Введение

Эволюция мобильных, промышленных и потребительских приложений приводит к большей потребности в графических пользовательских интерфейсах (GUI) и увеличении необходимых аппаратных ресурсов. Эти приложения требуют более качественной графики, дополнительных аппаратных и программных ресурсов (таких как память для графических примитивов или буфер кадра) и более высоких характеристик обработки.

Чтобы реагировать на этот растущий спрос, часто используются микропроцессорные блоки, что приводит к более высоким затратам и более сложным проектам с более длительным выходом на рынок. Чтобы справиться с этими требованиями, MCU STM32 предлагают большой графический портфолио.

Благодаря встроенному контроллеру LCD-TFT-дисплея (LTDC), MCM STM32 позволяют напрямую управлять дисплеями с высоким разрешением без вмешательства процессора. Кроме того, LTDC может автономно обращаться к внутренним запоминающим устройствам или внешним запоминающим устройствам для извлечения данных пикселей.

В этой заявке описывается контроллер LCD-TFT-дисплея микроконтроллеров STM32, перечисленных в таблице 1, и демонстрируется, как использовать и настраивать периферийное устройство LTDC. Он также освещает некоторые аппаратные, программные и архитектурные соображения, чтобы получить лучшие графические характеристики.

Связанные документы

Доступно на веб-сайте STMicroelectronics www.st.com:

• 32-битные микроконтроллеры STM32F75xxx и STM32F74xxx с расширенными ARM® (RM0385)

• STM32F76xxx и STM32F77xxx усовершенствованные 32-битные MCU на ARM® (RM0410)

• STM32F469xx и STM32F479xx усовершенствованные 32-битные MCU на ARM® (RM0386)

• STM32F405 / 415, STM32F407 / 417, STM32F427 / 437 и STM32F429 / 439 32-битные MCU на основе ARM® (RM0090)

• STM32F429 / 439, STM32F469 / 479, STM32F7x6, STM32F7x7, STM32F7x8, Табличные данные STM32F7x9

Table 1. Applicable products

Type	Product lines
Microcontrollers	STM32F429/439, STM32F469/479, STM32F7x6, STM32F7x7, STM32F7x8, STM32F7x9

Содержание

Список таблиц

Список рисунков

1 Обзор дисплея и графики

В этом разделе описываются основные термины, используемые для отображения и графического контекста, чтобы предоставить обзор общей среды отображения и графики. В этом разделе также представлены интерфейсы дисплея, поддерживаемые MCU STM32.

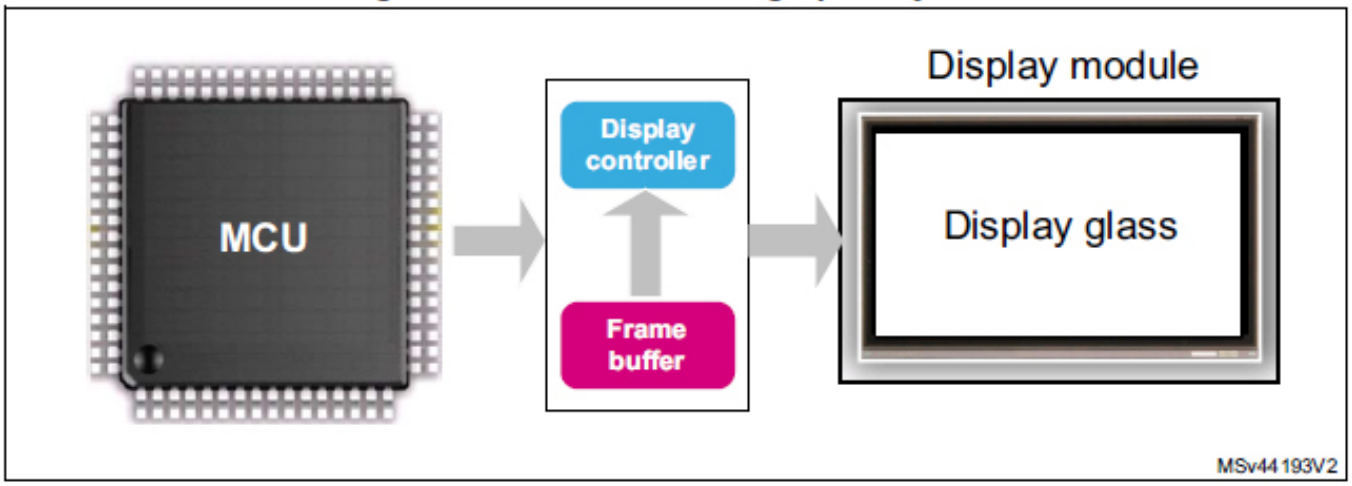
1.1 Основные концепции графики

В этом разделе описывается базовая встроенная графическая система, категории дисплея и технологии отображения.

Основная встроенная графическая система

Основная встроенная графическая система может быть схематизирована, как описано на рисунке 1.

Figure 1. Basic embedded graphic system



Основная встроенная графическая система состоит из микроконтроллера, буфер кадра, контроллера дисплея и стекла дисплея.

• Микроконтроллер вычисляет изображение, которое будет отображаться в буфере кадра, собирает графические примитивы, такие как значки или изображения. CPU выполняет эту операцию, запустив программное обеспечение графической библиотеки. Этот процесс можно ускорить с помощью специального оборудования, такого как DMA2D Chrom-Art Accelerator®, используемого графической библиотекой. Чем чаще обновляется буфер кадра, тем более плавные анимации (анимация fps).

• Буфер кадра - это энергозависимая память, используемая для хранения пиксельных данных изображения, которое будет отображаться. Эта память обычно называется графическим ОЗУ (GRAM). Требуемый размер буфера кадра зависит от разрешения и глубины цвета дисплея. Для получения дополнительной информации о требуемом размере буфера кадра см. Раздел 4.2.1: Требования размера и размера памяти Framebuffer.

- Двойная буферизация - это метод, который использует двойные буферы кадра, чтобы не отображать то, что записывается в буфер кадра.

• Контроллер дисплея постоянно «освежает» дисплей, передавая содержимое буфера кадра на стекле дисплея 60 раз в секунду (60 Гц). Контроллер дисплея может быть встроен либо в дисплейный модуль, либо в MCU.

• Стекло дисплея управляется контроллером дисплея и отвечает за отображение изображения (которое состоит из матрицы пикселей).

Дисплей характеризуется:

- Размер дисплея (разрешение): определяется количеством пикселей дисплея, которое представлено горизонтальным (число пикселей) x вертикально (номера строк).

- Глубина цвета: определяет количество цветов, в которых можно рисовать пиксель. Он представлен в битах на пиксель (bpp). Для глубины цвета 24 бит / с (что также может быть представлено RGB888) пиксель может быть представлен в 16777216 цветах.

- Частота обновления (в Гц): количество раз в секунду, которое панель дисплея обновляется. Дисплей должен обновляться 60 раз в секунду (60 Гц), поскольку более низкая частота обновления создает плохие визуальные эффекты.

Категории модулей

Модули дисплея классифицируются в двух основных категориях, в зависимости от того, встроены ли они или нет внутренний контроллер и ГРАМ.

• Первая категория соответствует дисплеям с контроллером отображения на стекле и GRAM (см. Рисунок 2).

• Вторая категория соответствует дисплеям со стеклянным дисплеем без главного контроллера и имеет только низкоуровневый контроллер синхронизации.

Чтобы взаимодействовать с дисплеями без контроллера или GRAM, используемый буфер кадра может быть расположен во внутренней SRAM MCU (см. Рис. 3) или расположен во внешней памяти (см. Рис. 4).

Figure 2. Display module with embedded controller and GRAM

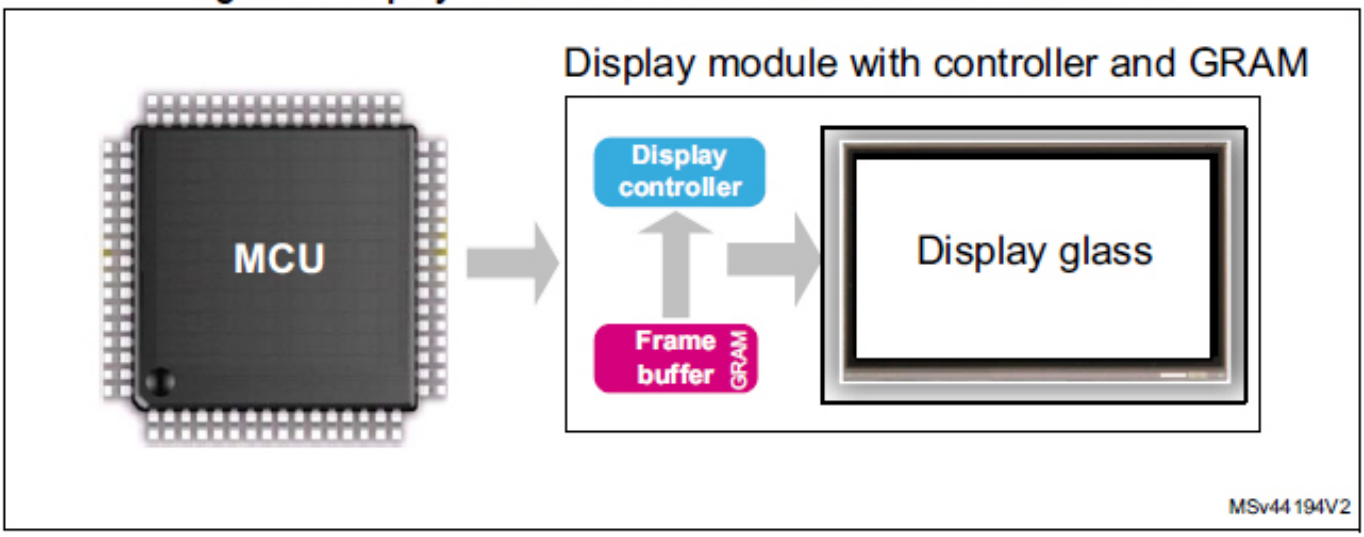


Figure 3. Display module without controller nor GRAM

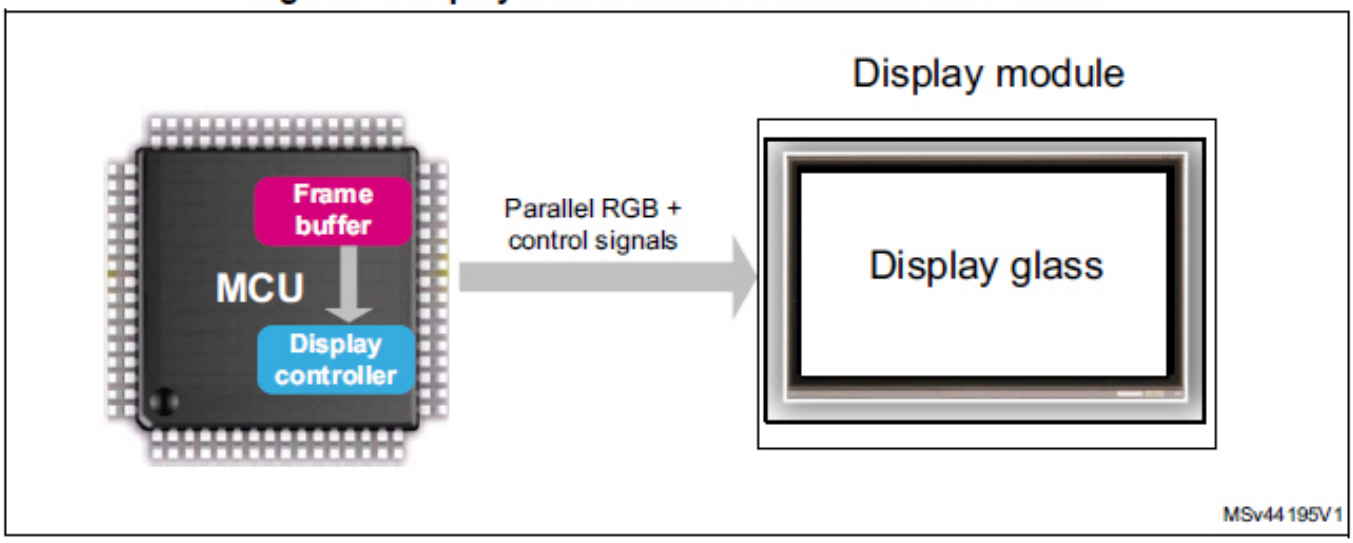
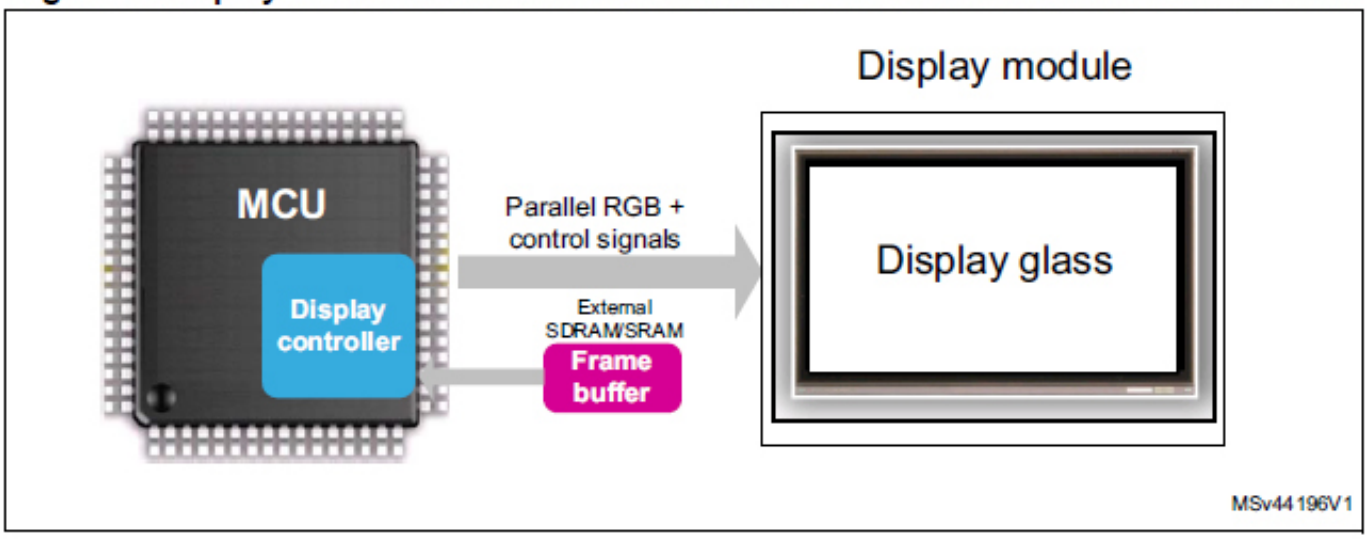


Figure 4. Display module without controller nor GRAM and with external framebuffer



Технологии отображения

На рынке существует множество технологий отображения, используемые две основные технологии описаны ниже:

• LCD-TFT-дисплеи (жидкокристаллический дисплей - тонкопленочный транзистор): это вариант ЖК-дисплея, который использует технологию TFT для улучшения управления каждым пикселем. Благодаря технологии TFT каждый пиксель может управляться транзистором, что обеспечивает быстрое время отклика и точное управление цветом.

• OLED-дисплеи (органический светодиод): пиксели сделаны из органических светодиодов, излучающих непосредственно свет, что обеспечивает лучший контраст и оптимальное потребление. Технология OLED позволяет использовать гибкие дисплеи, так как не требуется никакого стекла или подсветки. Время отклика очень быстрое и угол обзора свободен, так как он не зависит от какой-либо поляризации света.

Способ управления модулем отображения очень похож на технологии TFT и OLED, основное отличие заключается в необходимости подсветки, поскольку OLED не требует никаких настроек.

1.2 Стандарты интерфейса дисплея

Альянс MIPI (мобильный процессорный процессор) - глобальная совместная организация, которая занимается определением и продвижением спецификаций интерфейса для мобильных устройств. Альянс MIPI разрабатывает новые стандарты, но также стандартизирует существующие интерфейсы отображения:

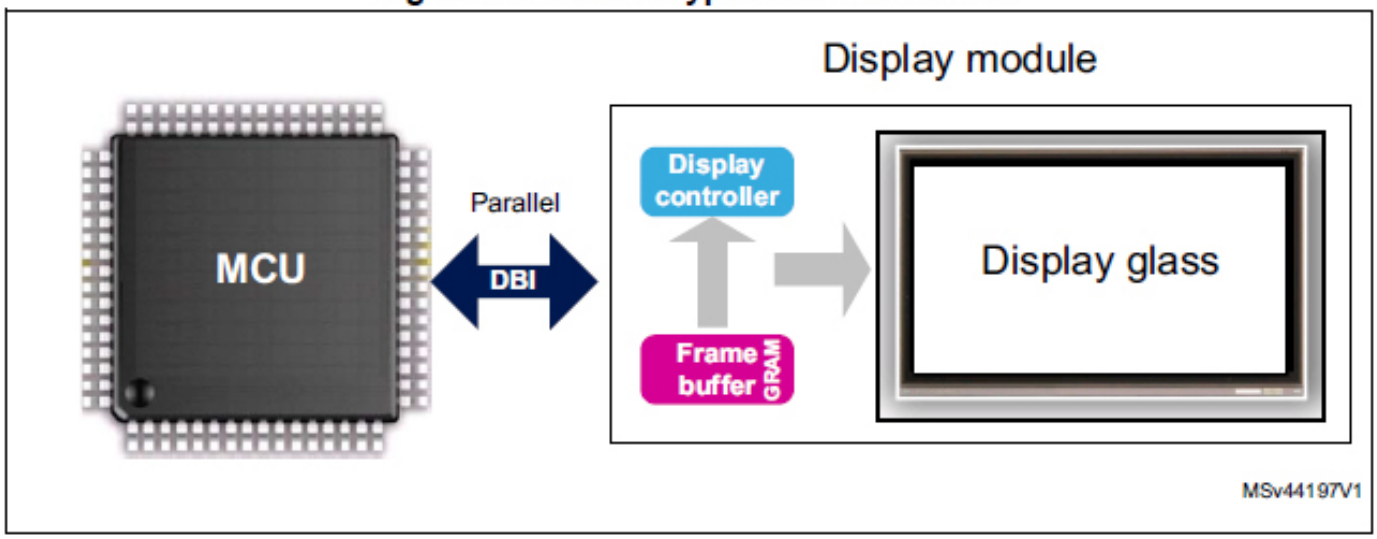
Интерфейс шины MIPI (MIPI-DBI)

MIPI-DBI является одним из первых стандартов отображения, опубликованных MIPI Alliance, для указания интерфейсов дисплея. Три типа интерфейсов, определенные внутри MIPI-DBI:

- Тип A: на базе шины Motorola 6800
- Тип B: на базе шины Intel® 8080
- Тип C: на основе протокола SPI

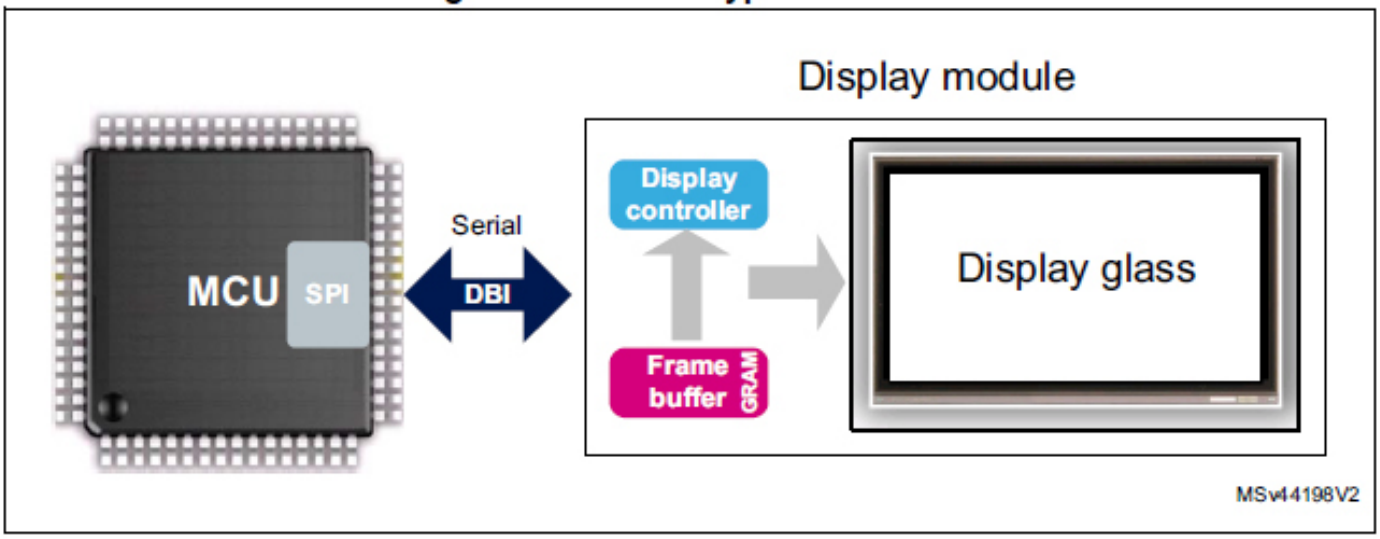
MIPI-DBI используется для интерфейса с дисплеем со встроенной графической памятью (GRAM). Данные пикселя обновляются в локальном GRAM дисплея. На рисунке 5 показан пример интерфейса сопряжения MIPI-DBI типа A или B.

Figure 5. MIPI-DBI type A or B interface



На рисунке 6 показан пример интерфейса интерфейса интерфейса MPI-типа DBI.

Figure 6. MIPI-DBI type C interface



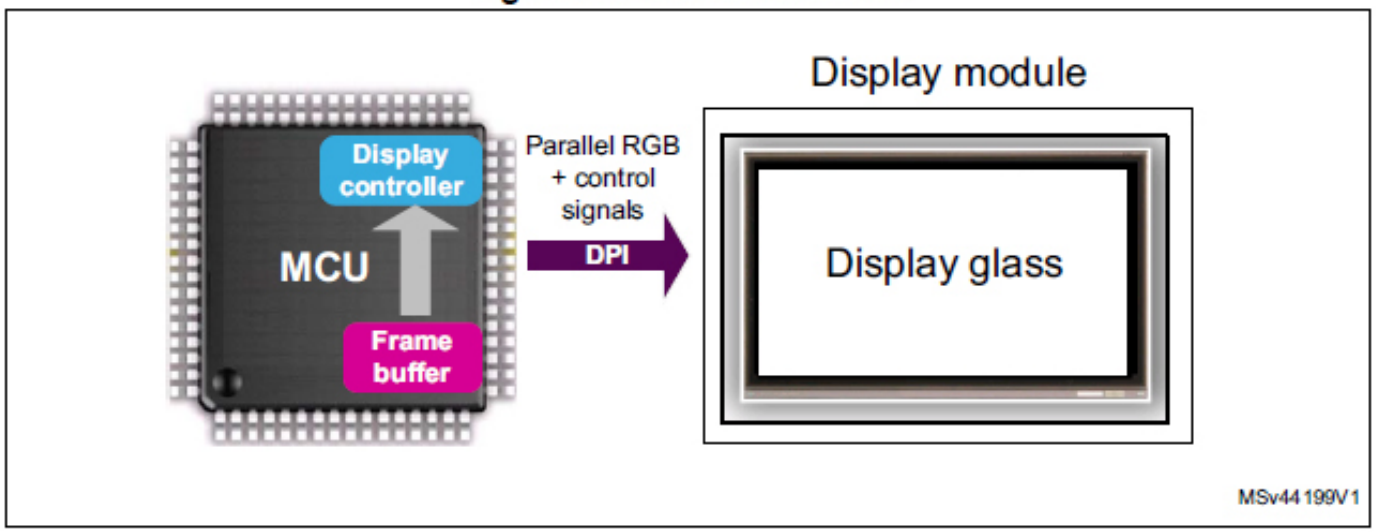
MIPI-параллельный интерфейс (MIPI-DPI)

DPI стандартизирует интерфейс через TFT-контроллер. Например, при использовании 16-24-битной сигнализации RGB в сочетании с сигналами синхронизации (HSYNC, VSYNC, EN и LCD_CLK).

DPI используется для интерфейса с дисплеем без буфера кадра. Данные пикселя должны быть переданы в реальном времени на дисплей.

Производительность в реальном времени отличная, но для MCU требуется высокая пропускная способность в MCU

Figure 7. MIPI-DPI interface



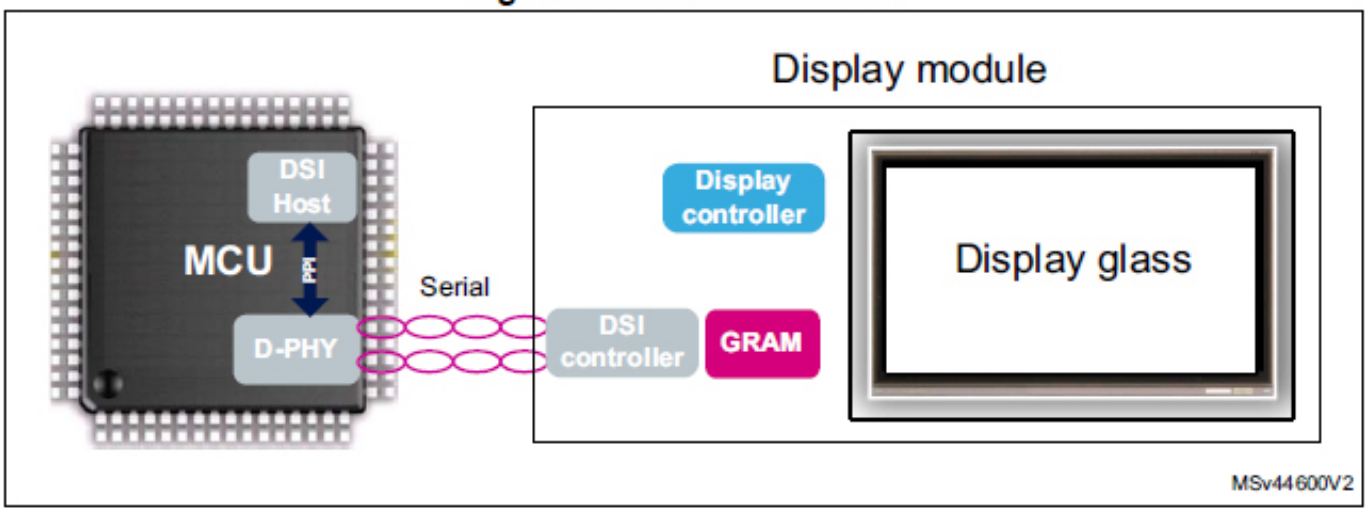
Последовательный интерфейс дисплея MIPI (MIPI-DSI)

Чтобы уменьшить количество линий для интерфейса с дисплеем, MIPI Alliance определил DSI. DSI - это высокоскоростная многоканальная дифференциальная линия; он использует стандартный MIPI D-PHY для физической ссылки.

DSI инкапсулирует сигналы DBI или DPI и передает их в D-PHY через протокол PPI.

На рисунке 8 показан пример интерфейса интерфейса MPI-DSI.

Figure 8. MIPI-DSI interface



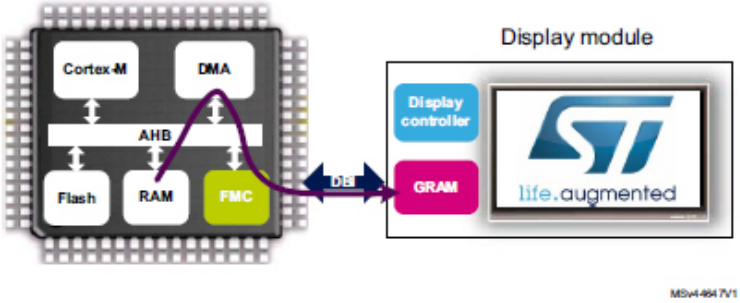
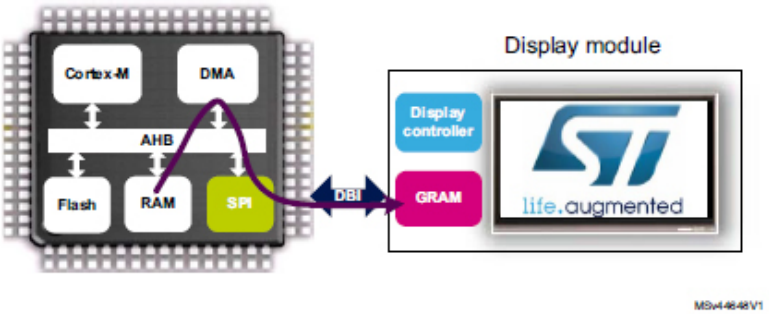
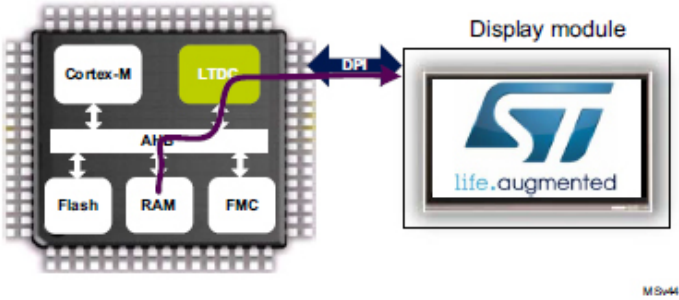
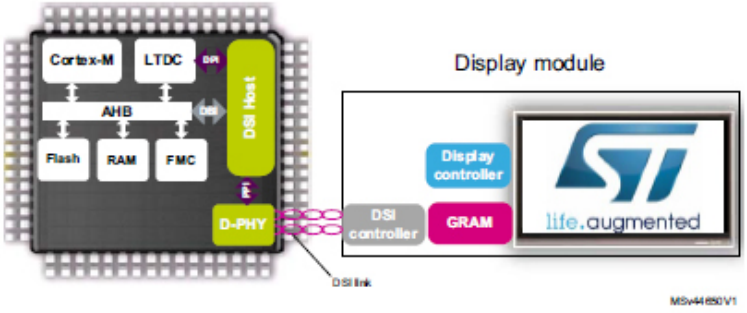
1.3 Интерфейсы дисплея, поддерживаемые MCU STM32

Ниже приведен краткий обзор интерфейсов отображения MIPI Alliance, поддерживаемых MCU STM32:

- Все STM32 MUC поддерживают интерфейс MIPI-DBI типа C (SPI)
- Все MCU STM32 с F (S) MC поддерживают интерфейсы MIPI-DBI типа A и B
- MCU STM32 с поддержкой LTDC поддерживают интерфейс MIPI-DPI
- MCU STM32, внедряющие хост DSI, поддерживают интерфейс MIPI-DSI

Таблица 2 иллюстрирует и суммирует интерфейсы дисплея, поддерживаемые микроконтроллерами STM32.

Table 2. Display interfaces supported by STM32 MCUs

Display interface		Connecting display panels to STM32 MCU(1)
DBI(2)	Motorola 6800 DBI Type A Intel 8080 DBI Type B	
	SPI DBI Type C	
DPI: Parallel RGB using LTDC(3)		
DSI(4)		

1. Фиолетовые стрелки показывают путь к пиксельным данным на дисплее.

2. Для получения дополнительной информации о том, как поддерживать Motorola 6800 и Intel 8080 с MC (ST) STM32, см. Примечание к приложению TFT LCD с интерфейсом FSMC (AN2790) высокой плотности STM32F10xxx.

3. Все другие MCM STM32 без периферийного устройства LTDC могут напрямую управлять панелями LCD-TFT с использованием FSMC и DMA. См. Примечание к приложению. Прямой привод QVGA TFT-LCD с использованием периферийного устройства FSMC STM32F10xx (AN3241).

4. Только MCM STM32, указанные в Таблице 3, встраивают хост DSI, могут поддерживать интерфейс DSI. Обратитесь к примечанию к приложению DSI Host на микроконтроллерах STM32 (AN4860) для получения дополнительной информации.

2 Обзор графического портфеля контроллеров LTDC и STM32 MCUs

Этот раздел иллюстрирует преимущества контроллера LTDC и суммирует графический портфель микроконтроллеров STM32.

2.1 Контроллер дисплея LCD-TFT на MCM STM32

LTDC на микроконтроллерах STM32 является встроенным контроллером ЖК-дисплея, который обеспечивает до 24-битных параллельных цифровых сигналов RGB для взаимодействия с различными панелями дисплея. LTDC также может управлять другими технологиями отображения с параллельным интерфейсом RGB, таким как дисплеи AMOLED. LTDC позволяет взаимодействовать с недорогими панелями дисплея, которые не встраивают ни контроллер, ни графическое ОЗУ. 2.2 Доступность LTPS и графический портфолио по семействам STM32

В Таблице 3 суммируется вложение STM32 LTDC и детализируется соответствующий доступный графический портфолио.

Table 3. STM32 MCUs embedding an LTDC and their available graphic portfolio

STM32 lines	FLASH (bytes)	On chip SRAM (bytes)	Quad-SPI(1)	Max AHB frequency (MHz)(2)	Max FMC SRAM and SDRAM frequency (MHz)	Max pixel clock (MHz) (3)	JPEG codec	DMA2D (4)	MIPI -DSI host (5)	Graphic libraries
STM32F429/ 439	Up to 2 M	256 k	No	180	90	83	No	Yes	No	TouchGFX Embedded wizard SEGGER STemWin
STM32F469/ 479	Up to 2 M	384 k	Yes	180	90	83	No	Yes	Yes	
STM32F7x6	Up to 1 M	320 k	Yes	216	100	83	No	Yes	No	
STM32F7x7	Up to 2 M	512 k	Yes	216	100	83	Yes	Yes	No	
STM32F7x8/ STM32F7x9			Yes	216	100	83	Yes	Yes	Yes	

1. Интерфейс Quad-SPI позволяет взаимодействовать с внешними памятью, чтобы расширить размер приложения. Для получения более подробной информации о STM32 MCUs интерфейс QSPI см. В описании приложения Интерфейс Quad-SPI (QSPI) на микроконтроллерах STM32 (AN476).

2. LTDC извлекает графические данные со скоростью AHB.

3. Максимальное значение тактового сигнала пикселя на уровне IO, см. Таблицу 10 и таблицу 11 для максимального количества пикселей на системном уровне.

Пиксельные часы (LCD_CLK) образуют соответствующее техническое описание STM32.

4. Chrom-Art Accelerator®

5. Интегрированный контроллер MIPI-DSI позволяет упростить конструкцию печатной платы с меньшим количеством контактов, более низким электромагнитным помехам (EMI) и меньшим энергопотреблением. Более подробную информацию о хосте MIPI-DSI STM32 см. В примечании к приложению AN4860.

2.3 LTDC в интеллектуальной архитектуре

LTDC является мастером архитектуры AHB, который обеспечивает доступ на чтение во внутренней и внешней памяти. LTDC имеет два независимых уровня, каждый из которых имеет свой собственный FIFO, что обеспечивает большую гибкость отображения.

Контроллер LTDC автономно извлекает графические данные со скоростью шины AHB из буфера кадра. Затем графические данные сохраняются в одном из внутренних слоев FIFO, затем приводятся на дисплей.

Архитектура системы позволяет создавать и отображать графику на экране без вмешательства ЦП. LTDC извлекает данные, принадлежащие изображению из буфера кадра, а Chrom-Art Accelerator® (DMA2D) готовит следующие изображения.

