37 Гибкий контроллер памяти (FMC)

Контроллер гибкой памяти (FMC) включает в себя три контроллера памяти:

- Контроллер памяти NOR / PSRAM
- Контроллер памяти NAND / PC Card
- Синхронный контроллер DRAM (SDRAM / Mobile LPSDR SDRAM)

Этот раздел относится только к STM32F42xxx и STM32F43xxx.

37.1 Основные функции FMC

Функциональный блок FMC обеспечивает интерфейс с синхронными и асинхронными статическими памятью, памятью SDRAM и 16-разрядными картами памяти ПК. Его основными целями являются:

- перевести транзакции AHB в соответствующий протокол внешних устройств
- для удовлетворения требований времени доступа к внешним устройствам памяти

Все внешние запоминающие устройства совместно используют адрес, данные и управляющие сигналы с контроллером.

Доступ к каждому внешнему устройству осуществляется с помощью уникального Chip Select. FMC одновременно выполняет только один доступ к внешнему устройству.

Основными функциями контроллера FMC являются:

•Интерфейс с отображенными устройствами со статической памятью, включая:

- Статическая память произвольного доступа (SRAM)
- Флэш-память NOR / Флэш-память OneNAND (см. стр. 24 у нас таокй нет)
- PSRAM (4 банка памяти)
- 16-разрядные устройства, совместимые с PC Card
- Два банка флэш-памяти NAND с оборудованием ECC для проверки до 8 Кбайт данных

- Интерфейс с синхронной памятью DRAM (SDRAM / Mobile LPSDR SDRAM)
- Поддержка режима Burst для более быстрого доступа к синхронным устройствам, таким как флэш-память (NOR, PSRAM и SDRAM)
- Программируемый непрерывный выходной сигнал для асинхронного и синхронного доступа

Шина данных шириной 8, 16 или 32 бит

- Независимый чип выбора элемента управления для каждого банка памяти
- Независимая конфигурация для каждого банка памяти
- Выбор разрешения и выходов выбора полосы байтов для использования с устройствами PSRAM, SRAM и SDRAM
- Внешний асинхронный контроль ожидания
- Запись данных FIFO с 16 x33-битной глубиной
- Задать адрес FIFO с глубиной 16x30 бит
- Cacheable Read FIFO с 6 x32-разрядной глубиной (6 x14-разрядный адресный тег) для контроллера SDRAM.

FMC объединяет два FIFO записи: записывает FIFO данных с глубиной 16x33 бит и адрес FIFO записи с глубиной 16x30 бит.

256 Кбайт RAM, 2 Мбайт Flash, 64 Мбайт SDRAM (через FMC)

• Функция записи данных FIFO хранит данные AHB для записи в память (до 32 бит) плюс один бит для передачи AHB (пакетный или не последовательный режим)

• Адрес записи FIFO сохраняет адрес AHB (до 28 бит) плюс размер данных AHB (до 2 бит). При работе в пакетном режиме сохраняется только начальный адрес, кроме случаев пересечения границы страницы (для PSRAM и SDRAM). В этом случае пакет AHB разбивается на две записи FIFO.

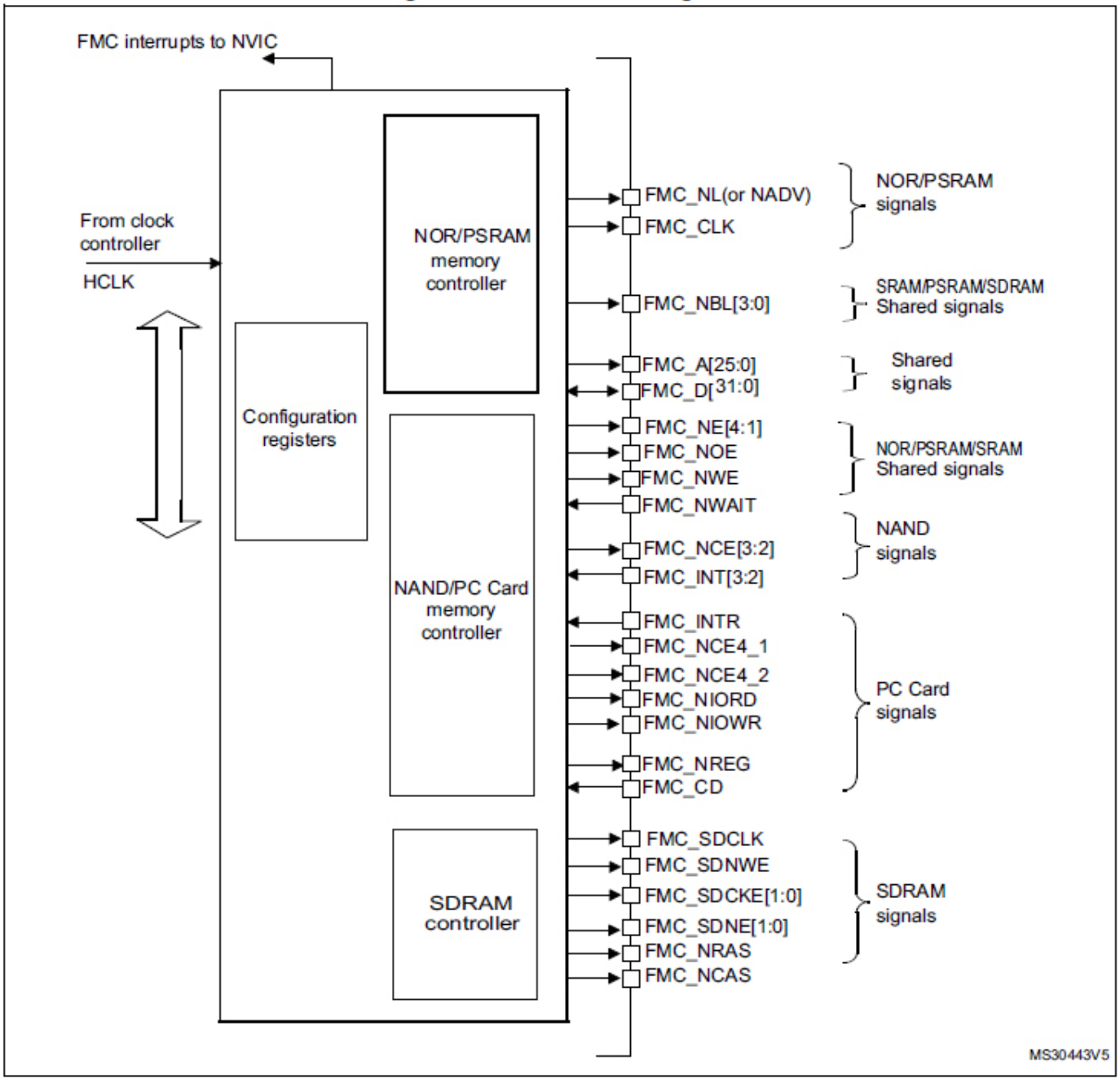
При запуске контакты FMC должны быть настроены пользовательским приложением. Штыри ввода / вывода FMC, которые не используются приложением, могут использоваться для других целей.

Регистры FMC, определяющие тип внешнего устройства, и связанные характеристики обычно устанавливаются во время загрузки и не изменяются до следующего сброса или включения питания. Тем не менее, настройки могут быть изменены в любое время.

37.2 Блок-схема

FMC состоит из пяти основных блоков:

Figure 454. FMC block diagram



- Интерфейс AHB (включая регистры конфигурации FMC)
- Контроллер NOR Flash / PSRAM / SRAM
- Контроллер NAND Flash / PC Card
- Контроллер SDRAM
- Интерфейс внешнего устройства

Блок-схема показана на рисунке 454.

37.3 Интерфейс AHB

Интерфейс ведомого устройства AHB позволяет внутренним процессорам и другим периферийным устройствам шины обращаться к внешним запоминающим устройствам.

Операции AHB транслируются во внешний протокол устройства. В частности, если выбранная внешняя память имеет ширину 16 или 8 бит, транзакции 32 бит в AHB разделяются на последовательные 16- или 8-битные обращения. FMC Chip Select (FMC_NEx) не переключается между последовательными обращениями, кроме случаев, когда выполняется доступ в режиме D с включенным расширенным режимом.

FMC генерирует ошибку AHB в следующих условиях:

- При чтении или записи в банк FMC (Банк 1 - 4), который не включен.
- При чтении или записи в банк NOR Flash, в то время как бит FACCEN сбрасывается в регистре FMC_BCRx.
- При чтении или записи в банки PC Card, когда входной контакт FMC_CD (обнаружение присутствия карты) низкий.
- При записи в банк SDRAM с защитой от записи (бит WP, установленный в регистре SDRAM_SDCRx).
- Когда диапазон адресов SDRAM нарушен (доступ к зарезервированному диапазону адресов)

Эффект ошибки AHB зависит от мастера AHB, который попытался получить доступ к R / W:

- При попытке доступа Cortex®-M4 с процессором FPU генерируется жесткое прерывание.
- Если доступ был выполнен контроллером DMA, генерируется ошибка передачи DMA, и соответствующий канал DMA автоматически отключается.

Часы AHB (HCLK) являются опорными часами для FMC.

37.3.1 Поддерживаемые воспоминания и транзакции

Общие правила транзакций

Запрошенный размер данных транзакции AHB может быть 8-, 16- или 32-битным, тогда как доступное внешнее устройство имеет фиксированную ширину данных. Это может привести к непоследовательным переводам.

Поэтому необходимо соблюдать некоторые простые правила транзакций:

Размер транзакции AHB и размер данных памяти равны

В этом случае нет никаких проблем.

Размер транзакции AHB больше, чем размер памяти:

В этом случае FMC разбивает транзакцию AHB на более мелкие последовательные обращения к памяти для соответствия ширине внешних данных. FMC Chip Select (FMC_NEx) не переключается между последовательными обращениями.

Размер транзакции AHB меньше, чем размер памяти:

Передача может быть или не быть последовательной в зависимости от типа внешнего устройства:

- Доступ к устройствам с функцией выбора байта (SRAM, ROM, PSRAM, SDRAM)

В этом случае FMC разрешает транзакции чтения / записи и обращается к правильным данным через свои байтовые полосы BL [3: 0].

байт, которые должны быть записаны, адресуются NBL [3: 0].

Все байты памяти считываются (NBL [3: 0] управляются низким во время транзакции чтения), и бесполезные отбрасываются.

- Доступ к устройствам, у которых нет функции выбора байта (16-разрядные флэш-память NOR и NAND)

Эта ситуация возникает, когда запрос байта запрашивается в 16-разрядную флэш-память. Поскольку устройство не может быть доступно в байтовом режиме (только 16-разрядные слова могут быть прочитаны / записаны из / во флэш-память), транзакции записи и транзакции чтения (контроллер считывает все 16-разрядное слово памяти и использует только требуемый байт).

Регистры конфигурации

FMC может быть настроен через набор регистров. Обратитесь к разделу 37.5.6 за подробным описанием регистров контроллера NOR Flash / PSRAM. Обратитесь к разделу 37.6.8 за подробным описанием регистров NAND Flash / PC Card и в разделе 37.7.5 для подробного описания регистров контроллера SDRAM.

