** Запрет слива принятых фреймов при недоступности дескриптора/буфера

— Биты 23:22 Резерв, не трогать.

— Бит 21 **TSF:** Режим Накопления (Store-and-forward) при передаче

1 - TxFIFO выдаются только полностью записанные фреймы, игнорируя биты TTC.

0 - TxFIFO учитывает порог начала выдачи, заданный в битах TTC.

Бит можно писать только при остановленной передаче.

— Бит 20 **FTF:** Слить TxFIFO

Контроллер TxFIFO и указатели сбрасываются в начальное состояние, все данные теряются. Бит снимается автоматически после завершения сброса. При стоящем бите писать в этот регистр нельзя.

— Биты 19:17 Резерв, не трогать.

— Биты 16:14 **TTC:** Порог передачи из TxFIFO (при TSF=0)

Передача начинается если размер части фрейма в TxFIFO превышает порог, или записан полный фрейм.

000: 64

001: 128

010: 192

011: 256

100: 40

101: 32

110: 24

111: 16

— Бит 13 **ST:** Старт/Стоп передачи

1 - Старт. Машина идёт в Работу, DMA ищет в списке передаваемый фрейм, начиная с текущей позиции (начала списка в ETH_DMATDLAR или сохранённого состояния). Если текущий дескриптор не принадлежит DMA, то машина идёт в Приостановку и ставится бит недоступности буфера (ETH_DMASR[2]). Старт работает только при остановленной передаче. Если команду выдать до установки регистра ETH_DMATDLAR, то DMA сходит с ума.

0 - Стоп. После завершения передачи текущего фрейма машина уходит в Останов и сохраняется позиция следующего фрейма для рестарта. Команда Стоп работает только после завершения передачи текущего фрейма или в состоянии Приостановки.

— Биты 12:8 Резерв, не трогать.

— Бит 7 **FEF:** Выдавать сбойные фреймы из RxFIFO

1 - DMA выдаются все фреймы, кроме коротышей (runt).

0 - RxFIFO отбрасывает все фреймы с ошибкой (ошибка CRC, коллизия, гигантский фрейм, таймаут сторожевого таймера, переполнение). Но, если в режиме порога начало фрейма уже поступило на сторону чтения из RxFIFO, то фрейм не выбрасывается.

— Бит 6 **FUGF:** Выдавать маломерные хорошие фреймы

1 - Rx FIFO выдаёт хорошие фреймы короче 64 байт, включая CRC и расширитель.

0 - Rx FIFO отбрасывает фреймы короче 64 байт, кроме прошедших порог RTC.

— Бит 5 Резерв, не трогать.

— Биты 4:3 **RTC:** Порог приёма

Если длина принятой части фрейма превышает порог, то начинается выдача его DMA. Фреймы короче порога выдаются автоматически.

NB: Значение порога 11 не применимо если глубина RxFIFO равна 128 байтам.

NB: Эти биты работают только при RSF=0.

00: 64
01: 32
10: 96
11: 128

— Бит 2 **OSF:** Обрабатывать второй фрейм

Если стоит, то DMA начинает обрабатывать второй фрейм ещё до получения статуса первого.

— Бит 1 **SR:** Старт/Стоп приёма

1 - Старт. Машина идёт в Работу. DMA выбирает из списка приёма принадлежащий DMA дескриптор (с текущего положения или начала списка в регистре ETH_DMARDLAR) и начинает передачу в память. Если такого дескриптора нет, то приём приостанавливается и ставится бит недоступности буфера (ETH_DMASR [7]). Команда Старт действует только при остановленном приёме. Выдавать команду надо после записи регистра DMA ETH_DMARDLAR, иначе DMA может спятить.

0 - После завершения передачи текущего фрейма машина уходит в Останов и сохраняется позиция следующего фрейма для рестарта. Команда Стоп работает только при ожидании приёма пакета или в состоянии Приостановки.

— Бит 0 Резерв, не трогать.

Регистр разрешения прерываний Ethernet DMA (ETH_DMAIER)

Смещение адреса: 0x101C

По сбросу: 0x0000 0000

Стоящая 1 в бите разрешает соответствующее прерывание из регистра ETH_DMASR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															NISE	AISE	ERIE	FBEIE	Reserved		ETIE	RWTIE	RPSIE	RBUIE	RIE	TUIE	ROIE	TJTIE	TBUIE	TPSIE	TIE
															rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:17 Резерв, не трогать.

— Бит 16 **NISE:** Общее нормальное прерывание

Разрешает общее прерывание по битам:

- ETH_DMASR [0]: прерывание Передачи
- ETH_DMASR [2]: Буфер передачи недоступен
- ETH_DMASR [6]: прерывание Приёма
- ETH_DMASR [14]: прерывание Раннего приёма

— Бит 15 **AISE:** Общее ненормальное прерывание

Разрешает общее прерывание по битам:

- ETH_DMASR [1]: Передача остановлена
- ETH_DMASR [3]: Таймаут трёпа
- ETH_DMASR [4]: Переполнение RxFIFO
- ETH_DMASR [5]: Исчерпание передачи
- ETH_DMASR [7]: Буфер приёма недоступен
- ETH_DMASR [8]: Приём остановлен
- ETH_DMASR [9]: Таймаут сторожевого таймера приёма
- ETH_DMASR [10]: прерывание Ранней передачи
- ETH_DMASR [13]: Фатальная ошибка шины

— Бит 14 **ERIE:** Ранний приём.

— Бит 13 **FBEIE:** Фатальная ошибка шины.

— Биты 12:11 Резерв, не трогать.

— Бит 10 **ETIE:** Ранняя передач.

— Бит 9 **RWTIE:** Таймаут сторожевого таймера приёма

— Бит 8 **RPSIE:** Приём остановлен.

— Бит 7 **RBUIE:** Буфер приёма недоступен.

— Бит 6 **RIE:** Прерывание Приёма.

— Бит 5 **TUIE:** Исчерпание буфера передачи.

— Бит 4 **ROIE:** Переполнение буфера приёма.

— Бит 3 **TJTIE:** Таймаут трёпа передачи.

— Бит 2 **TBUIE:** Буфер передачи недоступен.

— Бит 1 **TPSIE:** Передача остановлена.

— Бит 0 **TIE:** Прерывание передачи.

Регистр числа потерянных фреймов и переполнения буфера Ethernet DMA (ETH_DMAMFBOCR)

Смещение адреса: 0x1020

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			OFOC	MFA											OMFC	MFC															
			rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

— Биты 31:29 Резерв, не трогать.

— Бит 28 **OFOC:** Бит переполнения счётчика переполнений FIFO

— Биты 27:17 **MFA:** Фреймы, потерянные программой

— Бит 16 **OMFC:** Бит переполнения счётчика потерянных фреймов

— Биты 15:0 **MFC:** Фреймы, потерянные контроллером по недоступности буфера.

Регистр текущего дескриптора передачи хоста Ethernet DMA (ETH_DMACHTDR)

Смещение адреса: 0x1048

По сбросу: 0x0000 0000

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

— Биты 31:0 **HTDAP:** Адрес начала текущего дескриптора передачи, изменяется DMA.

Регистр текущего дескриптора приёма хоста Ethernet DMA (ETH_DMACHRDR)

Смещение адреса: 0x104C

По сбросу: 0x0000 0000

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

— Биты 31:0 **HRDAP:** Адрес начала текущего дескриптора приёма, изменяется DMA.

Регистр адреса текущего буфера передачи хоста Ethernet DMA (ETH_DMACHTBAR)

Смещение адреса: 0x1050

По сбросу: 0x0000 0000

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

— Биты 31:0 **HTBAP:** Адрес буфера передачи, изменяется DMA.

Регистр адреса текущего буфера приёма хоста Ethernet DMA (ETH_DMACHRBAR)

Смещение адреса: 0x1054

По сбросу: 0x0000 0000

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

— Биты 31:0 **HRBAP:** Адрес буфера передачи, изменяется DMA.