Интерфейс LTDC интегрирован в интеллектуальную архитектуру, позволяющую:

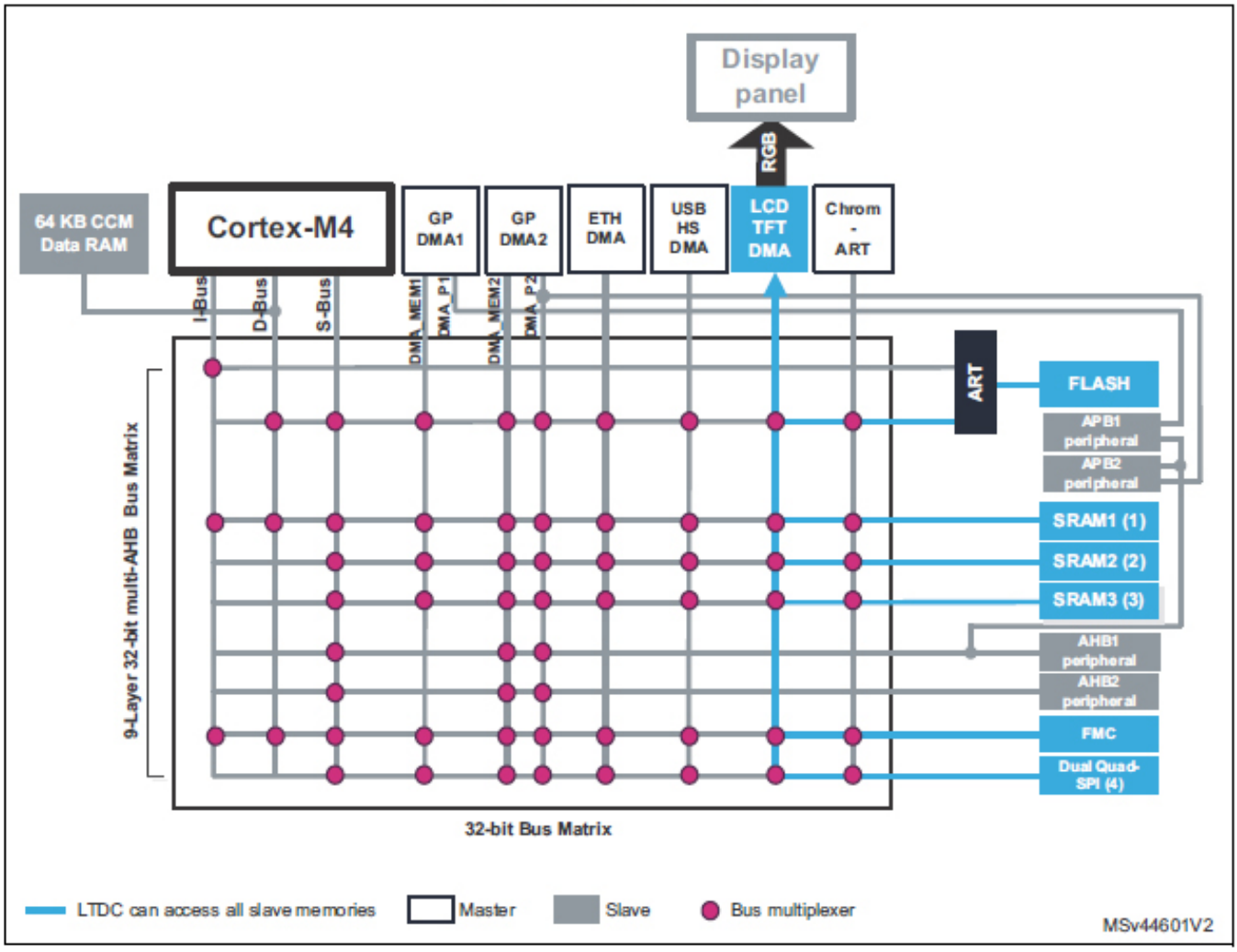
- LTDC автономно извлекает графические данные из буфера кадра (могут быть внутренние запоминающие устройства, такие как внутренняя вспышка, внутренняя SRAM или внешние запоминающие устройства, такие как FMC_SDRAM или Quad-SPI) и выводит их на дисплей.
- DMA2D как мастер AHB может использоваться для выгрузки процессора из графических задач.
- LTDC может продолжать показывать графику даже в спящем режиме, когда процессор не работает.
- Многослойная архитектура шины AHB улучшает пропускную способность памяти и ведет к повышению производительности.

Архитектура системы на микроконтроллерах STM32F429 / 439 и STM32F469 / 479

Системная архитектура линии STM32F429 / 439 и линии STM32F469 / 479 состоит в основном из 32-разрядной многослойной матрицы шины AHB, которая объединяет десять мастеров и девять подчиненных устройств (восемь подчиненных для STM32F429 / F439). LTDC является одним из десяти мастеров AHB на шинной матрице AHB.

LTDC может автономно получать доступ ко всем ведомым устройствам памяти на матрице шины AHB, таким как FLASH, SRAM1, SRAM2, SRAM3 FMC или Quad-SPI, что обеспечивает эффективную передачу данных, которая идеально подходит для графических приложений. На рисунке 9 показано соединение LTDC в системах линий STM32F429 / 439 и STM32F469 / 479.

Figure 9. LTDC AHB master in STM32F429/439 and STM32F469/479 lines smart architecture



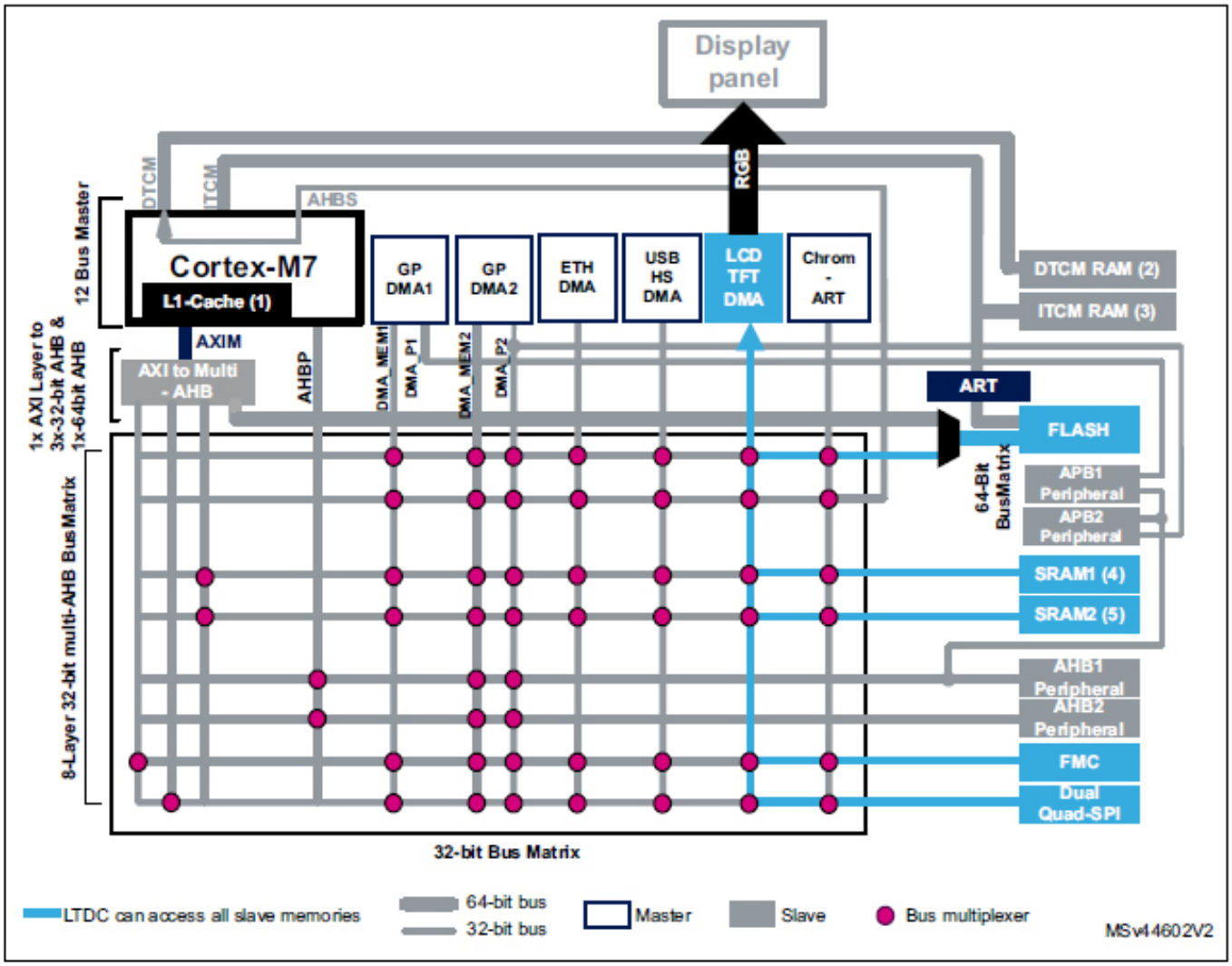
1. Размер SRAM1 = 112 Кбайт для STM32F429 / 439 и 160 Кбайт для STM32F469 / 479
2. Размер SRAM2 = 16 Кбайт для STM32F429 / 439 и 32 Кбайт для STM32F469 / 479
3. Размер SRAM3 = 64 Кбайт для STM32F429 / 439 и 128 Кбайт для STM32F469 / 479
4. Интерфейс Dual Quad-SPI доступен только для STM32F469 / 479

Архитектура системы на STM32F7x6, STM32F7x7, STM32F7x8 и STM32F7x9

Системная архитектура строк STM32F7x6, STM32F7x7, STM32F7x8 и STM32F7x9 состоит в основном из 32-разрядной многослойной матрицы шины AHB, которая соединяет двенадцать мастеров и восемь подчиненных устройств. LTDC является одним из двенадцати мастеров AHB на шинной матрице AHB.

LTDC может автономно получать доступ ко всем ведомым устройствам памяти на матрице шины AHB, таким как FLASH, SRAM1, SRAM2, FMC или Quad-SPI, что обеспечивает эффективную передачу данных, которая идеально подходит для графических приложений. На рисунке 10 показано соединение LTDC в системах STM32F7x6, STM32F7x7, STM32F7x8 и STM32F7x9.

Figure 10. LTDC AHB master in STM32F7x6, STM32F7x7, STM32F7x8 and STM32F7x9 smart architecture



1. Размер кеша I / D = 4 Кбайт для STM32F7x6
Размер кэша ввода-вывода = 16 Кбайт для STM32F7x7, STM32F7x8 и STM32F7x9
2. Размер оперативной памяти DTCM = 64 Кбайт для STM32F7x6
Размер оперативной памяти DTCM = 128 Кбайт для STM32F7x7, STM32F7x8 и STM32F7x9
3. Размер ОЗУ ITCM = 16 Кбайт для STM32F7x6, STM32F7x7, STM32F7x8 и STM32F7x9
4. SRAM1 размер = 240 Кбайт для STM32F7x6
SRAM1 размер = 368 Кбайт для STM32F7x7, STM32F7x8 и STM32F7x9
5. Размер SRAM2 = 16 Кбайт для STM32F7x6, STM32F7x7, STM32F7x8 и STM32F7x9

2.4. Преимущества использования контроллера STM32 LTDC

В таблице 4 приведены основные преимущества использования встроенного LTM-интерфейса STM32.

Таблица 4. Преимущества использования STM32 MCUs Контроллер LTDC

Преимущество	Комментарии
Экономия затрат	По сравнению с другими интерфейсами DBI (SPI, Motorola 6800 или Intel 8080), LTDC позволяет подключать любой недорогой дисплейный модуль без контроллера дисплея и GRAM.

Незагруженность ЦПУ	LTDC является AHB-мастером со своим собственным DMA, который извлекает данные автономно из любой AHB-памяти без вмешательства процессора.
Нет необходимости в дополнительном аппликативном слое	Оборудование LTDC полностью управляет извлечением данных, выводом RGB и управлением сигналами, поэтому нет необходимости в дополнительном аппликативном слое.
Полностью программируемое разрешение, поддерживающее пользовательские и стандартные дисплеи	Полностью программируемое разрешение с общей шириной до 4096 пикселей и общей высотой до 2048 строк и с тактовой частотой до 83 МГц. Поддержка пользовательских и стандартных разрешений (QVGA, VGA, SVGA, WVGA, XGA, HD и др.).
Гибкий цветной формат	Каждый уровень LTDC может быть настроен для извлечения буфера кадра в желаемом формате пикселей (см. Раздел 3.3.2: Программируемый слой: цветной буфер кадра).
Гибкий параллельный интерфейс RGB	Гибкий параллельный интерфейс RGB позволяет управлять 16-битными, 18-битными и 24-битными дисплеями.
Идеально подходит для маломощных и мобильных приложений, таких как smartwatches.	LTDC способен продолжить выборку графических данных и отображать вождение, когда CPU находится в режиме SLEEP.

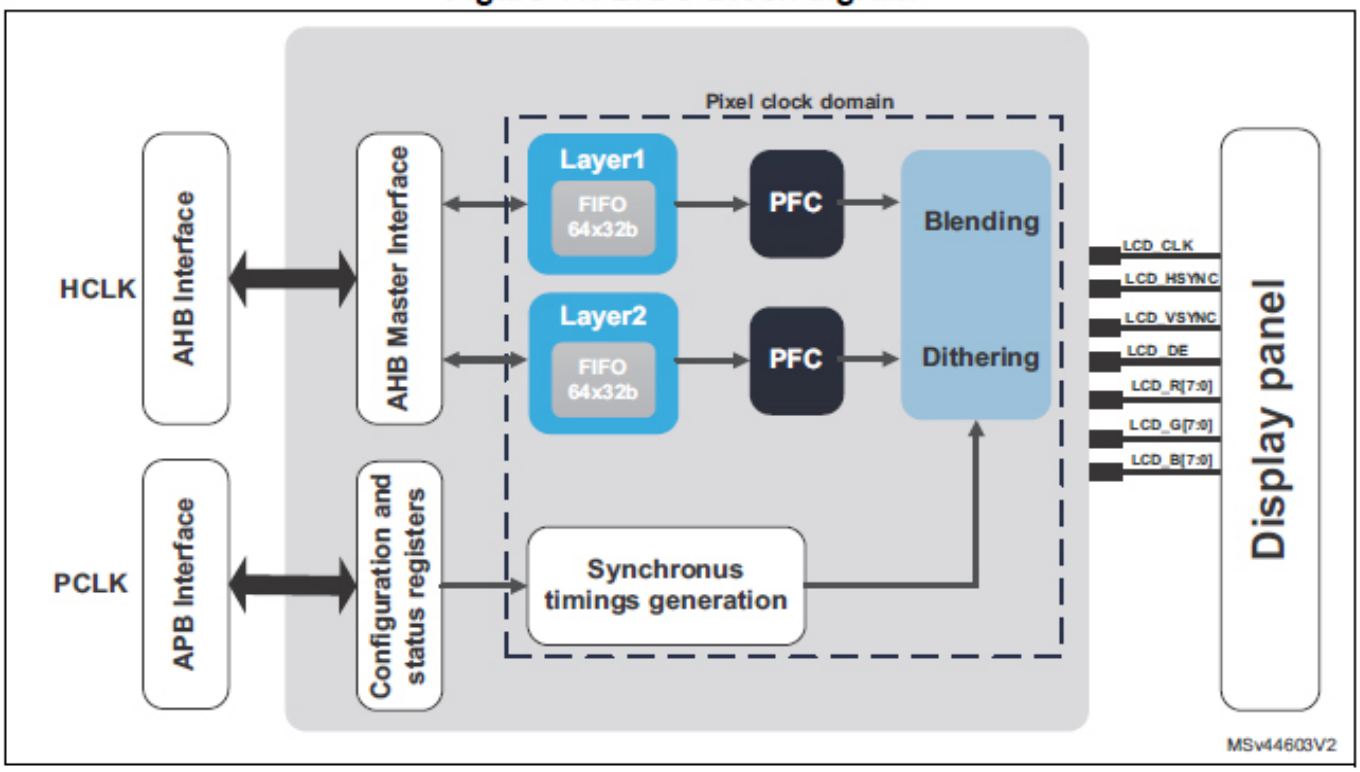
3 Описание контроллера дисплея LCD-TFT (LTDC)

LTDC – это контроллер, который считывает данные изображений по линии на линию. Режим доступа к памяти имеет длину 64 байта, но когда конец линии достигнут и осталось меньше 64 байт, LTDC извлекает оставшиеся данные.

3.1 Функциональное описание

На каждом пиксельном фронте нарастания тактового сигнала или краю синхронизации по времени и в активной области экрана слой LTDC извлекает один пиксель из своего FIFO, преобразует его во внутренний формат ARGB8888 и смешивает его с фоном и / или с другим цветом слоя. Полученный пиксель, закодированный в формате RGB888, проходит через блок сглаживания и приводится в интерфейс RGB. Затем на экране отображается пиксель.

Figure 11. LTDC Block digram



3.1.1 Домены часов LTDC

В периферийной части контроллера LCD-TFT используются три тактовых домена:

- AHB clock domain (HCLK): используется для передачи данных из памяти на уровень FIFO и наоборот.
- APB clock domain (PCLK): используется для доступа к регистрам конфигурации и состояния.
- Область пиксельных часов (LCD_CLK): используется для генерации сигналов интерфейса LCD-TFT. Выход LCD_CLK должен быть настроен в соответствии с требованиями панели через PLL.

3.1.2 Сброс LTDC

LTDC сбрасывается путем установки бит LTDCRST в регистре RCC_APB2RSTR.

3.2 Гибкие тайминги и аппаратный интерфейс

Благодаря гибкости и гибкости аппаратного интерфейса контроллер LCD-TFT способен управлять несколькими мониторами с различными разрешениями и полярностью сигнала.

3.2.1 Штыри LCD-TFT и интерфейс сигнала

Для управления LCD-TFT-дисплеями LTDC обеспечивает до 28 сигналов с использованием простой передачи сигналов 3,3 В, включая:

- Пиксельные часы LCD_CLK.
- Разрешение данных LCD_DE.
- Сигналы синхронизации (LCD_HSYNC и LCD_VSYNC).
- Пиксельные данные RGB888.

Примечание. *Контроллер LTDC может поддерживать другие технологии отображения, если их интерфейс совместим.*

Выходные сигналы интерфейса LTDC показаны в таблице 6.

Таблица 5. Выходные сигналы интерфейса LTDC.

LCD-TFT сигнал	Описание
LCD_CLK	LCD_CLK действует как сигнал достоверности данных для LCD-TFT. Данные рассматриваются дисплеем только на восходящем или падающем фронте LCD_CLK.
LCD_HSYNC	Сигнал синхронизации линии (LCD_HSYNC) управляет сканированием горизонтальной линии и выступает в качестве строкового стробирования.
LCD_VSYNC	Сигнал синхронизации кадра (LCD_VSYNC) управляет вертикальным сканированием и действует как строб обновления кадра.
LCD_DE	Сигнал DE указывает на LCD-TFT, что данные на шине RGB действительны и должны быть зафиксированы для вывода.
Pixel RGB data	Интерфейс LTDC может быть сконфигурирован для вывода более одной глубины цвета. Он может использовать до 24 строк данных (RGB888) в качестве шины интерфейса дисплея.

Другие сигналы

Обычно интерфейсы панели дисплея включают в себя другие сигналы, которые не являются частью сигналов LTDC, описанных в таблице 5. Эти дополнительные сигналы необходимы для того, чтобы модуль дисплея был полностью функциональным. Контроллер LTDC способен управлять только сигналами, описанными в таблице 5.

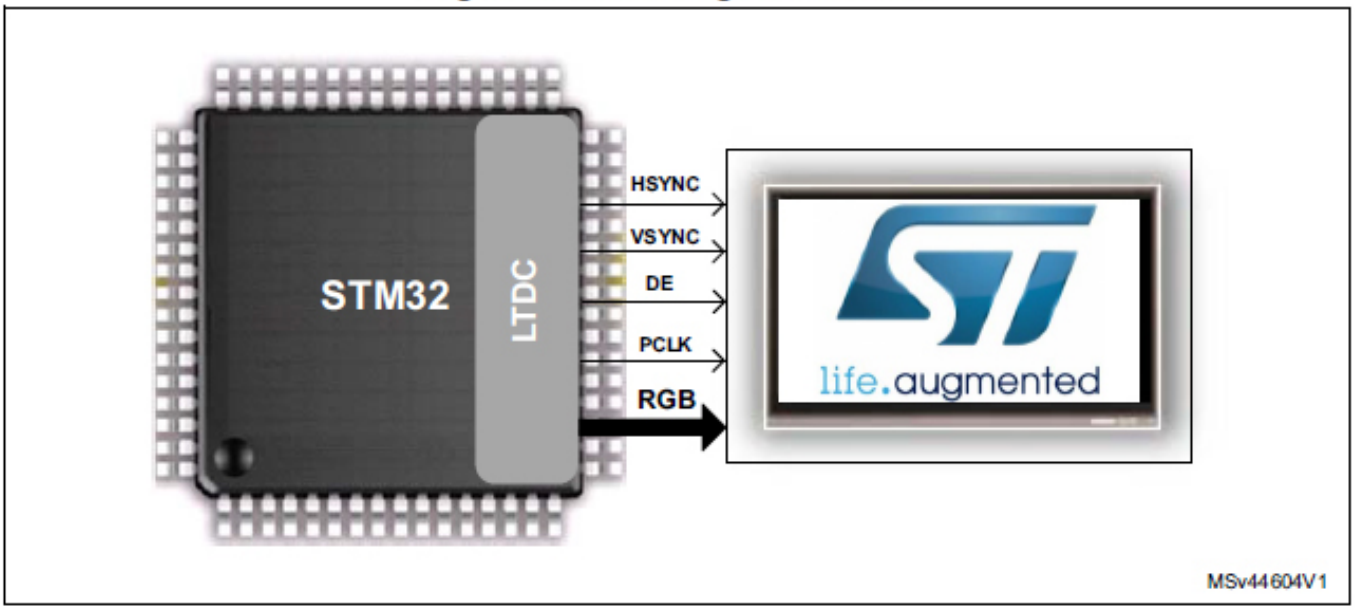
Сигналы, которые не являются частью LTDC, могут управляться с использованием GPIO и других периферийных устройств и могут нуждаться в конкретных схемах.

Обычно на панели дисплея встроен блок подсветки, для которого требуется дополнительная схема управления подсветкой и GPIO.

Некоторым панелям дисплея нужен сигнал сброса, а также последовательный интерфейс, такой как I2C или SPI. Эти интерфейсы используются в общем случае для команд инициализации дисплея или для управления сенсорной панелью.

На рисунке 12 показана панель дисплея, подключенная к MCM STM32 с использованием сигналов интерфейса LTDC, показанных в таблице 5.

Figure 12. LTDC signal interface



LTDC может выводить данные в соответствии со следующими параллельными форматами: RGB565, RGB666 и RGB888. Таким образом, можно подключить 16-разрядный RGB565, 18-битный RGB888 или 24-битный RGB888.

Программируемая полярность сигналов LTDC

Полярность сигналов управления LTDC программируется, позволяя микроконтроллеру STM32 управлять любым параллельным дисплеем RGB. Управляющие сигналы (Hsync, Vsync и разрешение данных DE), а также тактовые импульсы пикселей (LCD_CLK) могут быть определены как активные с высоким или низким уровнем активности через регистр LTDC_GCR

3.2.2 Полностью программируемые тайминги для разных размеров дисплея

Благодаря своей «гибкости таймингов» периферийное устройство LTDC может поддерживать любой размер дисплея, который соответствует максимальным программируемым параметрам синхронизации в регистрах и максимальным поддерживаемым пиксельным часам, описанным в таблице 3, таблице 11 и таблице 13.

Пользователь должен учитывать регистры тайминга, описанные в таблице 6, при программировании, поскольку таймеры LTDC и сигналы синхронизации должны быть запрограммированы в соответствии с спецификацией дисплея.

В таблице 6 приведены регистры таймингов, поддерживаемые LTDC.

Table 6. LTDC timing registers

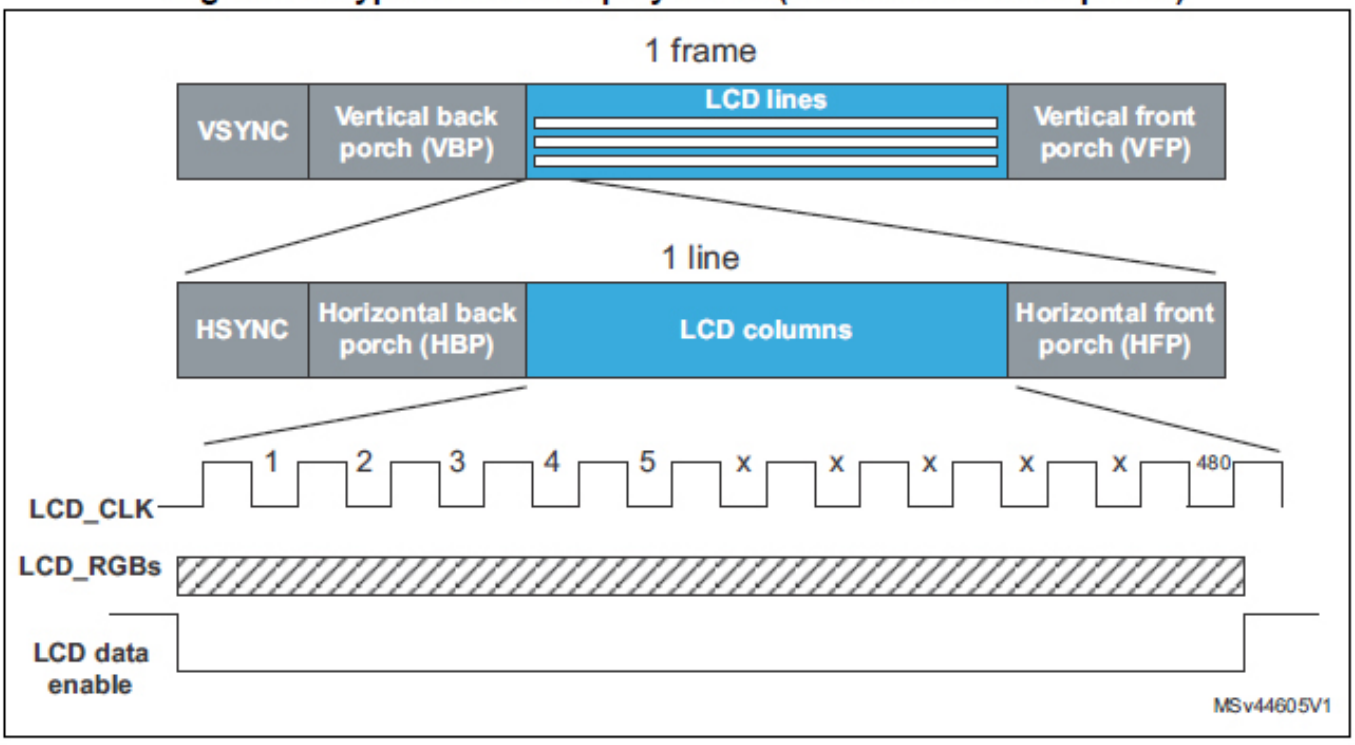
Register		Timing parameter	Value to be programmed
LTDC_SSCR(1)	HSW[11:0]	HSYNC width - 1	From 1 to 4096 pixels
	VSH[11:0]	VSYNC height - 1	From 1 to 2048 lines
LTDC_BPCR	AHBP[11:0]	HSYNC width + HBP - 1	From 1 to 4096 pixels
	AVBP[10:0]	VSYNC height + VBP - 1	From 1 to 2048 lines
LTDC_AWCR	AAW[11:0]	HSYNC width + HBP + active width - 1	From 1 to 4096 pixels
	AAH[10:0]	VSYNC height + BVBP + active height - 1	From 1 to 2048 lines
LTDC_TWCR	TOTALW[11:0]	HSYNC width + HBP + active width + HFP - 1	From 1 to 4096 pixels
	TOTALH[10:0]	VSYNC height + BVBP + active height + VFP - 1	From 1 to 2048 lines

1. Установка HSYNC в 0 в HSW [11: 0] дает одну ширину импульса одного LCD_CLK. Установка VSYNC в 0 в VSW [11: 0] дает один общий период строки

Пример типичного кадра дисплея LTDC

На рисунке 13 показан пример типичного кадра отображения LTDC, показывающего параметры синхронизации, описанные в таблице 6.

Figure 13. Typical LTDC display frame (active width = 480 pixels)

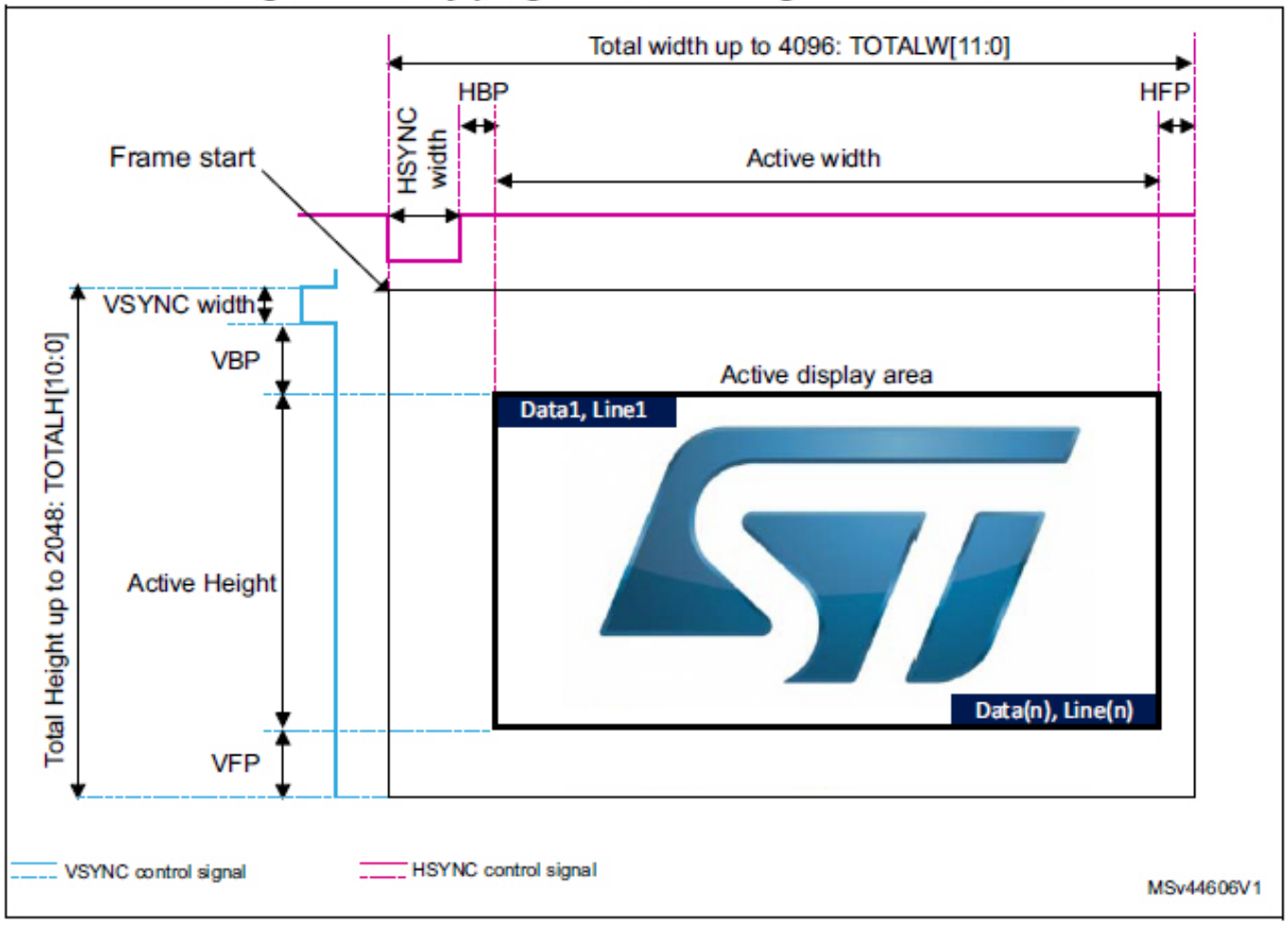


Гибкие тайминги LTDC

Периферийное устройство LTDC позволяет пользователю взаимодействовать с любым размером экрана с общей шириной до 4096 пикселей и общей высотой до 2048 строк (см. Таблицу 6).

На рисунке 14 показаны полностью программируемые тайминги и разрешения.

Figure 14. Fully programmable timings and resolutions



Внимание. *Любое разрешение дисплея, относящееся к максимальной общей площади в 4096 x 2048, как описано на рисунке 15, поддерживается LTDC только в том случае, если выполнены следующие условия:*

– Таймер пикселя панели дисплея не должен превышать максимальные часы пикселя LTDC в таблице 2

– Частота пикселов панели дисплея не должна превышать максимальные пиксельные часы STM32 в соответствии с шириной полосы кадров (см. Раздел 4.2: Проверка размера дисплея и совместимости с глубиной цвета с аппаратной конфигурацией).

На рисунке 15 показаны некоторые пользовательские и стандартные разрешения, относящиеся к максимальному 4096 x 2048, поддерживаемому LTDC.

Figure 16. Blending two layers with a background

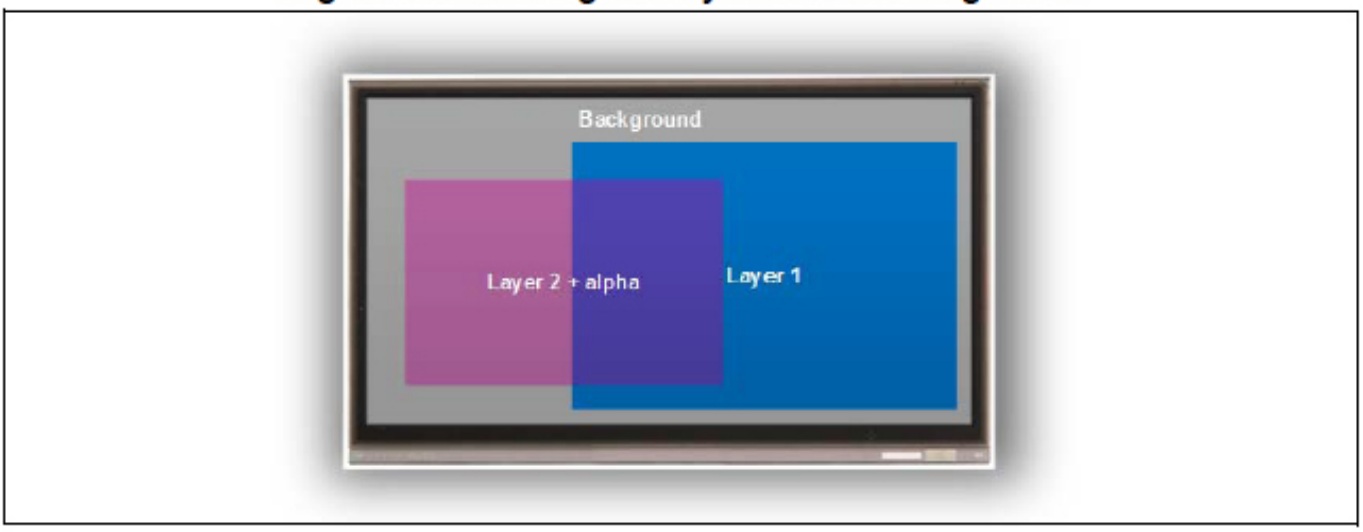
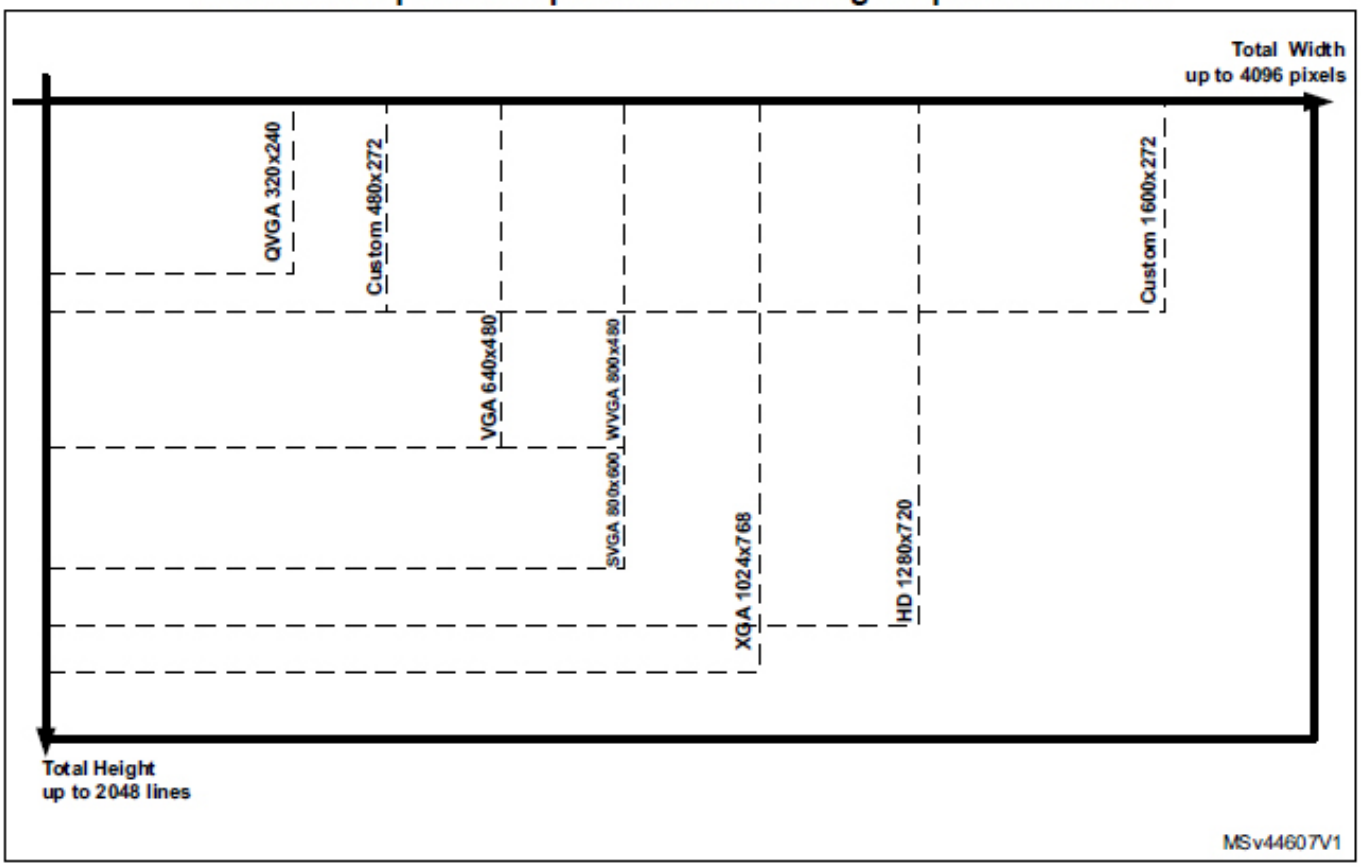


Figure 15. LTDC fully programmable display resolution with total width up to 4096 pixels and total height up to 2048 lines



1. На этом рисунке показана только активная область отображения.

