Fvoid LL\_I2C\_Enable(I2C\_TypeDef \*I2Cx)

Разрешает работу I2C (PE = 1)

void LL\_I2C\_Disable(I2C\_TypeDef \*I2Cx)

Запрещает работу I2C. Когда (PE = 0), I2C SCL и SDA линии освобождаются. Внутренний автомат состояний и биты статуса возвращаются в их значения по-умолчанию. При очистке, PE должен оставаться в низком уровне по крайней мере 3 APB тактовых цикла.

uint32\_t LL\_I2C\_IsEnabled(I2C\_TypeDef \*I2Cx)

void LL\_I2C\_ConfigFilters(I2C\_TypeDef \*I2Cx, uint32\_t AnalogFilter, uint32\_t DigitalFilter)

Настройка шумовых фильтров (аналоговых и цифровых). Фильтры могут программироваться только для отключенных I2C (PE = 0).

AnalogFilter может принимать значения:

LL\_I2C\_ANALOGFILTER\_ENABLE, LL\_I2C\_ANALOGFILTER\_DISABLE.

DigitalFilter этот параметр должен быть значением между Min\_Data = 0x00 (цифровой фильтр отключен) и Max\_Data=0x0F (цифровой фильтр включен и фильтрующая способность до **15\*ti2cclk**). Цифровой фильтр будет фильтровать выбросы с длительностью до **DNF[3:0]\*ticclk**.

void LL\_I2C\_SetDigitalFilter(I2C\_TypeDef \*I2Cx, uint32\_t DigitalFilter)

uint32\_t LL\_I2C\_GetDigitalFilter(I2C\_TypeDef \*I2Cx)

void LL\_I2C\_EnableAnalogFilter(I2C\_TypeDef \*I2Cx)

void LL\_I2C\_DisableAnalogFilter(I2C\_TypeDef \*I2Cx)

uint32\_t LL\_I2C\_IsEnabledAnalogFilter(I2C\_TypeDef \*I2Cx)

void LL\_I2C\_EnableDMAReq\_TX(I2C\_TypeDef \*I2Cx)

Разрешает запросы на передачу DMA.

void LL\_I2C\_DisableDMAReq\_TX(I2C\_TypeDef \*I2Cx)

Запрещает запросы на передачу DMA.

uint32\_t LL\_I2C\_IsEnabledDMAReq\_TX(I2C\_TypeDef \*I2Cx)

void LL\_I2C\_EnableDMAReq\_RX(I2C\_TypeDef \*I2Cx)

Разрешает запросы DMA на приём.

void LL\_I2C\_DisableDMAReq\_RX(I2C\_TypeDef \*I2Cx)

Запрещает запросы DMA на приём.

uint32\_t LL\_I2C\_IsEnabledDMAReq\_RX(I2C\_TypeDef \*I2Cx)

uint32\_t LL\_I2C\_DMA\_GetRegAddr(I2C\_TypeDef \*I2Cx, uint32\_t Direction)

Получить адрес регистра данных, используемый для DMA передачи.

Direction может принимать значения:

LL\_I2C\_DMA\_REG\_DATA\_TRANSMIT, LL\_I2C\_DMA\_REG\_DATA\_RECEIVE.

void LL\_I2C\_EnableClockStretching(I2C\_TypeDef \*I2Cx)

Разрешить Clock stretching. Этот бит может быть сброшен только для отключенных I2C.

void LL\_I2C\_DisableClockStretching(I2C\_TypeDef \*I2Cx)

Запретить Clock Stretching. Этот бит может быть установлен только для отключенных I2C.

uint32\_t LL\_I2C\_IsEnabledClockStretching(I2C\_TypeDef \*I2Cx)

void LL\_I2C\_SetMasterAddressingMode(I2C\_TypeDef \*I2Cx, uint32\_t AddressingMode)

Настроить мастер для работы с 7-битным или 10-битным режимом адресации. Изменение режима не позволяется, когда START бит установлен.

