**21.1 Введение**

Периферийное устройство I2C с несколькими ведущими устройствами обеспечивает интерфейс между ЦП и любым устройством, совместимым с шиной I2C, которое подключается через последовательную шину I2C. Внешние компоненты, подключенные к шине I2C, могут последовательно передавать/принимать до 8-битных данных в/из процессорного устройства через двухпроводной интерфейс I2C.

Шина I2C является шиной с несколькими мастерами(multimaster). Контроллер I2C поддерживает режим multi-master, который позволяет подключить к нему более одного устройства, способного управлять шиной. Каждое устройство I2C отличается уникальным адресом и может работать как передатчик или приемник, в зависимости от функции устройства. Устройство, подключенное к шине I2C, может быть не только передатчиком или приемником, но и рассматривается как ведущее или ведомое при выполнении передачи данных. Обратите внимание, что ведущим устройством является устройство которое инициирует передачу данных по шине и генерирует тактовые сигналы, разрешающие эту передачу. При такой передаче любое устройство, к которому обращается этот мастер, считается ведомым.

**21.1.1 Функции I2C**

Общие характеристики контроллера I2C:

- Соответствует спецификации Philips I2C версии 2.1.

- Поддерживает стандартный режим (до 100 000 бит/с) и быстрый режим (до 400 000 бит/с).

- Режим мультиведущего передатчика/ведомого приемника

- Режим мультиведущего приемника/ведомого передатчика

- Комбинированные основные режимы передачи/приема и приема/передачи

- 7-битный и 10-битный режимы адресации устройств

- Встроенный 32-байтовый FIFO для буферизованного чтения или записи в каждом модуле

- Генерация программируемого тактового сигнала

- Два канала прямого доступа к памяти, одна линия прерывания

**21.1.2 Неподдерживаемые функции I2C**

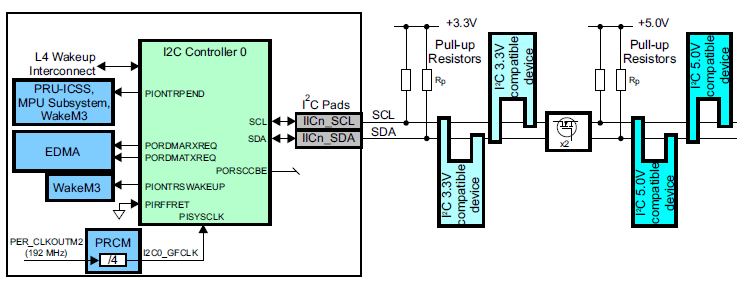
Функции модуля I2C, не поддерживаемые этим устройством, показаны в Таблице 21-1.

**Таблица 21-1. Неподдерживаемые функции I2C**

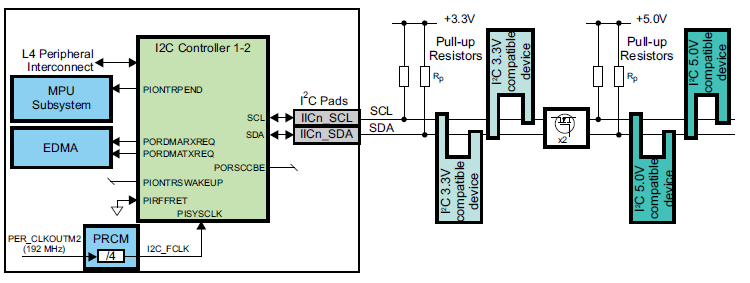
|  |  |
| --- | --- |
| Feature | Reason |
| SCCB Protocol | SCCB signal not pinned out |
| High Speed (3.4 MBPS) operation | Not supported |

**21.2 Интеграция**

Это устройство включает в себя три экземпляра модуля I2C. Это периферийное устройство реализует multimaster шину I2C, обеспечивающуб последовательную передачу 8-битных данных на/от других ведущих/ведомых устройств I2C по двухпроводной линии. Существует три экземпляра модуля I2C, которые называются I2C0, I2C1 и I2C2. Модуль I2C0 находится в домене Wake-up питания. На рис. 21-1 и рис. 21-2 показаны примеры системы с несколькими I2C-совместимыми устройствами.