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ETH_MACCR	Reserved								WD	JD	Reserved		IFG			CSD	Reserved	FES	ROD	LM	DM	IPCO	RD	Reserved	APCS	BL		DC	TE	RE	Reserved	
	Reset value									0	0			0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0		
0x04	ETH_MACFFR	RA	Reserved																				HPF	SAF		PCF		BFD	PAM	DAIF	HM	HU	PM
	Reset value	0																					0	0	0	0	0	0	0	0	0	0	0
0x08	ETH_MACHTHR	HTH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	ETH_MACHTLR	HTL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	ETH_MACMIIAR	Reserved																PA					MR				Reserved	CR			MW	MB
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	ETH_MACMIIDR	Reserved																MD															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	ETH_MACFCR	PT																Reserved								ZQPD	Reserved	PLT		UPFD	RFCE	TFCE	FCB/BPA
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0		0	0	0	0	0	0
0x1C	ETH_MACVLANTR	Reserved															VLANTC	VLANTI															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	ETH_MACRWUFFR	Frame filter reg0\Frame filter reg1\Frame filter reg2\Frame filter reg3\Frame filter reg4\...\Frame filter reg7																															
	Reset value	0																															
0x2C	ETH_MACPMTCSR	WFFRPR	Reserved																					GU	Reserved		WFR	MPR	Reserved			WFE	MPE	PD
	Reset value	0																						0			0	0				0	0	0
0x38	ETH_MACSR	Not applicable																Reserved						TSTS	Reserved		MMCTS	MMCRS	MMCS	PMTS	Reserved			
	Reset value																							0			0	0	0	0				
0x3C	ETH_MACIMR	Not applicable																Reserved						TSTIM	Reserved					PMTIM	Reserved			
	Reset value																							0						0				
0x40	ETH_MACA0HR	MO	Reserved															MACA0H															
	Reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x44	ETH_MACA0LR	MACA0L																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x48	ETH_MACA1HR	AE	SA	MBC[6:0]						Reserved								MACA1H															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x4C	ETH_MACA1LR	MACA1L																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x50	ETH_MACA2HR	AE	SA	MBC						Reserved								MACA2H															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x54	ETH_MACA2LR	MACA2L																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x58	ETH_MACA3HR	AE	SA	MBC						Reserved								MACA3H															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x5C	ETH_MACA3LR	MACA3L																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x100	ETH_MMCCR	Reserved																												MCF	ROR	CSR	CR
	Reset value																													0	0	0	0
0x104	ETH_MMCRIR	Reserved														RGUFS	Reserved										RFAES	RFCES	Reserved				
	Reset value															0											0	0					
0x108	ETH_MMCTIR	Reserved										TGFS	Reserved					TGFMSCS	TGFSCS	Reserved													
	Reset value											0						0	0														
0x10C	ETH_MMCRIMR	Reserved														RGUFM	Reserved										RFAEM	RFCEM	Reserved				
	Reset value															0											0	0					
0x110	ETH_MMCTIMR	Reserved										TGFM	Reserved					TGFMSCM	TGFSCM	Reserved													
	Reset value											0						0	0														
0x14C	ETH_MMCTGFSCCR	TGFSCC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x150	ETH_MMCTGFMSCCR	TGFMSCC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x168	ETH_MMCTGFCR	TGFC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x194	ETH_MMCRFCECR	RFCEC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x198	ETH_MMCRFAECR	RFAEC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C4	ETH_MMCRGUFCR	RGUFC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x700	ETH_PTPTSCR	Reserved																										TTSARU	TSITE	TSSTU	TSSTI	TSFCU	TSE
	Reset value																											0	0	0	0	0	0
0x704	ETH_PTPSSIR	Reserved																									STSSI						
	Reset value																										0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x708	ETH_PTPTSHR	STS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70C	ETH_PTPTSLR	STPNS	STSS																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x710	ETH_PTPTSHUR	TSUS																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x714	ETH_PTPTSLUR	TSUPNS	TSUSS																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x718	ETH_PTPTSAR	TSA																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x71C	ETH_PTPTTHR	TTSH																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x720	ETH_PTPTTLR	TTSL																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1000	ETH_DMABMR	Reserved						AAB	FPM	USP	RDP						FB	PM	PBL					EDFE	DSL					DA	SR	
	Reset value							0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	
0x1004	ETH_DMATPDR	TPD																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1008	ETH_DMARPDR	RPD																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100C	ETH_DMARDLAR	SRL																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1010	ETH_DMATDLAR	STL																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1014	ETH_DMASR	Reserved	TSTS	PMTS	MMCS	Reserved	EBS			TPS			RPS			NIS	AIS	ERS	FBES	Reserved	ETS	RWTS	RPSS	RBUS	RS	TUS	ROS	TJTS	TBUS	TPSS	TS		
	Reset value		0	0	0	Reserved	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0		
0x1018	ETH_DMAOMR	Reserved				DTCEFD	RSF	DFRF	Reserved	TSF	FTF	Reserved	TTC			ST	Reserved							FEF	FUGF	Reserved	RTC		OSF	SR	Reserved		
	Reset value					0	0	0		0	0		0	0	0	0								0	0		0	0	0	0			
0x101C	ETH_DMAIER	Reserved															NISE	AISE	ERIE	FBEIE	Reserved	ETIE	RWTIE	RPSIE	RBUIE	RIE	TUIE	ROIE	TJTIE	TBUIE	TPSIE	TIE		
	Reset value																0	0	0	0		0	0	0	0	0	0	0	0	0	0	0		
0x1020	ETH_DMAMFBOCR	Reserved			OFOC	MFA											OMFC	MFC															
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1048	ETH_DMACHTDR	HTDAP																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104C	ETH_DMACHRDR	HRDAP																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1050	ETH_DMACHTBAR	HTBAP																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1054	ETH_DMACHRBAR	HRBAP																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

30. Электронная сигнатура устройства

30.1. Регистр размера памяти

30.1.1. Регистр размера Флэш-памяти

Базовый адрес: 0x1FFF F7E0

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

— Биты 15:0 **F_SIZE:** Размер памяти от производителя в КБайтах, пример: 0x0080 = 128 КБ.

30.2. Регистр уникального ID устройства (96 бит)

Базовый адрес: 0x1FFF F7E8

Смещение адреса: 0x00

Только чтение = 0xXXXX от производителя.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID(15:0)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Смещение адреса: 0x02

Только чтение = 0xXXXX от производителя.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID(31:16)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Смещение адреса: 0x04

Только чтение = 0xXXXX XXXX от производителя.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID(63:48)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID(47:32)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Смещение адреса: 0x08

Только чтение = 0xXXXX XXXX от производителя.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID(95:80)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID(79:64)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

31. Поддержка отладки (DBG)

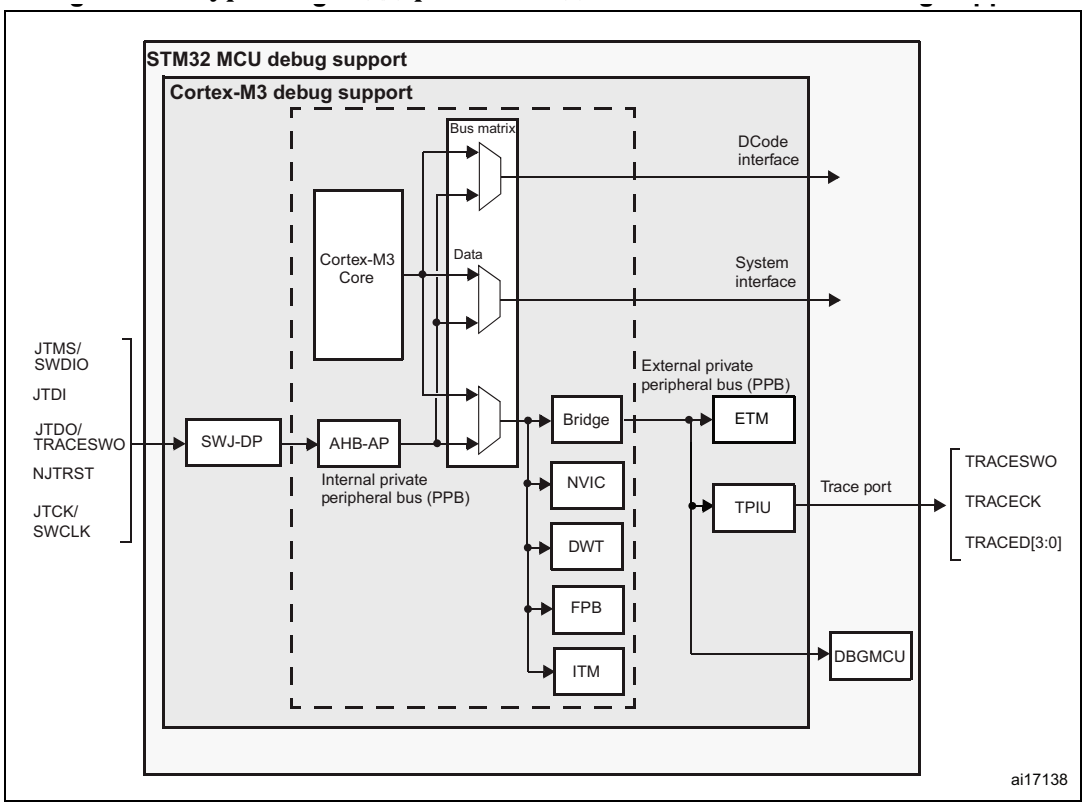
31.1. Обзор

STM32F10xxx построены вокруг ядра Cortex-M3 с аппаратной поддержкой отладки. Она позволяет останавливать ядро по выборке команды (контрольная точка) или доступу к данным (точка осмотра). Состояние остановленного ядра можно разглядывать на хосте.