AddressingMode может принимать значения:

LL\_I2C\_ADDRESSING\_MODE\_7BIT, LL\_I2C\_ADDRESSING\_MODE\_10BIT.

uint32\_t LL\_I2C\_GetMasterAddressingMode(I2C\_TypeDef \*I2Cx)

void LL\_I2C\_SetTiming(I2C\_TypeDef \*I2Cx, uint32\_t Timing)

Настроить времена установки и удержания SDA, и длительность периода высокого и низкого уровня SCL. Этот бит может быть запрограммирован, только когда I2C отключен (PE = 0).

Timing может принимать значения от Min\_Data=0 до Max\_Data=0xFFFFFFFF. Этот параметр вычисляется утилитой STM32CubeMX.

uint32\_t LL\_I2C\_GetTimingPrescaler(I2C\_TypeDef \*I2Cx)

Получить настройку прескалера тайминга.

uint32\_t LL\_I2C\_GetClockLowPeriod(I2C\_TypeDef \*I2Cx)

Получить настройку длительности периода низкого уровня SCL.

uint32\_t LL\_I2C\_GetClockHighPeriod(I2C\_TypeDef \*I2Cx)

Получить настройку длительности периода высокого уровня SCL.

uint32\_t LL\_I2C\_GetDataHoldTime(I2C\_TypeDef \*I2Cx)

Получить время удержания SDA.

uint32\_t LL\_I2C\_GetDataSetupTime(I2C\_TypeDef \*I2Cx)

Получить время установки SDA.

void LL\_I2C\_SetMode(I2C\_TypeDef \*I2Cx, uint32\_t PeripheralMode)

Настроить периферийный режим.

PeripheralMode может принимаеть значения:

LL\_I2C\_MODE\_I2C, LL\_I2C\_MODE\_SMBUS\_HOST, LL\_I2C\_MODE\_SMBUS\_DEVICE, LL\_I2C\_MODE\_SMBUS\_DEVICE\_ARP.

uint32\_t LL\_I2C\_GetMode(I2C\_TypeDef \*I2Cx)

NB: line 1207.

**Функции разрешения/запрещения прерываний**:

void LL\_I2C\_EnableIT\_TX(I2C\_TypeDef \*I2Cx)

Разрешения прерывания TXIE.

void LL\_I2C\_DisableIT\_TX(I2C\_TypeDef \*I2Cx)

Запрещение прерывания TXIE.

uint32\_t LL\_I2C\_IsEnabledIT\_TX(I2C\_TypeDef \*I2Cx)

void LL\_I2C\_EnableIT\_RX(I2C\_TypeDef \*I2Cx)

Разрешение прерывания RXIE.

void LL\_I2C\_DisableIT\_RX(I2C\_TypeDef \*I2Cx)

Запрещение прерывания RXNE.

uint32\_t LL\_I2C\_IsEnabledIT\_RX(I2C\_TypeDef \*I2Cx)

void LL\_I2C\_EnableIT\_NACK(I2C\_TypeDef \*I2Cx)

Разрешить прерывание неподтверждения приёма NACKIE.

void LL\_I2C\_DisableIT\_NACK(I2C\_TypeDef \*I2Cx)

Запретить прерывание неподтверждения приёма NACKIE.

uint32\_t LL\_I2C\_IsEnabledIT\_NACK(I2C\_TypeDef \*I2Cx)

void LL\_I2C\_EnableIT\_TC(I2C\_TypeDef \*I2Cx)

Разрешение прерывания завершения передачи TCIE. Любое из этих событий приводят к генерации прерывания: Transfer Complete, Transfer Complete Reload.

void LL\_I2C\_DisableIT\_TC(I2C\_TypeDef \*I2Cx)

Запрещение прерывания завершения передачи TCIE.

uint32\_t LL\_I2C\_IsEnabledIT\_TC(I2C\_TypeDef \*I2Cx)

void LL\_I2C\_EnableIT\_ERR(I2C\_TypeDef \*I2Cx)

Разрешение прерываний ошибок ERRIE. Любая из этих ошибок приводит к генерации прерывания:

Arbitration Loss (ARLO), Bus Error detection (BERR), Overrun/Underrun (OVR), SMBus Timeout detection (TIMEOUT)