**Рисунок 21-1. Интеграция модуля I2C0 и его применение.**



**Рисунок 21-2. Интеграция модулей I2C(1–2) и их применение.**

**21.2.1 Атрибуты подключения I2C**

Общие атрибуты подключения для модуля I2C показаны в Таблице 21-2 и Таблице 21-3.

**Таблица 21-2. Атрибуты подключения I2C0**

|  |  |
| --- | --- |
| Attributes | Type |
| Power Domain | Wakeup Domain |
| Clock Domain | PD\_WKUP\_L4\_WKUP\_GCLK (Interface/OCP)  PD\_WKUP\_I2C0\_GFCLK (Func) |
| Reset Signals | WKUP\_DOM\_RST\_N |
| Idle/Wakeup Signals | Smart Idle / Wakeup |
| Interrupt Requests | 1 прерывание на подсистему MPU (I2C0INT), PRU-ICSS и  WakeM3 |
| DMA Requests | 2 запроса DMA к EDMA (I2CTXEVT0, I2CRXEVT0) |
| Physical Address | L4 Wakeup slave port |

**Таблица 21-3. Атрибуты подключения I2C(1–2)**

|  |  |
| --- | --- |
| Attributes | Type |
| Power Domain | Peripheral Domain |
| Clock Domain | PD\_PER\_L4LS\_GCLK (Interface/OCP)  PD\_PER\_I2C\_FCLK (Func) |
| Reset Signals | PER\_DOM\_RST\_N |
| Idle/Wakeup Signals | Smart Idle |
| Interrupt Requests | 1 прерывание на экземпляр для подсистемы MPU (I2C1INT, I2C2INT) |
| DMA Requests | 2 запроса DMA на экземпляр к EDMA (I2CTXEVTx,  I2CRXEVTx) |
| Physical Address | L4 Peripheral slave port |

**21.2.2 Тактирование I2C и управление сбросом**

Контроллеры I2C имеют отдельный интерфейс шины и функциональное тактирование. В режиме пониженного энергопотребления I2Cx\_SCL и I2Cx\_SDA настроены как входы.

**Таблица 21-4. Тактовые сигналы I2C**

|  |  |  |  |
| --- | --- | --- | --- |
| Clock Signal | Max Freq | Reference / Source | Comments |
| I2C0 Clock Signals | | | |
| PIOCPCLK  Interface clock | MHz |  | pd\_wkup\_l4\_wkup\_gclk  From PRCM |
| PISYSCLK  Functional clock | 48 MHz | PER\_CLKOUTM2 / 4 | pd\_wkup\_i2c0\_gfclk  From PRCM |
| I2C(1-2) Clock Signals | | | |
| PIOCPCLK  Interface clock | 100 MHz | CORE\_CLKOUTM4 / 2 | pd\_per\_l4ls\_gclk  From PRCM |
| PISYSCLK  Functional clock | 48 MHz | PER\_CLKOUTM2 / 4 | pd\_per\_ic2\_fclk  From PRCM |

**21.2.3 Список выводов I2C**

Внешние сигналы (I2Cx\_SDA, I2Cx\_SCL) на устройстве используют стандартные входы/выходы LVCMOS и могут не полностью соответствовать спецификациям I2C для устройств Fast-mode для управления наклоном и входной фильтрацией (подавление всплесокв) для улучшения характеристик электромагнитной совместимости.

**Таблица 21-5. Список выводов I2C**

|  |  |  |
| --- | --- | --- |
| Pin | Type | Description |
| I2C*x*\_SCL | I/OD(1) | I2C serial clock (open drain) |
| I2C*x*\_SDA | I/OD | I2C serial data (open drain) |