37.4. Отображение адресов внешних устройств

С точки зрения FMC внешняя память делится на 6 банков фиксированного размера по 256 Мбайт каждый (см. Рис. 455):

• Банк 1 используется для адресации до 4 флэш-памяти NOR или устройств PSRAM. Этот банк разделен на 4 подбанки NOR / PSRAM с 4 выделенными Chip Selects, а именно:

- Банк 1 - NOR / PSRAM 1
- Банк 1 - NOR / PSRAM 2
- Банк 1 - NOR / PSRAM 3
- Банк 1 - NOR / PSRAM 4

• Банки 2 и 3 используются для обращения к устройствам флэш-памяти NAND (1 устройство на банк)

• Банк 4, используемый для обращения к PC-карте

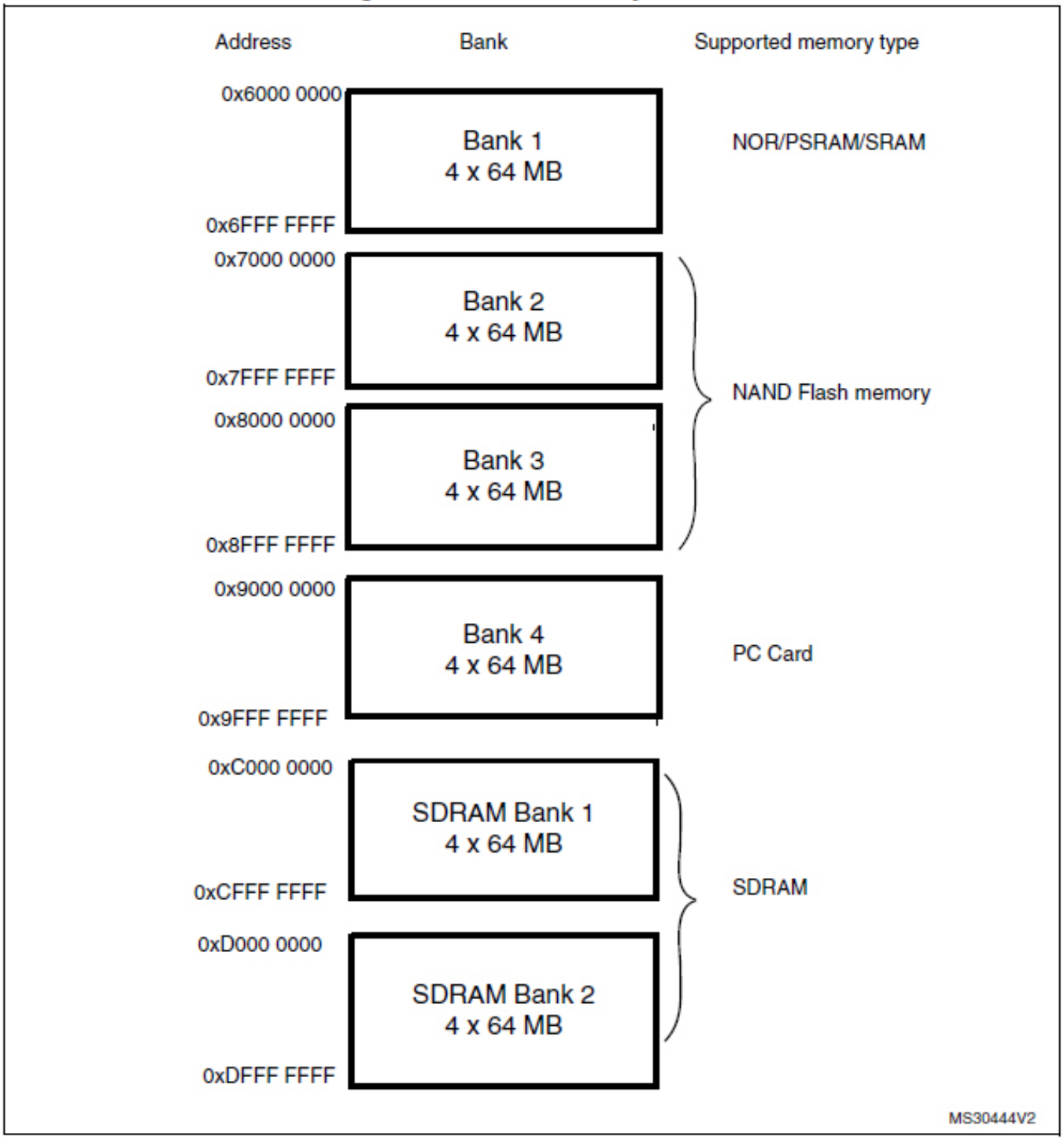
• Банк 5 и 6 используется для адресации устройств SDRAM (1 устройство на банк).

Для каждого банка тип используемой памяти может быть настроен пользовательским приложением через регистр конфигурации.

37.4.1 Отображение адресов NOR / PSRAM

Биты HADDR [27:26] используются для выбора одного из четырех банков памяти, как показано в таблице 250.

Figure 455. FMC memory banks



1. HADDR - это внутренние адресные строки AHB, которые переводится во внешнюю память.

Биты HADDR [25: 0] содержат адрес внешней памяти. Поскольку HADDR является байтовым адресом, тогда как память адресована на уровне слова, адрес, фактически выданный в память, зависит от ширины данных памяти, как показано в следующей таблице.

Table 250. NOR/PSRAM bank selection

HADDR[27:26](1)	Selected bank
00	Bank 1 - NOR/PSRAM 1
01	Bank 1 - NOR/PSRAM 2
10	Bank 1 - NOR/PSRAM 3
11	Bank 1 - NOR/PSRAM 4

1. В случае 16-битной внешней памяти, FMC будет внутренне использовать HADDR [25: 1] для генерации адреса для внешней памяти FMC_A [24: 0]. В случае 32-разрядной ширины памяти FMC будет внутренне использовать HADDR [25: 2] для генерации внешнего адреса.

Независимо от ширины внешней памяти, FMC_A [0] должен быть подключен к внешнему адресу памяти A [0].

Table 251. NOR/PSRAM External memory address

Memory width(1)	Data address issued to the memory	Maximum memory capacity (bits)
8-bit	HADDR[25:0]	64 Mbyte x 8 = 512 Mbit
16-bit	HADDR[25:1] >> 1	64 Mbyte/2 x 16 = 512 Mbit
32-bit	HADDR[25:2] >> 2	64 Mbyte/4 x 32 = 512 Mbit

Поддержка Wrap для NOR Flash / PSRAM

Режим пакетной печати для синхронных запоминающих устройств не поддерживается. Память должна быть настроена в режиме линейного пакета неопределенной длины.

37.4.2. Отображение адресов карты памяти NAND / ПК.

В этом случае доступны три банка, каждый из которых разделен на области памяти, как указано в таблице 252.

Для флэш-памяти NAND общие пространства памяти и атрибутов подразделяются на три секции (см. Таблицу 253 ниже), расположенных в нижних 256 Кбайтах:

Table 252. NAND/PC Card memory mapping and timing registers

Start address	End address	FMC bank	Memory space	Timing register
0x9C00 0000	0x9FFF FFFF	Bank 4 - PC card	I/O	FMC_PIO4 (0xB0)
0x9800 0000	0x9BFF FFFF		Attribute	FMC_PATT4 (0xAC)
0x9000 0000	0x93FF FFFF		Common	FMC_PMEM4 (0xA8)
0x8800 0000	0x8BFF FFFF	Bank 3 - NAND Flash	Attribute	FMC_PATT3 (0x8C)
0x8000 0000	0x83FF FFFF		Common	FMC_PMEM3 (0x88)
0x7800 0000	0x7BFF FFFF	Bank 2- NAND Flash	Attribute	FMC_PATT2 (0x6C)
0x7000 0000	0x73FF FFFF		Common	FMC_PMEM2 (0x68)

- Раздел данных (первые 64 Кбайта в общем пространстве памяти / атрибута)
- Команда (второй 64 Кбайт в общем пространстве памяти / атрибута)
- Раздел Address (следующие 128 Кбайт в общем пространстве памяти / атрибута)/

Прикладное программное обеспечение использует 3 раздела для доступа к флэш-памяти NAND:

- Чтобы отправить команду на флэш-память NAND, программное обеспечение должно записать значение команды в любое место памяти в разделе команд.

Table 253. NAND bank selection

Section name	HADDR[17:16]	Address range
Address section	1X	0x020000-0x03FFFF
Command section	01	0x010000-0x01FFFF
Data section	00	0x000000-0x0FFFF

• Чтобы указать NAND Flash-адрес, который должен быть прочитан или записан, программное обеспечение должно записать значение адреса в любое место памяти в разделе адреса. Поскольку адрес может быть длиной 4 или 5 байт (в зависимости от фактического объема памяти), для указания полного адреса требуется несколько последовательных операций записи в адресную секцию.

• Чтобы прочитать или записать данные, программа считывает или записывает данные из / в любую ячейку памяти в разделе данных.

Поскольку флэш-память NAND автоматически увеличивает адреса, нет необходимости увеличивать адрес раздела данных для доступа к последовательным ячейкам памяти.

37.4.3 Отображение адресов SDRAM

Бит HADDR [28] (внутренняя строка 28 AHB-адреса) используется для выбора одного из двух банков памяти, как указано в таблице 254.

В следующей таблице показано отображение SDRAM для 13-разрядной строки, 11-битного столбца и 4 внутренних банковских конфигураций

Table 254. SDRAM bank selection

HADDR[28]	Selected bank	Control register	Timing register
0	SDRAM Bank1	FMC_SDCR1	FMC_SDTR1
1	SDRAM Bank2	FMC_SDCR2	FMC_SDTR2

1. При взаимодействии с 16-разрядной памятью FMC внутренне использует внутренние адресные строки AAD HADDR [11: 1] для генерации внешнего адреса. При взаимодействии с 32-разрядной памятью FMC внутренне использует строки HADDR [12: 2] для генерации внешнего адреса. Независимо от ширины памяти, FMC_A [0] должен быть подключен к внешнему адресу памяти A [0].

2. Автозаполнение не поддерживается. FMC_A [10] должен быть подключен к внешнему адресу памяти A [10], но он всегда будет «низким».

Table 255. SDRAM address mapping

Memory width(1)	Internal bank	Row address	Column address(2)	Maximum memory capacity (Mbyte)
8-bit	HADDR[25:24]	HADDR[23:11]	HADDR[10:0]	64 Mbyte: 4 x 8K x 2K
16-bit	HADDR[26:25]	HADDR[24:12]	HADDR[11:1]	128 Mbyte: 4 x 8K x 2K x 2
32-bit	HADDR[27:26]	HADDR[25:13]	HADDR[12:2]	256 Mbyte: 4 x 8K x 2K x 4

Биты HADDR [27: 0] переводятся на внешний SDRAM-адрес в зависимости от конфигурации контроллера SDRAM:

- Размер данных: 8, 16 или 32 бит
- Различный размер: 11, 12 или 13 бит
- Размер столбца: 8, 9, 10 или 11 бит
- Количество внутренних банков: два или четыре внутренних банка

В таблице 256 в таблице показано отображение адресов SDRAM по сравнению с конфигурацией контроллера SDRAM.

1. БАНК [1: 0] - адрес банка BA [1: 0]. Когда используется только 2 внутренних банка, BA1 всегда должен быть установлен на «0».

2. Доступ к зарезервированному (Res.) Диапазону адресов генерирует ошибку AHB.

Table 256. SDRAM address mapping with 8-bit data bus width(1)(2)

Row size configuration	HADDR(AHB Internal Address Lines)																											
	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
11-bit row size configuration	Res.							Bank [1:0]		Row[10:0]											Column[7:0]							
	Res.								Bank [1:0]			Row[10:0]										Column[8:0]							
	Res.									Bank [1:0]			Row[10:0]									Column[9:0]						
	Res.										Bank [1:0]			Row[10:0]									Column[10:0]					
12-bit row size configuration	Res.								Bank [1:0]			Row[11:0]										Column[7:0]						
	Res.									Bank [1:0]			Row[11:0]									Column[8:0]						
	Res.										Bank [1:0]			Row[11:0]									Column[9:0]					
	Res.											Bank [1:0]			Row[11:0]									Column[10:0]				
13-bit row size configuration	Res.									Bank [1:0]			Row[12:0]									Column[7:0]						
	Res.										Bank [1:0]			Row[12:0]									Column[8:0]					
	Res.											Bank [1:0]			Row[12:0]									Column[9:0]				
	Res.												Bank [1:0]			Row[12:0]									Column[10:0]			

1. БАНК [1: 0] - адрес банка BA [1: 0]. Когда используется только 2 внутренних банка, BA1 всегда должен быть установлен на «0».

2. Доступ к зарезервированному пространству (Res.) Генерирует ошибку AHB.

3. BM0: это байтовая маска для 16-битного доступа.

Table 257. SDRAM address mapping with 16-bit data bus width(1)(2)

Row size Configuration	HADDR(AHB address Lines)																										
	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
11-bit row size configuration	Res.						Bank [1:0]		Row[10:0]										Column[7:0]									BM0(3)
	Res.						Bank [1:0]		Row[10:0]										Column[8:0]									BM0
	Res.						Bank [1:0]		Row[10:0]										Column[9:0]									BM0
	Res.						Bank [1:0]		Row[10:0]										Column[10:0]									BM0
12-bit row size configuration	Res.						Bank [1:0]		Row[11:0]										Column[7:0]									BM0
	Res.						Bank [1:0]		Row[11:0]										Column[8:0]									BM0
	Res.						Bank [1:0]		Row[11:0]										Column[9:0]									BM0
	Res.						Bank [1:0]		Row[11:0]										Column[10:0]									BM0
13-bit row size configuration	Res.						Bank [1:0]		Row[12:0]										Column[7:0]									BM0
	Res.						Bank [1:0]		Row[12:0]										Column[8:0]									BM0
	Res.						Bank [1:0]		Row[12:0]										Column[9:0]									BM0
	Res.						Bank [1:0]		Row[12:0]										Column[10:0]									BM0

1. БАНК [1: 0] - адрес банка BA [1: 0]. Когда используется только 2 внутренних банка, BA1 всегда должен быть установлен на «0».