Есть два интерфейса:

- Последовательный провод
- Порт отладки JTAG

Блок-схема уровней поддержки отладки STM32 MCU и Cortex-M3



NB: Средства отладки Cortex-M3 входят в Arm CoreSight Design Kit.

Средства отладки Arm Cortex-M3 включают:

- SWJ-DP: Последовательный провод / Порт отладки JTAG
- AHP-AP: Порт доступа к AHB
- ITM: Макроячейка инструментальной трассировки
- FPB: Контрольная точка патча Flash
- DWT: Включение точки просмотра данных
- TPUI: Интерфейс блока порта трассировки (у корпусов с нужными ножками)
- ETM: Встроенная макроячейка трассировки (у корпусов с нужными ножками)

У отладки STM32F10xxx есть:

- Гибкое назначение точек отладки
- Блок отладки MCU (поддержка экономных режимов, управление тактами и т.д.)

31.2. Документация от фирмы ARM

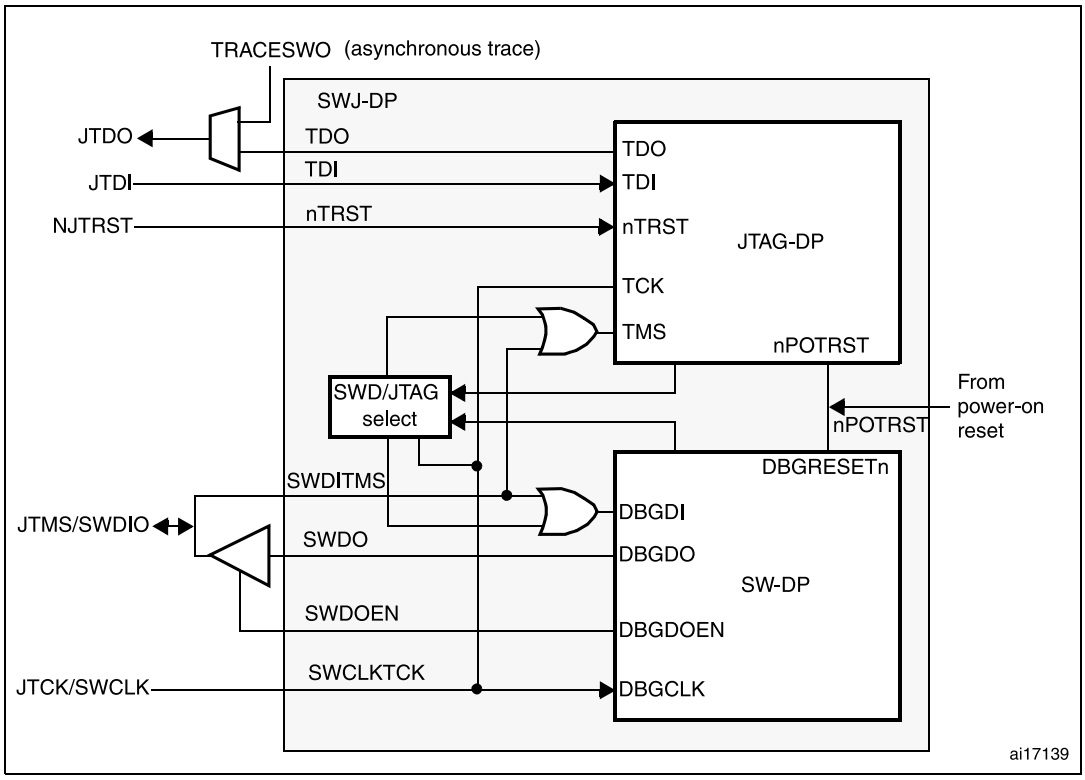
- Cortex-M3 r1p1 Technical Reference Manual (TRM) по адресу: *http://infocenter.arm.com/*
- Arm Debug Interface V5
- Arm CoreSight Design Kit revision r1p1 Technical Reference Manual

31.3. Порт отладки SWJ (последовательный провод и JTAG)

- JTAG Debug Port (JTAG-DP) это 5-контактный стандартный интерфейс JTAG с портом AHP-AP.
- The Serial Wire Debug Port (SW-DP) это 2-контактный (такты + данные) интерфейс с портом AHP-AP.

В SWJ-DP две ножки JTAG SW-DP мультиплексированы с некоторыми ножками JTAG из JTAG-DP.

На рисунке ниже асинхронный выход TRACE (TRACESWO) мультиплексируется с TDO. То есть асинхронная трассировка может использоваться с SW-DP, но не с JTAG-DP.



31.3.1. Механизм выбора JTAG-DP или SW-DP

По умолчанию активен порт JTAG-Debug.

Для переключения на SW-DP надо выдать JTAG последовательность TMS/TCK (на SWDIO и SWCLK):

1. Послать более 50 тактов TCK с TMS (SWDIO) =1
2. Послать 16 бит на TMS (SWDIO) = 0111100111100111 (старший бит первым)
3. Послать более 50 тактов TCK с TMS (SWDIO) =1

31.4. Pinout and debug port pins

Варианты MCU STM32F10xxx выпускаются в корпусах с разным числом ножек и некоторые функции в разных корпусах могут различаться.

31.4.1. Ножки порта SWJ

Имя ножки SWJ-DP	Порт JTAG		Порт SW		Ножка
	Тип	Описание	Тип	Назначение	
JTMS/SWDIO	I	Выбор режима теста JTAG	IO	Ввод/Вывод SW	PA13
JTCK/SWCLK	I	Такты теста JTAG	I	Такты SW	PA14
JTDI	I	Ввод данных JTAG	-	-	PA15
JTDO/ TRACESWO	O	Вывод данных JTAG	-	TRACESWO если разрешена асинхронная трассировка	PB3
NJTRST	I	nReset теста JTAG	-	-	PB4

31.4.2. Гибкое назначение ножек SWJ-DP

После RESET (SYSRESETn или PORESETn) все 5 ножек SWJ-DP сразу доступны отладчику (выходы трассировки надо назначать явно).

Распределением ножек ведает программа через регистр MCU (AFIO_MAPR).

Используются три бита, обнуляемых по сбросу.

• AFIO_MAPR (@ 0x40010004)

– READ: APB - Без состояний ожидания

– WRITE: APB - 1 состояние ожидания если буфер записи моста AHB-APB полон.

Биты 26:24= **SWJ_CFG[2:0]**

Ставятся и снимаются программно.

Это число ножек, отведённых порту SWJ, дабы освободить их для GPIO.

По умолчанию равно “000” (все от подключения JTAG-DP). Можно ставить только один бит.

Порты отладки	Ножки IO для SWJ				
	PA13 / JTMS / SWDIO	PA14 / JTCK / SWCLK	PA15 / JTDI	PB3 / JTDO	PB4 / NJTRST
Полн. SWJ (JTAG-DP + SW-DP) - Сброс	X	X	X	X	X
Полн, SWJ (JTAG-DP + SW-DP) без NJTRST	X	X	X	X	Свободно
JTAG-DP выкл. SW-DP вкл.	X	X	Свободно		
JTAG-DP выкл. SW-DP вкл.	Свободно				

NB: Если буфер записи моста APB полон, то запись в регистр AFIO_MAPR занимает один дополнительный такт APB из-за двух тактов деактивации ножек JTAGSW для гарантии чистого уровня входных сигналов ядра nTRST и TCK.

- Такт 1: входные сигналы JTAGSW ставятся в 1 для nTRST, TDI и TMS и в 0 для TCK
- Такт 2: контроллер GPIO занимает ножки для сигналов SWJTAG (вроде направления, подпорки/подтяжки, активации триггеров Шмитта и пр.).

31.4.3. Внутренняя подпорка и подтяжка ножек JTAG

Входы JTAG не должны быть плавающими, поскольку они напрямую подключены к триггерам. Особенно ножка SWCLK/TCK.