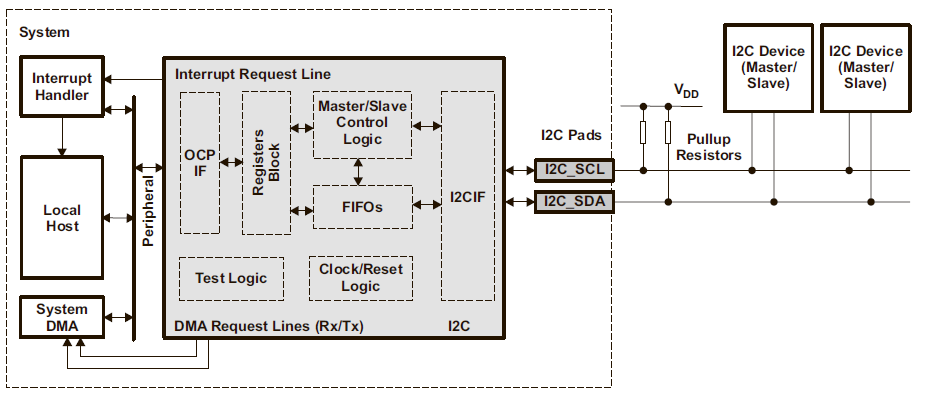
(1) Эти сигналы также используются в качестве входных данных для пересчета времени или синхронизации данных. Соответствующий бит CONF\_<module>\_<pin>\_RXACTIVE для этих сигналы должны быть установлены на 1, чтобы включить входы обратно в модуль. Также рекомендуется последовательно поставить резистор на 33 Ом. (близко к процессору) для каждого из этих сигналов, чтобы избежать отражения сигнала.

**21.3 Функциональное описание**

**21.3.1 Функциональная блок-схема**

На рис. 21-3 показан пример системы с несколькими совместимыми с I2C устройствами, в которых все порты последовательного интерфейса I2C соединены вместе для двусторонней передачи с одного устройства на другое.

**Рисунок 21-3. Функциональная блок-схема I2C**



Периферийное устройство I2C состоит из следующих основных блоков:

- Последовательный интерфейс: один вывод данных (I2C\_SDA) и один вывод синхронизации (I2C\_SCL).

- Регистры данных для временного хранения данных приема и передачи данных, перемещающихся между контактами I2C\_SDA и ЦП или контроллер прямого доступа к памяти.

- Регистры управления и состояния

- Интерфейс шины данных периферийных устройств, позволяющий ЦП и контроллеру прямого доступа к памяти получать доступ к периферийным регистрам устройств I2C.

- Синхронизатор тактирования для синхронизации входного тактирования I2C (от тактового генератора процессора) и тактовых импульсов на выводе I2C\_SCL, а также для синхронизации передачи данных с ведущими устройствами с разными тактовыми частотами.

- Предварительный делитель для разделения входного тактового сигнала, передаваемого на периферийное устройство I2C.

- Шумовой фильтр на каждом из двух контактов, I2C\_SDA и I2C\_SCL.

- Арбитр для управления арбитражем между периферийным устройством I2C (когда оно является ведущим) и другим ведущим устройством.

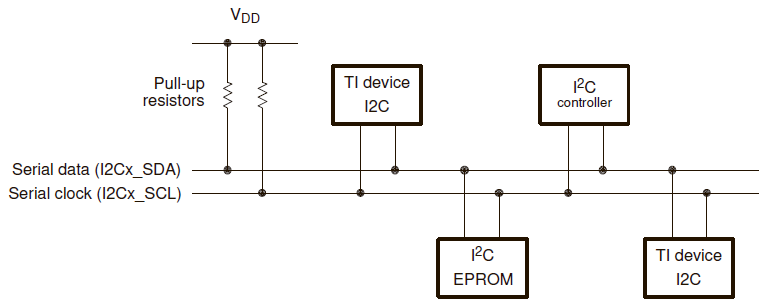
- Логика генерации прерывания, чтобы прерывание могло быть отправлено в ЦП

- Логика генерации событий DMA для отправки прерывания в ЦП при приеме и передаче данных.

**21.3.2 Сигналы главного/ведомого контроллера I2C**

Данные передаются на устройства, взаимодействующие с I2C, через последовательную линию передачи данных (SDA) и последовательные линию тактовых импульсов (SCL). Эти два провода могут передавать информацию между устройством и другими устройствами, подключенными к шине I2C. И SDA, и SCL являются двунаправленными контактами. Они должны быть подключены к положительному напряжению питания через подтягивающий резистор. Когда шина свободна, оба контакта имеют высокий уровень. Драйвер этих двух контактов имеет открытый сток чтобы иметь возможность выполнить функцию проводного И(wired-AND).

