Введение

Чтобы управлять широким спектром мультимедиа, более богатой графикой и другим содержимым с интенсивным использованием данных, встроенные приложения развиваются, предлагая более сложные функции. Эти сложные функции требуют дополнительных требований к ограниченной памяти микроконтроллера MCU.

Внешняя параллельная память используется для расширения встроенной памяти микроконтроллера и устранения ограничений по размеру памяти. Обычно это действие компрометирует количество выводов и требует более сложных конструкций.

Чтобы соответствовать этим требованиям, микроконтроллеры STM32 (см. Применимые продукты в Табл. 1 ниже) встраивают интерфейс внешней памяти с именем QUADSPI (более подробную информацию см. В Табл. 2 на стр. 9). Этот интерфейс позволяет подключать внешние высокоскоростные блоки памяти Quad-Spi (QSPI) с компактным размером. Этот интерфейс QUADSPI можно использовать для хранения данных, таких как изображения, значки или выполнение кода.

В этом примечании по применению описывается интерфейс QUADSPI на микроконтроллерах STM32 и объясняется, как использовать модуль для настройки, программирования и чтения внешней памяти QSPI. В нем описаны некоторые типичные случаи использования интерфейса QUADSPI на основе некоторых примеров программного обеспечения из пакета прошивки STM32Cube и из замечаний по применению серии STM32F7.

Этот документ ссылается на интерфейс Quad-SPI STM32 под названием «QUADSPI», а на память Quad-SPI - на «память QSPI».

Для получения более подробной информации о продуктах, перечисленных в таблице ниже, см. Соответствующие таблицы данных и справочные руководства, доступные на веб-сайте STMicroelectronics www.st.com.

1 Обзор

QUADSPI - это последовательный интерфейс, который обеспечивает связь по четырем линиям данных между хостом (STM32) и внешней памятью QSPI. QUADSPI поддерживает традиционный SPI (последовательный периферийный интерфейс), а также режим двойного SPI, который позволяет осуществлять связь по двум линиям. QUADSPI использует до шести строк в четырехугольном режиме: одна строка для выбора микросхемы, одна строка для тактового генератора и четыре строки для ввода и вывода данных.

Этот интерфейс встроен в микроконтроллер STM32, чтобы соответствовать приложениям, требующим памяти, чтобы упростить конструкцию печатных плат и снизить затраты.

1.1 Доступность и возможности QUADSPI для семейств STM32

Все микроконтроллеры STM32, показанные в таблице ниже, имеют в основном те же функции QUADSPI, за исключением устройств серий STM32L4 и STM32WB55, которые не поддерживают память с двумя флеш-накопителями.

Table 2. Quad-SPI availability and features across STM32 families

<table>
<tr><th rowspan="2">Products</th><th colspan="2">Maximum speed (MHz)(1)</th><th rowspan="2">Dual-Flash memory</th><th rowspan="2">FIFO size (byte)</th><th colspan="2">Max addressable space (2)</th></tr>
<tr><th>SDR</th><th>DDR</th><th>Memory mapped</th><th>Indirect mode</th></tr>
<tr><td>STM32F412 line</td><td rowspan="2">100</td><td rowspan="2">80</td><td rowspan="8">Yes</td><td rowspan="8">32</td><td rowspan="15">256 Mbytes</td><td rowspan="15">4 Gbytes</td></tr>
<tr><td>STM32F413/423 line(3)</td></tr>
<tr><td>STM32F446 line(4)</td><td rowspan="2">90</td><td>60</td></tr>
<tr><td>STM32F469/479 line</td><td rowspan="4">80</td></tr>
<tr><td>STM32F730xx devices
STM32F7x2 line(4)</td><td rowspan="3">108</td></tr>
<tr><td>STM32F750xx
STM32F7x3
STM32F7x5
STM32F7x6
STM32F7x7
STM32F7x8
STM32F7x9</td></tr>
<tr><td>STM32H743/753
STM32H750 Value line</td><td>133</td><td>100</td></tr>
<tr><td>STM32L471xx
STM32L412xx
STM32L422xx
STM32L432xx
STM32L442xx
STM32L475xx
STM32L476xx
STM32L486xx</td><td rowspan="5">60</td><td rowspan="4">48</td><td>No</td><td rowspan="4">16</td></tr>
<tr><td>STM32L431xx
STM32L451xx
STM32L452xx
STM32L462xx
STM32L4x3(5)</td><td rowspan="3">Yes</td></tr>
<tr><td>STM32L496xx
STM32L4A6xx</td></tr>
<tr><td>STM32WB55xx</td><td>50</td><td>No</td></tr>
<tr><td>STM32L4R5/S5
STM32L4R7/S7
STM32L4R9/S9(6)</td><td>86</td><td>60</td><td>Yes</td><td>32</td></tr>
</table>

1. Максимальная скорость QUADSPI из таблицы. Для получения более подробной информации о максимальной скорости QUADSPI обратитесь к соответствующей спецификации устройства.

2. 32-битный адресный режим должен использоваться для достижения 256 Мбайт в режиме отображения памяти и 4 Гбайт в косвенном режиме.

3. UFQFPN48 не поддерживает QUADSPI.

4. LQFP64 поддерживает только Bank1 и Single-SPI / Dual-SPI.

5. Для этого набора продуктов режим Dual-Flash поддерживается только с пакетами LQFP100 и UFBGA100.

6. Этот набор продуктов содержит два интерфейса Octo-SPI, каждый из которых может подключать одну или две памяти QSPI с режимами Single-Flash или Dual-Flash.

1.2 Преимущества QUADSPI по сравнению с классическими SPI и параллельными интерфейсами

QUADSPI обеспечивает более высокую производительность по сравнению с классическим SPI. Классический SPI использует только одну строку данных, в то время как QUADSPI использует четыре строки данных, что увеличивает пропускную способность почти в четыре раза.

По сравнению с FMC (гибкий интерфейс памяти) и другими параллельными интерфейсами, QUADSPI позволяет подключать более дешевую внешнюю флэш-память к небольшим пакетам, уменьшая площадь печатной платы, упрощая конструкцию печатной платы и сокращая использование GPIO (ввода / вывода общего назначения) , В режиме Quad-SPI используются только шесть GPIO: четыре строки для данных плюс одна строка для тактовой частоты и другая для выбора микросхемы. В режиме Dual-Flash Quad-SPI используются только 10 GPIO, из которых восемь строк предназначены для данных.

1.2.1 Основные преимущества встроенного интерфейса QUADSPI STM32

В следующей таблице приведены основные преимущества использования встроенного интерфейса QUADSPI STM32:

Таблица 3. Преимущества использования интерфейса STM32 QUADSPI

Выгоды	Комментарии
Низкое количество выводов	Поддерживает одиночную, двойную память и память QSPI. Использует шесть контактов в режиме Quad-SPI и четыре контакта для одиночного или двойного SPI. Сохраняет GPIO для использования в других целях.
Более простой дизайн печатной платы	Позволяет проще и быстрее проектировать печатную плату благодаря уменьшенному количеству выводов.
Экономия места для приложений меньшего размера	Может использоваться в приложениях небольшого размера из-за небольшого объема памяти QSPI.
Не высокая цена	Более простая и быстрая конструкция обеспечивает более низкую стоимость разработки. Более низкая стоимость печатной платы, так как возможно уменьшить количество слоев печатной платы благодаря низкому количеству выводов. Низкая стоимость решения для памяти.
Выполняемая	Расширяет ограниченную встроенную флэш-память, позволяя рассматривать память QSPI как внутреннюю память. Позволяет выполнять код (режим XIP) из флэш-памяти QSPI. Поддерживает режим SIOO, также называемый режимом непрерывного чтения некоторыми производителями памяти (см. Раздел 2.4.1: Отправка команды только один раз (SIOO) на стр. 32) для повышения производительности выполнения.
Расширенный размер для хранения данных	Режим отображения памяти позволяет автономно получать доступ к памяти QSPI с помощью любого режима master.32-битного AHB (усовершенствованного высокопроизводительного интерфейса) или AXI (усовершенствованного протокола расширяемого интерфейса), позволяющего адресовать до четырех Гбайт памяти QSPI. Режим флэш-памяти позволяет использовать две флэш-памяти QSPI для удвоения объема памяти (1).

Высокие показатели	Пропускная способность умножается на четыре по сравнению с традиционным SPI. Режим DDR удваивает пропускную способность. Режим Dual-Flash памяти удваивает пропускную способность. Идеально подходит для графических приложений.
Несколько решений для памяти	Существуют энергозависимые QSPI SRAM (статическая память с произвольным доступом), доступные от Microchip, ON Semiconductor и др. Доступные энергонезависимые флэш-памяти QSPI. NOR, NAND.
Поддерживает любую память QSPI, доступные на рынке	Его полностью конфигурируемый и гибкий формат кадра позволяет поддерживать практически все устройства QSPI, доступные на рынке.
Growing amount of manufacturers	Spansion, Windbond, Micron, Macronix, ONSemiconductors, Cypress, APmemory и ISSI и другие. Большие инвестиции в высокоплотные флэш-памяти QSPI, такие как NAND.

1. Максимальный размер 4 Гбайт может быть достигнут с помощью 32-битного адресного режима

1.3 QUADSPI в умной архитектуре

Интерфейс QUADSPI отображается на выделенном слое на AHB, что позволяет ему быть доступным как внутренняя память благодаря режиму Memory-mapped. Кроме того, QUADSPI интегрирован в интеллектуальную архитектуру, которая обеспечивает следующие функции:

- Мастера для доступа к внешней памяти QSPI без какого-либо вмешательства ЦП.
- Умение читать данные из памяти QSPI даже в спящем режиме, когда процессор остановлен благодаря интеллектуальной архитектуре STM32.
- CPU как мастер может получить доступ к QUADSPI и выполнить код из памяти.
- GP DMA для передачи из QSPI в другую внутреннюю или внешнюю память.
- Графический DMA2D для непосредственного построения видеокадров ОЗУ с использованием QSPI Flash.

1.3.1 Архитектура системы: серия STM32L4

Архитектура системы серии STM32L4 состоит в основном из 32-битной многослойной матрицы шины AHB, которая соединяет несколько мастеров с несколькими подчиненными.

Доступ к QUADSPI может осуществляться соответствующими мастерами, такими как Cortex-M4, либо через S-Bus, либо через I-bus и D-bus, если включен режим преобразования данных. QUADSPI также доступен для DMA1 и DMA2.

Включение физического переназначения по I-bus и D-bus повышает производительность при выполнении для Cortex-M4.

Доступ к QUADSPI может быть либо доступом к регистрам, либо доступом к области с отображением в памяти:

- Доступ к регистрам может быть сделан Cortex-M4 для конфигурации регистров или передачи данных. Доступ к регистру также может быть выполнен с помощью DMA1 и DMA2 для передачи данных.
- Доступ к области отображения памяти может быть сделан Cortex-M4 для получения кода и данных. DMA1, DMA2 и DMA2D также могут обращаться к обла-

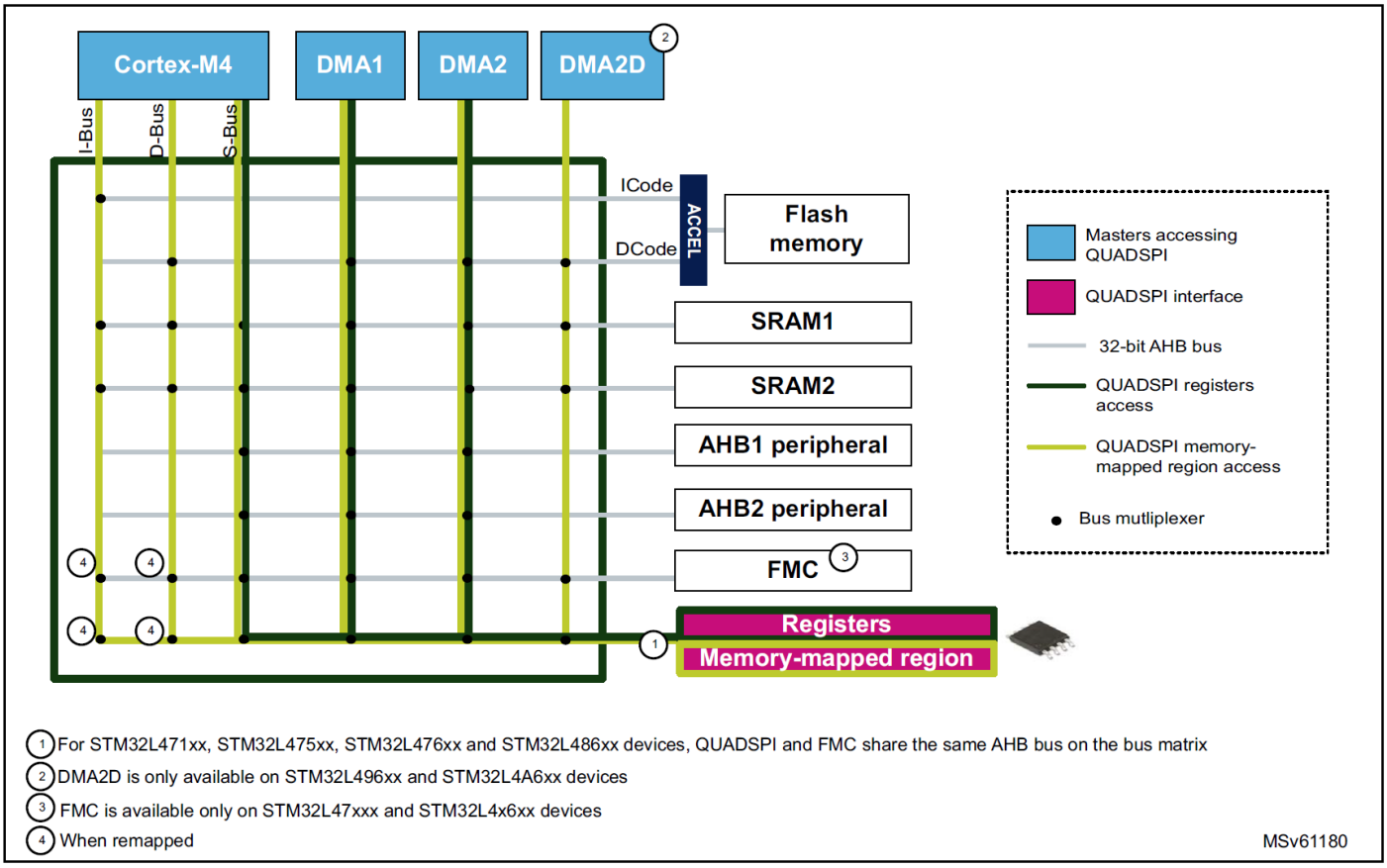
сти отображения памяти для передачи данных.

На рисунке ниже показано соединение QUADSPI в системе серии STM32L4.

Примечание:

DMA2D доступен только в устройствах STM32L496xx и STM32L4A6xx.

Figure 1. System architecture: STM32L4 Series



1.3.2 Архитектура системы: серия STM32F4

Архитектура системы серии STM32F4 состоит в основном из 32-битной многослойной матрицы шины AHB, которая соединяет несколько мастеров с несколькими ведомыми (подробности о применяемых продуктах см. На титульном листе).

Cortex®-M4 может получить доступ к внешней памяти QSPI через S-шину. QUADSPI также доступен всем мастерам на матрице шины AHB, таким как DMA1, DMA2, USB OTG HS, MAC Ethernet, LTDC и DMA2D. Эта доступность обеспечивает эффективную передачу данных (например, изображения для графических приложений).

Доступ к QUADSPI может быть либо доступом к регистрам, либо доступом к области с отображением в памяти:

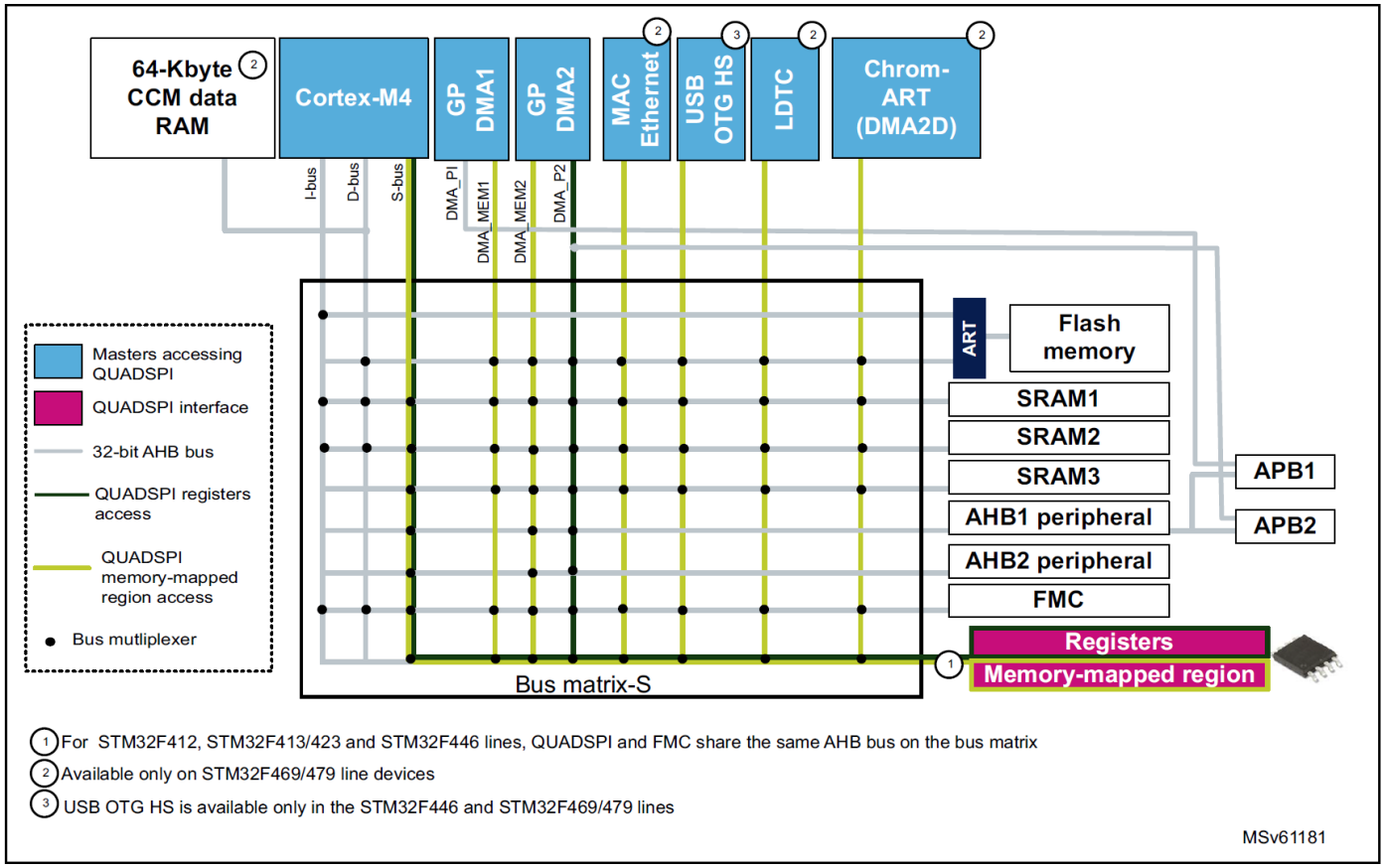
- Доступ к регистрам может быть выполнен Cortex-M4 через S-Bus для конфигурации регистров и передачи данных. Доступ к регистру можно сделать также GP DMA2 для передачи данных.
- Доступ к области с отображенной памятью может быть выполнен Cortex-M4 через S-Bus для извлечения кода и данных. Доступ к области с отображенной памятью может быть также выполнен с помощью GP DMA1, GP DMA2, MAC Ethernet, USB OTG HS, LTDC и DMA2D для передачи данных.

На рисунке ниже показано соединение QUADSPI в системе серии STM32F4.

Примечание:

Для MAC Ethernet, USB OTG HS, LTDC и DMA2D обратитесь к соответствующему продукту.

Figure 2. System architecture: STM32F4 Series



1.3.3 Архитектура системы: серия STM32F7

Основная архитектура системы основана на двух подсистемах: AXI (расширенный расширяемый интерфейс) для нескольких мостов AHB, преобразующих протокол AXI4 в протокол AHB-Lite и матрицу шин с несколькими AHB.

Матрица с несколькими шинами AHB соединяет несколько ведущих и нескольких ведомых устройств. Есть четыре доступа к шине AXI; QUADSPI доступен через второй доступ. Этот доступ позволяет Cortex®-M7 осуществлять доступ к области, отображаемой в памяти, для получения кода или данных. Этот доступ также позволяет Cortex®-M7 выполнять доступ к регистру для конфигурации регистров QUADSPI или для передачи данных.

QUADSPI отображается на выделенном уровне в матрице шины AHB, что позволяет Cortex®-M7 использовать L1-кэш при доступе к кэшированным данным с состояниями 0 ожидания.

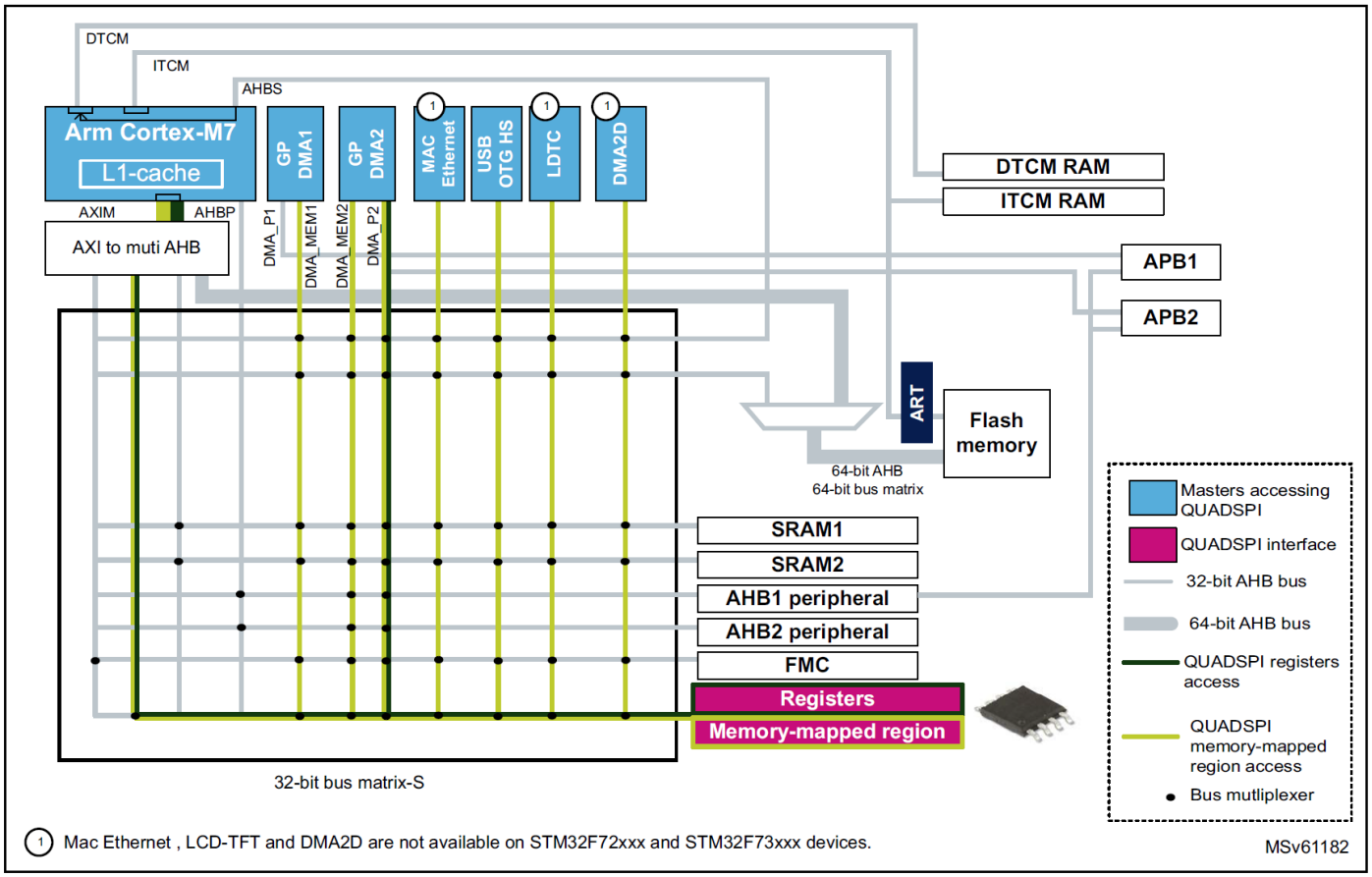
QUADSPI также доступен для всех мастеров на матрице шины AHB. Доступ к регистрам может выполняться GP DMA2 для передачи данных. Доступ к области памяти, отображаемой в память, может быть выполнен с помощью GP DMA1, MAC Ethernet, USB OTG HS, LTDC и DMA2D. Эта доступность обеспечивает эффективную передачу данных (например, изображения для графических приложений).

На следующем рисунке показано соединение QUADSPI в системе серии STM32F7.

Примечание:

Для MAC Ethernet, USB OTG HS, LTDC и DMA2D обратитесь к соответствующему продукту.

Figure 3. System architecture: STM32F7 Series



1.3.4 Архитектура системы: серия STM32H7

Основная архитектура системы основана на трех областях: D1, D2 и D3. Каждый домен содержит матрицу шины, которая обеспечивает соединение между несколькими мастерами и несколькими подчиненными. 64-разрядная матрица шины AXI для домена D1 и 32-разрядная матрица шины AHB для каждого из доменов D2 и D3.

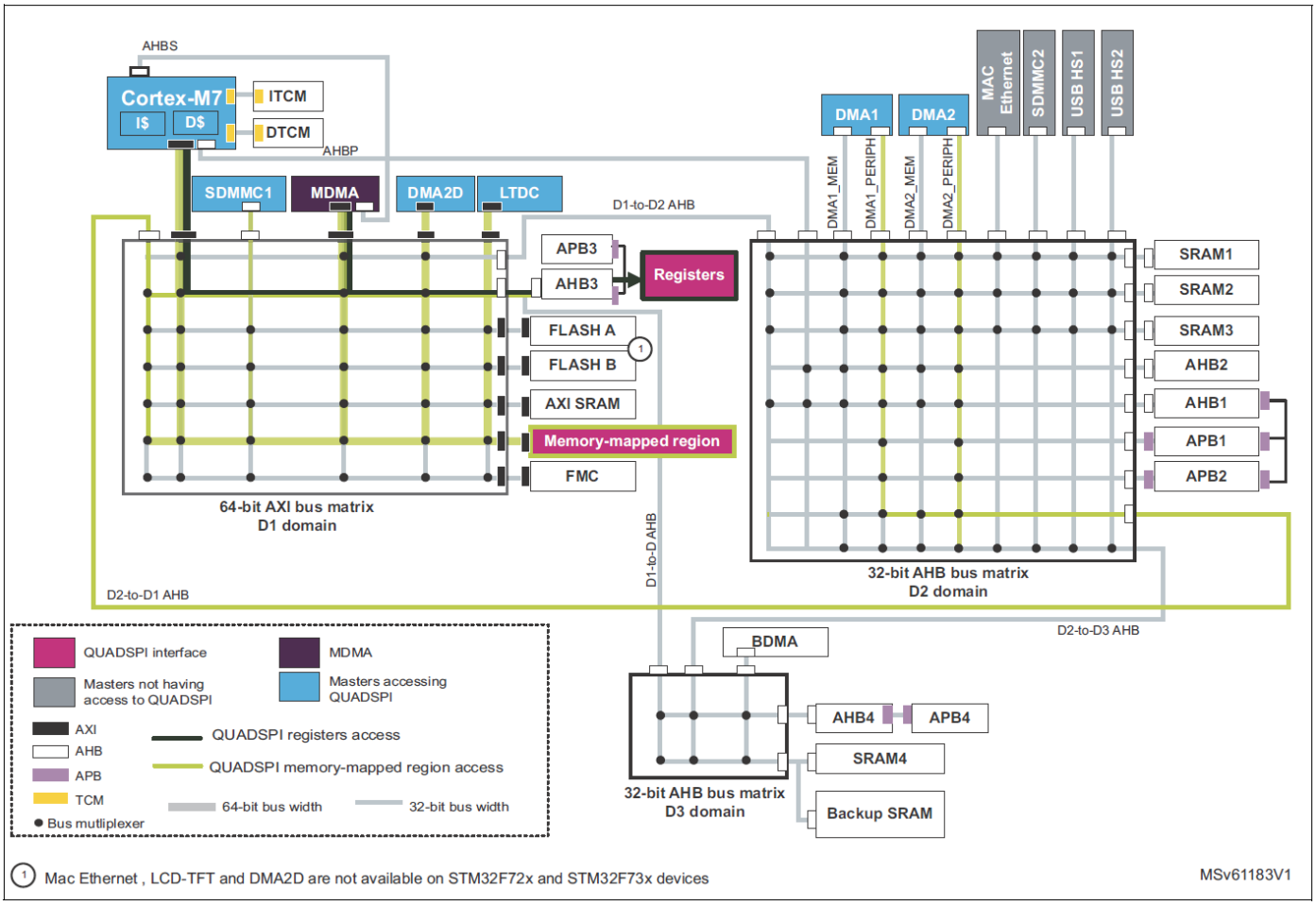
Три домена связаны друг с другом междоменными шинами AHB, которые позволяют мастерам из определенного домена получать доступ к подчиненным устройствам из другого домена.

QUADSPI подключен к домену D1 и доступен через:

- 64-битная шина AXI, подключенная напрямую к матрице шины AXI. Это позволяет нескольким мастерам выполнять доступ к области отображения памяти для кода и выборки данных из домена D1, такого как Cortex-M7. Это позволяет также передавать данные из домена D1 (SDMMC1, MDMA, DMA2D и LTDC) и из домена D2 (DMA1 и DMA2).
- 32-битная шина AHB, доступная через AHB3. Это позволяет Cortex-M7 и MDMA выполнять доступ к регистру для передачи данных или конфигурации регистров.

На следующем рисунке показано соединение QUADSPI в системе серии STM32H7.

Figure 4. System architecture: STM32H7 Series



1.3.5 Архитектура системы: устройства STM32WB55xx

Архитектура системы устройств STM32WB55xx состоит в основном из 32-битной многослойной матрицы шины AHB, которая соединяет несколько ведущих и подчиненных устройств. QUADSPI отображается на выделенном уровне в матрице шины AHB.

Доступ к QUADSPI может осуществляться соответствующими мастерами, такими как Cortex-M4, либо через S-Bus, либо через I-bus. Доступ к нему можно получить через D-шину, если включена переназначение. QUADSPI также доступен для DMA1 и DMA2.

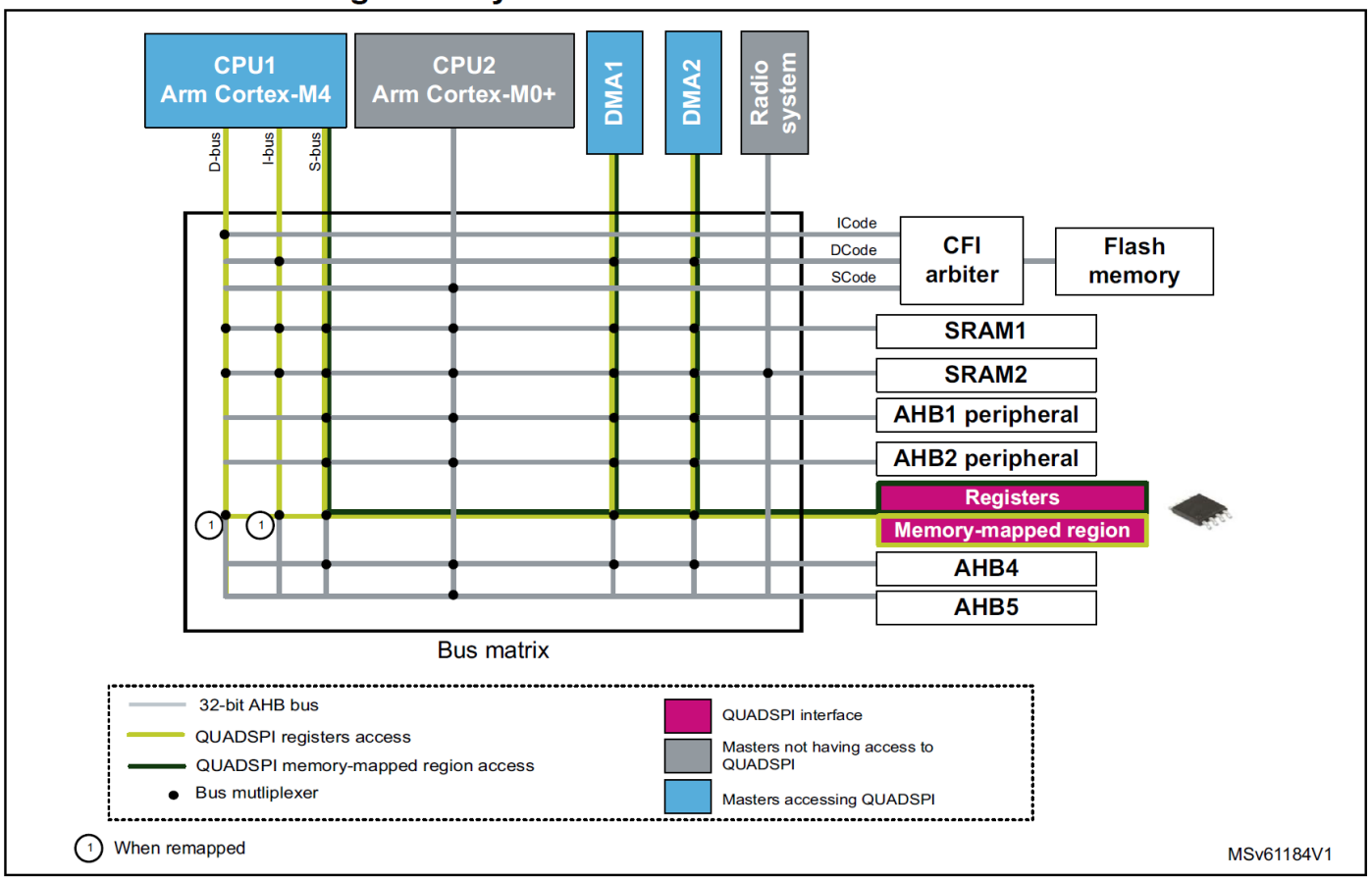
Включение физического переназначения по I-bus и D-bus повышает производительность при выполнении для Cortex-M4.

Доступ к QUADSPI может быть либо доступом к регистрам, либо доступом к области с отображением в памяти:

- Доступ к регистрам может быть сделан Cortex-M4 для конфигурации регистров или передачи данных. Доступ к регистру также может осуществляться с помощью DMA1 и DMA2 для передачи данных.
- Доступ к области отображаемой памяти может быть сделан Cortex-M4 для кода и выборки данных, а также DMA1 и DMA2 для передачи данных.

На следующем рисунке показано соединение QUADSPI в системе устройств STM32WB55xx.

Figure 5. System architecture: STM32WB55xx



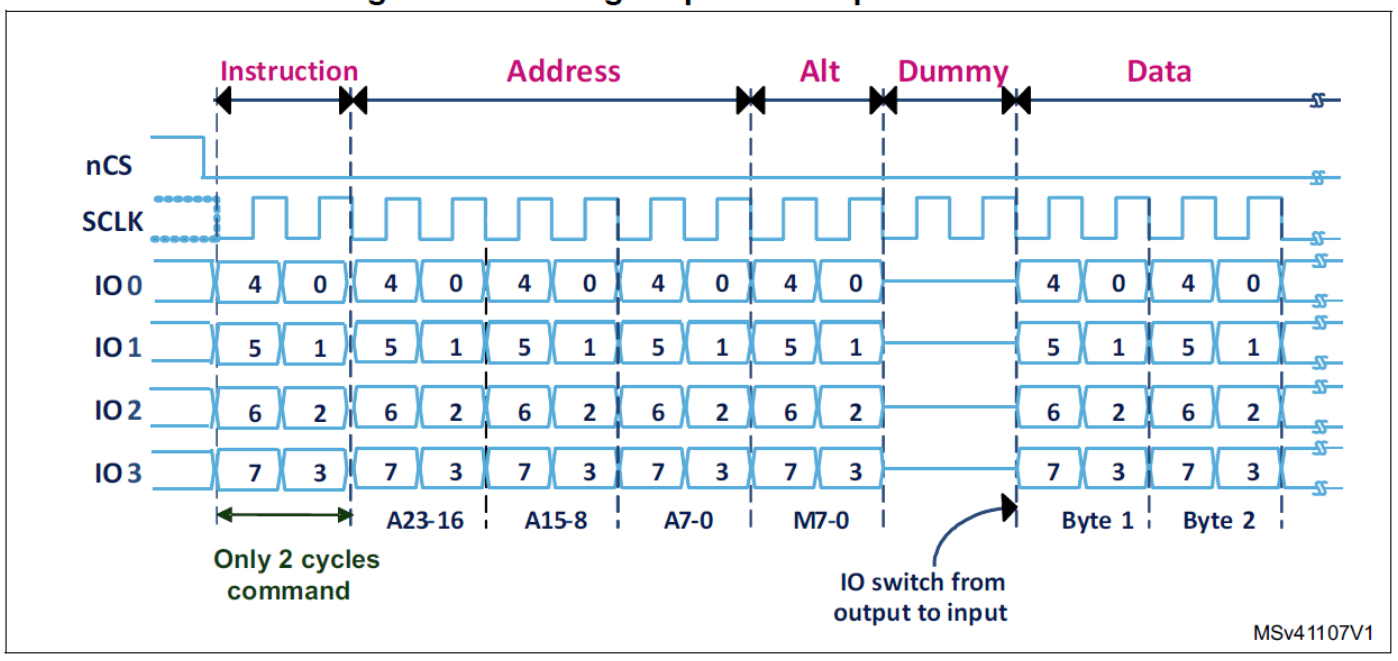
2 Описание интерфейса QUADSPI

2.1 Гибкий формат кадра

Интерфейс QUADSPI обеспечивает полностью программируемый фрейм, состоящий из пяти фаз, где каждая фаза полностью конфигурируется, что позволяет настраивать его отдельно по длине и количеству линий.

Формат кадра может быть настроен только в режиме косвенного доступа или в режиме отображения памяти, но не в режиме опроса флага состояния. На рисунке ниже показано чтение последовательности в режиме Quad I / O SDR.

Figure 6. Reading sequence in quad I/O SDR



2.1.1 Фаза инструкции

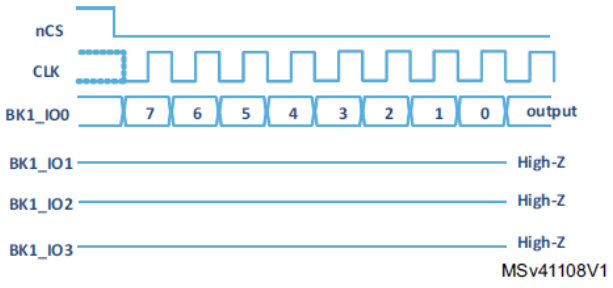
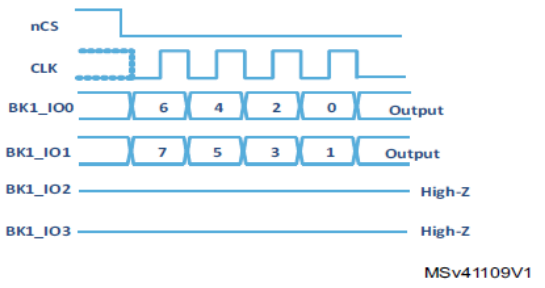
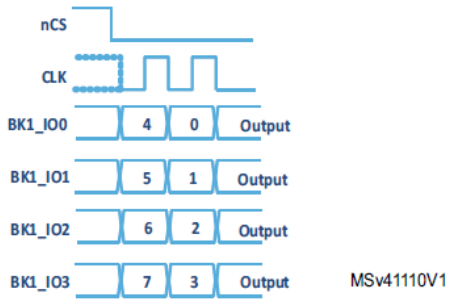
На этом этапе во флэш-память отправляется команда (8-разрядная инструкция), в которой указывается тип выполняемой операции. Эта команда полностью настраивается и позволяет отправлять любое значение. Пользователь может просто написать желаемую команду для отправки в поле INSTRUCTION регистра QUADSPI_CCR [7: 0].