2. Доступ к зарезервированному пространству (Res.) Генерирует ошибку AHB.

3. BM [1: 0]: это байтовая маска для 32-битного доступа.

Table 258. SDRAM address mapping with 32-bit data bus width(1)(2)

Row size configuration	HADDR(AHB address Lines)																										
	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
11-bit row size configuration	Res.						Bank [1:0]		Row[10:0]										Column[7:0]								BM[1:0](3)	
	Res.						Bank [1:0]		Row[10:0]										Column[8:0]								BM[1:0	
	Res.						Bank [1:0]		Row[10:0]										Column[9:0]								BM[1:0	
	Res.						Bank [1:0]		Row[10:0]										Column[10:0]								BM[1:0	
12-bit row size configuration	Res.						Bank [1:0]		Row[11:0]										Column[7:0]								BM[1:0	
	Res.						Bank [1:0]		Row[11:0]										Column[8:0]								BM[1:0	
	Res.						Bank [1:0]		Row[11:0]										Column[9:0]								BM[1:0	
	Res.						Bank [1:0]		Row[11:0]										Column[10:0]								BM[1:0	
13-bit row size configuration	Res.						Bank [1:0]		Row[12:0]										Column[7:0]								BM[1:0	
	Res.						Bank [1:0]		Row[12:0]										Column[8:0]								BM[1:0	
	Res.						Bank [1:0]		Row[12:0]										Column[9:0]								BM[1:0	
	Bank [1:0]						Row[12:0]												Column[10:0]								BM[1:0	

37.5 Контроллер NOR Flash / PSRAM

FMC генерирует соответствующие тайминги сигнала для управления следующими типами памяти:

• Асинхронный SRAM и ПЗУ
- 8 бит
- 16 бит
- 32 бита
• PSRAM (сотовая память)
- Асинхронный режим
- Режим серийного синхронного доступа
- мультиплексированные или не мультиплексированные
• NOR Флэш-память
- Асинхронный режим
- Режим серийного синхронного доступа
- мультиплексированные или не мультиплексированные

FMC выводит уникальный сигнал Chip Select, NE [4: 1], на банк. Все остальные сигналы (адреса, данные и управление) являются общими.

FMC поддерживает широкий диапазон устройств с помощью программируемых таймингов, среди которых:

• Программируемые состояния ожидания (до 15)
• Программируемые циклы оборотов шины (до 15)
• Программируемые разрешения на выход и запись разрешения (до 15)
• Независимые тайминги и протокол чтения и записи для поддержки самых разнообразных воспоминаний и таймингов
• Программируемое непрерывное время (FMC_CLK).

FMC Clock (FMC_CLK) - это подмножество часов HCLK. Он может быть доставлен на выбранное внешнее устройство либо во время синхронного доступа, либо во время асинхронного и синхронного доступа в зависимости от конфигурации бит CCKEN в регистре FMC_BCR1:

• Если бит CCLKEN сбрасывается, FMC генерирует часы (CLK) только при синхронном доступе (операции чтения / записи).

• Если бит CCLKEN установлен, FMC генерирует непрерывные часы во время асинхронного и синхронного доступа. Чтобы генерировать непрерывные часы FMC_CLK, банк 1 должен быть сконфигурирован в синхронном режиме (см. Раздел 37.5.6: регистры контроллера NOR / PSRAM). Поскольку такие же часы используются для всех синхронных запоминающих устройств, когда генерируется непрерывный выходной тактовый сигнал и выполняются синхронные обращения, размер данных AHB должен быть таким же, как и ширина данных памяти (MWID), в противном случае частота FMC_CLK будет изменена в зависимости от Передача данных AHB (см. Раздел 37.5.5: Синхронные транзакции для формулы отношения делителя FMC_CLK).

Размер каждого банка фиксирован и равен 64 Мбайт. Каждый банк сконфигурирован через специальные регистры (см. Раздел 37.5.6: регистры контроллера NOR / PSRAM).

Параметры программируемой памяти включают время доступа (см. Таблицу 259) и поддержку управления ожиданиями (для PSRAM и NOR Flash, доступ к которым осуществляется в пакетном режиме).

37.5.1 Сигналы интерфейса внешней памяти

Таблица 260, Таблица 261 и Таблица 262 перечисляют сигналы, которые обычно используются для взаимодействия с флэш-памятью NOR, SRAM и PSRAM.

Примечание. Префикс «N» идентифицирует активные сигналы низкого уровня.

Table 259. Programmable NOR/PSRAM access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Address setup	Duration of the address setup phase	Asynchronous	AHB clock cycle (HCLK)	0	15
Address hold	Duration of the address hold phase	Asynchronous, muxed I/Os	AHB clock cycle (HCLK)	1	15
Data setup	Duration of the data setup phase	Asynchronous	AHB clock cycle (HCLK)	1	256
Bust turn	Duration of the bus turnaround phase	Asynchronous and synchronous read/write	AHB clock cycle (HCLK)	0	15
Clock divide ratio	Number of AHB clock cycles (HCLK) to build one memory clock cycle (CLK)	Synchronous	AHB clock cycle (HCLK)	2	16
Data latency	Number of clock cycles to issue to the memory before the first data of the burst	Synchronous	Memory clock cycle (CLK)	2	17

37.5.3 Общие временные правила

Синхронизация сигналов

• Все выходные сигналы контроллера изменяются на переднем фронте внутренних часов (HCLK)

• В синхронном режиме (чтение или запись) все выходные сигналы изменяются по переднему фронту HCLK. Независимо от значения CLKDIV, все выходы изменяются следующим образом:

- NOEL / NWEL / NEL / NADVL / NADVH / NBLL / Исправлены действительные выходные адреса на заднем фронте часов FMC_CLK.

- NOEH / NWEH / NEH / NOEH / NBLH / Недействительные выходные адреса изменяются по нарастающему фронту часов FMC_CLK.

37.5.4 Асинхронные транзакции контроллера NOR Flash / PSRAM

Асинхронные статические памяти (NOR Flash, PSRAM, SRAM)

• Сигналы синхронизируются внутренними часами HCLK. Эти часы не выдаются в память

• FMC всегда производит выборку данных перед отменой сигнала NOE. Это гарантирует соблюдение временного ограничения хранения данных памяти (минимальный переход на высокочастотный переход к переходу данных обычно равен 0 нс)

• Если расширенный режим включен (бит EXTMOD установлен в регистре FMC_BCRx), доступны до четырех расширенных режимов (A, B, C и D). Можно использовать режимы A, B, C и D для операций чтения и записи. Например, операция чтения может выполняться в режиме A и записываться в режиме B.

• Если расширенный режим отключен (бит EXTMOD сбрасывается в регистре FMC_BCRx), FMC может работать в режимах 1 или режиме2 следующим образом:

- Режим 1 является режимом по умолчанию, когда выбран тип памяти SRAM / PSRAM (MTYP [1: 0] = 0x0 или 0x01 в регистре FMC_BCRx)

- Режим 2 - это режим по умолчанию, когда выбран тип памяти NOR (MTYP [1: 0] = 0x10 в регистре FMC_BCRx).

Управление WAIT при асинхронном доступе

Если асинхронная память утверждает сигнал WAIT, чтобы указать, что он еще не готов принять или предоставить данные, бит ASYNCWAIT должен быть установлен в регистре FMC_BCRx.

Если сигнал WAIT активен (высокий или низкий в зависимости от бит WAITPOL), вторая фаза доступа (фаза установки данных), запрограммированная битами DATAST, продлевается до тех пор, пока WAIT не станет активным. В отличие от фазы настройки данных, первые фазы доступа (этапы настройки адреса и фазы удержания адреса), запрограммированные битами ADDSET [3: 0] и ADDHLD, не чувствительны к WAIT и поэтому не продлеваются.

Фаза установки данных должна быть запрограммирована так, чтобы WAIT можно было обнаружить 4 цикла HCLK до завершения транзакции памяти. Необходимо учитывать следующие случаи:

1. Память соответствует сигналу WAIT, соответствующему NOE / NWE, который переключает: DATAST ≥ (4•HCLK) + max_wait_assertion_time

2. Память утверждает, что сигнал WAIT соответствует NEX (или NOE / NWE не переключается):

если: max_wait_assertion_time > address_phase + hold_phase

тогда:

DATAST ≥ (4•HCLK) + (max_wait_assertion_time – address_phase – hold_phase)

в противном случае DATAST ≥ 4•HCLK

где max_wait_assertion_time - это максимальное время, которое занимает память для подтверждения сигнала WAIT, когда NEX / NOE / NWE является низким.

На рис. 469 и рис. 470 показано количество тактовых циклов HCLK, которые добавляются в фазу доступа к памяти после того, как WAIT освобождается асинхронной памятью (независимо от вышеуказанных случаев).

37.5.5 Синхронные транзакции

Часы памяти, FMC_CLK, являются субпопуляцией HCLK. Это зависит от значения CLKDIV и размера данных MWID / AHB, следуя приведенной ниже формуле:

FMC_CLK divider ratio = max(CLKDIV + 1,MWID(AHB data size))

Если MWID составляет 16 или 8 бит, отношение делителя FMC_CLK всегда определяется запрограммированным значением CLKDIV.

Если MWID составляет 32 бита, отношение делителя FMC_CLK зависит также от размера данных AHB.

Пример:

•Если CLKDIV = 1, MWID = 32 бит, размер данных AHB = 8 бит, FMC_CLK = HCLK / 4.

•Если CLKDIV = 1, MWID = 16 бит, размер данных AHB = 8 бит, FMC_CLK = HCLK / 2.

Флэш-память NOR определяет минимальное время от утверждения NADV до максимума CLK. Для удовлетворения этого ограничения FMC не выдает часы в память в течение первого внутреннего тактового цикла синхронного доступа (до утверждения NADV). Это гарантирует, что нарастающий фронт тактовых импульсов памяти происходит в середине низкого импульса NADV.

Латентность данных по сравнению с задержкой памяти NOR

Задержка данных - это количество циклов, ожидающих до выборки данных. Значение DATLAT должно соответствовать значению задержки, указанному в регистре конфигурации NOR Flash. FMC не включает тактовый цикл, когда NADV является низким в подсчете времени ожидания данных.

Осторожно: некоторые флэш-память NOR включают в себя цикл NADV Low в подсчете латентности данных, так что точное соотношение между латентностью NOR Flash и параметром FMC DATLAT может быть:

- NOR Flash latency = (DATLAT + 2) CLK тактовые циклы
- или NOR Задержка вспышки = (DATLAT + 3) Часовые циклы CLK

Некоторые недавние воспоминания утверждают NWAIT во время фазы ожидания. В таких случаях DATLAT можно установить на минимальное значение. В результате FMC сэмплирует данные и ждет достаточно долго, чтобы оценить, действительно ли данные. Таким образом, FMC обнаруживает, когда память выходит из задержек и обрабатываются реальные данные.

Другие воспоминания не утверждают NWAIT во время латентности. В этом случае задержка должна быть правильно установлена как для FMC, так и для памяти, в противном случае неверные данные ошибочно принимаются за хорошие данные, или действительные данные теряются на начальной фазе доступа к памяти.

Однократная передача

Когда выбранный банк настроен в режиме пакетной передачи для синхронных обращений, если, например, в 16-разрядной памяти запрашивается одноразовая транзакция AHB, FMC выполняет пакетную транзакцию длиной 1 (если передача AHB равна 16 бит) или длина 2 (если передача AHB составляет 32 бита) и отменять сигнал выбора микросхемы, когда последние данные стробируются.

Такие передачи не являются наиболее эффективными с точки зрения циклов по сравнению с асинхронными операциями чтения. Тем не менее, для случайного

асинхронного доступа сначала потребуется перепрограммировать режим доступа к памяти, который будет длиться дольше.