Пример нескольких модулей I2C, подключенных для двусторонней передачи с одного устройства на другое показан на рис. 21-4.

**Рисунок 21-4. Подключение нескольких модулей I2С**

**Таблица 21-6. Сигнальные выводы.**

|  |  |  |
| --- | --- | --- |
| Name | I2C Mode | |
| **Default Operating**  **Mode** | **Description** |
| I2C\_SCL | In/ Out | I2C последовательная линия CLK  Выходной буфер с открытым стоком. Требуется внешний подтягивающий резистор (Rp). |
| I2C\_SDA | In/ Out | Линия последовательной передачи данных I2C  Выходной буфер с открытым стоком. Требуется внешний подтягивающий резистор (Rp). |

**21.3.3 Сброс I2C**

Модуль I2C можно сбросить тремя способами:

- Сброс системы (PIRSTNA = 0). Сброс устройства вызывает сброс системы. Все регистры сбрасываются на значения сброса при включении питания.

- Программный сброс путем установки бита SRST в регистре I2C\_SYSC. Этот бит имеет точно такое же действие на логику модуля в виде сброса системной шины. Все регистры сбрасываются до значений сброса при включении питания.

- Бит I2C\_EN в регистре I2C\_CON может использоваться для удержания модуля I2C в состоянии сброса. Когда удален сброс системной шины (PIRSTNA = 1), I2C\_EN = 0 сохраняет функциональную часть модуля I2C в состояние сброса, и все регистры конфигурации могут быть доступны. I2C\_EN=0 не сбрасывает регистры в значения сброса при включении питания.

**Таблица 21-7. Сигналы сброса состояния I2C**

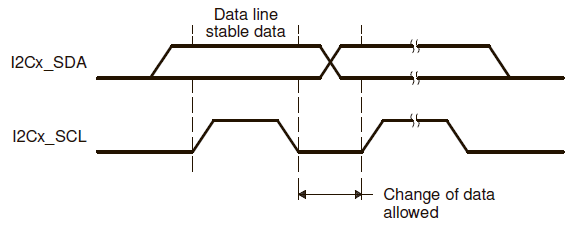
|  |  |  |  |
| --- | --- | --- | --- |
| Pin | I/O/Z (1) | System Reset | I2C Reset |
| **(I2C\_EN = 0)** |
| SDA | I/O/Z | High impedance | High impedance |
| SCL | I/O/Z | High impedance | High impedance |

(1) I = вход, O = выход, Z = высокое сопротивление(Третье состояние)

21.3.4 Достоверность данных

Данные на линии SDA должны быть стабильными в течение высокого периода тактового сигнала. Высокое и низкое состояние линии данных может измениться только тогда, когда тактовый сигнал на линии SCL имеет НИЗКИЙ уровень.

**Рисунок 21-5. Передача битов по шине I2C**



**21.3.5 Условия START и STOP**

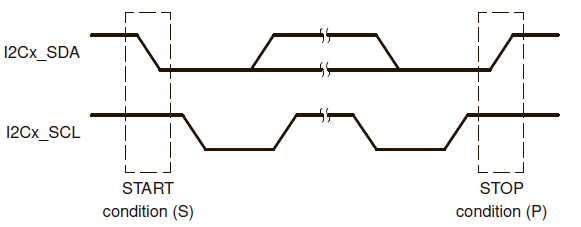
Модуль I2C генерирует условия СТАРТ и СТОП, когда он настроен как ведущий.

- Условие START – это переход с высокого на низкий уровень на линии SDA, когда SCL имеет высокий уровень.

- Состояние STOP – это переход с низкого уровня на высокий на линии SDA при высоком уровне SCL.

- Шина считается занятой после состояния START (BB = 1) и свободной после состояния STOP условие (ВВ = 0).