Входы JTAG имеют:

- NJTRST: подпорку
- JTDI: подпорку
- JTMS/SWDIO: подпорку
- TCK/SWCLK: подтяжку

Ножка JTAG IO возвращается контроллеру GPIO. По сбросу ножки GPIO получают состояние:

- NJTRST: подпорки
- JTDI: подпорки
- JTMS/SWDIO: подпорки
- JTCK/SWCLK: подтяжки
- JTDO: плавающие AF выход/вход

Это стандартные GPIO.

NB: Стандарт JTAG IEEE советует подпирать TDI, TMS и nTRST, для TCK советов нет. Но для JTCK внутренняя подтяжка нужна.

31.4.4. Последовательный провод и свободные ножки GPIO

Последовательный провод DP выделяется установкой release SWJ_CFG=010 сразу после сброса. Ножки PA15, PB3 и PB4 освобождаются.

При отладке хост:

- Под системным сбросом, назначает все ножки SWJ (JTAG-DP + SW-DP).
- Под системным сбросом, посылает последовательность переключения из JTAG-DP в SW-DP.
- Под системным сбросом, отладчик ставит контрольную точку на вектор сброса.

- Системный сброс снимается и ядро останавливается.
- Вся связь отладки идет по SW-DP, остальные ножки свободны.

NB: Освобождённые ножки отладки некоторое время остаются в состоянии подпорки-входа (nTRST, TMS, TDI), подтяжки (TCK) или в третьем состоянии выхода (TDO) вплоть до момента освобождения программой.

Если ножки отладки (JTAG or SW or TRACE) отведены отладке, то конфигурация контроллера IOPORT на них не влияет.

31.5. Подключение STM32F10xxx JTAG TAP

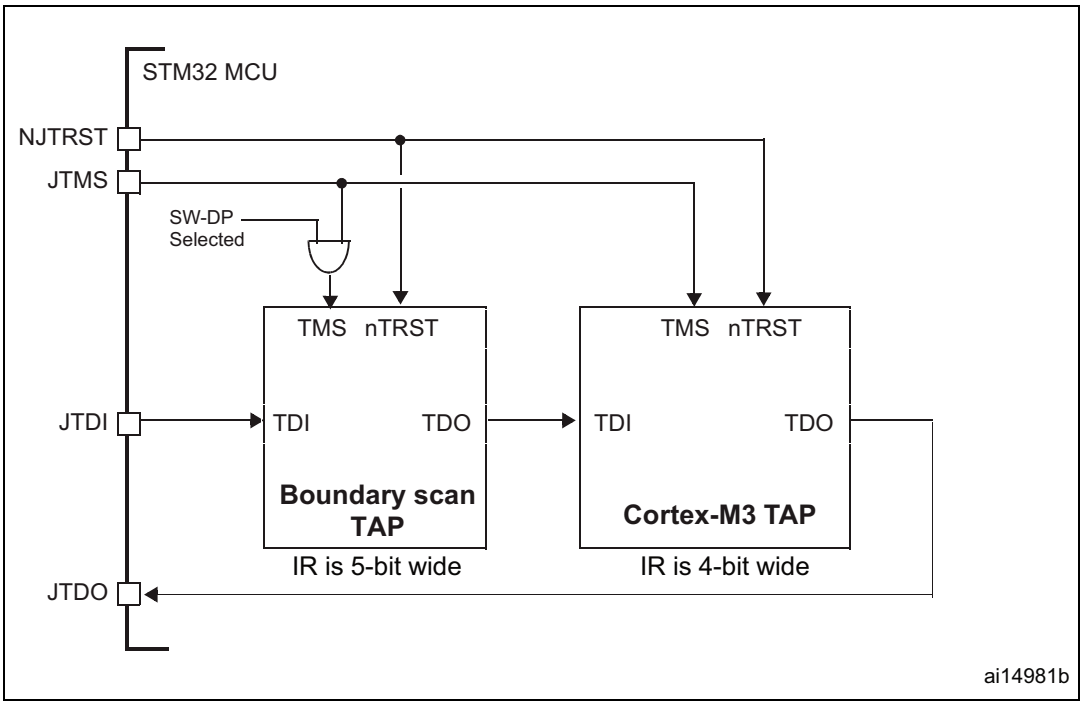
В STM32F10xxx MCU есть два последовательно включённых JTAG TAP, сканер границ TAP (IR шириной 5-бит) и Cortex-M3 TAP (IR шириной 4-бит).

Доступ к TAP Cortex-M3 для отладки:

1. Сначала вдвигаем команду BYPASS сканера границ TAP.
2. Затем, для каждого сдвига IR, цепочка скана содержит 9 бит (=5+4) и должна быть вдвинута команда неиспользованного TAP с помощью команды BYPASS.
3. По каждому сдвигу данных, неиспользованный TAP, стоящий в режиме BYPASS, добавляет 1 лишний бит в цепочку скана данных.

NB: **Важно**: Поскольку SW включается JTAG последовательностью, то сканер границ TAP автоматически отключается (JTMS высокий).

Подключение JTAG TAP



31.6. ID коды и блокировка

Разработчикам инструментов рекомендуется блокировать их изделия по коду DEVICE ID, лежащему во внешней PPB памяти по адресу 0xE0042000.

31.6.1. ID код устройства MCU

ID содержит номер части ST MCU ревизию кристалла. Это часть компонента DBG_MCU. Он доступен и портам отладки JTAG и SW и программе даже во время сброса MCU.

Отладчики должны использовать только `DEV_ID`[11:0].

DBGMCU_IDCODE

Адрес: 0xE004 2000

Только чтение, доступ 32 бита.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

— Биты 31:16 **REV_ID[15:0]:** Ревизия

Устройства низкой плотности:

– 0x1000 = Ревизия A

Устройства средней плотности:

– 0x0000 = Ревизия A
– 0x2000 = Ревизия B
– 0x2001 = Ревизия Z
– 0x2003 = Ревизия 1, 2, 3, X или Y

Устройства высокой плотности:

– 0x1000 = Ревизия A или 1
– 0x1001 = Ревизия Z
– 0x1003 = Ревизия 1, 2, 3, X или Y

Устройства XL-плотности:

– 0x1003 = Ревизия A или 1

Сетевые устройства:

– 0x1000 = Ревизия A
– 0x1001 = Ревизия Z

— Биты 15:12 Резерв, не трогать.

— Биты 11:0 **DEV_ID[11:0]**: Идентификатор устройства

Устройства низкой плотности: 0x412

Устройства средней плотности: 0x410

Устройства высокой плотности: 0x414

Устройства XL-плотности: 0x430

Сетевые устройства: 0x418

31.6.2. Сканер границ TAP (BSC)

ID код JTAG

- Устройства низкой плотности:
  – 0x06412041 = Ревизия A
- Устройства средней плотности:
  – 0x06410041 = Ревизия A
  – 0x16410041 = Ревизия B, Ревизия Z, Ревизия Y
- Устройства высокой плотности:
  – 0x06414041 = Ревизия A, Ревизия Z, Ревизия Y
- Устройства XL-плотности
  – 0x06430041 = Ревизия A
- Сетевые устройства:
  – 0x06418041 = Ревизия A, Ревизия Z

31.6.3. Cortex-M3 TAP

Cortex-M3 JTAG ID код неизменяемый, доступен только по JTAG DP, равен 0x3BA00477 (Cortex-M3 r1p1-01 rel 0).

31.6.4. Cortex-M3 JEDEC-106 ID код

Он лежит в 4KB таблице ROM на внутренней шине PPB по адресу 0xE00FF000_0xE00FFFFF. Доступен по JTAG DP, SW DP и программе.

31.7. Порт JTAG DP

Это стандартная машина состояний JTAG с 4-бит регистром команд (IR) и 4 регистрами данных.

Таблица 219. Регистры данных JTAG DP

IR(3:0)	Регистр	Детали
1111	BYPASS [1 бит]	—
1110	IDCODE [32 бита]	ID CODE = 0x3BA00477 (Arm Cortex-M3 r1p1-01rel0)
1010	DPACC [35 бит]	Регистр доступа к порту отладки, инициализирует порт и разрешает доступ к регистрам. – При передаче данных IN: Биты 34:3 = DATA[31:0] = 32-бит данные для запроса записи Биты 2:1 = A[3:2] = 2-бит адрес регистра порта отладки. См. *Таблицу 220*. Бит 0 = RnW = Запрос чтения (1) или записи (0). – При передаче данных OUT: Биты 34:3 = DATA[31:0] = 32-бит данные для запроса чтения Биты 2:0 = ACK[2:0] = 3-бит Подтверждения: 010 = OK/FAULT 001 = WAIT OTHER = резерв
1011	APACC [35 бит]	Регистр доступа к порту доступа, инициализирует порт и разрешает доступ к регистрам. – При передаче данных IN: Биты 34:3 = DATA[31:0] = 32-бит данные для запроса записи Биты 2:1 = A[3:2] = 2-бит адрес регистра AP. Бит 0 = RnW = Запрос чтения (1) или записи (0). – При передаче данных OUT: Биты 34:3 = DATA[31:0] = 32-бит данные для запроса чтения Биты 2:0 = ACK[2:0] = 3-бит Подтверждения: 010 = OK/FAULT 001 = WAIT OTHER = резерв Регистров AP много (см. AHB-AP) они адресуются комбинацией: – значения A[3:2] – текущего значения регистра DP SELECT
1000	ABORT [35 бит]	Аборт-регистр – Биты 31:1 = Резерв – Бит 0 = DAPABORT: запись 1 делает аборт DAP.