В зависимости от программного обеспечения и конфигурации оборудования, инструкция может быть отправлена в одну, две или четыре строки. В некоторых случаях, когда отправляется только адрес, этап инструкции можно пропустить. В следующей таблице приведены различные конфигурации для этапа инструкции.

Примечание:

Режим DDR не поддерживается на этом этапе, поэтому даже если включен режим DDR, команда всегда отправляется в режиме SDR.

Table 4. Instruction phase configurations

Register configurations		Indirect mode Automatic-polling mode Memory-mapped mode	Command formats
Instruction to be sent in QUADSPI_CCR[7:0]		INSTRUCTION [7: 0]	NA
Instruction phase QUADSPI_CCR[9:8]	No Instruction: skipped	IMODE[1:0] = 00	The instruction phase is skipped
	Instruction on 1 line: Single SPI mode	IMODE[1:0] = 01	
	Instruction on 2 lines: Dual SPI mode	IMODE[1:0] = 10	
	Instruction on 4 lines: Quad-SPI mode	IMODE[1:0] = 11	

2.1.2 Фаза адреса

На этом этапе адрес отправляется во флэш-память с указанием адреса данных для чтения или записи. Фаза адреса полностью настраивается, что позволяет отправлять один, два, три или четыре байта адреса. В косвенном режиме и режиме автоматического опроса пользователь может просто записать нужный адрес в регистр QUADSPI_AR.

В зависимости от программного обеспечения и конфигурации оборудования, адрес может быть отправлен через одну, две или четыре строки. В некоторых случаях, когда адрес не нужен, например, в операции массового стирания, фазу адреса можно пропустить

В таблице ниже приведены различные конфигурации фаз адреса.

Table 5. Address-phase configurations

Register configurations		Indirect mode	Automatic-polling mode	Memory-mapped mode
Address to be sent QUADSPI_AR[31:0]		ADDRESS[31:0]		Address is given directly via the AHB from any master on the bus matrix like Cortex® or DMA
Address size QUADSPI_CCR[13:12]	1-byte	ADSIZE[1:0] =00		
	2-byte	ADSIZE[1:0] =01		
	3-byte	ADSIZE[1:0] =10		
	4-byte	ADSIZE[1:0] =11		
Address phase QUADSPI_CCR[11:10]	No address: skipped	ADMODE[1:0] =00		
	Address on 1 line: Single SPI mode	ADMODE[1:0] =01		
	Address on 2 lines: Dual SPI mode	ADMODE[1:0] =10		
	Address on 4 lines: Quad SPI mode	ADMODE[1:0] =11		

Примечание:

В режиме Dual-Flash памяти, когда DFM = 1, адрес для отправки на Flash1 - это точно такой же адрес, который будет отправлен на Flash

2.2.1.3 Альтернативно-байтовая фаза

Это дополнительная фаза, поддерживаемая интерфейсом QUADSPI, обеспечивающая большую гибкость. Обычно используется для управления режимом работы; например, 1 байт может отправляться непрерывно, чтобы поддерживать устройство Quad-SPI в рабочем режиме. Это поддерживается некоторыми производителями памяти, такими как Spansion, Micron и Macronix, где альтернативный байт отправляется непрерывно, чтобы сохранить память в режиме выполнения на месте.

Фаза альтернативного байта полностью настраивается и позволяет отправлять один, два, три или четыре байта в зависимости от конфигурации файла ABRSIZE [1: 0]. Пользователь может просто записать нужные альтернативные байты в регистр QUADSPI_ABR.

В зависимости от программного и аппаратного обеспечения, альтернативный байт может быть отправлен по одной, двум или четырем строкам. Если в этом нет необходимости, фазу альтернативного байта можно пропустить.

В таблице ниже приведены различные конфигурации фазы альтернативного байта.

Table 6. Alternate-byte phase configurations

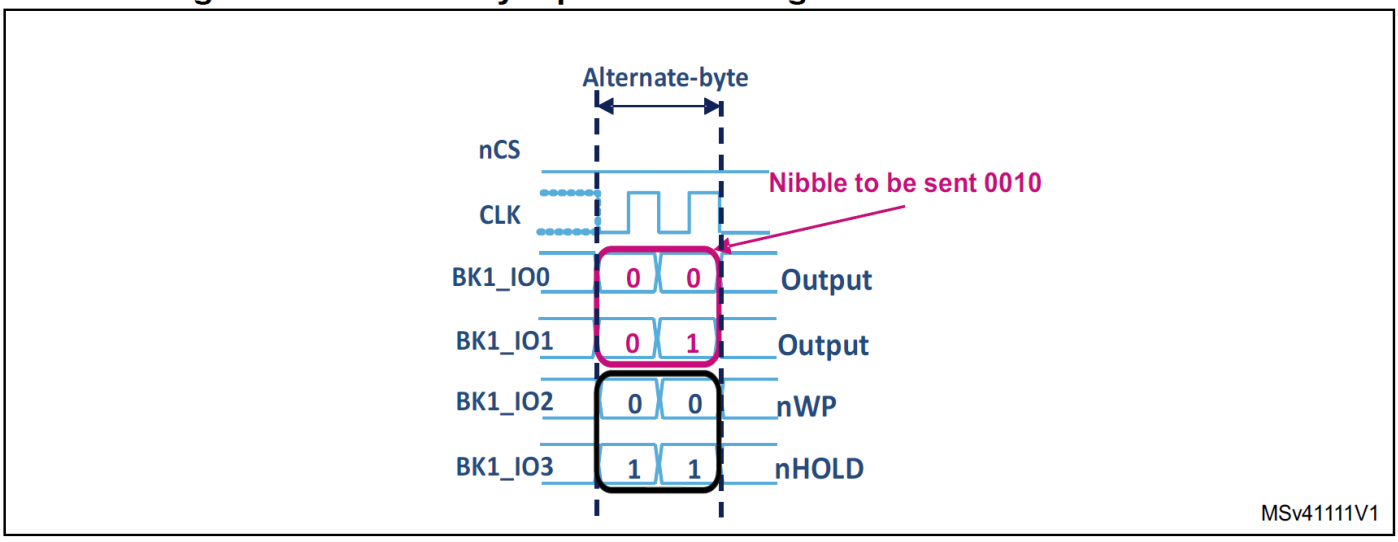
Register configuration		Indirect mode	Automatic -polling mode	Memory-mapped mode
Alternate-byte to be sent QUADSPI_ABR		QUADSPI_ABR		
Alternate-byte size QUADSPI_CCR[17:16]	1-byte	ABSIZE [1:0] =00		
	2-byte	ABSIZE [1:0] =01		
	3-byte	ABSIZE [1:0] =10		
	4-byte	ABSIZE [1:0] =11		
Alternate-byte phase QUADSPI_CCR[15:14]	No alternate-byte: skipped	ABMODE [1:0] = 00		
	Alternate-byte on 1 line: single SPI mode	ABMODE [1:0] = 01		
	Alternate-byte on 2 lines: dual SPI mode	ABMODE [1:0] = 10		
	Alternate-byte on 4 lines: Quad SPI mode	ABMODE [1:0] = 11		

Примечание:

В режиме Dual-Flash памяти, когда DFM = 1, альтернативный байт для отправки во Flash1 точно такой же, как и для отправки во Flash2.

Альтернативно-байтовая фаза: отправка полубайта в режиме Dual-SPI

Figure 7. Alternate-byte phase: sending a nibble in dual-SPI mode



В некоторых случаях необходимо отправлять только один полубайт в фазе альтернативных байтов в течение двух тактовых циклов, а не полный байт в течение четырех тактовых циклов. Например, когда используется режим Dual-SPI и только два цикла используются для фазы альтернативного байта.

Режим Quad I / O может быть активирован только для фазы альтернативного байта, чтобы послать альтернативный байт, где полубайт отправляется через IO0 и IO1, в то время как другой полубайт должен быть отправлен только для поддержания низкого IO2 и высокого IO3 во время альтернативного фаза байта.

2.1.4 Фаза фиктивного цикла

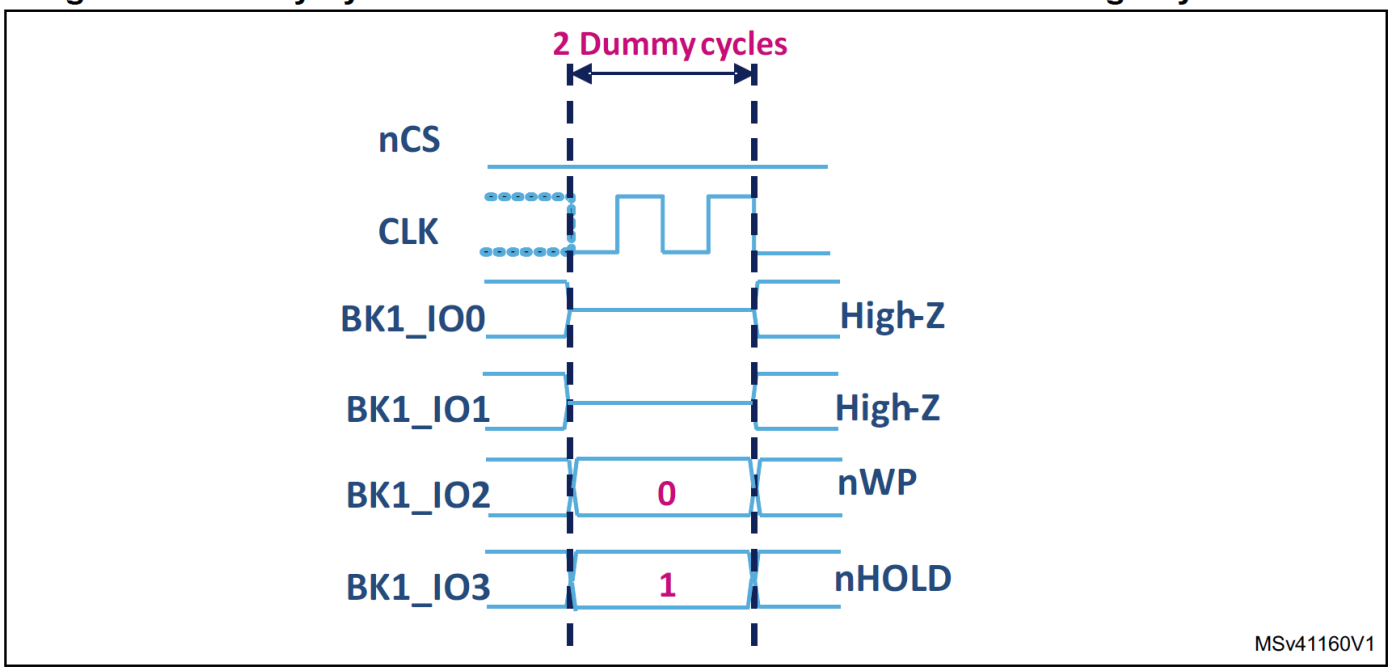
Фиктивная фаза цикла необходима в некоторых случаях при работе на высоких тактовых частотах. Эта фаза позволяет обеспечить достаточное время «оборота» для перевода сигналов данных из режима вывода в режим ввода.

Фиктивная фаза включена путем установки количества фиктивных циклов в DCYC [4: 0] поле QUADSPI_CCR регистрации. Число, определенное в поле DCYC [4: 0], может достигать 31 цикла, и это не зависит от конфигурации используемого оборудования.

Примечание:

В режиме SDR или DDR фиктивная фаза всегда представляет один тактовый цикл QUADSPI. Во время этой фазы, если используется аппаратная конфигурация QUADSPI и если фазы связи находятся в режимах Quad-SPI или Dual-SPI, IO2 принудительно устанавливается в «0». отключить функцию «защиты от записи», когда IO3 принудительно установлен на «1», чтобы отключить функцию «удержания» памяти QSPI. Это полностью управляется аппаратным обеспечением (периферийное устройство QUADSPI), поэтому пользователю ничего не нужно настраивать.

Figure 8. Dummy-cycle: IO2 maintained low and IO3 maintained high by hardware



2.1.5 Фаза данных

На этом этапе; данные отправляются или принимаются из или в память QSPI. Фаза данных полностью настраивается и позволяет отправлять, получать или оба байта любого количества на или с устройства памяти QSPI.

В режиме косвенного доступа и в режиме автоматического опроса число байтов, которые нужно отправить, получить или оба байта, указывается в регистре QUADSPI_DLR.

В режиме косвенной записи данные, подлежащие отправке во флэш-память, должны быть записаны в регистр QUADSPI_DR, в то время как в режиме косвенного чтения данные, полученные из флэш-памяти, получены путем чтения из регистра QUADSPI_DR.

В режиме отображения памяти данные могут быть считаны только с запоминающего устройства, но не записаны, а затем доступ к данным осуществляется непосредственно из QUADSPI FIFO. Все мастера на матрице шины могут считывать данные с устройства памяти QSPI, как если бы это была внутренняя память.

В зависимости от программного обеспечения и конфигурации оборудования, передача данных может выполняться в одну, две или четыре строки. В некоторых случаях использования, когда данные не нужны, например, операция стирания, этап данных может быть пропущен.

Следующая таблица суммирует конфигурацию фазы данных в различных функциональных режимах.

Table 7. Data phase configuration versus Quad-SPI functional modes

Register configuration		Indirect mode	Automatic-polling mode	Memory-mapped mode
Data	Read data	QUADSPI_DR		Data read is sent back directly over the AHB to any master on the bus matrix requesting for reading operation (Cortex®, DMA LTDC, DMA2D...).
	Write data	QUADSPI_DR		Not supported
Number of data to be sent/received	QUADSPI_DLR	1-byte 0x00000000 to undefined(1) 0xFFFFFFFF	1-byte 0x00000000 to 4-bytes 0x00000003	QUADSPI_DLR has no meaning in this mode. If DMA is used, number of data to be read is set only in DMA's DMA_SxNDTR register.
Data phase QUADSPI_CCR[15:14]	No Data: skipped	DMODE[1:0] = 00(2)		
	Data on 1 line: Single SPI mode	DMODE[1:0] = 01		
	Data on 2 lines: Dual SPI mode	DMODE[1:0] = 10		
	Data on 4 lines: Quad SPI mode	DMODE[1:0] = 11		

1. Когда QUADSPI_DLR = 0xFFFFFFFF, количество байтов, которые должны быть отправлены или получены, не определено, поэтому передача продолжается до конца памяти, как определено в FSIZE. Когда QUADSPI_DLR = 0xFFFFFFFF и FSIZE = 0x1F, передача продолжается бесконечно, останавливаясь только после запроса на прерывание или после отключения Quad-SPI. После считывания последнего адреса памяти (по адресу 0xFFFFFFFF) чтение продолжается с адреса = 0x00000000.

2. Этот режим должен использоваться только в Косвенном режиме.

2.2 Несколько аппаратных конфигураций

Микроконтроллеры STM32 предлагают очень гибкий интерфейс QUADSPI, который позволяет подключать внешние запоминающие устройства QSPI в различных аппаратных конфигурациях. Затем пользователь может выбрать свою собственную конфигурацию.

В зависимости от конфигурации используемого оборудования количество используемых GPIO может быть до 11. В таблице ниже приведены различные варианты использования.

Table 8. Hardware configurations versus used GPIO number

-	-	Single-Flash mode		Dual-Flash memory mode
Single/Dual SPI mode	Used GPIOs	Bank1 CLK BK1_IO0/SO BK1_IO1/SI BK1_nCS	Bank2 CLK BK2_IO0/SO BK2_IO1/SI BK2_nCS	CLK BK1_IO0/SO BK1_IO1/SI BK2_IO0/SO BK2_IO1/SI BK1_nCS(1) BK2_nCS(1)
	GPIOs number	4 GPIOs		6 or 7 GPIOs
Quad-SPI mode	Used GPIOs	Bank1 CLK BK1_IO0/SO BK1_IO1/SI BK1_IO2 BK1_IO3 BK1_nCS	Bank2 CLK BK2_IO0/SO BK2_IO1/SI BK2_IO2 BK2_IO3 BK2_nCS	CLK BK1_IO0/SO BK1_IO1/SI BK1_IO2 BK1_IO3 BK2_IO0/SO BK2_IO1/SI BK2_IO2 BK2_IO3 BK1_nCS(1) BK2_nCS(1)
	GPIOs number	6 GPIOs		10 or 11 GPIOs

1. В режиме Dual-Flash можно использовать один выбор чипа: BK1_nCS или BK2_nCS. Более подробную информацию о режиме двойной вспышки см. В разделе 2.2.4. Режим Dual-Flash на стр. 25.

Примечание:

Если ни одна из фаз не настроена на использование режима Quad-SPI, то GPIO, соответствующие IO2 и IO3, можно использовать для других функций, даже когда QUADSPI активен.

2.2.1 Режим с одним SPI (классический SPI)

Это классический SPI, в котором используются только четыре GPIO, а данные отправляются по линии SO и принимаются по линии SI.

Примечание:

Полнодуплексная передача не поддерживается.

Строки IO2 и IO3 являются необязательными:

• При использовании (IO2 и IO3 подключены к памяти QSPI): контакты IO2 и IO3 должны быть сконфигурированы как для IO0 и IO1. Чтобы разрешить связь с устройством памяти:

–IO2 находится в режиме вывода и принудительно установлен в «0», чтобы отключить функцию «защиты от записи»

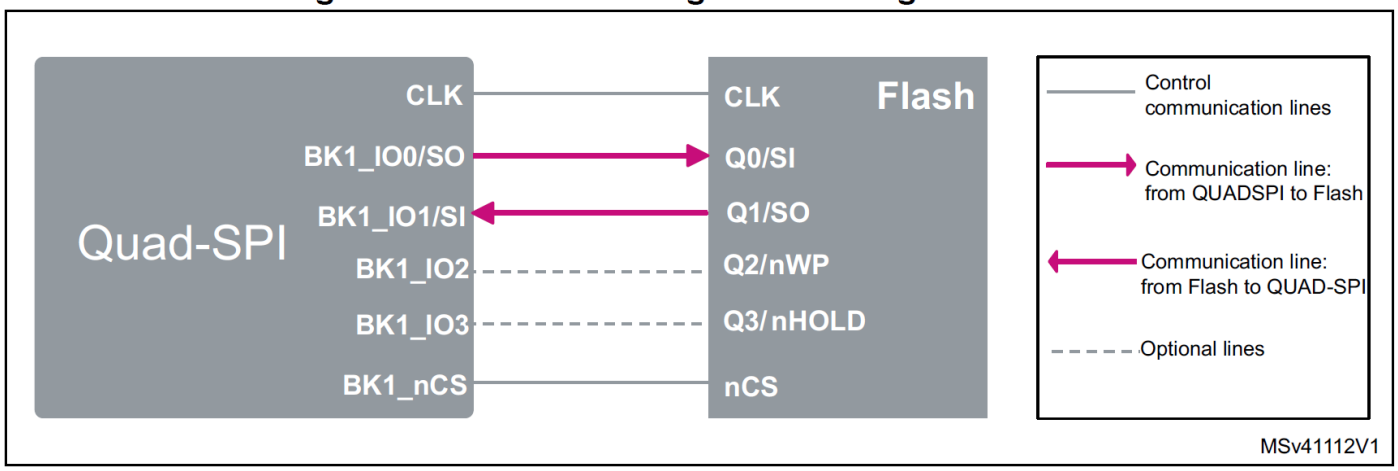
–IO3 находится в режиме вывода и принудительно установлен на «1», чтобы отключить функцию «удержания»

–Это управляется аппаратным обеспечением (Quad-SPI периферийное устройство) на всех этапах связи

• Если они не используются, контакты устройства памяти nWP и nHOLD должны быть подключены соответственно к VDD и VSS, в то время как контакты IO2 и IO3 могут использоваться для других целей.

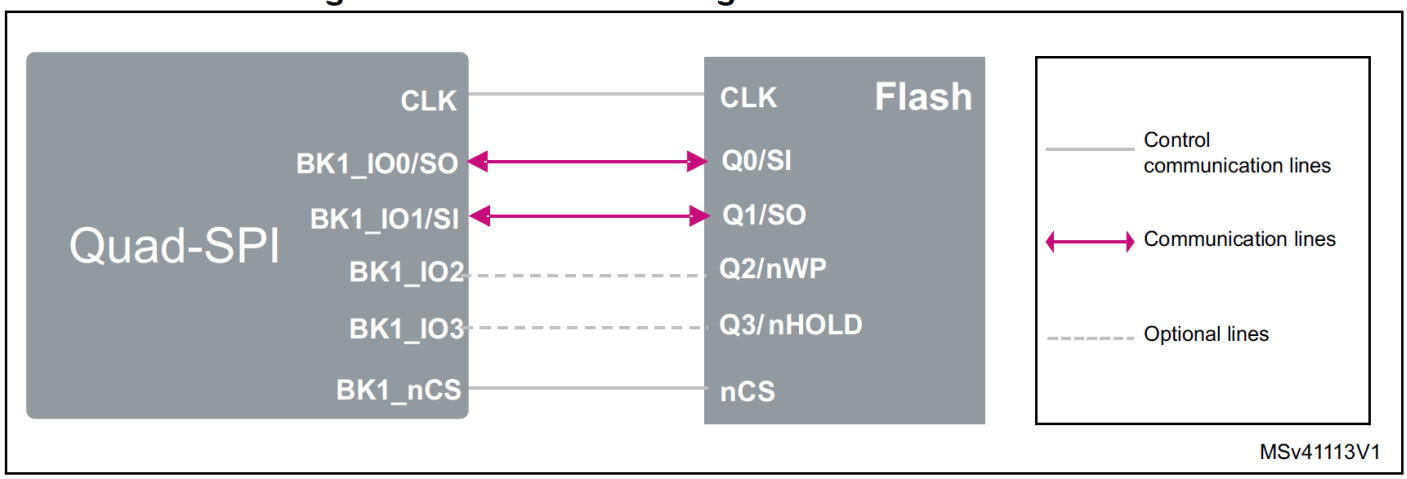
В этом режиме все фазы, такие как инструкция, адрес, альтернативный байт и данные, должны быть сконфигурированы в режиме с одним SPI, установив для полей IMODE / ADMODE / ABMODE / DMODE в QUADSPI_CCR значение 01.

Figure 9. Hardware configuration: Single-SPI mode



2.2.2 Режим Dual-SPI

Figure 10. Hardware configuration: dual-SPI mode



В режиме Dual-SPI аппаратная конфигурация аналогична конфигурации в одиночном режиме, но здесь для данных используются две строки, это означает, что данные отправляются и принимаются в две строки. Что касается режима Single-SPI, линии IO2 и IO3 являются дополнительными, если они не использу-

ются, контакты устройства nWP и nHOLD должны быть подключены соответственно к VDD и VSS.

В этом режиме все команды, адреса, альтернативные байты и фазы данных должны быть сконфигурированы в режиме Dual-SPI путем установки полей IMODE / ADMODE / ABMODE / DMODE в QUADSPI_CCR равными 10.

2.2.3 Режим Quad-SPI

В режиме Quad-SPI используются шесть контактов: четыре контакта для данных и два контакта для тактовой частоты и выбора микросхемы. В этой конфигурации оборудования можно использовать режим Single или Dual-SPI. В режиме Quad-SPI данные передаются, принимаются или оба по четырем линиям.

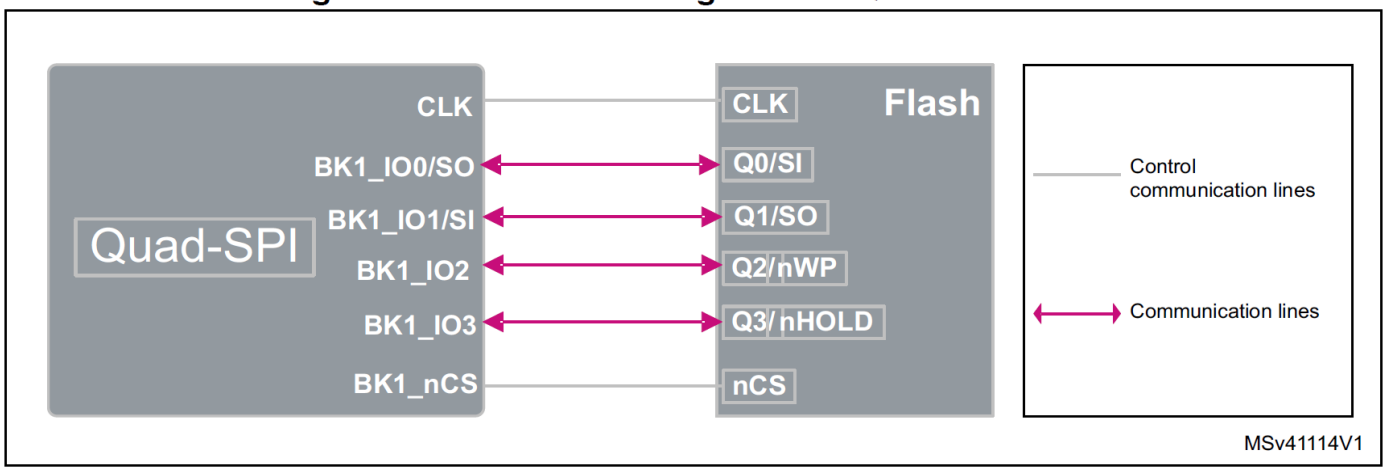
Примечание:

В этом режиме функции удержания и WP больше не доступны, поскольку IO3 и IO4 используются для связи.

В режиме Quad-SPI, в зависимости от марки памяти QSPI, пользователь может выбрать отправку каждой фазы в одиночном, двойном или четырехуровневом режиме. Многие производители памяти поддерживают следующие конфигурации, где Command-Address-Data: 1-1-4; 1-4-4; 4-4-4.

В общем случае, если адрес отправляется в четыре строки, то альтернативный байт также должен быть отправлен в четыре строки.

Figure 11. Hardware configuration: Quad-SPI mode



2.2.4 Режим двойной флэш-памяти

В режиме Dual-Flash памяти MCU одновременно обменивается данными с двумя внешними устройствами памяти. Этот режим полезен для удвоения пропускной способности и удвоения размера при использовании только 10 GPIO: восемь для данных, один чип для обоих устройств и один для CLK. Пропускная способность в два байта за цикл может быть достигнута с двойной флэш-памятью в режиме DDR Quad-SPI.

В этом режиме для обоих устройств можно использовать только один чип-выбор, а затем сохранить один GPIO для других целей, и к обоим устройствам можно подключить либо nCS_BK1, либо nCS_BK2. Часы должны быть подключены к обоим устройствам.

Разрешены различные аппаратные конфигурации, обеспечивающие высокую гибкость для пользователя. Таблица 8 на стр. 23 иллюстрирует все возможные конфигурации оборудования.

Режим двойной памяти позволяет удвоить пропускную способность, так как один байт может быть отправлен или получен в каждом цикле. Этот режим очень интересен, когда требуется больше производительности; не только удваивается пропускная способность в режиме Dual-Flash памяти, но также удваивается объем внешней памяти.

При использовании двух внешних модулей памяти QSPI размер, который необходимо настроить в FSIZE [4: 0], должен отражать общую емкость флэш-памяти, которая в два раза превышает размер одного отдельного компонента.

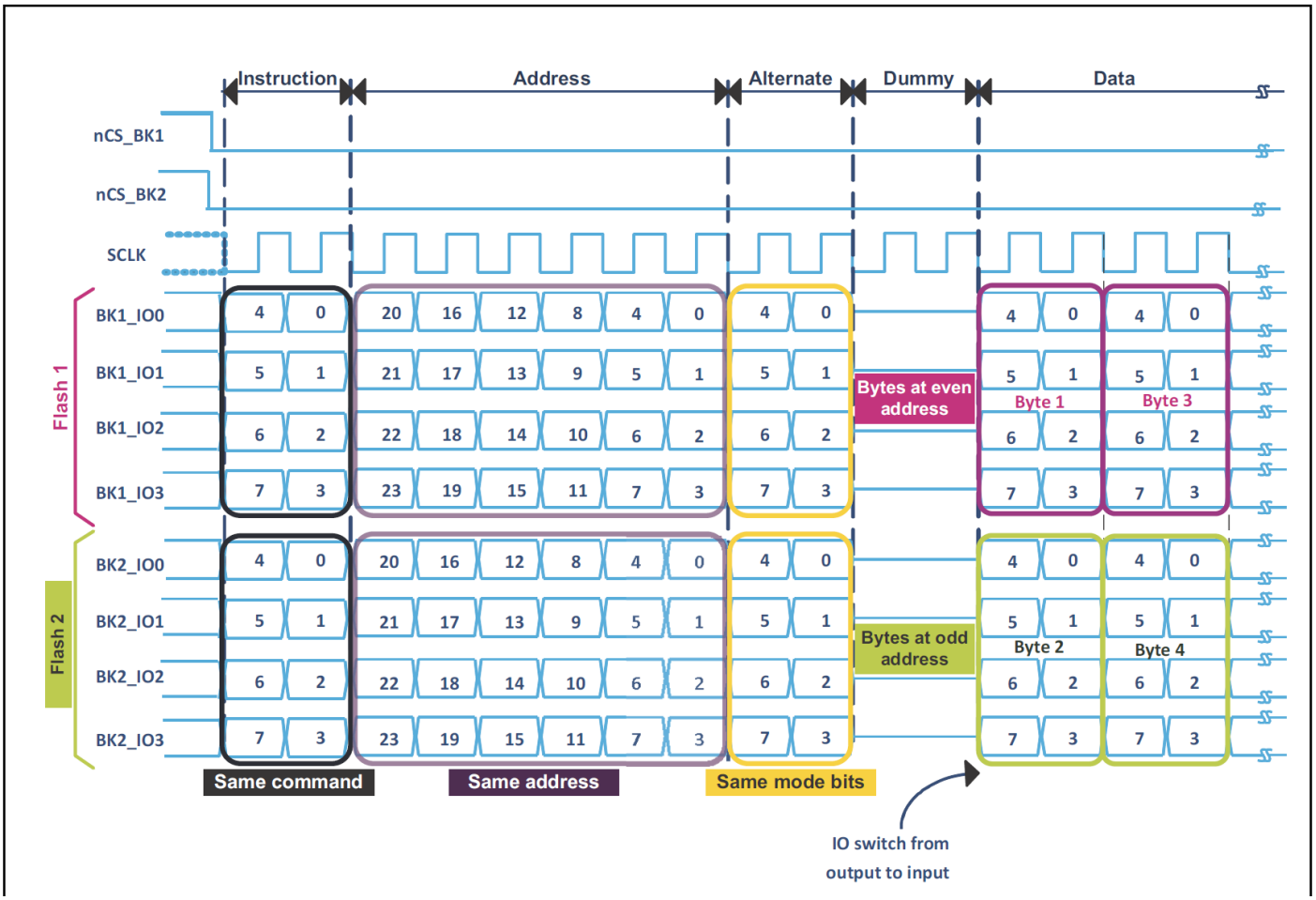
Для поддержки пакетов с двумя матрицами с двумя чипами и двумя устройствами QSPI размер FIFO всегда составляет 32 байта в режиме памяти с одной или двумя флеш-памятью.

Примечание:

Адресное пространство в режиме отображения памяти составляет до 256 Мбайт в режиме памяти с одной или двумя запоминающими устройствами.

На следующем рисунке показан пример последовательности чтения в режиме Quad I / O SDR с двойной флэш-памятью.

Figure 12. Read sequence in dual-Flash memory Quad I/O SDR mode

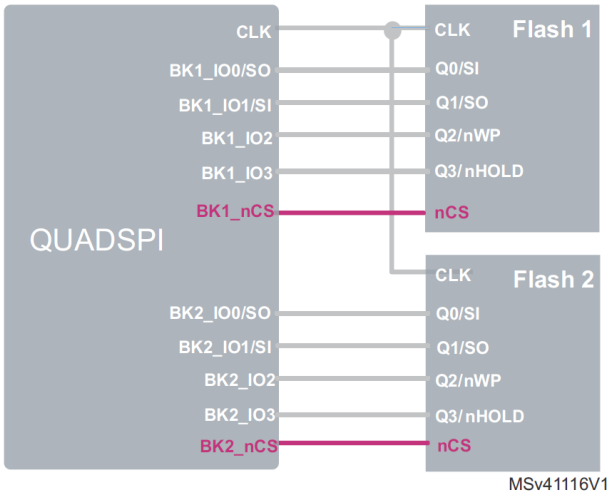
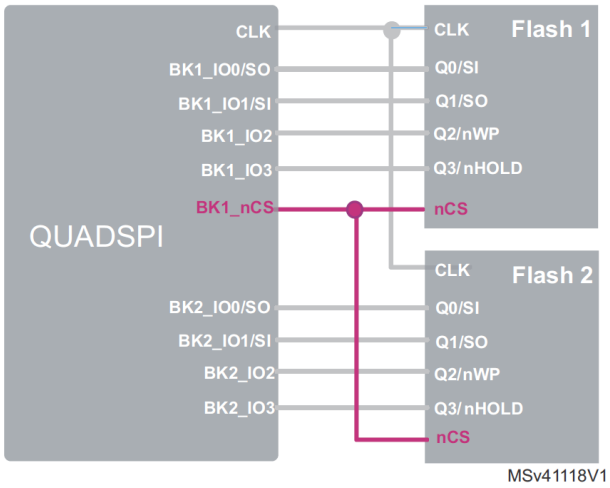
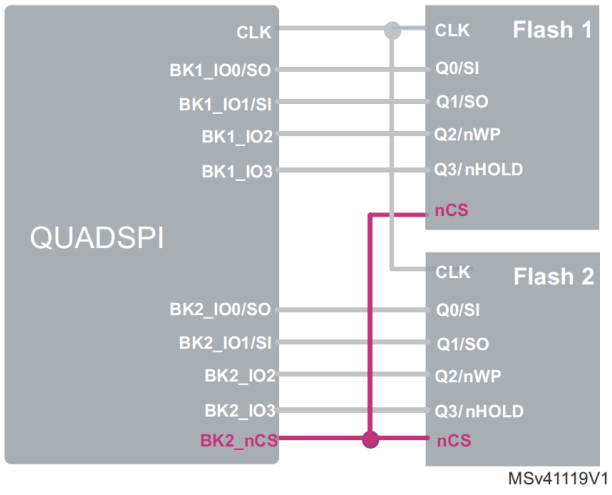


Обратите внимание, что все байты по четным адресам хранятся во Flash 1, а все байты по нечетным адресам хранятся во Flash 2. Как описано на рисунке 12, в режиме двойной флэш-памяти одна и та же команда, адрес и альтернатива отправляются как во Flash 1, так и во Flash 2. Например, чтобы прочитать первые четыре байта в режиме отображения памяти с двумя флэш-памятьми от 0x90000 000 до 0x9000 0003, периферийное устройство QUADSPI выполняет следующую последовательность действий:

• Адрес 0x0000 0000 отправляется на флэш-память, а байт 1 (по четному адресу 0x9000 0000) читается из Flash 1, а байт 2 (по нечетному адресу 0x9000 0001) читается из Flash 2.

• Затем адрес 0x0000 0001 отправляется на флэш-память, а байт 3 (по четному адресу 0x9000 0002) читается из Flash 1, а байт 2 (по нечетному адресу 0x9000 0003) читается из Flash 2.

Table 9. Dual-Flash memory hardware configurations

<table>
<tr><th>Used nCS</th><th>nCS configuration</th><th colspan="2">Flash mode</th><th>Hardware configuration</th></tr>
<tr><td rowspan="2">2 nCS enabled</td><td rowspan="2">Both nCS_BK1 and nCS_BK2 not connected together</td><td rowspan="2">Single Flash DFM = 0(1)</td><td>FSEL = 0
Flash 1 enabled</td><td rowspan="2"> 
</td></tr>
<tr><td>FSEL = 1
Flash 2 enabled</td></tr>
<tr><td rowspan="3">1 nCS enabled</td><td>nCS_BK1 connected to both devices</td><td colspan="2">Dual-Flash memory DFM = 1</td><td> 
</td></tr>
<tr><td>nCS_BK2 connected to both devices</td><td colspan="2">Dual-Flash memory DFM = 1</td><td> 
</td></tr>
</table>

Предостережения:

• В режиме двойной флэш-памяти обе модели устройств должны быть идентичны, поскольку в этом режиме одни и те же команды и адреса выдаются параллельно обеим флэш-памяти; это позволяет удвоить доступный размер внешней флэш-памяти QUADSPI. В случае, если два устройства флэш-памяти различаются, режим двойной флэш-памяти должен быть отключен (DFM = 0), и каждая флэш-память может использоваться автономно, что позволяет включить Flash 1 или Flash 2 с помощью QUADSPI_CR [7] FSEL немного.

• Для всех конфигураций оборудования, перечисленных в таблице ниже, каждое устройство памяти настроено в режиме Quad-SPI. Можно подключить каждое устройство в режиме Single или Dual-SPI. Если DFM = 1, оба устройства должны быть настроены одинаково. Это позволяет удвоить доступный внешний размер данных и пропускную способность.

• Размер флэш-памяти, указанный в FSIZE [4: 0] (QUADSPI_DCR [20:16]), должен отражать общую емкость флэш-памяти, которая является двойной величиной размера одного отдельного компонента.

2.2.5 Режимы DDR и SDR

Режим SDR активирован по умолчанию, режим DDR позволяет производить выборку с нарастающим и убывающим фронтом каждого тактового цикла и дает возможность удвоить пропускную способность. Режим DDR позволяет повысить пропускную способность и производительность исполнения; это также очень полезно, когда системные часы (HCLK) низки и не позволяют QUADSPI работать с максимальной скоростью.

• SDR: данные отправляются по переднему фронту CLK и отбираются по переднему фронту CLK.

• DDR: данные, отправленные на обоих фронтах CLK, во время адресации, альтернативной фазы или фазы данных. Образец сделан половиной CLK позже.

При использовании DDR (двойная скорость передачи данных), также известная как DTR (двойная скорость передачи), пользователь должен учитывать следующее:

• Запуск связи и процедура конфигурирования такие же, как в SDR.

• Команда отправляется каждый такт, как в режиме SDR.

• Альтернативные фазы, данные и адреса отправляются по обоим краям часов.

• Фиктивные циклы подсчитываются для каждого тактового цикла, как в режиме SDR.

2.3 Три режима работы

2.3.1 Indirect mode (косвенный режим)

Косвенный режим используется в следующих случаях:

• Для чтения, записи или удаления операций

• Если мастерам AHB не требуется автономный доступ к памяти QSPI (доступно в режиме отображения памяти)

• Для всех операций, выполняемых через регистры данных Quad-SPI с использованием ЦП или DMA

• Для настройки флэш-памяти QSPI. В косвенном режиме все операции выполняются через регистр QUADSPI, в котором операции чтения и записи доступны и управляются программным обеспечением. Интерфейс QUADSPI ведет себя как классический интерфейс SPI. Переданные данные проходят через регистр данных с FIFO. Обмен данными осуществляется программным обеспечением или DMA с использованием соответствующих флагов прерываний в регистрах состояния QUADSPI.

Операции чтения и записи всегда выполняются в пакете, если объем данных не равен единице. Количество данных для передачи задается в регистре QUADSPI_DLR. В этом режиме можно считывать или записывать данные с или на внешнюю флэш-память объемом до 4 Гбайт.

Режим автоматического опроса доступен для генерации прерывания при изменении регистра состояния во флэш-памяти (полезно для проверки конца стирания или окончания программирования). В случае операции стирания или программирования, режим косвенного должен использоваться, и все операции должны выполняться программным обеспечением.

В этом случае рекомендуется использовать режим опроса состояния, а затем опрашивать регистр состояния во флэш-памяти, чтобы узнать, когда завершится программирование или операция стирания.

2.3.2 Satus-flag polling mode (режим опроса статуса-флага)

Режим опроса Status-flag используется в следующих случаях:

• Для чтения регистра состояния флэш-памяти QSPI

• Для автономного опроса о завершении операции: QUADSPI опрашивает регистр состояния в памяти

Интерфейс может автоматически опрашивать указанный регистр в памяти и освобождать ЦП от этой задачи (полезно, например, при опросе конца программного флага). Это режим для проверки, например, когда операция стирания завершена, и чтобы узнать, что может быть сгенерировано прерывание.