3.3 Два программируемых слоя LTDC

LTDC имеет два уровня, и каждый уровень может быть включен, отключен и настроен отдельно. Порядок отображения слоя фиксирован, поэтому он всегда снизу вверх. Если два слоя включены, Layer2 является верхним отображаемым окном.

LTDC имеет настраиваемые коэффициенты смешивания. Смешивание всегда активно с использованием альфа-значения. Порядок смешивания фиксирован и всегда снизу вверх. Если включены два слоя, сначала слой Layer1 смешивается с цветом фона, а затем Layer2 смешивается с результатом смешанного цвета Layer1 и фона.

Цвет фона программируется через регистр LTDC_BCCR. Постоянный цвет фона может быть запрограммирован в формате RGB888, где для красного значения используется поле BCRED [7: 0], для зеленого значения используется BCGREEN [7: 0], и используется BCBLUE [7: 0] для синего значения.

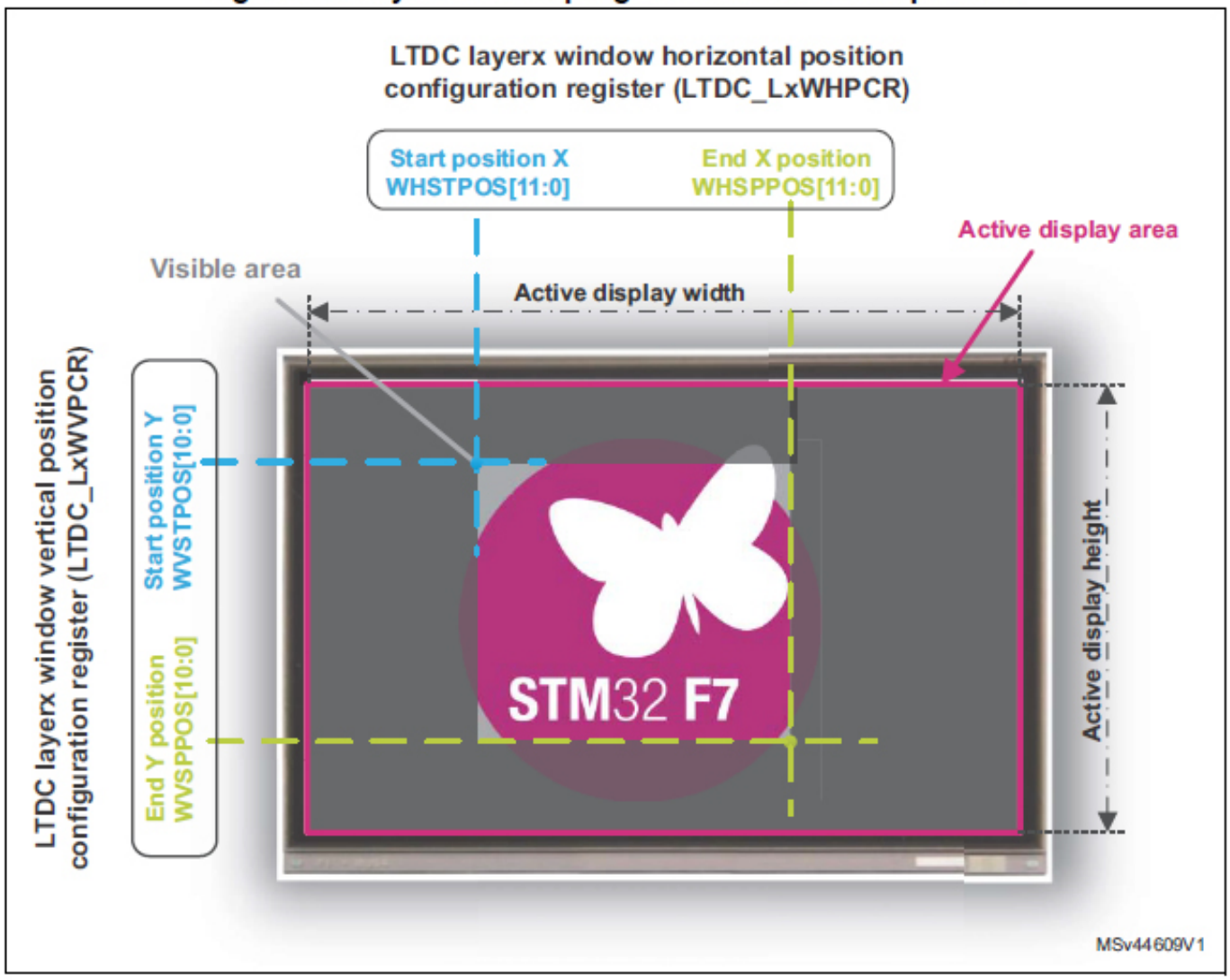
На рисунке 16 показано смешение двух слоев с фоном.

3.3.1 Гибкая конфигурация положения окна и размера

Каждый слой может быть размещен и изменен по размеру во время выполнения, и он должен находиться внутри активной области отображения. Поля и размер программируемого слоя определяют первый и последний видимый пиксель линии и первую и последнюю видимую строку в окне. Он позволяет отображать либо полное изображение (всю активную область отображения), либо только часть

кадра изображения. На рисунке 17 показано небольшое окно, в котором отображается только часть изображения, а оставшаяся область не отображается.

Figure 17. Layer window programmable size and position



1. LTDC_LxWHPCR и LTDC_LxWVPCR – это соответственно регистровые регистры конфигурации горизонтального и вертикального положения горизонтального уровня LTDC, где «x» может ссылаться на уровень 1 или уровень 2.

3.3.2 Программируемый слой: цветной буфер кадра

Каждый слой имеет выделенное настраиваемое количество строк и длину строки для цветного буфера кадра и для высоты тона.

Адрес цветовой рамки

Каждый уровень имеет начальный адрес для буферов кадров цвета, сконфигурированного в регистре LTDC_LxCFBAR.

Длина цветной рамки (размер)

Длина линии и количество параметров линий используются для остановки предварительной выборки данных из уровня FIFO в конце буфера кадра.

Длина строки (в байтах) настраивается в регистре LTDC_LxCFBLR.

Количество строк (в байтах) настраивается в регистре LTDC_LxCFBLNR.

Шаг цветной рамки

Шаг – это расстояние между началом одной строки и началом следующей строки в байтах. Он настроен в регистре LTDC_LxCFBLR.

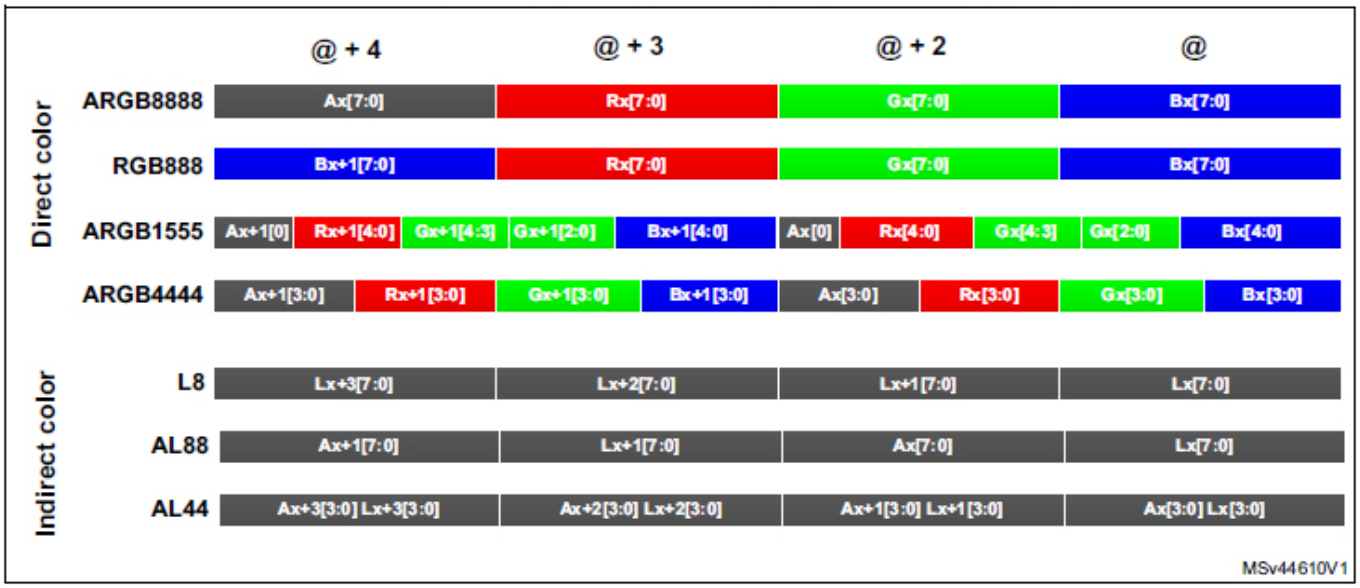
Формат ввода пикселей

Программируемый формат пикселей используется во всех данных, хранящихся в буфере кадра каждого уровня LTDC.

Для каждого слоя отдельный формат ввода пикселей может быть сконфигурирован отдельно. LTDC может быть сконфигурирован с до восьми программируемых форматов входных цветов для каждого слоя.

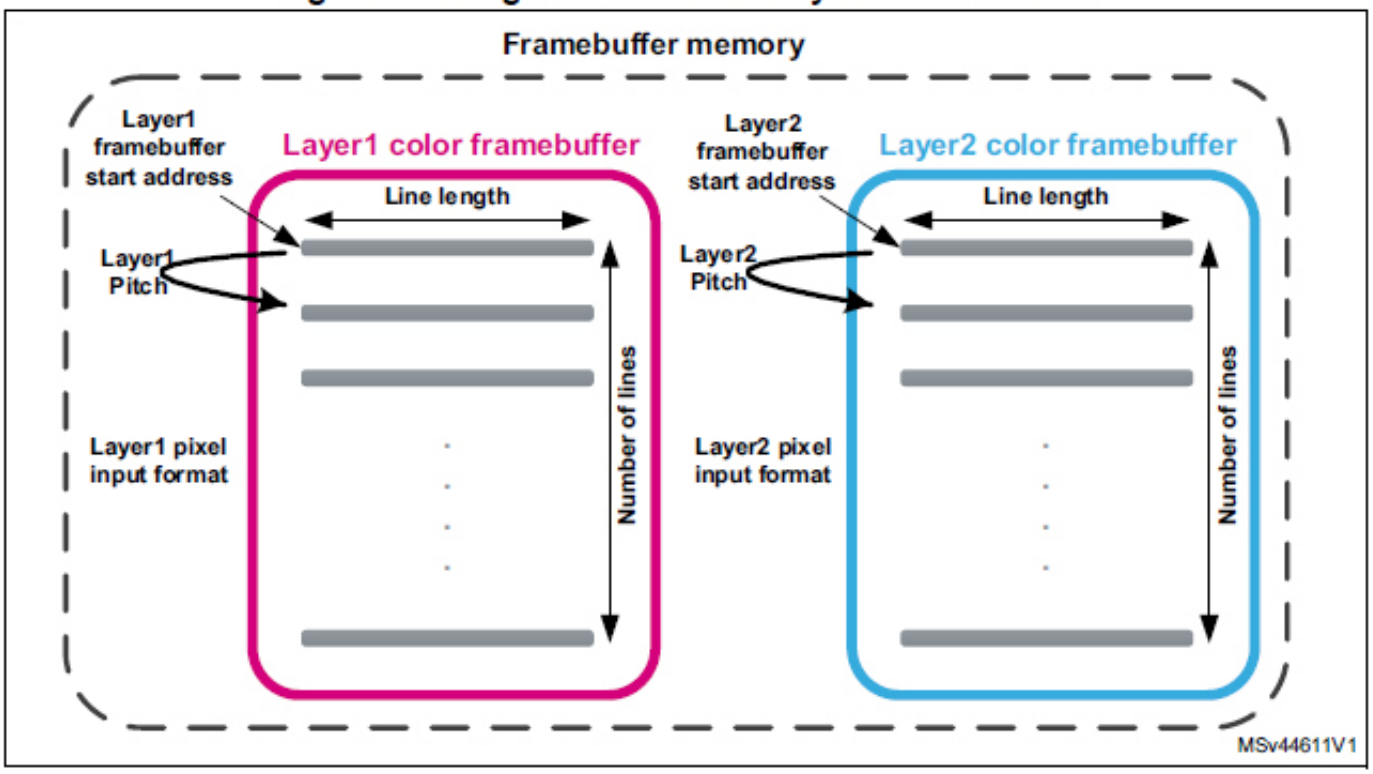
На рисунке 18 показано сопоставление данных пикселей в зависимости от выбранного формата входного цвета.

Figure 18. Pixel data mapping versus color format



На рисунке 19 приведены все настраиваемые параметры цветового буфера кадра.

Figure 19. Programmable color layer in framebuffer



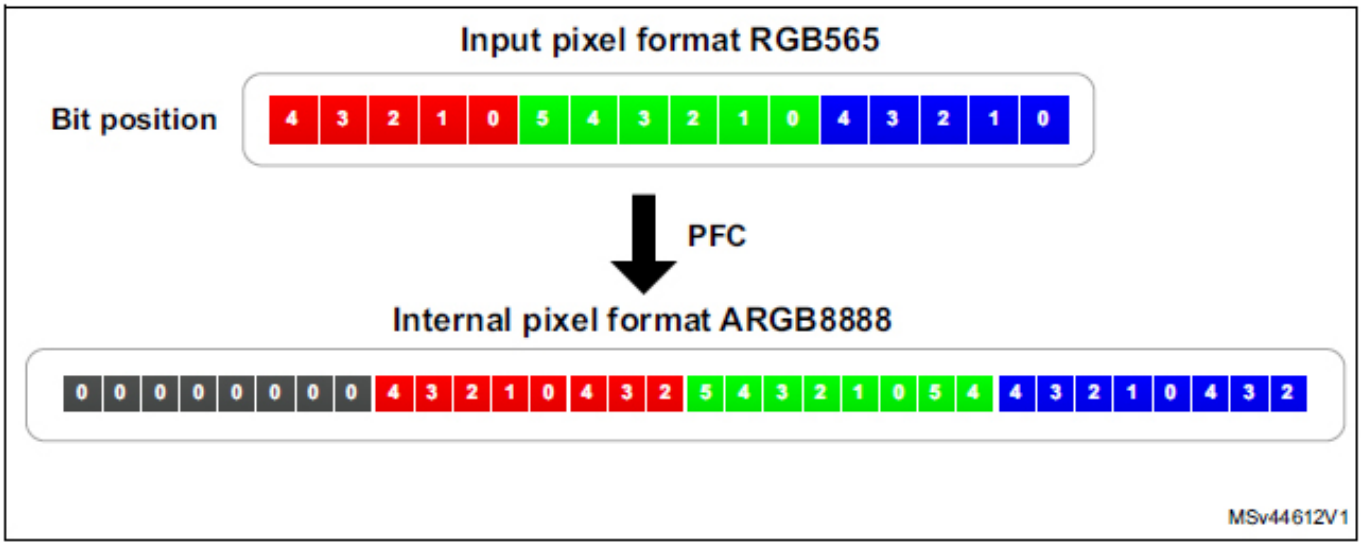
Преобразование формата пикселей (PFC)

После считывания из буфера кадра данные пикселя преобразуются из встроенного формата входного изображения во внутренний формат ARGB8888.

Компоненты с шириной менее 8 бит расширяются до 8 бит-бит.

Выбраны 8 бит MSB. На рисунке 20 показано преобразование формата входного пикселя RGB565 во внутренний формат ARGB8888.

Figure 20. Pixel format conversion from RGB565 input pixel format to the internal ARGB8888 format



Примечание. *Использование двух уровней создает ограничения пропускной способности в системе. Предпочтительно использовать только один слой и выполнять композицию с помощью Chrom-Art Accelerator® во время вычисления буфера кадра (см. Раздел 4.2.2 «Проверка совместимости дисплеев с учетом требований к пропускной способности памяти»).*

3.4 Прерывания

Внешняя периферия LTDC поддерживает два глобальных прерывания:

- Глобальное прерывание LTDC.
- Прерывание глобальной ошибки LTDC.

Каждое глобальное прерывание подключается к двум прерываниям LTDC (логически несвязанным), которые могут маскироваться отдельно через определенный регистр. В таблице 7 приведены все соответствующие прерывания и все конкретные случаи, когда генерируется каждое прерывание.

Таблица 7. Перечень LTDC прерываний

Связанное прерывание NVIC	Событие прерывания	Бит флага события (регистр LTDC_ISR)	Установить бит (регистр LTDC_IER)	Очистить бит (регистр LTDC_ICR)	Описание
LTDC GLOBAL INTER–RUPT	Линия	LIF	LIE	CLIF	Генерируется, когда отрисована линия на экране
	Переза-грузка регистра	RRIF	RRIE	CRRIF	Генерируется, когда происходит перезагрузка тени

LTDC GLOBAL ERROR INTER-RUPT	FIFO прерыва-ние запу-ска	FUIF	FUIE	CFUIF	Генерируется, когда пиксель запрашивается, а FIFO пуст
	Ошибка передачи	TERRIF	TERRIE	CTERRIF	Генерируетсяпри возникновении ошибки шины

1. Прерывание опустошения FIFO полезно для определения совместимости размеров дисплея (см. Раздел 4.2.2: Проверка совместимости дисплеев с учетом требований к пропускной способности памяти).

3.5 Маломощные режимы

Состояние мощности STM32 оказывает прямое влияние на периферию LTDC. В режиме ожидания режим LTDC не изменяется, и он продолжает показывать графические данные на экране. В режиме ожидания и остановки режим LTDC отключается, и через его параллельный интерфейс нет выхода. Выход из режима ожидания должен сопровождаться реконфигурацией LTDC.

В режиме ожидания можно управлять дисплеем в режиме ожидания, в то время как CPU остановлен благодаря интеллектуальной архитектуре, встроенной в микроконтроллеры STM32, которая позволяет включить все периферийные устройства даже в спящем режиме. Эта характеристика подходит для ношения в случаях, когда требуется потребление малой мощности.

LTDC в качестве ведущего устройства AHB может продолжить извлечение данных из FMC_SDRAM или Quad-SPI (когда используется режим отображения памяти) даже после ввода MCU в режиме SLEEP. Может быть сгенерировано прерывание линии или прерывание перезагрузки регистра, чтобы разбудить STM32 при достижении определенной линии на экране или при перезагрузке тени.

Более подробная информация о снижении энергопотребления приведена в разделе 5.

В таблице 8 суммируется состояние LTDC в сравнении с режимами малой мощности STM32.

Таблица 8. Внешнее состояние LTDC в режимах малой мощности STM32

Режим	Описание
Run	Активный
Sleep	Активный. Периферийные прерывания вызывают выход из режима сна
Stop	Замороженный. Содержимое периферийных регистров сохраняется
Standby	Выключенный. После выхода из режима ожидания периферийное устройство должно быть повторно инициализировано

4 Создание графического приложения с помощью LTDC

В этом разделе показаны различные этапы, необходимые до и во время разработки графического приложения с использованием LTDC. Сначала пользователь должен определить требования к графическому приложению, а затем проверить, соответствует ли требуемый размер дисплея аппаратной конфигурации. Во время этапа проверки совместимости графического приложения пользователь может использовать существующие эталонные платы STM32, описанные в таблице 17, для оценки его конфигурации аппаратного и программного обеспечения.

4.1 Определение требований графического приложения

Определение потребностей графического приложения – это важный шаг для начала. Некоторые из наиболее важных параметров, которые должны быть определены до начала создания графического приложения, – это разрешение экрана, глубина цвета, а также характер отображаемых данных (статические изображения, текст или анимация).

После определения основных параметров, указанных выше, пользователь должен определить графическую аппаратную архитектуру приложения, а также необходимые аппаратные ресурсы. Пользователь должен выбрать наиболее подходящий пакет STM32 (см. Таблицу 13) в соответствии со следующими параметрами:

- Если для буфера кадра требуется внешняя память
- Ширина шины памяти внешнего буфера кадра
- Интерфейс LTDC: RGB565, RGB666 или RGB888 в зависимости от модуля дисплея
- Если для хранения графических примитивов (QSPI или FMC_NOR) требуется внешняя память,

4.2 Проверка размера дисплея и совместимости цвета с конфигурацией оборудования

При запуске разработки графических приложений с использованием микроконтроллера STM32 пользователь обычно имеет определенный размер и глубину цвета. Ключевой вопрос, который должен ответить пользователь перед продолжением разработки, заключается в том, что такой размер дисплея и глубина цвета соответствуют конкретной конфигурации оборудования ?.

Чтобы ответить на этот вопрос, пользователь должен выполнить следующие шаги:

1. Определите необходимый размер буфера кадра и его местоположение.
2. Проверьте совместимость дисплея с требованиями к пропускной способности памяти буферов кадров.
3. Проверьте совместимость интерфейса панели дисплея с LTDC.

4.2.1 Требования к размеру и размеру памяти Framebuffer

Определение размера памяти буфера кадра и его местоположения является ключевым параметром для проверки совместимости дисплеев.

Объем памяти, необходимый в ОЗУ для поддержки буфера кадра, должен быть смежным и с минимальным размером, равным:

Framebuffer size = количество пикселей x бит на пиксель

Как показано в приведенной выше формуле, требуемый размер буфера кадра зависит от разрешения дисплея и от его глубины цвета.

Нет необходимости, чтобы глубина цвета рамки (bpp) была такой же, как глубина цвета дисплея. Например, дисплей RGB888 можно управлять с помощью буфера кадра RGB565.

Примечание. *Необходимый размер буфера кадра удваивается для конфигурации с двумя буферами кадрами. Обычно используется конфигурация двойного буфера, где один графический буфер используется для хранения текущего изображения, а второй буфер используется для подготовки следующего изображения.*

В таблице 9 показан размер буфера кадра, необходимый для стандартных разрешений экрана с различными форматами пикселей.

Table 9. Framebuffer size for different screen resolutions

Screen resolution	Number of pixels	Framebuffer size (Kbyte)(1)			
		8 bpp	16 bpp	24 bpp	32 bpp
QVGA (320 x 240)	76800	75	150	225	300
Custom (480 x 272)(2)	130560	128	255	383	510
HVGA(480 x 320)	153600	150	300	450	600
VGA (640 x 480)	307200	300	600	900	1200
WVGA(800 x 480)	384000	375	750	1125	1500
SVGA (800 x 600)	480000	469	938	1407	1875
XGA (1024 x 768)	786432	768	1536	2304	3072
HD (1280 x 720)	921600	900	1800	2700	3600

1. Требуемый размер буфера кадра удваивается для конфигурации с двойной рамкой.

2. Пример пользовательского дисплея 480 x 272 – это ROCKTECH, встроенный в набор для обнаружения STM32F746 (32F746GDISCOVERY).

Расположение буфера кадра

В зависимости от требуемого размера буфера кадра он может быть расположен либо во внутренней SRAM, либо во внешнем SRAM / SDRAM.

Если внутренней памяти недостаточно для буфера кадра, пользователь должен использовать внешний SDRAM / SRAM, подключенный к FMC.

Следовательно, требуемый размер буфера кадра будет определять, требуется ли использование внешней памяти или нет. Требуемый размер буфера кадра зависит от размера экрана и глубины цвета.

Расположение буфера кадра во внутренней SRAM

В зависимости от размера буфера кадра его можно разместить либо во внутренней SRAM, либо в SRAM или SDRAM.

Использование внутренней SRAM в качестве буфера кадра обеспечивает максимальную производительность и позволяет избежать любых проблем с ограничением полосы пропускания для LTDC.

Использование внутренней SRAM вместо внешней SRAM или SDRAM имеет много преимуществ:

• Обеспечивает более высокую пропускную способность (0 доступ к состоянию ожидания).

• Сокращает количество требуемых контактов и сложность конструкции печатной платы.

• Уменьшает спецификацию, следовательно, стоимость, поскольку не требуется внешняя память.

Единственным ограничением при использовании внутреннего SRAM является его ограниченный размер (сотни Килобитов). Если размер буфера кадра превышает доступную память, следует использовать внешний SDRAM или SRAM (управляемый FMC-интерфейсом). Однако при работе с внешними запоминающими устройствами пользователь должен быть осторожным, чтобы избежать ограничения полосы пропускания. Более подробную информацию см. В разделе 4.5: Оптимизация графической производительности.

Примечание. *Таблица уменьшения цвета CLUT может использоваться для уменьшения требуемого размера буфера кадра. (Подробнее см. Соответствующее руководство по эксплуатации STM32 MCU).*

4.2.2 Проверка совместимости дисплеев с учетом требований к пропускной способности памяти

Объем этого раздела объясняет, как проверить совместимость дисплея с учетом пропускной способности памяти буфера кадра. Для этого в этом разделе описываются некоторые важные аспекты полосы пропускания и объясняется, как определить требуемую полосу пропускания для часов пикселей и LTDC. Наконец, в этом разделе показан простой метод, который позволяет заключить, совместим ли желаемый размер дисплея с конкретной аппаратной конфигурацией.

Аспекты пропускной способности памяти буфера кадра

После того, как местоположение буфера кадра зафиксировано (во внутренней или внешней памяти), пользователь должен проверить, может ли его пропускная способность поддерживать аппаратную конфигурацию.

Чтобы проверить, может ли пропускная способность памяти поддерживать требуемую полосу пропускания LTDC, пользователь должен учитывать любые другие одновременные обращения к памяти.

В общем, буфер кадра с малым размером, расположенный во внутренней ОЗУ, не требует высокой пропускной способности. Это связано с тем, что малогабаритный буфер кадра означает низкие пиксельные часы, следовательно, требуется низкая пропускная способность LTDC.

Более сложный случай использования для анализа – это когда буфер кадра находится во внешней памяти (SDRAM или SRAM).

Совместимость памяти шины памяти Framebuffer • LTDC, DMA2D и CPU master

• LTDC, DMA2D и процессоры

В типичном графическом приложении, где внешняя память SDRAM или SRAM используется в качестве буфера кадра, два или три основных мастера AHB одновременно используют одну и ту же память.

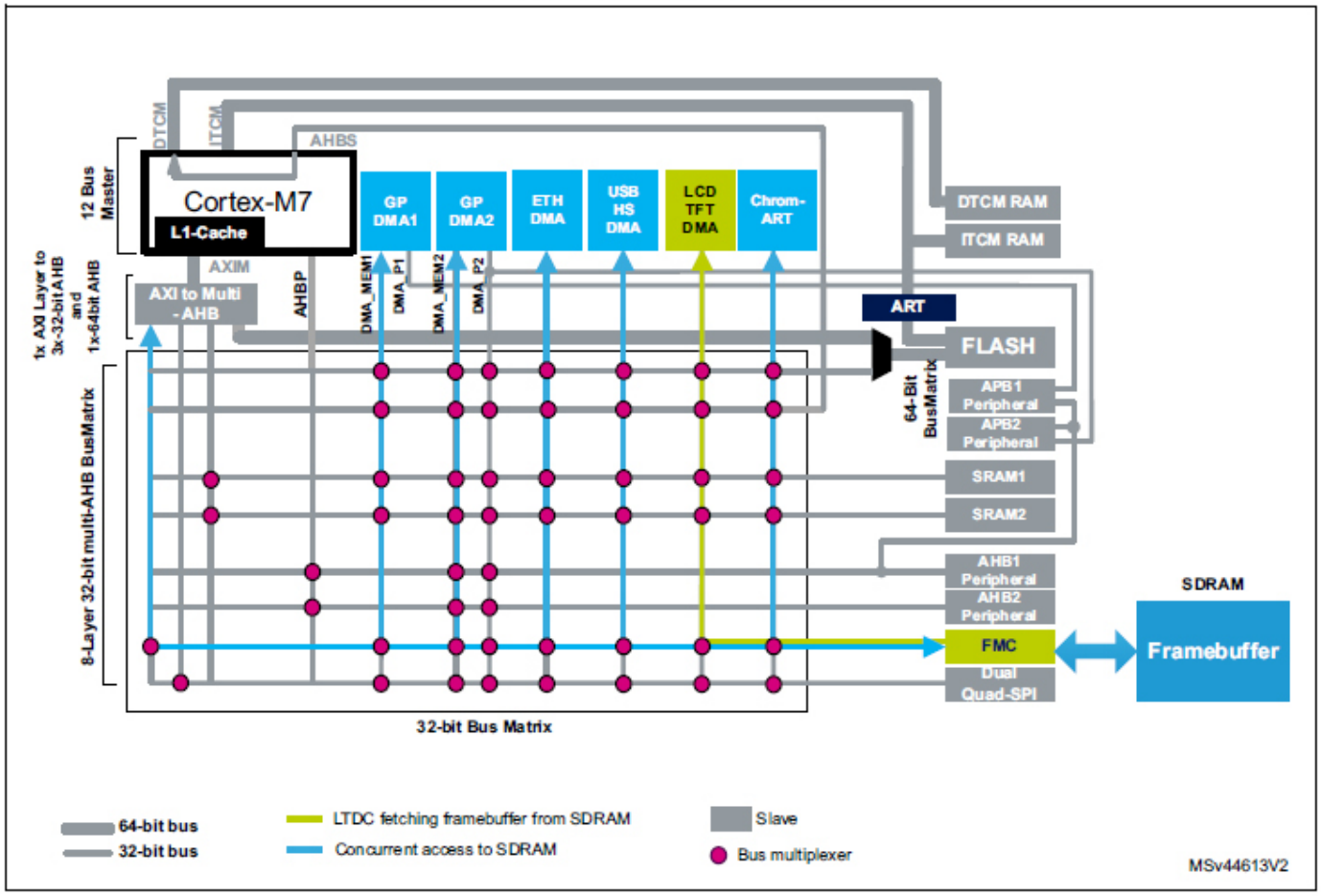
DMA2D (или ЦП) обновляет следующее изображение, которое будет отображаться, когда LTDC извлекает и отображает фактическое изображение. Нагрузка шины памяти зависит в основном от необходимой полосы пропускания LTDC.

• Другие мастера AHB

Общеизвестно, что внешняя память SDRAM или SRAM используется другими мастерами, а не только теми, которые используются для графики. Этот параллелизм приводит к большой нагрузке на шины и может влиять на графические характеристики.

На рисунке 21 показаны все мастера AHB с одновременным доступом к SDRAM.

Figure 21. AHB masters concurrent access to SDRAM



Ширина шины памяти SDRAM / SRAM

При размещении буфера кадра во внешнем SDRAM / SRAM пользователь должен учитывать, что рабочая частота внешней памяти составляет около половины или третьей частоты системы. Именно по этой причине пропускную способность памяти следует рассматривать как узкое место всей графической системы.

Одним из необходимых параметров проверки совместимости с дисплеем является ширина шины памяти. Для SDRAM пользователь может использовать 8-битную, 16-битную или 32-битную конфигурацию.

Как было сказано ранее, наиболее сложным для анализа является использование, когда буфер кадра помещается во внешнюю память:

Одновременный доступ мастеров к одной и той же внешней памяти приводит к большей задержке и влияет на ее пропускную способность.

Определение тактовых импульсов пикселей и требуемой полосы пропускания LTDC

Вычисление тактовых импульсов пикселей

Частота пикселов является ключевым параметром для проверки совместимости размера экрана с конкретной аппаратной конфигурацией.

Чтобы получить типичные пиксельные часы дисплея, обратитесь к таблице данных дисплея. Вычисленные пиксельные часы должны соответствовать спецификациям дисплея.

Частота пикселов для конкретной частоты обновления, рассчитанные по следующей формуле:

LCD_CLK (МГц) = (общий размер экрана) x (частота обновления), где общий размер экрана = (общая ширина) x (общая высота).

Потребляемая полоса LTDC

Требуемая пропускная способность LTDC зависит в основном от трех факторов:

- Количество используемых слоев LTDC
- Глубина цвета слоя LTDC
- Часы на пикселях (в зависимости от разрешения панели дисплея и частоты обновления)

Максимальная требуемая полоса пропускания может быть рассчитана, как описано ниже:

- Если используется только один слой LTDC – Требуемая полоса пропускания LTDC = LCD_CLK x BppL1
- Если используются два слоя LTDC – Требуемая полоса пропускания LTDC = LCD_CLK x (BppL1 + BppL2)

Где BppL1 и BppL2 являются соответственно глубиной цвета для LTDC Layer1 и Layer2.

Требуемая полоса пропускания LTDC не должна превышать доступную полосу пропускания памяти, в противном случае будут возникать проблемы с отображением, и будет установлен флаг недогрузки FIFO (если включено прерывание опусстошения FIFO).

Примечание. *Если память, используемая для хранения буфера кадра, также используется для других целей приложения, это может повлиять на графические характеристики системы.*

Проверьте, соответствует ли используемое разрешение дисплея аппаратной конфигурации

Общий метод проверки того, совместим ли размер дисплея с определенной глубиной цвета с полосой пропускания памяти:

1. Скомпонуйте пиксельные часы в соответствии с размером экрана или извлеките его из таблицы данных дисплея.

2. Убедитесь, что часы пикселя дисплея не превышают максимальные опорные часы пикселя, указанные в таблице 10 или таблице 11. Пользователь должен использовать следующие параметры для извлечения из таблицы 10 или таблицы 11

максимальных поддерживаемых часов пикселя, соответствующих используемому оборудованию конфигурация:

a) Количество используемых слоев LTDC.

b) Скорость памяти HCLK используемой системы и скорость памяти буфера кадра.

c) Ширина шины памяти внешнего буфера кадра.

d) Количество мастеров AHB, одновременно получающих доступ к внешней памяти буфера кадра.

3. Пользователь должен выполнить некоторые тесты, чтобы подтвердить совместимость оборудования с желаемым размером дисплея и глубиной цвета. Для этого пользователь должен следить за флагом прерывания LTF FIFO в регистре LTDC_ISR. Если флаг прерывания FIFO всегда сбрасывается, пользователь подтверждает, что желаемый размер дисплея совместим с конфигурацией оборудования. Если FIFO недоиспользуется установлен флаг, пользователь должен проверить следующие пункты

a) Убедитесь, что он извлек из таблицы 10 или таблицы 11 правильные максимальные часы пикселя, соответствующие правильному оборудованию (например, пользователь работает с 16-разрядным

SDRAM, но он извлек пиксельные часы, соответствующие 32-разрядной SDRAM, что было бы ошибкой).

b) Ширина линии цветного буфера кадра не выровнена на 64 байта (см. раздел 4.5.2: Оптимизация загрузки буфера кадра LTDC из внешних запоминающих устройств (SDRAM или SRAM).

c) Для серии STM32F7 MPU настроен неправильно, чтобы избежать возможности просмотра CATRIX-M7 в SDRAM (см. раздел 4.6: Специальные рекомендации для Cortex®-M7 (серия STM32F7)).

d) Если прерывание FIFO все еще установлено, потому что существует более двух AHB-мастеров одновременного доступа во внешнюю память, пользователь должен уменьшить пропускную способность памяти, используя приведенные ниже рекомендации:

– использовать только один слой LTDC

– используйте максимально возможную ширину шины памяти (используйте 32-битную вместо 16-битной или 8-битной SDRAM / SRAM)

– обновлять содержимое буфера кадра в течение периода гашения, когда LTDC не выбирает

– используйте максимально возможные системные часы HCLK и максимальную скорость памяти (FMC_SDRAM / FMC_SRAM)

– уменьшить глубину цвета изображений (BPP)

– Более подробную информацию об оптимизации пропускной способности памяти см. В разделе 4.5: Оптимизация графической производительности.