Перекрестная страница для Cellular RAM 1.5

Сотовая оперативная память 1.5 не позволяет пакетному доступу пересекать границу страницы. Контроллер FMC позволяет автоматически разбить пакетный доступ, когда достигнут размер страницы памяти, путем настройки битов CPSIZE в регистре FMC_BCR1 после размера страницы памяти.

Управление ожиданиями

Для синхронных флэш-памяти NOR NWAIT оценивается после запрограммированного периода ожидания, что соответствует тактовым циклам (DATLAT + 2) CLK.

Если NWAIT активен (низкий уровень, когда WAITPOL = 0, высокий уровень, когда WAITPOL = 1), состояния ожидания вставляются до тех пор, пока NWAIT не будет активен (высокий уровень, когда WAITPOL = 0, низкий уровень, когда WAITPOL = 1).

Когда NWAIT неактивен, данные считаются действительными либо сразу (бит WAITCFG = 1), либо на следующем фронте такта (бит WAITCFG = 0).

Во время вставки состояния ожидания через сигнал NWAIT контроллер продолжает посылать импульсы синхронизации в память, сохраняя допустимые сигналы разрешения выбора и выхода. Он не считает данные действительными.

В режиме пакетной передачи для NOR-сигнала NWAIT есть две конфигурации синхронизации:

- Флэш-память подает сигнал NWAIT на один цикл данных до состояния ожидания (по умолчанию после сброса).
- Флэш-память утверждает сигнал NWAIT во время состояния ожидания

FMC поддерживает обе конфигурации состояния ожидания NOR Flash для каждого Chip Select, благодаря бит WAITCFG в регистрах FMC_BCRx (x = 0..3).

2.2 Организация памяти

Программная память, память данных, регистры и порты ввода / вывода организованы в пределах одного линейного адресного пространства 4 Гбайта.

Байты закодированы в памяти в маленьком концевом формате. Самый младший байт в слове считается наименее значимым байтом слова и наивысшим номером байта, это слово наиболее значимо.

Подробное сопоставление периферийных регистров приведено в соответствующих главах.

Адресное пространство памяти разделено на 8 основных блоков, каждый из которых составляет 512 МБ.

Все области памяти, которые не выделены для встроенных памяти и периферийных устройств, считаются «зарезервированными»). См. Рисунок карты памяти в техническом описании продукта.

2.3.1 Встроенная SRAM

STM32F405xx / 07xx и STM32F415xx / 17xx имеют 4 Кбайт резервного SRAM (см. Раздел 5.1.2: Аккумуляторный резервный домен) плюс 192 Кбайт системной SRAM.

STM32F42xxx и STM32F43xxx имеют 4 Кбайт резервного SRAM (см. Раздел 5.1.2: Аккумуляторный резервный домен) плюс 256 Кбайт системной SRAM.

Доступ к встроенной SRAM можно получить в виде байтов, полусловов (16 бит) или полных слов (32 бита).

Операции чтения и записи выполняются со скоростью процессора с 0 состояния ожидания. Встроенная SRAM разделена на три блока:

- SRAM1 и SRAM2 отображаются по адресу 0x2000 0000 и доступны для всех мастеров AHB.
- SRAM3 (доступен на STM32F42xxx и STM32F43xxx), отображаемый по адресу 0x2002 0000 и доступный для всех мастеров AHB.
- CCM (память с сердечником), отображаемая по адресу 0x1000 0000 и доступная только процессору через D-шину.

Мастера AHB поддерживают одновременный доступ к SRAM (от Ethernet или USB OTG HS): например, MAC-адрес Ethernet может считывать / записывать из / в SRAM2, в то время как CPU считывает / записывает с / на SRAM1 или SRAM3.

CPU может обращаться к SRAM1, SRAM2 и SRAM3 через системную шину или через шины I-Code / D-Code при выборе загрузки из SRAM или при выборе физического переназначения (раздел 9.2.1: регистр переназначения памяти SYSCFG (SYSCFG_MEMRMP) в контроллере SYSCFG). Чтобы получить максимальную производительность при выполнении SRAM, необходимо выбрать физический переназначение (загрузка или выбор программного обеспечения).

2.3.2 Обзор флэш-памяти

Интерфейс флэш-памяти управляет I-Code AHB и D-кодами доступа к флэш-памяти. Он реализует операции стирания и программной флэш-памяти и механизмы защиты чтения и записи. Это ускоряет выполнение кода с помощью системы предварительной выборки команд и строк кэша.

Флэш-память организована следующим образом:

• Основной блок памяти разделен на сектора.

• Системная память, с которой устройство загружается в режиме загрузки системной памяти

• 512 OTP (одноразовые программируемые) байты для пользовательских данных.

• Общие байты для настройки защиты чтения и записи, уровня BOR, программного обеспечения / аппаратного обеспечения сторожевого таймера и сброса, когда устройство находится в режиме ожидания или остановки.

Дополнительную информацию см. В разделе 3: Интерфейс встроенной флэш-памяти.

2.3.3 Бит-полоса

Карта Cortex®-M4 с памятью FPU включает в себя две области бит-диапазона. Эти регионы отображают каждое слово в области псевдонимов памяти бит в области бит-диапазона памяти. Запись в слово в области псевдонимов имеет тот же эффект, что и операция чтения-изменения-записи на целевом бите в области бит-диапазона.

В устройствах STM32F4xx как периферийные регистры, так и SRAM отображаются в область битовой области, так что допускаются операции записи и чтения по одной бит-полосе. Операции доступны только для Cortex®-M4 с доступом FPU, а не от других мастеров шины (например, DMA).

Формула отображения показывает, как привязывать каждое слово в области псевдонимов к соответствующему биту в области бит-диапазона. Формула отображения: bit_word_addr = bit_band_base + (byte_offset x 32) + (bit_number x 4)

где:

- bit_word_addr - это адрес слова в области памяти псевдонимов, который сопоставляется с целевым битом

- bit_band_base - это начальный адрес области псевдонимов

- byte_offset - номер байта в области бит-диапазона, который содержит целевой бит

- bit_number - битная позиция (0-7) целевого бита

пример

В следующем примере показано, как сопоставить бит 2 байта, расположенного по адресу SRAM 0x20000300, в область псевдонимов:

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

Запись на адрес 0x22006008 имеет тот же эффект, что и операция чтения-изменения-записи в бит 2 байта по адресу SRAM 0x20000300.

Адрес чтения 0x22006008 возвращает значение (0x01 или 0x00) бит 2 байта на адрес SRAM 0x20000300 (0x01: бит установлен, 0x00: бит сброс).

Для получения дополнительной информации о бит-диапазонах, пожалуйста, обратитесь к Cortex®-M4 с руководством по программированию FPU (см. Соответствующие документы на стр. 1).

3 Встроенный интерфейс флэш-памяти

3.1 Введение

Интерфейс флэш-памяти управляет I-Code AHB и D-кодами доступа к флэш-памяти. Он реализует операции стирания и программной флэш-памяти и механизмы защиты чтения и записи.

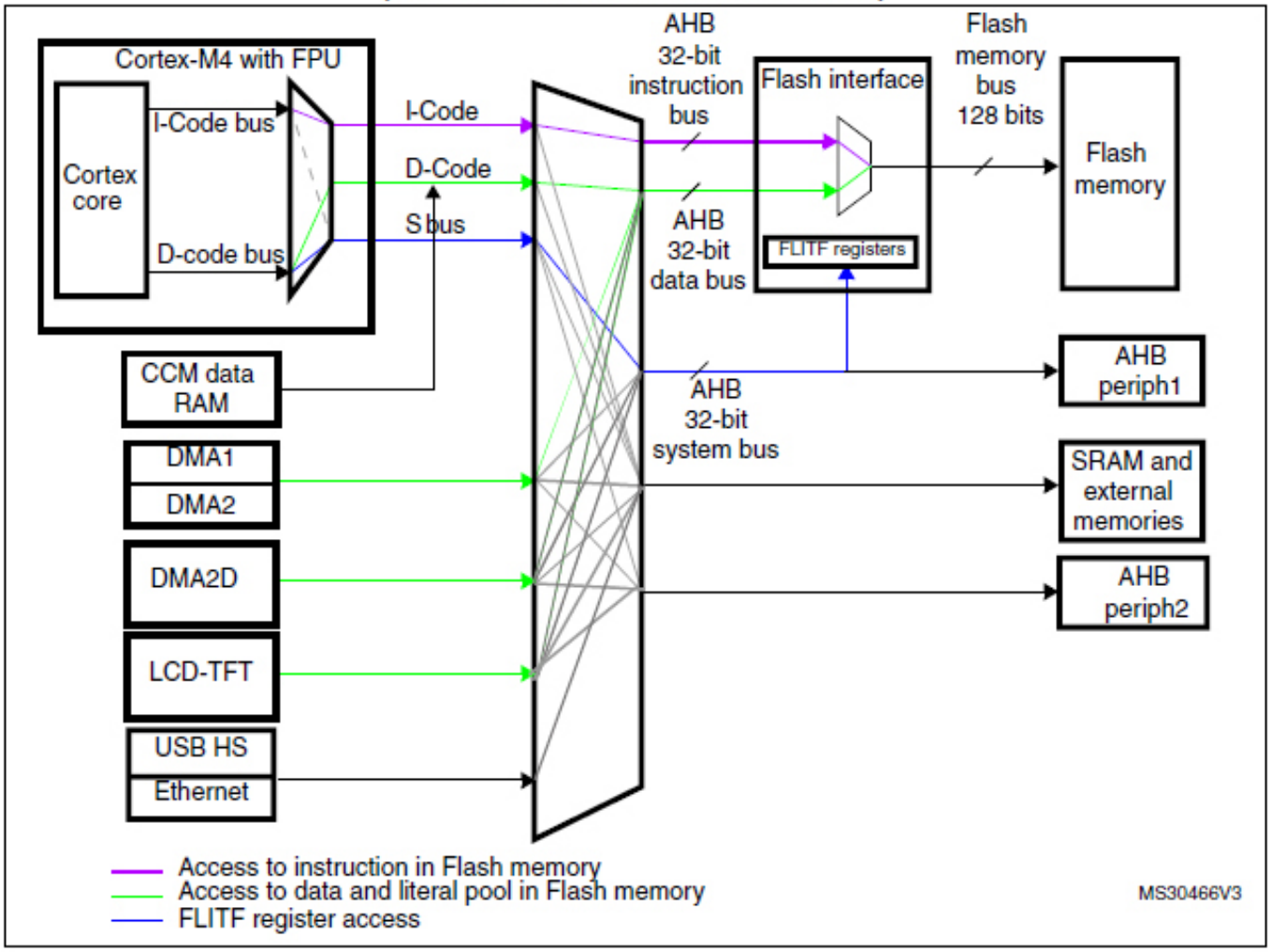
Интерфейс флэш-памяти ускоряет выполнение кода с помощью системы предварительной выборки команд и строк кэша.

3.2 Основные функции

- операции чтения флэш-памяти
- Программа памяти / операции стирания памяти
- Защита от записи / записи
- Подробление по I-коду
- 64 строки кеша 128 бит на I-Code
- 8 строк кэша из 128 бит в D-коде

На рисунке 3 показано соединение интерфейса флэш-памяти внутри архитектуры системы.

Figure 4. Flash memory interface connection inside system architecture (STM32F42xxx and STM32F43xxx)



3.4 Встроенная флэш-память в STM32F42xxx и STM32F43xxx

Флэш-память имеет следующие основные функции:

- Емкость до 2 Мбайт с двойной банковской архитектурой, поддерживающей возможность чтения и записи (RWW)

Чтение данных шириной 12 бит

• Byte, полуслое, слово и двойное слово write

«Сектор, банк и массовое уничтожение (оба банка)

• Организация банковской памяти

Флэш-память организована следующим образом:

- для каждого банка основной блок памяти (1 Мбайт) разделен на 4 сектора 16 Кбайт, 1 сектор 64 Кбайт и 7 секторов 128 Кбайт

- Системная память, с которой устройство загружается в режиме загрузки системной памяти

- 512 байтов OTP (одноразовые программируемые) для пользовательских данных

Область OTP содержит 16 дополнительных байтов, используемых для блокировки соответствующего блока данных OTP.