Интерфейс QUADSPI также можно настроить для периодического чтения с определенной скоростью регистра во флэш-памяти QSPI. Возвращенные данные могут быть замаскированы, чтобы выбрать биты для оценки. Выбранные биты сравниваются по битам с их необходимыми значениями, хранящимися в регистре совпадений. Сравнение результатов может быть обработано двумя способами:

• Режим ANDed: если все выбранные биты совпадают, при успешном завершении генерируется прерывание (флаг остановки при совпадении)

• Режим ORed: если один из выбранных битов совпадает, при успешном завершении генерируется прерывание (флаг остановки на совпадении)

При совпадении интерфейс QUADSPI может автоматически останавливаться. Команда READ STATUS REGISTER используется многими производителями памяти как Micron или Spansion для постоянного считывания регистра состояния.

2.3.3 Memory-mapped mode (режим отображения памяти)

Режим отображения памяти используется в следующих случаях:

• Для операций чтения

• Использовать внешнюю флэш-память QSPI как внутреннюю память, чтобы любой мастер AHB мог читать данные автономно

• Для выполнения кода из внешней флэш-памяти QSPI.

В режиме отображения памяти внешняя память рассматривается системой как внутренняя память. Этот режим позволяет всем мастерам AHB обращаться к памяти QSPI как к внутренней памяти. Процессор также может выполнять код из памяти QSPI.

При использовании режима отображения в память механизм предварительной выборки, полностью управляемый аппаратным обеспечением, позволяет оптимизировать производительность чтения и выполнения из внешней памяти QSPI. Учитывая, что все фазы связи, такие как отправка кода операции или адреса, управляются периферийным устройством QUADSPI, для предварительной выборки используется 32-байтовый FIFO (16 байтов для серии STM32L4); Этот оптимизированный механизм предварительной выборки позволяет избежать затрат на программное обеспечение.

Запрограммированные инструкции и кадр отправляются автоматически, когда мастер AHB получает доступ к отображенному в памяти пространству. Как только периферийное устройство QUADSPI сконфигурировано, доступ к памяти QSPI осуществляется, как только на AHB появляется запрос на чтение; это делается в диапазоне адресов памяти QSPI. Это действие полностью прозрачно для пользователя.

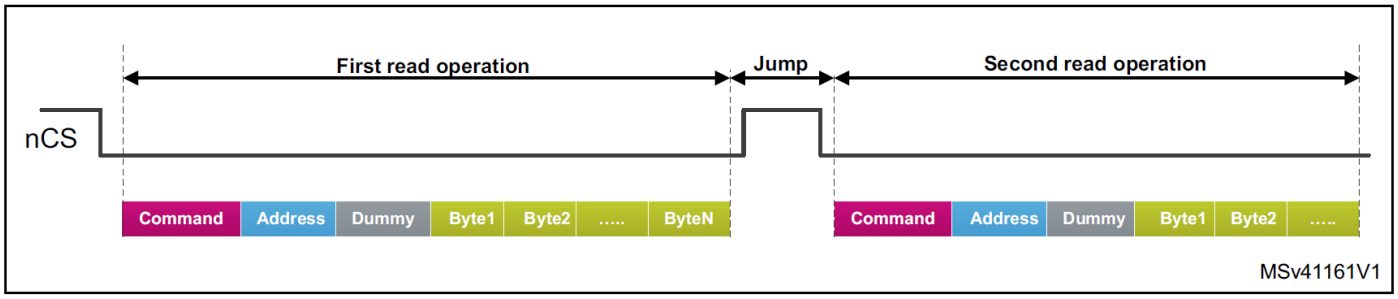
Например, мастер LTDC может автономно обращаться к внешней флэш-памяти, где все операции доступа полностью управляются интерфейсом QUADSPI. Тем временем процессор Cortex®-M выполняет код из внутренней флэш-памяти.

Интерфейс QUADSPI может управлять до 256 Мбайт памяти, начиная с 0x90000000 до 0x9FFFFFFF в режиме отображения памяти.

Execute in place (XIP) (выполнить на месте (XIP))

Буфер предварительной выборки поддерживает выполнение на месте, поэтому код может быть выполнен непосредственно из внешней памяти QSPI. QUADSPI ожидает следующего доступа к ЦП и заранее загружает байт по следующему адресу. Если последующий доступ действительно осуществляется по непрерывному адресу, доступ завершается быстрее, поскольку значение уже предварительно выбрано.

Figure 13. Executing non-sequential code from Quad-SPI



Booting from QSPI Flash memory (загрузка из флэш-памяти QSPI)

Загрузка из памяти QSPI не поддерживается, но пользователь может загрузиться с внутренней флэш-памяти, а затем сконфигурировать QUADSPI в режиме отображения памяти, а затем выполнение начнется из памяти QSPI. Подробнее о выполнении из внешней памяти QSPI см. В разделе 5.2 на стр. 67.

Примечание:

Чтение QUADSPI_DR в режиме отображения памяти не имеет смысла и возвращает 0.

Для всех поддерживаемых устройств STM32 память QSPI доступна Cortex®-M по системной шине. Для серии STM32L4 QUADSPI также доступен через шины I-Code и D-Code, когда физическое переназначение включено по адресу 0, что позволяет повысить производительность выполнения.

Когда QUADSPI переназначается по адресу 0x00000000, переопределяется только 128 Мбайт. Даже при наличии псевдонима в области загрузочной памяти, память QSPI все еще доступна в своей исходной области памяти.

Примечание:

Регистр длины данных QUADSPI_DLR не имеет значения в режиме отображения памяти.

2.4 Особенности

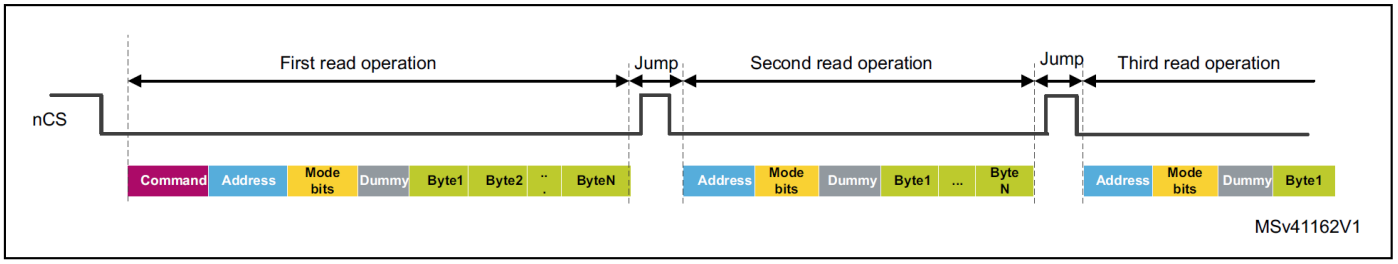
2.4.1 Отправить инструкцию только один раз (SIOO)

Функция SIOO также названа некоторыми производителями памяти как «режим непрерывного чтения», «пакетный режим» или даже как «режим повышенной производительности». Эта функция доступна для всех режимов Quad-SPI: косвенный, автоматический опрос и отображение в память. Рекомендуется использовать эту функцию, чтобы уменьшить накладные расходы команд и повысить производительность выполнения. Когда SIOO включен, команда отправляется только один раз при запуске операции чтения, тогда отправляется только адрес.

Команда отправляется только сначала при запуске операции чтения. Если происходит новая операция чтения, отправляется только один адрес; это действие позволяет сократить до восьми циклов (в режиме одиночного ввода-вывода) для команды. Это очень интересная функция для снижения накладных расходов на доступ к памяти QSPI.

Данные предварительно выбираются непрерывно, пока FIFO не заполнен, когда обнаружен прерывистый доступ, QUADSPI увеличивает выбор микросхемы и начинает новую операцию чтения, не отправляя команду, но отправляя непосредственно новый адрес.

Figure 14. Executing non-sequential code from QUADSPI with SIOO enabled



Функция SIOO поддерживается многими производителями памяти QSPI, такими как Micron, Spansion и Macronix, тем не менее, прежде чем использовать ее, пользователь должен проверить, поддерживается ли эта функция используемой памятью.

Чтобы включить режим SIOO, пользователь должен:

• Настройте память, войдя в режим SIOO. Более подробную информацию о том, как войти в этот режим, см. В техническом описании соответствующего производителя (убедитесь, что используемая команда чтения поддерживает этот режим). Обратите внимание, что для сохранения устройства в этом режиме необходимо отправить альтернативный байт (биты режима). Обратитесь к примеру SIOO в Разделе 5.2 на странице 67 для более подробной информации о включении этой функции.

• Настройте периферийное устройство QUADSPI, установив бит SIOO в регистре QUADSPI_CCR.

2.4.2 Delayed data sampling (отсроченная выборка данных)

Для операций чтения из внешней памяти выборка отсроченных данных полезна, когда сигналы задерживаются из-за ограничений на оптимизацию компоновки печатной платы; следовательно, оптимизация компенсирует эту задержку. Часы выборки могут быть сдвинуты на дополнительный полупериод после того, как данные управляются флэш-памятью, это сделано для того, чтобы гарантировать, что данные готовы в момент выборки. Эта функция не поддерживается в режиме DDR. Чтобы включить сдвиг выборки, установите бит SSHIFT в регистре QUADSPI_CR.

Для операций записи из внешней памяти в режиме DDR выходные данные могут быть сдвинуты на одну четверть тактового цикла вывода QUADSPI, чтобы ослабить ограничения удержания. Чтобы включить эту задержку выходных данных, установите бит DHHC в регистре QUADSPI_CR.

Для получения более подробной информации о характеристиках синхронизации QUADSPI обратитесь к техническому описанию соответствующих продуктов.

2.4.3 imeout counter (счетчик тайм-аута)

Счетчик тайм-аута может быть использован для уменьшения энергопотребления памяти QSPI путем освобождения nCS и последующего перевода памяти в состояние с более низким потреблением.

После каждого доступа в режиме отображения памяти QUAD-SPI предварительно выбирает последующие байты и хранит эти байты в FIFO. Когда FIFO заполнен, коммуникационные часы останавливаются, но вывод nCS остается низким, чтобы сохранить выбранную флэш-память и не отправлять полную команду для чтения следующих байтов, когда местоположение доступно в FIFO.

Чтобы избежать какого-либо дополнительного энергопотребления во внешней флэш-памяти, когда часы останавливаются на длительное время, счетчик тайм-аута может освободить вывод nCS; это действие переводит внешнюю флэш-память в состояние с более низким потреблением после истечения времени ожидания без доступа. Как только FIFO становится пустым, nCS становится низким и разрешает операции чтения.

Чтобы использовать счетчик времени ожидания, пользователь должен:

• Включите его, установив бит TCEN в регистр QUADSPI_CR

• Запрограммируйте период ожидания TIMEOUT [15: 0] в регистре QUADSPI_LPTR.

Примечание:

Когда счетчик тайм-аута включен, например, в режиме отображения памяти, если тайм-аут происходит, nCS повышается; и для любого нового доступа для чтения новая полная команда команд чтения запускается интерфейсом QUADSPI.

2.4.4 Aditional status bits (дополнительные биты состояния)

Помимо флагов состояния, описанных в Таблице 13, регистр состояния QUADSPI_SR включает в себя дополнительные биты состояния, в таблице ниже приведены флаги состояния.

Table 10. Additional status bits

Имя	Размер	Описание
FLEVEL	5 bits (4 bits for STM32L4 Series)	Количество действительных байтов, хранящихся в FIFO (для косвенного режима)
BUSY	1 bit	Этот бит устанавливается, когда операция продолжается. Сбрасывается автоматически, когда операции завершены, а FIFO пуст.

2.4.5 Busy bit (бит занят) и функциональность отмены

BUSY bit

Бит BUSY используется для указания состояния интерфейса QUADSPI, он устанавливается в регистр QUADSPI_SR, когда операция продолжается, и автоматически очищается, когда операции завершаются или прерываются.

Примечание:

Некоторые регистры QUADSPI не могут быть записаны, когда установлен бит BUSY, поэтому пользователь должен проверить, сбрасывается ли он, перед записью в регистры. Обратитесь к соответствующему справочному руководству, чтобы проверить, может ли быть записан регистр или нет, когда установлен BUSY.

В следующей таблице приведены различные случаи сброса бита BUSY в разных режимах работы QUADSPI:

Table 11. BUSY bit reset in different Quad-SPI modes

QSPI mode	BUSY bit reset
Indirect mode	– QUADSPI выполнил запрошенную последовательность команд, а FIFO пуст – Из-за прерывания операции
Automatic-polling mode	– После завершения последнего периодического доступа из-за совпадения, когда APMS = 1 – Из-за прерывания операции
Memory-mapped mode	– В случае тайм-аута – Из-за прерывания операции – QUADSPI периферийное устройство отключено

ABORT bit

Когда приложение работает, любая текущая операция QUADSPI может быть прервана путем установки бита ABORT в регистре QUADSPI_CR. Как только прерывание завершено, бит BUSY и бит ABORT автоматически сбрасываются, и FIFO сбрасывается. Если прерывание происходит в текущей операции пакета

AXI / AHB, QUADSPI позволяет текущему пакету завершиться правильно перед сбросом бита BUSY и бита ABORT.

Примечание:

Некоторые флэш-памяти могут работать неправильно, если операция записи в регистры состояния прервана.

2.4.6 4 byte address mode (4-байтовый адресный режим)

Этот режим также называют некоторыми брендами «расширенный адресный режим». В этом режиме 4-байтовый адрес отправляется; это действие позволяет обращаться к флэш-памяти размером до четырех Гбайт. Этот режим поддерживается многими производителями памяти QSPI, такими как Micron, Spansion и Macronix.

Перед использованием 4-байтового режима пользователь должен проверить, поддерживается ли оно используемым устройством.

Чтобы включить этот режим, пользователь должен:

• Настройте память, войдя в 4-байтовый режим. Обратитесь к спецификации соответствующего производителя для получения более подробной информации о том, как войти в этот режим.

- Например, для устройств Micron следует использовать команду ENTER 4-BYTE ADDRESS MODE «B7h», а затем пользователь должен использовать выделенные 4-байтовые адресные команды для некоторых операций, таких как чтение, программирование или удаление, из таблицы данных устройства.

- Для устройств Micron, если требуется операция чтения, пользователь должен использовать 4-байтную команду READ «13h» вместо READ «03»

• Сконфигурируйте периферийное устройство QUADSPI в режиме 32-битного адреса, установив поле ADSIZE [1: 0] = 11 в регистре QUADSPI_CCR.

Примечание:

Размер памяти должен быть настроен в соответствии с фиксированным размером адреса в соответствии со следующей формулой: количество байтов во флэш-памяти = 2 [FSIZE + 1], где [FSIZE + 1] фактически является количеством битов адреса, необходимых для обращения к флэш-памяти ,

В следующей таблице приведены различные режимы адреса в зависимости от максимального адресуемого объема памяти.

Table 12. Address mode versus maximum addressable memory space

Address length	QUADSPI_CCR ADSIZE [1:0]	QUADSPI_DCR [20:16] FSIZE [4:0]	Memory size $2^{[FSIZE+1]}$
8-Bits address	00	00111	Up to 256 bytes
16-Bits address	01	01111	Up to 64 Kbytes
24-Bits address	10	10111	Up to 16 Mbytes
32-Bits address	11	11111	Up to 4 Gbytes(1)

1. Только первые 256 Мбайт памяти QSPI могут быть прочитаны в режиме отображения памяти (от 0x9000 0000 до 0x9FFF FFFF)

Предостережения о 4-байтовом адресном режиме:

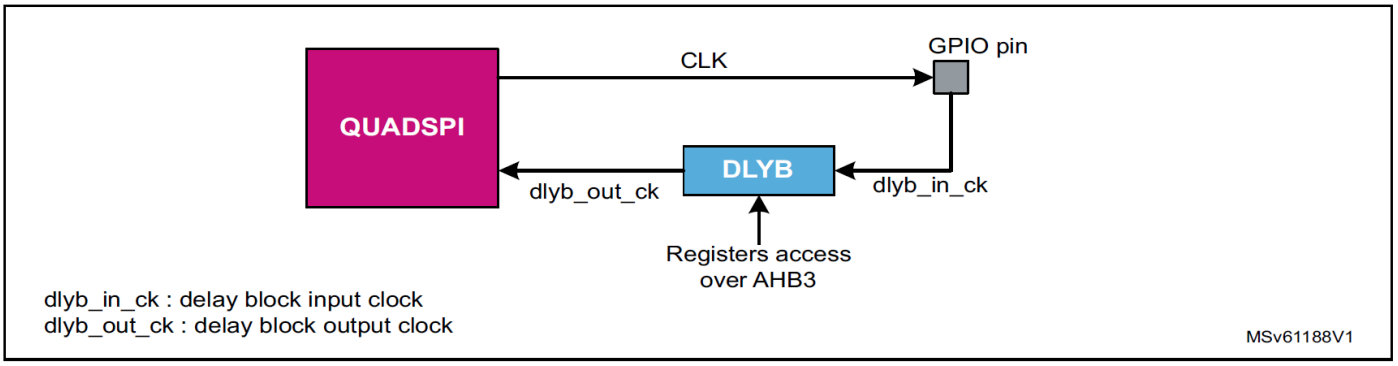
• В косвенном режиме, если адрес плюс длина данных превышает размер флэш-памяти, флаг TEF устанавливается при запуске доступа.

• В режиме отображения памяти, если осуществляется доступ к адресу вне диапазона, определенного FSIZE, но все еще в пределах диапазона 256 Мбайт, тогда выдается ошибка AHB. Эффект этой ошибки зависит от мастера AHB, который пытался получить доступ; если это процессор Cortex®, генерируется прерывание из-за серьезного сбоя, а если это DMA, генерируется ошибка передачи DMA, в то время как соответствующий канал DMA автоматически отключается.

2.4.7 QUADSPI и блок задержки в серии STM32H7

В серии STM32H7 QUADSPI имеет свой собственный блок задержки, доступный через AHB3. Блок задержки может использоваться для генерации тактовой частоты дискретизации, которая сдвинута по фазе от выходных тактовых импульсов, отправляемых на вывод микросхемы. Блок задержки выравнивает тактовую частоту выборки по входящим данным.

Figure 15. QUADSPI and delay block



2.5 Прерывания и использование прямого доступа к памяти

2.5.1 Использование прерываний Периферийное устройство QUADSPI поддерживает пять различных прерываний, каждое из которых полезно в конкретном случае. Для использования каждое прерывание должно быть включено путем установки соответствующего бита включения и включения глобального прерывания QUADSPI на стороне NVIC. В таблице ниже приведены все поддерживаемые прерывания.

Таблица 13. Сводка прерываний QUADSPI

Interrupt	Event Flag(1)	Enable control bit(2)	Clear bits(3)	Description
Timeout	TOF	TOIE	CTOF	Произошел тайм-аут
Status match	SMF	SMIE	CSMF	Сопоставление маскированных полученных данных с регистром сопоставления (только в режиме автоматического опроса)
FIFO threshold	FTF	FTIE	-	Порог FIFO достигнут (непрямой режим)

Transfer complete	TCF	TCIE	CTCF	Indirect mode: Косвенный режим: было передано правильное количество данных, заданное в регистре QUADSPI_DLR. Все режимы: передача была прервана
Transfer error	TEF	TEIE	CTEF	Indirect mode: Косвенный режим: был получен адрес вне диапазона

1. Все флаги событий доступны в регистре QUADSPI_SR.
2. Все биты включения-контроля доступны в регистре QUADSPI_CR.
3. Все биты сброса доступны в регистре QUADSPI_FCR.

2.5.2 Использование DMA

DMA может использоваться для передачи данных из или в внешнюю память QSPI, это возможно, когда интерфейс QUADSPI настроен либо в режиме косвенного чтения / записи, либо в режиме отображения памяти. В режиме отображения памяти разрешено только чтение из памяти QSPI.

Использование прямого доступа к памяти с QUADSPI в Косвенном режиме

В режиме DMA DMA является контроллером потока. Когда порог QUADSPI FIFO достигнут, когда установлен бит DMAEN, запросы DMA генерируются из QUADSPI в DMA.

Передача начинается, когда

- Бит DMAEN установлен, а QSPI и DMA настроены
- Флаг FTF устанавливается, когда порог FIFO достигнут

Если DMAEN = 1 уже, контроллер DMA должен быть отключен перед изменением FTHRES / FMODE.

В косвенном режиме при настройке прямого доступа к памяти для передачи данных из / в QUADSPI QUADSPI следует рассматривать как периферийное устройство:

- Режим памяти на периферию в случае записи данных в QUADSPI из внутренней памяти
- Периферийный режим памяти в случае чтения данных из QUADSPI для передачи во внутреннюю память.

Также адрес QUADSPI должен быть записан в регистр периферийных адресов (канал DMA / поток x периферийный адрес).

В таблице ниже приведены различные запросы DMA и направления передачи по сравнению с серией STM32.

Table 14. DMA requests mapping and transfer directions versus STM32 series

Product (1)	DMA1	DMA2	MDMA
STM32L4 Series	Request 5 Channel 5	Request 3 Channel 7	NA
STM32F4 Series	NA	Stream 7 Channel 3	NA
STM32F7 Series	NA	Stream 7 Channel 3	NA
STM32H7 Series	NA	NA	quadspi_ft_trg channel X[0..15]/Stream22
	NA	NA	quadspi_tc_trg channel X[0..15]/Stream23

Использование DMA с QUADSPI в режиме отображения памяти

В режиме отображения памяти QUADSPI обеспечивает доступ к внешней памяти для операции чтения через область адресов отображенной памяти (от 0x9000 0000 до 0x9FFF FFFF) и позволяет видеть внешнюю память как внутреннюю память.

В этом случае при настройке прямого доступа к памяти для передачи данных из области отображения памяти QUADSPI в другую внутреннюю память область отображения памяти QUADSPI должна рассматриваться как память при настройке регистров DMA:

Режим памяти в память в случае чтения данных из отображенной области памяти QUADSPI для передачи во внутреннюю память.

Также адрес QUADSPI должен быть записан в регистр периферийных адресов (канал DMA / поток x адрес памяти).

В режиме отображения памяти DMA является контроллером потока, поскольку QUADSPI не генерирует запросы DMA

В режиме отображения памяти DMA1 или DMA2 могут использоваться для передачи данных из внешней памяти QSPI в любую другую память или периферийное устройство. Чтобы выполнить передачу с использованием DMA из внешней памяти QSPI в любую другую память или периферийное устройство, пользователь должен настроить DMA, задав адрес источника (от 0x9000 0000), адрес назначения и количество данных, которые должны быть переданы.

Бит DMAEN не действует в режиме отображения памяти, передача начинается, как только DMA получает доступ к диапазону адресов QUADSPI (от 0x90000000 до 0x9FFFFFFF).

Как только сконфигурированная передача DMA запускается программным обеспечением, DMA считывает данные из памяти QSPI точно как внутреннюю память. Периферийное устройство QUADSPI управляет связью с внешней памятью и помещает считанные данные в FIFO.

Количество элементов данных, подлежащих передаче, управляется DMA, поэтому пользователь должен сконфигурировать количество данных в регистре DMA DMA_SxNDTR (или в регистре DMA_CNDTRx для STM32L4x6xx). Нет необходимости настраивать регистр QUADSPI_DLR, поскольку он не действует в режиме отображения памяти, где DMA является контроллером потока.

Примечание:

FIFO DMA может использоваться, например, если требуется режим DMA Burst, чтобы уменьшить издержки передачи на матрице шины.

QUADSPI и мастер DMA в серии STM32H7

В серии STM32H7 MDMA управляет доступом DMA к интерфейсу QUADSPI. MDMA может получить доступ к QUADSPI следующими способами:

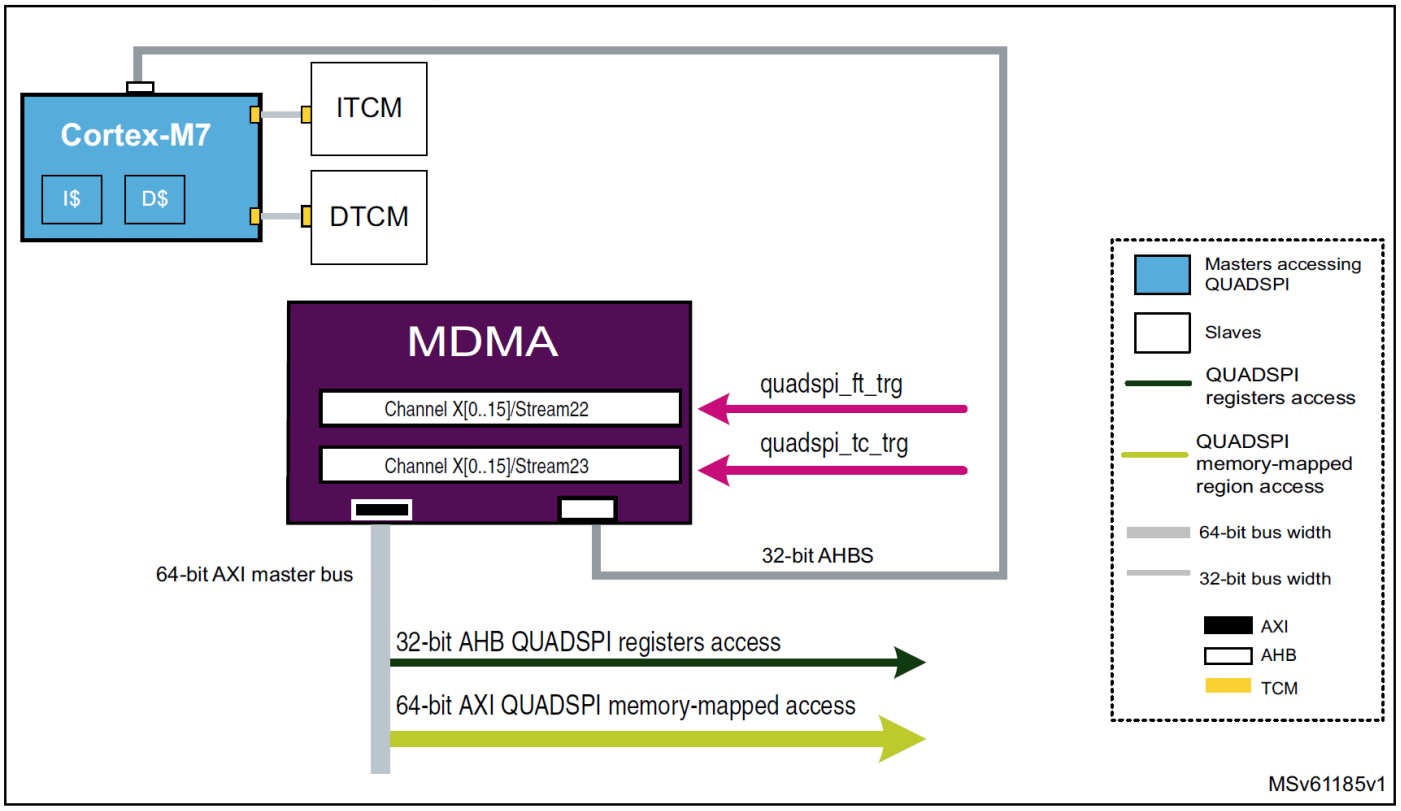
- Непосредственно из матрицы шины AXI через 64-разрядную шину AXI для доступа к отображенной памяти
- От AHB3 через 32-битную шину AHB для доступа к регистрам

Главный DMA предлагает два триггерных сигнала от интерфейса QUADSPI, чтобы обеспечить большую гибкость для пользовательского приложения. Два

триггерных сигнала:

- quadspi_ft_trg: пороговый триггер QUADSPI FIFO
- quadspi_tc_trg: триггер завершения передачи QUADSPI

Figure 16. QUADSPI and master DMA



2.6 Режимы пониженного энергопотребления

Состояние питания STM32 является важным требованием, которое необходимо учитывать, поскольку оно напрямую влияет на состояние интерфейса QUADSPI. Например, если MCU находится в режиме ожидания, тогда QUADSPI должен быть перенастроен после выхода из этого режима.

Примечание:

Память QSPI также может быть переведена в режим пониженного энергопотребления. В зависимости от марки памяти, некоторые устройства поддерживают как режим ожидания, так и режим глубокого отключения, тогда как другие устройства поддерживают только режим ожидания.

Чтобы сэкономить больше энергии, когда приложение находится в режиме низкого энергопотребления, рекомендуется перевести память QSPI в режим низкого энергопотребления, прежде чем вводить STM32 в режиме низкого энергопотребления. Более подробная информация о снижении энергопотребления доступна в разделе 6.2 на стр. 90.

Передача возможна в спящем режиме, когда процессор остановлен, благодаря интеллектуальной архитектуре STM32 и тому факту, что в спящем режиме все периферийные устройства могут быть включены. Это может подходить для носимых приложений, где низкое энергопотребление является обязательным.

Мастер AHB, такой как DMA, может продолжить передачу из QUADSPI (при использовании режима отображения памяти) даже после входа в MCU в спящем режиме. Как только передача завершена, может быть сгенерировано прерывание для активации STM32.

Обратитесь к справочному руководству для получения подробной информации о конфигурации режима с низким энергопотреблением

3 Конфигурация QUADSPI

В этом разделе описываются все этапы настройки QUADSPI, необходимые для выполнения операций чтения, записи или удаления.

3.1 Конфигурация GPIO

Пользователь должен настроить GPIO, которые будут использоваться для взаимодействия с памятью QSPI, и это зависит от предпочтительной конфигурации оборудования. Подробнее о конфигурации оборудования см. В Таблице 8 на стр. 23.

Примечание:

Рекомендуется выполнить сброс периферийного устройства QUADSPI перед началом настройки, а также гарантировать, что периферийное устройство находится в состоянии сброса.

В зависимости от доступности GPIO пользователь может настроить GPIO Bank1 или Bank2. Пользователь также может настроить оба банка, если подключены две памяти QSPI.

Примечание:

Все GPIO должны быть настроены в высокоскоростном режиме.

3.1.1 Настройка GPIO с помощью инструмента STM32CubeMX

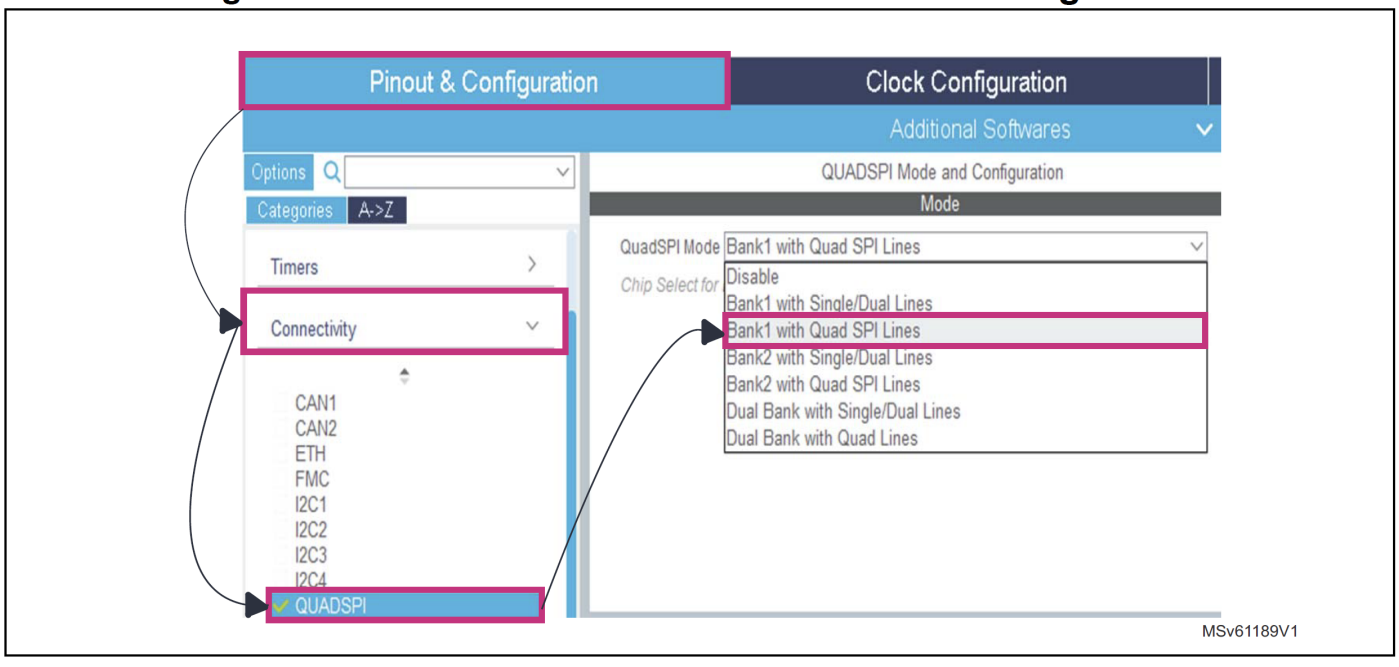
В следующем примере показано, как настроить GPO QUADSPI в режиме четырехканального ввода-вывода с использованием GPIO Bank1.

Использование инструмента STM32CubeMX - это очень простой, легкий и быстрый способ настройки периферийного устройства QUADSPI и его GPIO, поскольку он позволяет создавать проект с предварительно настроенным QUADSPI.

После создания проекта STM32CubeMX в окне «Режим» можно выбрать аппаратную конфигурацию. Это окно можно найти на вкладке «Распиновка» и «Конфигурация» в разделе «Связь» и при выборе меню QUADSPI.

На рисунке 17 показано, как выбрать аппаратную конфигурацию QUADSPI с помощью STM32CubeMX, где используются GPIO Bank1.

Figure 17. STM32CubeMX: QUADSPI GPIOs configuration

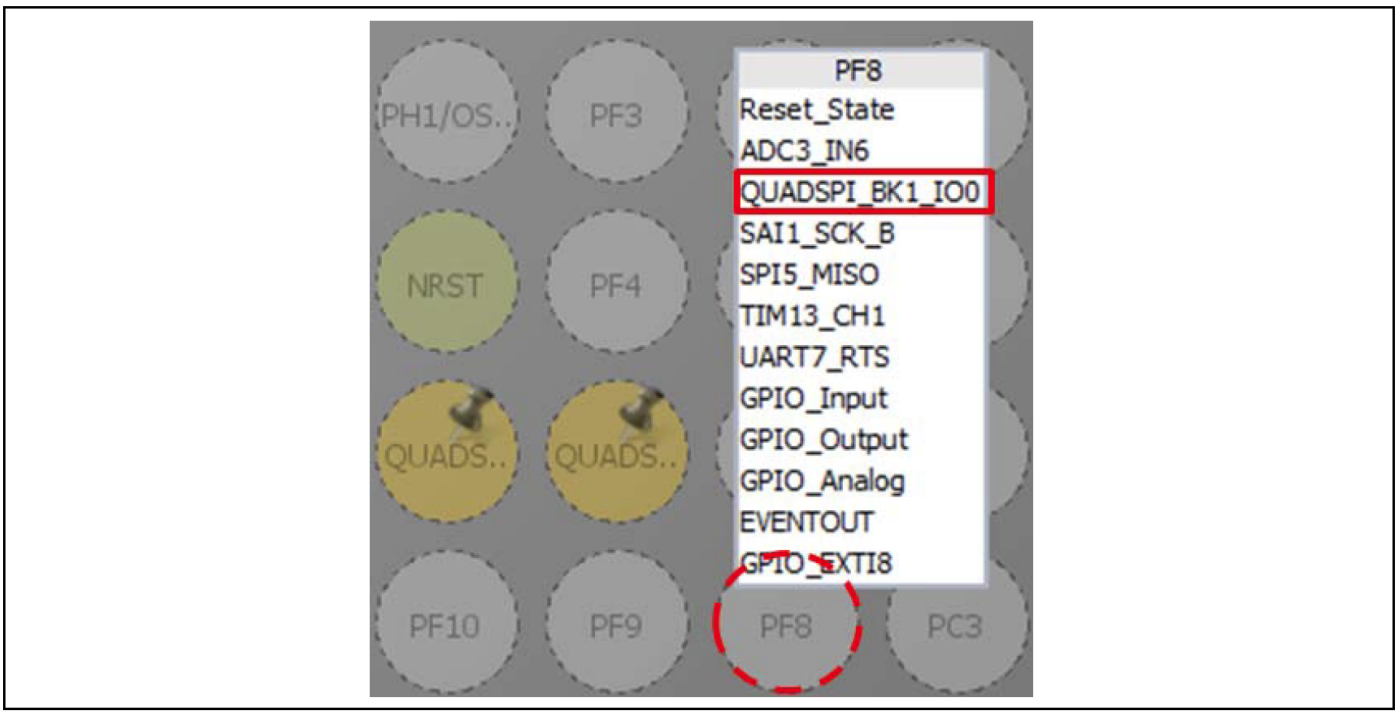


Если после выбора одной аппаратной конфигурации (как показано на рисунке 17) используемые GPIO не соответствуют плате подключения памяти, пользователь может настроить альтернативную функцию непосредственно на соответствующих выводах.

Для получения дополнительной информации о доступности альтернативных функций QUADSPI по сравнению с GPIO см. Таблицу сопоставления альтернативных функций в соответствующей таблице данных.

На рисунке ниже показано, как вручную настроить вывод PF8 для альтернативной функции QUADSPI_BK1_IO0

Figure 18. STM32CubeMX: PF8 pin configuration to QUADSPI_BK1_IO0 alternate function

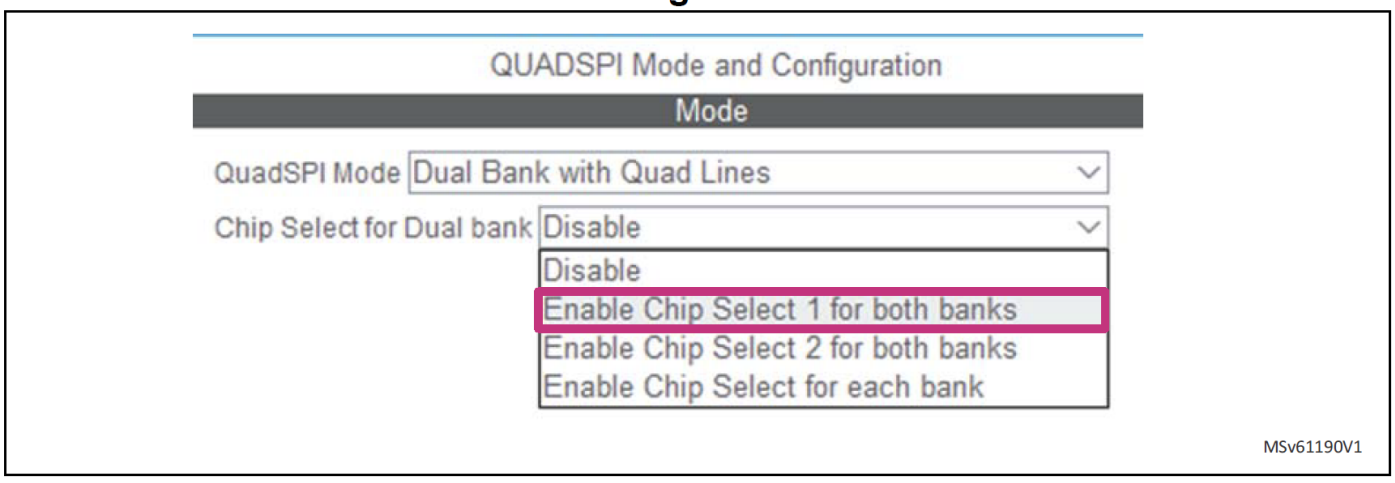


Используемые контакты подсвечиваются зеленым, когда GPIO интерфейса QUADSPI правильно настроены.

Случай использования двойной флеш-памятьи

Если выбран двойной банк, пользователь должен выбрать одну из пере-

Figure 19. STM32CubeMX: Dual-Flash memory QUADSPI with chip-select 1 configuration



численных конфигураций выбора микросхемы. Для получения дополнительной информации о различных конфигурациях выбора микросхемы памяти с двумя флеш-накопителями см. Раздел 2.2.4: Режим памяти с двумя флеш-накопителями.