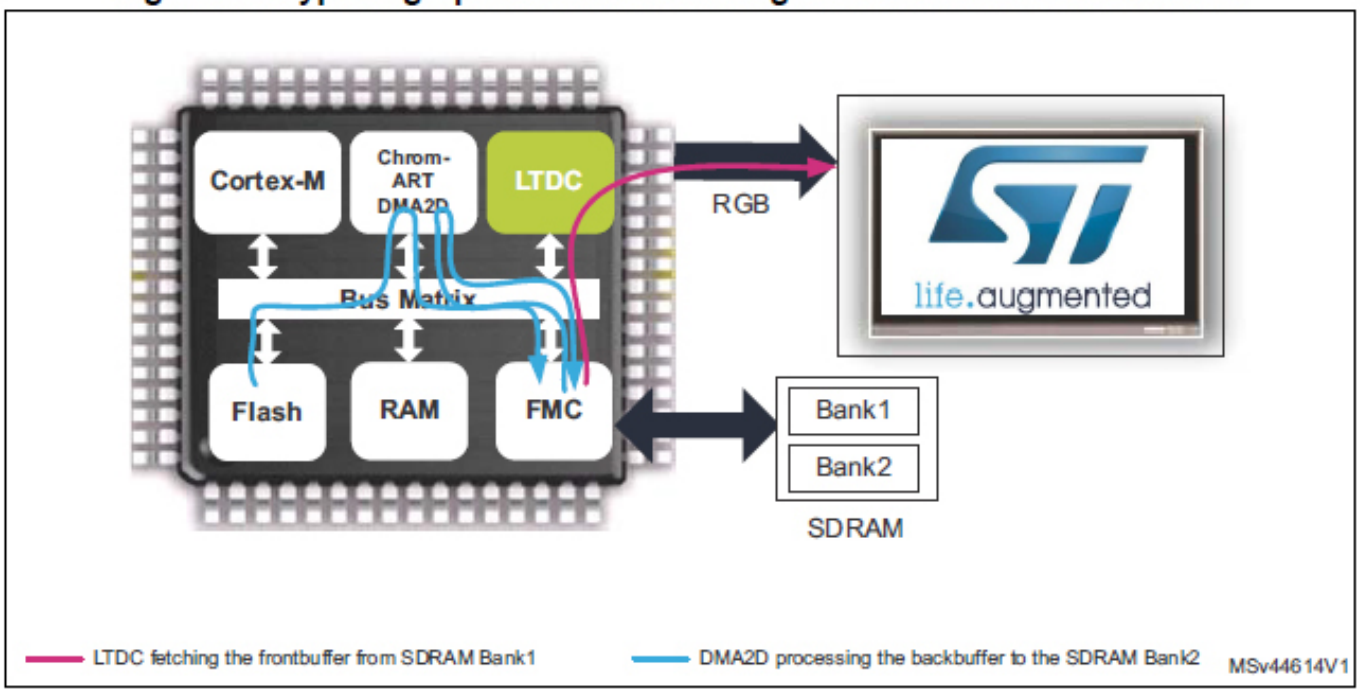
Примечание. *Чтобы оценить графические возможности STM32 в конкретной аппаратной конфигурации, пользователь может использовать платы STM32, описанные в таблице 17: эталонные платы STM32 с встраиванием LTDC и с встроенной жидкокристаллической панелью TFT.*

На рисунке 22 показана типичная графическая аппаратная конфигурация, в которой внешняя SDRAM подключена к FMC, которая используется для буфера

кадров. Ширина полосы памяти SDRAM зависит от ширины шины и рабочего времени.

Ширина шины SDRAM может быть 32-разрядной, 16-разрядной или 8-разрядной, тогда как рабочие часы зависят от системных часов HCLK и сконфигурированного предделителя (HCLK / 2 или HCLK / 3).

Figure 22. Typical graphic hardware configuration with external SDRAM



В таблице 10 перечислены максимальные поддерживаемые пиксельные часы на уровне системы для линии STM32F4x9, а в таблице 11 приведены максимальные поддерживаемые пиксельные часы на системном уровне для строк STM32F7x6, STM32F7x7, STM32F7x8 и STM32F7x9 в следующих условиях:

- Для линии STM32F4x9 системные часы HCLK работают на частоте 180 МГц, а SDRAM составляет 90 МГц.
- Для линий STM32F7x6, STM32F7x7, STM32F7x8 и STM32F7x9 системные часы HCLK работают на частоте 200 МГц, а SDRAM составляет 100 МГц.
- Один или два AHB контролируют одновременный доступ к SDRAM (LTDC или LTDC + DMA2D).

Ширина шины SDRAM составляет 16 бит или 32 бит.

- Используется только один слой LTDC или два слоя.
- Глубина цвета слоя LTDC составляет 8 bpp, 16 bpp, 24 bpp или 32 bpp.

Table 10. STM32F4x9 with HCLK @ 180 MHz and SDRAM @ 90 MHz maximal supported pixel clock versus LTDC configuration and SDRAM bus width

Used LTDC layers	Color depth (bpp)	Maximum pixel clock (MHz)			
		LTDC		LTDC + DMA2D	
		SDRAM 16-bit	SDRAM 32-bit	SDRAM 16-bit	SDRAM 32-bit
1 layer	32	38	67	22	35
	24	51	83	30	47
	16	76	83	45	70
	8	83	83	83	83
2 layers	32/32	19	33	NA	18
	32/24	22	38	13	21
	32/16	25	44	15	25
	32/8	30	53	19	30
	24/24	26	44	15	24
	24/16	31	53	18	30
	24/8	38	67	23	38
	16/16	39	67	22	37
	16/8	51	83	31	50
	8/8	78	83	46	74

Table 11. STM32F7x6, STM32F7x7, STM32F7x8 and STM32F7x9 with HCLK @ 200 MHz and SDRAM@ 100 MHz maximal supported pixel clock versus LTDC configuration and SDRAM bus width

Used LTDC layers	Color depth (bpp)	Maximum pixel clock (MHz)			
		LTDC		LTDC + DMA2D	
		SDRAM 16-bit	SDRAM 32-bit	SDRAM 16-bit	SDRAM 32-bit
1 layer	32	42	74	25	39
	24	56	83	34	52
	16	83	83	51	78
	8	83	83	83	83
2 layers	32/32	21	37	12	20
	32/24	24	42	14	23
	32/16	28	49	17	28
	32/8	34	59	21	34
	24/24	29	49	17	27
	24/16	34	59	20	33
	24/8	42	74	26	42
	16/16	43	74	25	41
	16/8	57	83	34	56
	8/8	83	83	51	82

Примечание. Уменьшение системных часов (HCLK, затем LTDC) приводит к ухудшению графических характеристик.

Пример поддерживаемых разрешений дисплея для линии STM32F4x9 и серии STM32F7

В таблице 12 приведен пример некоторых стандартных и пользовательских размеров дисплея, поддерживаемых линейкой STM32F4x9 и STM32F7, в следующих условиях:

• Для линии STM32F4x9 системные часы HCLK работают на частоте 180 МГц, а SDRAM составляет 90 МГц.

• Для линий STM32F7x6, STM32F7x7, STM32F7x8 и STM32F7x9 системные часы HCLK работают на частоте 200 МГц, а SDRAM составляет 100 МГц.

• Используется только один слой LTDC.

• Два AHB контролируют одновременный доступ к SDRAM (LTDC + DMA2D).

Table 12. Example of supported display resolutions in specific STM32 hardware configurations

<table>
<tr><th colspan="4">Display characteristics</th><th colspan="2">STM32's LTDC configuration</th></tr>
<tr><th rowspan="2">Resolution</th><th rowspan="2">Refresh rate (Hz)</th><th rowspan="2">Pixel clock (MHz)</th><th rowspan="2">Display standard</th><th colspan="2">Color depth</th></tr>
<tr><th>SDRAM 16-bit</th><th>SDRAM 32-bit</th></tr>
<tr><td>320 x 240 (QVGA)</td><td rowspan="8">60</td><td>5.6</td><td rowspan="2">Custom</td><td colspan="2" rowspan="2">Up to 32 bpp</td></tr>
<tr><td>480 x 272</td><td>9.5</td></tr>
<tr><td>640 x 480 (VGA)</td><td>25.175</td><td>Industry standard</td><td>Up to 24 bpp</td><td>Up to 32 bpp</td></tr>
<tr><td>800 x 600 (SVGA)</td><td>40.000</td><td rowspan="3">VESA guidelines(1)</td><td>Up to 16 bpp</td><td>Up to 24 bpp</td></tr>
<tr><td>1024 x 768 (XGA)</td><td>65</td><td rowspan="3">8 bpp</td><td rowspan="2">Up to 16 bpp</td></tr>
<tr><td>1280 x 768</td><td>68.250</td><td>CVT R.B(2)</td></tr>
<tr><td>1280 x 720 (HD)</td><td>74.25</td><td>CEA(3)</td><td rowspan="2">Up to 16 bpp(4)</td></tr>
<tr><td>1920 x1080</td><td>30</td><td>74.25</td><td>CEA(3)</td></tr>
</table>

1. VESA (ассоциация стандартов видеоэлектроники) – это организация технических стандартов для стандартов отображения компьютеров, обеспечивающих стандарты времени монитора (DMT).

2. CVT R.B: скоординированные видеопотоки сократили стандарт гашения VESA.

3. CEA = ассоциация бытовой электроники.

4. До 8 бит / с для микроконтроллеров STM32F4x9

4.2.3. Проверьте совместимость интерфейса панели дисплея с LTDC

Пользователь должен выбрать ЖК-панель в зависимости от потребностей приложения. Двумя основными факторами, которые следует учитывать при выборе

ЖК-панели, являются разрешение и глубина цвета. Эти два фактора напрямую влияют на следующие параметры:

- Требуемый номер GPIO.
- Размер и расположение буфер кадра.
- Часы пикселей на дисплее.

При выборе панели дисплея пользователь должен:

- Убедитесь, что интерфейс дисплея совместим с LTDC (параллельный RGB с управляющими сигналами).
- Проверьте, могут ли управляющие сигналы управлять LTDC (иногда требуются дополнительные GPIO).
- Убедитесь, что уровни сигнала дисплея соответствуют уровням сигнала интерфейса LTDC (VDD от 1,8 В до 3,6 В).
- Убедитесь, что часы пикселя дисплея поддерживаются максимальным пиксельным тактовым сигналом LTDC, определенным в соответствующем техническом описании микроконтроллера STM32.
- Убедитесь, что параметры таймингов дисплея поддерживаются таймингом LTDC (см. Таблицу 6: регистры времени LTDC).
- Убедитесь, что размер дисплея и глубина цвета поддерживаются LTDC (см. Раздел 4.2.2. Проверка совместимости дисплея с учетом требований к пропускной способности памяти).

4.3 Руководство по выбору пакета STM32

На этом этапе разработки графического приложения пользователь уже определил требования приложения с точки зрения GPIO:

- Требуется ли внешняя память и какая ширина шины.
- Какую конфигурацию LTDC использовать: RGB565, RGB666 или RGB888.

При выборе пакета STM32 пользователь должен учитывать доступность интерфейсов RGB и требования к приложениям с точки зрения количества GPIO. Пользователь должен обратиться к соответствующему каталогу STM32, чтобы получить доступные пакеты с GPIO.

Простой способ проверить, соответствует ли пакет STM32, в котором заинтересовался пользователь, потребности приложения в терминах номера GPIO, заключается в использовании STM32CubeMX (вкладка распиновки).

В таблице 13 представлены доступные пакеты и RGB-интерфейс MCU STM32, встраивающих LTDC.

Table 13. STM32 packages with LTDC peripheral versus RGB interface availability(1)

Product	LQFP100	TFBGA100	LQFP144	UFBGA169	UFBGA176	LQFP176	LQFP208	TFBGA216	WLCSP143	WLCSP168	WLCSP180
STM32F429 SRM32F439	18	NA	18	24	24	24	24	24	18	NA	NA
STM32F469 STM32F479(2)	18	NA	18	24	24	24	24	24	NA	24	NA
STM32F7x6	18	18	24	NA	24	24	24	24	24	NA	NA
STM32F7x7	18	NA	24	NA	24	24	24	24	NA	NA	NA
STM32F7x8(2)	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	24
STM32F7x9(2)	NA	NA	NA	NA	NA	24	24	24	NA	NA	24

1. Gray cells with "NA" = the package is not available for that specific product.
Cells with "18" value = only RGB565 and RGB666 parallel outputs are supported.
Cells with "24" value = all of RGB565, RGB666 and RGB888 outputs are supported.
2. The integrated MIPI-DSI controller allows easier PCB design with fewer pins, for more details on STM32's MIPI-DSI host refer to application note AN4860.

4.4 Синхронизация LTDC с DMA2D и CPU

4.4.1 Использование DMA2D

DMA2D является мастером на матрице шины AHB, выполняющей графические передачи данных между памятью. Рекомендуется использовать DMA2D для выгрузки процессора.

DMA2D реализует четыре основные задачи:

- Заполните прямоугольную форму уникальным цветом.
- Скопируйте рамку или прямоугольную часть кадра из памяти в другую.
- Преобразование формата пикселя кадра или прямоугольной части кадра при передаче его из одной памяти в другую память.
- Смешайте два изображения разных размеров и формата пикселей и сохраните полученное изображение в одной результирующей памяти.

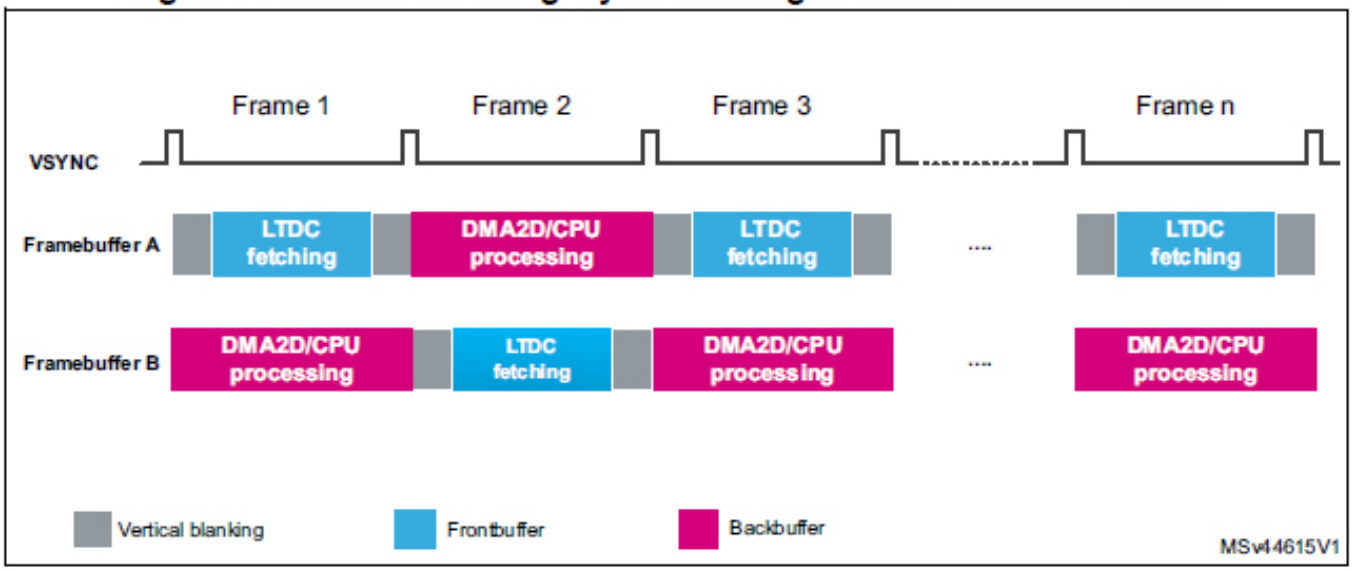
4.4.2 Синхронизация LTDC и DMA2D / CPU

Когда используется только один буфер кадра, существует риск того, что вычисление буфер кадра будет отображаться на экране. Несколько методов буферизации, таких как двойная буферизация, обычно используются, чтобы не отображать расчет буфер кадра на экране.

Даже при использовании метода двойной буферизации эффект разрыва может появиться из-за несинхронизации между LTDC и обновлением буфер кадра (либо CPU, либо DMA2D). Способом решения этой проблемы является использование сигнала VSYNC для синхронизации рабочего процесса этих двух мастеров (LTDC и CPU или DMA2D).

LTDC извлекает графические данные из буфера (называемого frontbuffer), в то время как DMA2D готовит следующий кадр в другом буфере (называемый backbuffer). Период VSYNC указывает конец фактического отображения кадра и что два буфера должны быть перевернуты

Figure 23. Double buffering: synchronizing LTDC with DMA2D or CPU



LTDC предоставляет различные варианты синхронизации этого рабочего процесса:

- Запрограммируйте прерывание линии со значением последней строки экрана, обработчик прерываний должен перевернуть буферы кадра и начать следующий расчет буфера кадра.
- Запрограммируйте регистр повторной загрузки тени (LTDC_SRCR) на перезагрузку вертикального гашения, чтобы изменить адрес буфера кадров LTDC в периоде VSYNC и опрос на бит VSYNC регистра LTDC_CDSR, чтобы разблокировать DMA2D.

4.5 Оптимизация графической производительности

Как указывалось ранее в этом документе, пропускная способность памяти буфера кадра является самым важным параметром для графического приложения. В этом разделе приведены некоторые рекомендации по оптимизации графических характеристик на основе оптимизации полосы пропускания памяти буфера кадра.

4.5.1 Распределение памяти

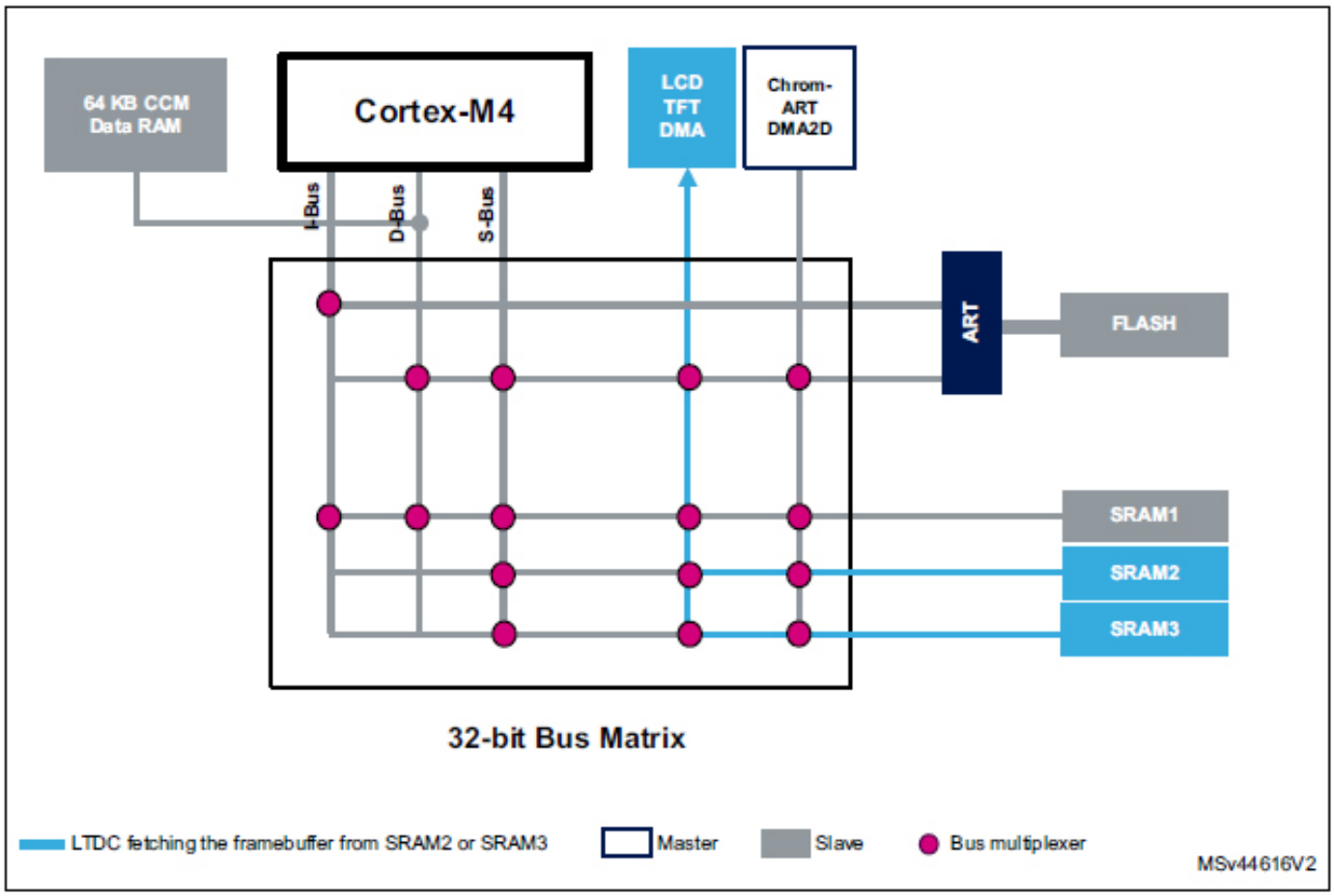
Интеллектуальная архитектура MCU STM32 позволяет значительно повысить производительность системы при использовании внутренней памяти SRAM, разделенной на два или более ведомых.

Разделение ведомых памяти между мастерами помогает уменьшить конкуренцию между ними, когда они одновременно получают один и тот же SRAM.

Это действие также создает дополнительную пропускную способность системной шины.

Как показано в примере, показанном на рисунке 24, SRAM2 и SRAM3 предназначены для графики для буфера кадра, в то время как SRAM1 используется ЦП.

Рисунок 24. Пример использования преимуществ ведомых устройств памяти, разделенных на MCU STM32F4x9



4.5.2 Оптимизация загрузки фреймбуффов LTDC из внешних запоминающих устройств (SDRAM или SRAM)

Еще одно соображение, связанное с SDRAM / SRAM, заключается в размещении буфера кадра и размера данных длины линии. Поскольку матрица AHB Bus запрещает доступ к пакету памяти, который пересекает границу одного Kilobyte, и поскольку LTDC выполняет пакетное считывание 64 байтов, размещение содержимого буфера кадра в адресе на краю одного килобайта разбивает пакет, считанный на один доступ, которые могут сильно повлиять на графические характеристики.

Такая же проблема может возникнуть, когда размер данных одной строки пикселей не кратен 64 байтам. В этих условиях и после целого ряда доступов считываемый пакет LTDC пересекает границу Kilobyte, которая разбивает пакетный запрос на одноразовый доступ.

Как следствие, когда LTDC не генерирует пакет, каждый доступ прерывается процессором или другим основным доступом (Chrom-Art Accelerator®, Ethernet или другим). Эти прерывания сильно сокращают пропускную способность LTDC в памяти с высокой задержкой, например, внешнюю SDRAM, что приводит к недоработку.

Чтобы решить проблему, описанную выше, пользователь может выбрать глубину цвета, которая не приведет к описанной проблеме, или использовать один из двух следующих способов:

• Уменьшите ширину окна и ширину линии рамки.

• Добавьте несколько фиктивных байтов в конце каждой строки пикселей, чтобы соответствовать самой близкой ширине линии кадра, кратной 64 байтам.

Пример: 480 x 272 дисплей с 24 bpp

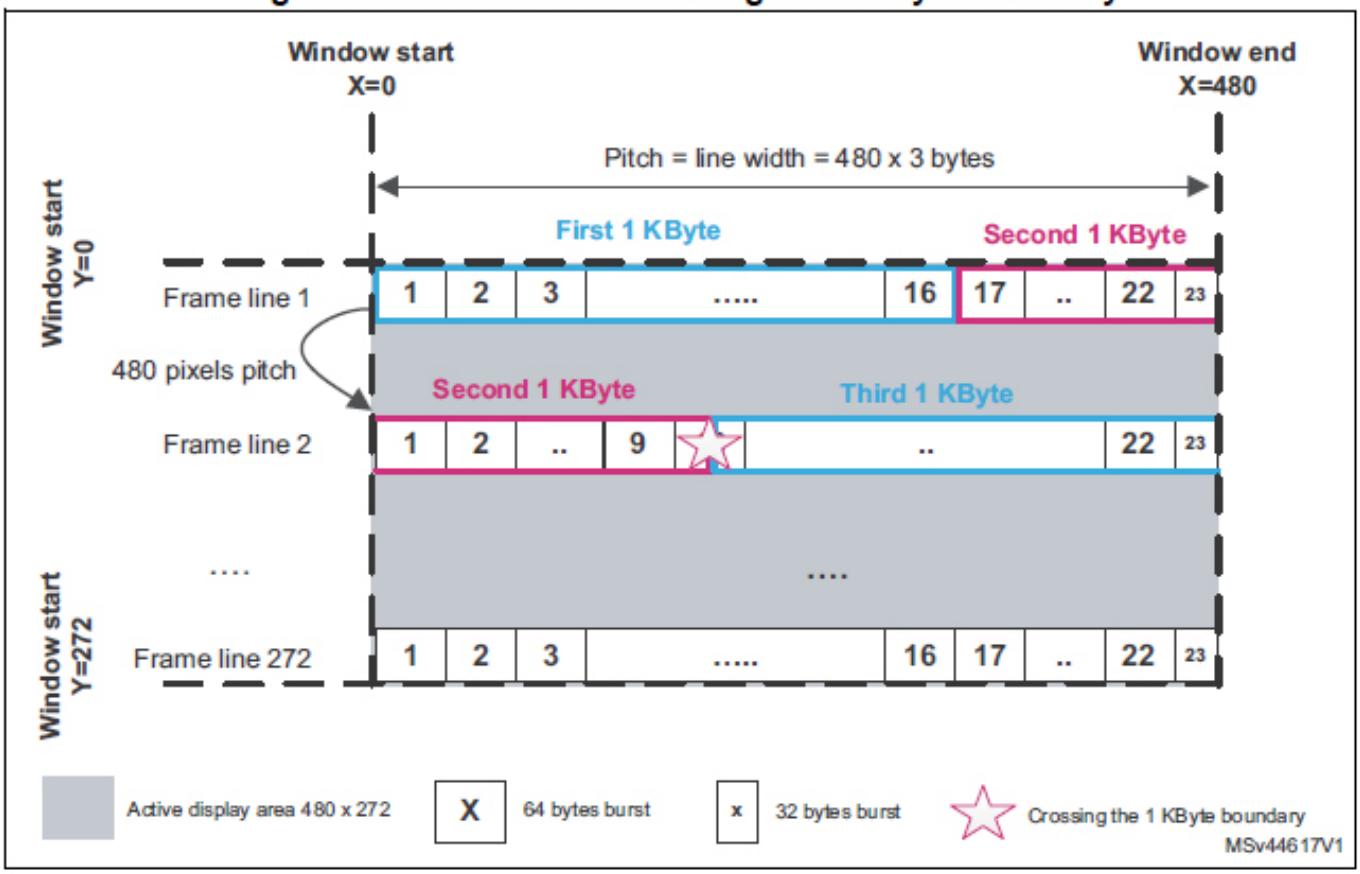
Для дисплея 480 x 272 (ширина линии кадра – 480 пикселей) и с глубиной цвета 24 bpp размер ширины линии равен 1440 байтам, который не является кратным 64 байтам.

Примечание. Для этого разрешения, чтобы иметь ширину ширины нескольких строк в 64 байта, пользователь может использовать другую глубину цвета, такую как RGB565.

Так как строка кадра состоит из 22 пакетов 64 байта и одного 32 байта, 10-й всплеск второй строки кадра пересекает границу Килобита. Это приводит к расколу операции чтения в единый доступ.

На рисунке 25 показана краевая крошечная проблема в килобайте для данного примера.

Figure 25. Burst access crossing the kilobyte boundary



В этом примере мы описываем два метода решения проблемы пересечения границы Килобита:

• Уменьшите ширину окна и ширину линии буфера кадра: используйте функцию оконного окна слоя LTDC, уменьшив размер окна, чтобы соответствовать ближайшей ширине линии кадра, кратной 64 байтам.

Так как ширина окна уменьшена, размер буфера кадра также должен быть

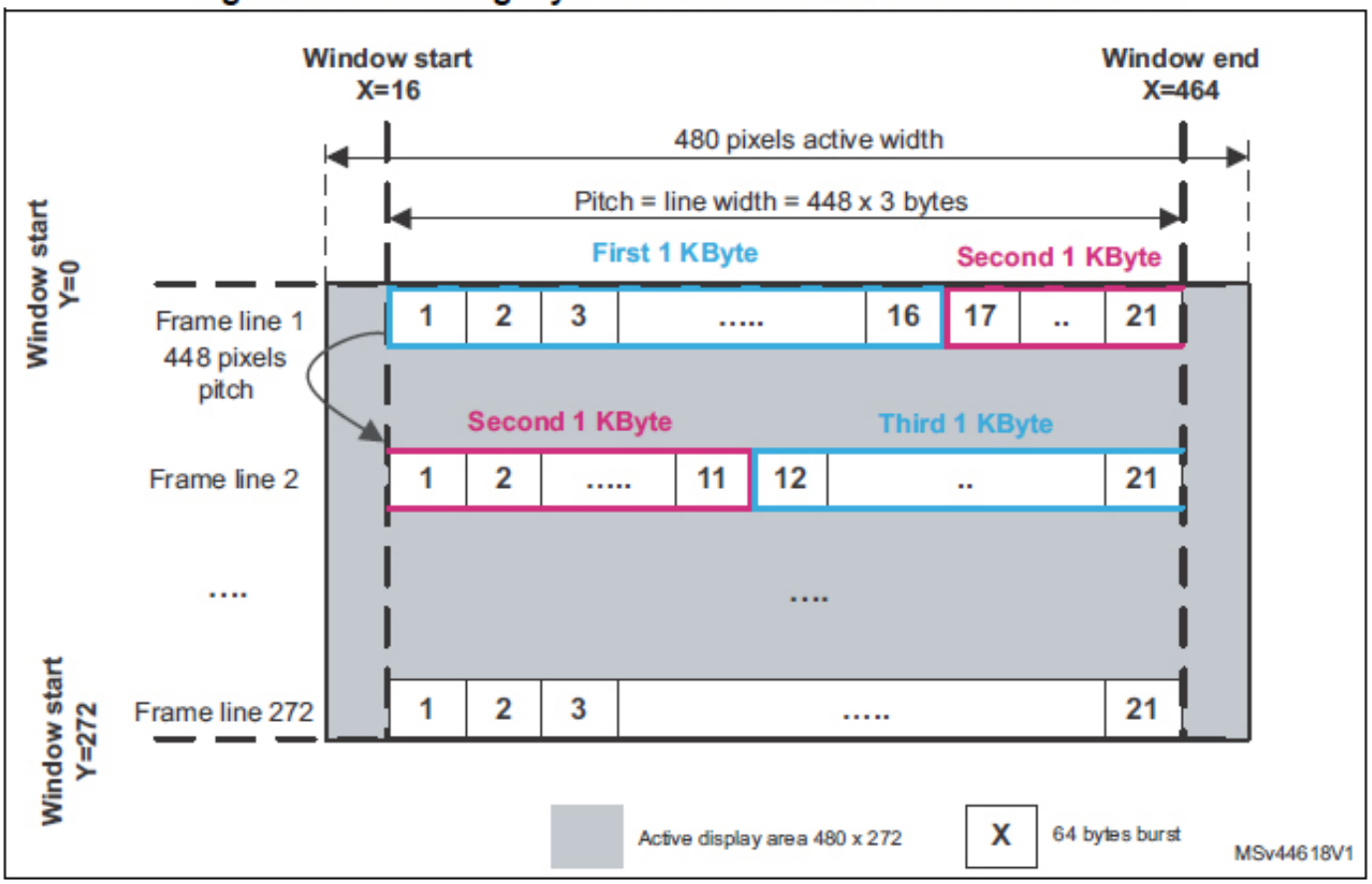
уменьшен, так как дополнительные 22 и 23 всплеска для всех линий фрейма не извлекаются и не отображаются LTDC.

Этот метод решает проблему перехода границы 1 Кбайт с небольшим уменьшением ширины окна (см. Рис. 26).

Нижеприведенный код основан на драйверах HAL и показывает пример настройки высоты тона, как показано на рисунке 26:

```
/* Установка окна Layer1 на 448x272 в положениях X = 16 и Y = 0 */
pLayerCfg.WindowX0 = 16;
pLayerCfg.WindowX1 = 464;
pLayerCfg.WindowY0 = 0;
pLayerCfg.WindowY1 = 272;
pLayerCfg.PixelFormat = LTDC_PIXEL_FORMAT_RGB888;
pLayerCfg.Alpha = 255;
pLayerCfg.Alpha0 = 0;
pLayerCfg.BlendingFactor1 = LTDC_BLENDING_FACTOR1_PAxCA;
pLayerCfg.BlendingFactor2 = LTDC_BLENDING_FACTOR1_PAxCA;
/* Начальный адрес Framebuffer: LTDC извлекает изображение непосредственно из внутренней Flash, а реальная ширина изображения составляет 448 пикселей. Отображается только ширина 448 пикселей*/
pLayerCfg.FBStartAdress = (uint32_t)&image_data_Image_RGB888_448x272;
pLayerCfg.ImageWidth = 448;
pLayerCfg.ImageHeight = 272;
pLayerCfg.Backcolor.Blue = 0;
pLayerCfg.Backcolor.Green = 0;
pLayerCfg.Backcolor.Red = 0;
```

Figure 26. Reducing layer window and framebuffer line widths



```
if (HAL_LTDC_ConfigLayer(&hltdc, &pLayerCfg, 0) != HAL_OK)
{
Error_Handler();

}
```

• Добавьте несколько фиктивных байтов в конце каждой строки пикселей, чтобы соответствовать самой близкой ширине линии кадра, кратной 64 байтам. Это можно сделать, используя шаг уровня LTDC (см. Раздел 3.3: Два программируемых слоя LTDC). Чтобы сделать это, пользователь должен рассмотреть два пункта ниже:

– буфер кадра должен содержать фиктивные байты (как описано на рисунке 27):

при записи данных в буфер кадра, это можно сделать, запрограммировав выходное смещение DMA2D, равное разности между ближайшим кратным пакетом и фактическим размером данных длины строки.

– Длина линии LTDC всегда должна быть равна размеру активных данных, но шаг LTDC должен быть запрограммирован со значением ближайшего числа байтов, кратного 64 байтам.

Функция HAL_LTDC_SetPitch, предоставляемая под драйвером hal_ltdc, может использоваться для программирования желаемого значения шага в количестве пикселей. В предыдущем примере значение шага для передачи этой функции должно быть равно 512 (512 – количество пикселей в строке, соответствующее размеру длины строки 1536 байт, что кратно 64 байтам).