SMBus PEC error detection (PECERR), SMBus Alert pin event detection (ALERT)

void LL\_I2C\_DisableIT\_ERR(I2C\_TypeDef \*I2Cx)

Запрещение прерываний ошибок ERRIE.

uint32\_t LL\_I2C\_IsEnabledIT\_ERR(I2C\_TypeDef \*I2Cx)

**Функции индикации факта наличия прерывания**:

uint32\_t LL\_I2C\_IsActiveFlag\_TXE(I2C\_TypeDef \*I2Cx)

Индицирует факт опустошения регистра данных на передачу: RESET возвращает при записи очередных данных в регистр передачи. SET возвращает для пустого буфера (NB: прерывание «TX буфер пуст»).

uint32\_t LL\_I2C\_IsActiveFlag\_TXIS(I2C\_TypeDef \*I2Cx)

Индицирует статус флага прерывания передачи TXIS.

uint32\_t LL\_I2C\_IsActiveFlag\_RXNE(I2C\_TypeDef \*I2Cx)

Индицирует факт непустого регистра данных на приём: RESET возвращает когда регистр данных прочитан, SET возвращает когда принятые данные скопированы в регистр приёма данных.

uint32\_t LL\_I2C\_IsActiveFlag\_NACK(I2C\_TypeDef \*I2Cx)

Индицирует факт неподтверждения приёма NACKF.

uint32\_t LL\_I2C\_IsActiveFlag\_STOP(I2C\_TypeDef \*I2Cx)

Индицирует факт обнаружения стоп-условия STOPF.

uint32\_t LL\_I2C\_IsActiveFlag\_TC(I2C\_TypeDef \*I2Cx)

Индицирует факт завершения передачи TC. Возвращает SET, когда RELOAD=0, AUTOEND=0 и NBYTES данных были переданы.

uint32\_t LL\_I2C\_IsActiveFlag\_TCR(I2C\_TypeDef \*I2Cx)

Индицирует факт завершения передачи TCR. Возвращает SET, когда RELOAD=1 и NBYTES данных были переданы.

uint32\_t LL\_I2C\_IsActiveFlag\_BERR(I2C\_TypeDef \*I2Cx)

Индицирует факт наличия ошибки на шине BERR. Возвращает SET когда обануржены старт- и стоп- условия в недопустимых местах.

uint32\_t LL\_I2C\_IsActiveFlag\_BUSY(I2C\_TypeDef \*I2Cx)

Индицирует факт занятости шины BUSY. Возвращает SET, когда старт-условие обнаружено.

**Функции сброса флагов прерывания**:

void LL\_I2C\_ClearFlag\_NACK(I2C\_TypeDef \*I2Cx)

Сбрасывает флаг неподтверждения NACKCF.

void LL\_I2C\_ClearFlag\_STOP(I2C\_TypeDef \*I2Cx)

Сбрасывает флаг обнаружения стоп- условия STOPCF.

void LL\_I2C\_ClearFlag\_TXE(I2C\_TypeDef \*I2Cx)

Сбрасывает флаг признака пустого регистра данных на передачу TXE. Этот бит может быть очищен программой для того, чтобы выгрузить (flush) содержимое регистра данных на передачу.

void LL\_I2C\_ClearFlag\_BERR(I2C\_TypeDef \*I2Cx)

Очистить флаг признака прерывания ошибки на шине.

**Функции управления передачей данных**:

void LL\_I2C\_EnableAutoEndMode(I2C\_TypeDef \*I2Cx)

Разрешение автоматической генерации стоп-условия в режиме мастера. Стоп-условие будет автоматически выставлено, когда будет передано NBYTES данных.

void LL\_I2C\_DisableAutoEndMode(I2C\_TypeDef \*I2Cx)

Запрещение автоматической генерации стоп-условия.

uint32\_t LL\_I2C\_IsEnabledAutoEndMode(I2C\_TypeDef \*I2Cx)

void LL\_I2C\_EnableReloadMode(I2C\_TypeDef \*I2Cx)

Разрешает режим перезагрузки. Передача не завершается после передачи NBYTES данных, NBYTES будет перезагружен когда TCR флаг будет установлен.

void LL\_I2C\_DisableReloadMode(I2C\_TypeDef \*I2Cx)

Запрещает режим перезагрузки. Передача завершается после передачи NBYTES данных (последуют стоп- или повтороное старт- условия).

uint32\_t LL\_I2C\_IsEnabledReloadMode(I2C\_TypeDef \*I2Cx)

void LL\_I2C\_SetTransferSize(I2C\_TypeDef \*I2Cx, uint32\_t TransferSize)

Установить количество байт на передачу. Изменение этих битов после старт-условия не допускается.