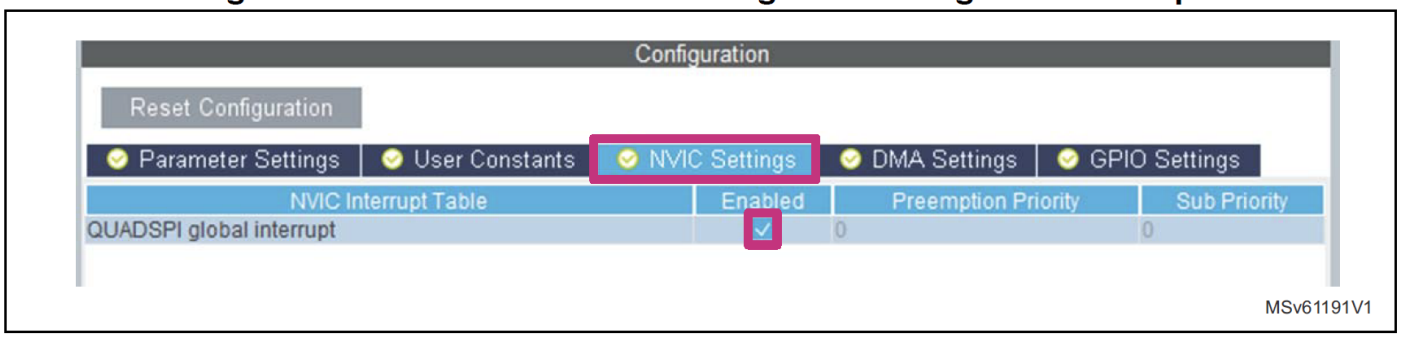
На рисунке выше показано, как настроить выбор микросхемы 1 для обоих банков с помощью STM32CubeMX.

Включение прерываний QUADSPI

Чтобы иметь возможность использовать прерывания QUADSPI, пользователь должен включить глобальное прерывание QUADSPI на стороне NVIC. После этого каждое прерывание включается отдельно путем включения его соответствующего бита разрешения (биты разрешения прерываний доступны в регистре QUADSPI_CR, описанном в Табл. 13: Сводка прерываний QUADSPI).

На рисунке ниже показано окно конфигурации, в котором можно включить глобальное прерывание QUADSPI на вкладке NVIC.

Figure 20. STM32CubeMX: enabling QUADSPI global interrupt



3.2 QUADSPI периферийная конфигурация и тактовая частота

3.2.1 Конфигурация периферии QUADSPI (регистр QUADSPI_CR)

Периферийное устройство QUADSPI настраивается с использованием QUADSPI_CR. Пользователь должен сконфигурировать коэффициент деления часового прескалера и параметры сдвига выборки для входящих данных.

Запросы DMA разрешены, устанавливая бит DMAEN (бит DMAEN недоступен для продуктов серии STM32H7). В случае использования прерывания, их соответствующий бит разрешения также может быть установлен во время этой фазы. Уровень FIFO как для генерации запроса DMA, так и для генерации прерывания программируется в битах FTHRES.

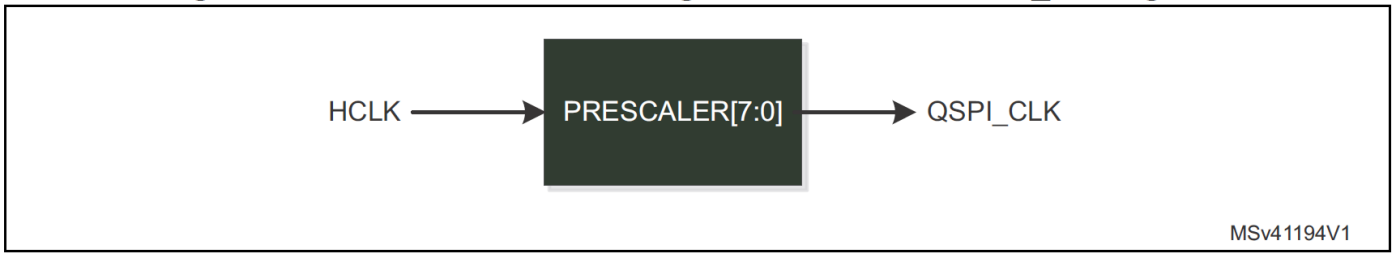
Если требуется счетчик времени ожидания, бит TCEN можно установить и значение времени ожидания можно запрограммировать в регистре QUADSPI_LPTR.

Режим двойной флэш-памяти можно активировать, установив DFM в 1.

Источником тактирования QUADSPI является AHB, в котором используется прескалер для генерации CLAD QUADSPI. В зависимости от запрограммированного прескалера в регистре QUADSPI_CR, возможно, что и CPU, и QUADSPI работают с одинаковой скоростью (PRESCALER [7: 0] = 0).

На следующем рисунке показана схема синхронизации QUADSPI для серий STM32F4, STM32L4, STM32F7 и STM32WB, где QSPI_CLK = HCLK / (Prescaler + 1).

Figure 21. QUADSPI clock configuration on QUADSPI_CR register



В устройствах серии STM32H7 QUADSPI содержит два разных источника синхронизации:

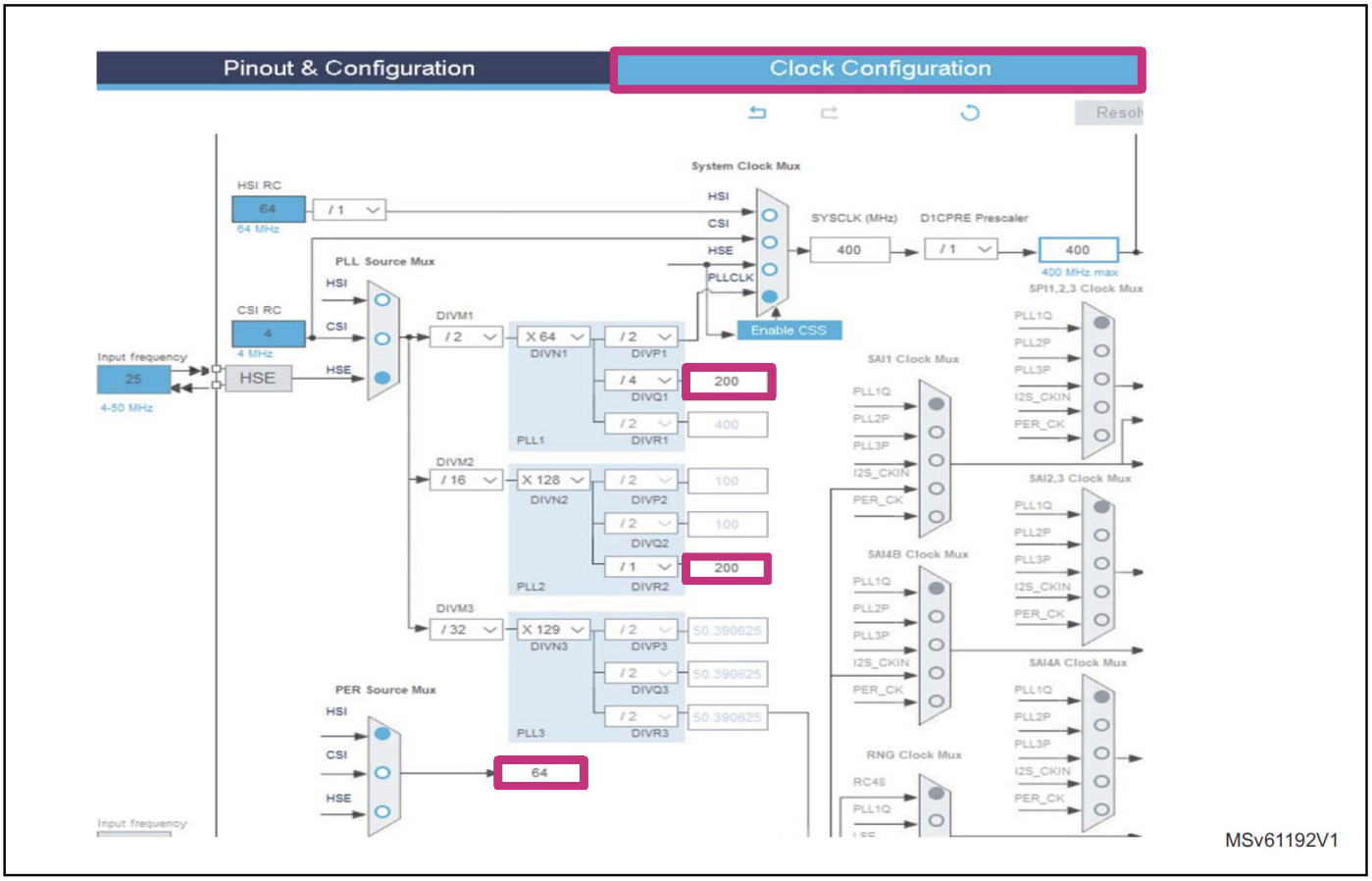
• quadspi_ker_ckIt - это исходные часы для генерации QUADSPI CLK, используя следующее соотношение (QSPI_CLK = quadspi_ker_ck / (Prescaler + 1).

• quadspi_hclk (hclk3) Это исходные часы для интерфейса регистра. Эти часы не влияют на QUADSPI CLK.

STM32CubeMx разрешает настройку часов источника quadspi_ker_ck в разделе конфигурации часов.

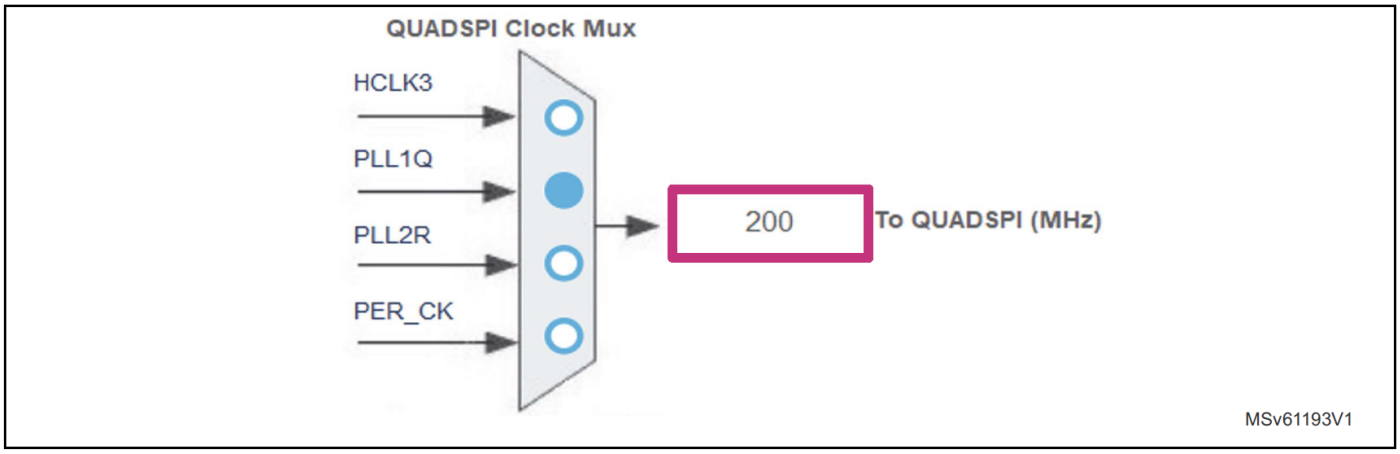
На следующем рисунке показаны часы с несколькими источниками для quadspi_ker_ck с использованием STM32CubeMx.

Figure 22. STM32CubeMX: quadspi_ker_ck source clock configuration in STM32H7 Series



Исходные часы для quadspi_ker_ck можно выбрать с помощью мультиплексора часов QUADSPI, как показано на следующем рисунке.

Figure 23. STM32CubeMX: quadspi_ker_ck source clock selection in STM32H7 Series



3.2.2 Настройка параметров флэш-памяти QSPI (регистр QUADSPI_DCR)

Параметры, относящиеся к целевой внешней флэш-памяти, настраиваются через регистр QUADSPI_DCR. Пользователь должен запрограммировать объем флэш-памяти в поле FSIZE, минимальное время ожидания выбора микросхемы в поле CSHT и функциональный режим (режим 0 или режим 3) в бите MODE в регистре QUADSPI_DCR.

Примечание:

Параметры QUADSPI могут быть изменены, когда приложение работает, но не во время текущей передачи, другими словами, не когда бит занят установлен. Если во время текущей передачи требуется изменение параметров, бит прерывания можно использовать для остановки текущей операции, затем можно изменить конфигурацию.

Конфигурация QUADSPI с использованием STM32CubeMX

Инструмент STM32CubeMX можно использовать для настройки периферийного устройства QUADSPI. Как только GPIO будут правильно настроены, как уже описано, параметры QUADSPI могут быть настроены. Однако все конфигурации, связанные с началом обмена данными, должны быть добавлены пользователем вручную в проекте.

В окне конфигурации QUADSPI выберите вкладку «Параметры», затем настройте их. На рисунке ниже показан пример конфигурации QUADSPI в соответствии со следующими условиями:

- Clock prescaler = 2 => QSPI_CLK = FAHB/3 for STM32H7 Series QSPI_CLK=quadspi_ker_ck/3 (Прескейлер тактовых импульсов = 2 => QSPI_CLK = FAHB / 3 для серии STM32H7 QSPI_CLK = quadspi_ker_ck / 3)
- FIFO threshold = 4 => FTF flag... (Порог FIFO = 4 => Флаг FTF устанавливается, как только пять или более свободных байтов доступны для записи в FIFO в случае операции записи, или устанавливается FTF, если есть пять или более действительных байтов, из которых можно прочитать ФИФО)
- Sample shifting half cycle enabled =>sample the dat... (Включен полупериод сдвига выборки => отсчитать данные, считанные из полупериода памяти, позже (отрегулируйте время выборки в случае накопленных задержек на печатной плате))

• Flash size is 16 Mbytes => number of bytes in Flash memory = 2[FSIZE+1] = 2[23+1] => FSIZE = 23 (Размер флэш-памяти составляет 16 Мбайт => количество байтов во флэш-памяти = 2 [FSIZE + 1] = 2 [23 + 1] => FSIZE = 23)

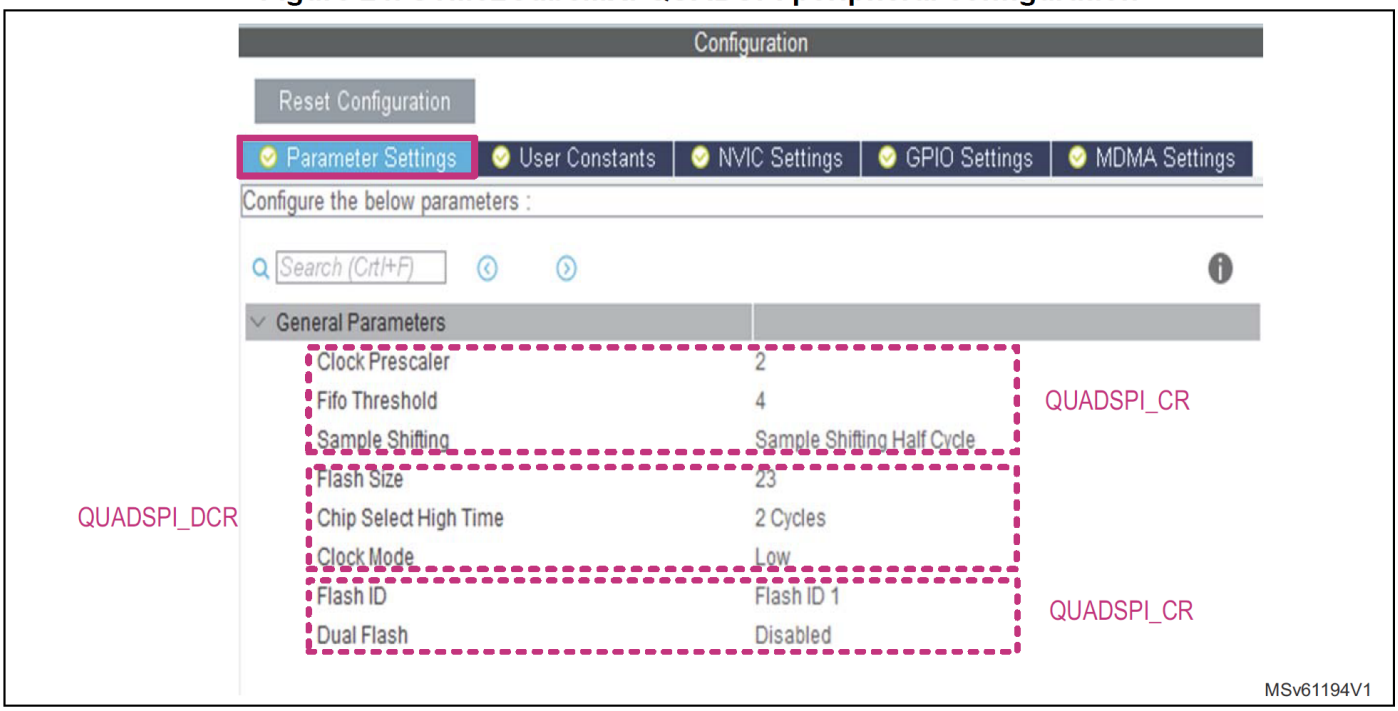
• Chip select high time = 2 cycles => CSHT[2:0] = 1 (Указать наибольшее время выбора ЧИПа = 2 цикла => CSHT [2: 0] = 1)

• Clock mode is low => clock mode 0 enabled (Режим часов низкий => режим часов 0 включен)

• Flash ID = Flash ID 1 => выберите флэш-память 1 для адресации в режиме одиночной Flash FSEL = 0 Обратите внимание, что FSEL игнорируется, когда DFM = 1

• Dual Flash = disabled ==> Dual-Flash mode disabled DFM = 0 (Dual Flash = отключено ==> Режим Dual Flash отключен DFM = 0)

Figure 24. STM32CubeMX: QUADSPI peripheral configuration



3.2.3 Конфигурация QUADSPI и MPU

Блок защиты памяти (MPU) можно использовать для повышения надежности и безопасности пользовательского приложения за счет защиты области памяти.

При использовании продукта STM32 на базе Cortex-M7 (например, STM32F7 и STM32H7Series) настоятельно рекомендуется сконфигурировать область памяти QUADSPI, доступную в режиме отображения памяти, как строго упорядоченную память.

Область памяти QUADSPI, доступная в режиме отображения памяти, изменяется от 0x90000000 до 0x9FFFFFFF. При выполнении этого действия центральному процессору запрещен доступ к этой области, пока режим отображения памяти не включен.

В той же области (от 0x90000000 до 0x9FFFFFFF) после определения размера внешней флэш-памяти и включения режима отображения памяти рекомендуется настроить неиспользуемую оставшуюся область памяти также как строго упорядоченную память.

Для получения дополнительной информации о настройке модуля защиты памяти см. Примечание по применению Управление модулем защиты памяти (MPU) в микроконтроллерах STM32 (AN4838).

3.2.4 Конфигурация устройства памяти QSPI

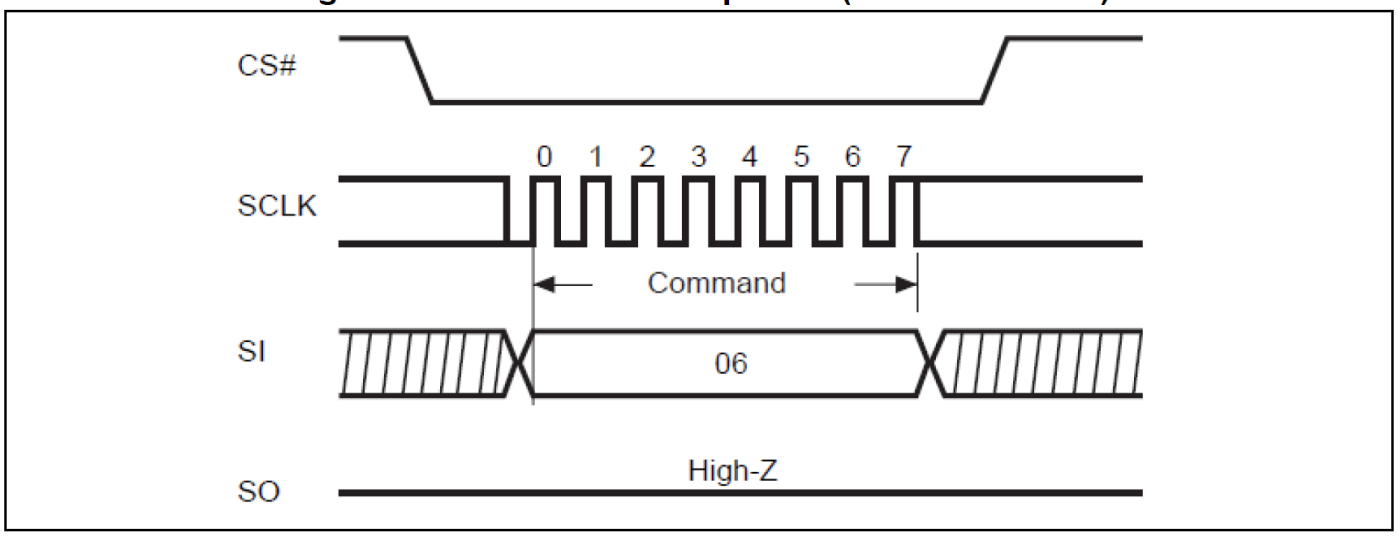
Память QSPI должна быть настроена после того, как было настроено периферийное устройство QUADSPI. Конфигурация памяти QSPI зависит от выбранной конфигурации для QUADSPI для косвенного режима.

Enable writing to QSPI Flash memory (ключить запись во флэш-память QSPI)

Команда write-enable должна быть отправлена в память, чтобы установить бит фиксации разрешения записи (WEL).

Бит WEL должен быть установлен перед каждой операцией регистра состояния программирования, стирания и записи в память. Эта команда обычно отправляется в одну строку без указания адреса или данных. Используемый режим QUADSPI - 1-0-0 (инструкция в 1 строке - нет адреса - нет данных)

Figure 25. Write enable sequence (command 0x06)



Configuring dummy cycle (настройка фиктивного цикла)

Количество фиктивных циклов должно быть сконфигурировано в памяти QSPI в соответствии с тактовой частотой работы, для этого пользователь должен обратиться к таблице данных устройства памяти.

3.2.5 Начало связи (регистр QUADSPI_CCR)

После настройки периферийного устройства QUADSPI связь может быть запущена в одном из следующих режимов:

- Режим косвенной записи или косвенного чтения
- Режим опроса статуса-флага
- Режим отображения памяти.

Режим работы настраивается в поле FMODE [1: 0] в регистре QUADSPI_CCR.

Перед началом связи пользователь должен настроить формат кадра в регистре QUADSPI_CCR. Обратитесь к Разделу 2.1: Гибкий формат кадра для получения дополнительной информации о конфигурации формата кадра.

Indirect-write mode (FMODE = 00) (режим косвенной записи)

Связь начинается немедленно, если:

- Запись выполняется в INSTRUCTION [7: 0] (QUADSPI_CCR), если адрес не требуется (ADMODE = 00) и данные не требуются для встроенного программного обеспечения (DMODE = 00)

• Запись выполняется в ADDRESS [31: 0] (QUADSPI_AR), если необходим адрес (ADMODE! = 00) и если прошивка не требует данных (DMODE = 00)

• Запись выполняется в DATA [31: 0] (QUADSPI_DR), если необходим адрес (когда ADMODE! = 00) и если данные должны быть предоставлены встроенным программным обеспечением (DMODE! = 00)

Indirect-read mode (FMODE = 01) (режим косвенного чтения)

Связь начинается немедленно, если:

• Запись выполняется в INSTRUCTION [7: 0] (QUADSPI_CCR), и если адрес не требуется (ADMODE = 00)

• Запись выполняется в ADDRESS [31: 0] (QUADSPI_AR) и, если требуется адрес (ADMODE! = 00)

Status-flag polling mode (FMODE = 10) (режим опроса флага состояния)

Доступ к флэш-памяти начинается так же, как и в режиме косвенного чтения, связь начинается немедленно, если:

• Запись выполняется в INSTRUCTION [7: 0] (QUADSPI_CCR), и если адрес не требуется (ADMODE = 00)

• Запись выполняется в ADDRESS [31: 0] (QUADSPI_AR) и, если требуется адрес (ADMODE! = 00)

Memory-mapped mode (FMODE = 11) (режим отображения в память)

Как только режим отображения в память настроен, связь начинается, как только появляется запрос на доступ от любого мастера AHB

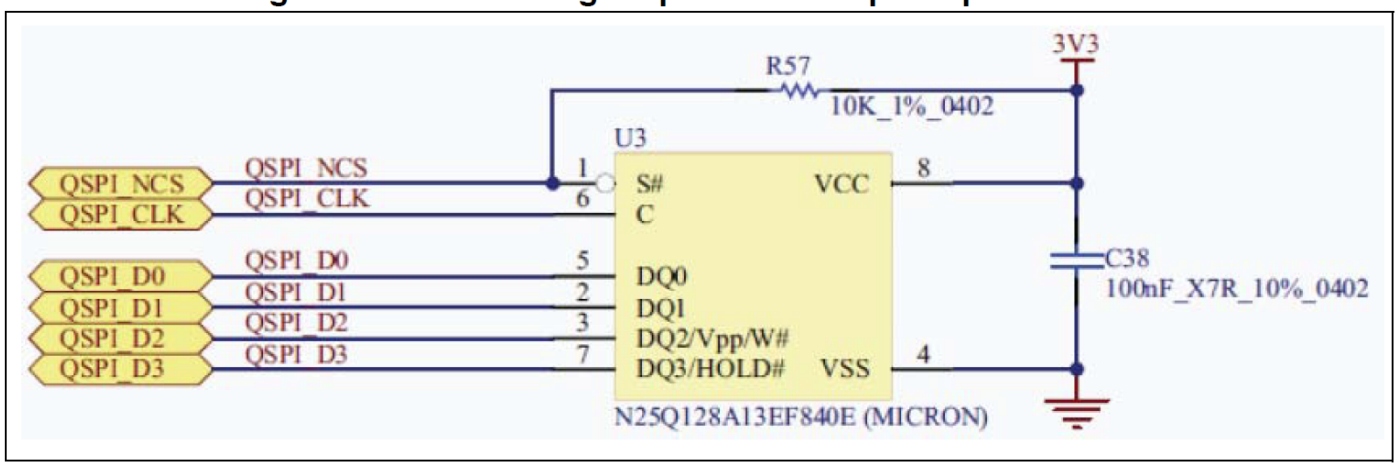
3.3 Особенности оборудования

3.3.1 Pull-up resistance (сопротивление подтягивания)

Большинство производителей памяти QSPI рекомендуют подключать сопротивление подтягивания к VCC на выводе CS. Это необходимо для того, чтобы при включении питания вывод выбора микросхемы отслеживал свое напряжение от VCC. Для получения более подробной информации об электрических рекомендациях обратитесь к техническому описанию соответствующих частей.

На следующем рисунке показан пример подключения памяти QSPI, где сопротивление подтягивания подключено к выводу выбора микросхемы.

Figure 26. Connecting chip-select to a pull-up resistance



3.3.2 Хорошая конструкция печатной платы позволяет максимальную скорость QUADSPI

Скорость, с которой работает интерфейс QUADSPI, зависит от многих факторов, включая расположение платы и скорости пэдов. При удачной компоновке должна быть возможность достичь максимальных скоростей, описанных в Таблице 2. Доступность Quad-SPI и функции для семейств STM32 и зафиксированные в техническом описании продукта.

Макет должен быть как можно лучше, чтобы получить наилучшие характеристики. Чтобы ознакомиться с рекомендациями по маршрутизации на печатной плате, см. Примечание по применению. Начало работы с разделом «Разработка аппаратного обеспечения MCU серии STM32F7 (AN4661)» «Квадратурный последовательный параллельный интерфейс (Quad SPI)», доступный на веб-сайте ST.

3.3.3 Chip-select high time (CSHT) (Наибольшее время выбора ЧИПа)

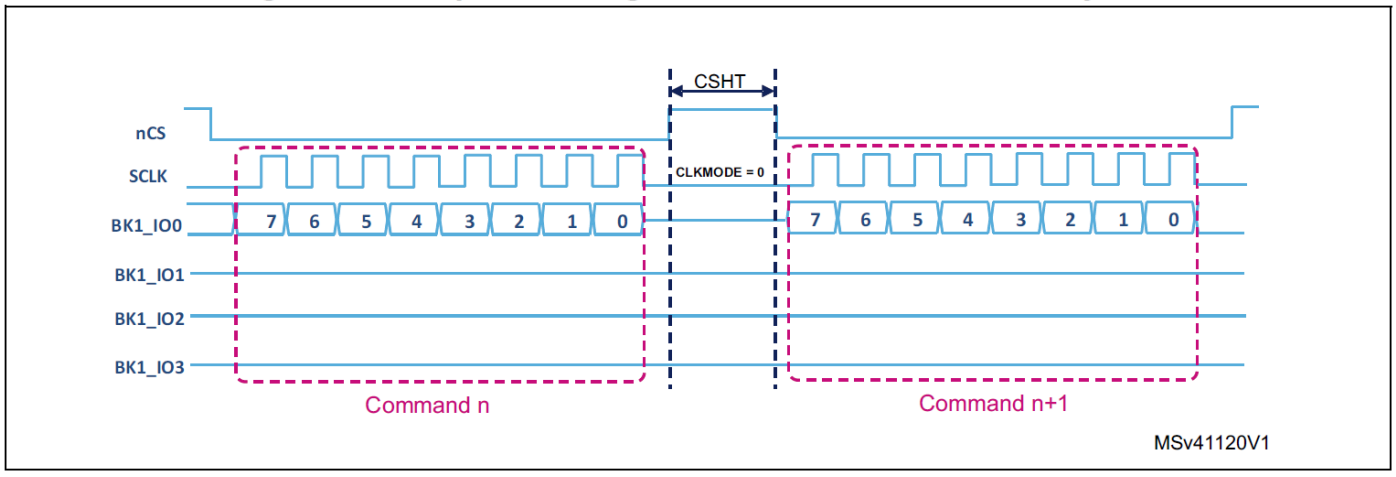
Когда QUADSPI отправляет две команды, одну сразу за другой, он поднимает высокий уровень сигнала выбора микросхемы (nCS) между двумя командами только для одного цикла CLK по умолчанию.

Если во флэш-памяти требуется больше времени между командами, поле CSHT [2: 0] выбора микросхемы в регистре QUADSPI_DCR можно использовать для указания минимального количества циклов QSPI_CLK (до восьми), которое nCS должно оставаться высоким.

3.3.4 CKMODE

Режим синхронизации указывает уровень, который CLK принимает между командами, когда nCS высокий. При высоком nCS поддерживаются два режима: режим 0, где CLK остается низким, и режим 3, где CLK остается высоким.

Figure 27. Chip select high time: CSHT = two clock cycles



3.3.5 Некоторые соображения при использовании QUADSPI в классическом режиме SPI

При использовании интерфейса QUADSPI в классическом режиме SPI пользователь должен учитывать следующие эквивалентности:

- Режим 0 для QUADSPI эквивалентен CPOL = 0 и CPHA = 0 для классического SPI
- Режим 3 для QUADSPI эквивалентен CPOL = 1, а CPHA = 1 для классического SPI

Основное различие между этими двумя режимами заключается в полярности часов, когда мастер шины находится в режиме ожидания и не передает никаких данных.

- SCLK остается в логическом низком состоянии с CPOL = 0, CPHA = 0
- SCLK остается в состоянии высокого логического уровня с CPOL = 1, CPHA = 1

Примечание:

Полный дуплекс не поддерживается при использовании интерфейса QUADSPI в классическом режиме SPI, поддерживается только полудуплекс.

На следующем рисунке показан классический пример кадра SPI, подчеркивающий эквивалентность режимов синхронизации Quad-SPI с классическим SPI.

3.3.5 Некоторые соображения при использовании QUADSPI в классическом режиме SPI

При использовании интерфейса QUADSPI в классическом режиме SPI пользователь должен учитывать следующие эквивалентности:

- Mode 0 (режим 0) для QUADSPI эквивалентен CPOL = 0 и CPHA = 0 для классического SPI
- Mode 3 (режим 3) для QUADSPI эквивалентен CPOL = 1, а CPHA = 1 для классического SPI

Основное различие между этими двумя режимами заключается в полярности часов, когда мастер шины находится в режиме ожидания и не передает никаких данных.

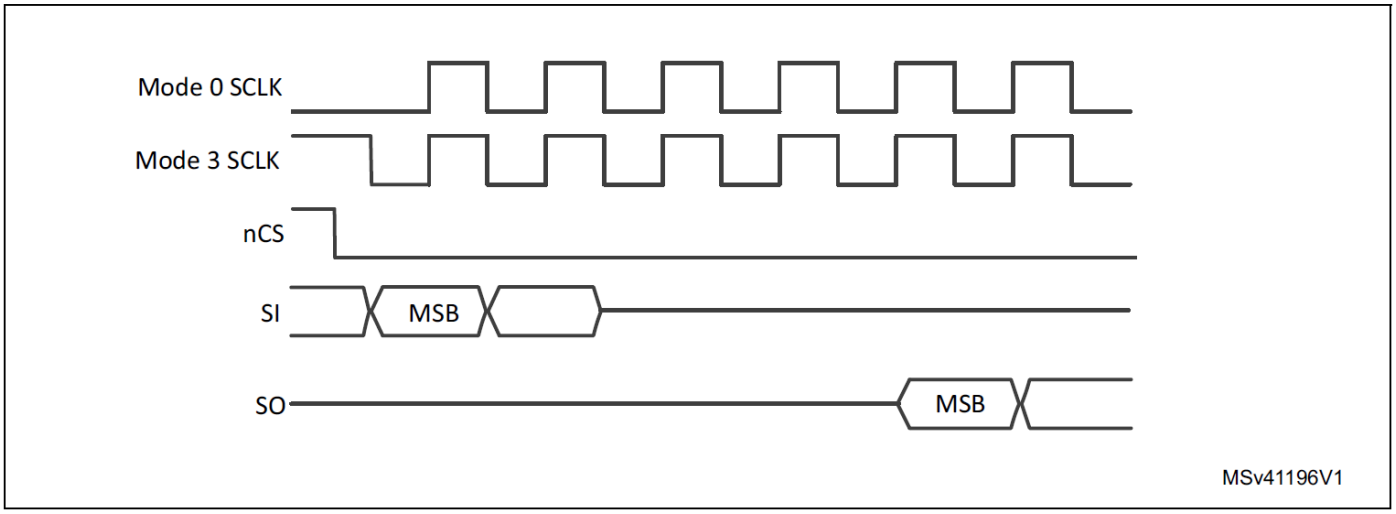
- SCLK остается в логическом низком состоянии с CPOL = 0, CPHA = 0
- SCLK остается в состоянии высокого логического уровня с CPOL = 1, CPHA = 1

Примечание:

Полный дуплекс не поддерживается при использовании интерфейса QUADSPI в классическом режиме SPI, поддерживается только полудуплекс.

На следующем рисунке показан классический пример кадра SPI, подчеркивающий эквивалентность режимов синхронизации Quad-SPI с классическим SP

Figure 28. QUADSPI in classical SPI mode frame example



4 Программирование QSPI Flash памяти

В этом разделе описывается программирование флэш-памяти QSPI в следующих случаях использования:

• Для конечной разработки приложения: в этом случае память QSPI программируется во время разработки продукта со статическими данными или кодом, которые будут использоваться в конечном продукте. Для размещения данных или кода, который будет использоваться в приложении, необходим специальный загрузчик флэш-памяти. Флэш-загрузчики, предоставляемые ST, могут использоваться для программирования, если пользователь использует одну из плат ST EVAL или Discovery, в противном случае пользователь должен разработать свой собственный загрузчик флэш-памяти.

• «На лету», когда приложение работает: в этом случае флэш-память QSPI используется в конечном продукте в качестве внешнего устройства хранения данных, что позволяет приложению хранить данные в любое время, когда это необходимо.

Примечание:

В обоих случаях принцип программирования одинаков. Единственное отличие состоит в том, что в первом случае операция программирования выполняется с помощью инструмента и загрузчика флэш-памяти во время разработки приложения, тогда как во втором случае операция программирования выполняется во время работы приложения в конечном продукте. Для программирования должен использоваться только косвенный режим, независимо от того, является ли это записью или стиранием.

В зависимости от используемой марки флэш-памяти доступны различные команды программирования, поэтому пользователь сам может настроить желаемую команду, поддерживаемую устройством.

Этапы команды, адреса и данных могут быть отправлены в одну, две или четыре строки для фазы команды в зависимости от марки устройства.

4-байтовый адресный режим можно использовать для программирования вспышек QSPI с размерами до 4 Гбайт.

Режим автоматического опроса может использоваться для ожидания, пока идет операция программирования; когда операция завершена, может быть сгенерировано прерывание.

4.1 Программный код или данные для конечного приложения

В этом разделе описывается, как программировать статический код или данные, которые будут использоваться в конечном приложении.

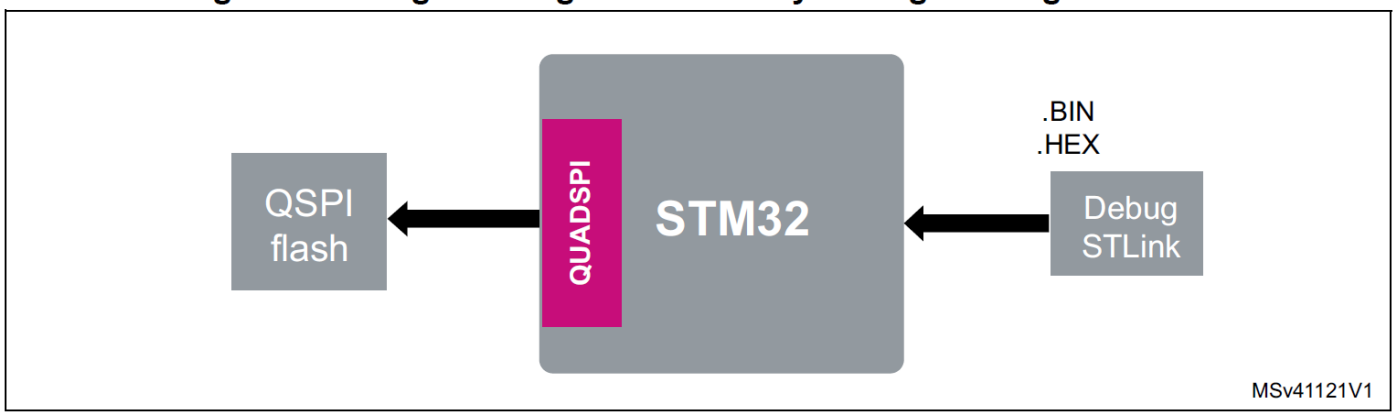
• Программный код для конечного приложения: код для приложения помещается в память QSPI для выполнения процессором, а затем для расширения во встроенную память на кристалле. Пример хранения кода во флэш-памяти QSPI описан в разделе 5.2: Выполнение из внешней памяти QSPI: расширение объема внутренней памяти.

• Данные программирования для конечного приложения: это полезно в графических приложениях, например, для хранения графического содержимого, такого как значки или изображения. Пример хранения данных во флэш-памяти

QSPI для конечного приложения описан в разделе 5.1: Чтение данных из памяти QSPI: графическое приложение.

Для программирования QSPI Flash для конечного приложения можно использовать утилиту STM32 ST-LINK или интегрированную среду разработки. Эта операция выполняется с использованием интерфейса отладки (SW, JTAG) через STM32.

Figure 29. Programming QSPI memory through debug interface



Внешний Flash-загрузчик: Выделенный алгоритм используется для выполнения операции программирования. Алгоритм загружается в STM32 через интерфейс отладки, затем выполняется для выполнения операции программирования. Входные данные для алгоритма - это двоичный файл, который нужно запрограммировать.

Предоставляются только флэш-загрузчики QSPI для модулей памяти, установленных на платах оценки и обнаружения STM32. Для другого оборудования пользователь должен разработать свой собственный загрузчик.

4.1.1 Программирование флэш-памяти QSPI с помощью утилиты STM32 ST-LINK

Если используемая среда IDE не поддерживает возможности программирования памяти QSPI, такие как System Workbench для STM32, пользователь может просто использовать утилиту STM32 ST-LINK.