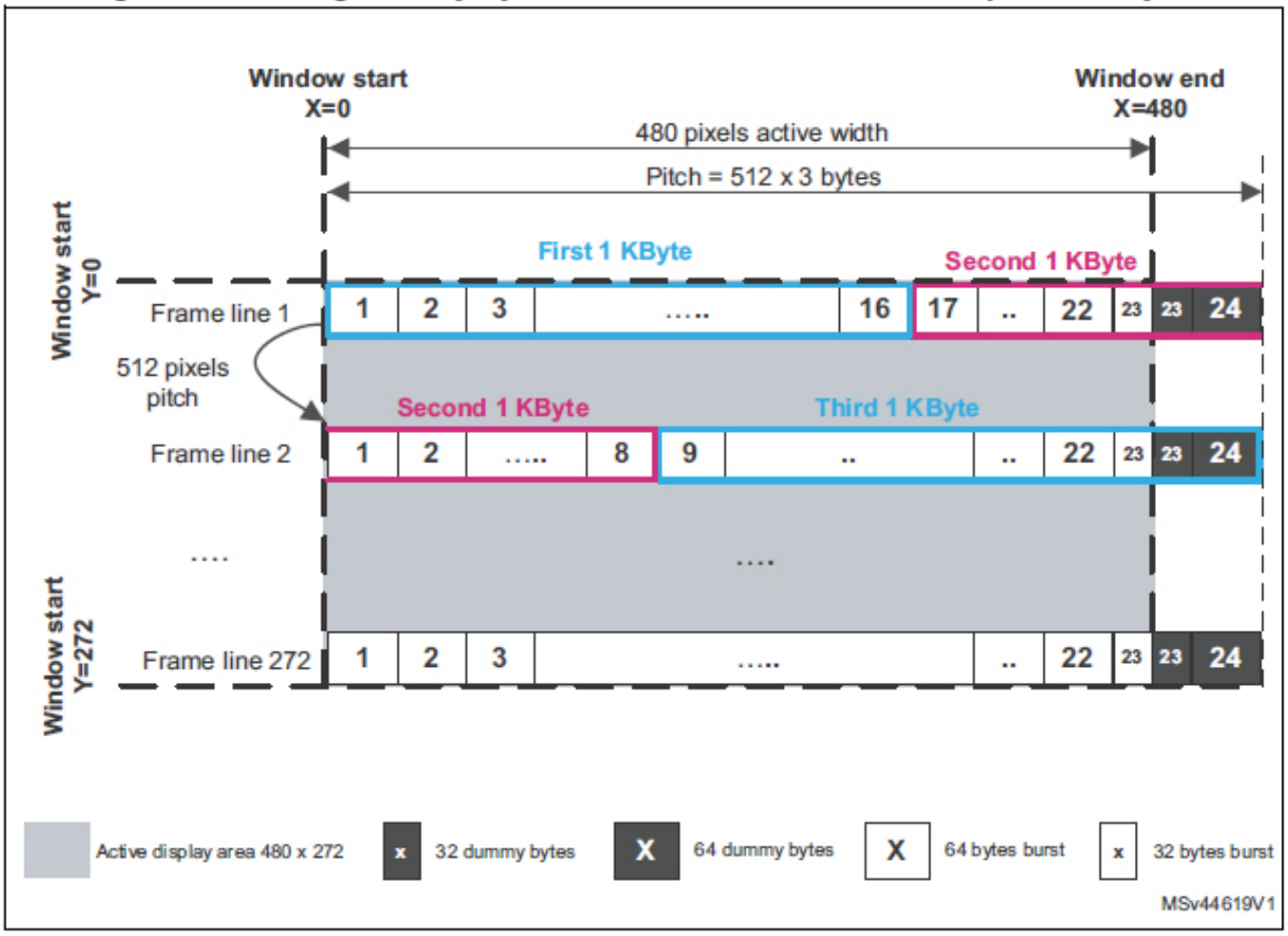
Нижеприведенный код основан на драйверах HAL и показывает пример настройки высоты тона, как показано на рисунке 27:

```
/* Установка окна Layer1 на 480x272 в положениях X = 0 и Y = 0 */
pLayerCfg.WindowX0 = 0;
pLayerCfg.WindowX1 = 480;
pLayerCfg.WindowY0 = 0;
pLayerCfg.WindowY1 = 272;
pLayerCfg.PixelFormat = LTDC_PIXEL_FORMAT_RGB888;
pLayerCfg.Alpha = 255;
pLayerCfg.Alpha0 = 0;
pLayerCfg.BlendingFactor1 = LTDC_BLENDING_FACTOR1_PAxCA;
pLayerCfg.BlendingFactor2 = LTDC_BLENDING_FACTOR1_PAxCA;
/* Начальный адрес Framebuffer: LTDC извлекает изображение непосредственно из внутренней Flash, а реальная ширина изображения составляет 480 пикселей, а дополнительные 32 пикселя добавляются к каждой строке, чтобы получить шаг пикселя в 512 пикселей.
Отображается только ширина 480 пикселей*/
pLayerCfg.FBStartAdress = (uint32_t)&image_data_Image_RGB888_512x272;
pLayerCfg.ImageWidth = 480;
pLayerCfg.ImageHeight = 272;
pLayerCfg.Backcolor.Blue = 0;
pLayerCfg.Backcolor.Green = 0;
pLayerCfg.Backcolor.Red = 0;
```

```
if (HAL_LTDC_ConfigLayer(&hltdc, &pLayerCfg, 0) != HAL_OK)
{
Error_Handler();
}
/*Устанавливает Layer1 (индекс 0 относится к Layer1) Шаг до 512 пикселей */
HAL_LTDC_SetPitch(&hltdc, 512, 0);
```

Рисунок 27 описывает управление памятью после решения предыдущей проблемы.

Figure 27. Adding dummy bytes to make the line width multiple of 64 bytes



4.5.3 Оптимизация загрузки буфера кадра LTDC из SDRAM

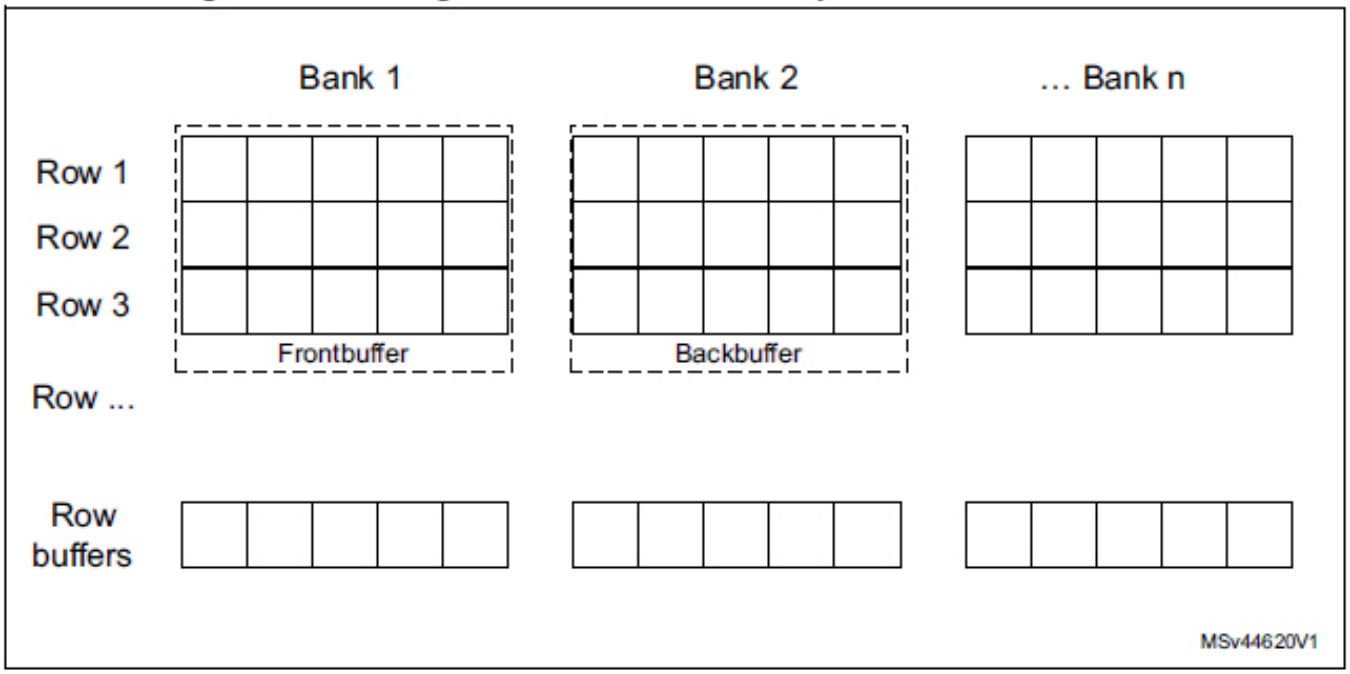
Выполнение произвольного доступа в банк генерирует несколько циклов предварительной зарядки, которые увеличивают задержку SDRAM, наблюдаемую LTDC. Поскольку LTDC выполняет последовательный доступ, важно, чтобы никакие другие мастера не делали доступа в один и тот же банк SDRAM.

Внешняя SDRAM состоит из нескольких банков. Учитывая, что произвольный доступ к банку генерирует некоторую предварительную загрузку и активирует некоторые циклы, буфер кадра должен быть помещен в независимый банк, к которому обращается только LTDC. Это действие уменьшает задержку внешней памяти и приводит к большей пропускной способности. Как следствие, при использовании метода двойной буфер кадра рекомендуется иметь эти буферы в двух отдельных банках.

Это можно сделать, сохранив передний буфер в первом адресе SDRAM и об-

ратившись к буферу буфера, добавив смещение с размером одного банка. См. Рисунок 28.

Figure 28. Placing the two buffers in independent SDRAM banks



Например, если размер банка SDRAM равен 4 Мбайт, можно использовать следующий код строки:

```
/* Адреса Framebuffer во внешней SDRAM */
/* Frontbuffer в банке 1 памяти SDRAM */
uint32_t FrontBuffer = LCD_FB_START_ADRESS;
/* Backbuffer в банке 2 памяти SDRAM */
uint32_t BackBuffer = LCD_FB_START_ADRESS + 1024 * 1024 * 4;
```

SDRAM RBURST

Еще одной интересной особенностью, позволяющей оптимизировать показания чтения из SDRAM, является использование RBURST.

Контроллер SDRAM добавляет кешируемый считываемый FIFO с глубиной 6 32-разрядных строк. Чтение FIFO используется, когда включен пакет чтения, и позволяет прогнозировать следующий доступ к чтению во время латентностей CAS.

4.5.4 Обновление содержимого Framebuffer в течение периода BLANKING

Способ оптимизации графической производительности (особенно, когда узким местом производительности является пропускная способность памяти буфер кадра), заключается в обновлении содержимого буфера кадра во время периода гашения. Поскольку в этот период LTDC не извлекает данные пикселя из буфера кадра, ширина полосы пропускания шины ослабляется и позволяет обновлять буфер кадра.

4.6 Специальные рекомендации для Cortex®-M7 (серия STM32F7)

В этом разделе приведены некоторые рекомендации для серии STM32F7, встра-

ивающей процессор Cortex®-M7. Эти рекомендации относятся к Cortex®-M7, поскольку он имеет следующие особенности по сравнению с процессором Cortex®-M4:

• Cortex®-M7 выполняет некоторые умозрительные обращения к нормальным областям памяти. Эти спекулятивные обращения к чтению могут вызывать большие задержки или системные ошибки при выполнении на внешних запоминающих устройствах, таких как SDRAM или Quad-SPI. Это влияет на мастеров AHB (например, LTDC), получающих доступ к FMC или Quad-SPI, и, в частности, снижает графические характеристики и может привести к системным ошибкам (если буфер кадра LTDC находится во внешней памяти и / или если используется память Quad-SPI для графики).

• Процессор Cortex®-M7 использует L1-Cache (см. Рисунок 10). Некоторые графические проблемы могут возникать из-за неподходящих настроек кеша; могут возникнуть плохие графические визуальные эффекты, если обслуживание кеша не выполняется должным образом. Если подходящий метод обслуживания кеша не используется, могут пострадать графические характеристики.

4.6.1 Отключить FMC bank1, если он не используется

После сброса FMC bank1 всегда позволяет загружать внешние записи. Поскольку Cortex®-M7 делает некоторые предположения, он может генерировать спекулятивный доступ для чтения к первому банку FMC.

Конфигурация FMC по умолчанию очень медленная, этот спекулятивный доступ блокирует доступ к FMC другими мастерами AHB в течение очень долгого времени, что приводит к недоразвитию на стороне LTDC.

Чтобы предотвратить этот доступ к умозрительному чтению процессора в банке FMC1, рекомендуется отключить его, когда он не используется. Это можно сделать, сбросив бит MBKEN в регистр FMC_BCR1, который по умолчанию включен после сброса.

Чтобы отключить FMC Bank1, пользователь может использовать следующий код:

/* Отключение FMC Bank1: после сброса FMC_BCR1 = 0x000030DB, где MBKEN = 1b означает, что FMC_Bank1 включен, и MTYP [1: 0] = 10 означает, что для типа памяти установлено значение NOR Flash / OneNAND Flash*/

FMC_Bank1->BTCR[0] = 0x000030D2;

Более подробную информацию о конфигурации FMC см. в соответствующем справочном руководстве STM32

4.6.2 Конфигурирование блока защиты памяти (MPU)

В этом разделе описываются атрибуты системной памяти серии STM32F7 и основные концепции MPU. В нем также описано, как настроить MPU, чтобы предотвратить проблемы с графической производительностью, связанные с умозрительными чтениями Cortex®-M7 и обслуживанием кеша.

Примечание. *В этом разделе описываются только некоторые базовые концепции MPU, необходимые для конфигурации. Для получения дополнительной информации о MPU и кеше обратитесь к следующим документам:*

– Замечание по применению AN4838 «Управление блоком защиты памяти (MPU) в MCU STM32»

– Замечание по применению AN4839 «Кэш уровня 1 на STM32F7 Series»

– Руководство по программированию процессора Cortex®-M7 серии STM32F7 (PM0253)

– Техническое руководство ARM Cortex®-M7

Конфигурация атрибутов MPU

Чтобы предотвратить проблемы с графической производительностью, связанные с умозрительным доступом чтения Cortex®-M7, пользователь должен просмотреть всю карту памяти приложения и настроить MPU в соответствии с оборудованием. Таким образом, пользователь должен установить следующие конфигурации:

• Определите область MPU буфер кадра и другие области MPU приложения.

• MPU должен быть настроен в соответствии с размером памяти, используемой приложением.

• Атрибуты MPU для неиспользуемых регионов должны быть настроены на строго упорядоченное выполнение никогда (XN). Например, для Quad-SPI, если подключена память объемом 8 Мбайт, оставшееся неиспользуемое пространство 248 Мбайт (из общего адресуемого пространства 256 МБ) должно быть настроено на строго упорядоченное XN. См. Пример в разделе 6.2.7.

• Предотвратите доступ к умозрительному чтению Coretx-M7 во внешний SDRAM / SRAM (если включена замена FMC, см. Рисунок 29), для этого необходимо, чтобы область MPU SDRAM / SRAM была настроена так, чтобы выполняться никогда (XN).

• Если процессор Cortex®-M7 используется для обработки буфера кадра (запись на SDRAM / SRAM), атрибут MPU области буфера кадра должен быть установлен на обычный кешируемый с правами доступа на чтение и запись.

Примечание. *Атрибут области MPU буфера кадра должен быть установлен так, чтобы он выполнялся никогда, поскольку он предназначен только для создания графического содержимого.*

Рисунок 29 описывает банки FMC STM32F7 и атрибуты памяти Quad-SPI MPU на карте памяти системной памяти по умолчанию

Конфигурация политики MPU и кэша

Использование кеша Cortex®-M7 позволяет повысить производительность системы и графики. Это повышение производительности особенно заметно, когда процессор получает доступ к внешним запоминающим устройствам, таким как SDRAM или Quad-SPI.

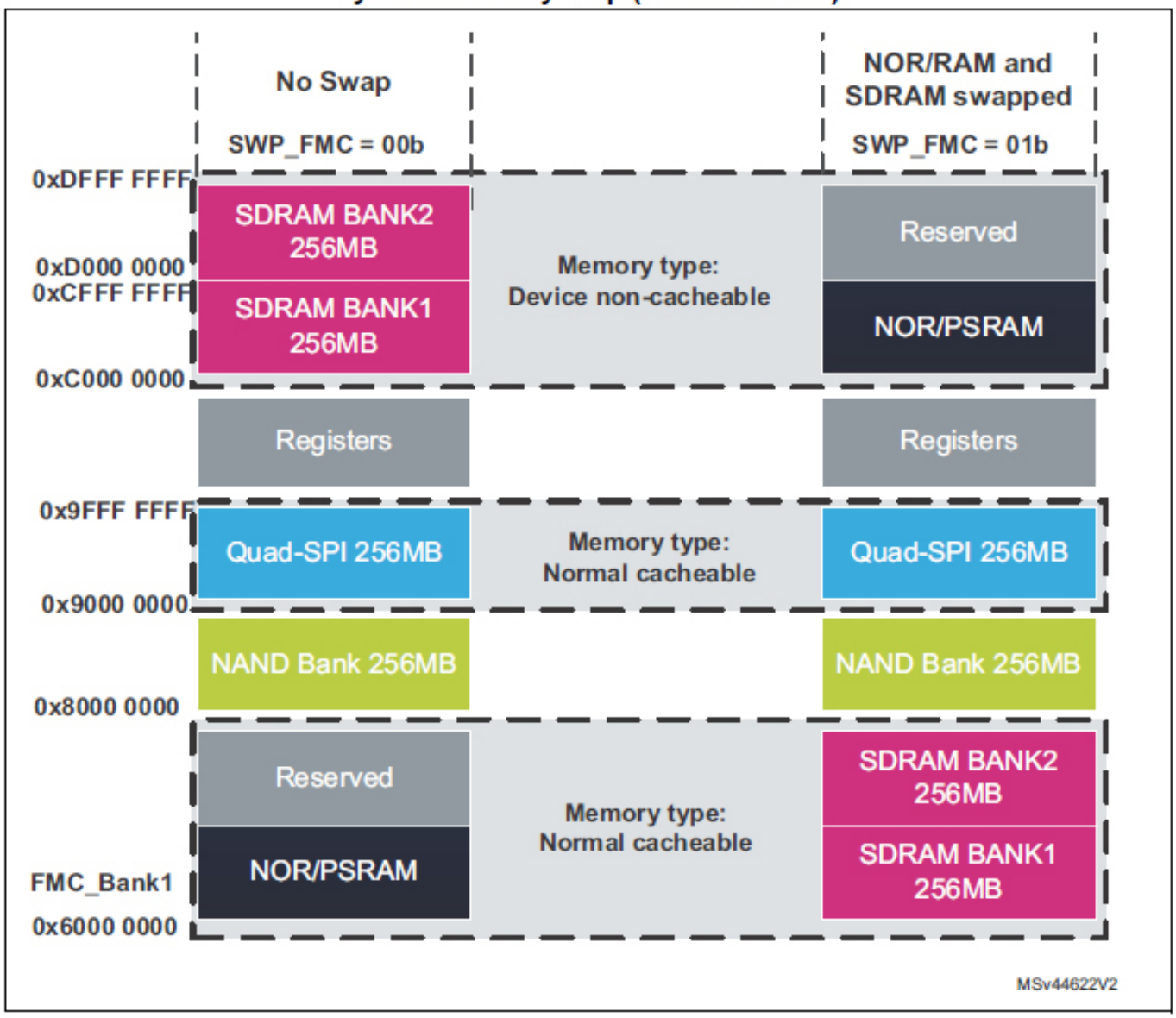
В графическом приложении, когда процессор используется для обработки буфера кадра, рекомендуется использовать кеш, особенно если буфер кадра находится во внешней памяти, такой как SDRAM или SRAM. В этом случае пользователь должен учитывать следующие моменты при использовании кеша:

• Кэшируемость области памяти MPU. Как показано ранее на рисунке 29, в атрибутах MPU области по умолчанию системной памяти, некоторые области системной памяти являются нормальными кэш-память, в то время как другие являются не кэшируемыми устройствами. Когда процессор используется для обработки буфера кадра, пользователь должен изменить атрибут MPU области framebuffer на обычный кешируемый (или выполнить обмен FMC, см. Рис. 29).

• Поддержка кэширования и согласованность данных: визуальное воздействие WBWA без обслуживания кеша

Проблема когерентности данных часто встречается при выполнении обработки буфера кадра с использованием процессора Cortex®-M7 с включенным L1-кешем

Figure 29. FMC SDRAM and NOR/PSRAM memory swap at default system memory map (MPU disabled)



и политикой кэша WBWA. Эта проблема возникает, когда несколько мастеров, таких как Cortex®-M7 и LTDC, используют одну и ту же область (буфер кадра), а обслуживание кеша не выполняется. Когда ЦП обрабатывает буфер кадра (записывает в буфер кадра), и если в области буфера кадра имеется политика кэша обратной записи, обработанный результат (изображение, которое будет отображаться) не будет видно на буфере кадра (может быть SRAM или SDRAM), а затем он не отображается.

Чтобы избежать этой проблемы, пользователь должен использовать один из следующих способов:

– Настроить атрибут кеша области буфера кадра для сквозной записи (WT), в этом случае каждая операция записи выполняется в кеше и буфере фреймов.

– Настройте атрибут кеша области буфера кадра, чтобы записать резервное копирование записи (WBWA) и выполнить обслуживание кэша программным обеспечением.

Считывание данных более безопасно для когерентности данных, но может влиять на графические характеристики

• Обслуживание кэша может повлиять на графические характеристики. Поль-

зователь должен использовать подходящую политику кэша, соответствующую его приложению. У каждого метода есть свои минусы и плюсы. Таким образом, пользователь должен учитывать следующие особенности для каждого метода:

– Прошивка: очень простая в управлении (нет необходимости выполнять обслуживание кешей программным обеспечением) и более безопасна для согласованности данных, но она генерирует много операций одиночной записи в буфер кадра, что может повлиять на доступ к LTDC.

Примечание. *Пользователь должен также учитывать, что обслуживание кеша может влиять на графические характеристики, даже если процессор не используется для обработки буфера кадра. Таким образом, в некоторых приложениях ЦП получает доступ к внешней SDRAM или SRAM для целей, отличных от графики с включенным кешем. В этом случае обслуживание кеша может повлиять на доступ к LTDC.*

– Записать обратную запись: лучше использовать WBWA и программную процедуру и синхронизировать операцию обслуживания кеша с LTDC во время гашения. Это позволяет создать дополнительную полосу пропускания в памяти буфера кадра (SRAM или SDRAM). Операция обслуживания кеша должна выполняться программным обеспечением после записи данных в область памяти буфера кадра; это делается путем принудительной очистки Dcache с использованием функции CMSIS SCB_CleanDCache (). Таким образом, все грязные строки в кеше записываются обратно в буфер кадра.

Пример конфигурации MPU

Пример конфигурации MPU описан в разделе 6.2.7, где показано, как установить атрибут MPU буфера кадра при использовании ЦП (с включенным кешем) для графических операций. Описанный пример создан для конфигурации аппаратного обеспечения платы STM32F746G-DISCO, где внешняя SDRAM используется для буфера фреймов, а внешняя флэш-память QSPI содержит графические примитивы.

4.7 периферийная конфигурация LTDC

В этом разделе описаны шаги, необходимые для настройки периферийного устройства LTDC.

Примечание. *Перед началом настройки рекомендуется сбросить периферийное устройство LTDC, а также рекомендуется гарантировать, что периферийное устройство находится в состоянии сброса. LTDC можно сбросить, установив соответствующий бит в регистре RCC_APB2RSTR, который сбрасывает три тактовых домена.*

4.7.1 Подключение панели дисплея

Аппаратный интерфейс LTDC обеспечивает восемь бит на цветную шину и идеально подходит для истинных цветных панелей RGB888. Аппаратный интерфейс LTDC обеспечивает также синхронизирующие сигналы: LCD_HSYNC, LCD_VSYNC, LCD_DE и LCD_CLK.

GPIO LTDC следует настроить на соответствующую альтернативную функцию. Для получения дополнительной информации о доступности альтернативных функций LTDC и GPIO см. Таблицу сопоставления альтернативных функций в соответствующем техническом паспорте.

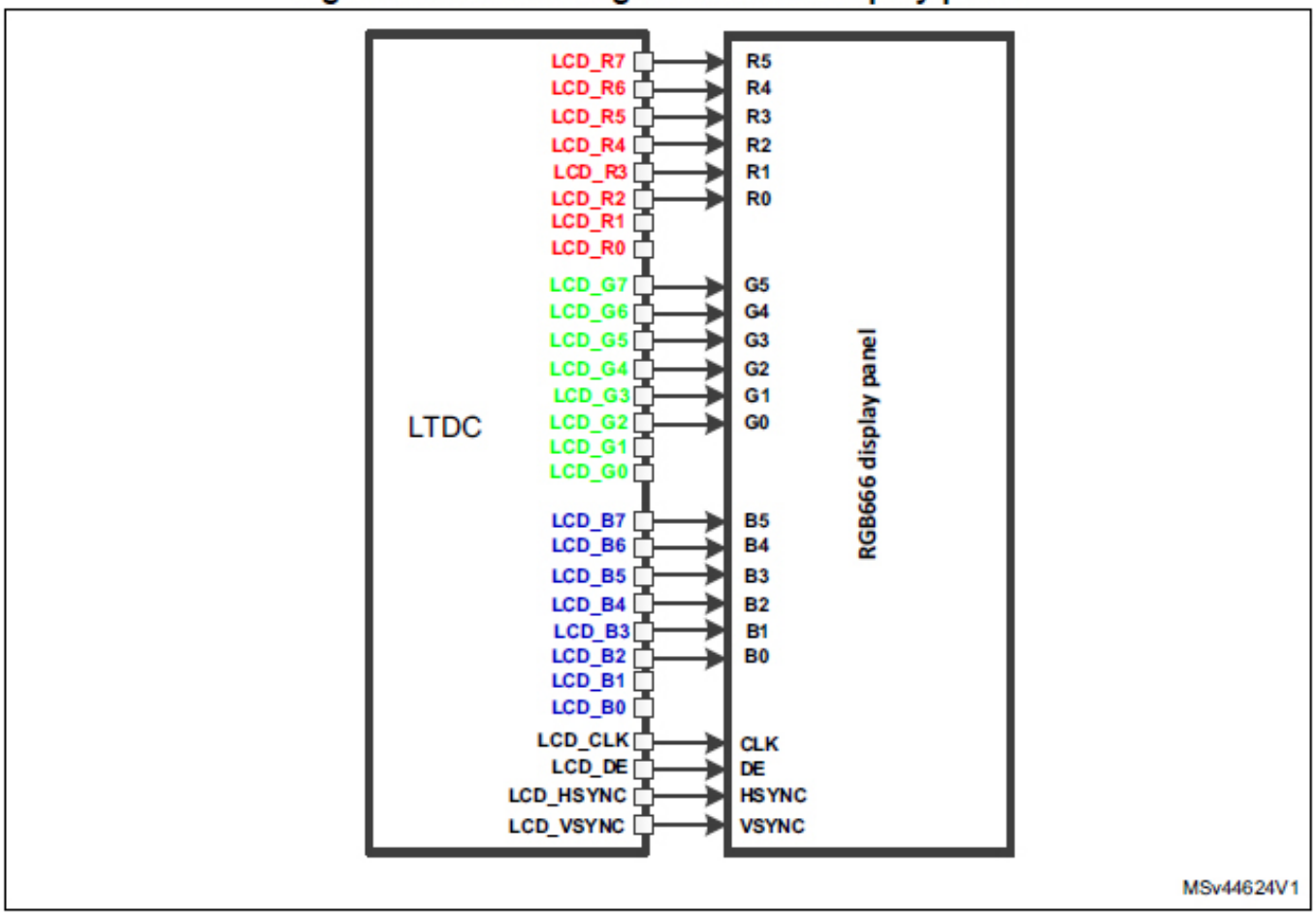
Примечание. *Все GPIO должны быть настроены в очень высокоскоростном режиме.*

Подключение пониженной панели дисплея

Для панелей дисплея с более низкой цветовой палитрой (например, RGB666 и RGB565) подключение к шине должно выполняться с использованием наиболее значимых бит сигналов данных. На рисунке 30 показан пример подключения панели дисплея RGB666.

Конфигурация GPIO с использованием инструмента STM32CubeMX

Figure 30. Connecting an RGB666 display panel



Чтобы подключить панель дисплея к MCU STM32, пользователь должен настроить GPIO, которые будут использоваться для сопряжения.

Использование инструмента STM32CubeMX – очень простой, простой и быстрый способ настройки периферийного устройства LTDC и его GPIO, поскольку он позволяет генерировать проект с предварительно сконфигурированным LTDC.

Раздел 6.2.3: Конфигурация GPIO в LTDC предоставляет руководство по настройке GPIO LTDC.

Конфигурация конкретных контактов модуля дисплея

Некоторым модулям отображения могут потребоваться, чтобы другие сигналы были полностью работоспособными. Для управления этими сигналами пользователь может использовать GPIO и некоторые периферийные устройства.

Пример использования GPIO для управления выводом разрешения дисплея (LCD_DISP) на панели дисплея описан в разделе Раздел 6.2.3: Конфигурация GPIO для LTDC.

Включение прерываний LTDC

Чтобы иметь возможность использовать прерывания LTDC, пользователь должен включить глобальные прерывания LTDC на стороне NVIC. Затем каждое пре-

рывание активируется отдельно, разрешая его соответствующий бит разрешения. Биты разрешения прерывания LTDC доступны в регистре LTDC_IER, описанном в таблице 7: сводка прерываний LTDC.

Примечание. *Прерывания опустошения буфера и ошибки передачи включены в функции hal_ltdc драйвера HAL_LTDC_Init ()*

Пример включения прерываний LTDC с использованием STM32CubeMX описан в разделе 6.2.3: Конфигурация GPIO для LTDC.

4.7.2. Настройка часов и времени синхронизации LTDC.

В этом разделе описываются шаги, необходимые для настройки часов и времени LTDC, соответствующих спецификациям дисплея. Он также представляет собой пример конфигурации для дисплея ROCKTECH (RK043FN48H), встроенного в плату STM32F746G-DISCO.

Конфигурация системных часов

Для получения наилучших графических характеристик рекомендуется использовать самые высокие системные часы. Эта рекомендация применяется также для буфера кадра внешней памяти. Поэтому, если для буфера кадра используется внешняя память, максимальная допустимая тактовая частота должна использоваться для получения наилучшей пропускной способности памяти.

Например, для микроконтроллеров STM32F4x9 максимальная скорость системы составляет 180 МГц, поэтому, если внешняя SDRAM подключена к FMC, максимальные часы SDRAM составляют 90 МГц (HCLK / 2).

Для серии STM32F7 максимальная скорость системы составляет 216 МГц, но с этой скоростью и предварительным делителем HCLK / 2 скорость SDRAM превышает максимально допустимую скорость (более подробную информацию см. В техническом описании продукта). Поэтому, чтобы получить максимальную SDRAM, рекомендуется настроить HCLK на 200 МГц, тогда скорость SDRAM установлена на 100 МГц.

Конфигурация часов, обеспечивающая самые высокие показатели:

- Устройства STM32F4x9: HCLK @ 180 МГц и SDRAM @ 90 МГц.
- Серия STM32F7: HCLK @ 200 МГц и SDRAM @ 100 МГц.

Пример конфигурации LTDC с использованием STM32CubeMX описан в разделе 6.2.4: периферийная конфигурация LTDC.

Конфигурация часов и таймингов

На этом этапе разработки графического приложения пользователь должен был уже проверить и подтвердить, что желаемый размер и глубина цвета совместимы с конфигурацией оборудования. Поэтому синхронизируемые часы пикселей должны быть уже известны, либо извлечены из таблицы данных, либо вычислены (см. Раздел 4.2.2 «Проверка совместимости дисплея с учетом требований к пропускной способности памяти»).

Пример: Конфигурация таймингов LTDC для дисплея ROCKTECH RK043FN48H, встроенного в плату STM32F746G-DISCO

Сначала пользователь должен извлечь временные параметры из таблицы данных дисплея (см. Таблицу 14). Рекомендуется использовать типичные тайминги дисплея.

1. Серая ячейка выделяет значения, используемые в приведенном ниже примере.

Table 14. LCD-TFT timings extracted from ROCKTECH RK043FN48H datasheet (1)

Item		Symbol	Min.	Typ.	Max.	Unit
DCLK frequency		Fclk	5	9	12	MHz
DCLK period		Tclk	83	110	200	ns
Hsync	Period time	Th	490	531	605	DCLK
	Display period	Thdisp	-	480	-	DCLK
	Back porch	Thbp	8	43	-	DCLK
	Front porch	Thfp	2	8	-	DCLK
	Pulse width	Thw	1	-	-	DCLK
Vsync	Period time	Tv	275	288	335	H
	Display period	Tvdisp	-	272	-	H
	Back porch	Tvbp	2	12	-	H
	Front porch	Tvfp	1	4	-	H
	Pulse width	Tvw	1	10	-	H

На основании таблицы 14, выделенные временные параметры:
– Период отображения (активная ширина) = 480 пикселей
– Длина заднего строчного защитного интервала HBP = 43 пикселя
– Длина переднего строчного защитного интервала HFP = 8 пикселей
– Ширина импульса HSYNC = 1 пиксель (минимальное значение)
– Период отображения (активная высота) = 272 пикселя
– Длина заднего кадрового защитного интервала VBP = 12 пикселей
– Длина переднего кадрового защитного интервала VFP = 4 пикселя
– Ширина импульса VSYNC = 10 пикселей

Временные параметры программы: как только параметры синхронизации будут извлечены, они используются для программирования регистров времени LTDC, в таблице 15 приведены все параметры, которые необходимо запрограммировать.

Конфигурация параметров синхронизации с помощью STM32CubeMX: очень просто запрограммировать параметры синхронизации с помощью STM32CubeMX, пользователь должен просто заполнить извлеченные параметры в окне конфигурации LTDC (см. Раздел Раздел 6.2.4: периферийная конфигурация LTDC).

Конфигурация пиксельных часов с помощью STM32CubeMX: часы пикселей вычисляются с частотой обновления 60 Гц, как показано ниже:

LCD_CLK = TOTALW x TOTALH x частота обновления

На основании таблицы 15 TOTALW = 531 и TOTALH = 297.

И для этого примера: LCD_CLK = 531 x 297 x 60 = 9,5 МГц

Обратитесь к примеру STM32CubeMX конфигурации синхронизации LTDC в разделе 6.2.4.

Table 15. Programming LTDC timing registers

Register			Value to be programmed
LTDC_SSCR	HSW[11:0]	HSYNC Width - 1	0
	VSH[11:0]	VSYNC Height - 1	9
LTDC_BPCR	AHBP[11:0]	HSYNC Width + HBP - 1	43
	AVBP[10:0]	VSYNC Height + VBP - 1	21
LTDC_AWCR	AAW[11:0]	HSYNC Width + HBP + Active Width - 1	523
	AAH[10:0]	VSYNC Height + BVBP + Active Height - 1	293
LTDC_TWCR	TOTALW[11:0]	HSYNC Width + HBP + Active Width + HFP - 1	531
	TOTALH[10:0]	VSYNC Height+ BVBP + Active Height + VFP - 1	297

Конфигурация полярности сигналов управления LTDC

Сигналы управления LTDC (HSYNC, VSYNC, DE и LCD_CLK) должны быть настроены с учетом спецификаций дисплея. Пользователь должен учитывать, что только управляющий сигнал DE должен быть инвертирован в сравнении с полярностью DE, указанной в таблице данных дисплея. Другие управляющие сигналы должны быть сконфигурированы точно так же, как и таблицей данных.

4.7.3 Конфигурация слоя (-ов) LTDC

В этом разделе описываются необходимые шаги для настройки уровня (-ов) LTDC с учетом размера экрана и глубины цвета.

Как было сказано ранее в разделе 3.3.2, LTDC имеет два независимо настраиваемых уровня, где пользователь может включать один или два уровня. По умолчанию оба слоя отключены, поэтому отображается только настроенный цвет фона (цвет по умолчанию черный).

Пользователь может отображать слой 1 + фон или слой отображения 1 + слой 2 + фон.

Отображать только фон

Если ни один слой не был включен, отображается только фон. Если фоновый цвет не настроен, отображается фоновый черный цвет по умолчанию

(LTDC_BCCR = 0x00000000).

Чтобы установить синий цвет фона, регистр LTDC_BCCR должен быть установлен в 0x000000FF.

Конфигурация параметров слоя

После того, как GPIO, часы и тайминги LTDC будут правильно настроены, пользователь должен настроить следующие параметры уровня LTDC. Каждый уровень LTDC имеет свои собственные параметры, поэтому его следует настраивать отдельно.

- Размер и расположение окна
- Формат ввода пикселей
- Начальный адрес Framebuffer
- Размер Framebuffer (ширина изображения и высота изображения) и шаг
- Цвет по умолчанию для слоя в формате ARGB8888

- Layer constant alpha для смешивания
- Фактор смешивания слоев1 и фактор2

Пример настройки параметров уровня LTDC с использованием STM32CubeMX описан в разделе «Конфигурация параметров уровня LTDC» на стр. 75.

Примечание. *Все параметры слоя могут быть изменены «на лету», за исключением CLUT. Новая конфигурация должна быть либо перезагружена немедленно, либо во время периода вертикального гашения, путем настройки регистра LTDC_SRCR.*

4.7.4 Конфигурация панели дисплея

Некоторые дисплеи должны быть настроены с использованием последовательных интерфейсов связи, таких как I2C или SPI.

Например, STM32F429I-DISCO встраивает модуль отображения ILI9341, который инициализируется через интерфейс SPI.

Специальный драйвер для этого модуля отображения (ili9341.c), включая команды инициализации и конфигурации, доступен в пакете прошивки STM32Cube по адресу:

STM32Cube_FW_F4_Vx.xx.x \ Drivers \ BSP \ Components \ ili9341

Пример последовательности инициализации дисплея, основанный на функции ili9341_Init (), включен в примеры STM32Cube для платы STM32F429I-Discovery в STM32Cube_FW_F4_Vx.xx.x \ Projects \ STM32F429I-Discovery \

Примеры \ LTDC \ LTDC_Display_2Layers

4.8 Сохранение графических примитивов

Графические примитивы – это базовые элементы (например, изображения или шрифты), которые могут объединяться для создания отображаемого содержимого буфера кадра.