TransferSize может принимать значения от Min\_Data=0x00 до Max\_Data=0xFF.

uint32\_t LL\_I2C\_GetTransferSize(I2C\_TypeDef \*I2Cx)

void LL\_I2C\_GenerateStartCondition(I2C\_TypeDef \*I2Cx)

Генерирует старт- или потвороное старт- условие. Старт бит может быть установлен, даже если шина занята, или I2C работает в режиме ведомого. Это действие не имеет эффекта, если установлен RELOAD.

void LL\_I2C\_GenerateStopCondition(I2C\_TypeDef \*I2Cx)

Генерирует стоп-условие после передачи текущего байта.

void LL\_I2C\_SetTransferRequest(I2C\_TypeDef \*I2Cx, uint32\_t TransferRequest)

Установить направление передачи. Изменение этих битов после установки бита START не допускается.

TransferRequest может принимать значения:

LL\_I2C\_REQUEST\_WRITE, LL\_I2C\_REQUEST\_READ.

uint32\_t LL\_I2C\_GetTransferRequest(I2C\_TypeDef \*I2Cx)

void LL\_I2C\_SetSlaveAddr(I2C\_TypeDef \*I2Cx, uint32\_t SlaveAddr)

Установить адрес ведомого для передачи. Изменение этих битов после установки START бита не допускается.

SlaveAddr может принимать значения от Min\_Data = 0x00 до Max\_Data=0x3F.

uint32\_t LL\_I2C\_GetSlaveAddr(I2C\_TypeDef \*I2Cx)

**30.4.2 Требования тактирования I2C.**

I2C ядро тактируется частотой I2CCLK. Период этой частоты должен соответствовать следующим условиям:

и .

– это, соответственно, длительность периода высокого уровня и низкого.

– когда включен фильтр, это время означает сумму задержек, вносимых аналоговым фильтром и цифровым фильтром. Максимальная задержка аналогового фильтра составляет 260нс. Задержка цифрового фильтра составляет .

Тактовый период PCLK должен удовлетворять слудующим условиям:

, где – это период SCL.

**30.4.4 Инициализация I2C.**

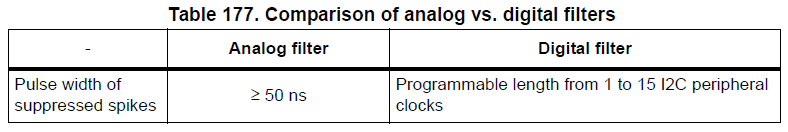
Разрешение и запрещение периферии.

Тактирование блока I2C должно быть настроено и разрешено в контроллере тактовой частоты. Затем I2C может быть включен установкой бита PE в регистре I2C\_CR1. Когда I2C запрещен (PE = 0), I2C выполняет программный сброс.

Шумовые фильтры.

Перед разрешением I2C периферии установкой бита PE в регистре I2C\_CR1, пользователь должен скнофигурировать шумовые фильтры, если необходимо. По-умолчанию, аналоговый шумовой фильтр присутствует на SDA и SCL входах. Этот аналоговый фильтр совместим с I2C-спецификацией, которая требует подавление выбросов с длительностью до 50нс в Fast-mode и Fast-mode Plus. Пользователь может отключить этот аналоговый фильтр, путём установки ANFOFF бита, и/или выбрать цифровой фильтр установкой битов DNF[3:0] в регистре I2C\_CR1.