Как создать новый загрузчик флэш-памяти QSPI и добавить его в утилиту ST-LINK

Для каждой конфигурации оборудования и для каждой марки флэш-памяти QSPI должен быть разработан специальный загрузчик флэш-памяти. Пользователь должен разработать собственный выделенный загрузчик флэш-памяти (файл .stldr), если используемое оборудование отличается от плат ST.

Проект представлен в каталоге установки утилиты ST-LINK «STMicroelectronics \ STM32 Утилита ST-LINK \ Утилита ST-LINK \ ExternalLoader \ N25Q256A_ STM32L476G-EVAL_Cube», позволяя пользователю разработать внешний загрузчик для флэш-памяти N25Q256A на STM32L476G-EVAL доска. Этот проект может быть легко адаптирован к пользовательскому оборудованию для генерации внешнего загрузчика.

Более подробную информацию о том, как разработать внешний загрузчик флэш-памяти QSPI для утилиты STM32 ST-LINK, см. В руководстве пользователя STM32 Описание программного обеспечения утилиты ST-LINK (UM0892), раздел

«Разработка пользовательских загрузчиков для внешней памяти», доступный на сайте www.st.st. .com.

Внимание:

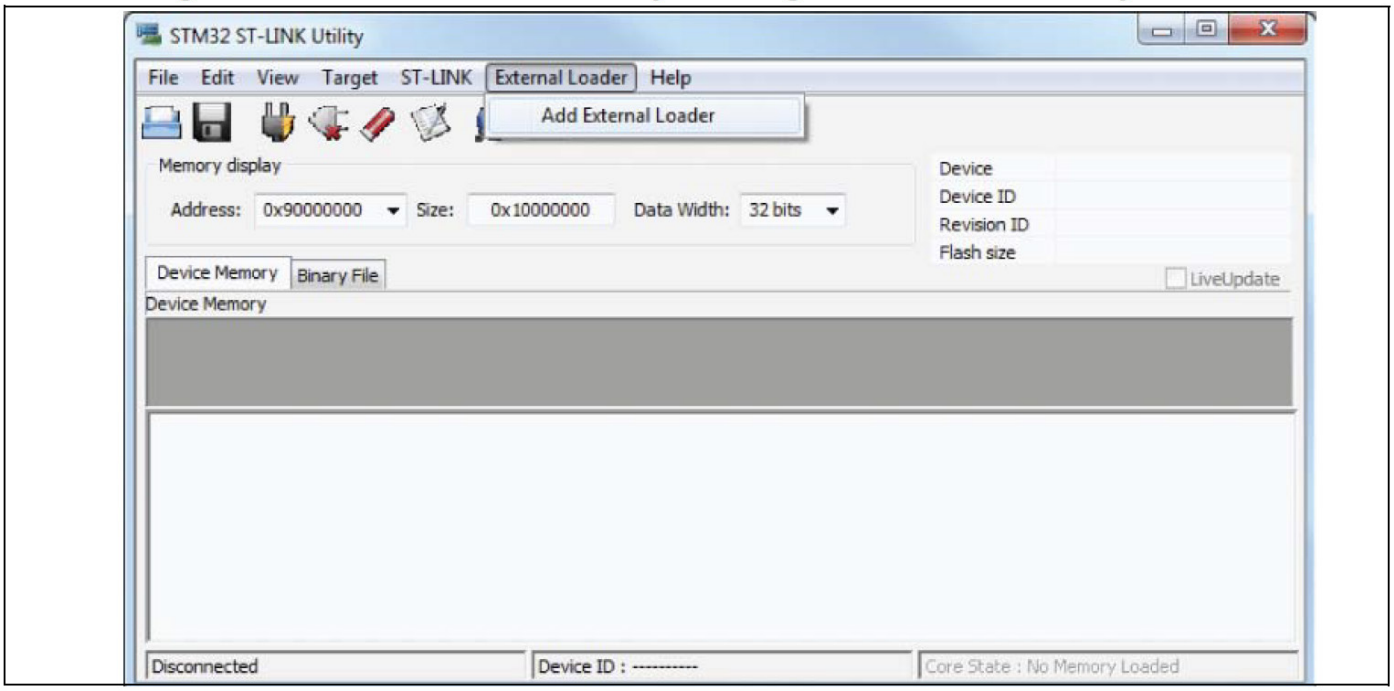
Цепочка инструментов / компилятор, используемый для генерации файла HEX / BIN для программирования памяти QSPI, должна быть точно такой же, как та, которая использовалась для разработки приложения.

Выделенный загрузчик флэш-памяти должен быть добавлен в утилиту ST-LINK, чтобы иметь возможность программировать флэш-память QSPI.

Как добавить загрузчик флэш-памяти QSPI в утилиту ST-LINK

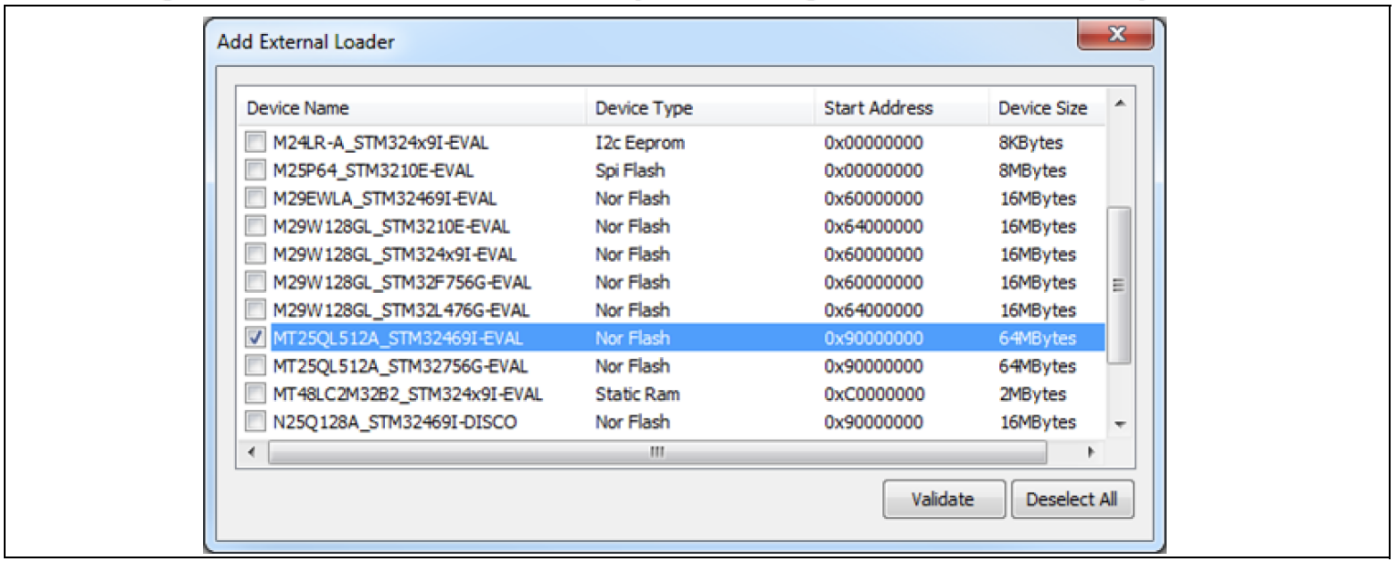
Чтобы добавить внешний загрузчик флэш-памяти, перейдите во внешний загрузчик и нажмите кнопку «Добавить внешний загрузчик».

Figure 30. STM32 ST-LINK utility: adding QSPI Flash memory loader



Появится окно, где пользователь должен выбрать свое устройство. Смотрите пример ниже:

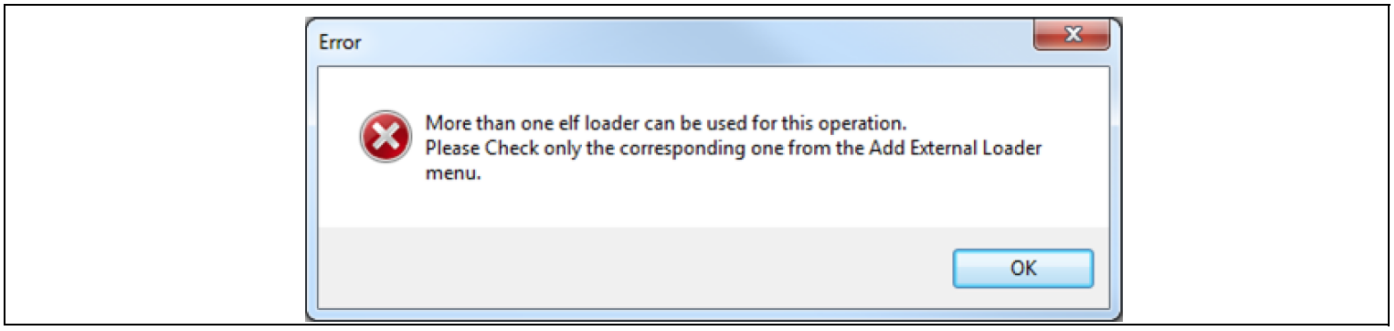
Figure 31. STM32 ST-LINK utility: selecting QSPI Flash memory loader



Примечание:

Можно добавить только один внешний загрузчик, в противном случае появится сообщение об ошибке на рисунке ниже. Пользователь может удалить один внешний загрузчик и при необходимости заменить его другим.

Figure 32. STM32 ST-LINK utility: error message



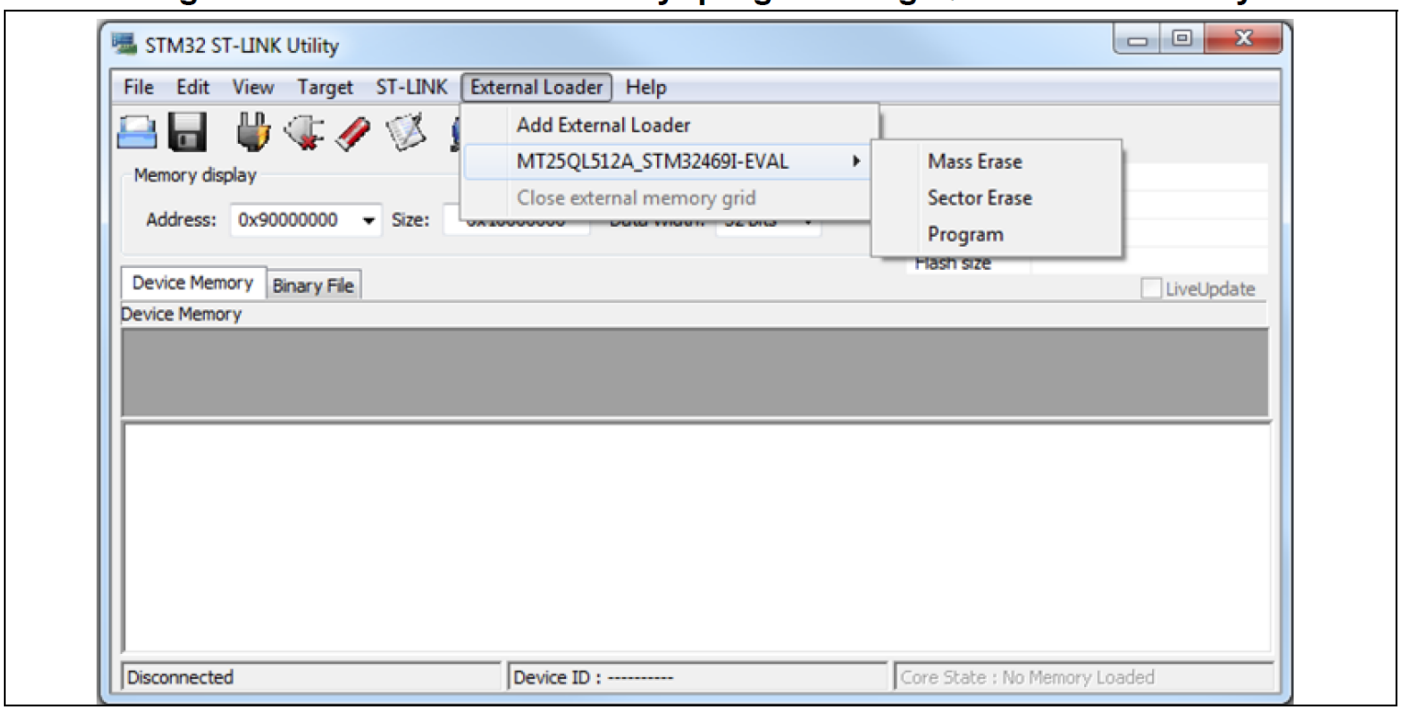
Как программировать флэш-память QSPI с помощью утилиты ST-LINK

Для программирования флэш-памяти QSPI выполните следующие действия:

1. Подключите плату с помощью USB-кабеля через отладчик ST-LINK.

2. В External Loader перейдите к добавленному внешнему загрузчику Flash памяти, затем выберите Program, как показано ниже

Figure 33. STM32 ST-LINK utility: programming QSPI Flash memory



3. Появится следующее окно, позволяющее пользователю перейти к файлу данных, который будет сохранен во флэш-памяти, который может быть двоичным файлом, файлом HEX или файлами S-записи Motorola (.srec или .s19).

Стирание памяти QSPI

Для выполнения операции массового стирания выберите «Массовое стирание», для стирания секторов нажмите «Стирание секторов», затем выберите сектора, которые нужно удалить, как показано на рисунке ниже.

Figure 34. STM32 ST-LINK utility: selecting HEX file for programming

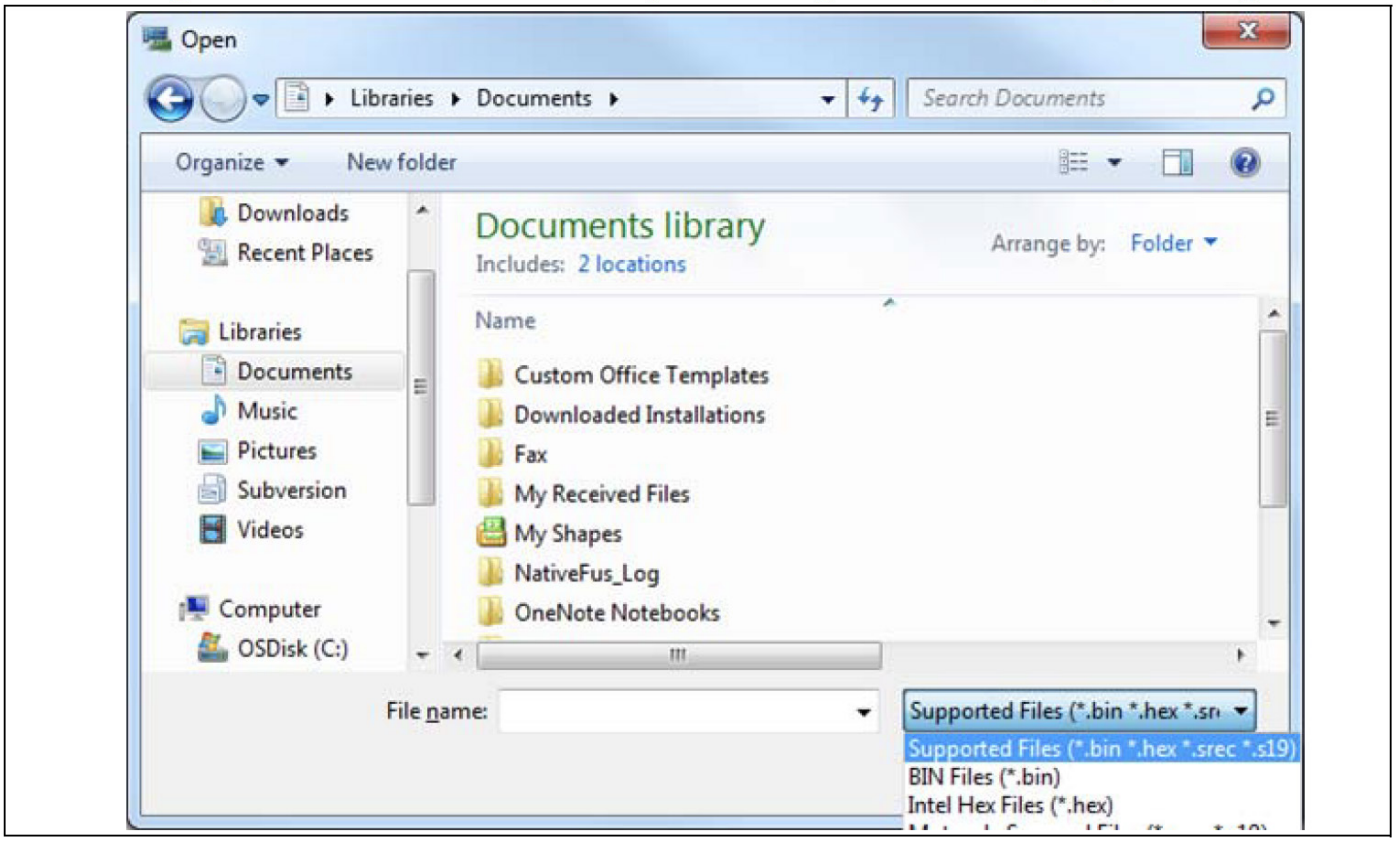
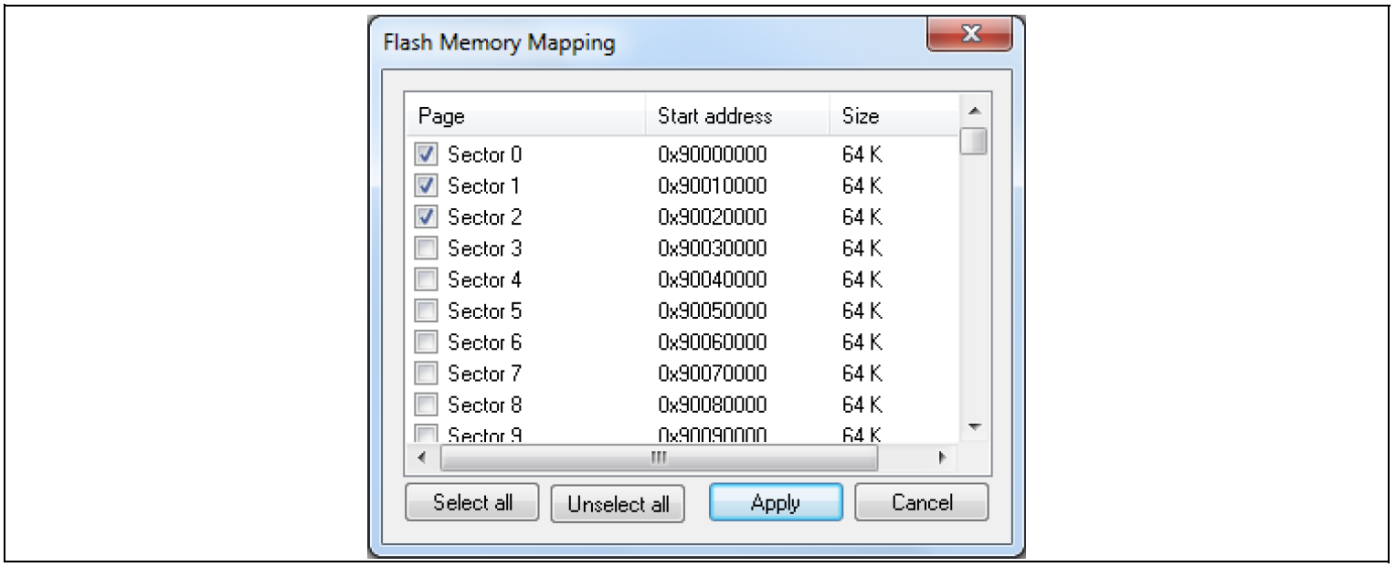


Figure 35. STM32 ST-LINK utility: erasing sectors



4.1.2 Программирование флэш-памяти QSPI с использованием IDE

Программирование флеш-памяти QSPI с помощью Keil

В пользовательском проекте код, область данных или и то, и другое должны быть запрограммированы в памяти QSPI, должны быть указаны компоновщику перед программированием.

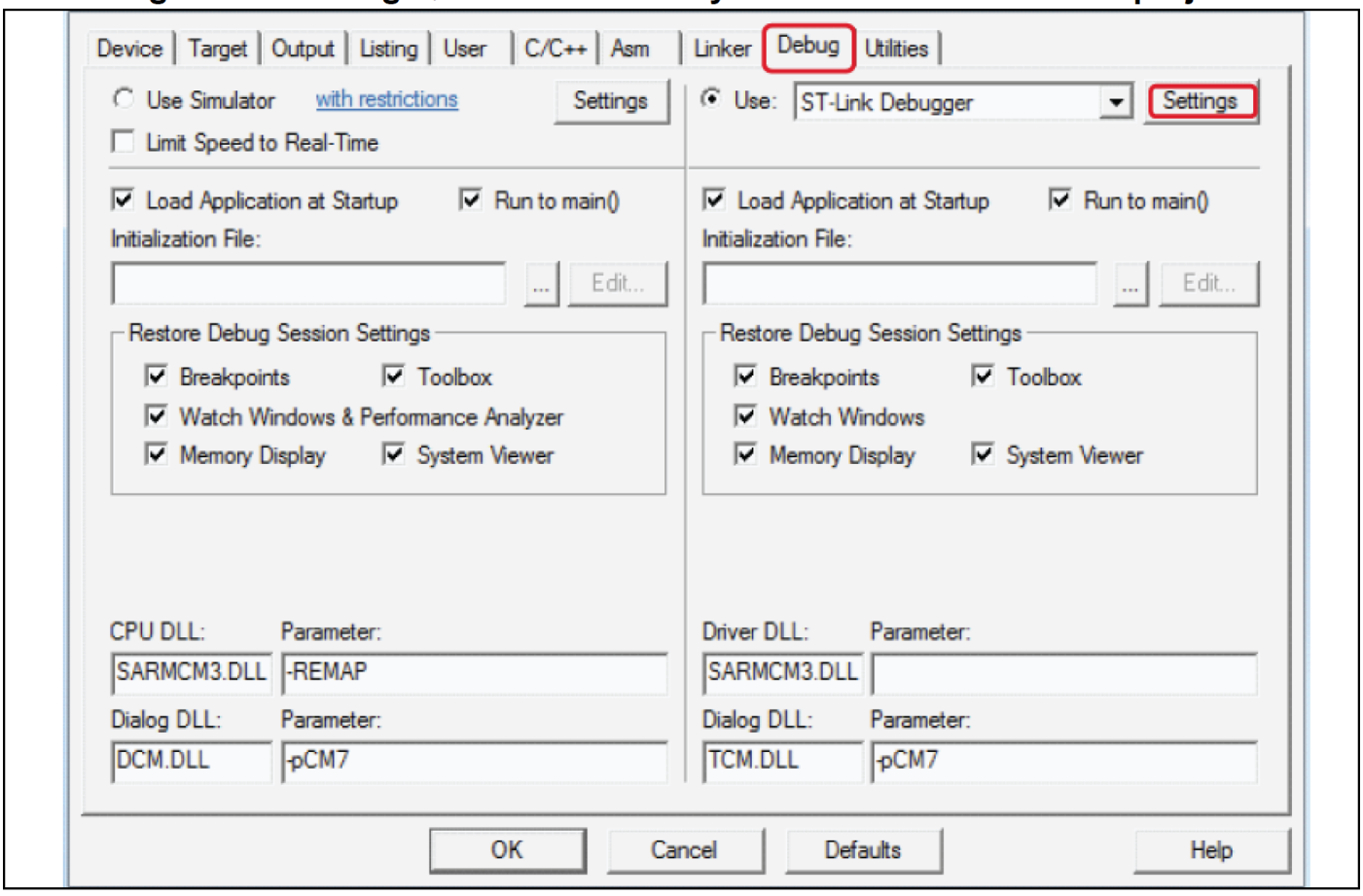
В следующем примере показано, как добавить выделенную область загрузки памяти QSPI в разбросанный файл Keil, где также создается область исполнения, что позволяет выполнять код из памяти QSPI.

```
; *************************************************************
; *** Scatter-Loading Description File generated by uVision ***
; *************************************************************
LR_IROM1 0x08000000 0x00100000 { ; load region size_region
ER_IROM1 0x08000000 0x00100000 { ; load address = execution address
*.o (RESET, +First)
*(InRoot$$Sections)
.ANY (+RO)
}
RW_IRAM1 0x20000000 0x00050000 { ; RW data
.ANY (+RW +ZI)
}
}
LR_QSPI 0x90000000 0xFFFFFFF {
ER_QSPI 0x90000000 0xFFFFFFF {
*.o (.textqspi)
}
}
```

Как добавить загрузчик флэш-памяти QSPI в Keil MDK-ARM

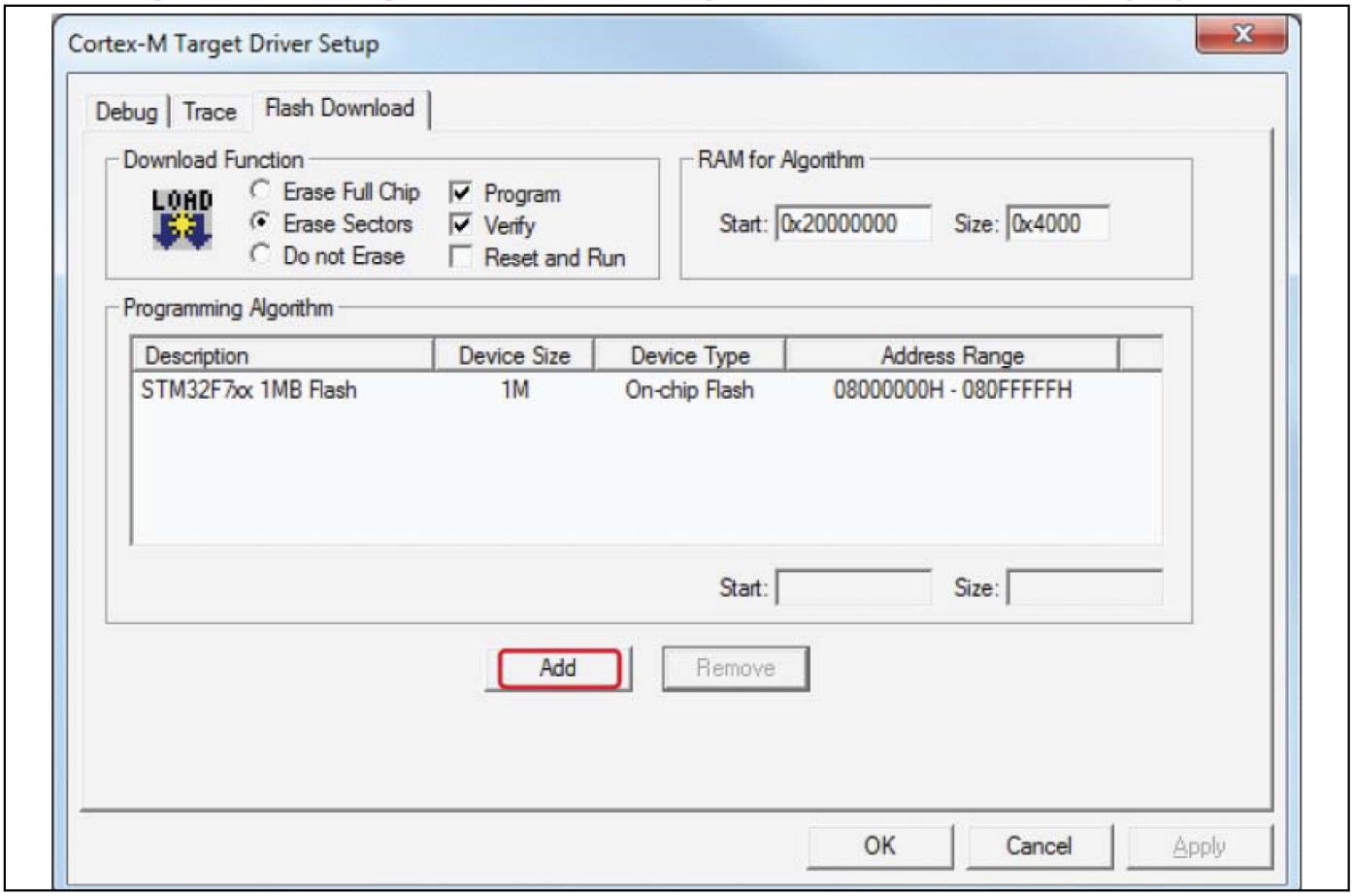
Чтобы добавить загрузчик флэш-памяти QSPI, перейдите в Options for Target «Опции для цели», затем в окне «Параметры цели» выберите вкладку «Отладка» и нажмите кнопку «Настройки», как показано на рисунке ниже.

Figure 36. Adding QSPI Flash memory loader to Keil MDK-ARM project



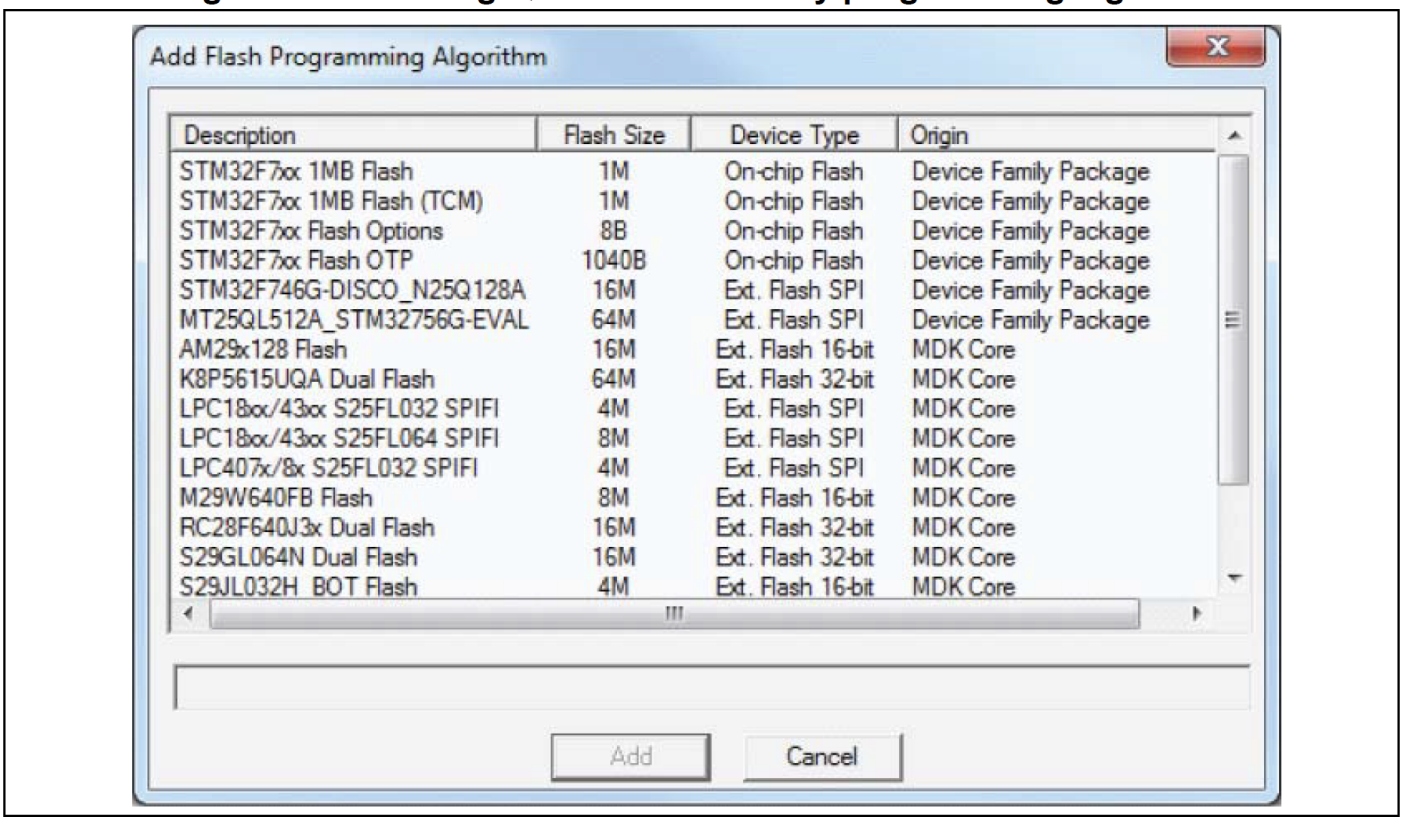
Загрузчик флэш-памяти QSPI будет добавлен в следующем окне.

Figure 37. Adding QSPI Flash memory loader to Keil MDK-ARM project



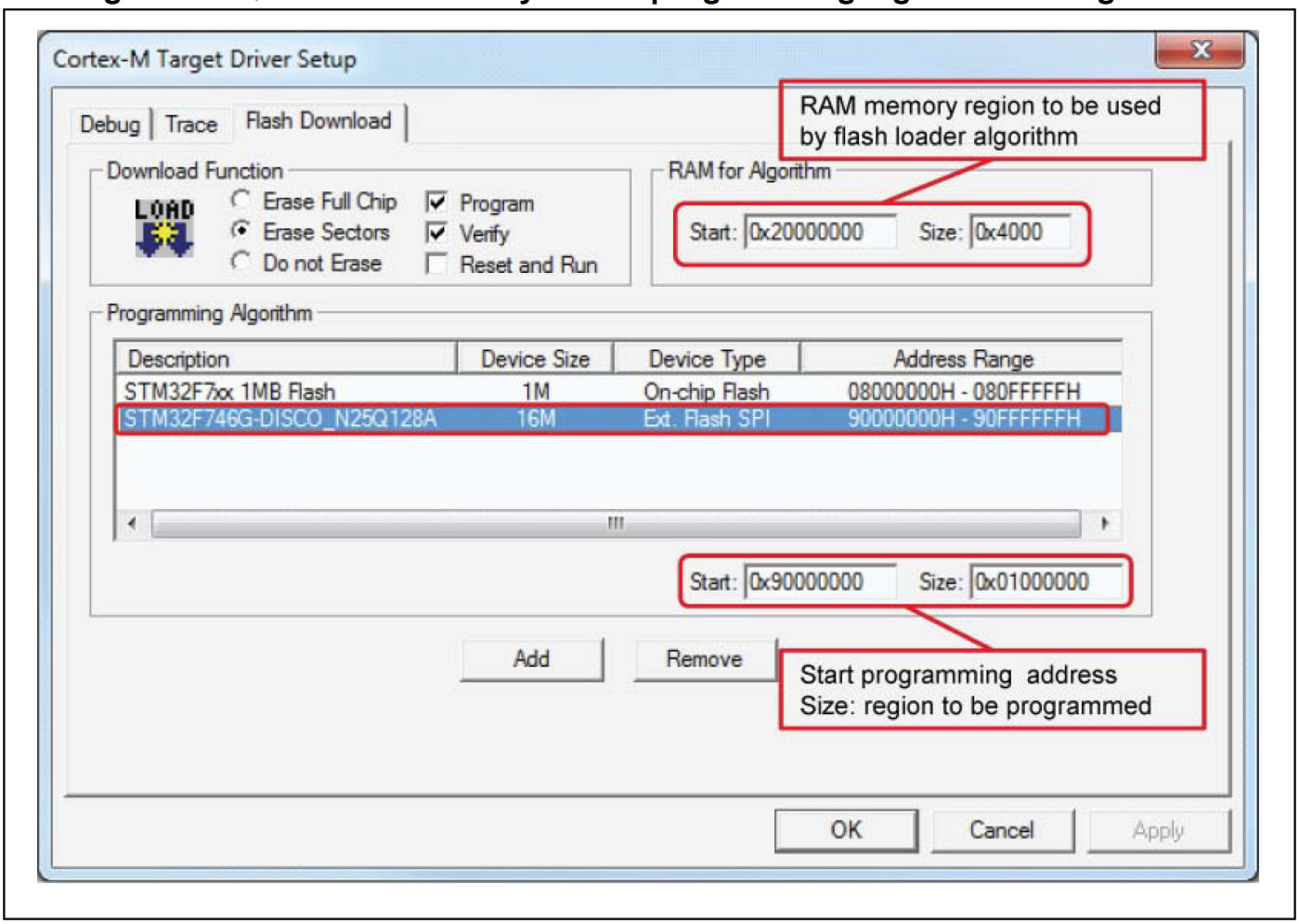
В следующем окне выберите из списка соответствующий загрузчик Flash памяти:

Figure 38. Selecting QSPI Flash memory programming algorithm



После добавления соответствующего загрузчика флэш-памяти он появляется в списке алгоритмов программирования, как показано на рисунке ниже.

Figure 39. QSPI Flash memory loader programming algorithm configuration



После добавления загрузчика флэш-памяти QSPI программирование можно выполнить, нажав кнопку «Загрузить» или нажав клавишу F8 на клавиатуре.

Примечание:

Программируемая область определяется по умолчанию во внешнем загрузчике и может быть изменена путем изменения начального адреса и полей размера.

Как перейти к программе QSPI Flash memory только один раз

Во время разработки приложения пользователю необходимо отладить свой проект и много раз загрузить код во внутреннюю флэш-память. Для каждой операции загрузки-прошивки также загружается флэш-память QSPI, что делает операцию загрузки слишком длинной. В этом случае, если пользователь уже загрузил данные в QSPI Flash и не нуждается в повторении операции, пользователь может просто сгенерировать файл определения символа, связанный с данными QUADSPI, и добавить его в проект. Файл определения символа представляет собой файл * .txt для Keil MDK-ARM и файл * .o для IAR EWARM; он должен заменить в проекте исходные файлы исходного кода (* .c или * .h), которые уже были запрограммированы.

Другая более простая альтернатива состоит в простом удалении уже добавленного загрузчика флэш-памяти QSPI.

STM32 system workbench

Системная рабочая среда STM32 не поддерживает внешний Flash-загрузчик QSPI, вместо этого пользователь может использовать утилиту STM32 ST-LINK, как описано ранее.

4.2 Хранение и стирание данных на лету во время работы приложения

4.2.1 Хранение данных

В некоторых приложениях полезно использовать внешнюю флэш-память QSPI для хранения данных. В этом случае интерфейс QUADSPI должен быть сконфигурирован в режиме косвенной записи, чтобы обеспечить оперативное хранение данных. Данные, которые должны быть сохранены, могут быть результатом обработки, такой как обработка сигналов для аудиоприложений; он также может хранить изображения, снятые камерой через интерфейс цифровой камеры DCMI или любые другие данные.

Перед каждой операцией программирования должна быть выполнена операция стирания. Для этой операции должен использоваться режим косвенной записи.

Примечание:

Скорость записи на внешнюю флэш-память ниже, чем скорость чтения. Поскольку операция программирования занимает значительный период времени, пользователь может использовать режим опроса флага состояния, чтобы опрашивать регистр состояния памяти, и после завершения операции может быть разрешено прерывание.

Шаги для выполнения операции программирования перечислены ниже:

1. Настройте интерфейс QUADSPI в регистрах QUADSPI_CR и QUADSPI_DCR.
2. Настройте флэш-память: включите запись во Flash, установите число фиктивных циклов в зависимости от скорости док-станции.
3. Настройте формат кадра в регистре QUADSPI_CCR и запустите последовательность программирования.
4. После завершения последовательности программирования. Сконфигурируйте QUADSPI в режиме опроса, чтобы проверить состояние памяти и убедиться, что она готова.

На следующем рисунке показан пример последовательности программирования страницы, где размер страницы составляет 256 байт.