**Рисунок 21-6. События START и STOP на шине.**

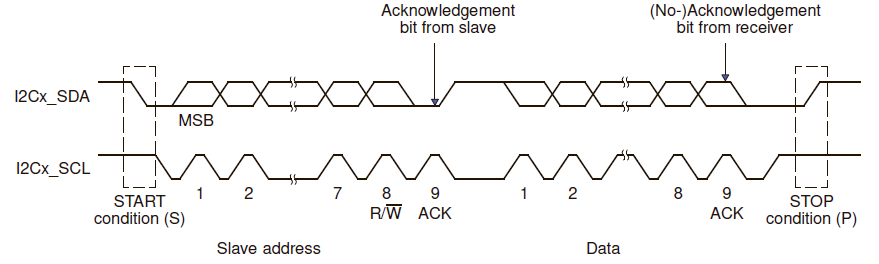


**21.3.6 Работа I2C**

**21.3.6.1 Форматы последовательных данных**

Контроллер I2C работает в формате 8-битных словных данных (доступ для записи байтов поддерживается для последнего доступа). Каждый байт, помещаемый в линию SDA, имеет длину 8 бит. Количество байтов, которое может быть передано или получено, ограничено значением, запрограммированным в регистре DCOUNT. Данные передаются с в порядке (MSB) максимальный значащий бит передается первым. За каждым байтом следует бит подтверждения от модуля I2C, если он находится в режиме приемника.

**Рисунок 21-7. Передача данных I2C**



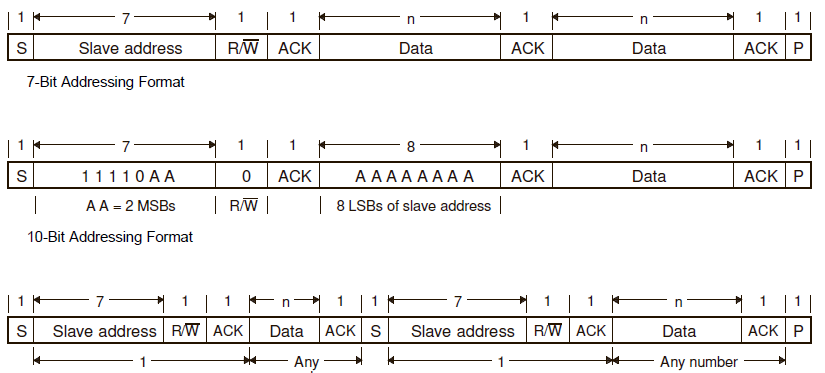
Модуль I2C поддерживает два формата данных, как показано на рис. 21-8:

- 7-битный/10-битный формат адресации.

- 7-битный/10-битный формат адресации с повторяющимся начальным условием.

Первый байт после начального условия (S) всегда состоит из 8 бит. В режиме подтверждения дополнительный бит предназначенный для подтверждения, вставляется после каждого байта. В форматах адресации с 7-битными адресами первый байт состоит из 7 старших разрядов адреса подчиненного устройства и 1 младшего бита R/nW. В форматах адресации с 10-битными адресами первый байт состоит из 7 старших бит адреса подчиненного устройства, например 11110XX, где XX — это два старших бита 10-битных адресов и 1 младший бит R/nW, который в данном случае равен 0.

Младший значащий R/nW адресного байта указывает направление передачи следующих данных байт. Если R/nW равно 0, ведущий записывает данные в выбранное ведомое устройств; если он равен 1, мастер считывает данные из ведомого устройства.

**Рисунок 21-8. Форматы передачи данных I2C** 

7-битный формат адресации с повторяющимся условием START

**21.3.6.2 Мастер передатчик**

В этом режиме данные, собранные в одном из ранее описанных форматов данных, выводятся на последовательный порт линии данных SDA и синхронизируются с самогенерируемыми тактовыми импульсами на последовательной линии синхронизации SCL. Тактовые импульсы запрещаются, а SCL удерживается на низком уровне, когда требуется вмешательство процессора (XUDF) после того, как байт был передан.