- байты опций для настройки защиты чтения и записи, уровня BOR, сторожевого таймера, режима двойной банковской загрузки, функции двойного банка, программного обеспечения / оборудования и сброса, когда устройство находится в режиме ожидания или остановки.

• Учетная банковская организация на устройствах 1 Мбайт

Функция двойного банка на устройствах 1 Мбайт активируется путем установки бита опции DB1M.

Для получения двойной банковской флэш-памяти последние 512 Кбайт единого банка (сектора [8:11]) реструктурируются так же, как и первые 512 Кбайт.

Нумерация сектора организации с двойной банковской памятью отличается от единого банка: единичная банковская память содержит 12 секторов, тогда как двойная банковская память содержит 16 секторов (см. Таблицу 7: 1 Мбайт флеш-памяти одного банка против двойного банка

(STM32F42xxx и STM32F43xxx)).

Для операции стирания нумерация нужного сектора должна учитываться в соответствии с битком DB1M.

- Когда бит DB1M сбрасывается, операция стирания должна выполняться по номеру сектора по умолчанию.

- Когда бит DB1M установлен, чтобы выполнить операцию стирания на банке 2, номер сектора должен быть запрограммирован (номер сектора от 12 до 19). Обратитесь к регистру FLASH_CR для конфигурации SNB (номер сектора).

См. Таблицу 8: 1 Мбайт однобанковской организации флэш-памяти (STM32F42xxx и STM32F43xxx) и Таблица 9: 1-мегабайтная организация с двойной банковской флэш-памятью (STM32F42xxx и STM32F43xxx) для получения подробной информации о единичных банках 1 Мбайт и 1 Мбайт в двух банках.

3.5 Интерфейс чтения

3.5.1. Связь между частотой тактовой частоты процессора и временем считывания флэш-памяти

Чтобы правильно считывать данные из флэш-памяти, количество состояний ожидания (LATENCY) должно быть правильно запрограммировано в регистре управления доступом к вспышке (FLASH_ACR) в соответствии с частотой тактовых импульсов CPU (HCLK) и напряжением питания устройства.

Буфер предварительной выборки должен быть отключен, когда напряжение питания ниже 2,1 В. Соответствие между состояниями ожидания и тактовой частотой процессора приведено в таблице 10 и таблице 11.

Примечание. На устройствах STM32F405xx / 07xx и STM32F415xx / 17xx:

- когда VOS = '0', максимальное значение fHCLK = 144 МГц.
- когда VOS = '1', максимальное значение fHCLK = 168 МГц.

На устройствах STM32F42xxx и STM32F43xxx:

- когда VOS [1: 0] = '0x01', максимальное значение fHCLK равно 120 МГц.
- когда VOS [1: 0] = '0x10', максимальное значение fHCLK равно 144 МГц. Его можно расширить до 168 МГц, активировав режим перегрузки.
- когда VOS [1: 0] = '0x11, максимальное значение fHCLK составляет 168 МГц. Его можно расширить до 180 МГц, активируя режим перегрузки.
- Режим перегрузки недоступен, если VDD колеблется от 1,8 до 2,1 В.

См. Раздел 5.1.4: Регулятор напряжения для STM32F42xxx и STM32F43xxx для получения подробной информации о том, как активировать режим перегрузки.

После сброса тактовая частота процессора составляет 16 МГц, а состояние ожидания (WS) настроено в регистре FLASH_ACR.

Настоятельно рекомендуется использовать следующие программные последовательности для настройки количества состояний ожидания, необходимых для доступа к флэш-памяти с частотой процессора.

Увеличение частоты процессора

1. Запрограммировать новое количество состояний ожидания на бит LATENCY в регистре FLASH_ACR
2. Убедитесь, что новый номер состояний ожидания принят во внимание, чтобы получить доступ к флэш-памяти, прочитав регистр FLASH_ACR
3. Измените источник синхронизации ЦП, записав бит SW в регистре RCC_CFGR
4. При необходимости измените предварительный делитель частоты процессора, записав бит HPRE в RCC_CFGR
5. Убедитесь, что новый источник синхронизации ЦПУ или / и новое значение предварительного делителя ЦП принимается во внимание, считывая состояние источника синхронизации (бит SWS) или / и значение предварительного делителя AHB (бит HPRE), соответственно, в Регистр RCC_CFGR.

Уменьшение частоты процессора

1. Измените источник синхронизации ЦП, записав бит SW в регистре RCC_CFGR

2. При необходимости измените предварительный делитель частоты процессора, записав бит HPRE в RCC_CFGR

3. Убедитесь, что новый источник синхронизации ЦПУ или / и новое значение предварительного делителя ЦП / учитываются путем считывания состояния источника синхронизации (бит SWS) или / и значения предварительного делителя AHB (бит HPRE), соответственно, в Регистр RCC_CFGR

4. Запрограммируйте новое количество состояний ожидания на бит LATENCY в FLASH_ACR

5. Убедитесь, что новый номер состояний ожидания используется для доступа к флэш-памяти, считывая регистр FLASH_ACR

Примечание. *Изменение конфигурации часов процессора или конфигурации состояния ожидания (WS) может быть неэффективным сразу. Чтобы удостовериться, что текущая тактовая частота процессора - та, которую вы настроили, вы можете проверить коэффициент предварительного делителя AHB и значения состояния источника синхронизации. Чтобы убедиться, что количество запрограммированных WS действительно, вы можете прочитать регистр FLASH_ ACR.*

3.5.2 Адаптивный ускоритель памяти в реальном времени (ART Accelerator ™)

Запатентованный адаптивный ускоритель памяти реального времени (APT) оптимизирован для промышленного стандарта STM32 ARM® Cortex®-M4 с процессорами FPU. Он уравновешивает присущие производительности ARM® Cortex®-M4 с технологией FPU по технологии флэш-памяти, что обычно требует, чтобы процессор дождался флэш-памяти на более высоких рабочих частотах.

Чтобы выпустить полную производительность процессора, ускоритель реализует очередь предварительной выборки команд и кеш филиалов, что увеличивает скорость выполнения программы из 128-битной флэш-памяти. На основе теста CoreMark производительность, достигаемая благодаря ускорителю ART, эквивалентна выполнению программы состояния ожидания 0 из флэш-памяти с частотой процессора до 180 МГц.

Предварительная выборка инструкций

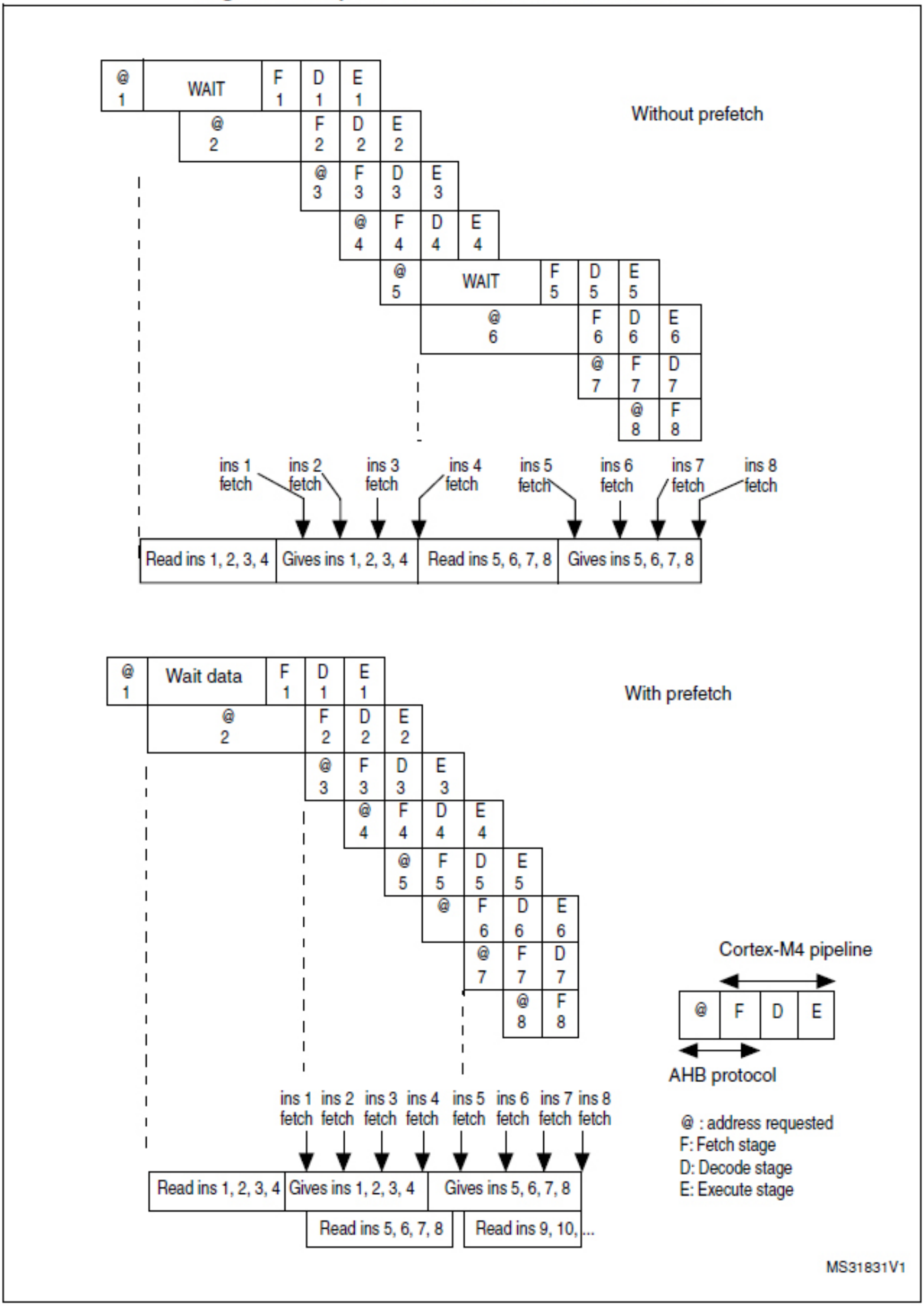
Каждая операция чтения флэш-памяти обеспечивает 128 бит из четырех команд из 32 бит или 8 инструкций по 16 бит в соответствии с запущенной программой. Таким образом, в случае последовательного кода требуется, по меньшей мере, четыре цикла ЦП для выполнения предыдущей строки инструкции чтения.

Предварительная выборка на шине I-Code может использоваться для чтения следующей последовательной строки команд из флэш-памяти, когда CPU запрашивает текущую строку команд. Предварительная выборка активируется путем установки бита PRFTEN в регистре FLASH_ACR. Эта функция полезна, если для доступа к флэш-памяти требуется хотя бы одно состояние ожидания.

На рисунке 5 показано выполнение последовательных 32-разрядных команд с предварительной выборкой и без нее, когда для доступа к флэш-памяти требуется 3 WS

Если код не является последовательным (ветвь), эта команда может отсутствовать в текущей используемой строке команд или в предварительно заданной строке инструкции. В этом случае (промах) штраф в терминах количества циклов по крайней мере равен количеству состояний ожидания.

Figure 5. Sequential 32-bit instruction execution



Кэш-память инструкций

Чтобы ограничить время, потерянное из-за переходов, можно сохранить 64 строки из 128 бит в кэше команд. Эту функцию можно включить, установив бит разрешения кэша команд (ICEN) в регистр FLASH_ACR. Каждый раз, когда происходит промах (запрошенные данные не присутствуют в текущей используемой строке команд, в предварительно заданной строке инструкции или в кэше команд), считывание строки копируется в кэш-память команды. Если некоторые данные, содержащиеся в кэш-памяти команд, запрашиваются ЦП, они предоставляются без вставки какой-либо задержки. После того, как все строки кэша кэша команд заполнены, политика LRU (наименее используемая) используется для определения строки для замены в кеше памяти команд. Эта особенность особенно полезна в случае кода, содержащего циклы.

Управление данными

Литеральные пулы извлекаются из флэш-памяти через шину D-кода во время этапа выполнения конвейера CPU. Следовательно, конвейер CPU останавливается до тех пор, пока не будет предоставлен запрошенный литеральный пул. Чтобы ограничить время, утерянное из-за литеральных пулов, доступ через D-код DATA-кода AHB имеет приоритет над доступом через I-Code I-Code шины AHB.

Если используются некоторые литеральные пулы, кэш-память данных может быть активирована путем установки бит разрешения кэша данных (DCEN) в регистре FLASH_ACR. Эта функция работает как память кэша команд, но размер сохраненных данных ограничен 8 рядами по 128 бит.