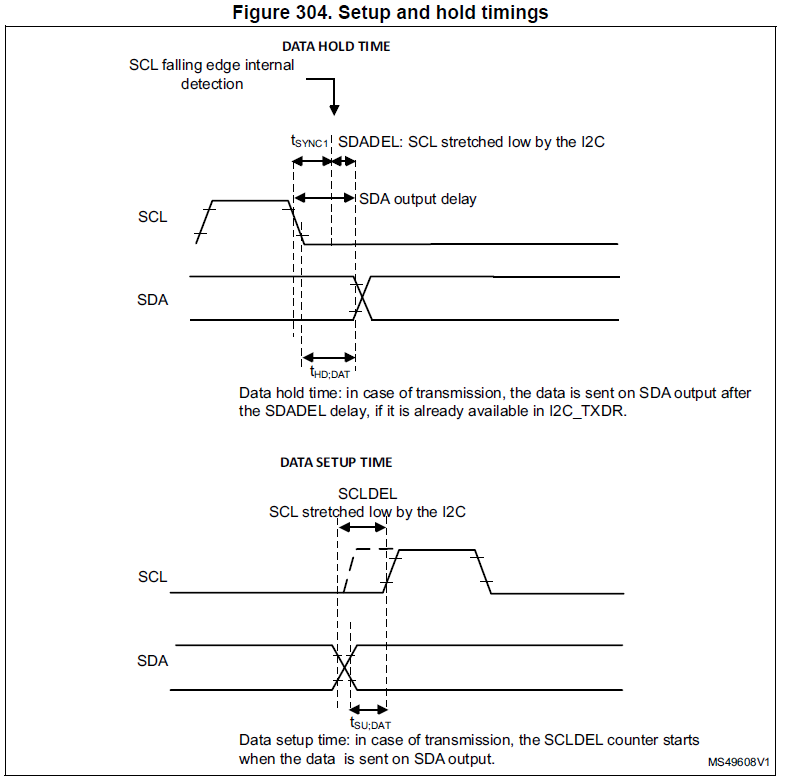
Когда цифровой фильтр включен, уровень SCL или SDA линии внутренне изменяется только если он остается стабильным в течение более DNF\*I2CCLK периодов. Это позволяет подавлять выбросы с программируемым временем от 1 до 15 периодов I2CCLK.



Примечание: изменение конфигурации фильтров не допускается, если I2C включен.

Тайминги I2C.

Тайминги должны быть настроены так, чтобы гарантировать корректные времена установки и удержания данных, используемых в режимах мастера и ведомого. Это достигается путём программирования битов PRESC[3:0], SCLDEL[3:0] и SDADEL[3:0] в регистре I2CTIMINGR.



На верхней диаграмме приведено определение времени удержания данных. Вертикальной стрелочкой показан момент обнаружения заднего фронта SCL. С этого момента отсчитывается задержка SDADEL. В случае передачи, данные посылаются на SDA вывод спустя время задержки SDADEL, если они уже доступны в I2C\_TXDR. Время удержания, таким образом, складывается из времени и линия SDA остается стабильной это время (NB: по-видимому, это сделано для того, чтобы затянуть момент изменения данных в линии SDA по заднему фронту SCL).

На нижней диаграмме приведено определение времени установления данных. Здесь видно, что SCL затянут искусственно в низком уровне, на время, соответствующее установке SCLDEL (NB: по-видимому, это сделано для того, чтобы данные надежно были защекнуты передним фронтом SCL).

Когда задний фронт SCL обнаружен внутренне, вставляется задержка перед отправкой в SDA выход. Эта задержка составляет время:

, где .

влияет на время удержания данных.

Суммарная задержка SDA составит: . Время зависит от следующих параметров:

\* наклона фронта SCL

\* входной задержки, вносимой аналоговым фильтром: нс.

\* входной задержки, вносимой цифровым фильтром: .

\* задержки, возникаемой из-за SCL синхронизации с тактовой частотой I2CCLK (2 или 3 I2CCLK периода).

Для того, чтобы «обойти» неопредленное место – задний фронт SCL, пользователь должен запрограммировать SDADEL так, чтобы:

– это часть уравнения, только когда аналоговый фильтр разрешен. Обратитесь к datasheet на ваше устройство за конкретными значениями.