Таблица 220. 32-бит регистры порта отладки по адресу A[3:2]

Адрес	A[3:2]	Описание
0x0	00	Резерв, не трогать.
0x4	01	Регистр DP CTRL/STAT для: – Запроса включения питания системы или отладки – Конфигурации передач доступа к AP – Управление сравнением и проверкой вталкивания. – Чтения некоторых флагов (переполнение, подтверждения включения)
0x8	10	Регистр DP SELECT для выбора порта доступа и 4-слов окна регистров: – Биты 31:24: APSEL: выбор текущего AP – Биты 23:8: резерв – Биты 7:4: APBANKSEL: выбор 4-слов окна регистров текущего AP – Биты 3:0: резерв
0xC	11	Регистр DP RDBUFF для получения результата последовательности операций (без запроса новой операции JTAG-DP)

31.8. Порт SW DP

31.8.1. Введение в протокол SW

Имеет две линии:

- SWCLK: такты от хоста
- SWDIO: двунаправленная

Протокол имеет два банка регистров (DPACC и APACC) для чтения и записи.

Биты передаются начиная с младшего.

Линия SWDIO должна иметь внешнюю подпорку на плате (рекомендуется 100 KΩ).

При каждом изменении направления SWDIO вставляется задержка переключения. Обычно это время одного бита, зависит от частоты SWCLK.

31.8.2. Последовательность протокола SW

Состоит из трёх фаз:

1. Пакет запроса (8 бит) от хоста
2. Ответ подтверждения (3 бита) для хоста
3. Передача данных (33 бита) от или для хоста

Таблица 221. Пакет запроса (8-бит)

Бит	Имя	Описание
0	Start	Должен быть “1”
1	APnDP	0: DP, 1: AP
2	RnW	0: Запрос записи 1: Запрос чтения
4:3	A[3:2]	Поле адреса регистров DP или AP (см. *Таблицу 220.*)
5	Parity	Бит чётности
6	Stop	0
7	Park	Не выдаётся хостом. Должен читаться “1” из-за подпорки

Пакет запроса всегда сопровождается задержкой переключения (обычно 1 бит).

Таблица 222. Ответ ACK (3 бита)

Бит	Имя	Описание
0..2	ACK	001: FAULT 010: WAIT 100: OK

Ответ ACK должен сопровождаться задержкой переключения для передачи READ или при получении ответа WAIT или FAULT.

Таблица 223. Передача DATA (33 бита)

Бит	Имя	Описание
0..31	WDATA RDATA	Данные записи или чтения
32	Parity	Бит чётности для 32 битов данных

Передача DATA должна сопровождаться задержкой переключения только для передачи READ.

31.8.3. Машина состояний SW-DP (сброс, простой, ID код)

Код ID машины состояний SW-DP соответствует стандарту JEP-106 и равен 0x1BA01477 (соответствует Cortex-M3 r1p1).

NB: Пока устройство читает ID код, машина состояний SW-DP неактивна.

- Машина идёт в состояние RESET после сброса по питанию, после переключения DP из JTAG в SWD или после более чем 50 тактов высокого состояния линии
- Машина идёт в состояние IDLE если линия низкая не менее двух тактов после состояния RESET.
- После состояния RESET при входе в IDLE надо сначала **обязательно** выполнить READ к регистру DP-SW ID CODE. На другой запрос устройство выдаст ответ FAULT.

См. *Cortex-M3 r1p1 TRM* и *CoreSight Design Kit r1p0 TRM*.

31.8.4. Доступ чтения/записи DP и AP

- Чтение DP не постируется: ответ может быть немедленным (ACK=OK) или задержанным (ACK=WAIT).
- Чтение AP постируется, то есть результат возвращается в следующей передаче. Если следующее обращение это не доступ к AP, то результат надо получить из регистра DP-RDBUFF. Флаг успешного чтения READOK в регистре DP-CTRL/STAT обновляется по каждому запросу чтения из AP или из RDBUFF.
- У SW-DP есть буфер записи (для обоих DP или AP). Если он полон, то устройство отвечает “WAIT” за исключением чтения IDCODE или CTRL/STAT или записи ABORT.
- Из-за асинхронности доменов SWCLK и HCLK после передачи бита чётности операции записи нужны два дополнительных такта SWCLK низкого состояния линии (состояние IDLE). Это особенно важно при записи запроса включения питания CTRL/STAT. Иначе следующая передача будет сбойной.

31.8.5. Регистры SW DP

Доступ к ним идёт при APnDP=0

Таблица 224. Регистры SW DP

A[3:2]	R/W	Бит CTRLSEL регистра SELECT	Регистр	Заметки
0	Read	-	IDCODE	The manufacturer code is not set to ST code. 0x1BA01477 (identifies the SW-DP)
0	Write	-	ABORT	-
1	Read/ Write	0	DP- CTRL/STAT	Назначение: – запрос включения питания системы или отладки – конфигурация передач для доступа к AP – управление сравнением и проверкой вталкивания. – чтение некоторых флагов(переполнение, подтверждения включения)
1	Read/ Write	1	WIRE CONTROL	Для управления физическим протоколом (вроде длины задержки)
10	Read	-	READ RESEND	Повторная передача данных без повторения запроса AP.
10	Write	-	SELECT	Выбор текущего порта доступа и активного окна регистров
11	Read/ Write	-	READ BUFFER	Чтение буфера после предыдущего запроса чтения AP

31.8.6. Регистры SW AP

Регистры доступны при APnDP=1

Регистров AP много (см. AHB-AP), адресуются они с помощью:

- Значения A[3:2]
- Текущего значения регистра DP SELECT

31.9. AHB-AP (порт доступа к AHB) - для JTAG-DP и SW-DP

Свойства:

- Доступ к системе не зависит от состояния процессора.
- AHB-AP доступен и SW-DP и JTAG-DP.
- AHB-AP является ведущим AHB на Матрице Шин. Отсюда, ему доступны все шины данных (Dcode Bus, System Bus, внутренняя и внешняя шина PPB), кроме шины ICode.
- Поддерживаются Bitband-передачи.
- Передачи AHB-AP минуют FPB.

Адрес 32-бит регистров AHP-AP 6-битный (до 64 слов или 256 байт) и состоит из:

c) Биты [7:4] = биты[7:4] APBANKSEL регистра DP SELECT

d) Биты [3:2] = 2 бита адреса из A[3:2] из 35-бит пакета запроса SW-DP.

AHB-AP Cortex-M3 имеет 9 x 32-бит регистров:

Таблица 225. Регистры Cortex-M3 AHB-AP

Смещение адреса	Имя регистра	Замечания
0x00	Слово управления и статуса AHB-AP	Configures and controls transfers through the AHB interface (size, hprot, status on current transfer, address increment type
0x04	Адрес передачи AHB-AP	—
0x0C	Данные чтения/записи AHB-AP	—
0x10	Банк данных 0 AHB-AP	Прямое отображение 4 выравненных слов данных без изменения регистра Адреса Передачи.
0x14	Банк данных 1 AHB-AP	
0x18	Банк данных 2 AHB-AP	
0x1C	Банк данных 3 AHB-AP	
0xF8	Адрес ROM отладки AHB-AP	Базовый адрес интерфейса отладки
0xFC	Регистр ID AHB-AP	—

31.10. Отладка ядра

Она доступна по регистрам отладки через порт AHB-AP. Процессор может обращаться к ним по внутренней шине PPB. Их четыре:

Таблица 226. Регистры отладки ядра

Регистр	Описание
DHCSR	32-бит регистр статуса и управления пошаговой отладки ядра.
DCRSR	17-бит регистр выбора регистра ядра.
DCRDR	32-бит регистр данных из/для регистра ядра.
DEMCR	32-бит регистр отладки исключений и мониторинга, содержит бит ***TRCENA*** разрешения TRACE.