**21.3.6.3 Главный приемник**

В этот режим можно войти только из режима мастер передатчика. С любым из форматов адреса (Рисунок 21-8 (a), (b) и (c)), главный приемник вводится после байта адреса подчиненного устройства и передачи бита R/W\_, если R/W\_ имеет высокий уровень. Биты последовательных данных, принимаемые по шине SDA, сдвигаются синхронно с самогенерируемые тактовые импульсы на SCL. Тактовые импульсы подавляются, а SCL удерживается на низком уровне, когда требуется вмешательство процессора (ROVR) после передачи байта. В конце передачи, он генерирует условие остановки.

**21.3.6.4 Ведомый передатчик**

В этот режим можно войти только из режима ведомого приемника. С любым из форматов адреса (Рисунок 21-8 (a), (b) и (c)) ведомый передатчик вводится, если байт адреса ведомого совпадает с его собственным адресом передачи и бит R/W\_, если R/W\_ имеет высокий. Ведомый передатчик сдвигает последовательные данные на линии данных SDA синхронно с тактовыми импульсами, которые генерируются ведущим устройством. Он не генерирует свои тактовые импульсы, но может удерживать SCL линии тактирования на низком уровне, пока требуется вмешательство ЦП (XUDF).

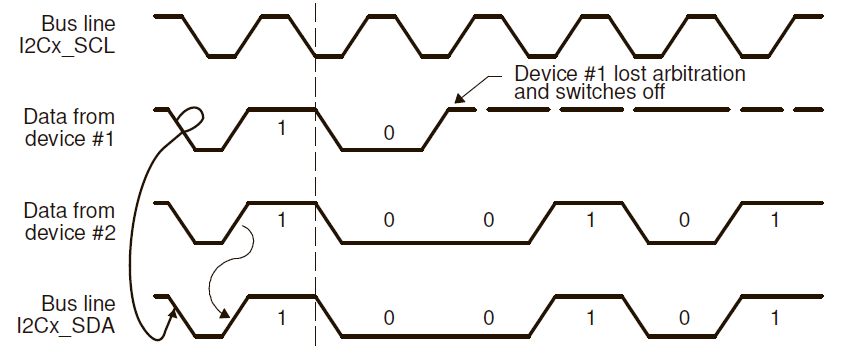
**21.3.6.5 Ведомый приемник**

В этом режиме биты последовательных данных, принимаемые по шине SDA, вводятся синхронно с тактовыми импульсами на SCL, генерируемыми ведущим устройством. Он не генерирует тактовые импульсы, но может держать линию синхронизации SCL в низком уровне, когда требуется вмешательство ЦП (ROVR) после приема байта.

**21.3.7 Арбитраж**

Если два или более ведущих передатчика начинают передачу по одной и той же шине почти одновременно, применяется арбитражная процедура. Процедура арбитража использует данные, представленные на последовательной шине конкурирующими передатчиками. Когда передатчик обнаруживает, что высокий уровень сигнала, который он представил на шине, был отменен низким сигналом, он переключается в режим ведомого приемника, устанавливает флаг потери арбитража (AL), и генерирует прерывание потери арбитража. На рис. 21-9 показана арбитражная процедура между двумя устройствами. Процедура арбитража отдает приоритет устройству, передающему последовательный поток данных с наименьшим двоичным значением. Если два или более устройств передают одинаковые первые байты, арбитраж продолжается последующими байтами.

**Рисунок 21-9. Арбитражная процедура между двумя ведущими передатчиками**



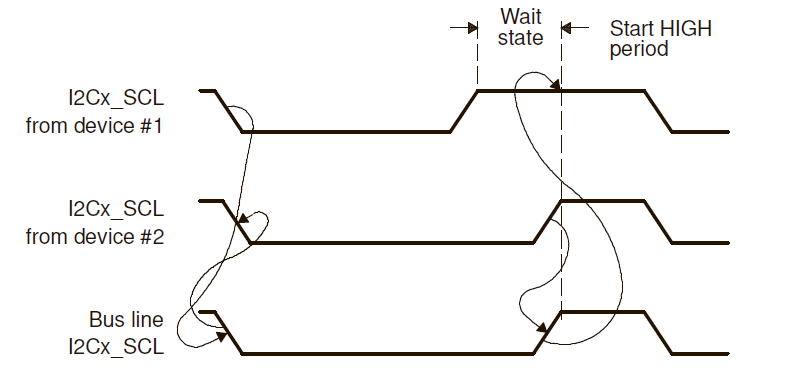
21.3.8 Генерация тактирования I2C и синхронизация тактирования I2C