Статические данные должны быть помещены в энергонезависимую память. Когда количество данных для хранения относительно невелико, можно использовать внутреннюю FLASH-память. В противном случае графическое содержимое должно быть размещено во внешних запоминающих устройствах.

Микроконтроллеры STM32 предлагают параллельный (FMC) или последовательный (QSPI) интерфейс для внешних вспышек NOR (см. Таблицу 3).

Чтобы создать содержимое буфера кадра, DMA2D может напрямую считывать графические примитивы из параллельной NOR Flash или Quad-SPI Flash.

См. Примечание к приложению. Интерфейс Quad-SPI (QSPI) на микроконтроллерах STM32 (AN4760) для получения дополнительной информации о хранении графического содержимого в QSPI-памяти.

4.8.1 Преобразование изображений в файлы C

Чтобы добавить графические примитивы к пользовательскому проекту, они должны быть преобразованы в файлы C или заголовков. Для этого пользователь может использовать некоторые специальные инструменты, позволяющие создавать файлы C или * .h.

Предупреждение. Пользователь должен преобразовать изображения в файлы C в соответствии с форматом ввода настроенного пикселя, описанным в разделе: Формат ввода пикселей на стр. 27. Некоторые инструменты могут генерировать

файлы C или * .h с красными и синими цветами. Чтобы избежать этой проблемы (показывая изображение с красным и синим цветом), пользователь может использовать инструмент преобразования ЖК-дисплея.

Конвертер ЖК-изображений – это очень настраиваемый бесплатный инструмент, позволяющий преобразовывать изображения в файлы C и предлагая возможность сгенерировать файл C в нужном формате. Пример описан в разделе 6.2.5: Отображение изображения из внутренней Flash.

4.9 Аппаратные соображения

В этом разделе приведены некоторые аппаратные соображения для графического приложения. В общем, необходимо тщательно спроектировать два важных аппаратных интерфейса: интерфейс LTDC и интерфейс внешней памяти (используется для фрейм-буфера), например FMC_SDRAM или FMC_SRAM.

Параллельный интерфейс LTDC

Когда часы с пикселями ниже 40 МГц (SVGA), можно использовать простую сигнализацию 3,3 В.

Можно достичь 83 МГц с параллельным RGB при уменьшении нагрузки и длины провода (например, для интерфейса на той же плате с бортовым транспондером LVDS или приемопередатчиком HDMI).

Рекомендуется настроить GPIO LTDC с максимальной рабочей скоростью OSPEEDRy [1: 0] = 11b. Обратитесь к соответствующему справочному руководству STM32 за описанием регистра скорости передачи портов GPIOx_SPEEDR GPIO.

Интерфейс FMC SDRAM / SRAM

При использовании внешней памяти SRAM или SDRAM для буфера кадра скорость интерфейса FMC-SDRAM и FMC-SRAM зависит от многих факторов, включая раскладку платы и скорость пэда. Хорошая конструкция печатной платы позволяет достичь максимальных часов пикселя, описанных в таблице 10 и таблице 11.

Макет должен быть как можно лучше, чтобы получить лучшие результаты. Чтобы получить дополнительную информацию о принципах маршрутизации на печатной плате, см. Следующие примечания к приложениям, доступные на вебсайте STMicroelectronics:

- Раздел «Интерфейс контроллера гибкой памяти (FMC)» примечания к приложению Начало работы с аппаратной разработкой MCU серии STM32F7 (AN4661)
- Раздел «Интерфейс контроллера гибкой памяти (FMC)» примечания к приложению Начало работы с разработкой аппаратного обеспечения MCU STM32F4xxxx (AN4488)

5 Экономия энергопотребления

Когда приложение находится в режиме ожидания и отображает только экранную заставку, важно сэкономить чип STM32 в спящем режиме, чтобы снизить энергопотребление. В спящем режиме все периферийные устройства могут быть включены (например, FMC-SDRAM и LTDC), когда CPU остановлен.

Внешние запоминающие устройства, такие как SDRAM или QSPI FLASH, могут управляться в режимах с низким энергопотреблением, когда это необходимо, чтобы избежать потери энергии.

Если приложение находится в состоянии с низким энергопотреблением, но требует отображения графики, LTDC может быть активным, и SDRAM может быть переведен в режим самообновления (для экономии энергии). Если приложение установлено в режиме пониженного энергопотребления, оно экономит еще большую мощность.

Дисплей также можно отключить или включить в режим низкого энергопотребления, если он не нужен при запуске приложения.

6 Примеры применения LTDC

В этом разделе приведены:

• Некоторые примеры графической реализации с учетом требований к ресурсам

• Пример создания основного графического приложения

• Сводка эталонных плат STM32 с внедрением LTDC и оснащена встроенной панелью LCD-TFT

6.1 Примеры реализации и требования к ресурсам

В этом разделе приведены примеры графической реализации, в которых подробно описаны требования к аппаратным ресурсам.

6.1.1 Одиночный микроконтроллер MCU

Благодаря интегрированному SRAM, MCM STM32 можно использовать для графических приложений без необходимости использования внешней памяти SDRAM / SRAM для буфера кадра. Кроме того, благодаря высокой внутренней внутренней Flash (до 2 Мбайт) пользователь может использовать их для хранения графических примитивов. Использование внутренних запоминающих устройств позволяет уменьшить используемые контакты, облегчить проектирование печатной платы и снизить затраты.

Чтобы использовать одночиповый микроконтроллер для графического приложения, пользователь может использовать следующую аппаратную конфигурацию:

• Внутренняя вспышка до 2 Мбайт: сохранение кода пользовательского приложения и графических примитивов.

• Буфер кадра должен располагаться во внутренней SRAM, поэтому в зависимости от внутреннего размера SRAM для каждого MCU STM32 пользователь может взаимодействовать с соответствующим размером дисплея и глубиной цвета, как показано ниже:

– Строка STM32F7x7: используйте SRAM1 (368 Кбайт) для поддержки разрешений 400 x 400 16 бит / с (313 Кбайт) или 480 x 272 16 бит / с (255 Кбайт).

– Строка STM32F7x6: используйте SRAM1 (240 Кбайт) для поддержки разрешения 320 x 320 с 16 бит / с (200 Кбайт).

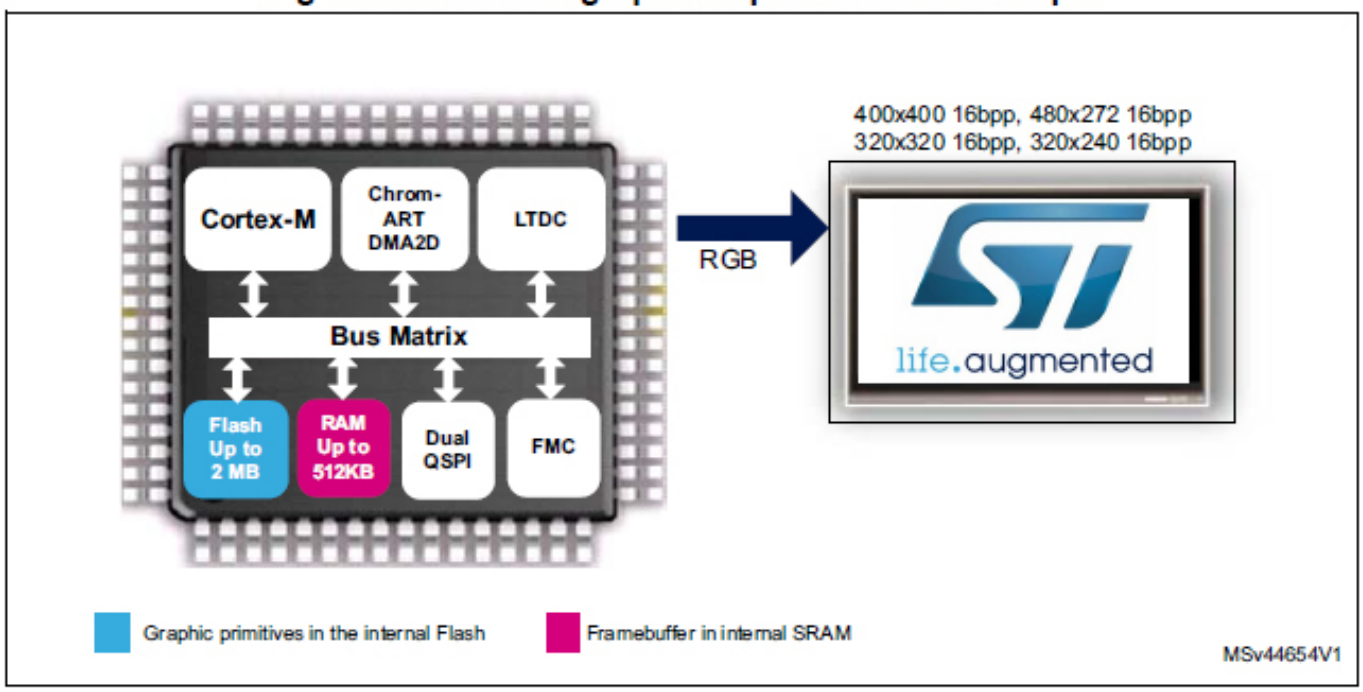
– Строка STM32F469 / F479: используйте SRAM1 (160 Кбайт) для поддержки разрешения 320 x 240 с 16 бит / с (154 Кбайт).

– Строка STM32F429 / F439: используйте SRAM1 (112 Кбайт) для поддержки разрешения 320 x 240 с 8 bpp (75 Кбайт).

– Пакеты STM32 MCU: LQFP100 или TFBGA100

На рисунке 31 показан пример графической реализации, в котором используется только один чип без внешних запоминающих устройств.

Figure 31. Low-end graphic implementation example



6.1.2 MCU с внешней памятью

Для взаимодействия с дисплеями с более высоким разрешением для буфера кадра необходима внешняя память, подключенная к FMC. Внешнюю флэш-память QSPI можно использовать для хранения графических примитивов.

Для графических приложений среднего или высокого уровня пользователь может использовать следующий пример конфигурации оборудования:

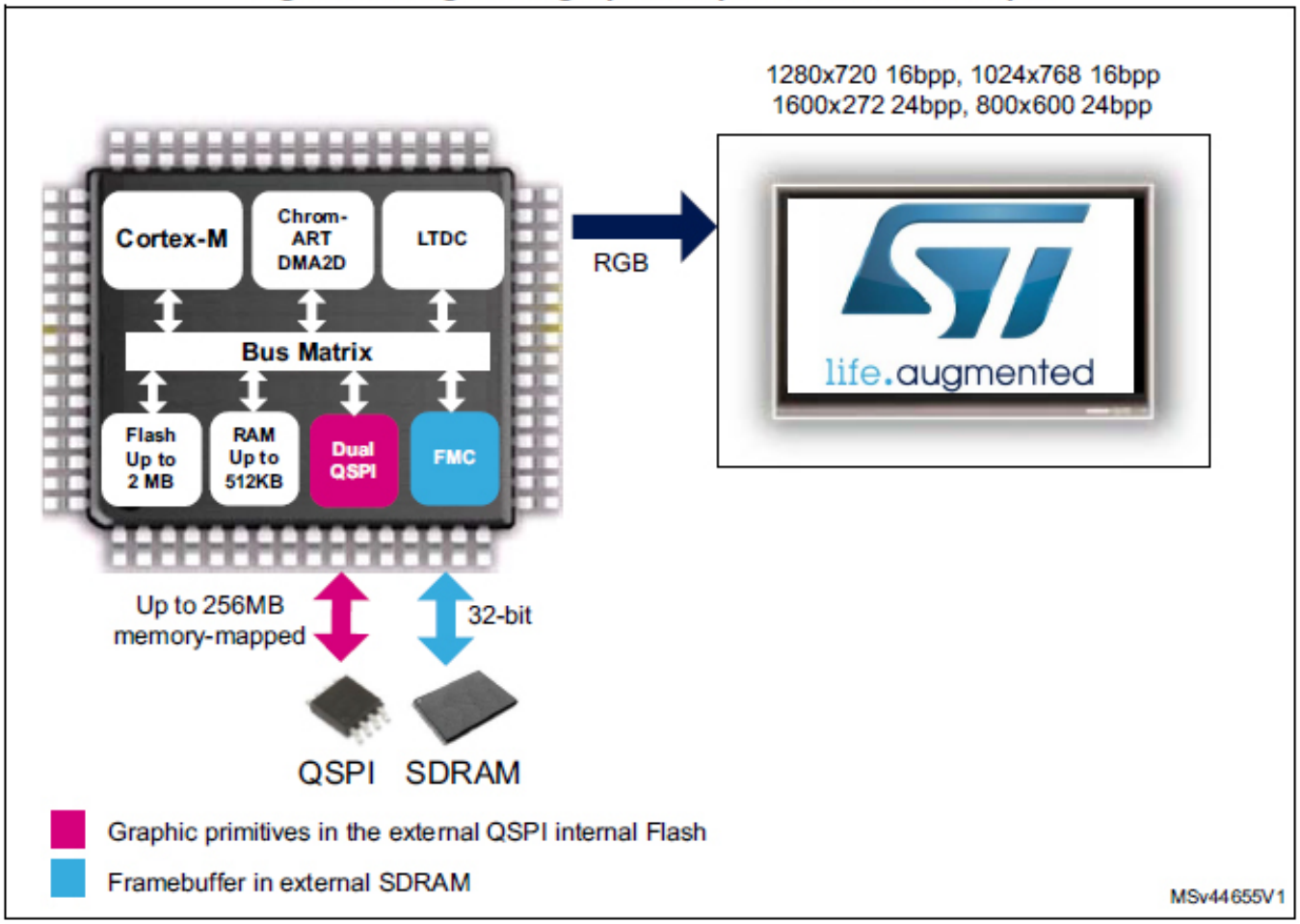
- Внешняя QSPI флэш-память с адресной памятью до 256 Мбайт предназначена для хранения графических примитивов.
- Внешняя SDRAM 32-разрядная память, используемая для буфера кадра.
- Пакеты STM32 MCU: UFBGA169, UFBGA176, LQFP176, LQFP208, TFBGA216, WLCSP168 и WLCSP180.

На рисунке 32 показан пример графической реализации, в котором две внешние запоминающие устройства подключены к MCU STM32, один для буфера кадра, а другой для графических примитивов.

В таблице 16 представлен пример графических реализаций в различных конфигурациях аппаратного обеспечения STM32.

Таблица 16. Пример графических имплантаций с STM32

Figure 32. High-end graphic implementation example



в различных конфигурациях оборудования

1. Доступность пакета STM32 MCU для внедрения LTDC приведена в таблице 13.

Variant	Display size	Color depth	External memory - SDRAM	Display interface	STM32 package(1)
Hig-end	1280 x 720	16 bpp	32-bit	RGB888	UFBGA176 TFBGA216/UFBGA169/ LQFP176/LQFP208 WLCSP180/WLCSP168
	1024 x 768				
	1600 x 272	24 bpp			
	800 x 600				
Mid-end	800 x 600	16 bpp	16-bit	RGB666	LQFP144/WLCSP143
	800 x 480	24 bpp			
	640 x 480				
	400 x 400(2)				
Low-end	400 x 400(2)	16 bpp	No	RGB666	LQFP100/TFBGA100
	480 x 272				
	320 x 320(2)				
	320 x 240				

2. 400x400 и 320 x 320 – это конкретные разрешения дисплея, обычно используемые для смартватов

6.2 Пример: создание базового графического приложения

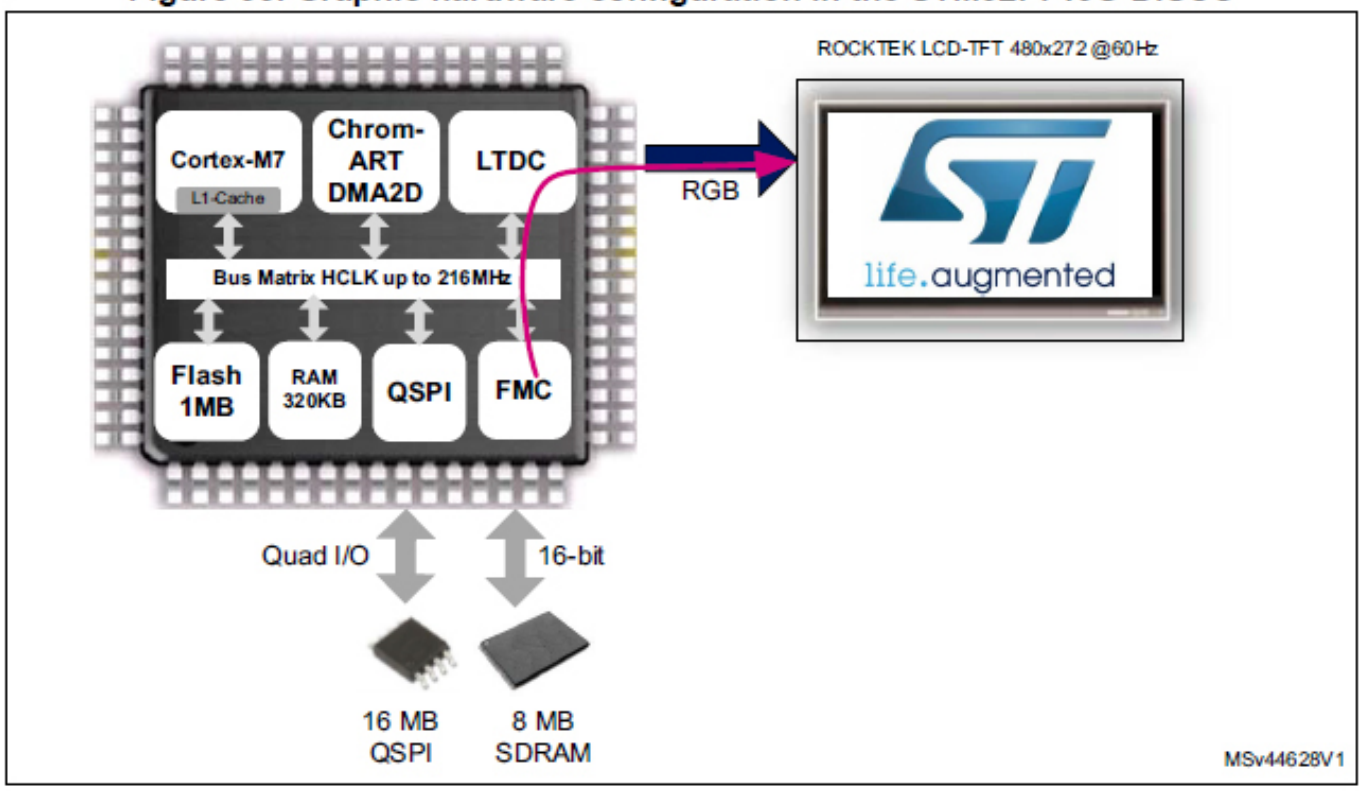
В этом разделе приведен пример, основанный на плате STM32F746G-DISCO, описывающий шаги, необходимые для создания базового графического приложения.

6.2.1 Описание оборудования

В этом примере используются аппаратные ресурсы, встроенные в плату STM32F746G-DISCO. На рисунке 33 описаны графические аппаратные ресурсы, которые будут использоваться:

1. Розовая стрелка показывает путь данных пикселя к дисплею.

Figure 33. Graphic hardware configuration in the STM32F746G-DISCO



Плата STM32F746G-DISCO оснащена параллельной цветной жидкокристаллической панелью RGB888 с разрешением 480 x 272.

Более подробную информацию о плате STM32F746G-DISCO см. В руководстве пользователя Discovery для серии STM32F7 с MCM STM32F746NG (UM1907), доступном на веб-сайте ST Microelectronics.

На рисунке 34 показана истинная цветная панель ROCKTECH RK043FN48H (RGB888), подключенная к MCU STM32F746.

Как показано на рисунке 34, модуль дисплея подключается к MCU через две разные категории контактов:

• контакты интерфейса LTDC:

– 24-битный интерфейс RGB.

– Сигналы синхронизации: LCD_HSYNC, LCD_VSYNC, LCD_DE и LCD_CLK.

• Другие специальные контакты:

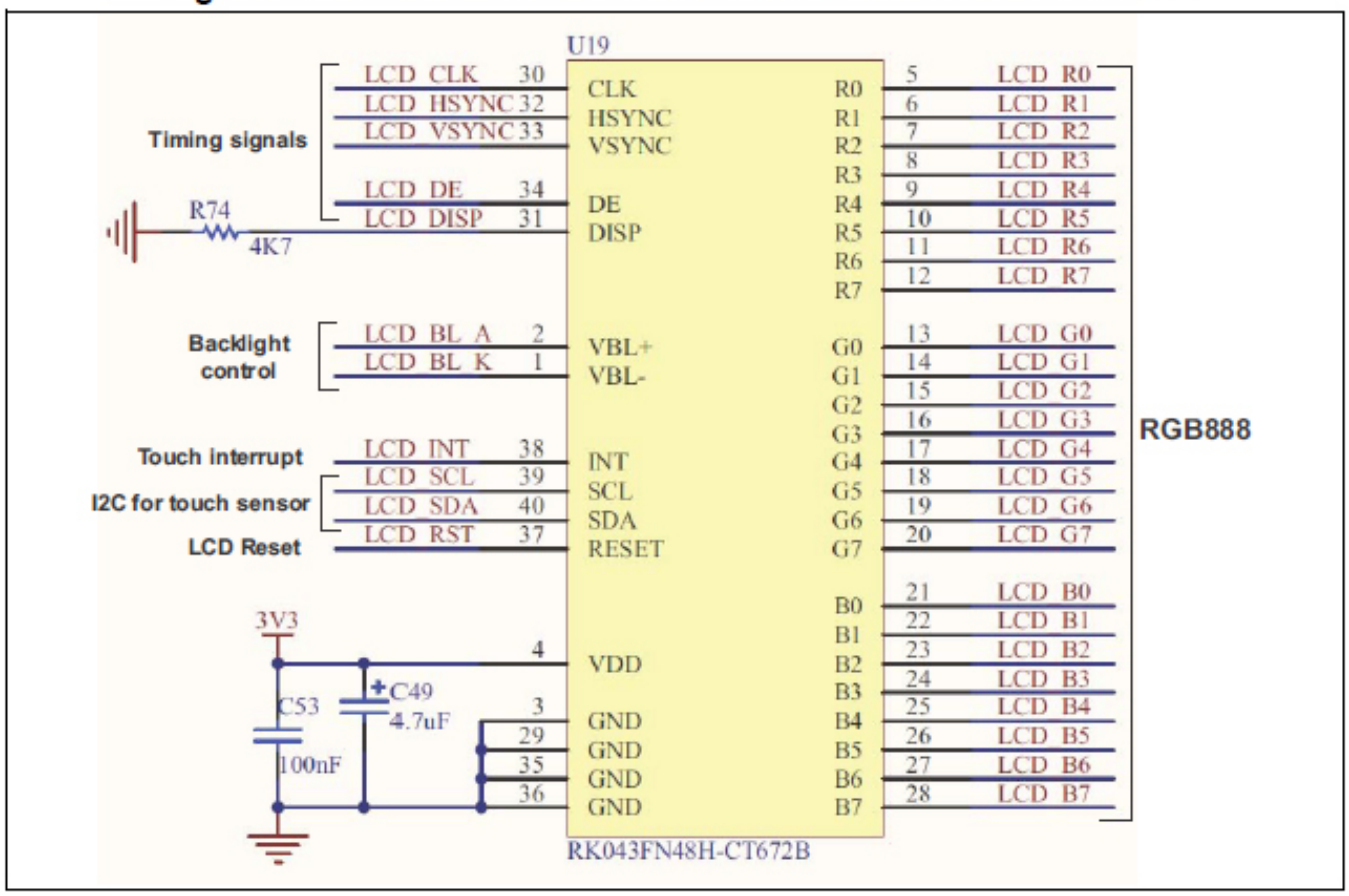
– LCD_DISP для включения / выключения режима ожидания дисплея.

– INT прерывание: позволяет сенсорному датчику генерировать прерывания.

– Интерфейс I2C для управления сенсорным датчиком.

– Вывод сброса LCD_RST, позволяющий сбросить LCD-TFT, он подключен к

Figure 34. LCD-TFT connection in the STM32F746G-DISCO board



глобальному выводу сброса MCU (NRST).

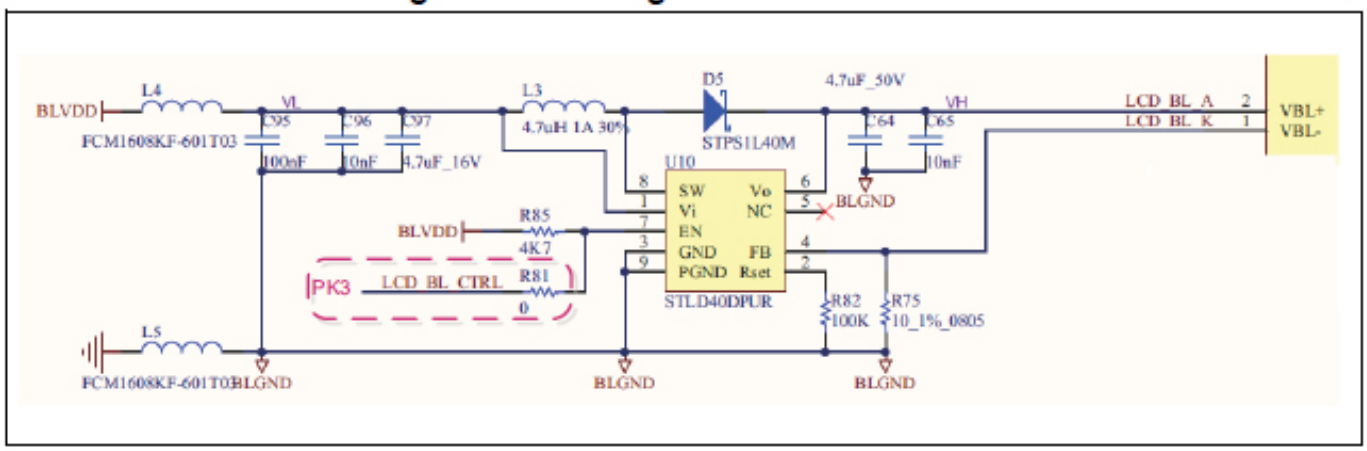
– LCD_BL_A и LCD_BL_K контакты для управления светодиодной подсветкой: подсветка управляется схемой STLD40DPUR.

Контроллер подсветки: схема STLD40DPUR, описанная на рисунке 35, представляет собой ускорительный преобразователь, работающий от 3,0 В до 5,5 В. Он может обеспечивать выходное напряжение до 37 В и может приводить в действие до десяти белых светодиодов последовательно. Более подробную информацию о контроллере подсветки см. В техническом описании STLD40D.

Высокий уровень сигнала LCD_BL_CTRL (PK3) загорается подсветкой, а низкий уровень отключается.

Примечание. *Можно изменить яркость дисплея (уменьшить интенсивность подсветки), применяя сигнал ШИМ с низкой частотой (от 1 до 10 кГц) к контакту 7 разъема STLD40D. Это действие требует переделки, так как на выводе PK3 не существует альтернативной функции выхода PWM таймера, поэтому пользователь должен удалить сопротивление R81 и подключить другой*

Figure 35. Backlight controller module



вывод GPIO с альтернативной функцией выхода PWM.

6.2.2. Как проверить, соответствует ли определенный размер дисплея аппаратной конфигурации

В этом разделе предполагается, что на этом этапе пользователь имеет желаемый размер экрана 480 x 272 при 60 Гц и с глубиной цвета 24 bpp и что следующим шагом будет выбор правильной конфигурации оборудования.

Желаемая панель дисплея

Желательным дисплеем является дисплей ROCKTECH RK043FN48H-CT672B:

• Разрешение дисплея: 480 x 272 пикселей со светодиодной подсветкой и емкостной сенсорной панелью.

• Интерфейс дисплея: 24-бит RGB888 (всего 28 сигналов).

Определение размера и местоположения буфера кадра

В зависимости от его размера и внутреннего доступного размера SRAM буфер кадра может быть расположен либо во внутренней SRAM, либо во внешней SDRAM. Общий размер встроенной SRAM для MCU STM32F746NGH6 составляет 320 Кбайт, где может использоваться SRAM1 (240 Кбайт) (см. Рисунок 10: Мастер LTDC AHB в STM32F7x6, STM32F7x7, STM32F7x8 и STM32F7x9).

Размер буфера кадра рассчитывается следующим образом:

• Для 24 bpp

– буфер кадра (Кбайт) = 480 x 272 x 3/1024 = 382,5

• Для 16 bpp

– framebuffer (Kbyte) = 480 x 272 x 2/1024 = 255

• Для 8 bpp:

– framebuffer (Kbyte) = 480 x 272/1024 = 128

Таким образом, на основе этих результатов требуемый размер буфера кадра составляет около 128 Кбайт для 8 бит/с. В этом случае буфер кадра может быть расположен во внутреннем SRAM1 (240 Кбайт). Это недействительно для случая с двойным буфером кадра, поскольку размер 128 x 2 Кбайт превышает внутренний размер SRAM.

Для глубины цвета 16 бит / с и в конфигурации с двойным буфером кадра требуемый размер буфера кадра (2 x 255 Кбайт) превышает внутренний размер SRAM, поэтому использование внешней SRAM или SDRAM является обязательным для этой конфигурации.

Для глубины цвета 24 bpp и в конфигурации с двойным буфером кадра требуемый размер буфера кадра превышает внутренний размер SRAM (2 x 382,5 Кбайт), поэтому использование внешней SRAM или SDRAM является обязательным для этой конфигурации.

Следующий шаг – проверить, может ли ширина 16-битной шины SDRAM поддерживать требуемое разрешение и глубину цвета.

Проверьте, соответствует ли разрешение 480 x 272 с 24 бит / с в соответствии с 16-разрядной конфигурацией SDRAM

На этом этапе пользователь должен был решить использовать внешнюю SDRAM, но все же должен проверить, соответствует ли ширина 16-разрядной шины SDRAM (фактическая аппаратная реализация на плате обнаружения) размер дисплея 480 x 272 @ 60 Гц и глубину цвета 24 бит / с ,

Чтобы заключить, что такая аппаратная конфигурация может поддерживать

желаемый размер и глубину цвета, пользователь должен сначала вычислить пиксельные часы.

Вычисленный LCD_CLK составляет около 9,5 МГц (для вычисления пиксельных часов см. Раздел 6.2.3: Конфигурация GPIO для LTDC.

Затем пользователь должен проверить на основе следующих параметров, если вычисленные пиксельные часы не превышают максимальный LCD_CLK, указанный в таблице 11.

– Количество используемых слоев LTDC: в этом примере используется только один уровень.

– Системная тактовая частота HCLK и скорость памяти буфера кадра: HCLK @ 200 МГц и SDRAM @ 100 МГц.

– Ширина шины памяти внешнего буфера кадра SDRAM 16 бит.

– Количество мастеров AHB, одновременно обращающихся к внешней SDRAM: 2 мастера (DMA2D и LTDC).

Ссылаясь на таблицу 11 пикселей пикселей в столбце «LTDC + DMA2D» и на одну строку слоев, тактовая частота пикселов может достигать 34 МГц для SDRAM 16-бит.

Таким образом, ширина 16-битной шины SDRAM вполне достаточна, чтобы поддерживать разрешение 480 x 272 при 60 Гц (LCD_CLK = 9,5 МГц) с глубиной цвета 24 бит.

6.2.3 Конфигурация GPIO для LTDC

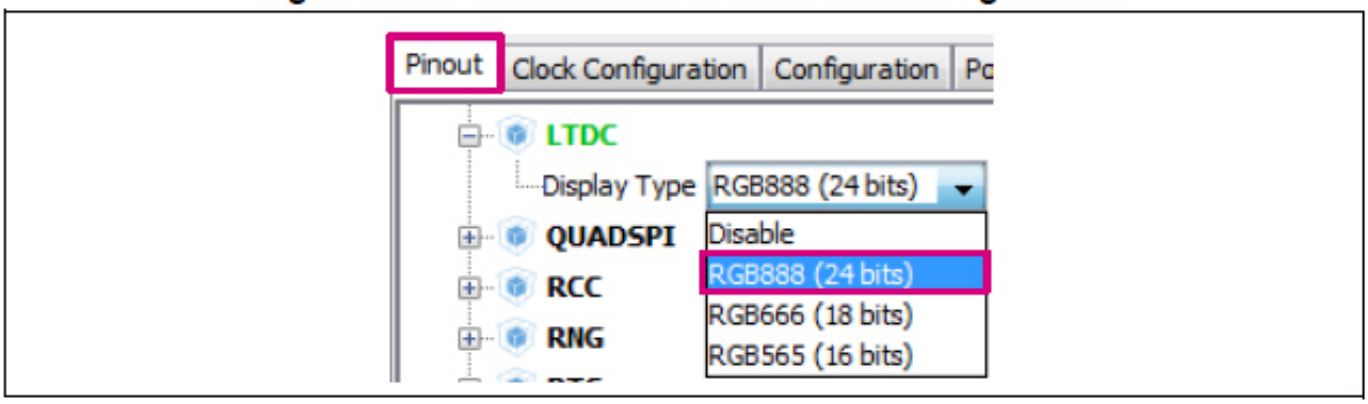
Как показано на рисунке 34, дисплей ROCKTECH RK043FN48H подключен к STM32F746xx с использованием параллельного RGB888 из 24 бит. Конфигурация контактов интерфейсов LTDC RGB

После создания проекта STM32CubeMX на вкладке «Вывод» выберите один из перечисленных аппаратных конфигураций. На рисунке 36 показано, как выбрать аппаратную конфигурацию RGB888 с помощью STM32CubeMX.

Пользователь также может настроить все GPIO, установив правильную альтернативную функцию для каждого GPIO по одному.

Если после выбора одной аппаратной конфигурации (RGB888, как показано на рисунке 36) используемые GPIO не соответствуют панели подключения пане-

Figure 36. STM32CubeMX: LTDC GPIOs configuration



ли дисплея, пользователь может изменить желаемый GPIO и настроить альтернативную функцию непосредственно на выводе. На рисунке 37 показано, например,

как настроить вручную пин-код PJ7 на альтернативную функцию LTDC_G0.

Используемые контакты подсвечиваются зеленым цветом, когда все интерфейсы GPIO интерфейса LTDC правильно настроены.

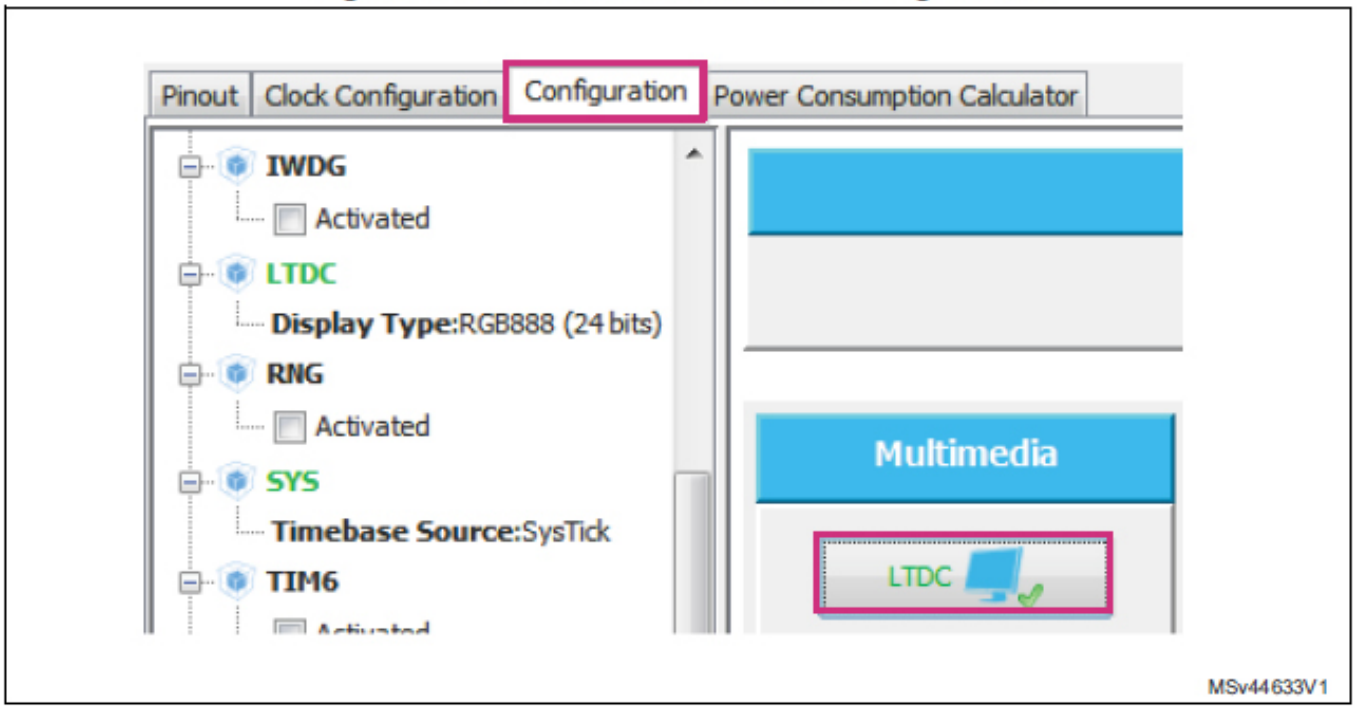
Figure 37. STM32CubeMX: PJ7 pin configuration to LTDC_G0 alternate function

После того, как настроены все контакты LTIO GPIO, пользователь должен установить свою скорость очень высоко.