Примечание. *Данные в области конфигурации пользователя не могут быть кэшируемыми.*

3.6 Стирание и программные операции

Для любой работы программы флэш-памяти (стирание или программа) тактовая частота процессора (HCLK) должна быть не менее 1 МГц. Содержимое флэш-памяти не гарантируется, если сброс устройства происходит во время операции флэш-памяти.

Любая попытка прочитать флэш-память на STM32F4xx во время ее записи или стирания приводит к остановке шины. Операции чтения обрабатываются правильно после завершения работы программы. Это означает, что выбор кода или данных не может выполняться во время операции записи / стирания.

На устройствах STM32F42xxx и STM32F43xxx доступны два банка, позволяющие выполнять операцию чтения из одного банка, в то время как операция записи / стирания выполняется в другом банке.

3.6.1 Разблокировка регистра управления вспышкой

После сброса запись в регистре управления Flash (FLASH_CR) не допускается, чтобы защитить флэш-память от возможных нежелательных операций, вызванных, например, электрическими помехами. Для разблокировки этого регистра используется следующая последовательность:

1. Запишите KEY1 = 0x45670123 в регистр ключа Flash (FLASH_KEYR)
2. Напишите KEY2 = 0xCDEF89AB в регистре флэш-ключа (FLASH_KEYR)

Любая неправильная последовательность вернет ошибку шины и заблокирует регистр FLASH_CR до следующего сброса.

Регистр FLASH_CR может быть снова заблокирован программным обеспечением, установив бит LOCK в регистр FLASH_CR.

Примечание. *Регистр FLASH_CR недоступен в режиме записи, когда бит BSY в регистре FLASH_SR установлен. Любая попытка записать его с установленным битом BSY приведет к остановке шины AHB до тех пор, пока бит BSY не будет очищен.*

3.6.2 Программный / стирающий параллелизм

Размер параллелизма настраивается через поле PSIZE в регистре FLASH_CR. Он представляет собой количество байтов, которые должны быть запрограммированы каждый раз при выполнении операции записи во флэш-память. PSIZE ограничивается напряжением питания и зависит от того, используется ли внешний источник VPP или нет. Поэтому он должен быть правильно настроен в регистре FLASH_CR до любой операции программирования / стирания.

Операция стирания флэш-памяти может выполняться только сектором, банком или всей флэш-памятью (стирание массы). Время стирания зависит от запрограммированного значения PSIZE. Более подробную информацию о времени стирания см. В разделе электрических характеристик в техническом описании устройства.

В таблице 12 приведены правильные значения PSIZE.

Table 12. Program/erase parallelism

	Voltage range 2.7 - 3.6 V with External V_{PP}	Voltage range 2.7 - 3.6 V	Voltage range 2.4 - 2.7 V	Voltage range 2.1 - 2.4 V	Voltage range 1.8 V - 2.1 V
Parallelism size	x64	x32	x16		x8
PSIZE(1:0)	11	10	01		00

Примечание. *Любая программа или операция стирания, начатая с несовместимых параметров параллелизма программ / напряжения, может привести к непредсказуемым результатам. Даже если последующая операция чтения указывает, что логическое значение было эффективно записано в память, это значение может не сохраниться.*

Для использования VPP внешнее высоковольтное питание (от 8 до 9 В) должно быть применено к пэду VPP. Внешнее питание должно поддерживать этот диапазон напряжения, даже если потребление постоянного тока превышает 10 мА. Рекомендуется ограничить использование VPP начальным программированием на заводской линии. Поставка VPP не должна применяться более часа, иначе флэш-память может быть повреждена.

3.6.3 Стереть

Операция стирания флэш-памяти может выполняться на уровне сектора или на всей флэш-памяти (Mass Erase). Mass Erase не влияет на сектор OTP или сектор конфигурации.

Стереть сектор

Чтобы удалить сектор, выполните следующие действия:

1. Убедитесь, что операция флэш-памяти не выполняется, проверяя бит BSY в регистре FLASH_SR

2. Установите бит SER и выберите сектор из 12 секторов (для STM32F405xx / 07xx и STM32F415xx / 17xx) и из 24 (для STM32F42xxx и STM32F43xxx) в основном блоке памяти, который вы хотите стереть (SNB) в FLASH_CR регистр

3. Установите бит STRT в регистре FLASH_CR
4. Подождите, пока бит BSY будет очищен

Удаление банка в устройствах STM32F42xxx и STM32F43xxx

Чтобы удалить банк 1 или банк 2, выполните следующие действия:

1. Убедитесь, что операция флэш-памяти не выполняется, проверяя бит BSY в регистре FLASH_SR
2. Установите бит MER или MER1 соответственно в регистр FLASH_CR
3. Установите бит STRT в регистре FLASH_CR
4. Подождите, пока бит BSY не будет сброшен.

Массовое стирание

Чтобы выполнить Mass Erase, рекомендуется следующая последовательность:

1. Убедитесь, что операция флэш-памяти не выполняется, проверяя бит BSY в регистре FLASH_SR
2. Установите бит MER в регистр FLASH_CR (на устройствах STM32F405xx / 07xx и STM32F415xx / 17xx)
3. Установите биты MER и MER1 в регистр FLASH_CR (на устройствах STM32F42xxx и STM32F43xxx).
4. Установите бит STRT в регистре FLASH_CR
5. Подождите, пока бит BSY не будет очищен

Примечание. *Если биты MERx и SER установлены в регистре FLASH_CR, выполняется стирание массы.*

Если оба бита MERx и SER сбрасываются и бит STRT установлен, непредсказуемое поведение может происходить без генерирования какого-либо флага ошибки. Это условие должно быть запрещено.

3.6.4 Программирование

Стандартное программирование

Последовательность программирования Flash-памяти следующая:

1. Убедитесь, что основная операция флэш-памяти не выполняется, проверяя бит BSY в регистре FLASH_SR.
2. Установите бит PG в регистр FLASH_CR
3. Выполните операции записи данных на требуемый адрес памяти (внутри основного блока памяти или области OTP):
   - Доступ к байтам в случае x8-параллелизма
   - Доступ к полуслову в случае параллелизма x16
   - Доступ к Word в случае параллелизма x32
   - Доступ к двойному словарю в случае параллелизма x64
4. Подождите, пока бит BSY будет очищен.

Примечание. *Последовательные операции записи возможны без необходимости операции стирания при изменении битов от «1» до «0». Для записи «1» требуется операция стирания флэш-памяти.*

Если одновременно запрашивается операция стирания и программы, сначала выполняется операция стирания.

Ошибки программирования

Невозможно запрограммировать данные во флэш-память, которая пересечет границу 128-битной строки. В этом случае операция записи не выполняется, и в регистре FLASH_SR устанавливается флаг ошибки выравнивания программы (PGAERR).

Тип доступа к записи (байт, полуслов, слово или двойное слово) должен соответствовать выбранному типу параллелизма (x8, x16, x32 или x64). Если нет, операция записи не выполняется, и в регистре FLASH_SR установлен флаг ошибки параллелизма программ (PGPERR).

Если стандартная последовательность программирования не соблюдается (например, если есть попытка записать на адрес флэш-памяти, когда бит PG не установлен), операция прерывается и флаг ошибки последовательности программ (PGSERR) устанавливается в Регистр FLASH_SR.

Программирование и кэширование

Если доступ к записи на флэш-памяти относится к некоторым данным в кеше данных, доступ к записи на флэш-памяти изменяет данные во флэш-памяти и данные в кеше.

Если операция стирания во флэш-памяти также относится к данным в кеше данных или команд, вы должны убедиться, что эти данные перезаписаны до того, как они будут доступны во время выполнения кода. Если это невозможно сделать безопасно, рекомендуется сбросить кеши, установив биты DCRST и ICRST в регистр FLASH_CR. Примечание. Кэш ввода-вывода следует очищать только тогда, когда он отключен (I / DCEN = 0).

3.6.5 Read-while-write (RWW)

В устройствах STM32F42xxx и STM32F43xxx флэш-память делится на два банка, позволяющие выполнять операции чтения и записи. Эта функция позволяет выполнять операцию чтения из одного банка, в то время как операция удаления или программы выполняется в другом банке.

Примечание. *Операции Write-while-write не разрешены. В качестве примера невозможно выполнить операцию стирания на одном банке при программировании другого.*

Чтение из банка 1 при удалении банка 2

Выполняя программный код из банка 1, можно выполнить операцию стирания на банке 2 (и наоборот). Выполните следующую процедуру:

1. Убедитесь, что операция флэш-памяти не выполняется, проверяя бит BSY в регистре FLASH_SR (BSY активен, когда операция стирания / программы переходит в банк 1 или банк 2)
2. Установите бит MER или MER1 в регистр FLASH_CR
3. Установите бит STRT в регистре FLASH_CR
4. Подождите, пока бит BSY будет сброшен (или используйте прерывание EOP).

Чтение из банка 1 при программировании банка 2

Выполняя программный код (по шине I-кода) из банка 1, можно выполнить программную операцию с банком 2 (и наоборот). Выполните следующую процедуру:

1. Убедитесь, что операция флэш-памяти не выполняется, проверяя бит BSY в регистре FLASH_SR (BSY активен, когда операция стирания / программы продолжается в банке 1 или в банке 2)
2. Установите бит PG в регистр FLASH_CR
3. Выполните операции (операции) записи данных на нужный адрес памяти внутри основного блока памяти или области OTP
4. Подождите, пока бит BSY не будет сброшен.

3.6.6 Прерывания

Установка бита разрешения прерывания завершения (EOPIE) в регистре FLASH_CR позволяет генерировать прерывание при завершении операции стирания или программы, то есть когда бит занятости (BSY) в регистре FLASH_SR очищается (операция выполнена правильно или нет). В этом случае устанавливается бит окончания операции (EOP) в регистре FLASH_SR.

Если во время программы возникает ошибка, в регистре FLASH_SR устанавливается запрос стирания или запрос операции чтения, один из следующих флагов ошибки:

- PGAERR, PGPERR, PGSERR (флаги программной ошибки)
- WRPERR (флаг ошибки защиты)
- RDERR (флаг ошибки защиты) для устройств STM32F42xxx и STM32F43xxx.

В этом случае, если бит разрешения прерывания ошибки (ERRIE) установлен в регистре FLASH_CR, генерируется прерывание и бит операции (OPERR) устанавливается в регистре FLASH_SR.

Примечание. *Если обнаружено несколько последовательных ошибок (например, в случае передачи DMA во флэш-память), флаги ошибок не могут быть удалены до конца последовательных запросов на запись.*

64-Мбит SDRAM (1 Мбит x 16 бит x 4-банка)

64-Мбит SDRAM представляет собой высокоскоростную CMOS, динамическую память с произвольным доступом, предназначенную для работы в системах памяти 3,3 В, содержащих 67108,864 бит. Он внутренне сконфигурирован как четырехбалочная DRAM с синхронным интерфейсом. Каждый 16,777,216-бит банк организован как 4096 строк на 256 столбцов на 16 бит. 64-Мбит SDRAM включает в себя автоматическое обновление, энергосбережение и режим отключения питания. Все сигналы регистрируются на положительном фронте тактового сигнала CLK.

STM32F429ZIT6 считывает и записывает данные на частоте 80 МГц.

NOR Flash Memory

Память NOR, названная так в честь особой разметки данных (Not OR – логическое Не-ИЛИ), является высокоскоростной памятью Flash. Память NOR предоставляет возможность высокоскоростного, случайного доступа к информации, и обладает способностью записывать и считывать данные в определенном месте без необходимости обращаться к памяти последовательно. В отличие от NAND памяти, память NOR позволяет обращаться к данным размером до одного байта. Технология NOR выигрывает в ситуациях, когда данные случайным образом записываются или читаются. Поэтому NOR чаще всего встраивают в сотовые телефоны (для хранения операционной системы) и планшеты, а также используется в компьютерах для хранения BIOS.

NAND Flash Memory

NAND память была изобретена после NOR, и также названа в честь особой разметки данных (Not AND – логическое Не-И). NAND память записывает и считывает данные с высокой скоростью, в режиме последовательного чтения,

упорядочивая данные в небольшие блоки (страницы). Память NAND может считывать и записывать информацию постранично, однако не может обращаться к конкретному байту, как NOR. Поэтому NAND обычно используют в твердотельных накопителях (SSD), аудио и видео проигрывателях, телевизионных приставках, цифровых камеры, мобильных телефонах (для хранения пользовательской информации) и других устройствах, в которых данные, как правило, записываются последовательно.