NB: **Важно**: эти регистры сбрасываются только по включению питания.

См. *Cortex-M3 r1p1 TRM*. Для Останова по сбросу надо:

- поставить бит 0 (`VC_CORRESET`) регистра `DEMCR`.
- поставить бит 0 (`C_DEBUGEN`) регистра `DHCSR`.

31.11. Отладка под сбросом системы

Источники сброса системы:

- POR (включение питания) выставляет RESET по каждому включению питания.
- Внутренний сторожевой таймер
- Программный сброс
- Внешний сброс

Cortex-M3 различает сброс отладки (обычно PORRESETn) и другой (SYSRESETn). То есть отладчик может подключиться во время сброса, остановив ядро при чтении вектора сброса. Тогда хост освобождает системный сброс и ядро останавливается без выполнения команды. Кроме того, под системным сбросом можно ставить любые функции отладки.

NB: Настоятельно рекомендуется ставить контрольную точку вектора сброса под системным сбросом.

31.12.FPB (Точка патча Flash)

Блок FPB:

- реализует аппаратные контрольные точки
- направляет команды и данные из пространства кода в системное пространство. Так можно исправлять ошибки в пространстве кода.

Использование Программного патча и Аппаратных точек взаимно исключающее.

FPB содержит:

- 2 компаратора литералов для чтения из пространства кода и направления в пространство системы.
- 6 компараторов команд для выборки из пространства кода. Их можно использовать для переключения в соответствующее пространство системы или выдачи команды контрольной точки.

31.13.DWT (Запуск точки осмотра данных)

Блок DWT состоит из 4 компараторов. Они конфигурируются как:

- аппаратная контрольная точка
- запуск ETM
- читатель PC
- читатель адреса данных

Для профилирования DWT имеет счётчики:

- Тактов
- Вложенных команд
- Блоков операций загрузки /записи (LSU)
- Тактов сна
- CPI (тактов на команду)
- Прерываний

31.14.ITM (Макроячейка инструментальной трассировки)

31.14.1.Общее описание

ITM поддерживает трассировку ОС и программных событий в стиле *printf*. ITM выдаёт пакеты:

- **Software trace.** Программа может писать в регистр стимула ITM.
- **Hardware trace.** Пакеты создаёт DWT, а выдаёт ITM их.
- **Time stamping.** Метки времени выдаются относительно пакетов. Для этого в ITM есть 21-бит счётчик. Считает он такты Cortex-M3 или выходные такты от *Serial Wire Viewer* (SWV).

Пакеты от ITM выдаются TPIU (Trace Port Interface Unit). Форматтер TPIU добавляет свои пакеты (см. TPIU) и выводит полную последовательность пакетов хосту отладчика.

При использовании ITM бит `TRCEN` в регистре `DEMCR` должен стоять.

31.14.2.Пакеты меток времени, синхронизации и переполнения

Пакеты меток времени содержат время, общее управление и синхронизацию. Они используют счётчик меток времени (21 бит) с возможным предделителем, который сбрасывается по каждой выдаче метки. Он тактируется от тактов CPU или SWV.

Пакет синхронизации состоит из 6 байтов `0x80_00_00_00_00_00`, которые передаются TPIU как `00 00 00 00 00 80` (младший бит вперёд).

Пакет синхронизации управляет пакетами меток, он выдаётся по каждому запуску DWT.

Для этого DWT должен запускать ITM: стоит `CYCCNTENA` в регистре управления DWT. Кроме того, должен стоять бит `SYNCENA` регистре управления ITM.

Если бит `SYNCENA` не стоит, то DWT выдаёт запуск синхронизации TPIU, который посылает только пакеты синхронизации TPIU, но не пакеты синхронизации ITM.

Пакет переполнения состоит из специальных пакетов меток времени, извещающих о том, что данные писались в полный FIFO.

Таблица 227. Главные регистры ITM

Адрес	Регистр	Детали
@E0000FB0	Блокировка доступа к ITM	Запись 0xC5ACCE55 разрешает запись в другие регистры ITM
@E0000E80	Управление трассировкой ITM	Биты 31-24 = всегда 0
		Бит 23 = Занят
		Биты 22-16 = 7-бит ATB ID источника данных трассировки.
		Биты 15-10 = всегда 0
		Биты 9:8 = TSPrescale = Предделитель меток времени
		Биты 7-5 = Резерв
		Бит 4 = SWOENA = Разрешить такты SWV для меток времени.
		Бит 3 = DWTENA: Разрешить Стимул DWT
		Бит 2 = SYNCENA: 1 разрешает DWT выдавать запуск синхронизации TPIU для выдачи пакетов синхронизации.
		Бит 1 = TSENA (Разрешение меток времени)
		Бит 0 = ITMENA: Глобальный бит разрешения ITM
@E0000E40	Привилегии трассировки ITM	Бит 3: маска разрешения портов трассировки 31:24
		Бит 2: маска разрешения портов трассировки 23:16
		Бит 1: маска разрешения портов трассировки 15:8
		Бит 0: маска разрешения портов трассировки 7:0
@E0000E00	Разрешение трассировки ITM	Каждый бит разрешает один порт Стимула для выдачи трассировки.
@E0000000-E000007C	Регистры портов Стимула 0-31	Запиши 32-бит данные в выбранный порт Стимула (их 32) для его трассировки.

Пример конфигурации

Выводим простое число в TPIU:

- Организуем TPIU и назначаем TRACE I/O записью в `DBGMCU_CR`
- Пишем 0xC5ACCE55 в регистр блокировки ITM
- Пишем 0x00010005 в регистр управления трассировкой ITM для включения ITM с разрешённым Sync и ATB ID, отличным от 0x00
- Пишем 0x1 в регистр разрешения трассировки ITM для включения порта Стимула 0
- Пишем 0x1 в регистр привилегий трассировки ITM для снятия маски портов Стимула 7:0
- Пишем выводимое число в порт Стимула, регистр 0

31.15.ETM (Встроенная макроячейка трассировки)

31.15.1.Общее описание

ETM позволяет отследить выполнение программы. За данными следят через компонент DWT макроячейки ITM, за командами через ETM.

ETM передаёт информацию пакетами по запуску от встроенных ресурсов. Эти ресурсы программируются независимо, источник запуска определяется регистром События запуска (0xE0041008). Это может быть простым событием (от компаратора адреса) или логическим

уравнением 2 событий. Источник запуска это один из 4 компараторов модуля DWT. Отслеживаются события:

- Совпадение такта
- Совпадение адреса данных

ETM выдаёт пакеты TPIU. Форматтер TPIU добавляет свои пакеты и выдаёт полную последовательность пакетов отладчику хоста.

31.15.2.Протокол сигналов ETM и типы пакетов

Эта часть описана в главе 7 документа *ETMv3 Signal Protocol of the Arm IHI 0014N*.

31.15.3.Главные регистры ETM

Таблица 228. Главные регистры ETM

Адрес	Регистр	Детали
0xE0041FB0	Блокировка доступа ETM	Запись 0xC5ACCE55 разрешает запись в другие регистры ETM.
0xE0041000	Управление ETM	Управление общими операциями ETM.
0xE0041010	Статус ETM	Информация о текущем статусе трассировки и логике запуска.
0xE0041008	Событие запуска ETM	Определяет событие запуска.
0xE004101C	Компаратор трассировки ETM	Выбирает компаратор.
0xE0041020	Событие трассировки ETM	Определяет событие трассировки.
0xE0041024	Старт/Стоп трассировки ETM	Определяет события Старта и Стопа трассировки.

31.15.4.Пример конфигурации ETM

Выводим простое число в TPIU:

- Конфигурируем TPIU и разрешаем I/IO_TRACEN для назначения TRACE I/O в устройствах высокой и XL-плотности.
- Пишем 0xC5AC CE55 to the ETM Lock Access Register to unlock the write access to the ITM registers
- Пишем 0x0000 1D1E в регистр управления ETM (конфигурируем трассировку)
- Пишем 0x0000 406F в регистр события запуска ETM (выбираем событие)
- Пишем 0x0000 006F в регистр разрешения события ETM (выбираем событие start/stop)
- Пишем 0x0000 0001 в регистр Start/stop ETM (включаем трассировку)
- Пишем 0x0000191E в регистр управления ETM (конец конфигурации)

31.16.Компонент отладки MCU (DBGMCU)

Он поддерживает :

- Режимы пониженного питания
- Управление тактированием таймеров, сторожевых таймеров, I²C и bxCAN в контрольных точках
- Назначение ножек трассировки

31.16.1.Поддержка отладки в режимах пониженного питания

В эти режимы попадают после команд WFI и WFE. Их несколько.