Примечание:

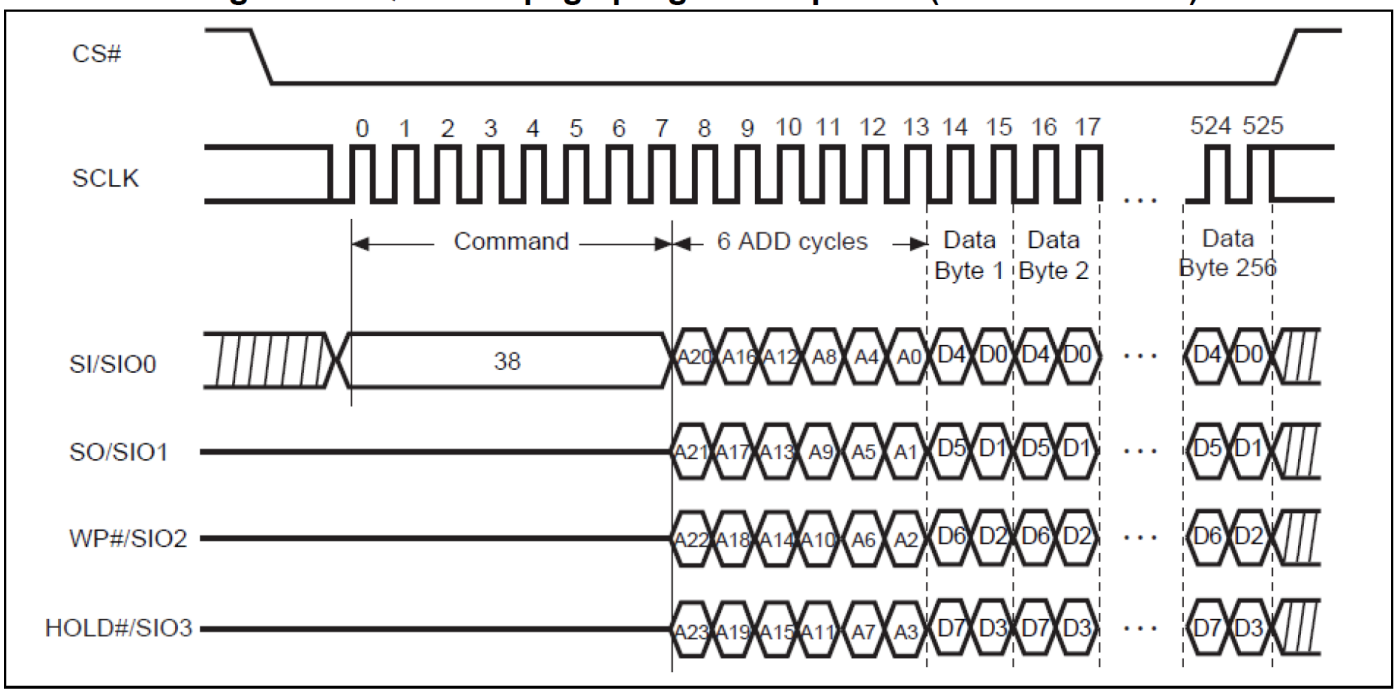
Байт 1 данных сначала записывается в регистр данных QUADSPI_DR, затем байт 256 данных является последним. При записи 32-битного слова в регистр QUADSPI_DR обратите внимание, что байт LSB записывается сначала в FIFO, а затем передается первым.

Для программирования памяти QSPI можно использовать либо ЦП с прерываниями, либо DMA.

1. Программирование памяти QSPI с использованием режима косвенной записи

При использовании режима косвенной записи все операции программирования обрабатываются программным обеспечением путем прямой записи в регистр QUADSPI_DR. Прерывание генерируется при идентификации завершения передачи или при достижении порога FIFO.

Figure 40. Quad I/O page program sequence (command 0x38)



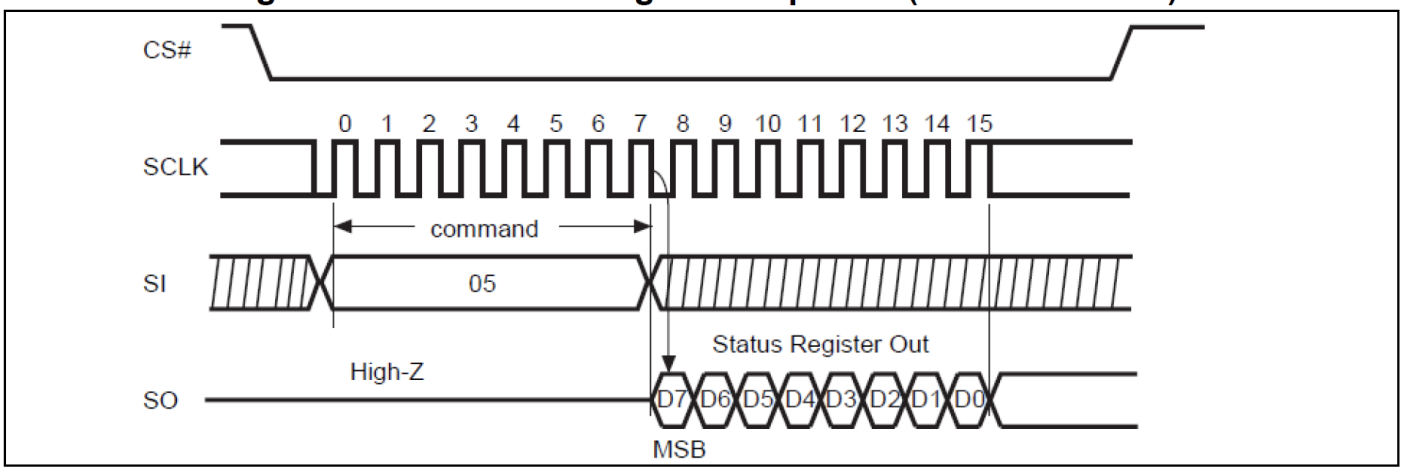
2. Программирование памяти QSPI с использованием режима косвенной записи с DMA

Как правило, рекомендуется использовать DMA для программирования памяти QSPI с использованием режима косвенной записи, поскольку он разгружает ЦП, тем не менее окончательная рекомендация зависит от пользовательского приложения. В некоторых случаях, когда объем данных, которые должны быть записаны в память, относительно мал, нет необходимости использовать DMA. Как только DMA настроен и операция программирования запущена, вмешательство ЦП не требуется, и операция завершается автономно. Для получения более подробной информации об использовании DMA, обратитесь к Разделу 2.5.2: Использование DMA.

3. Использование режима опроса статуса

Пользователь может использовать этот режим для опроса регистра состояния памяти. На рисунке ниже показан пример последовательности чтения регистра состояния.

Figure 41. Read status register sequence (command 0x05)



4.2.2 Стирание данных

Операция стирания должна выполняться перед каждой операцией программирования. Режим косвенной записи должен использоваться как для этой операции, так и для программирования.

Конфигурация, необходимая для выполнения операции стирания, такая же, как конфигурация, необходимая для выполнения операции программирования, за исключением того, что данные не должны записываться в память. Поскольку фаза данных не требуется, требуются только этапы инструкции и адреса.

Поскольку операция удаления занимает определенный период времени, пользователь может использовать режим опроса флага состояния, чтобы опрашивать регистр состояния памяти. После завершения операции можно включить прерывание:

Шаги для выполнения операции удаления перечислены ниже:

1. Настройте интерфейс QUADSPI в регистрах QUADSPI_CR и QUADSPI_DCR.
2. Настройте флэш-память: включите запись во Flash
3. Настройте формат кадра и косвенный режим в регистре QUADSPI_CCR.
4. При необходимости отправьте команду и адрес удаления (адрес необходим для удаления сектора).
5. Поместите интерфейс QUADSPI в опрос состояния, чтобы опросить окончание операции.

Большинство устройств флэш-памяти поддерживают удаление секторов и операцию полного удаления микросхемы, а некоторые из них поддерживают дополнительную операцию стирания, обеспечивая большую гибкость для пользовательских приложений. Обратитесь к таблице производителя для получения дополнительной информации о поддерживаемых операциях стирания.

Примечание:

Если объем используемой памяти превышает 16 МБ, необходимо использовать 4-байтовый адресный режим, в этом случае пользователь должен выбрать 4-байтовую команду из таблицы данных памяти.

Секторная последовательность стирания

Чтобы удалить сектор в памяти, необходимо отправить команду удаления сектора и адрес начального сектора.

Пример: чтобы выполнить операцию удаления сектора в памяти MICRON N25Q512A, регистр QUADSPI_CCR должен быть настроен, как показано ниже:

```
QUADSPI->CCR = 0x000025D8;
 /* Instruction= 0xD8; IMODE = 0x01; ADMODE = 0x01; ADSIZE = 0x02 */
QUADSPI->AR = 0x00000000;
/* Address 0x00000000 is sent to erase the first sector */
```

Ниже приведен пример последовательности удаления сектора:

Последовательность стирания полного чипа (массовое стирание)

Нет необходимости отправлять адрес, чтобы стереть всю память, достаточно отправить команду. Ниже приведен пример последовательности удаления чипа:

Figure 42. Sector erase sequence

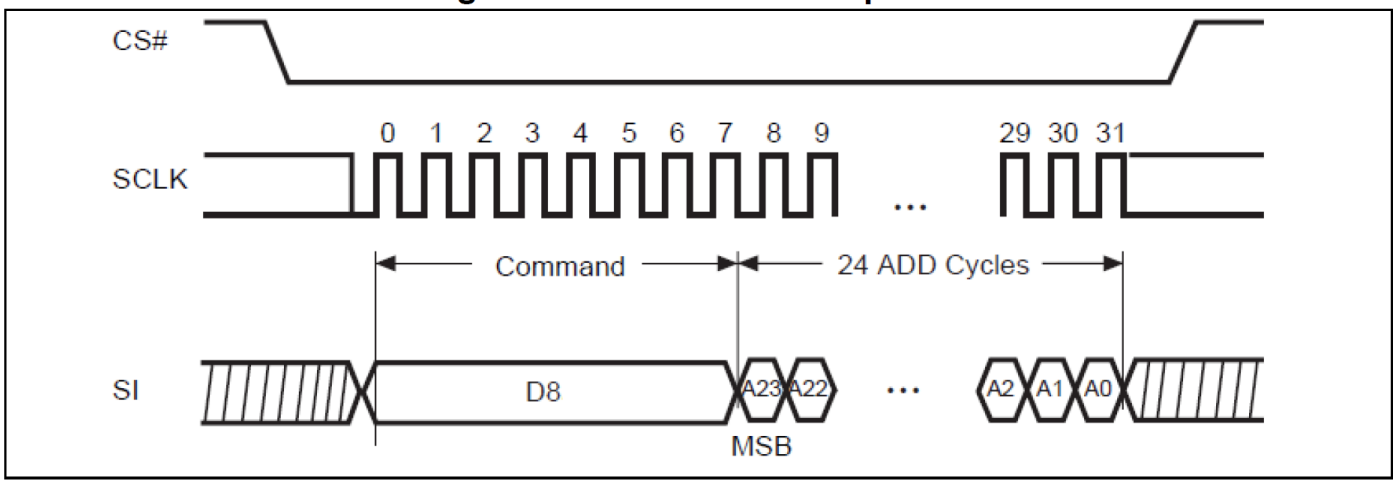
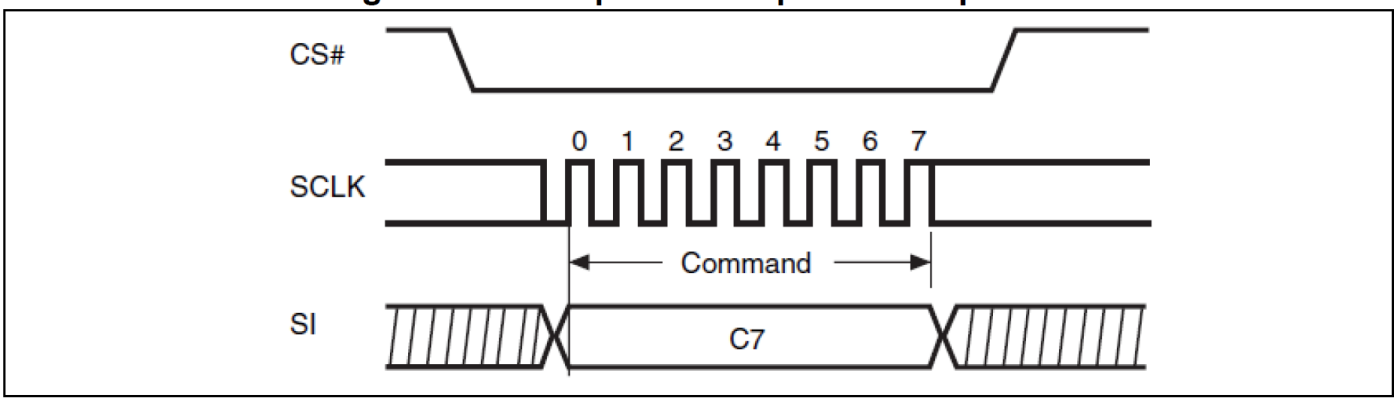


Figure 43. Example: full chip-erase sequence



5 Примеры применения QUADSPI

В этом разделе приведены некоторые типичные примеры использования QUADSPI, показывающие, как использовать интерфейс в режиме косвенного доступа, в режиме опроса флагов состояния и в режиме отображения памяти.

Некоторые из этих примеров представлены в пакете прошивки STM32Cube, в то время как другие основаны на других замечаниях по применению, также доступных на веб-сайте ST. Некоторые примеры реализации оборудования также приведены в конце этого раздела.

В этом разделе описываются следующие варианты использования:

- Режим отображения в памяти: чтение данных в графическом приложении
- Режим отображения в память: выполнение кода из флэш-памяти QSPI
- Косвенный режим: хранение данных на лету во время работающего приложения
- Косвенный режим: стирание данных
- Пример аппаратной реализации

5.1 Чтение данных из памяти QSPI: графическое приложение

Поскольку графические приложения требуют большого количества статических данных (библиотеки шрифтов, стиль HMI, значки), внешняя флэш-память QSPI может быть полностью выделена для хранения статических данных. В этом разделе представлены два графических варианта использования, где QUADSPI используется для хранения данных:

- Генерация содержимого кадрового буфера из памяти QSPI
- Отображение изображений непосредственно из памяти QSPI

5.1.1 Генерация содержимого буфера кадра из памяти QSPI

В этом разделе приведен пример, в котором периферийное устройство DMA2D считывает изображения, хранящиеся во внешней флэш-памяти QSPI, для записи их на внешнюю SDRAM. Он также подготавливает содержимое буфера кадров, которое будет считываться LTDC, а затем отображаться на TFT-LCD дисплее.

Пример основан на демонстрации программного обеспечения из пакета микропрограммы STM32F7Cube, доступного на веб-сайте ST. Этот пример включает в себя один проект, который был разработан для платы STM32F746G-DISCO.

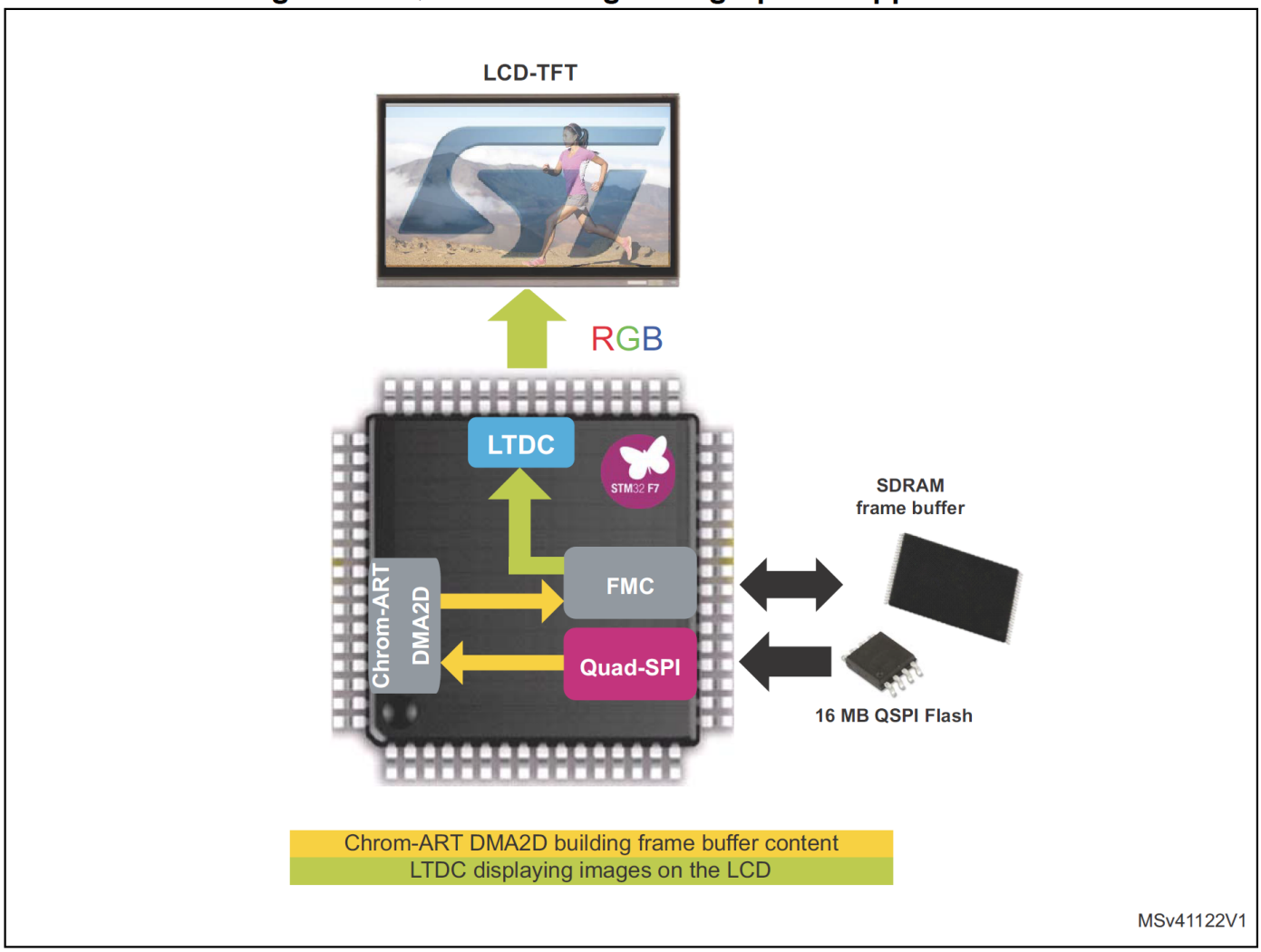
Проект находится по следующему пути: STM32Cube_FW_F7_V1.14.0 \ Projects \ STM32746G-Discovery \ Applications \ QSPI \ QSPI_perfs

Описание использования

В этом примере описано, как использовать изображения, хранящиеся во флэш-памяти QSPI, для создания содержимого буфера кадров.

DMA2D используется для визуализации нескольких анимированных слоев на LCD-TFT. Все изображения и значки хранятся во флэш-памяти QSPI. DMA2D используется для передачи изображений из флэш-памяти QSPI в память SDRAM. Во время этой передачи измеряется время передачи, затем рассчитывается скорость передачи и отображается на TFT-LCD.

Figure 44. QUADSPI usage in a graphical application



На той же скорости DMA2D смешивает изображения и загружает их в SDRAM (кадровый буфер), в то время как LTDC обновляет LCD-TFT с частотой 60 Гц.

Хранение изображений и значков для использования в приложении

Все изображения и значки, необходимые для приложения, должны храниться во внешней памяти QSPI. В проекте доступно шесть изображений и три иконки, все они включены в файл «main images.c», где каждое из них определяется как константа в отдельном заголовочном файле.

В этой конфигурации проекта только два изображения (img2 и img6 определены соответственно в файлах «img2.h» и «img6.h») и три значка (icon_S, icon_T и icon_M определены соответственно в файлах «icon_S.h», «icon_T»). .h »и« icon_M.h ») используются и должны быть сохранены во флэш-памяти QSPI.

Значки и изображения определяются как константы. Чтобы сохранить эти значки и изображения в памяти QSPI, они помещаются в специальный раздел «.textqspi», как описано ниже:

- определение Img2 и img6

__attribute __ ((section («. textqspi»))) const unsigned char img2 [261120] = {}
__attribute __ ((section («. textqspi»))) const unsigned char img6 [261120] = {}

- Определение Icon_S, icon_T и icon_M:

__attribute __ ((section («. textqspi»))) const unsigned char icon_S [30800] = {}
__attribute __ ((section («. textqspi»))) const unsigned char icon_T [30800] = {}
__attribute __ ((section («. textqspi»))) const unsigned char icon_M [42000] = {}

Как описано в Разделе 4.1.1: Программирование флэш-памяти QSPI с использованием IDE, для программирования памяти QSPI с использованием Keil MDK-ARM или IAR EWARM необходимо создать специальный раздел для всех данных, подлежащих программированию.

После раздела QUADSPI «.textqspi» в разбросанном файле Keil MDK-ARM:

LR_IROM2 0x90000000 0xFFFFFFF { ; load region size_region
ER_IROM2 0x90000000 0xFFFFFFF { ; load address = execution address
*.o (.textqspi)
}
}

Чтобы запрограммировать данные в QSPI Flash, пользователь должен сначала проверить, добавлен ли в проект загрузчик QSPI Flash (Keil MDK-ARM или IAR EWARM). Если он уже добавлен, пользователь может просто построить проект и запрограммировать память. Для получения дополнительной информации о конфигурации проекта обратитесь к файлу readme в каталоге проекта.

QUADSPI интерфейс и конфигурация памяти

QUADSPI сконфигурирован в режиме отображения памяти, чтобы позволить DMA2D считывать изображения и значки из QUADSPI и записывать в SDRAM через интерфейс FMC.

Cortex®-M7 работает на частоте 200 МГц, а частота QUADSPI составляет 100 МГц. При этой тактовой частоте максимальная достижимая пропускная способность составляет 50 Мбайт / с. QUADSPI настроен для работы в режиме 1-4-4.

Поскольку встроенная память QSPI не поддерживает режим DDR, используется режим SDR.

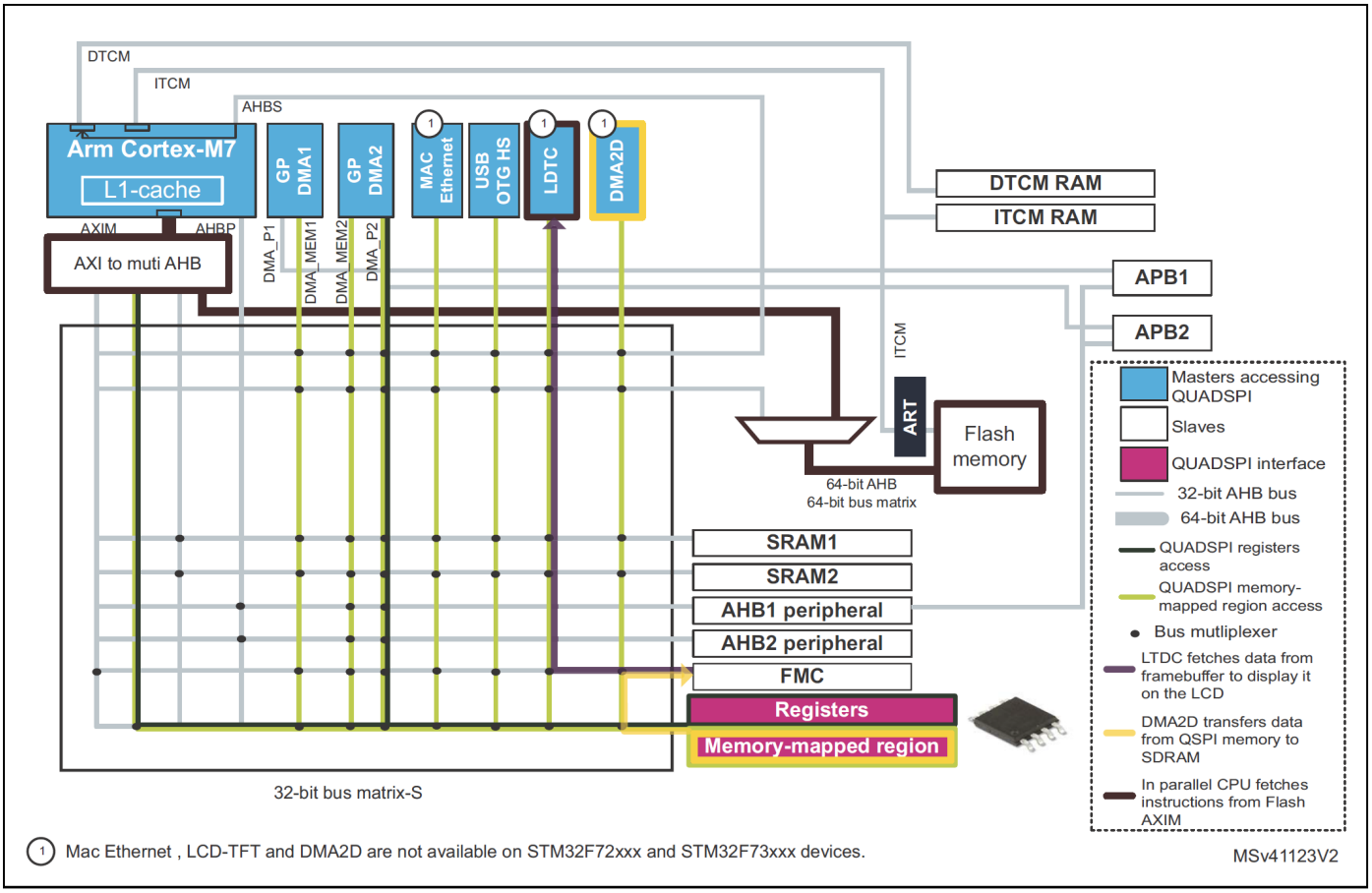
Используемая команда чтения - QUAD INPUT / OUTPUT FAST READ (0xEB), которая позволяет отправлять адрес в четыре строки и считывать данные в четыре строки, в то время как команда отправляется в одну строку.

Интеллектуальная архитектура серии STM32F7 допускает разгрузку процессора. DMA2D действует как мастер AHB и выполняет все передачи из QUADSPI в буфер кадров (SDRAM) вместо ЦПУ. В то же время, когда LTDC отображает графику, Cortex®-M7 может выполнять код из внутренней флэш-памяти.

Эта демонстрация показывает, как связать 16-Мбайтную внешнюю флэш-память с устройством STM32F7x6.

Флэш-память QSPI используется в качестве энергонезависимой поддержки, содержащей все графические элементы (значки, изображения), необходимые для приложения.

Figure 45. DMA2D reading images from Quad-SPI to build frame buffer content



5.1.2 Отображение изображений непосредственно из памяти QSPI

Как упоминалось ранее, все мастера AHB могут обращаться к памяти QSPI в режиме отображения памяти. Это очень интересная функция, когда приложение требует отображать сохраненную графику QSPI непосредственно на ЖК-дисплее без какого-либо вмешательства процессора.

В этом разделе приведен пример, в котором периферийное устройство LTDC считывает изображение, хранящееся во внешней флэш-памяти QSPI. Пример основан на демонстрации программного обеспечения из заметки приложения «Управление низким энергопотреблением на микроконтроллерах серии STM32F7» (AN4749).

AN4749 поставляется с программным пакетом X-CUBE-LPDEMO-F7 и доступен на веб-сайте ST. Он включает в себя один проект, который был разработан для платы STM32F746G-DISCO.

Примечание:

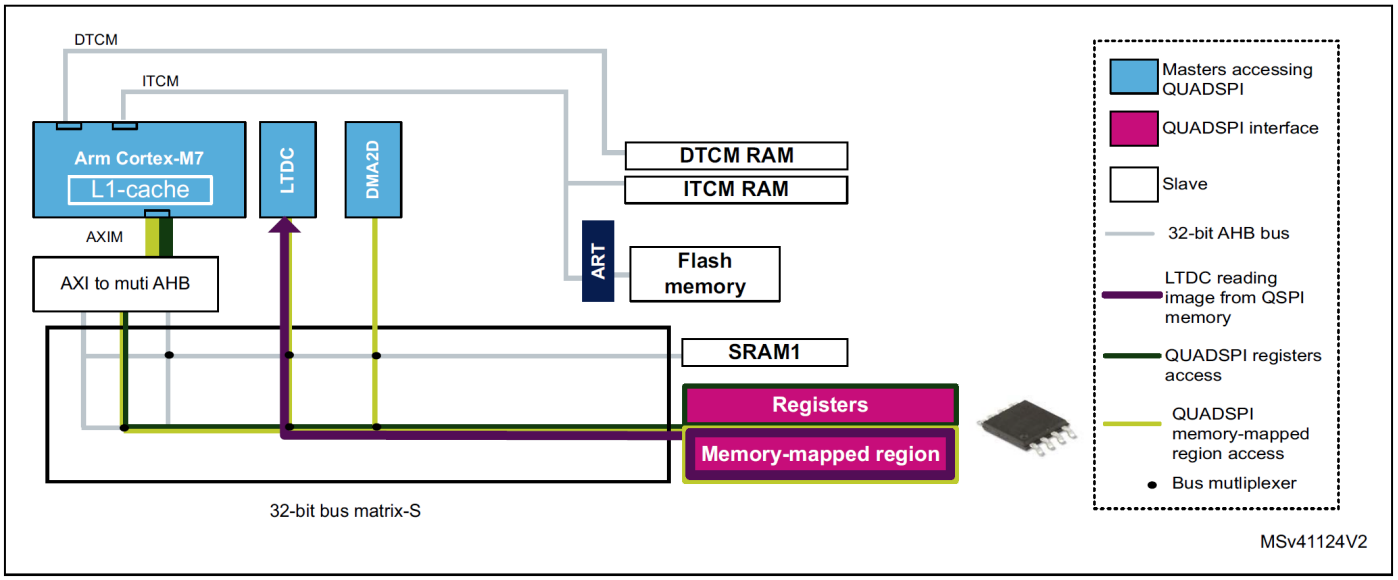
В этом примечании по применению описано только использование QUADSPI в данном случае.

Описание демонстрации программного обеспечения

Заставка изображения находится в трех разных областях: внутренняя флэш-память, внешняя флэш-память QSPI по адресу 0x90000000 и SDRAM по адресу 0xC0000000. Пользователь должен выбрать, из какого региона LTDC должен отображать изображение.

При первом запуске время отображается на LCD-TFT в течение пяти секунд, затем появляется меню, показывающее окно, в котором пользователь может выбрать ячейку памяти, из которой считывается изображение. Если выбрана флэш-память QSPI, то LTDC считывает изображение непосредственно из внешней флэш-памяти. Смотрите графическое представление этого варианта использования ниже.

Figure 46. LTDC reading an image directly from QSPI memory



Хранение изображения, которое будет использоваться для приложения

Для этого приложения изображение должно быть сохранено только один раз для использования в конечном приложении (конечный продукт).

Изображение для программирования определяется в файле RGB565_480x272.h как постоянные данные и находится в выделенном разделе RGB565_480x272_qspi. Процесс размещения изображения в этом разделе с помощью Keil MDK-ARM описан ниже:

```
const uint16_t RGB565_480x272_QSPI [] __attribute __ ((section («. RGB565_480x272_qspi»)));
/ * Keil MDK-ARM: размещение изображения в разделе .RGB565_480x272_qaspi * /
const unit16_t RGB565_480x272_QSPI [] = {};
```

Как описано в Разделе 4.1.1: Программирование флэш-памяти QSPI с использованием IDE, для программирования памяти QSPI с использованием Keil MDK-ARM или IAR EWARM необходимо создать специальный раздел для всех данных, подлежащих программированию.

После раздела QUADSPI «.RGB565_480x272_qspi» в разбросанном файле Keil MDK-ARM:

```
 LR_IROM2 0x90000000 0xFFFFFFF {; область загрузки size_region
ER_IROM2 0x90000000 0xFFFFFFF {; адрес загрузки = выполнение
адрес
* .o (.RGB565_480x272_qspi);
 }
}
```

Чтобы запрограммировать данные во флэш-память QSPI, пользователь должен сначала проверить, добавлен ли в проект загрузчик флэш-памяти QSPI (Keil MDK-ARM или IAR EWARM). Если он уже добавлен, пользователь может просто построить проект и запрограммировать память. Подробнее о конфигурации проекта см. В файле readme в каталоге проекта.

Интерфейс QSPI и конфигурация памяти

QUADSPI сконфигурирован в режиме отображения памяти, чтобы позволить LTDC считывать изображение непосредственно из памяти QSPI.

После настройки QUADSPI для отображения изображения, хранящегося в памяти QSPI, в файле LCDConf.c вызывается следующий API:

HAL_LTDC_ConfigLayer (& hltdc_F, & pLayerCfg, 0).

pLayerCfg - указатель на структуру LTDC_LayerCfgTypeDef, которая содержит адрес изображения в памяти QSPI.

5.2 Выполнение из внешней памяти QSPI: увеличьте объем внутренней памяти

Использование внешней памяти QSPI позволяет расширить общее доступное пространство памяти приложения. Устройство STM32F746, встроенное в платы Discovery или Evaluation, оснащено 1-мегабайтной флэш-памятью; следовательно, подключение внешней флэш-памяти QSPI увеличивает доступное пространство памяти до 64 Мбайт для оценочной платы.

В этом разделе описывается, как использовать внешнюю память QSPI для расширения внутренней флэш-памяти, чтобы разрешить выполнение кода из внешней памяти QSPI. Демонстрация программного обеспечения из примечания по применению Архитектура и производительность системы серии STM32F7 (AN4667) выбрана в качестве справочной, чтобы показать, как:

- Настройте QSPI в режиме отображения памяти во время инициализации системы и перед переходом к коду памяти QSPI.
- Поместите код приложения во внешнюю память QSPI

Описание демонстрации программного обеспечения

AN4667 поставляется со встроенным пакетом программного обеспечения X-CUBE-32F7PERF, доступным на веб-сайте ST, который включает в себя два проекта: STM32F7_performances и STM32F7_performances_DMAs. Оба проекта обеспечены цепочкой инструментов Keil MDK-ARM. Этот раздел посвящен проекту STM32F7_performances.

Проект STM32F7_performances показывает производительность устройства STM32F746 при выполнении кода из внутренней и внешней памяти. Он включает в себя семь конфигураций, каждая из которых позволяет выбирать данные и места

расположения кода. Поскольку этот раздел посвящен описанию выполнения кода из памяти QSPI, для демонстрационного описания принимаются во внимание только конфигурации 6_1-Quad-SPI_rwRAM-DTCM и 6_2-Quad-SPI_rwRAM-DTCM.

Количество циклов, потребляемых процессом FFT, рассчитывается на основе системного таймера. Пример был запущен на STM32756G-EVAL, и результаты отображаются на LCD-TFT или на гипертерминале через UART или в средстве просмотра IDE printf.

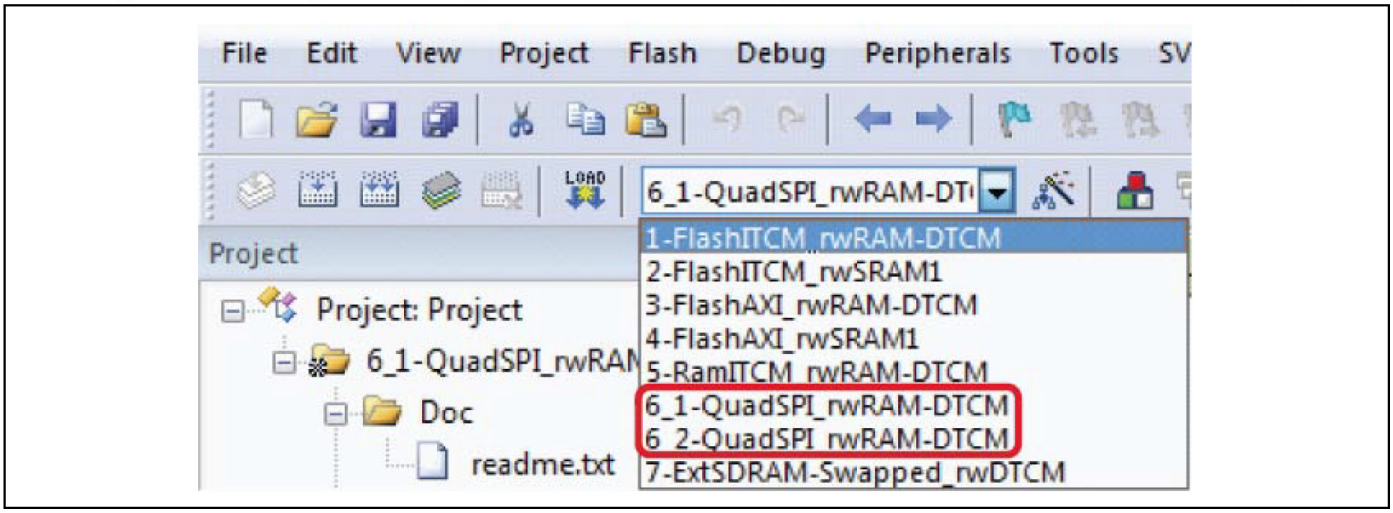
Конфигурация отображается и показывает текущую конфигурацию проекта, частоту системы, различные конфигурации кэшей, ART, ART-Prefetch (ON / OFF) и конфигурацию памяти в случае внешней памяти (SDRAM или QSPI).

Демонстрация программного обеспечения разработана для платы STM32756G-EVAL и может быть легко адаптирована к плате STMF746-DISCO. Для получения более подробной информации об этой заявке см. Документ AN4667, доступный на сайте www.st.com.

Проект STM32f7_performances находится по следующему пути: x-cube-32F7perf \ STM32CubeExpansion_AN4667_F7_V4.0.0 \ Projects \ STM32756G_EVAL \ stm32f7_performances.

На рисунке ниже показаны две конфигурации проекта в Keil MDK-ARM, описанные в этом документе.

Figure 47. Project configurations: executing code from QSPI Flash memory



Конфигурация проектов

Для обеих конфигураций проекта, 6_1-QuadSPI_rwRAM-DTCM и 6_2-QuadSPI_rwRAM-DTCM, пользователь может изменить нужные параметры QUADSPI в поле «Параметры для целевого объекта», как показано на следующем рисунке.

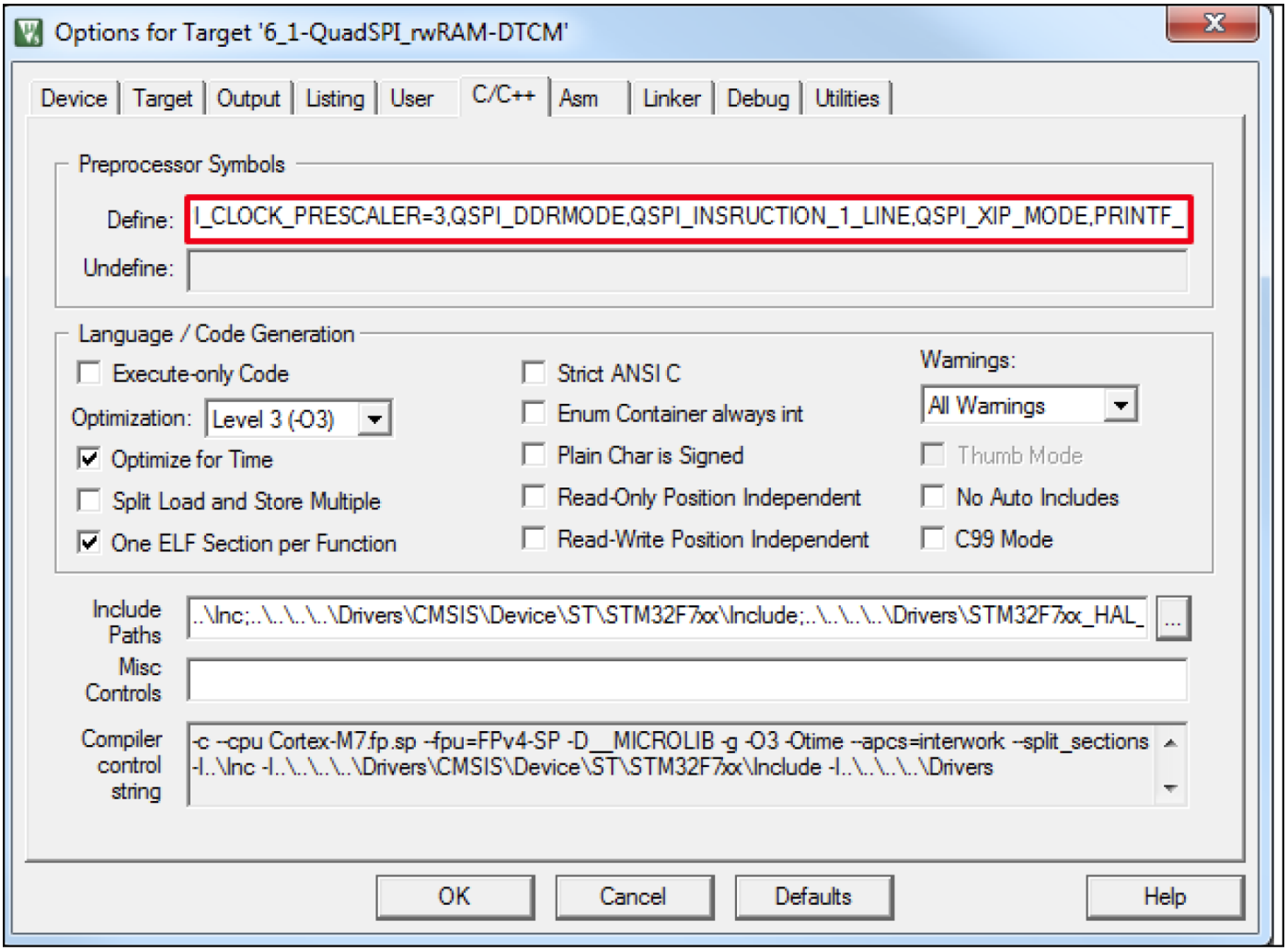
Обратите внимание, что тактовая частота операционной системы во время инициализации системы составляет 16 МГц, поэтому в этот момент QSPI_CLK = fAHB / 1 = 16 МГц (по умолчанию прескалер = 0). После завершения инициализации системы ЦПУ переходит к основной функции (в файле arm_fft_bin_example_f32.c), где выполняется конфигурация системных часов. Системные часы настроены на работу на частоте 216 МГц.