Например, большинство цифровых камер используют память на основе технологии NAND, так как изображения снимаются и записываются последовательно. Технология NAND является более эффективной еще и при чтении, так как она способна передавать целые страницы данных очень быстро. Как последовательная память, NAND идеальна для хранения данных. Цена на SSD с использованием NAND меньше, чем с NOR, а микросхемы NAND обладают большей плотностью информации на матрицу. Память, которая хранит один бит на ячейку, называется Single-Level Cell (SLC) Flash.

PSRAM

Когда необходима ёмкость более 16 Мбит, одним из решений может быть ЗУ псевдо-SRAM – PSRAM. Это ЗУ с ядром DRAM и интерфейсом, подобным SRAM. Одна запоминающая ячейка ядра DRAM состоит из одного транзистора и одного конденсатора.

Как уже упоминалось, техника DRAM обеспечивает большую ёмкость и меньшую цену одного бита, однако требуется периодическое обновление (регенерация) ячеек. В то время как в традиционных DRAM управление обновлением находится вне памяти, в PSRAM оно интегрировано в само устройство. Соответственно, можно использовать PSRAM для того, чтобы надстроить ёмкость других асинхронных SRAM.

SRAM и SDRAM

SRAM (Static Random Access Memory) — это тип памяти, который не требует частого обновления (прим.: для этого типа памяти вообще не требуются циклы перезаписи). Это означает, что, прочитав какую-либо область памяти, нет необходимости перезаписывать данные обратно в эту же область каждый раз, поэтому память и называется статичной. В то время как SDRAM (Synchronous Dynamic Random Access Memory) — это тип памяти, который требует регулярных обновлений данных и имеет синхронный интерфейс. Это означает, что для запроса\ответа необходима опорная (синхронная) частота от микропроцессора (микроконтроллера), и тогда память будет синхронна с системной шиной.

Поскольку SRAM не требует циклов обновления, скорость работы выше, чем в SDRAM, в которой скорость также зависит и от синхронности работы. Время доступа включает в себя задержку и время передачи. Задержка — это сумма времени синхронизации сигналов и обновления запрошенных данных (прим.: на чтение или запись). Тем не менее наиболее применяема SRAM из-за её простоты интерфейса. Для неё нет необходимости делать циклы перезаписи и шины адреса и данных доступны напрямую (прим.: без тактового сигнала).

А что насчёт объёма памяти? В основе SRAM используется тип памяти, называемый flip-flop. Этот тип памяти может хранить своё значение сколь угодно долго, пока есть питание. В основе же SDRAM лежит тип памяти, называемый емкостная память, который требует периодического обновления. Хоть в памяти типа flip-flop содержится всего несколько транзисторов, это занимает значительно больше места, чем конденсатор. Чип DRAM может содержать несколько гигабит, в то время как чип SDRAM может содержать только несколько десятков мегабит.

Ещё одна особенность, которую необходимо учитывать, — ток потребления. Учитывая, что SDRAM требует периодического обновления данных, она получает заряд каждые несколько наносекунд. Поэтому она потребляет ток ощутимо. Окружающая температура также влияет на потребление памяти. Потребление у SRAM стабильно в температурном диапазоне от -55 °C до +125 °C. Чего нельзя сказать про SDRAM и другие виды памяти типа DRAM, где высокая частота обновления в сочетании с высокими температурами требует значительно большего потребления тока, даже если нет обращения к памяти.

Напоследок, все мы знаем правила в мире технологий. Чем быстрее и проще, тем дороже это может быть. Хоть SRAM значительно быстрее в скорости, но она так же ощутимо дороже, чем SDRAM. Однако всегда надо выбирать по потребностям.

В итоге

1. SRAM статична (нет необходимости в перезаписи), в то время как SDRAM динамична (требуется периодическая перезапись).
2. Время доступа в SDRAM зависит от тактового сигнала, в то время как в SRAM доступ осуществляется напрямую.
3. Чип DRAM может содержать несколько гигабит, в то время как чип SDRAM может содержать только несколько десятков мегабит.
4. Ток потребления SRAM стабильный, в то время как у SDRAM он динамический и возрастает с увеличением циклов перезаписи.
5. SRAM более дорогостоящая, чем SDRAM, т.к. более высокоскоростная.

9.2.2 Наборы для обнаружения

Платы обнаружения помогают пользователю открывать высокопроизводительные микроконтроллеры серии STM32 F4 и легко разрабатывать приложения.

Доступны различные платы обнаружения. Например, STM32F429I-DISCO включает следующие функции:

- SDRAM 64Mbits
- L3GD20, датчик движения MEMS, трехосный цифровой выходной гироскоп
- USB OTG с разъемом micro AB

Table 12. Program/erase parallelism

	Voltage range 2.7 - 3.6 V with External V_{PP}	Voltage range 2.7 - 3.6 V	Voltage range 2.4 - 2.7 V	Voltage range 2.1 - 2.4 V	Voltage range 1.8 V - 2.1 V
Parallelism size	x64	x32	x16		x8
PSIZE(1:0)	11	10	01		00

Table 13. Flash interrupt request

Interrupt event	Event flag	Enable control bit
End of operation	EOP	EOPIE
Write protection error	WRPERR	ERRIE
Programming error	PGAERR, PGPERR, PGSERR	ERRIE
Read protection error	RDERR	ERRIE

8.4.2 Интерфейс контроллера гибкой памяти (FMC)

Подключение интерфейса

Контроллер FMC и, в частности, контроллер памяти SDRAM, который имеет много сигналов, большинство из них имеют аналогичную функциональность и работают вместе. Сигналы ввода-вывода контроллера можно разделить на четыре группы следующим образом:

• Группа адресов, состоящая из адреса строки / столбца и адреса банка.

• Группа команд, которая включает строковый адресный строб (NRAS), строковый адрес столбца (NCAS) и разрешение записи (SDWE).

• Контрольная группа, которая включает в себя блок выбора чипа 1 и bank2 (SDNE0 / 1), банкнот с тактовой частотой 1 и bank2 (SDCKE0 / 1) и выходную байтовую маску для доступа к записи (DQM).

• Группа данных / полоса, которая содержит 8 сигналов (a): восемь D (D7-D0) и маску данных (DQM).

Заметка:

Это зависит от используемой памяти: SDRAM с шириной шины x8 имеет только одну группу данных, тогда как SDRAM с шиной x16 и x32 имеет две и четыре полосы соответственно.

Рекомендации по компоновке интерфейса:

• Ссылка на плоскость с использованием GND или PWR (если PWR, добавьте колпачок 10nf между PWR и GND

• Отслеживание импеданса: 50 Ом ± 10%

• Максимальная длина трассы должна быть ниже 120 мм. Если тракт сигнала превышает этот критерий длины трассы / скорости, то следует использовать прерывание.

• Уменьшите перекрестные помехи, поместите дорожки данных на разных уровнях с адресной и управляющей полос, если это возможно. В то же время, когда треки данных и адреса / управления сосуществуют на одном слое, они должны быть изолированы друг от друга не менее чем на 5 мм.

• Сопоставьте длины трасс для группы данных в пределах ± 10 мм друг от друга, чтобы уменьшить косы. Трассировки серпентина (обратные и четвертые следы в шаблоне «S» для увеличения длины трассировки) могут использоваться для соответствия длинам.

• Размещение сигнала часов (SDCLK) на внутреннем слое, минимизирует шум (EMI).

• Прокладывайте тактовый сигнал по крайней мере на расстоянии 3x от проводов от других сигналов. Используйте как можно меньше отверстий, чтобы избежать изменения импеданса и отражения. Избегайте использования серпантинной маршрутизации.

• Сопоставьте трассировки часов с трассировкой группы данных / адреса в пределах ± 10 мм.

• Сопоставьте трассировки часов с каждой трассой сигнала в адресных и командных группах с точностью ± 10 мм (максимум <= 20 мм).

• Проследите за емкостями:

- На 3,3 В сохраняйте трассировку в пределах 20 пФ с общей емкостной нагрузкой (включая данные, адрес, SDCLK и управление) не более 30 пФ.

- При 1,8 В сохраняйте трассировку в пределах 15 пФ с общей емкостной нагрузкой (включая данные, адрес, SDCLK и управление) не более 20 пФ

59 Универсальный драйвер HAL SDRAM

59.1 SDRAM Драйвер прошивки регистрирует структуры

59.1.1 SDRAM_HandleTypeDef

Поля данных

- FMC_SDRAM_TypeDef * Экземпляр
- FMC_SDRAM_InitTypeDef Init
- __IO HAL_SDRAM_StateTypeDef State
- HAL_LockTypeDef Lock
- DMA_HandleTypeDef * hdma

Описание полей

- FMC_SDRAM_TypeDef* SDRAM_HandleTypeDef::Instance Register base address (Базовый адрес регистра экземпляра)
- FMC_SDRAM_InitTypeDef SDRAM_HandleTypeDef::Init SDRAM device configuration parameters (Параметры конфигурации)
- __IO HAL_SDRAM_StateTypeDef SDRAM_HandleTypeDef::State SDRAM access state (Состояние доступа к SDRAM)
- HAL_LockTypeDef SDRAM_HandleTypeDef::Lock SDRAM locking object (Блокировать объект SDRAM)
- DMA_HandleTypeDef* SDRAM_HandleTypeDef::hdma Pointer DMA handler (Указатель DMA-обработчик)

59.2 Описание API драйвера прошивки SDRAM

59.2.1 Как использовать этот драйвер

Этот драйвер является универсальным многоуровневым драйвером, который содержит набор API, используемых для управления памятью SDRAM. Он использует функции уровня FMC для взаимодействия с устройствами SDRAM. Для настройки FMC для взаимодействия с памятью SDRAM необходимо следовать следующей последовательности:

1. Объявите структуру дескриптора SDRAM_HandleTypeDef, например: SDRAM_HandleTypeDef hdsram

Заполните поле «Init» в SDRAM_HandleTypeDef с допустимыми значениями элемента структуры.

Заполнение дескриптора SDRAM_HandleTypeDef дескриптора «Экземпляр» с предопределенным экземпляром базового регистра для устройства NOR или SDRAM

2. Объявите структуру FMC_SDRAM_TimingTypeDef; например: FMC_SDRAM_TimingTypeDef Timing; и заполнить его поля допустимыми значениями элемента структуры.

3. Инициализируйте контроллер SDRAM, вызвав функцию HAL_SDRAM_Init (). Эта функция выполняет следующую последовательность:

a. Конфигурация аппаратного уровня MSP с использованием функции HAL_SDRAM_MspInit ()

b. Конфигурация управляющего регистра с использованием функции интерфейса FMC SDRAM FMC_SDRAM_Init ()

c. Конфигурация регистра времени с использованием функции интерфейса FMC SDRAM FMC_SDRAM_Timing_Init ()

d. Запрограммируйте внешнее устройство SDRAM, выполнив его последовательность инициализации в соответствии с подключенным к вашему оборудованию устройством. Этот шаг является обязательным для доступа к устройству SDRAM.

4. На этом этапе вы можете читать / записывать обращения в SDRAM Bank. Вы можете выполнить либо опрос, либо передачу DMA, используя следующие API:

• HAL_SDRAM_Read () / HAL_SDRAM_Write () для опроса доступа для чтения / записи

• HAL_SDRAM_Read_DMA () / HAL_SDRAM_Write_DMA () для передачи данных DMA

5. Вы можете таким образом управлять устройством SDRAM с помощью вызова API для управления HAL_SDRAM_WriteOperation_Enable () / HAL_SDRAM_WriteOperation_Disable (), чтобы соответственно включить / отключить SDRAM оперативной или написать функцию HAL_SDRAM_SendCommand (), чтобы послать указанную команду к SDRAM устройства. Команда должна быть сконфигурирована с помощью структуры FMC_SDRAM_CommandTypeDef.