Ядро не позволяет при отладке выключать FCLK или HCLK. Для отладки ставим регистры конфигурации:

- В режиме Сна отладчик заранее ставит бит `DBG_SLEEP` в регистре `DBGMCU_CR`. Так HCLK запитывается от FCLK.

- В режиме Останова отладчик заранее ставит бит DBG_STOP. Так FCLK и HCLK запитываются от внутреннего RC генератора.

31.16.2.Поддержка отладки для таймеров, сторожевиков, bxCAN и I²C

Надо решить будут ли таймеры работать внутри контрольной точки:

- Могут считать. Обычно это нужно при ШИМ управлении моторами и т.п.
- Не могут считать. Это надо сторожевым таймерам.

Для bxCAN можно выбрать блокировку обновления регистра приёма.

Для I²C можно выбрать блокировку таймаута SMBUS.

У остановленных таймеров (DBG_TIMx_STOP = 1) с инверсными выходами, они выключены (как будто снят бит MOE).

31.16.3.Регистр конфигурации отладки MCU

Это относится к поддержке:

- Режимов пониженного питания
- Таймеров и сторожевиков and
- bxCAN
- Назначения ножек трассировки

Регистр DBGMCU_CR лежит на внешней шине PPB по адресу 0xE0042004

Он асинхронно сбрасывается PORESET (не системным). Отладчик может писать в него под системным сбросом. Если хост отладчика не поддерживает этих функций, то это можно делать программно.

DBGMCU_CR register

Адрес: 0xE004 2004

PORESET: 0x0000 0000

Доступен только словами (32 бита)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_TIM9_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	Reserved			DGB_CAN2_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM8_STOP	DBG_I2C2_SMBUS_TIMEOUT
	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_I2C1_SMBUS_TIMEOUT	DBG_CAN1_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP	DBG_TIM1_STOP	DBG_WWDG_STOP	DBG_IWDG STOP	TRACE_MODE [1:0]		TRACE_IOEN	Reserved		DBG_STANDBY	DBG_STOP	DBG_SLEEP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw

— Бит 31 Резерв, не трогать.

— Биты 30:25 **DBG_TIMx_STOP:** Счётчик TIMx стоит при стоящем ядре (x=9..14)

0: Счётчик считает, его выводы выводят.

1: Счётчик стоит, его выводы отключены (как при останове по событию break).

— Биты 24:22 Резерв, не трогать.

— Бит 21 **DBG_CAN2_STOP:** CAN2 стоит при стоящем ядре

0: Нормальное поведение

1: Регистры приёма CAN2 заморожены

— Биты 20:17 **DBG_TIMx_STOP:** Счётчик TIMx стоит при стоящем ядре (x=8..5)

0: Счётчик считает, его выводы выводят.

1: Счётчик стоит, его выводы отключены (как при останове по событию break).

— Бит 16 **DBG_I2C2_SMBUS_TIMEOUT:** Таймаут SMBUS стоит при стоящем ядре

0: Нормальное поведение

1: Таймаут SMBUS заморожен

— Бит 15 **DBG_I2C1_SMBUS_TIMEOUT:** SMBUS timeout mode stopped when Core is halted

0: Нормальное поведение

1: Таймаут SMBUS заморожен

— Бит 14 **DBG_CAN1_STOP:** CAN1 стоит при стоящем ядре
0: Нормальное поведение
1: Регистры приёма CAN1 заморожены

— Биты 13:10 **DBG_TIMx_STOP:** Счётчик TIMx стоит при стоящем ядре (x=4..1)
0: Счётчик считает
1: Счётчик стоит

— Бит 9 **DBG_WWDG_STOP:** Оконный сторожевик стоит при стоящем ядре
0: Работает
1: Стоит

— Бит 8 **DBG_IWDG_STOP:** Независимый i сторожевик стоит при стоящем ядре
0: Работает
1: Стоит

— Биты 7:5 **TRACE_MODE[1:0]** и **TRACE_IOEN**: Назначение ножек трассировки
– TRACE_IOEN=0:
TRACE_MODE=xx: TRACE ноги не назначены (по умолчанию)
– TRACE_IOEN=1:
– TRACE_MODE=00: Назначения TRACE для асинхронного режима
– TRACE_MODE=01: Назначения TRACE для синхронного режима с TRACEDATA=1
– TRACE_MODE=10: Назначения TRACE для синхронного режима с TRACEDATA=2
– TRACE_MODE=11: Назначения TRACE для синхронного режима с TRACEDATA=4

— Биты 4:3 Резерв, не трогать.

— Бит 2 **DBG_STANDBY**: Режим Standby отладки
0: (FCLK=Off, HCLK=Off) Вся цифровая часть отключена.
Для программы выход из Standby равен выбору вектора сброса (кроме битов статуса)
1: (FCLK=On, HCLK=On) Цифровая часть запитана, FCLK и HCLK кормятся от внутреннего RC генератора. При выходе из Standby MCU выдаёт системный сброс.

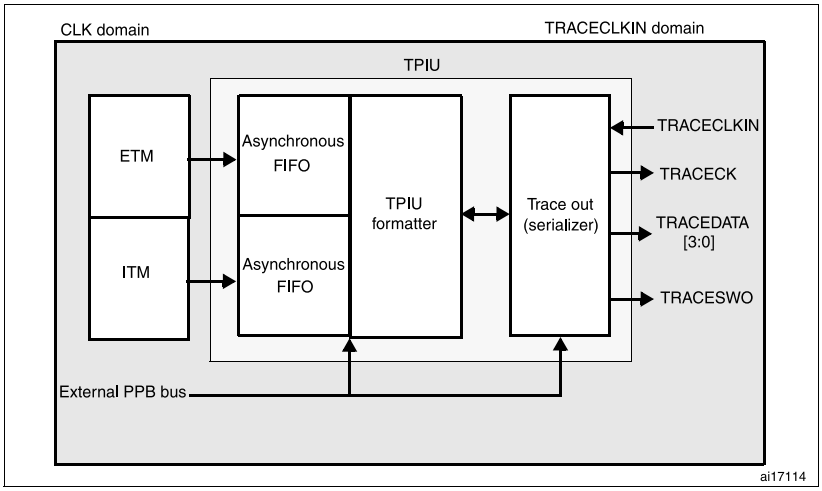
— Бит 1 **DBG_STOP:** Режим Stop отладки
0: (FCLK=Off, HCLK=Off) Все такты, включая HCLK и FCLK отключены. Выход из STOP равен выходу после RESET (CPU тактируется от внутреннего 8 MHz RC генератора (HSI)).
1: (FCLK=On, HCLK=On) FCLK и HCLK питаются от внутреннего RC генератора.
В обоих случаях при выходе из STOP надо восстановить нормальное тактирование

— Бит 0 **DBG_SLEEP:** Режим Sleep отладки
0: (FCLK=On, HCLK=Off) FCLK кормится системными тактами, HCLK выключен. Контроллер тактов не сбрасывается.
1: (FCLK=On, HCLK=On) HCLK и FCLK получают системные такты (как заповедано Программой).

31.17.TPIU (Блок интерфейса порта трассировки)

31.17.1.Введение

TPIU это мост для данных от ITM и ETM. Выходной поток данных включает ID источника трассировки и отправляется *анализатору порта трассировки* (TPA). Тут мы имеем простенький TPIU, специальную версию CoreSight TPIU.



31.17.2.Назначение ног TRACE

- Асинхронный режим

Ему нужна 1 дополнительная нога, она есть всегда. Доступен только в режиме SW (не в JTAG).

Таблица 229. Асинхронная нога TRACE

Имя ножки TPUI	Type	Description	STM32F10xxx pin assignment
TRACESWO	O	TRACE Async Data Output	PB3

- Синхронный режим

Здесь нужны от 2 до 6 дополнительных ног, они есть только в больших корпусах. Доступен в режимах JTAG и SW.

Таблица 230. Синхронные ноги TRACE

Имя ножки TPUI	Type	Description	Нога
TRACECK	O	TRACE Clock	PE2
TRACED[3:0]	O	TRACE Sync Data Outputs Can be 1, 2 or 4.	PE[6:3]

Назначение ножек TRACE TPUI

По умолчанию, эти ножки не назначены. Они назначаются битами `TRACE_IOEN` и `TRACE_MODE` в регистре конфигурации отладки MCU. Это выполняет хост. Число ног зависит от режима работы (асинхронный или синхронный).