Чтобы установить скорость GPIO с помощью STM32CubeMX, выберите вкладку конфигурации и нажмите кнопку LTDC, как показано на рисунке 38.

В окне конфигурации LTDC, описанном на рисунке 39, выберите все контакты LTDC, а затем установите максимальную скорость вывода на очень высокую.

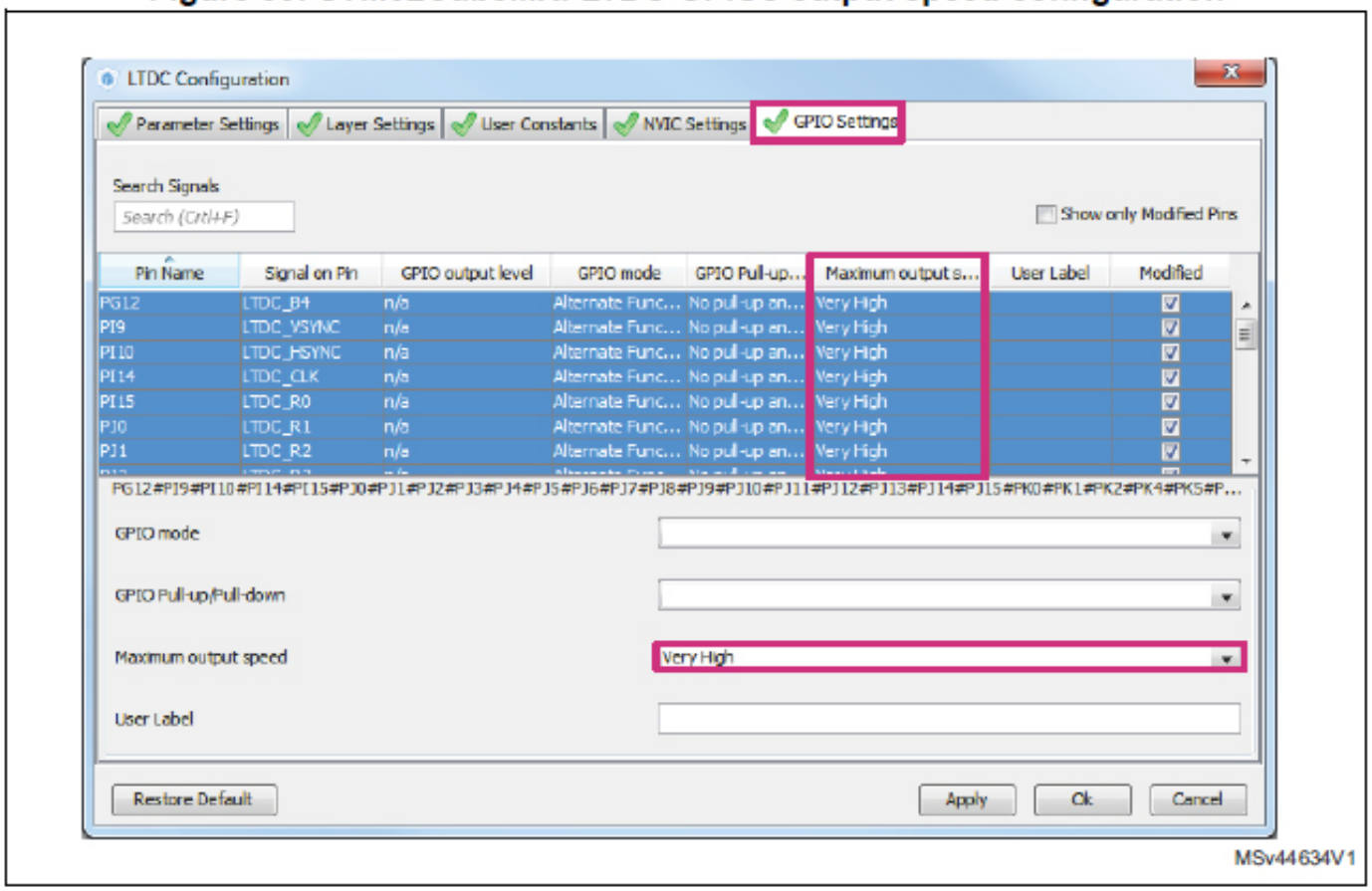
Figure 38. STM32CubeMX: LTDC configuration



Конфигурация отдельных контактов модуля дисплея

Как только все контакты интерфейса LTDC правильно настроены в отношении

Figure 39. STM32CubeMX: LTDC GPIOs output speed configuration



подключения панели LCD-TFT, пользователь должен настроить другие конкретные контакты, подключенные к дисплею (LCD_DISP, INT pin и интерфейс I2C)

Вывод LCD_DISP (вывод PI12) должен быть сконфигурирован как выходное нажатие с высоким уровнем для включения дисплея, иначе дисплей остается в режиме ожидания.

To configure the LCD_DISP pin in output mode with STM32CubeMX, in the pinout tab click on the PI12 pin then select GPIO_Output (see Figure 40).

Затем вывод LDC_DISP (PI12) должен быть настроен на высокий уровень, чтобы сделать это, на вкладке конфигурации нажмите кнопку GPIO. Затем в окне

Figure 40. STM32CubeMX: display enable pin (LCD_DISP) configuration

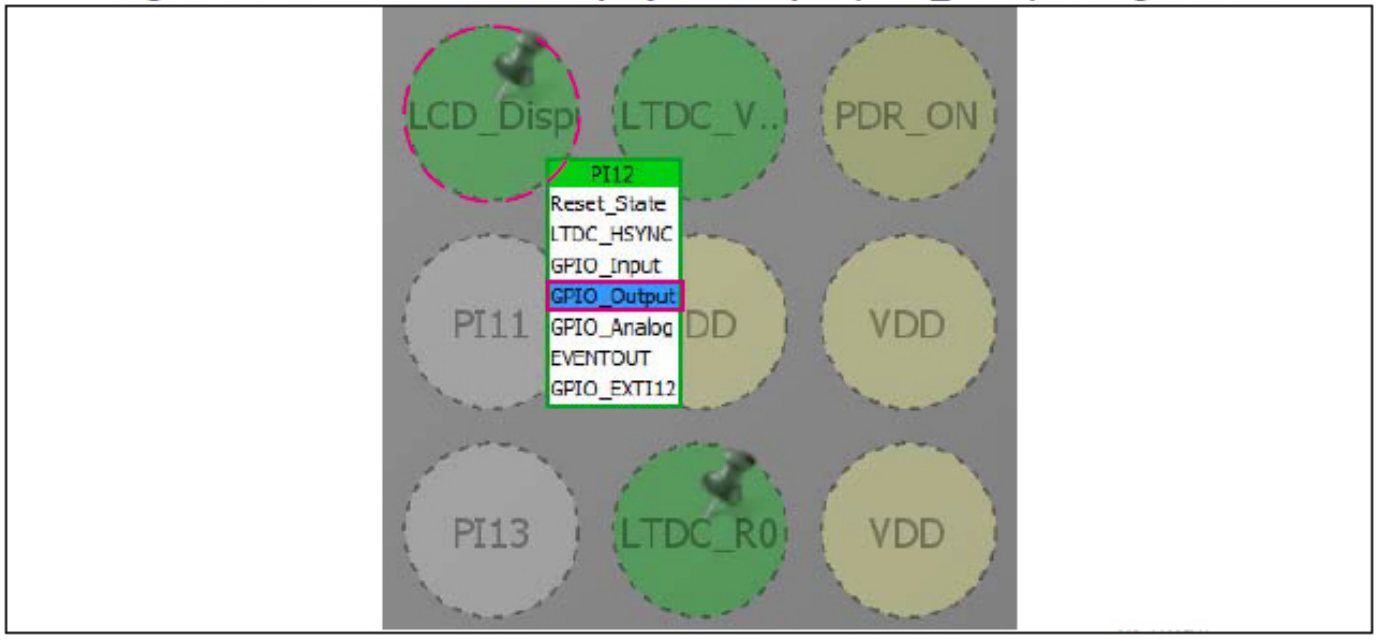
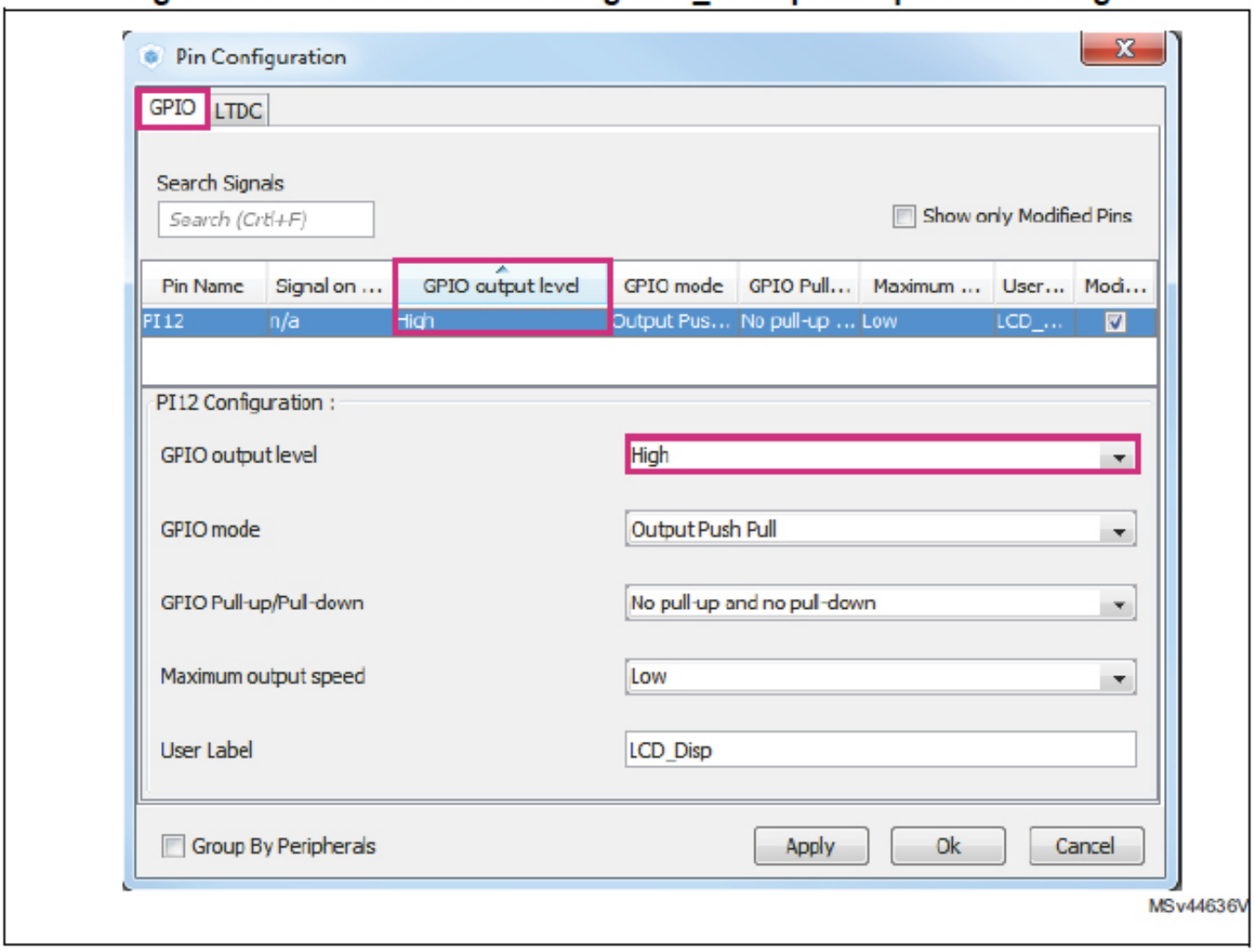


Figure 41. STM32CubeMX: setting LCD_DISP pin output level to high



конфигурации контактов установите уровень выходного сигнала GPIO на высокий, как описано на рисунке 41.

Из-за сопротивления подъема R85 подсветка находится на самом высоком уровне по умолчанию, если контакт LCD_BL_CTRL (PK3) поддерживается плавающим, поэтому нет необходимости настраивать этот вывод.

Включение прерываний LTDC

Прерывание опустошения FIFO и ошибки передачи включены в функции hal_ltdc драйвера HAL_LTDC_Init (). Таким образом, пользователь должен просто включить глобальное прерывание LTDC на стороне NVIC.

Чтобы включить глобальные прерывания LTDC с использованием STM32CubeMX, выберите вкладку конфигурации и нажмите кнопку LTDC, как показано на рисунке 38.

В окне конфигурации LTDC, показанном на рисунке 42, выберите вкладку настроек NVIC, проверьте глобальные прерывания LTDC и нажмите кнопку OK.

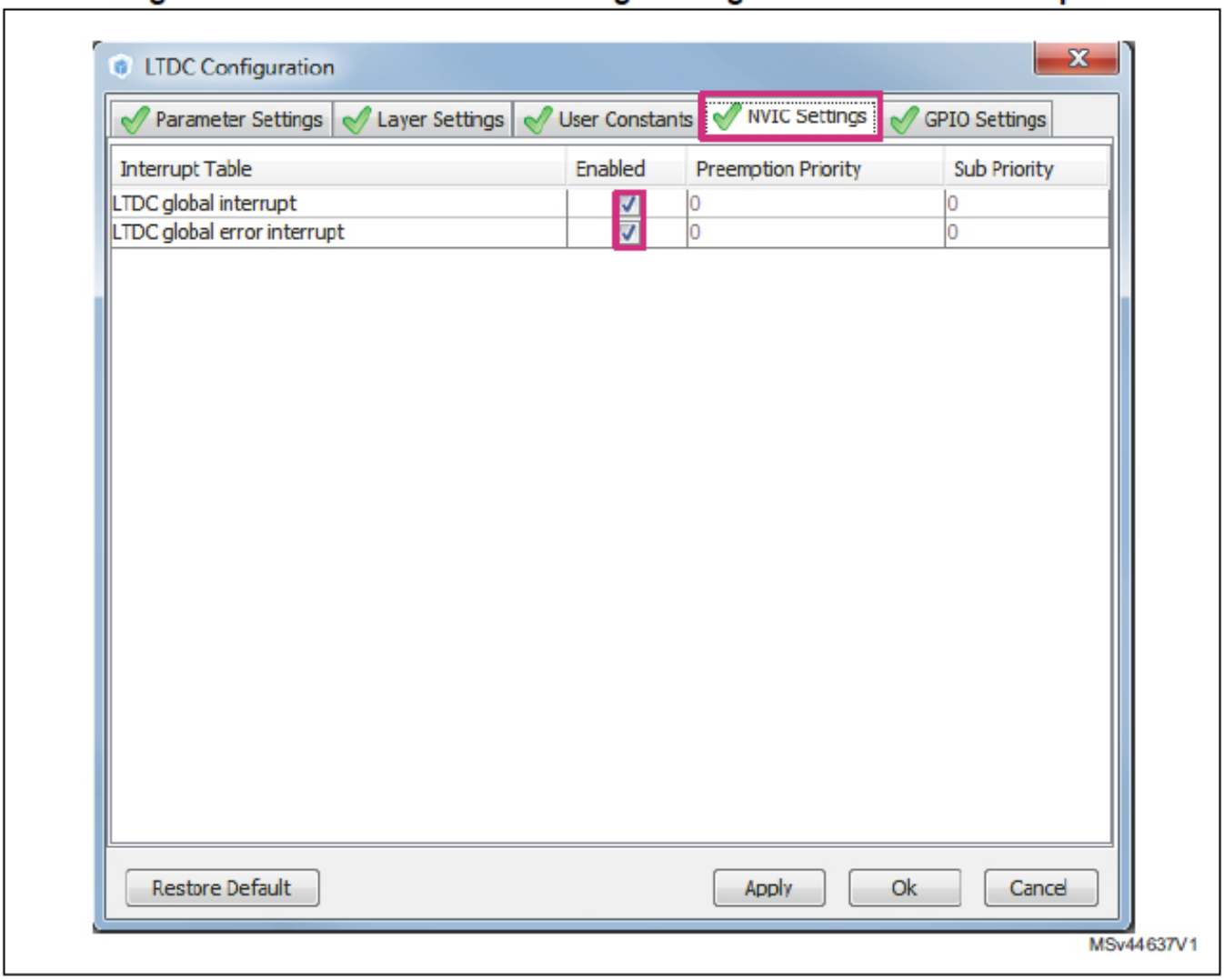
6.2.4 Внешняя конфигурация LTDC

В этом разделе показано, как настроить часы / тайминги и параметры уровня LTDC с помощью инструмента STM32CubeMX.

Конфигурация часов и таймингов LTDC

Конфигурация системных часов

Figure 42. STM32CubeMX: enabling LTDC global and error interrupts



В этом примере системные часы сконфигурированы, как показано на рисунке 44, и имеют следующую конфигурацию:

– Использование внутреннего HSI RC, где основной PLL используется в качестве источника сигнала системы.

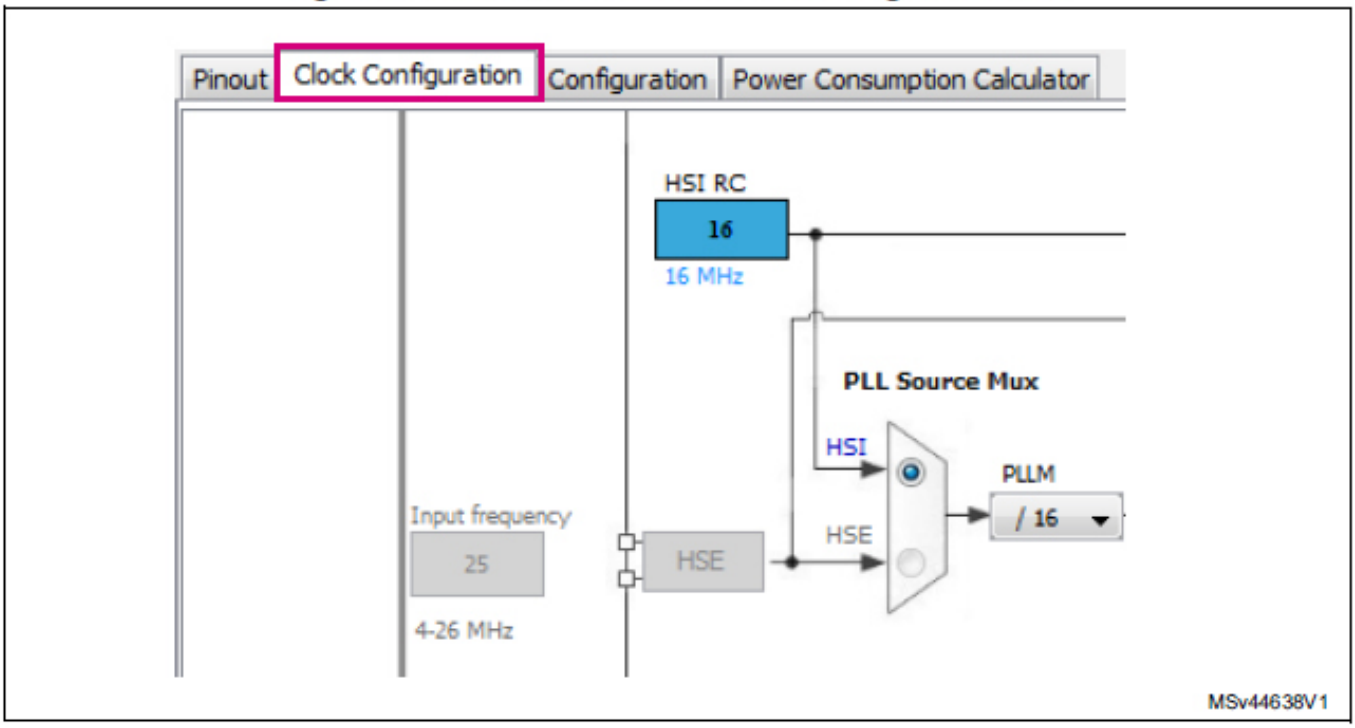
– HCLK @ 200 МГц, поэтому Cortex®-M7 и LTDC работают на частоте 200 МГц.

Примечание: *HCLK установлен на 200 МГц, но не 216 МГц, это означает, что SDRAM_FMC на максимальной скорости 100 МГц с предварительным делителем HCLK / 2.*

Чтобы настроить системные часы с помощью STM32CubeMX, выберите часы как показано на рисунке 43.

Затем, чтобы получить системные часы HCLK @ 200 МГц, установите PLL и

Figure 43. STM32CubeMX: clock configuration tab

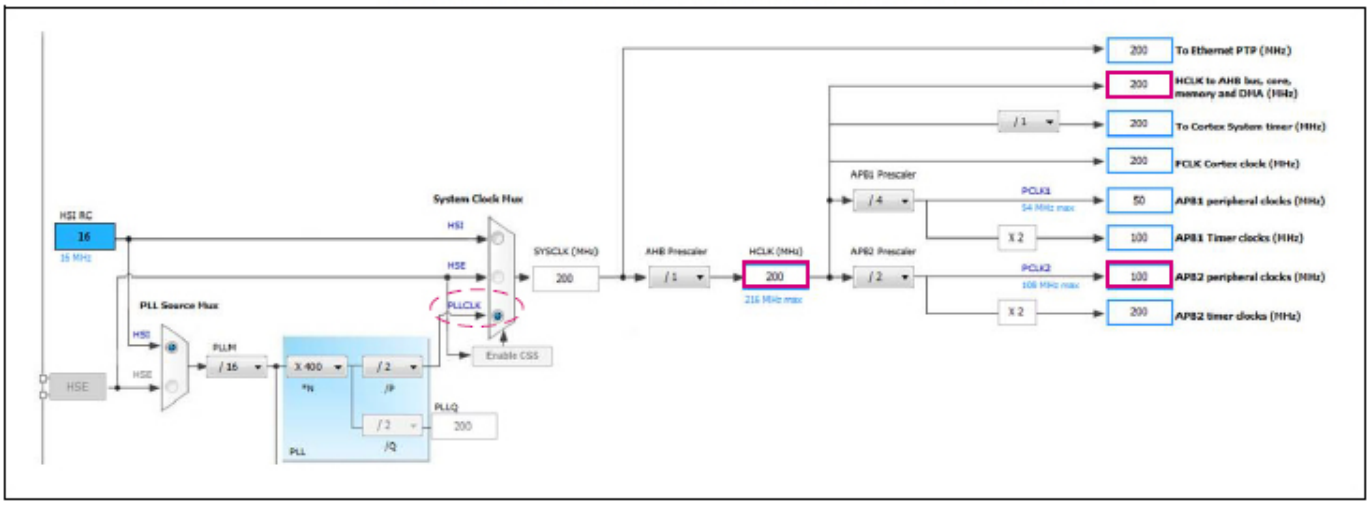


предварительные делители на вкладке конфигурации часов, как показано на рисунке 44.

Конфигурация пиксельных часов

LCD_CLK следует рассчитывать, используя параметры, найденные в таблице

Figure 44. STM32CubeMX: System clock configuration



данных дисплея.

Чтобы выполнить расчет, пользователь должен определить общую ширину и общую высоту.

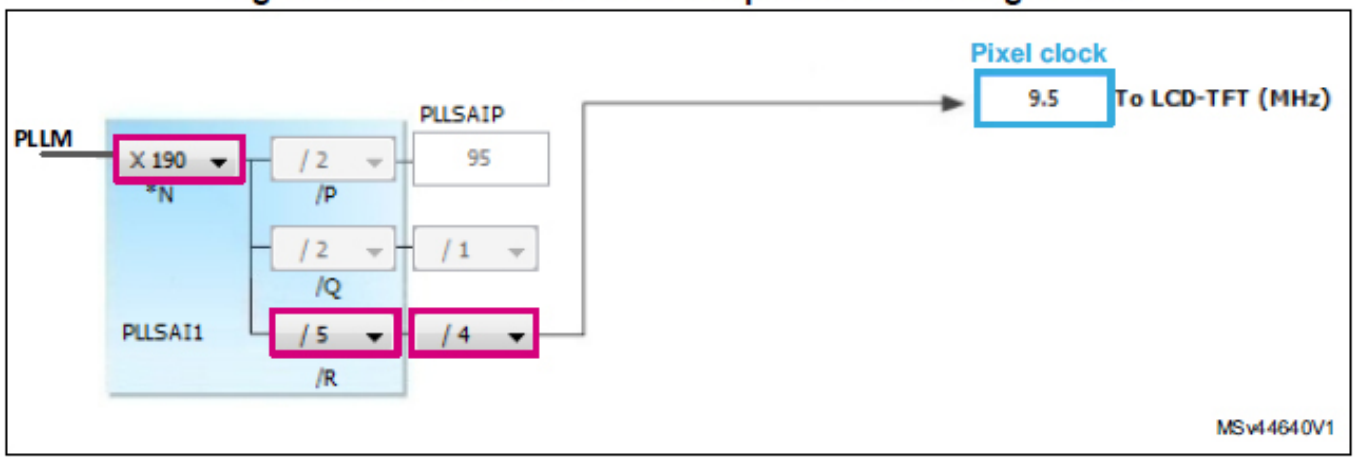
Частота пикселей вычисляется с частотой обновления 60 Гц, как показано ниже:

LCD_CLK = TOTALW x TOTALH x частота обновления (см. Раздел «Исходные параметры отображения времени» в разделе 4.7.2: Конфигурация часов и времени таймера)

LCD_CLK = 531 x 297 x 60 = 9,5 МГц

Чтобы настроить часы пикселов LTDC на 9,5 МГц с использованием

Figure 45. STM32CubeMX: LTDC pixel clock configuration



STM32CubeMX, выберите вкладку конфигурации часов, затем установите PLLSAI и предварительные делители, как показано на рисунке 45.

Конфигурация параметров синхронизации

Чтобы настроить тайминги дисплея с помощью STM32CubeMX, пользователь должен извлечь параметры синхронизации из таблицы данных устройства. В этом примере см. Выдержку из таблицы ROCKTECH в таблице 14. Рекомендуется использовать типичные тайминги дисплея.

Чтобы настроить время отображения, пользователь должен перейти на вкладку конфигурации, как показано на рисунке 38, а затем нажать кнопку LTDC.

В окне конфигурации LTDC пользователь должен выбрать вкладку «Параметры параметров» и заполнить временные значения (см. Рис. 46).

Конфигурация полярности сигналов управления LTDC

Ссылаясь на таблицу данных, HSYNC и VSYNC должны быть активными, и сигнал DE должен быть активным высоким. Поскольку сигнал DE инвертируется на выходе, он также должен быть установлен на активный минимум. Сигнал LCD_CLK не должен инвертироваться.

На рисунке 46 показана конфигурация полярности управляющего сигнала, а также конфигурация LTDC в соответствии с таблицей данных ROCKTECH.

Конфигурация параметров уровня LTDC

На этом этапе все часы и тайминги LTDC должны были быть установлены в проекте STM32CubeMX.

Пользователь должен настроить параметры уровня LTDC1 в соответствии с размером экрана и глубиной цвета.

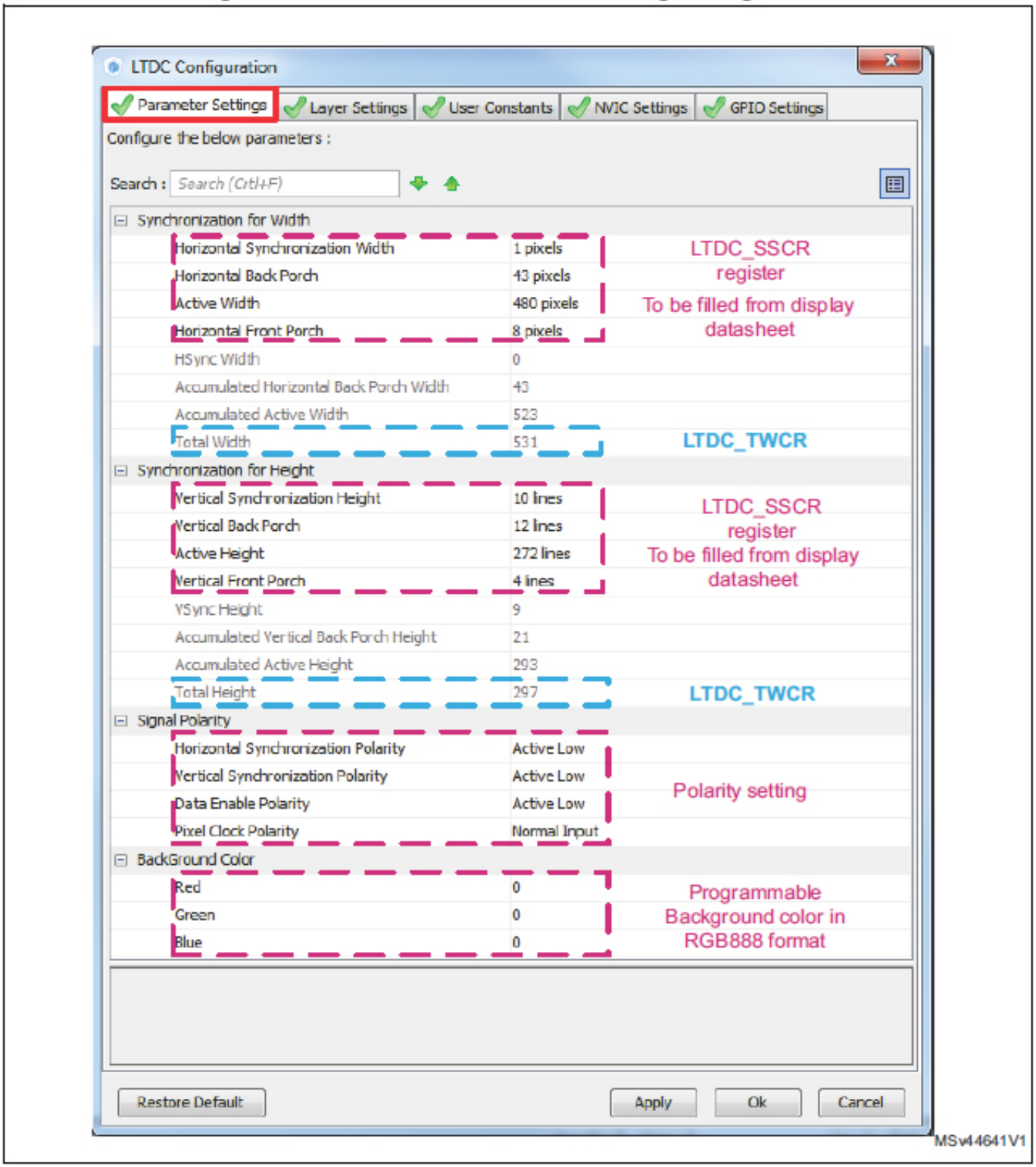
При необходимости пользователь также может включить слой2, установив «2 слоя» в поле «Количество слоев» в окне конфигурации LTDC, показанном на рисунке 47.

Чтобы установить параметры LTDC layer1 с помощью STM32CubeMX, пользователь должен выбрать вкладку конфигурации, а затем нажать кнопку LTDC, как показано на рисунке 37.

В окне конфигурации LTDC, показанном на рисунке 47, пользователь должен выбрать вкладку «Параметры слоя», установить параметры уровня LTDC1 и нажать кнопку «OK».

На этом этапе пользователь может сгенерировать проект с помощью желаемой инструментальной цепочки, нажав «Project-> Generate Code»,

Figure 46. STM32CubeMX: LTDC timing configuration



6.2.5 Отображение изображения с внутренней вспышки

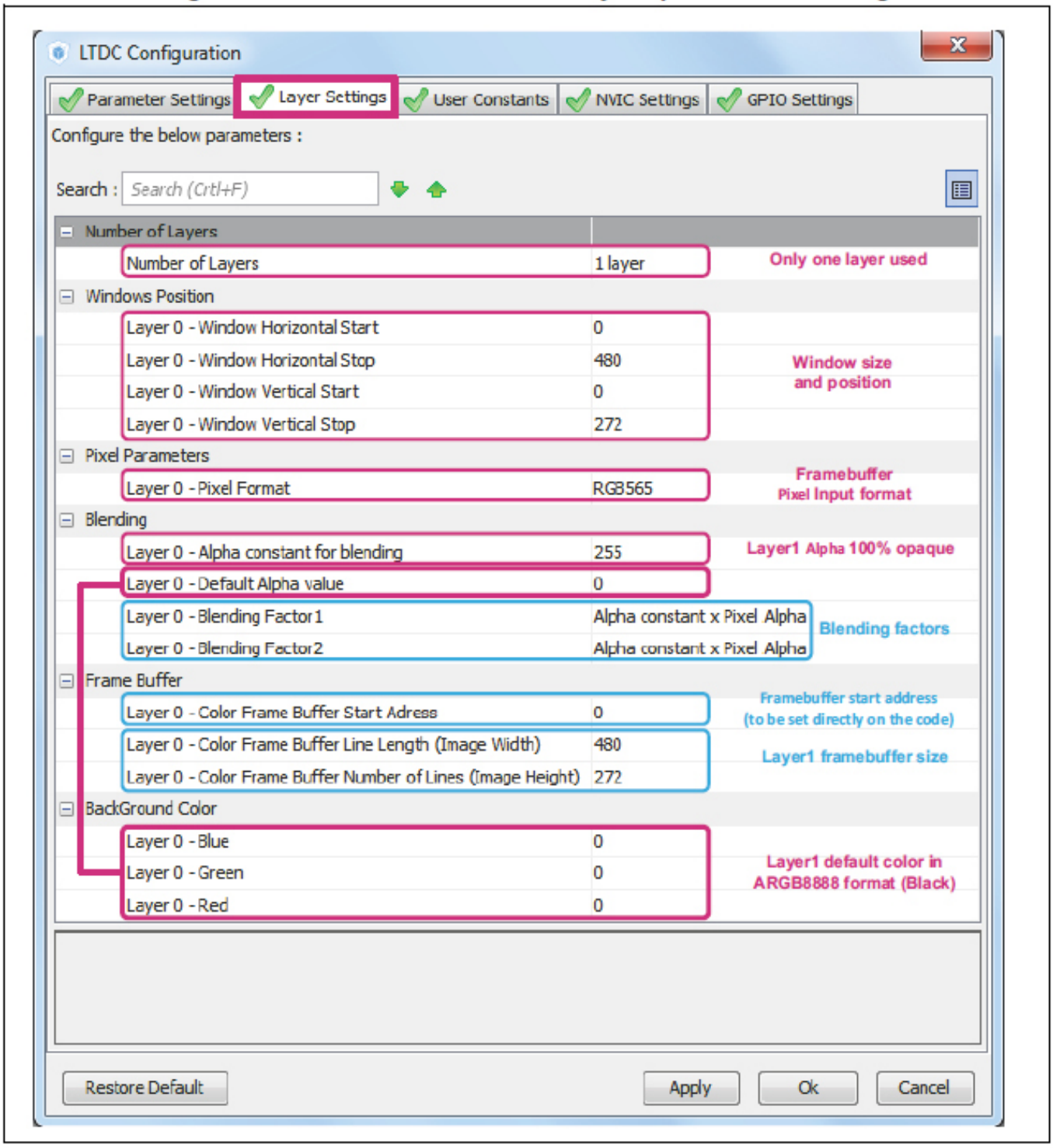
Чтобы обеспечить правильную настройку LTDC в соответствии со спецификациями панели дисплея, важно отобразить изображение с внутренней вспышки.

Чтобы сделать это, пользователь должен сначала преобразовать изображение в C или заголовочный файл и добавить его в проект.

Преобразование изображения в файл заголовка с помощью инструмента преобразования ЖК-изображения

Пользователь должен сгенерировать файл заголовка в соответствии с настроенным входным форматом пикселя уровня LTDC RGB565 (см. Раздел: Формат

Figure 47. STM32CubeMX: LTDC Layer1 parameters setting



ввода пикселей на стр. 27 и Раздел 4.8: Сохранение графических примитивов на стр. 58).