6. Вы можете непрерывно контролировать состояние HAL устройства SDRAM, вызывая функцию HAL_SDRAM_GetState ()

1-2 Лежит в файле stm32f4xx_ll_fmc.h

* @brief Определение структуры конфигурации FMC SDRAM

typedef struct {

uint32_t SDBank; /*!< Указывает устройство памяти SDRAM, которое будет использоваться. Этот параметр может быть значением @ref FMC_SDRAM_Bank. */

uint32_t ColumnBitsNumber; /*!< Определяет номер бита адреса столбца. Этот параметр может быть значением @ref FMC_SDRAM_Column_Bits_number.*/

uint32_t RowBitsNumber; /*!< Определяет количество бит адреса столбца. This parameter can be a value of @ref FMC_SDRAM_Row_Bits_number. */

uint32_t MemoryDataWidth; /*!< определяет ширину шины памяти. This parameter can be a value of @ref FMC_SDRAM_Memory_Bus_Width. */

uint32_t InternalBankNumber; /*!< Определяет количество устройств внутри банка. This parameter can be of @ref FMC_SDRAM_Internal_Banks_Number. */

uint32_t CASLatency; /*!< Определяет задержку SDRAM CAS в количестве тактовых импульсов памяти. This parameter can be a value of @ref FMC_SDRAM_CAS_Latency. */

uint32_t WriteProtection; /*!< Включить доступ к устройству SDRAM в режиме записи. This parameter can be a value of @ref FMC_SDRAM_Write_Protection. */

uint32_t SDClockPeriod; /*!< Определите период синхронизации SDRAM для обоих устройств SDRAM, и они позволяют отключить часы перед изменением частоты. This parameter can be a value of @ref FMC_SDRAM_Clock_Period. */

uint32_t ReadBurst; /*!< Этот бит позволяет контроллеру SDRAM предвидеть следующие команды чтения во время задержки CAS и сохраняет данные в Read FIFO. This parameter can be a value of @ref FMC_SDRAM_Read_Burst. */

uint32_t ReadPipeDelay; /*!< Определить задержку в тактовых циклах системы на пути чтения данных. This parameter can be a value of @ref FMC_SDRAM_Read_Pipe_Delay. */

}FMC_SDRAM_InitTypeDef;

/**
* @brief FMC SDRAM Определение структуры параметров времени */
typedef struct
{
uint32_t LoadToActiveDelay; /*!< Определяет задержку между командой Load Mode Register и активной или командой Refresh (обновление)в количестве тактовых импульсов памяти. This parameter can be a value between Min_Data = 1 and Max_Data = 16 */ ***(время между записью в MODE-REGISTER и ACTIVATE)***

uint32_t ExitSelfRefreshDelay; /*!< Определяет задержку от отпускания команды самообновления для выдачи команды Activate (активировать) в количестве тактовых импульсов памяти. This parameter can be a value between Min_Data = 1 and Max_Data = 16 */ ***(время между SELF-REFRESHING и ACTIVATE (exit self-refresh mode).)***

uint32_t SelfRefreshTime; /*!< Определяет минимальный период Self Refresh (самообслуживания) в количестве тактовых импульсов памяти. This parameter can be a value between Min_Data = 1 and Max_Data = 16 */ **(минимальное время между SELF-REFRESH)**

uint32_t RowCycleDelay; /*!< Определяет задержку между командой Refresh и командой Activate и задержкой между двумя последовательными командами Refresh в количестве тактовых импульсов памяти. This parameter can be a value between Min_Data = 1 and Max_Data = 16 */ ***(время между двумя командами REFRESH.)***

uint32_t WriteRecoveryTime; /*!< Определяет время восстановления записи в количестве тактовых импульсов памяти. This parameter can be a value between Min_Data = 1 and Max_Data = 16 */ ***(задержка между командой WRITE и вызовом PRECHARGE.)***

uint32_t RPDelay; /*!< Определяет задержку между командой Precharge и другой командой в количестве тактовых импульсов памяти. This parameter can be a value between Min_Data = 1 and Max_Data = 16 */ ***(время между командой PRECHARGE и любой другой командой.)***

uint32_t RCDDelay; /*!< Определяет задержку между командой Activate (Активировать) и командой Read/Write в количестве тактовых импульсов памяти.

This parameter can be a value between Min_Data = 1 and Max_Data = 16 */ ***(время между подачей команды ACTIVATE и появлением данных на шине.)***

}FMC_SDRAM_TimingTypeDef;

Эти 2 структыры заполняются Cube в static void MX_FMC_Init(void)

/**
* @brief Определение структуры параметров команды SDRAM */
typedef struct
{
uint32_t CommandMode; /*!< Определяет команду, выдаваемую устройству SDRAM. This parameter can be a value of @ref FMC_SDRAM_Command_Mode. */

uint32_t CommandTarget; /*!< Определяет, какое устройство (1 или 2) выдаст команду. This parameter can be a value of @ref FMC_SDRAM_Command_Target. */

uint32_t AutoRefreshNumber; /*!< Определяет количество последовательных команд автоматического обновления, выданных в режиме автоматического обновления. This parameter can be a value between Min_Data = 1 and Max_Data = 16 */

uint32_t ModeRegisterDefinition; /*!< Определяет содержимое Mode register (регистра режима) SDRAM */
}FMC_SDRAM_CommandTypeDef;

59.2.2 Функции инициализации SDRAM и деинициализации

В этом разделе представлены функции инициализации / деионизации SDRAM-памяти

Этот раздел содержит следующие API:

- HAL_SDRAM_Init ()
- HAL_SDRAM_DeInit ()
- HAL_SDRAM_MspInit ()
- HAL_SDRAM_MspDeInit ()
- HAL_SDRAM_IRQHandler ()
- HAL_SDRAM_RefreshErrorCallback ()
- HAL_SDRAM_DMA_XferCpltCallback ()
- HAL_SDRAM_DMA_XferErrorCallback ()

59.2.3 Функции ввода и вывода SDRAM

В этом разделе приведены функции для использования и управления памятью SDRAM

Этот раздел содержит следующие API:

- HAL_SDRAM_Read_8b ()
- HAL_SDRAM_Write_8b ()
- HAL_SDRAM_Read_16b ()
- HAL_SDRAM_Write_16b ()
- HAL_SDRAM_Read_32b ()
- HAL_SDRAM_Write_32b ()
- HAL_SDRAM_Read_DMA ()
- HAL_SDRAM_Write_DMA ()

59.2.4 Функции управления SDRAM

В этом подразделе представлен набор функций для динамического управления интерфейсом SDRAM.

Этот раздел содержит следующие API:

- HAL_SDRAM_WriteProtection_Enable ()
- HAL_SDRAM_WriteProtection_Disable ()
- HAL_SDRAM_SendCommand ()
- HAL_SDRAM_ProgramRefreshRate ()
- HAL_SDRAM_SetAutoRefreshNumber
- HAL_SDRAM_GetModeStatus()

59.2.5 Функции государства SDRAM

Этот подраздел позволяет получить статус контроллера SDRAM и потока данных во время выполнения.

Этот раздел содержит следующие API:

- HAL_SDRAM_GetState ()

59.2.6 Подробное описание функций

HAL_SDRAM_Read_DMA

Имя функции HAL_StatusTypeDef HAL_SDRAM_Read_DMA
(SDRAM_HandleTypeDef * hsdram, uint32_t * pAddress,
uint32_t * pDstBuffer, uint32_t BufferSize)

Описание функции:

Чтение данных Words из памяти SDRAM с использованием DMA передачи

Параметры:

• hsdram: указатель на структуру SDRAM_HandleTypeDef, которая содержит информацию о конфигурации для модуля SDRAM.

• pAddress: Указатель для чтения начального адреса

• pDstBuffer: Указатель на буфер назначения

• BufferSize: Размер буфера для чтения из памяти

Возвращаемые значения:

• HAL: status

HAL_SDRAM_Write_DMA

Имя функции HAL_StatusTypeDef HAL_SDRAM_Write_DMA
(SDRAM_HandleTypeDef * hsdram, uint32_t * pAddress,
uint32_t * pSrcBuffer, uint32_t BufferSize)

Описание функции:

Записывает буфер данных Words в память SDRAM с использованием передачи DMA.

Параметры:

• hsdram: Указатель на структуру SDRAM_HandleTypeDef, которая содержит информацию о конфигурации для модуля SDRAM.

• pAddress: Указатель начального адреса записи

• pSrcBuffer: Указатель на исходный буфер для записи

• BufferSize: Размер буфера для записи в память

Возвращаемые значения:

• HAL: status

24 HAL FLASH Generic Driver

24.1 FLASH-драйвер встроенного программного обеспечения регистрирует структуры

24.1.1 FLASH_ProcessTypeDef

Поля данных

- __IO FLASH_ProcedureTypeDef ProcedureOnGoing
- __IO uint32_t NbSectorsToErase
- __IO uint8_t VoltageForErase
- __IO uint32_t Sector
- __IO uint32_t Bank
- __IO uint32_t Address
- HAL_LockTypeDef Lock
- __IO uint32_t ErrorCode

Описание полей

- __IO FLASH_ProcedureTypeDef FLASH_ProcessTypeDef :: ProcedureOnGoing
- __IO uint32_t FLASH_ProcessTypeDef :: NbSectorsToErase
- __IO uint8_t FLASH_ProcessTypeDef :: VoltageForErase
- __IO uint32_t FLASH_ProcessTypeDef :: Sector
- __IO uint32_t FLASH_ProcessTypeDef :: Bank
- __IO uint32_t FLASH_ProcessTypeDef :: Address
- HAL_LockTypeDef FLASH_ProcessTypeDef :: Lock
- __IO uint32_t FLASH_ProcessTypeDef :: ErrorCode

24.2 Описание API драйвера прошивки FLASH

24.2.1 Функции периферийного устройства FLASH

Интерфейс флэш-памяти управляет I-Code AHB и D-кодами доступа к флэш-памяти. Он реализует операции стирания и программной флэш-памяти и механизмы защиты чтения и записи.

Интерфейс флэш-памяти ускоряет выполнение кода с помощью системы предварительной выборки команд и строк кэша.

Основные функции FLASH:

- Операции чтения флэш-памяти
- Операции с флэш-памятью / стирание
- Защита чтения / записи
- Предварительная выборка по I-коду
- 64 кеш-строки из 128 бит в I-Code
- 8 строк кеша 128 бит в D-коде

24.2.2. Как использовать этот драйвер

Этот драйвер предоставляет функции и макросы для настройки и программирования FLASH-памяти всех устройств STM32F4xx.

1. Функции программирования памяти FLASH Memory IO:

- Блокировать и разблокировать интерфейс FLASH с помощью функций HAL_FLASH_Unlock () и HAL_FLASH_Lock ()

- Функции программы: байт, слово, слово и двойное слово
- Там два режима программирования:
- Режим опроса с использованием функции HAL_FLASH_Program ()
- Режим прерывания с использованием функции HAL_FLASH_Program_IT ()

2. Функции управления прерываниями и флагами:

- Обрабатывать прерывания FLASH, вызывая HAL_FLASH_IRQHandler ()
- Подождите, пока последняя операция FLASH в соответствии с ее статусом
- Получить статус флага ошибки, вызвав HAL_SetErrorCode ()

В дополнение к этим функциям этот драйвер включает в себя набор макросов, позволяющих обрабатывать следующие операции:

- Установите задержку
- Включение / выключение буфера предварительной выборки
- Включить / отключить кеш инструкций и кеш данных
- Сбросить кеш инструкций и кеш данных
- Включить / отключить прерывания FLASH
- Мониторинг состояния FLASH-флагов

24.2.3 Функции программирования

В этом подразделе представлен набор функций, позволяющих управлять операциями программы FLASH.

Этот раздел содержит следующие API:

- HAL_FLASH_Program ()
- HAL_FLASH_Program_IT ()
- HAL_FLASH_IRQHandler ()
- HAL_FLASH_EndOfOperationCallback ()
- HAL_FLASH_OperationErrorCallback ()

24.2.4 Функции периферийного управления

В этом подразделе представлен набор функций, позволяющих управлять операциями FLASH-памяти.

Этот раздел содержит следующие API:

- HAL_FLASH_Unlock ()
- HAL_FLASH_Lock ()
- HAL_FLASH_OB_Unlock ()
- HAL_FLASH_OB_Lock ()
- HAL_FLASH_OB_Launch ()

24.2.5 Функции периферийных ошибок

Этот подраздел позволяет получить во время выполнения ошибки окружения FLASH.

Этот раздел содержит следующие API:

- HAL_FLASH_GetError ()
- FLASH_WaitForLastOperation ()

24.2.6 Подробное описание функций