Обе конфигурации проекта имеют следующие настройки QUADSPI:

- QSPI_CLOCK_PRESCALER = 3
- Системные часы 216 МГц => QSPI_CLK = 54 МГц

• QSPI_DDRMODE => DDR mode enabled (Режим DDR включен)
• QSPI_INSTRUCTION_1_LINE => instruction is issued in one line (инструкция выдается в одну строку)
• QSPI_XIP_MODE => execute in place with SIOO enabled (выполнить на месте с включенным SIOO.)

Figure 48. Changing QUADSPI configuration in the project settings



5.2.1 Настройка QSPI в режиме отображения памяти во время инициализации системы

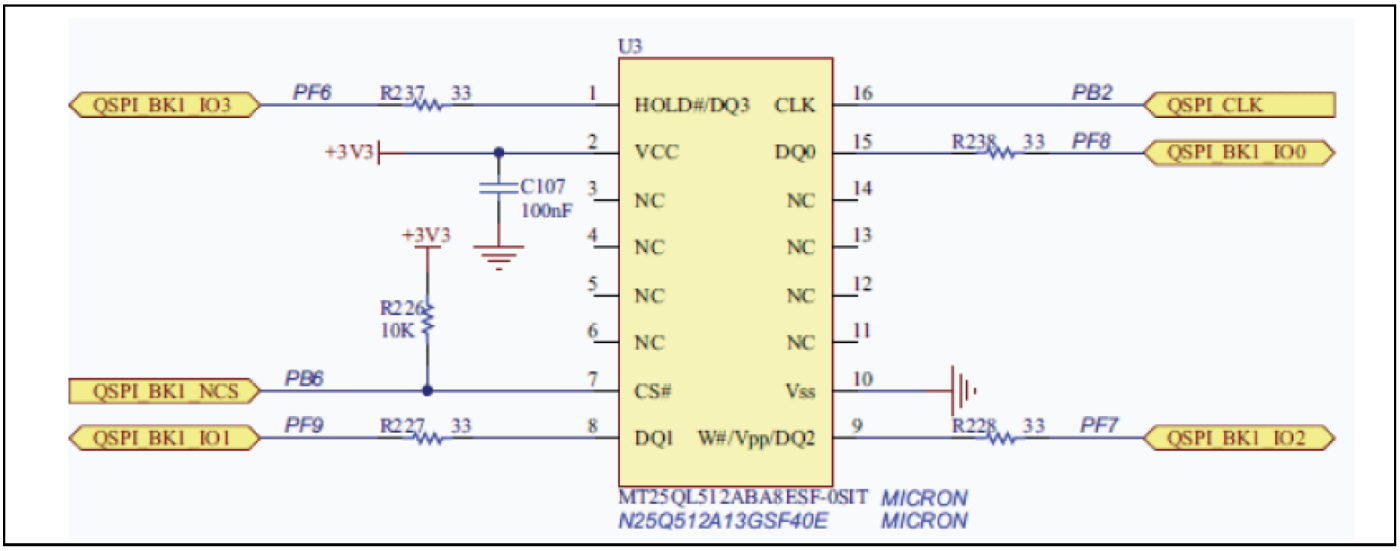
Загрузка из флэш-памяти QSPI не поддерживается, поэтому пользователь может загружаться из внутренней флэш-памяти, настраивать периферийное устройство QUADSPI в режиме отображения памяти и затем переходить к выполнению из внешней памяти QSPI.

В этом примере конфигурация QUADSPI выполняется во время инициализации системы в функции SystemInit_ExtMemCtl () в файле system_stm32f7xx.c. Все необходимые конфигурации периферийных устройств и памяти QSPI выполняются в файле system-stm32f7xx.c и описаны ниже.

Конфигурация GPIO

Как показано на следующем рисунке, флэш-память QSPI подключена в режиме Quad I / O, поэтому для интерфейса QUADSPI необходимо настроить шесть GPIO.

Figure 49. QSPI Flash memory connection in STM32756-EVAL board



Конфигурация GPIO интерфейса QUADSPI в функции SystemInit_ExtmemCtl () описана ниже:

```
RCC-> AHB1ENR | = 0x00000022; / * Включить часы интерфейса GPIOB и GPIOF * /
/ * Подключите контакты PB2 и PB6 к альтернативной функции Quad-SPI * /
GPIOB-> AFR [0] = 0x0A000900; GPIOB-> AFR [1] = 0x00000000;
/ * Настройка контактов PBx в режиме альтернативной функции * /
GPIOB-> MODER | = 0x00002020;
/ * Настройка скорости выводов PBx на 100 МГц * /
GPIOB-> OSPEEDR | = 0x00003030;
/ * Настройка выводов PBx на двухтактный тип вывода * /
GPIOB-> OTYPER = 0x00000000;
 / * Нет подтягивания, раскрытие для выводов PBx * /
GPIOB-> PUPDR | = 0x00000000;
 / * Подключите контакты PF6, PF7, PF8 и PF9 к альтернативной функции Quad-SPI * /
GPIOF-> AFR [0] | = 0x99000000; GPIOF-> AFR [1] | = 0x000000AA;
 / * Настройка выводов PFx в режиме альтернативной функции * /
GPIOF-> MODER | = 0x000AA000;
/ * Настройка скорости выводов PFx на 100 МГц * /
GPIOF-> OSPEEDR | = 0x000FF000;
/ * Настройка выводов PFx Тип вывода для двухтактного * /
GPIOF-> OTYPER = 0x00000000;
/ * Без подтягивания, без подтягивания для выводов PFx * /
GPIOF-> PUPDR = 0x00000000;
```

Включение периферийного устройства QUADSPI

Для настройки памяти QSPI необходимо включить периферийное устройство QUADSPI, чтобы оно могло обмениваться данными с внешней памятью. / * Включить синхронизацию интерфейса QSPI * /

```
RCC-> AHB3ENR | = 0x00000002;
/ * Сбросить QSPI периферийное устройство * /
RCC-> AHB3RSTR | = (RCC_AHB3RSTR_QSPIRST);
```

```
/* Сброс */
RCC-> AHB3RSTR & = ~ (RCC_AHB3RSTR_QSPIRST);
/ * Отмена сброса * /
 / * Включить Quad-SPI периферийное устройство * /
QUADSPI-> CR = 0x00000001;
```

Конфигурация памяти QSPI

После включения периферийного устройства QUADSPI можно связываться с памятью QSPI, чтобы настроить ее в желаемом режиме работы. Ссылаясь на таблицу данных MICRON N25Q512A, режим SDR single-SPI можно использовать для связи с памятью, что означает, что команда, адрес и данные отправляются в одну строку для настройки памяти.

Обратите внимание, что косвенный режим Quad-SPI должен использоваться для конфигурации внешней памяти.

Сбросте память QSPI:

Перед сбросом регистров памяти QSPI команда RESET ENABLE 0x66 должна быть отправлена в режиме 1-0-0, поэтому только команда отправляется в косвенном режиме, в то время как фазы адреса и данных пропускаются.

```
/ * Отправить команду RESET ENABLE (0x66), чтобы разрешить сброс регистров памяти * /
QUADSPI-> CCR = 0x00000166;
```

Для сброса регистров памяти QSPI команду RESET 0x99 следует вводить в режиме 1-0-0, только команда отправляется в косвенном режиме, а фазы адреса и данных пропускаются.

```
/ * Отправить команду RESET (0x99) для сброса регистров памяти * /
QUADSPI-> CCR = 0x00000199;
```

Настройте память для приема команд в четыре строки:

В обеих конфигурациях проекта память QSPI сконфигурирована для приема команд в одну строку (определено QSPI_INSTRUCTION_1_LINE), но память может быть сконфигурирована для приема команд в четыре строки. Если это желаемый режим, пользователь должен использовать определение QSPI_INSTRUCTION_4_LINES вместо QSPI_INSTRUCTION_1_LINE.

Для включения операции QUADSPI регистр расширенной энергозависимой конфигурации внешней памяти N25Q512A должен быть сконфигурирован с 0x7F.

Чтобы записать 0x7F в регистр расширенной энергозависимой конфигурации, необходимо отправить команду 0x61. В этом случае QUADSPI должен отправить команду 0x61 и данные 0x7F, в то время как адрес отправлять не нужно, тогда используемый режим - 1-0-1.

```
/ * Включить запись cmd: 0x06. Это позволяет писать в расширенный энергозависимый регистр, чтобы инструкции могли быть записаны в 4 строки * /
while (QUADSPI-> SR & 0x20); / * Ожидание сброса флага занятости * /
QUADSPI-> CCR = 0x0106;


/ * Запись в регистр расширенной энергозависимой конфигурации внешней памяти (MT25QL512): разрешить ввод команды ввода-вывода quad. Запись в регистр расширенной энергозависимой конфигурации cmd = 0x61, конфигурация: 0x7F * /
while (QUADSPI-> SR & 0x20);
```

```
 / * Ожидание сброса флага занятости * / QUADSPI-> CCR = 0x01000161;
while (! (QUADSPI-> SR & 0x04));
/ * Дождаться установки флага FTF * / QUADSPI-> DR = 0x7F;
while (! (QUADSPI-> SR & 0x02)); / * Дождаться установки флага TCF * /
```

Включение режима SIOO (названного режимом XIP в техническом описании MICRON):

```
/ * Включить запись cmd: 0x06. Это сделано для того, чтобы разрешить запись в энергозависимый регистр конфигурации. Для более подробной информации обратитесь к таблице данных MT25QL512. * /
QUADSPI-> CCR = (0x0106 | QSPI_CCR_IMODE);
while (QUADSPI-> SR & 0x20); / * Ожидание сброса флага занятости * /
/ * Сконфигурировать Quad-SPI в режиме 1-0-1 для записи в VOLATILE CONFIGURATION REGISTER * /
QUADSPI-> CCR = (0x00000081 | QSPI_CCR_IMODE | QSPI_CCR_DMODE);
while (! (QUADSPI-> SR & 0x04)); / * Дождаться установки флага FTF * /
/ * Записать 0x83 в регистр энергозависимой конфигурации: бит 3 = 0 для включения XIP и биты [7: 4] = 8 для установки восьми фиктивных циклов * /
QUADSPI-> DR = ((MEM_DUMMY_CYCLE_XIP << 4) | 0x3);
while (! (QUADSPI-> SR & 0x02)); / * Дождаться установки флага TCF * /
```

QUADSPI периферийная конфигурация

Настройте периферийное устройство QUADSPI (регистр QUADSPI_CCR):

Как только память QSPI настроена, интерфейс QUADSPI должен быть настроен в режиме отображения памяти; формат кадра устанавливается в регистр QUADSPI_CCR, как описано ниже:

- Используйте команду DTR QUAD INPUT / OUTPUT FAST READ: INSTRUCTION [7: 0] = 0xED
- Отправить команду в одну строку: IMODE = 0b01
- Отправить адрес в четыре строки: ADMODE = 0b11
- Настройте адрес 3 байта: ADSIZE = 0b10
- Отправить альтернативный байт в четыре строки: ABMODE = 0b11
- Настройте 1 альтернативный байт: ABSIZE = 0b00 • Получите данные в 4 строки: DMODE = 0b11
- Настройте 7 фиктивных циклов: DCYC = 0b00111
- Включить режим отображения памяти: FMODE = 0b11
- Включить режим SIOO: SIOO = 1
- Включить режим DDR: DDRM = 1

Поскольку включен режим SIOO (в спецификации MICRON он называется XIP), необходимо отправлять альтернативный байт (QUADSPI_ABR = 0x00) при каждой новой последовательности чтения, чтобы сохранить память QSPI в режиме SIOO.

Согласно данным MICRON, при использовании команды 0xED в режиме Extended-SPI должна быть установлена задержка в восемь фиктивных циклов перед получением данных. Это зависит также от QSPI_CLK.

Для периферийного устройства QUADSPI сконфигурировано только семь фиктивных циклов, однако на стороне памяти QSPI это восемь циклов; это связано с дополнительным циклом альтернативных байтов, который необходим для отправки одного байта в режиме DDR.

После фазы адреса в общей сложности задержка составляет восемь циклов перед фазой данных: семь фиктивных циклов + один цикл альтернативных байтов.

Смотрите ниже код конфигурации:

```
QUADSPI -> CCR = (0x0F002C00 | QSPI_CCR_DDRM | QSPI_CCR_DCYC | FAST_READ_CMD | QSPI_CCR_IMODE | QSPI_CCR_SIOO | QSPI_CCR_ABMODE);
```

Настройте периферийное устройство QUADSPI (регистр QUADSPI_CR)

Пользователь может выбрать работу в режиме DDR или в режиме SDR, в зависимости от конфигурации проекта. Конфигурацией по умолчанию является QSPI_DDRMODE, определенный в обеих конфигурациях. Поскольку режим DDR включен, отложенное смещение выборки должно быть отключено.

В следующем коде описывается, как включить или отключить режим DDR, а также как настроить прескалер и объем памяти QSPI:

```
#ifdef QSPI_DDRMODE
QUADSPI-> CR | = QSPI_CLK_PRESCALER;
 / * SSHIFT = 0 смещение отсрочки с задержкой отключено в режиме DDR * /
#else
QUADSPI-> CR | = QSPI_CLK_PRESCALER | 0x10; / * 0x10: SSHIFT = 1 * /
#endif
QUADSPI-> DCR = 0x00190000;
/ * Объем памяти: 512 МБ (64 МБ): 2 ^ (26-1) -> 2 ^ (25) -> 2 ^ (0x19) * /
#endif
```

5.2.2 Размещение кода приложения во внешней памяти QSPI

Код, который должен быть загружен в память QSPI, состоит из алгоритмов вычисления, используемых для получения максимальной энергии бина в частотной области входного сигнала с использованием комплексного БПФ, комплексной величины и функций максимума.

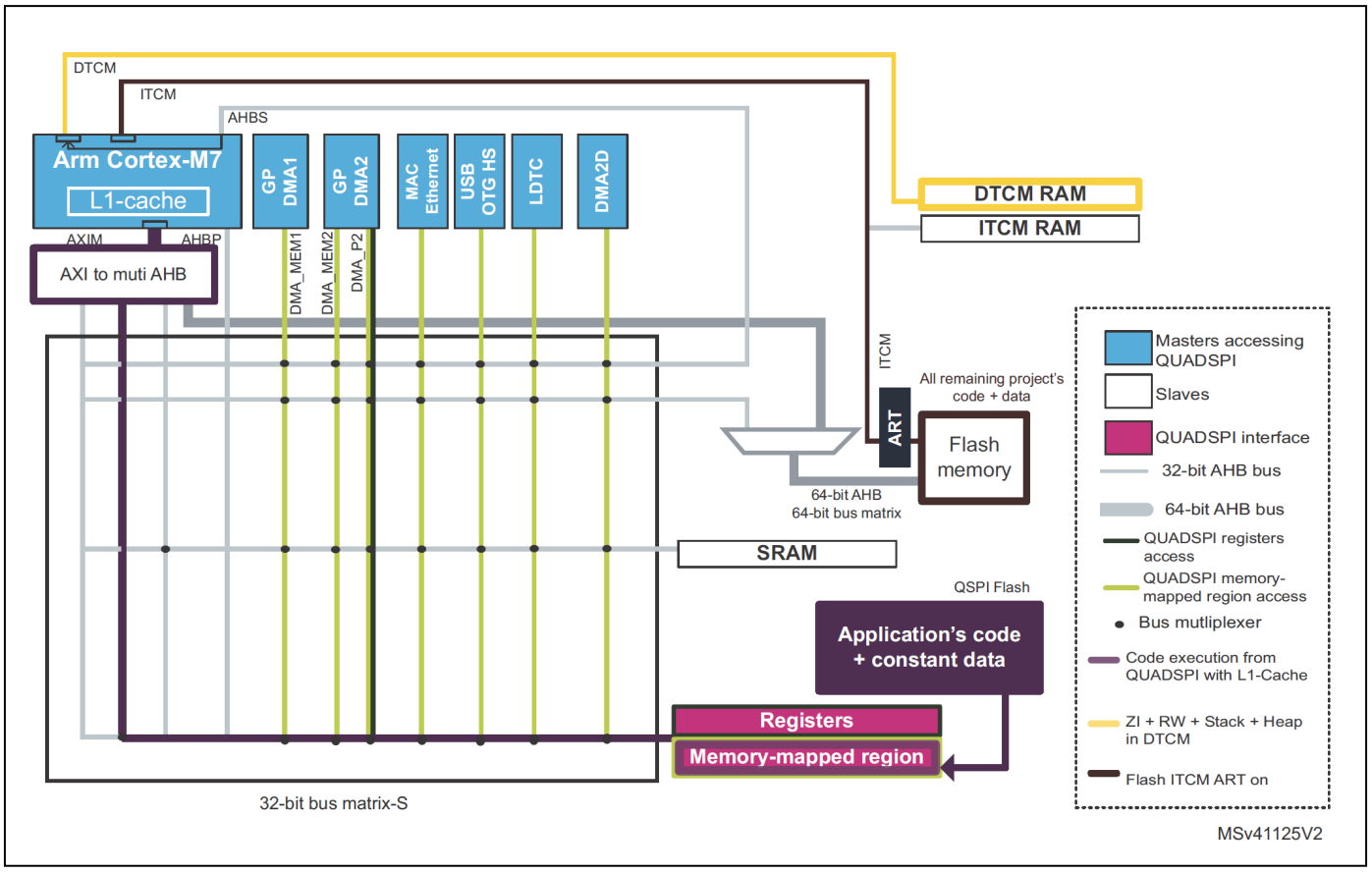
Чтобы поместить этот код во внешнюю память QSPI, в файле компоновщика проекта должна быть создана выделенная область загрузки. В проекте доступны две конфигурации проекта: одна, где код приложения и постоянные данные помещаются в память QSPI, а другая - когда код приложения помещается в память QSPI, где постоянные данные помещаются в память. ITCM Flash. Для обеих конфигураций проекта включена L1-DCache.

Код и постоянные данные помещаются в память QSPI: 6_1-Quad-SPI_rwRAM-DTCM

В этой конфигурации проекта код приложения и связанные с ним постоянные данные помещаются в память QSPI; поэтому Cortex®-M7 должен извлекать их из внешней памяти.

Все оставшиеся коды проектов в качестве периферийных драйверов и векторных таблиц помещаются во флэш-память ITCM. На следующем рисунке описана конфигурация проекта 6_1-Quad-SPI_rwRAM-DTCM.

Figure 50. 6_1-Quad-SPI_rwRAM-DTCM project configuration: code and data in QSPI memory



Чтобы поместить код и постоянные данные в память QSPI, необходимо создать выделенную область загрузки, как показано в следующем файле разброса Keil MDK-ARM:

```
; ********************************************** **********
; *** Scatter-Загрузка файла описания, созданного uVision ***
; ********************************************** **********
LR_IROM1 0x00200000 0x00100000 {; область загрузки size_region ER_IROM1 0x00200000 0x00100000 {; адрес загрузки = адрес выполнения
* .o (СБРОС, + Первый)
* (InRoot $$ разделы)
; Поместите все оставшиеся данные кода и const во Flash TCM.
.ANY (+ RO)
  }
}
LR_IROM2 0x90000000 0x00100000 {; область загрузки size_region
ER_IROM2 0x90000000 0x00100000 {; адрес загрузки = адрес выполнения
arm_fft_bin_example_f32.o (+ RO-CODE)
arm_bitreversal2.o (+ RO-CODE)
arm_cfft_f32.o (+ RO-CODE)
arm_cfft_radix8_f32.o (+ RO-CODE)
arm_cmplx_mag_f32.o (+ RO-CODE)
arm_max_f32.o (+ RO-CODE)
arm_fft_bin_example_f32.o (+ RO-DATA)
```

```
 arm_common_tables.o (+ RO-DATA)
arm_const_structs.o (+ RO-DATA)
 }
RAM_RW_ZI 0x20000000 0x4000 {
.ANY (+ RW + ZI)
}
RAM_STACK 0x20004000 0x4000 {
.ANY (STACK)} RAM_HEAP 0x20008000 0x8000 {
.ANY (HEAP)
}
}
```

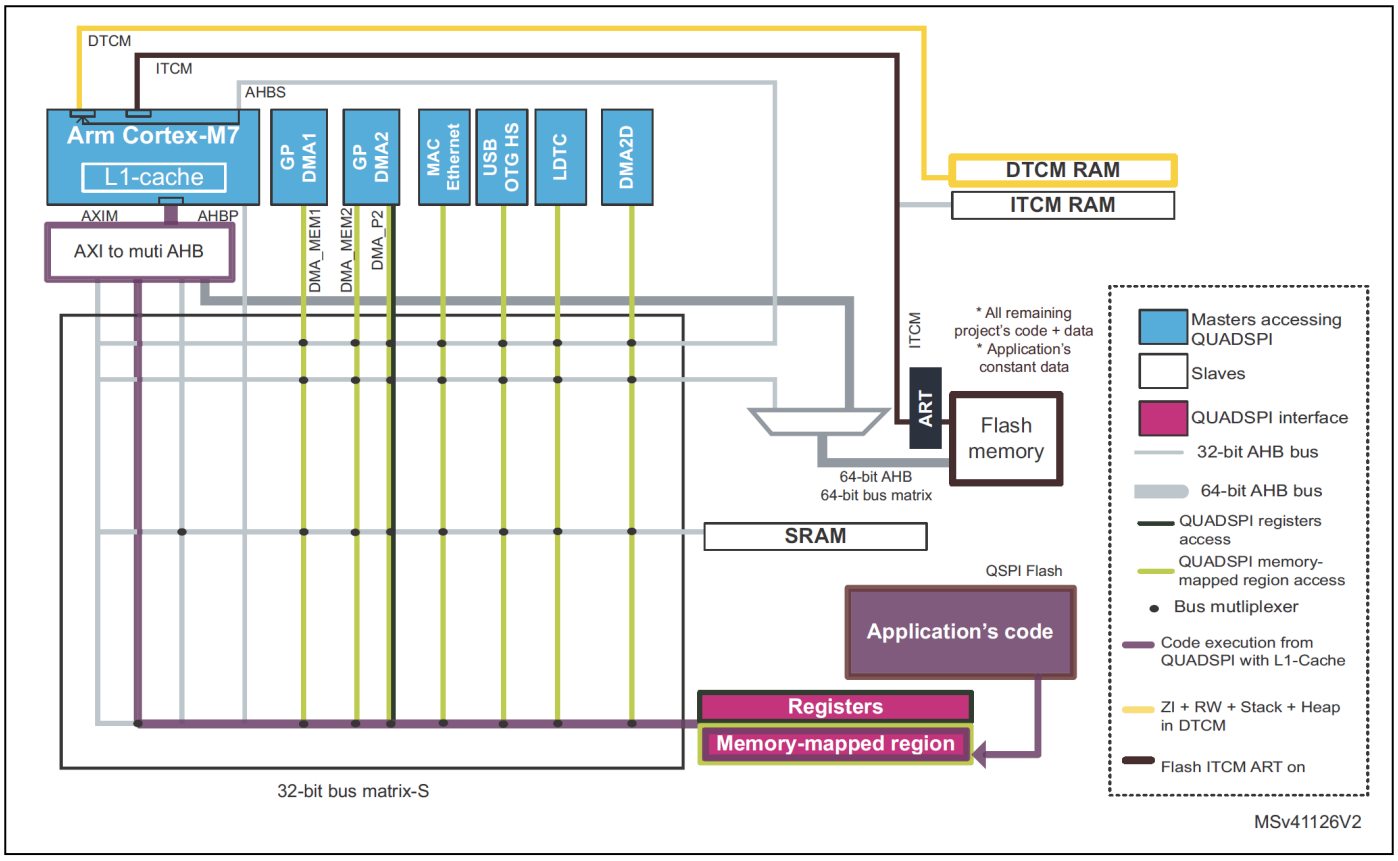
Код, помещенный в память QSPI, а постоянные данные во флэш-память ITCM:

6_2-Quad-SPI_rwRAM-DTCM

В этой конфигурации проекта код приложения помещается в память QSPI, а связанные с ним постоянные данные помещаются во Flash ITCM. Cortex®-M7 должен извлекать код из памяти QSPI и данные из Flash ITCM.

Все оставшиеся коды проектов в качестве периферийных драйверов и векторных таблиц помещаются во флэш-память ITCM. На рисунке ниже показана конфигурация проекта 6_2-Quad-SPI_rwRAM-DTCM.

Figure 51. 6_2-Quad-SPI_rwRAM-DTCM project configuration: only code in QSPI memory



Чтобы поместить код и постоянные данные в память QSPI, необходимо создать выделенную область загрузки, как показано в следующем файле разброса Keil MDK-ARM:

```
; ********************************************** **********
; *** Scatter-Загрузка файла описания, созданного uVision ***
; ********************************************** **********
LR_IROM1 0x00200000 0x00100000 {; область загрузки size_region
ER_IROM1 0x00200000 0x00100000 {; адрес загрузки = адрес выполнения
* .o (СБРОС, + Первый)
* (InRoot $$ разделы)
 ; Поместите все оставшиеся данные кода и константы во Flash TCM.
.ANY (+ RO)
}
}
LR_IROM2 0x90000000 0x00100000 {; область загрузки size_region
ER_IROM2 0x90000000 0x00100000 {; адрес загрузки = адрес исполнения arm_
fft_bin_example_f32.o (+ RO-CODE)
arm_bitreversal2.o (+ RO-CODE)
arm_cfft_f32.o (+ RO-CODE)
arm_cfft_radix8_f32.o (+ RO-CODE)
arm_cmplx_mag_f32.o (+ RO-CODE)
arm_max_f32.o (+ RO-CODE)
 }
RAM_RW_ZI 0x20000000 0x4000 {
.ANY (+ RW + ZI)
}
RAM_STACK 0x20004000 0x4000 {
.ANY (CTEK)
}
RAM_HEAP 0x20008000 0x8000 {
.ANY (HEAP)
}
}
```

Анализ производительности

Результаты получены с STM32756G-EVAL, процессор работает на частоте 216 МГц, VDD = 3,3 В и имеет семь состояний ожидания доступа к внутренней флэш-памяти. QUADSPI настроен в режиме DDR 1-4-4 с включенным SIOO и QSPI_CLK = 54 МГц.

В таблице ниже приведены полученные результаты для демонстрации FFT для MDK-ARM в каждой конфигурации.

Table 15. Execution performances versus configuration

Feature configuration	Memory location configuration	CPU cycle number(1)
-	5-RAMITCM_rwRAM-DTCM	112428
I-cache + D-cache ON (constant data in QSPI memory)	6_1-Quad-SPI_rwRAM-DTCM	171056
I-cache + ART + ART-PF ON (constant data in Flash TCM)	6_2-Quad-SPI_rwRAM-DTCM	126900

1. Количество циклов может меняться от версии к другой цепочке инструментов.

Если сравнить результаты первого случая и второго варианта конфигурации «6-Quad SPI_rwRAM-DTCM», отметим, что существует значительная разница в показателях производительности, поскольку при демонстрации используются огромные постоянные данные.

• Для первого случая (6_1-Quad SPI_rwRAM-DTCM), поскольку данные только для чтения и инструкции находятся во флэш-памяти QSPI, задержка возникает из-за одновременного доступа к выборке команд и данным только для чтения на интерфейсе QUADSPI.

• Для второго случая (6_2-Quad SPI_rwRAM-DTCM) данные и код только для чтения разделяются. Данные только для чтения находятся во Flash-TCM, поэтому параллелизм данных, доступных только для чтения, и выбор команд исключаются, и ЦП может извлекать инструкцию из AXI, в то время как данные загружаются из TCM одновременно. Это причина, почему производительность второго случая явно лучше, чем первый.

Сравнивая случай 6_2-Quad-QPI_rwRAM-DTCM с 5-RAMITCM_rwRAM-DTCM (который дает лучшие характеристики при 112428 циклах ЦП согласно документу AN4667), видно, что они близки по показателям производительности.

Это пример того, насколько важно извлечь выгоду из интеллектуальной архитектуры STM32F7x5 / F7x6 (в этом примере), чтобы улучшить производительность выполнения из внешней памяти QSPI. Для получения более подробной информации о том, как улучшить производительность выполнения из памяти QSPI, обратитесь к Разделу 6.1: Как получить наилучшие результаты.

5.3 Хранение (программирование) данных на лету во время работающего приложения

В этом разделе, основанном на двух примерах микропрограммы STM32Cube, описывается, как программировать флэш-память QSPI на лету во время работы приложения. Программирование может быть выполнено либо с помощью программного обеспечения, путем записи непосредственно в регистр QUADSPI_DR, либо с использованием DMA.

Пакет прошивки STM32Cube предоставляет два примера чтения и программирования памяти QSPI:

• QSPI_ReadWrite_DMA: использование DMA

• QSPI_ReadWrite_IT: использование прерываний.

Эти примеры приведены для плат STM32L476G-EVAL, STM32446E-EVAL, STM32469I-EVAL и STM32756G-EVAL и могут быть легко адаптированы к платам Discovery. В этом разделе описываются примеры программирования STM32756G-EVAL.

Примечание:

Так как флэш-воспоминания должны быть удалены перед записью; пользователь должен выполнить операцию стирания перед программированием флэш-памяти. Только тот регион, который нужно запрограммировать, должен быть удален; нет необходимости выполнять операцию стирания массовой микросхемы, если нужно запрограммировать только один сектор.

5.3.1 QUADSPI indirect write (непрямая запись QUADSPI): программирование памяти QSPI с использованием DMA

В этом разделе описывается пример STM32756G-EVAL, который доступен в STM32Cube_FW_F7 в каталоге «Projects \ STM32756G_EVAL \ examples \ QSPI \ QSPI_ReadWrite_DMA».

В этом примере показано, как запрограммировать память QSPI с данными из внутренней SRAM. DMA2 используется для передачи данных из внутренней SRAM в интерфейс QUADSPI. Данные для записи «aTxBuffer» - это буфер, сгенерированный в SRAM. Содержит строку

**** QSPI Communication на основе DMA ****.

В этом примере следующая последовательность выполняется в цикле forever:

1. Сектор 1 памяти QSPI стирается
2. Данные записываются в память в режиме DMA
3. Данные читаются в режиме DMA для сравнения с источником

–LED1 переключается каждый раз, когда новое сравнение хорошо

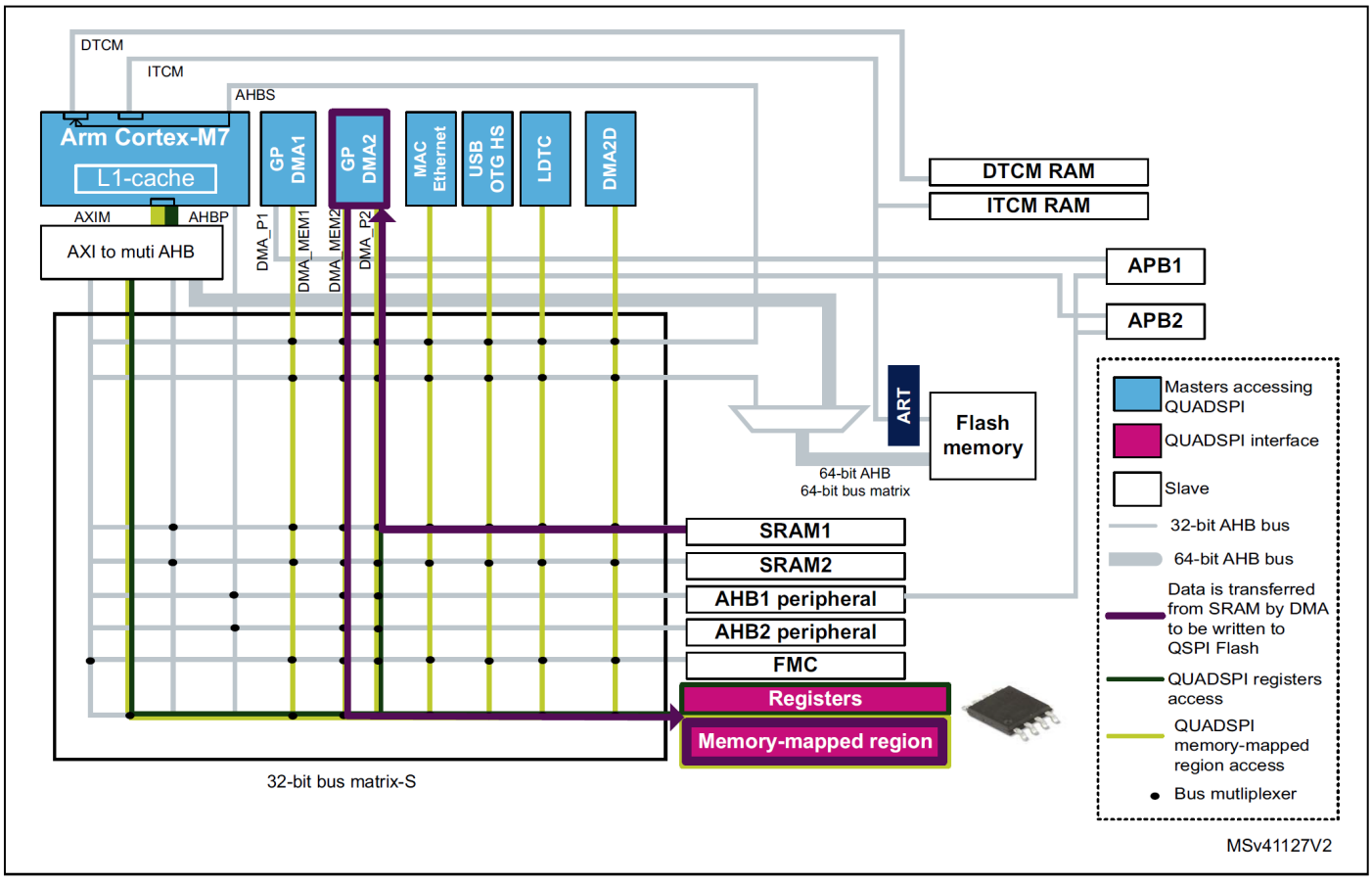
–LED3 включается, как только возникает ошибка сравнения

–LED3 переключается, как только HAL API возвращает ошибку.

Этот раздел посвящен записи в память QSPI. Благодаря интеллектуальной архитектуре STM32F7x5 / F7x6 DMA можно использовать для программирования или чтения флэш-памяти QSPI, а затем для разгрузки процессора. После настройки прямого доступа к памяти и начала передачи вмешательство ЦП не требуется. Прерывание может быть сгенерировано после завершения передачи.

Как описано на рисунке ниже, DMA считывает данные «aTxBuffer» из SRAM и записывает их в память QSPI, в то время как CPU может выполнять код из внутренней Flash-памяти.

Figure 52. Indirect write mode: programming QSPI memory using DMA



Конфигурация QUADSPI GPIO и DMA

GPIO и DMA2 настраиваются в функции HAL_QSPI_MspInit () в файле stm32f7xx_hal_msp.c. Функция HAL_QSPI_MspInit () вызывается в функции HAL_QSPI_Init (), которая включает в себя все требуемые конфигурации QUADSPI.

Глобальное прерывание QUADSPI включено. Конфигурация GPIO выполняется с учетом подключения памяти QSPI на плате STM32756G-EVAL.

DMA2 настроен следующим образом:

- Конфигурация DMA2: поток 7, канал 3 включен
- Увеличение памяти включено
- Прерывание DMA2 включено.

QUADSPI периферийная конфигурация

Периферийное устройство QUADSPI настраивается в функции HAL_QSPI_Init (), как описано ниже:

- Тактовый прескалер = 2 => QSPI_CLK = (HCLK / 3) = 72 МГц. • Порог FIFO FTHRES = 0x03.
- Задержка смещения образца отключена.
- Размер флэш-памяти устанавливается в регистре QUADSPI_DCR: объем памяти составляет 64 МБ = 2 [FSIZE + 1] = 2 [19 + 1], поэтому FSIZE = 0x19.
- Высокое время выбора микросхемы (CSHT) установлено на один цикл.
- Периферийное устройство QUADSPI активируется путем установки бита EN в регистре QUADSPI_CR.

Конфигурация памяти QSPI

Как упоминалось ранее, команда write-enable должна быть отправлена сначала. Это делается путем вызова функции QSPI_WriteEnable ().

Запустить последовательность программирования

Для запуска последовательности программирования регистр QUADSPI_CCR настраивается в функции HAL_QSPI_Command (), как описано ниже:

- Инструкция: QUAD_IN_FAST_PROG_CMD (0x32)
- IMODE: одна строка
- ADMODE: одна строка
- ADSIZE: 24 бита
- DMODE: четыре строки
- FMODE: непрямой режим
- Количество записываемых байтов задается в регистре QUADSPI_DLR в функции HAL_QSPI_Command (). Количество записываемых байтов = QUADSPI_DLR + 1.

Последовательность программирования запускается в функции HAL_QSPI_Transmit_DMA (), поэтому в этой функции выполняются следующие конфигурации:

- Настройка направления DMA: DMA_MEMORY_TO_PERIPH
- Количество байтов, которые должны быть переданы в регистр S7NDTR (S7NDTR = QUADSPI_DLR + 1)
- Передача DMA включается установкой бита DMAEN в регистре QUADSPI_CR.

Адрес, используемый в регистре QUADSPI_AR, равен 0x00000000, поскольку запись выполняется в первом секторе. Таким образом, последовательность программирования запускается сразу после установки бита DMAEN.

Поскольку передача DMA в FIFO происходит быстрее, чем по шине QUADSPI, QUADSPI контролирует поток передачи, устанавливая пороговый флаг FIFO (FTF) каждый раз, когда доступно (FTHRESH + 1) свободных байтов, доступных для записи в FIFO. Передачи DMA инициируются, только если установлен флаг FTF.