- асинхронный режим: 1 дополнительная нога
- синхронный режим: от 2 до 5 ног, в зависимости от размера регистра порта (1, 2 или 4):
  - TRACECK
  - TRACED(0) при размере порта 1, 2 или 4
  - TRACED(1) при размере порта 2 или 4
  - TRACED(2) при размере порта 4
  - TRACED(3) при размере порта 4

Хост назначает ноги TRACE битами `TRACE_IOEN` и `TRACE_MODE`[1:0] регистра `DBGMCU_CR`.

Он лежит на внешней шине PPB и сбрасывается PORESET, отладчик может писать всего под системным сбросом.

Таблица 231. Гибкое назначение ног TRACE

DBGMCU_CR		Режим	Нога TRACE IO					
TRACE _IOEN	TRACE _MODE [1:0]		PB3 /JTDO/ TRACESWO	PE2/ TRACECK	PE3 / TRACED[0]	PE4 / TRACED[1]	PE5 / TRACED[2]	PE6 / TRACED[3]
0	XX	No Trace	Свободно[1]	-				
1	0	Асихнрон.	TRACESWO	-	-	Свободно[1]		
1	1	Синхр. 1 бит	Свободно[1]	TRACECK	TRACED[0]	-	-	-
1	10	Синхр. 2 бита		TRACECK	TRACED[0]	TRACED[1]	-	-
1	11	Синхр. 4 бита		TRACECK	TRACED[0]	TRACED[1]	TRACED[2]	TRACED[3]

1. Свободна в режиме SW. В режиме JTAG назначена JTDO.

NB: По умолчанию вход тактов TPIU (TRACECLKIN) замкнут на GND. Он подключается к HCLK через 2 такта после установки бита `TRACE_IOEN`.

Отладчик ставит режим трассировки в битах `PROTOCOL`[1:0] регистра `SPP_R` TPIU.

- `PROTOCOL`=00: Синхронный
- `PROTOCOL`=01 или 10: SW (Manchester или NRZ) (асинхронный). По умолчанию - 01

Размер порта TRACE пишется в биты[3:0] регистра CPSPS_R:

- 0x1 — 1 нога (по умолчанию)
- 0x2 — 2 ноги
- 0x8 — 4 ноги

31.17.3.Форматтер TPUI

Он выводит данные 16-байт фреймами:

- 7 байтов данных
- 8 байтов, содержащих:
  - 1 бит (младший) индикатор байта DATA (‘0) или ID (‘1).
  - 7 битов (старшие) с данными или ID источника.
- 1 вспомогательный байт в котором 1 бит соответствует 1 байту из предыдущих 8:
  - для байта с данными это бит 0 данных.
  - для байта со сменой ID бит показывает, что ID change меняется.

31.17.4.Пакеты синхронизации фреймов TPUI

TPUI выдаёт два типа пакетов синхронизации:

- Пакет Синхронизации Фреймов (пакет Синхронизации Полным Словом)
  Это слово: 0x7F_FF_FF_FF (младший бит выдаётся первым). Оно больше нигде не появляется, источника с ID е 0x7F нету.
  Он периодически выдаётся ***между*** фреймами. В непрерывном режиме TPA их выбрасывает.
- Пакет Синхронизации Полу-словом
  Это полу-слово: 0x7F_FF (младший бит выдаётся первым).
  Он периодически выдаётся ***между*** или ***внутри*** фреймов.
  Выдаётся в непрерывном режиме, показывая TPA, что TRACE порт стоит в режиме IDLE (считывать TRACE не надо). TPA его выбрасывает.

31.17.5.Передача пакета синхронизации фреймов

Регистра Счётчика Синхронизации в TPIU нет и запуск синхронизации выдаёт **DWT**.

Пакет фрейма синхронизации (0x7F_FF_FF_FF) выдаётся:

- после каждого снятия сброса TPIU. Он снимается синхронно с передним фронтом такта TRACECLKIN. То есть выдаётся при стоящем бите TRACE_IOEN регистра DBGMCU_CFG. В этом случае за словом 0x7F_FF_FF_FF не следует форматированный пакет.
- по каждому запуску DWT (DWT уже сообразован). Вариантов два:
  - Если бит SYNENA ITM снят, то выдаётся только слово 0x7F_FF_FF_FF без форматированного пакета за ним.
  - Если бит SYNENA ITM стоит, то последуют пакеты синхронизации ITM (0x80_00_00_00_00_00), форматированные TPUI (добавлен ID источника).

31.17.6.Синхронный режим

Данные могут выдаваться по 4, 2 или 1 ножкам: TRACED(3:0) Для отладчика подаются такты (TRACECK), которые подключаются к HCLK только при работе TRACE.

NB: В этом режиме стабильность тактов не нужна.

TRACE I/O (включая TRACECK) управляются передним фронтом TRACLKIN (равно HCLK). Отсюда частота TRACECK равна HCLK/2.

31.17.7.Асинхронный режим

Это дешёвый вариант вывода через одну ножку TRACESWO. Полоса ограничена.

Ножка TRACESWO мультиплексируется с JTDO и доступна во всех корпусах STM32F10xxx.

Здесь нужна постоянная частота TRACECLKIN с точностью 5% для UART (NRZ) и 10% для Манчестерского кода.

31.17.8.Подключение TRACECLKIN внутри STM32F10xxx

Вход TRACECLKIN уже подключён к HCLK. То есть в асинхронном режиме надо использовать фреймы времени при стабильной частоте CPU.

NB: **Важно:** в асинхронном режиме остерегайтесь того, что:

MCU тактируется от внутреннего генератора RC. Под сбросом его частота отличается от частоты после снятия сброса. То есть под сбросом TPA не должен включать трассировку битом TRACE_IOEN поскольку время в пакетах синхронизации под сбросом и без оного будет различаться.

31.17.9.registers TPIU

Регистры TPIU APB доступны только при стоящем бите TRCENA в регистре DEMCR. Иначе всегда читаются нули (их вывод разрешает PCLK в TPIU).

Таблица 232. Важные регистры TPIU

Адрес	Регистр	Описание
0xE0040004	Размер текущего порта	Бит 0: Размер порта = 1 Бит 1: Размер порта = 2 Бит 2: Размер порта = 3, не поддерживается Бит 3: Размер порта = 4 Должен стоять только один бит. По умолчанию размер равен 1 (0x00000001)
0xE00400F0	Протокол выбранной ножки	Биты 1:0 = 00: Синхронный порт 01: SW вывод - Манчестер (по умолчанию) 10: SW вывод - NRZ 11: резерв
0xE0040304	Управление форматтером и сливом	Биты 31-9 = всегда ‘0 Бит 8 = TrigIn = индикация запуска, всегда ‘1 Биты 7-4 = всегда 0 Биты 3-2 = всегда 0 Бит 1 = EnFCont. У синхронного порта (в регистре протокола биты 1:0=00), этот бит всегда ‘1: форматер разрешён в непрерывном режиме. В асинхронном режиме (в регистре протокола биты 1:0 <> 00), этот бит ставится самостоятельно. Бит 0 = всегда 0 По умолчанию общее значение равно 0x102 **NB:** В синхронном режиме ножка TRACECTL наружу не выводится и форматтер всегда разрешён в непрерывном режиме - он вставляет пакеты идентификации источника.
0xE0040300	Статус форматтера и слива	В Cortex-M3 не используется, всегда читается как 0x00000008

31.17.10.Пример конфигурации

- Ставим бит TRCENA в регистре DEMCR
- В регистр размера порта TPIU пишем нужное число (по умолчанию там 0x1 для 1-бит порта)
- В регистр управления форматтером и сливом TPIU пишем 0x102 (значение по умолчанию)
- В регистре протокола TPIU выбираем синхронный или асинхронный протокол. Пример: 0x2 для асинхронного режима NRZ (как у UART)
- В регистр DBGMCU пишем 0x20 (бит IO_TRACEN) для назначения TRACE I/O асинхронного режима. В это время выдаётся пакет синхронизации TPIU (FF_FF_FF_7F)
- Конфигурируем ITM и пишем в регистр Стимула ITM для выдачи числа.

31.18.Карта регистров DBG

Addr.	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE0042000	DBGMCU_IDCODE	REV_ID																Reserved				DEV_ID											
	Reset value(1)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0xE0042004	DBGMCU_CR	Reserved										DGB_CAN2_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM8_STOP	DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	DBG_CAN1_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP	DBG_TIM1_STOP	DBG_WWDG_STOP	DBG_IWDGSTOP	TRACE_MODE[1:0]		TRACE_IOEN	Reserved		DBG_STANDBY	DBG_STOP	DBG_SLEEP
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0

Это ВСЁ!