В этом примере используется инструмент LCD-Image-Converter-20161012 (подробнее об этом инструменте см. В разделе 4.8).

Чтобы преобразовать изображение, пользователь должен сначала запустить инструмент LCD-Image-Converter, затем на домашней странице, показанной на рисунке 48, нажмите «Файл-> Открыть», затем выберите файл изображения, который нужно преобразовать.

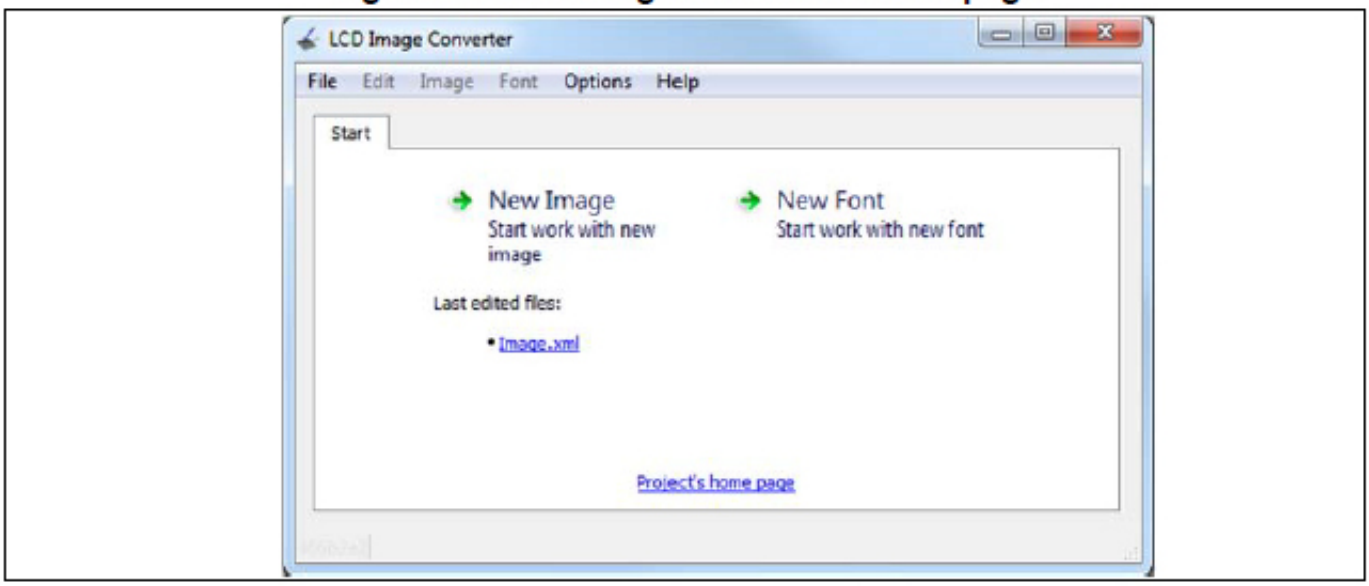
Использованный размер изображения должен быть выровнен с конфигурацией LTDC layer1 (480 x 272). Если используемый размер изображения не согласован с конфигурацией уровня LTDC1, пользователь может изменить размер изображе-

ния, перейдя в Image-> Resize или выбрав другое изображение с правильным размером.

В этом примере размер используемого изображения составляет 480 x 272, и он отображает логотип ST (см. Рис. 49).

Затем изображение отображается на домашней странице инструмента, как показано на рисунке 49.

Figure 48. LCD Image Converter: home page



Чтобы преобразовать изображение в файл заголовка, избегая проблемы с красным и синим подкачкой, описанной в разделе 4.8: Сохранение графических примитивов, пользователь должен настроить инструмент для преобразования изображения в таблицу из 32-битных слов. Для этого в меню домашней страницы

Figure 49. LCD Image Converter: image project

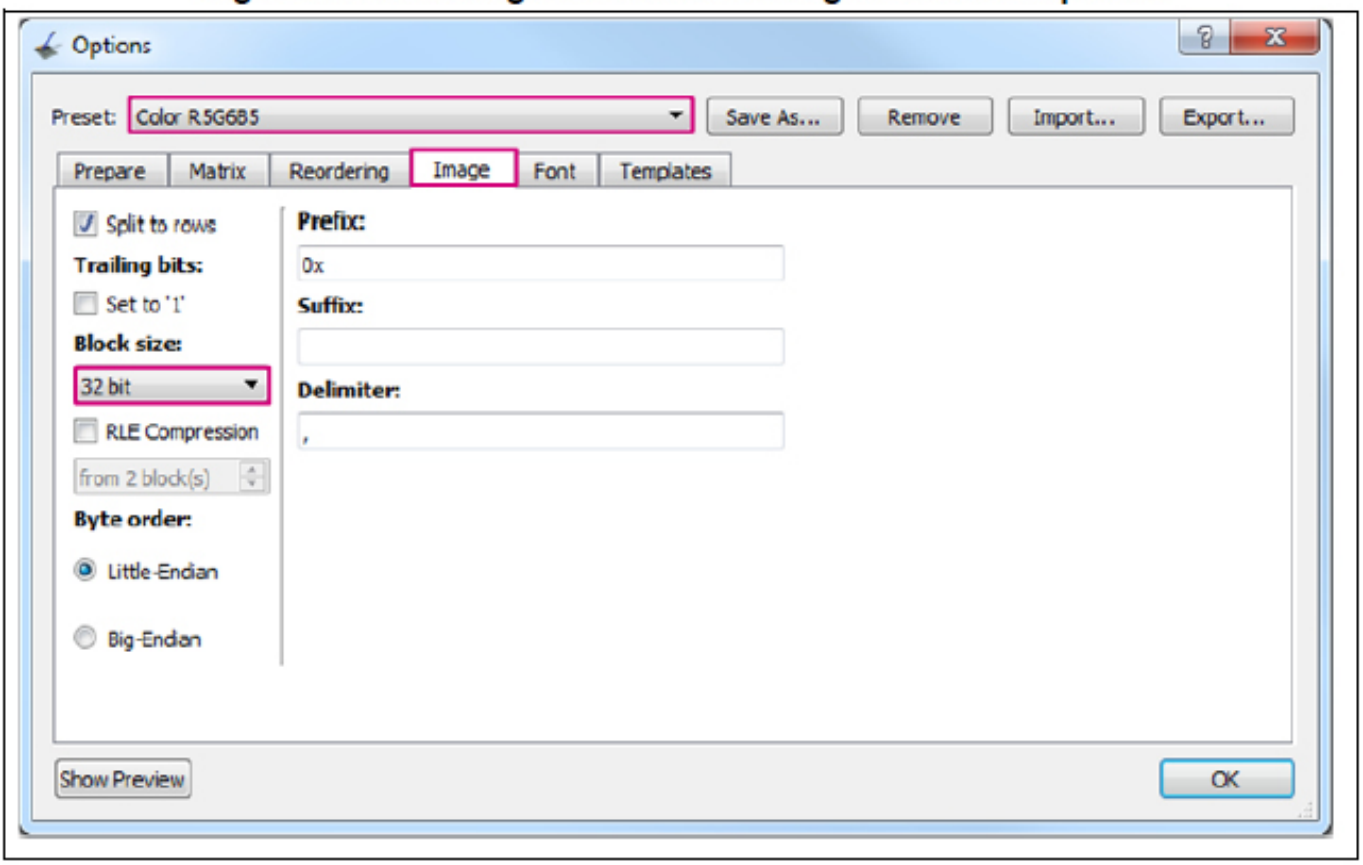
нажмите «Параметры»> «Конверсия», как показано на рисунке 49.

В окне «Параметры», показанном на рисунке 50, выберите вкладку «Изображение», затем выберите «Цвет RGB565» в поле «Предустановка», затем установите для поля «Размер блока» значение 32 бит и нажмите кнопку «OK».

Примечание. *Пользователь также может преобразовать изображение в таблицу байтов, но в этом случае он должен поменять красные и синие цвета на вкладке матрицы окон преобразования.*

Чтобы сгенерировать файл заголовка, нажмите File-> Convert. Затем в отображаемом окне, показанном на рисунке 51, установите тип файла в «Заголовки C /

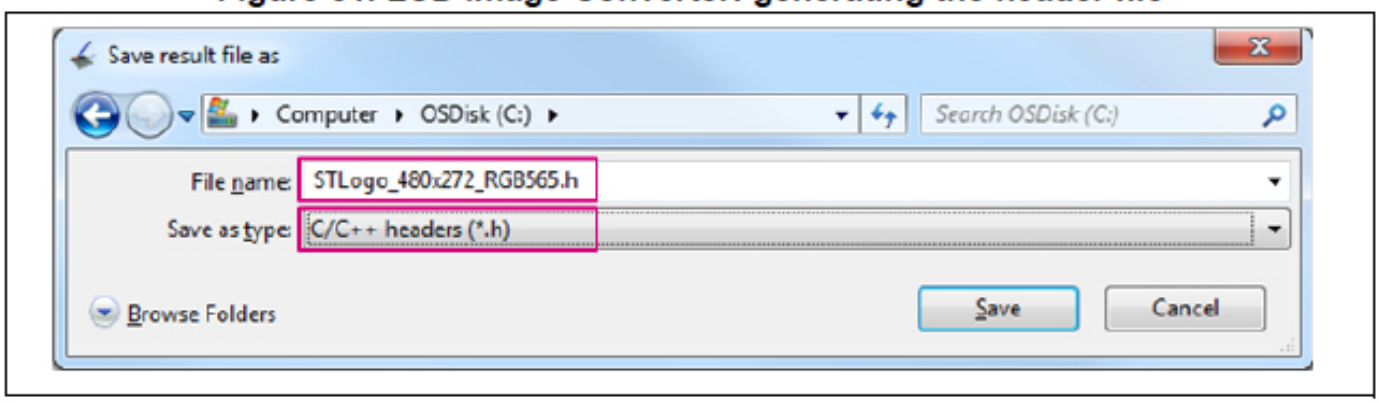
Figure 50. LCD Image Converter: setting conversion options



C ++ (* .h)», затем сохраните * .h файл в каталоге «\ Inc» (то же место, что и файл main.h) нажав кнопку «Сохранить».

Сгенерированный заголовочный файл должен быть включен в файл main.c. Он включает таблицу из 32-битных слов, где каждое слово представляет два пиксе-

Figure 51. LCD Image Converter: generating the header file



ля.

В этом заголовочном файле пользователь должен прокомментировать определение структуры, расположенное сразу после таблицы, и сохранить только определение таблицы, как показано ниже:

```
/ * Преобразованное изображение: image_data_STLogo definition * /
const uint32_t image_data_STLogo [65280] = {0xffffffff, 0xffffffff,
.....};
```

Установка начального адреса буфера кадра LTDC Layer1 во внутреннюю Flash (адрес изображения во Flash)

Сгенерированный проект STM32CubeMX должен включать в файл main.c функцию MX_LTDC_Init (), которая позволяет настроить периферийное устройство LTDC.

Чтобы отобразить изображение, пользователь должен установить начальный адрес framebuffer LTDC Layer1 по адресу изображения во внутренней Flash.

Ниже представлена функция MX_LTDC_Init () с настройкой начального адреса буфера кадра.

```
/ * Функция настройки LTDC, сгенерированная инструментом STM32CubeMX
* /
static void MX_LTDC_Init(void)
{
LTDC_LayerCfgTypeDef pLayerCfg;
hltdc.Instance = LTDC;
/* Настройка полярности сигналов управления LTDC */
hltdc.Init.HSPolarity = LTDC_HSPOLARITY_AL;
hltdc.Init.VSPolarity = LTDC_VSPOLARITY_AL;
hltdc.Init.DEPolarity = LTDC_DEPOLARITY_AL;
hltdc.Init.PCPolarity = LTDC_PCPOLARITY_IPC;
/* Конфигурация времени */
hltdc.Init.HorizontalSync = 0;
hltdc.Init.VerticalSync = 9;
hltdc.Init.AccumulatedHBP = 43;
hltdc.Init.AccumulatedVBP = 21;
hltdc.Init.AccumulatedActiveW = 523;
hltdc.Init.AccumulatedActiveH = 293;
hltdc.Init.TotalWidth = 531;
hltdc.Init.TotalHeigh = 297;
/* Цвет фона */
hltdc.Init.Backcolor.Blue = 0;
hltdc.Init.Backcolor.Green = 0;
hltdc.Init.Backcolor.Red = 0x0;
if (HAL_LTDC_Init(&hltdc) != HAL_OK)
{
Error_Handler();
}
/* Layer1 Размер окна и положение */
pLayerCfg.WindowX0 = 0;
```

```
pLayerCfg.WindowX1 = 480;
pLayerCfg.WindowY0 = 0;
pLayerCfg.WindowY1 = 272;
/* Настройка формата входного пикселя Layer1 */
pLayerCfg.PixelFormat = LTDC_PIXEL_FORMAT_RGB565;
/* Постоянная Layer1 Настройка альфы 100% непрозрачная */
pLayerCfg.Alpha = 255;
/* Layer1 Настройка коэффициентов смешивания */
pLayerCfg.BlendingFactor1 = LTDC_BLENDING_FACTOR1_PAxCA;
pLayerCfg.BlendingFactor2 = LTDC_BLENDING_FACTOR2_PAxCA;
/* Пользователь должен установить начальный адрес буфера кадра (может быть
0xC0000000, если используется внешняя SDRAM)*/
pLayerCfg.FBStartAdress = (uint32_t)&image_data_STLogo;
pLayerCfg.ImageWidth = 480;
pLayerCfg.ImageHeight = 272;
/* Layer1 Настройка цвета по умолчанию */
pLayerCfg.Alpha0 = 0;
pLayerCfg.Backcolor.Blue = 0;
pLayerCfg.Backcolor.Green = 0;
pLayerCfg.Backcolor.Red = 0;
if (HAL_LTDC_ConfigLayer(&hltdc, &pLayerCfg, 0) != HAL_OK)
{
Error_Handler();
}
}
```

После того, как LTDC правильно настроен в проекте, пользователь должен построить проект, а затем запустить его.

6.2.6 Конфигурация FMC SDRAM

Внешняя SDRAM должна быть сконфигурирована, так как она содержит буфер кадра LTDC. Для настройки FMC_SDRAM и устройства памяти SDRAM, смонтированного на плате STM32746G-Discovery, пользователь может использовать STM32CubeMX или использовать существующий драйвер BSP.

Чтобы настроить FMC_SDRAM с помощью драйвера BSP, выполните следующие действия:

1. Добавьте в проект следующие файлы: BSP stm32746g_discovery_sdram.c и stm32746g_discovery_sdram.h. Включите stm32f7xx_hal_sdram.h в файл main.c. Добавьте в проект stm32f7xx_hal_sdram.c и stm32f7xx_ll_fmc.c драйверы HAL

2. Включите модуль SDRAM в файле stm32f7xx_hal_conf.h, раскомментировав определение модуля SDRAM. Включите файл stm32f7xx_hal_sdram.h в файл main.c.

3. Вызовите функцию BSP_SDRAM_Init () в функции main ().

6.2.7 Настройка MPU и кеша

Как показано в разделе 4.6, атрибуты MPU должны быть правильно настроены,

чтобы предотвратить проблемы с графической производительностью, связанные с умозрительными чтениями Cortex®-M7 и обслуживанием кеша.

В этом разделе описывается пример конфигурации атрибута MPU в отношении конфигурации аппаратного обеспечения платы STM32F746G-DISCO.

Атрибуты памяти MPU можно легко настроить с помощью STM32CubeMX. Пример кода конфигурации MPU, сгенерированный с использованием STM32CubeMX, описан в конце этого раздела.

Пример конфигурации MPU: FMC_SDRAM

В этом примере конфигурации используется метод двойного буфера кадра; фронтбуфер помещается в банк SDRAM1, в то время как буферный буфер помещается в банк SDRAM2 в отношении оптимизации полосы SDRAM, описанной в разделе 4.5.3: Оптимизация загрузки буфера кадра LTDC из SDRAM.

Создаются следующие области MPU (FMC без пилы):

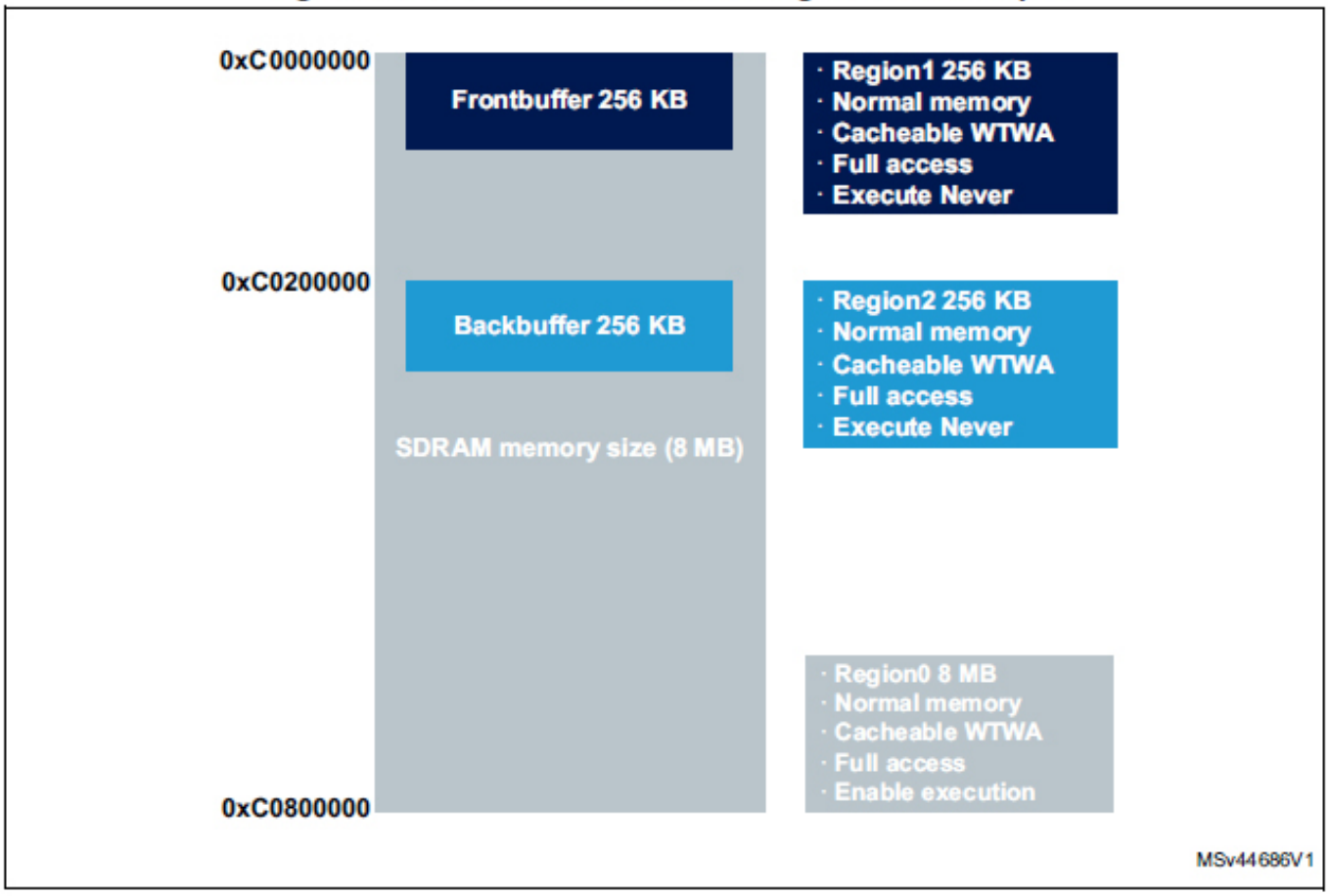
- Region0: определяет размер памяти SDRAM 8 Мбайт
- Region1: определяет фронтбуфер 256 Кбайт (16 bpp x 480 x 272), он перекрывает область 0
- Region2: определяет буферный буфер 256 Кбайт (16 bpp x 480 x 272), он перекрывает область 0

На рисунке 52 показана конфигурация MPU в области SDRAM.

Пример конфигурации MPU: Quad-SPI с отображением в памяти

В этом примере показано, как настроить MPU для интерфейса Quad-SPI. Па-

Figure 52. FMC SDRAM MPU configuration example



мять QSPI содержит графические примитивы, к ней могут обращаться Cortex®-M7, DMA2D или LTDC. Для этого интерфейс Quad-SPI должен быть установлен в ре-

жим отображения с памятью, а области MPU должны быть сконфигурированы, как описано ниже:

• Region3: определяет все адресное пространство Quad-SPI, оно должно быть настроено на строго упорядоченное, чтобы запретить любой доступ к умозрительному чтению процессора в этот регион.

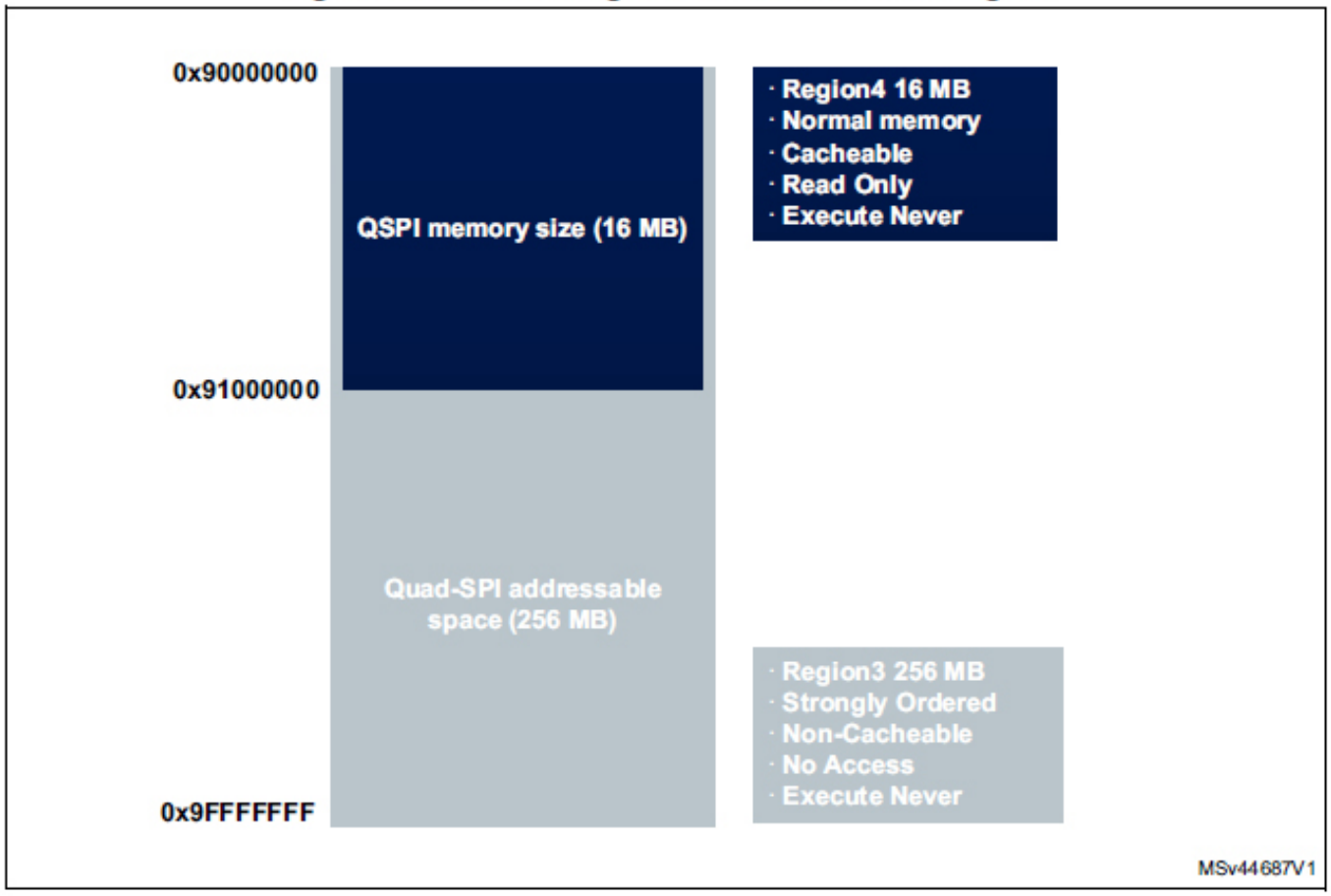
• Region4: определяет реальное пространство памяти QSPI, отражающее размер памяти, к которой может обращаться любой мастер.

На рисунке 53 показана конфигурация MPU в области Quad-SPI.

Пример конфигурации SDRAM и Quad-SPI MPU

Следующий код (сгенерированный STM32CubeMX) показывает, как устано-

Figure 53. MPU configuration for Quad-SPI region



вить атрибуты MPU для FMC_SDRAM и Quad-SPI в соответствии с ранее описанными конфигурациями.

```
/* Конфигурация MPU */
void MPU_Config(void)
{
MPU_Region_InitTypeDef MPU_InitStruct;
/* Отключает MPU */
HAL_MPU_Disable();
/* Настройте атрибуты MPU для области 0 */
/* Настроить атрибуты MPU для SDRAM на обычную память*/
MPU_InitStruct.Enable = MPU_REGION_ENABLE;
MPU_InitStruct.Number = MPU_REGION_NUMBER0;
MPU_InitStruct.BaseAddress = 0xC0000000;
```

```
  MPU_InitStruct.Size = MPU_REGION_SIZE_8MB;
  MPU_InitStruct.SubRegionDisable = 0x0;
  MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL1;
  MPU_InitStruct.AccessPermission = MPU_REGION_FULL_ACCESS;
  MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_ENABLE;
  MPU_InitStruct.IsShareable = MPU_ACCESS_NOT_SHAREABLE;
  MPU_InitStruct.IsCacheable = MPU_ACCESS_CACHEABLE;
  MPU_InitStruct.IsBufferable = MPU_ACCESS_BUFFERABLE;
  HAL_MPU_ConfigRegion(&MPU_InitStruct);
  /* Настройте атрибуты MPU для области 1 */
  /* Настроить атрибуты MPU для буфера для нормальной работы*/
  MPU_InitStruct.Enable = MPU_REGION_ENABLE;
  MPU_InitStruct.Number = MPU_REGION_NUMBER1;
  MPU_InitStruct.BaseAddress = 0xC0000000;
  MPU_InitStruct.Size = MPU_REGION_SIZE_256KB;
  MPU_InitStruct.SubRegionDisable = 0x0;
  MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL1;
  MPU_InitStruct.AccessPermission = MPU_REGION_FULL_ACCESS;
  MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_DISABLE;
  MPU_InitStruct.IsShareable = MPU_ACCESS_NOT_SHAREABLE;
  MPU_InitStruct.IsCacheable = MPU_ACCESS_CACHEABLE;
  MPU_InitStruct.IsBufferable = MPU_ACCESS_BUFFERABLE;
  HAL_MPU_ConfigRegion(&MPU_InitStruct);
  /* Настройте атрибуты MPU для области 2 */
  /* Настройте атрибуты MPU для резервного буфера в обычную память*/
  MPU_InitStruct.Enable = MPU_REGION_ENABLE;
  MPU_InitStruct.Number = MPU_REGION_NUMBER2;
  MPU_InitStruct.BaseAddress = 0xC0200000;
  MPU_InitStruct.Size = MPU_REGION_SIZE_256KB;
  MPU_InitStruct.SubRegionDisable = 0x0;
  MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL1;
  MPU_InitStruct.AccessPermission = MPU_REGION_FULL_ACCESS;
  MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_DISABLE;
  MPU_InitStruct.IsShareable = MPU_ACCESS_NOT_SHAREABLE;
  MPU_InitStruct.IsCacheable = MPU_ACCESS_CACHEABLE;
  MPU_InitStruct.IsBufferable = MPU_ACCESS_BUFFERABLE;
  HAL_MPU_ConfigRegion(&MPU_InitStruct);
  /* Настройте атрибуты MPU для области 3 */
  /* Настройте атрибуты MPU для области Quad-SPI на сильно упорядоченную
память*/
  MPU_InitStruct.Enable = MPU_REGION_ENABLE;
  MPU_InitStruct.Number = MPU_REGION_NUMBER3;
  MPU_InitStruct.BaseAddress = 0x90000000;
  MPU_InitStruct.Size = MPU_REGION_SIZE_256MB;
  MPU_InitStruct.SubRegionDisable = 0x0;
  MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL0;
```

```
MPU_InitStruct.AccessPermission = MPU_REGION_NO_ACCESS;
MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_DISABLE;
MPU_InitStruct.IsShareable = MPU_ACCESS_NOT_SHAREABLE;
MPU_InitStruct.IsCacheable = MPU_ACCESS_NOT_CACHEABLE;
MPU_InitStruct.IsBufferable = MPU_ACCESS_NOT_BUFFERABLE;
HAL_MPU_ConfigRegion(&MPU_InitStruct);
/* Настройте атрибуты MPU для области 4 */
/* Настроить атрибуты MPU для памяти QSPI на обычную память*/
MPU_InitStruct.Enable = MPU_REGION_ENABLE;
MPU_InitStruct.Number = MPU_REGION_NUMBER4;
MPU_InitStruct.BaseAddress = 0x90000000;
MPU_InitStruct.Size = MPU_REGION_SIZE_16MB;
MPU_InitStruct.SubRegionDisable = 0x0;
MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL0;
MPU_InitStruct.AccessPermission = MPU_REGION_PRIV_RO;
MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_DISABLE;
MPU_InitStruct.IsShareable = MPU_ACCESS_NOT_SHAREABLE;
MPU_InitStruct.IsCacheable = MPU_ACCESS_CACHEABLE;
MPU_InitStruct.IsBufferable = MPU_ACCESS_NOT_BUFFERABLE;
HAL_MPU_ConfigRegion(&MPU_InitStruct);
/* Включите MPU */
HAL_MPU_Enable(MPU_PRIVILEGED_DEFAULT);
}
```

6.3 Справочные платы с LCD-TFT-панелью

ST предлагает широкий спектр эталонных плат, таких как платы NUCLEO, Discovery и EVAL, многие из которых включают в себя панели дисплея. Для эталонных плат STM32 с встроенным дисплеем, но не для встраивания LTDC, интерфейс DBI (FMC или SPI) используется для подключения STM32 к дисплею.

Для других плат STM32 LTDC используется для взаимодействия с панелью дисплея.

Эти эталонные платы могут использоваться для оценки графических возможностей в конкретных конфигурациях оборудования / программного обеспечения.

В таблице 17 приведены эталонные платы STM32, встраивающие LTDC и имеющие встроенную панель TFT-LCD.

1. Доступная плата B-LCDAD-HDMI1 (приобретается отдельно) позволяет конвертировать DSI в формат HDMI для подключения дисплеев HDMI.

Table 17. STM32 reference boards with embedding LTDC and featuring an on-board LCD-TFT panel

Product	Board	TFT-LCD panel					Internal SRAM (Kbyte)	External SDRAM	External SRAM	QSPI (MB)
		Interface	Size (Inch)	Reso-lution	Color depth	Touch sensor				
STM32 F429xx/ STM32 F439xx	32F429I DISCOVERY	DPI	2.4	240 x 320	RGB666	Resistive	256	16-bit	NA	NA
	STM32439I-EVAL2	DPI	5.7	640 x 480	RGB666	Capacitive		32-bit	16-bit	NA
	STM32429I-EVAL1	DPI	4.3	480 x 272	RGB888	Resistive				
STM32 F469xx/ STM32 F479xx	32F469IDISCOVERY	MIPI-DSI	4	800 x 480	RGB888	Capacitive	384	32-bit	NA	16
	STM32469I-EVAL(1)	MIPI-DSI	4	800 x 480	RGB888	Capacitive		32-bit	16-bit	64
STM32 F7x6 line	32F746GDISCOVERY	DPI	4.3	480 x 272	RGB888	Capacitive	320	16-bit	NA	16
	STM32746G-EVAL	DPI	5.7	640 x 480	RGB666	Capacitive		32-bit	16-bit	64
		DPI	4.3	480 x 272	RGB888	Resistive				
STM32 F7x9 line(1)	STM32F769I-DISCO(2)	MIPI-DSI	4	800x480	RGB888	Capacitive	512	32-Bit	NA	64
	STM32F779I-EVAL STM32F769I-EVAL	MIPI-DSI	4	800x480	RGB888	Capacitive		32-Bit	16-Bit	64

Плата адаптера DSI-LCD B-LCDAD-RPI1 (приобретается отдельно) обеспечивает гибкий разъем от материнской платы микроконтроллера до стандартного разъема дисплея (TE 1-1734248).

2. Еще одна панель обнаружения доступна STM32F769I-DISC1, но без встроенного дисплея, дисплей можно приобрести отдельно как B-LCD40-DSI1дополнительный код.

7 Поддерживаемые панели дисплея

Контроллер дисплея включает в себя очень гибкий интерфейс, который предоставляет ниже функции, которые позволяют MCU STM32 поддерживать несколько параллельных панелей дисплея (таких как TFT-LCD и OLED-дисплеи), доступных на рынке:

- Различные полярности сигнала.
- Программируемые тайминги и разрешения.

Частота пикселов на панели дисплея (как указано в техническом описании производителя) не должна превышать максимальные часы пикселя STM32. Поэтому пользователь должен ссылаться на таблицу данных дисплея, чтобы гарантировать, что работающие часы панели меньше, чем максимальные часы пикселя.

8 Часто задаваемые вопросы

В этом разделе представлены наиболее часто задаваемые вопросы, касающиеся

использования и конфигураций LTDC.

Вопрос	Ответ
Каково максимальное поддерживаемое разрешение LTDC?	Абсолютного максимального разрешения нет, поскольку оно зависит от нескольких параметров, таких как: – глубина цвета – используемая ширина шины SDRAM – скорость работы системы (HCLK) – количество мастеров AHB, одновременно обращающихся к памяти, используемой для буфера кадра. См. Раздел 4.2.2: Проверка совместимости дисплеев с учетом требований к пропускной способности памяти.
Поддерживает ли STM32F4 Series или STM32F7 Series разрешение 1280 x 720p 60 Гц?	Да, см. Примеры в разделе 4.2.2: Проверка совместимости дисплеев с учетом требований к пропускной способности памяти.
Какую ширину шины SDRAM следует использовать для определенного разрешения?	Не существует конкретной конкретной ширины шины, она зависит от разрешения, глубины цвета и того, является ли SDRAM совместно с другими мастерами AHB или нет. Чем выше ширина шины SDRAM, тем лучше. 32-разрядная SDRAM обеспечивает наилучшие возможности.
Как получить максимальное поддерживаемое разрешение для определенного оборудования?	См. Раздел 4.2.2: Проверка совместимости дисплеев с учетом требований к пропускной способности памяти.
Поддерживает ли LTDC дисплеи OLED?	Да, если OLED-дисплей имеет параллельный интерфейс RGB.
Поддерживает ли LTDC дисплеи STN?	Нет, STN-дисплеи не поддерживаются LTDC.
Почему изображение отображается с красными и синими цветами?	Это связано с тем, что изображение не сохраняется в памяти в соответствии с форматом ввода настроенного пикселя (см. Раздел 4.8.1: Преобразование изображений в файлы C).
Поддерживает ли LTDC шкалу серого?	Да, это возможно, используя режим L8 и используя правильный CLUT (R = G = B).

Почему дисплей плохой (отображает плохие визуальные эффекты)?	Многие факторы могут привести к плохому визуальному эффекту, пользователь может выполнить следующие проверки: – Проверьте правильность инициализации / настройки используемого дисплея (для некоторых дисплеев требуется последовательность инициализации / конфигурации) – Проверьте, правильно ли установлены тайминги LTDC и параметры уровня (см. Пример в разделе 6.2.4) – Отображение изображения непосредственно из внутренней Flash (см. Пример в разделе 6.2.5) – Проверьте, нет ли синхронизации между LTDC и обновлением буфера кадра (по DMA2D или CPU), см. Раздел 4.4.2.

9 Заключение

MCU STM32 обеспечивают очень гибкий контроллер дисплея, позволяющий взаимодействовать с широким спектром дисплеев по более низкой цене и предлагая высокие характеристики.

Благодаря интеграции в интеллектуальную архитектуру, LTDC автономно извлекает графические данные из буфера кадра и выводит его на дисплей без вмешательства ЦП.

LTDC может продолжать получать графические данные и управлять дисплеем, когда процессор находится в режиме SLEEP, что идеально подходит для маломощных и мобильных приложений, таких как smartwatches.

В этой заметке приложения описаны графические возможности STM32 и представлены некоторые соображения и рекомендации, чтобы в полной мере использовать интеллектуальную архитектуру системы.