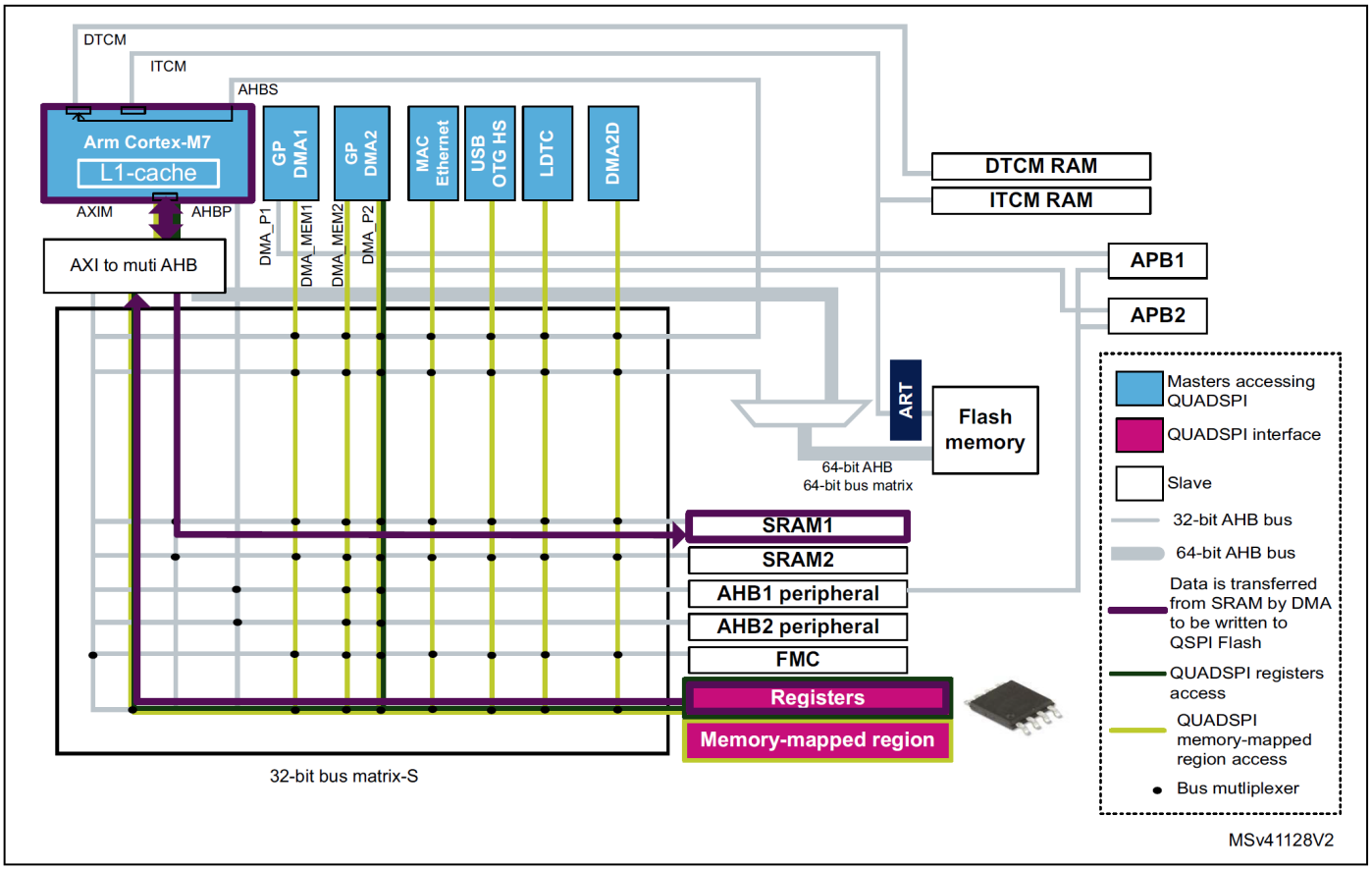
5.3.2 QUADSPI indirect write (непрямая запись QUADSPI): программирование памяти QSPI с использованием прерываний

Этот пример доступен для всех STM32, встраивающих интерфейс QUADSPI. Соответствующие продукты перечислены в Таблице 2: доступность Quad-SPI и функции для семейств STM32.

В этом разделе описывается пример STM32756G_EVAL, который доступен на STM32Cube_FW_F7 в каталоге Projects \ STM32756G_EVAL \ examples \ QSPI \ QSPI_ReadWrite_IT.

В этом примере показано, как запрограммировать память QSPI с данными из внутренней SRAM, используя ЦП и прерывания. Данные для записи (aTxBuffer) представляют собой буфер, сгенерированный в SRAM.

Figure 53. Indirect write mode: programming QSPI memory using interrupt



Конфигурация QUADSPI GPIO

GPIO и DMA2 настраиваются в функции HAL_QSPI_MspInit () в файле stm32f7xx_hal_msp.c. Обратите внимание, что функция HAL_QSPI_MspInit () вызывается в функции HAL_QSPI_Init (), которая включает все необходимые конфигурации Quad-SPI.

Глобальное прерывание QUADSPI включено. Конфигурация GPIO выполняется с учетом подключения памяти QSPI на плате STM32756G-EVAL.

Периферийная конфигурация QUADSPI

Периферийное устройство QUADSPI настраивается в функции HAL_QSPI_ Init (), как описано ниже:

• Настройте прескалер часов = 2 => QSPI_CLK = (HCLK / 3) = 72 МГц.

• Порог FIFO FTHRES = 0x03 => QUADSPI прерывает процессор для записи в FIFO. Каждый раз, когда устанавливается флаг FTF (каждый раз, когда есть (FTHRESH + 1), свободные байты доступны для записи в FIFO).

• Задержка смещения образца отключена.

• Размер флэш-памяти устанавливается в регистре QUADSPI_DCR: объем памяти составляет 64 МБ = 2 [FSIZE + 1] = 2 [19 + 1], поэтому FSIZE = 0x19.

• Высокое время выбора микросхемы (CSHT) установлено на один цикл.

• Периферийное устройство QUADSPI активируется путем установки бита EN в регистре QUADSPI_CR.

Конфигурация памяти QSPI

Как упоминалось ранее, команда write-enable должна быть отправлена сначала. Это делается путем вызова функции QSPI_WriteEnable ().

Запустить последовательность программирования

Для запуска последовательности программирования регистр QUADSPI_CCR настраивается в функции HAL_QSPI_Command (), как описано ниже:

• Инструкция: QUAD_IN_FAST_PROG_CMD (0x32)

• IMODE: одна строка • ADMODE: одна строка

• ADSIZE: 24 бита

• DMODE: четыре строки

• FMODE: непрямой режим

• Количество записываемых байтов задается в регистре QUADSPI_DLR в функции HAL_QSPI_Command ().

Последовательность программирования запускается в функции HAL_QSPI_ Transmit_IT (), поэтому в этой функции выполняются следующие конфигурации:

• Включите ошибку передачи TEAD QUADSPI, порог FTIO FIFO и завершите прерывания TCIE.

Обратите внимание, что в регистре QUADSPI_AR используется адрес 0x00000000, так как запись выполняется в первом секторе.

Как только этот флаг FTF установлен, в CPU генерируется прерывание. Процессор переходит из основного кода в подпрограмму прерывания HAL_QSPI_ IRQHandler (), расположенную в stm32f7xx_hal_qspi.c, и проверяет источник прерывания, затем процессор начинает передачу данных в FIFO. По окончании CPU очищает флаг FTF, выходит из HAL_QSPI_IRQHandler () и возвращается к основному коду.

5.4 Пример стирания данных

В этом разделе описывается, как стереть флэш-память QSPI. Как упоминалось ранее, для стирания флэш-памяти QSPI следует использовать косвенный режим.

Все предоставленные примеры QSPI в пакете прошивки STM32Cube включают пример операции стирания. Для платы EVAL продуктов STM32F7x5 / F7x6 примеры доступны в следующем каталоге STM32Cube: STM32Cube_FW_F7_VX.X.X \ Projects \ STM32756G_EVAL \ examples \ QSPI.

Конфигурация памяти QSPI

Перед выполнением операции стирания в память должна быть отправлена команда включения записи, это делается с помощью функции QSPI_WriteEnable ().

Старт последовательности стирания

Чтобы начать стирание последовательности, регистр QUADSPI_CCR настраивается в функции HAL_QSPI_Command_IT (), как описано ниже:

- Инструкция: SECTOR_ERASE_CMD (0xD8)
- IMODE: одна строка
- ADMODE: одна строка
- ADSIZE: 24 бита
- DMODE: нет данных
- FMODE: режим косвенной записи
- QUADSPI_AR = 0x0000 0000.

После конфигурирования регистра QUADSPI_CCR, и поскольку данные не нужно отправлять, последовательность стирания запускается сразу же после предоставления адреса первого сектора.

5.5 Пример аппаратной реализации

В этом разделе приводятся некоторые примеры реализации аппаратного обеспечения подключения флэш-памяти QSPI к STM32 на основе существующих плат обнаружения и оценки ST. В следующей таблице приведены различные платы STM32 с встроенной флэш-памятью QSPI.

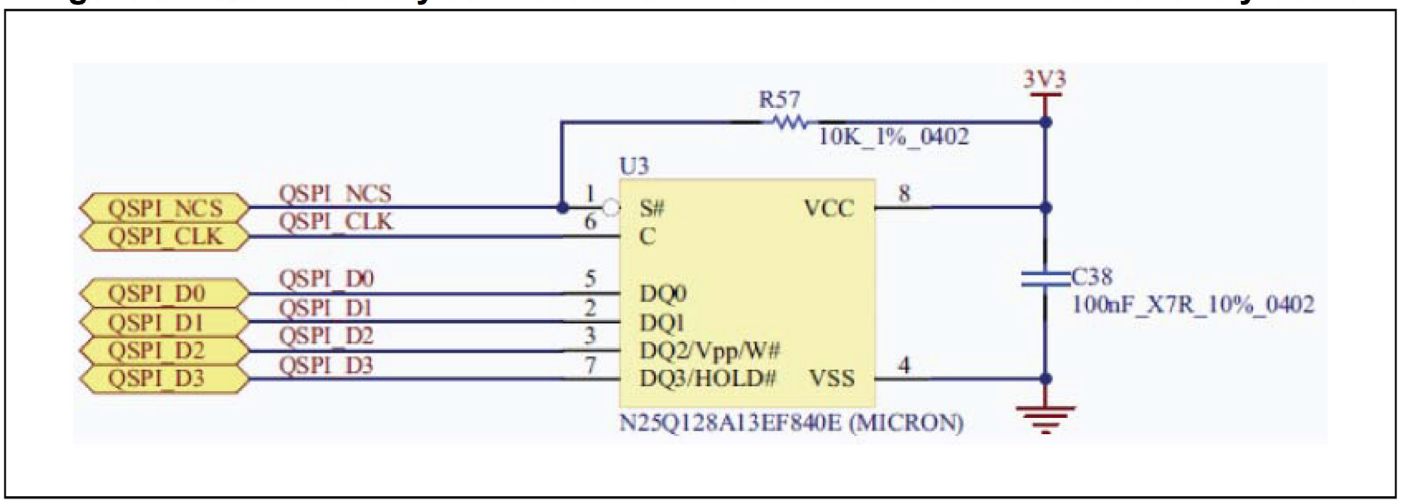
Table 16. Different STM32 boards embedding QSPI Flash memory

Product families	Board	QSPI Flash model	Size (Mbytes)
STM32L475	B-L475E-IOT01A	MX25R6435F	8
STM32L476	STM32L476G-DISCO	N25Q128A13EF840E	16
	STM32L476G-EVAL	N25Q256A13EF840E	32
STM32L496	STM32L496G-DISCO	MX25R6435FM2IL0	8
STM32F412	STM32F412G-DISCO	N25Q128A13EF840F	16
STM32F413	STM32F413H-DISCO	-	16
STM32F446	STM32446E-EVAL	N25Q256A13EF840E	32
STM32F469	STM32F469I-DISCO	N25Q128A13EF840F	16
	STM32469I-EVAL	MT25QL512ABA8ESF-0SIT	64
STM32F479	STM32479I-EVAL	MT25QL512ABA8ESF-0SIT	64
STM32F723	STM32F723e-DISCO	MX25L51245G	64
STM32F746	STM32F746G-DISCO	N25Q128A13EF840E	16
	STM32746G-EVAL	MT25QL512ABA8ESF-0SIT	64
STM32F750	STM32F7508-DISCO	-	16
STM32F756	STM32756G-EVAL	N25Q512A13GSF40E	64
STM32F769	STM32F769I-EVAL	MT25QL512ABA8ESF-0SIT	64
	STM32F769I-DISCO	MT25QL512ABB1EW9 /MX25L51245G	64
STM32F779	STM32F779I-EVAL	N25Q512A13GSF40E	64
STM32H743/STM32H753	STM32H7xxI-EVAL	MT25TL 01GHBB8ESF-0SIT	128

STM32F746G-DISCO отладочная плата

На рисунке ниже показан пример подключения флэш-памяти MICRON QSPI в режиме Quad I / O на плате обнаружения STM32F746G-DISCO.

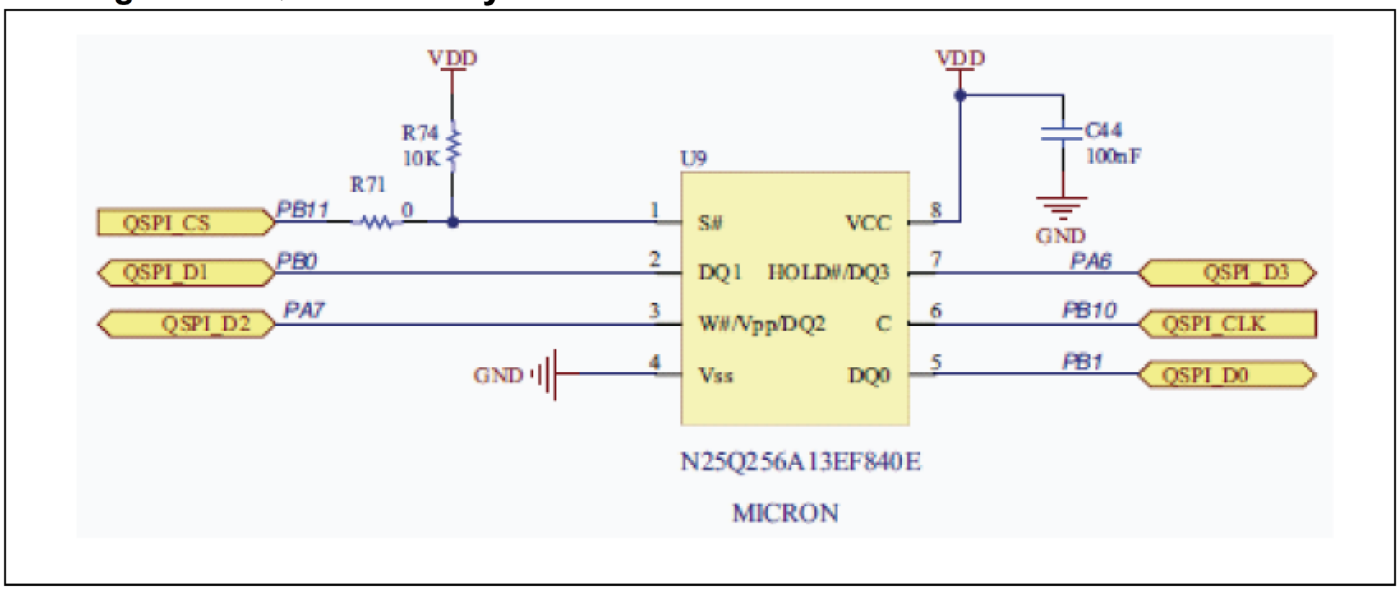
Figure 54. QSPI memory connection on the STM32F746G-DISCO discovery board



STM32L476G-EVAL плата

На рисунке ниже показан пример подключения флэш-памяти MICRON QSPI в режиме Quad I / O на плате обнаружения STM32L476G-EVAL.

Figure 55. QSPI memory connection on the STM32L476G-EVAL board



6 Производительность и мощность

В этом разделе представлены некоторые рекомендации по повышению производительности и снижению энергопотребления для приложений, использующих интерфейс QUADSPI.

6.1 Как получить лучшие выступления

6.1.1 Запись производительности

Поскольку скорость записи флэш-памяти QSPI значительно ниже, при оптимизации скорости передачи имеется ограниченное преимущество. Полезно использовать пакетную запись (программирование страниц), чтобы уменьшить накладные расходы команд. Кроме того, DMA может использоваться для разгрузки процессора.

6.1.2 Производительность чтения

В этом разделе описывается, как получить оптимальные характеристики чтения с использованием периферийных функций QUADSPI, следуя описанным рекомендациям.

Настройте QUADSPI на максимальной скорости

Одним из наиболее важных параметров, позволяющих повысить производительность чтения, является тактовая частота QUADSPI, которая должна быть как можно выше. Как упоминалось ранее (см. Раздел 3.3.2: Хорошая конструкция печатной платы допускает максимальную скорость QUADSPI), максимально достижимая рабочая скорость QUADSPI зависит главным образом от качества конструкции печатной платы, поэтому пользователь должен оптимизировать конструкцию печатной платы.

Например, если пользовательское оборудование позволяет QUADSPI работать на частоте 60 МГц в режиме DDR QuadI / O, то в случае последовательного доступа изображение может быть прочитано со скоростью 60000000/1024/1024 = 57,22 Мбайт / с. В этом случае для изображения размером 4 КБ общее время передачи составляет 4 107 циклов: восемь циклов для команды + три цикла для адреса (24 бита в режиме DDR) + 4096 циклов для изображения 4 КБ.

Другим важным параметром, который должен учитывать пользователь при выборе устройства памяти QSPI, является максимальная тактовая частота, поддерживаемая памятью QSPI в режимах SDR и DDR.

Использовать режим DDR (DTR)

Используйте режим DDR для удвоения пропускной способности, например, в режиме SDRQuadI / O один байт считывается каждые два такта, в то время как в режиме DDR один байт считывается каждый такт.

Другое интересное использование режима DDR заключается в том, что он может быть очень хорошей альтернативой в приложениях с низким энергопотреблением, требующих работы на низких системных часах для экономии энергии, когда невозможно, чтобы QUADSPI работал на своей максимальной скорости. В этом случае пользователь может использовать режим DDR для удвоения скорости чтения, но с одинаковыми рабочими часами QUADSPI.

Примечание:

Пользователь должен убедиться, что используемая команда чтения поддерживает режим DDR

Используйте режим Quad I / O

Использование режима Quad I / O позволяет повысить производительность чтения, поэтому пользователь должен использовать режим Quad I / O, а не режим Single или Dual I / O. Рекомендуется использовать режим QuadI / O для всех фаз: команды, адреса, альтернативного байта и данных.

Обратите внимание, что отправка команды в четыре строки поддерживается некоторыми устройствами памяти, такими как Micron или Spansion.

Используйте режим Dual-Flash

Использование режима Dual-Flash требует добавления только четырех дополнительных GPIO (IO4 ... IO7) и позволяет удвоить пропускную способность. Например, в режиме двойной флэш-памяти SDR Quad I / O один цикл считывается в каждом цикле QSPI_CLK, в то время как в режиме двойной флэш-памяти SDR Quad I / O два байта считывается в каждом цикле QSPI_CLK.

Уменьшить командные накладные расходы

Каждый доступ к памяти QSPI требует команды и адреса для отправки, что приводит к накладным расходам команды. Чтобы уменьшить накладные расходы команд и повысить производительность чтения, пользователь должен использовать следующие рекомендации:

• Используйте большие пакетные передачи для косвенного режима

Поскольку каждый доступ к памяти QSPI должен отправлять команду и адрес, выгодно выполнять большие пакетные передачи, а не небольшие повторные передачи; это действие позволяет уменьшить командные издержки.

• Последовательный доступ в режиме отображения памяти

Наилучшая производительность чтения достигается при последовательном считывании сохраненных данных, что позволяет избежать накладных расходов на команды и адреса, а затем приводит к достижению максимальных характеристик при рабочей тактовой частоте QUADSPI.

Как правило, это тот случай, когда память QSPI используется для приложений хранения, таких как графическое или мультимедийное содержимое (см. Примеры в Разделе 5.1: Чтение данных из памяти QSPI: графическое приложение.)

• Рассмотрим счетчик времени ожидания

Пользователь должен учитывать, что включение счетчика тайм-аута в режиме отображения памяти может увеличить накладные расходы команды. Когда происходит таймаут, QUADSPI повышается чип-выбор. После этого для чтения из памяти QSPI должна быть инициирована новая последовательность чтения; это означает, что команда чтения должна быть выполнена снова, что приводит к накладным расходам команды (см. Раздел 2.4.3: Счетчик времени ожидания).

Счетчик тайм-аута позволяет снизить энергопотребление (см. Раздел 6.2.2: Использование счетчика тайм-аута), но если производительность вызывает беспокойство, пользователь может увеличить период тайм-аута в регистре QUADSPI_LPTR или даже отключить его.

Если потребление энергии также вызывает беспокойство, пользователь может включить функцию SIOO без необходимости отключения счетчика тайм-аута.

• Используйте функцию SIOO (режим непрерывного чтения) для случайного и непоследовательного доступа

При случайном и непоследовательном доступе командные издержки увеличиваются. Как описано на рисунке 13, команда и адрес отправляются в память при каждой новой последовательности чтения. В этом случае пользователь должен включить функцию SIOO, чтобы уменьшить накладные расходы на команды (см. Раздел 2.4.1: Отправка команды только один раз (SIOO)).

Примечание:

Не все команды чтения поддерживают режим непрерывного чтения (режим повышенной производительности), поэтому пользователь должен учитывать эту информацию при выборе команды чтения.

Повышение производительности

Чтобы повысить производительность выполнения из флэш-памяти QSPI, пользователь должен следовать ранее описанным рекомендациям по производительности чтения.

Выполнение из памяти QSPI обычно характеризуется ее случайным и непоследовательным доступом. Как уже упоминалось, важной рекомендацией для повышения производительности является включение функции SIOO.

Как видно на рисунке 50: конфигурация проекта 6_1-Quad-SPI_rwRAM-DTCM: код и данные в памяти QSPI, размещение как кода, так и данных только для чтения в памяти QSPI приводит к параллелизму интерфейса QUADSPI во время выполнения.

Чтобы избежать этого параллелизма, пользователь может разделить данные только для чтения и код. Например, данные только для чтения могут быть расположены в памяти QSPI, а код - во Flash-TCM. Это действие позволяет избежать параллелизма данных только для чтения и выборки команд; следовательно, CPU может получать инструкции из Flash-TCM, в то время как данные загружаются из памяти QSPI одновременно.

Когда приложение содержит огромные данные констант, пользователь может разделить константы и код, каждый в отдельном разделе. Если размер раздела кода соответствует размеру внутренней флэш-памяти, код может быть загружен во внутреннюю флэш-память, в то время как константы загружаются в память QSPI.

Для STM32F7x5 / F7x6 рекомендуется включить Cortex®-M7 L1-Cache.

Общие рекомендации

- Используйте DMA для передачи данных, чтобы разгрузить процессор.
- Используйте режим опроса статуса флага, а не проверку программного флага.
- Используйте режим отображения памяти, чтобы разрешить любому мастеру AHB доступ к памяти QSPI без вмешательства ЦП.

6.2 Снижение энергопотребления

Одним из наиболее важных требований в носимых и мобильных приложениях является энергопотребление. Чтобы снизить энергопотребление, можно следовать некоторым рекомендациям.

Чтобы уменьшить общее энергопотребление приложения, пользователь обычно переводит STM32 в режим пониженного энергопотребления. Чтобы еще больше снизить потребление тока, подключенную память QSPI также можно перевести в режим пониженного энергопотребления (также известный как режим глубокого отключения).

Для большинства устройств флэш-памяти QSPI режимом по умолчанию после последовательности включения является режим ожидания. В режиме ожидания текущая операция не выполняется; nCS высока, но потребление тока может быть уменьшено еще больше. Режим DPD позволяет снизить энергопотребление.

6.2.1 Использование счетчика тайм-аута

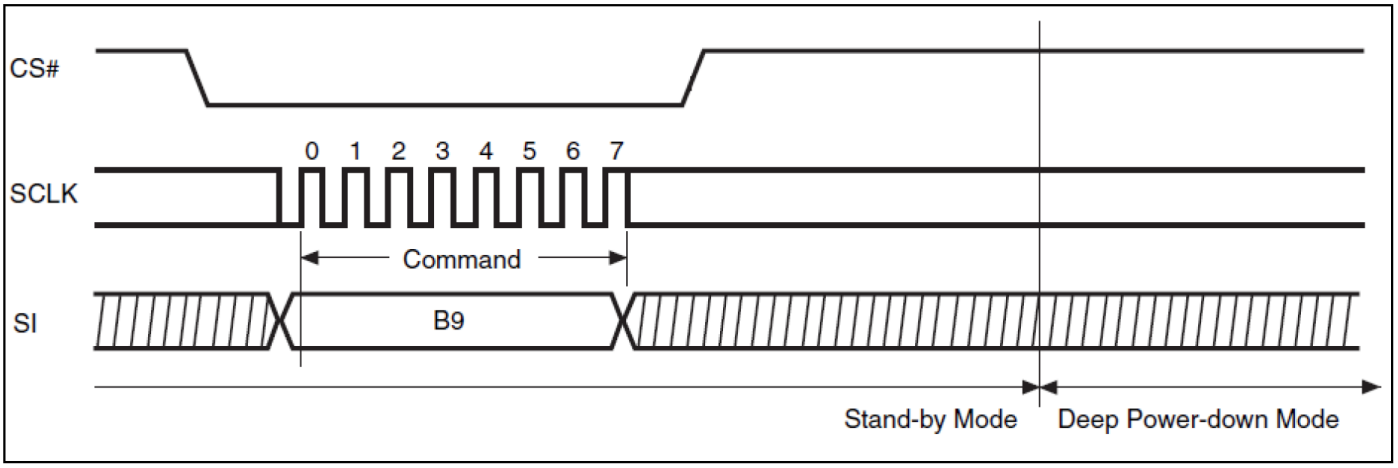
Функция тайм-аута nCS может использоваться, чтобы избежать любого дополнительного энергопотребления во внешней флэш-памяти. Когда часы останавливаются на длительное время, счетчик тайм-аута может освободить вывод nCS, чтобы перевести внешнюю флэш-память в состояние с более низким потреблением после истечения периода тайм-аута без какого-либо доступа (см. Раздел 2.4.4: Счетчик тайм-аута).

6.2.2 Перевести память QSPI в режим глубокого отключения

Режим глубокого выключения требует ввода инструкции глубокого выключения. В режиме глубокого выключения устройство не активно, и все инструкции по записи, программированию и стиранию игнорируются. Когда уровень CS # становится высоким, он находится только в режиме ожидания, а не в режиме глубокого отключения питания. Режим глубокого отключения отличается от режима ожидания.

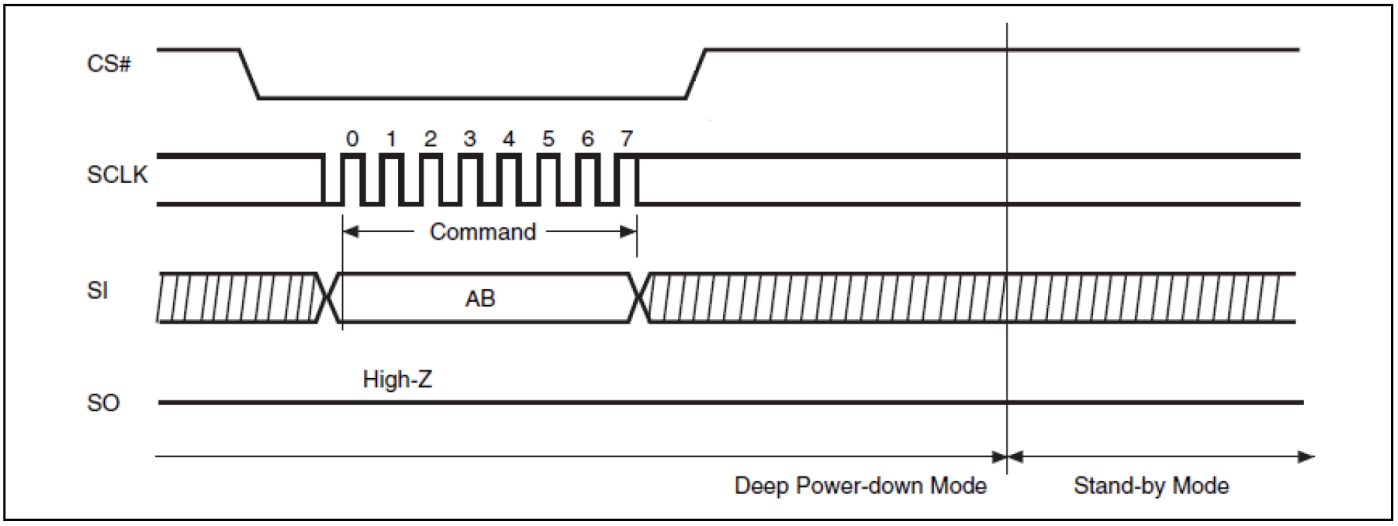
Перед входом в режим пониженного энергопотребления или когда память QSPI не используется, ее можно перевести в режим DPD, чтобы уменьшить общее энергопотребление приложения. Для нескольких марок памяти команда 0xB9 должна быть отправлена на внешнюю последовательную флэш-память для перехода в режим глубокого отключения питания. На следующем рисунке показана последовательность DPD:

Figure 56. Deep power-down (DPD) sequence (command B9)



Для выхода из режима DPD необходимо отправить команду RELEASE FROM DEEP POWERDOWN (0xAB). На рисунке ниже показана последовательность RPD:

Figure 57. Release from deep power-down (RDP) sequence (command AB)



Примечание:

Не все последовательные флэш-памяти поддерживают режим глубокого отключения питания. Если выбранная внешняя последовательная память не поддерживает режим глубокого отключения питания, STM32 может управлять переключателем внешнего питания через GPIO, чтобы отключить питание внешней флэш-памяти QSPI и отменить ее потребление тока.

6.2.3 Флэш-память QSPI с поддержкой режима DPD

Многие бренды памяти QSPI поддерживают режим DPD, ниже приведен пример:

- WINBOND: W25Q64CV
- Spansion: S25FL032P
- MICRON: MT25QL512AB
- Macronix: MX25L12865F

7 Поддерживаемые устройства

Интерфейс STM32 QUADSPI имеет очень гибкий формат кадра, который позволяет следующее:

- Отправить до пяти этапов: инструкция - адрес - альтернативный байт - фиктивный - данные
- пропустить любой этап
- Отправить каждую фазу в одну, две или четыре строки
- Отправить адрес в один, два, три или четыре байта
- Отправить один, два, три или четыре альтернативных байта
- Отправьте до 31 фиктивного такта.

Кроме того, интерфейс STM32 QUADSPI позволяет отправлять любую команду, поэтому пользователь может запрограммировать нужную команду в регистре QUADSPI_CCR в поле INSTRUCTION [7: 0].

Интерфейс STM32 QUADSPI полностью настраивается с точки зрения формата кадра и аппаратного обеспечения и поддерживает большую часть памяти QSPI на рынке.

Существует несколько поставщиков совместимых с QUADSPI модулей памяти, таких как Winbond, Spansion, Macronix, MICRON (Numonyx), Microchip (SST) и другие.

8 Заключение

Микроконтроллеры STM32 обеспечивают очень гибкий и полезный интерфейс QUADSPI, который подходит для приложений, потребляющих память, при меньших затратах на разработку. QUADSPI позволяет избежать сложности проектирования с использованием внешних параллельных флэш-памяти за счет уменьшения количества выводов и повышения производительности. В этой заметке по применению демонстрируются характеристики и гибкость интерфейса STM32 QUADSPI, что позволяет снизить затраты на разработку и ускорить вывод на рынок.

9 История изменений

02-May-2019 Документ обновлен, чтобы расширить область применения для других семейств продуктов. Все разделы затронуты обновлением

49.2 Описание API драйвера прошивки QSPI

49.2.1 Как использовать этот драйвер

Инициализация

1. В качестве предварительного условия заполните HAL_QSPI_MspInit ():

- Включите интерфейс синхронизации QuadSPI с помощью __HAL_RCC_QSPI_CLK_ENABLE ().

- Сбросить QuadSPI IP с помощью __HAL_RCC_QSPI_FORCE_RESET () и __HAL_RCC_QSPI_RELEASE_RESET ().

- Включите часы для QuadSPI GPIOS с помощью __HAL_RCC_GPIOx_CLK_ENABLE ().

- Настройте эти выводы QuadSPI в альтернативном режиме, используя HAL_GPIO_Init ().

- Если используется режим прерывания, включите и настройте глобальное прерывание QuadSPI с помощью HAL_NVIC_SetPriority () и HAL_NVIC_EnableIRQ ().

- Если используется режим DMA, включите часы для канала DMA QuadSPI с помощью __HAL_RCC_DMAx_CLK_ENABLE (), настройте DMA с помощью HAL_DMA_Init (), ссылка

с помощью дескриптора QuadSPI с помощью __HAL_LINKDMA (), включения и настройки глобального прерывания канала DMA с помощью HAL_NVIC_SetPriority () и

HAL_NVIC_EnableIRQ ().

2. С помощью функции HAL_QSPI_Init () настройте размер вспышки, прескалер часов, порог fifo, режим часов, сдвиг выборки и время CS.

Indirect functional mode (косвенный функциональный режим)

1. Настройте последовательность команд с помощью функций HAL_QSPI_Command () или HAL_QSPI_Command_IT ():

- Фаза инструкции: используемый режим и, если присутствует, код операции инструкции.

- Фаза адреса: используемый режим и, если он указан, размер и значение адреса.

- Фаза альтернативных байтов: используемый режим и, если присутствует, значения размера и альтернативных байтов.

- Фаза фиктивных циклов: количество фиктивных циклов (используемый режим такой же, как и фаза данных).

- Фаза данных: используемый режим и, если присутствует, число байтов.

- Режим двойной скорости передачи данных (DDR): активация (или нет) этого режима и задержка, если активирована.

- Режим отправки инструкции только один раз (SIOO): активация (или нет) этого режима.

2 Если для команды не требуются данные, она отправляется непосредственно в память:

- В режиме опроса вывод функции выполняется после завершения передачи.

- В режиме прерывания HAL_QSPI_CmdCpltCallback () будет вызван после завершения передачи.

3. Для режима косвенной записи используйте HAL_QSPI_Transmit (), HAL_QSPI_Transmit_DMA () или HAL_QSPI_Transmit_IT () после настройки команды:

- В режиме опроса вывод функции выполняется после завершения передачи.
- В режиме прерывания HAL_QSPI_FifoThresholdCallback () будет вызываться при достижении порога fifo, а HAL_QSPI_TxCpltCallback () - после завершения передачи.
- В режиме DMA HAL_QSPI_TxHalfCpltCallback () будет вызываться при половинной передаче, а HAL_QSPI_TxCpltCallback () - по завершении передачи.

4. Для режима косвенного чтения используйте HAL_QSPI_Receive (), HAL_QSPI_Receive_DMA () или HAL_QSPI_Receive_IT () после настройки команды:

- В режиме опроса вывод функции выполняется после завершения передачи.
- В режиме прерывания HAL_QSPI_FifoThresholdCallback () будет вызываться при достижении порога fifo, а HAL_QSPI_RxCpltCallback () - при завершении передачи.
- В режиме DMA HAL_QSPI_RxHalfCpltCallback () будет вызываться при половинной передаче, а HAL_QSPI_RxCpltCallback () будет вызываться после завершения передачи.

Auto-polling functional mode (режим авто-опроса)

1. Сконфигурируйте последовательность команд и функциональный режим автоматического опроса, используя функции HAL_QSPI_AutoPolling () или HAL_QSPI_AutoPolling_IT ():

- Фаза инструкции: используемый режим и, если присутствует, код операции инструкции.
- Фаза адреса: используемый режим и, если он указан, размер и значение адреса.
- Фаза альтернативных байтов: используемый режим и, если присутствует, значения размера и альтернативных байтов.
- Фаза фиктивных циклов: количество фиктивных циклов (используемый режим такой же, как и фаза данных).
- Фаза данных: используемый режим.
- Режим двойной скорости передачи данных (DDR): активация (или нет) этого режима и задержка, если активирована.
- Режим отправки инструкции только один раз (SIOO): активация (или нет) этого режима.
- Размер байтов состояния, значение совпадения, используемая маска, режим сопоставления (ИЛИ / И), интервал опроса и активация автоматической остановки.

2. После настройки:

В режиме опроса вывод функции выполняется при достижении соответствия состояния. Автоматический останов активируется, чтобы избежать бесконечного цикла.

- В режиме прерывания HAL_QSPI_StatusMatchCallback () будет вызываться каждый раз, когда достигается соответствие состояния.

Memory-mapped functional mode (режим отображения памяти)

1. Сконфигурируйте последовательность команд и отображенный в память функциональный режим, используя функции HAL_QSPI_MemoryMapped ():

- Фаза инструкции: используемый режим и, если присутствует, код операции инструкции.

- Адресная фаза: используемый режим и размер.
- Фаза альтернативных байтов: используемый режим и, если присутствует, значения размера и альтернативных байтов.
- Фаза фиктивных циклов: количество фиктивных циклов (используемый режим такой же, как и фаза данных).
- Фаза данных: используемый режим.
- Режим двойной скорости передачи данных (DDR): активация (или нет) этого режима и задержка, если активирована.
- Режим отправки инструкции только один раз (SIOO): активация (или нет) этого режима.
- Время ожидания активации и время ожидания.

2. После настройки QuadSPI будет использоваться, как только будет выполнен доступ к AHB в диапазоне адресов. HAL_QSPI_TimeOutCallback () будет вызван по истечении времени ожидания.

Управление ошибками и завершением

1. Функция HAL_QSPI_GetError () выдает ошибку, возникшую во время последней операции.

2. Функции HAL_QSPI_Abort () и HAL_QSPI_AbortIT () отменяют любую текущую операцию и сбрасывают fifo:

- В режиме опроса вывод функции выполняется, когда бит завершения передачи установлен и бит занят очищен.
- В режиме прерывания, HAL_QSPI_AbortCpltCallback () будет вызываться, когда будет установлен bi завершения передачи.

Контрольные функции

1. Функция HAL_QSPI_GetState () отображает текущее состояние драйвера HAL QuadSPI.

2. Функция HAL_QSPI_SetTimeout () настраивает значение времени ожидания, используемое в драйвере.

3. Функция HAL_QSPI_SetFifoThreshold () настраивает пороговое значение для Fifo QSPI IP.

4. Функция HAL_QSPI_GetFifoThreshold () выдает текущее значение порога Fifo.

Обходные пути, связанные с кремниевым ограничением

1. Временные решения, реализованные в драйвере HAL

Дополнительные данные, записанные в FIFO в конце передачи чтения

49.2.2 Функции инициализации и конфигурации

Этот подраздел предоставляет набор функций, позволяющих:

- Инициализировать QuadSPI.
- Деинициализировать QuadSPI.

Этот раздел содержит следующие API:

HAL_QSPI_Init ()
HAL_QSPI_DeInit ()
HAL_QSPI_MspInit ()
HAL_QSPI_MspDeInit ()

49.2.3 Операционные функции ввода-вывода

Этот подраздел предоставляет набор функций, позволяющих:
- Обрабатывать прерывания.
- Обработайте последовательность команд.
- Передача данных в режиме блокировки, прерывания или DMA.
- Получать данные в режиме блокировки, прерывания или DMA.
- Управление режимом автоматического опроса.
- Управление отображенным в память функциональным режимом.

Этот раздел содержит следующие API:
HAL_QSPI_IRQHandler ()
HAL_QSPI_Command()
HAL_QSPI_Command_IT()
HAL_QSPI_Transmit()
HAL_QSPI_Receive()
HAL_QSPI_Transmit_IT()
HAL_QSPI_Receive_IT()
HAL_QSPI_Transmit_DMA()
HAL_QSPI_Receive_DMA()
HAL_QSPI_AutoPolling()
HAL_QSPI_AutoPolling_IT()
HAL_QSPI_MemoryMapped()
HAL_QSPI_ErrorCallback()
HAL_QSPI_AbortCpltCallback()
HAL_QSPI_CmdCpltCallback()
HAL_QSPI_RxCpltCallback()
HAL_QSPI_TxCpltCallback()
HAL_QSPI_RxHalfCpltCallback()
HAL_QSPI_TxHalfCpltCallback()
HAL_QSPI_FifoThresholdCallback()
HAL_QSPI_StatusMatchCallback()

49.2.4 Периферийный контроль и функции состояния

Этот подраздел предоставляет набор функций, позволяющих:
- Проверьте во время выполнения состояние водителя.
- Проверьте код ошибки, установленный во время последней операции.
- Прервать любую операцию. .....

Этот раздел содержит следующие API:
HAL_QSPI_GetState()
HAL_QSPI_GetError()
HAL_QSPI_Abort()
HAL_QSPI_Abort_IT()
HAL_QSPI_SetTimeout()
HAL_QSPI_SetFifoThreshold()
HAL_QSPI_GetFifoThreshold()
HAL_QSPI_ErrorCallback()
HAL_QSPI_AbortCpltCallback()

HAL_QSPI_FifoThresholdCallback()
HAL_QSPI_CmdCpltCallback()
HAL_QSPI_RxCpltCallback()
HAL_QSPI_TxCpltCallback()
HAL_QSPI_RxHalfCpltCallback()
HAL_QSPI_TxHalfCpltCallback()
HAL_QSPI_StatusMatchCallback()
HAL_QSPI_TimeOutCallback()