Максимальное время может быть 3.45мкс, 0.9мкс и 0.45мкс для стандартного режима, Fast-режима и Fast Plus режима соответственно, но оно должно быть меньше, чем на время перехода. Максимальные допуски должны в’ыдерживаться только если устройство не затягивает низкий уровень SCL. Если имеет место быть затяжка, данные должны быть валидны в течении времени установления, перед тем как будет отпущен тактовый сигнал.

Передний фронт SDA, как правило, соответствует худшему случаю, так что в этом случае предыдущее уравнение становится:

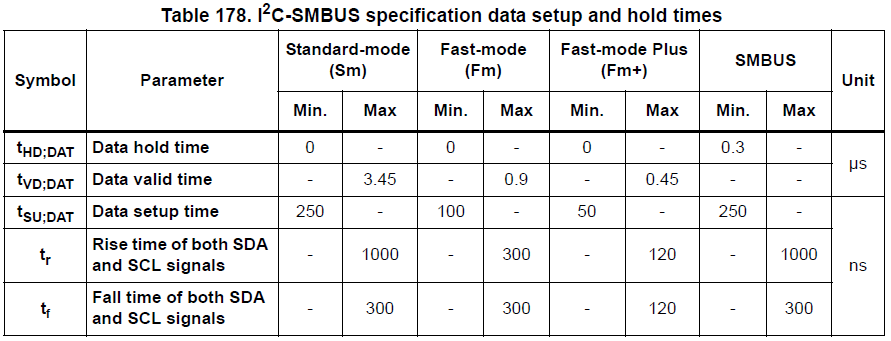
Это условие может быть нарушено, когда NOSTRETCH = 0, поскольку устройство затягивает SCL в низком уровне, чтобы гарантировать время установления данных, в соответствии с SCLDEL.

Спустя время (NB: верхний рисунок), или после отправки данных в SDA в случае, когда ведомому необходимо затянуть SCL (NB: нижний рисунок), поскольку данные не были еще записаны в I2C\_TXDR, SCL-линия остается в низком уровне в течение времени установления (NB: необходимое для установления стабильных данных на SDA). Это время установления составляет , где . влияет на время установления данных .

Для того, чтобы «обойти» неизвестно место – SDA переход (нарастающий фронт обычно худший случай (NB: логично, так как заряжается через подтяжку)), пользователь должен запрограммировать SCLDEL так, чтобы выполнялось неравенство:

*.*

Обратитесь к таблице 178 за стандартными временами установления и удержания, в соответствии со спецификацией I2C-SMBUS.



Времена переходов SDA и SCL являются параметрами приложения. Использование максимальных значений из стандарта ужесточают ограничения для SDADEL и SCLDEL, но обеспечивают работоспособность независимо от приложения.

Примечание: каждый тактовый импульс, после обнаружения заднего фронта SCL, I2C мастер или ведомый затягивают SCL в низком уровне по крайней мере на время

в обоих режимах – передачи и приёма. В режиме передачи, в случае, если данные ещё не были записаны в I2C\_TXDR на момент завершения счетчика SDADEL, I2C блок оставляет линию SCL в затяжке, до тех пор, пока следующие данные не будут записаны. Затем новые данные старшим битом вперёд продвигаются на SDA выход, и SCLDEL счетчик начинает считать, отсчитывая перид затяжки тактового импульса для гарантированного выдерживания времени установления.

Если в режиме ведомого выставлен бит NOSTRETCH=1, SCL не затягивается. Следовательно, SDADEL должен быть запрограммирован так, чтобы гарантировать также достаточное время установления.

Дополнительно, в режиме мастера, должны быть сконфигурированы длительности высокого и низкого уровня SCL, посредством установки полей PRESC[3:0], SCLH[7:0] и SCLL[7:0] битов в регистре I2C\_TIMINGR.

Когда задний фронт SCL внутренне обнаруживается, вставляется задержка перед освобождением SCL выхода. Эта задержка составляет:

, где . влияет на длительность низкого уровня SCL .

Когда нарастающий фронт SCL внутренне обнаружен, задержка вставляется перед тем, как притянуть SCL выход к низкому уровню. Эта задержка составляет , где . влияет на длительность высокого уровня .