В нормальных условиях только одно ведущее устройство генерирует тактовый сигнал SCL. Во время процедуры арбитража, однако есть два или более ведущих устройства, и тактовые импульсы должны быть синхронизированы, чтобы вывод данных можно было сравнить. Свойство проводного И линии синхронизации означает, что устройство, которое первым генерирует низкий период тактовой линии, подавляет другие устройства. При этом переходе с высокого на низкий тактовые импульсы генератора других устройств вынуждены начинать генерацию своего низкого периода. Линия тактирования затем удерживается на низком уровне устройством с самым длинным периодом низкого уровня, в то время как другие устройства, которые заканчивают свои периоды низкого уровня должны дождаться освобождения тактовой линии, прежде чем начинать свои периоды высокого уровня. Таким образом получается тактовая линия с синхронизированным сигналом, где самое медленное устройство определяет продолжительность периода низкого уровня ,а самое быстрое – длину периода высокого уровня.

Если устройство отключает тактовую линию в течение более длительного времени, результатом является то, что все тактовые генераторы должны войти в состояние ОЖИДАНИЯ. Таким образом, ведомое устройство может замедлить быстрое ведущее устройство, а медленное устройство может создать достаточно времени для сохранения принятого байта или для подготовки байта к передаче (растягивание тактовой частоты). Рисунок 21-10 иллюстрирует синхронизацию тактовых импульсов.

***Примечание***. Если линии SCL или SDA застряли на низком уровне, поддерживается операция очистки шины. Если тактовая линия (SCL) застряла на низком уровне, предпочтительной процедурой является сброс шины с помощью сигнала аппаратного сброса, если ваше I2C устройство имеет входы аппаратного сброса. Если устройства I2C не имеют входов аппаратного сброса, выключите и снова включите питание устройства для активации принудительной внутренней схемы сброса при включении питания (POR). Если линия передачи данных (SDA) застряла на низком, мастер должен отправить девять тактовых импульсов. Устройство, удерживающее шину на низком уровне, должно освободить ее где-то в пределах этих девяти импульсов. Если нет, используйте аппаратный сброс или выключите питание, чтобы очистить шину.

**Рисунок 21-10. Синхронизация двух тактовых генераторов I2C**

****

**21.3.9 Предделитель (SCLK/ICLK)**

Модуль I2C работает с частотой функционального тактирования(SCLK), которая может находиться в диапазоне 12-100 МГц, в соответствии с режимом I2C, который необходимо использовать (в случае режима работы F/S e). Обратите внимание, что частота функционального тактирования напрямую влияет на производительность шины и тайминги у I2C.

Внутреннее тактирование, используемое для логики I2C — ICLK — генерируются через блок предварительного делителя I2C. Предделитель состоит из 4-битного регистра — I2C \_PSC и используется для деления системной частоты (SCLK) для получения внутренних необходимых тактовых импульсов для модуля I2C.

**21.3.10 Шумовой фильтр**

Шумовой фильтр используется для подавления любого шума длительностью 50 нс или меньше в случае режима работы F/S. Он предназначен для подавления шума с помощью одного ICLK. Шумовой фильтр всегда составляет один цикл ICLK, независимо от скорости шины. Для режима FS (предделитель = 4, ICLK = 24 МГц) максимальная ширина подавляемых всплесков равна 41.6 нс. Чтобы обеспечить правильную фильтрацию, предварительный делитель должен быть запрограммирован соответствующим образом.

**21.3.11 Прерывания I2C**

Модуль I2C генерирует 12 типов прерываний: адресованное как ведомое, свободной шины (обнаружено состояние остановки),ошибки доступа, условия запуска, потери арбитража, отсутствия подтверждения(ACK), общего вызова, готовых к доступу регистров, приема и передачи данных, приема и передачи слива. Эти 12 прерываний сопровождаются 12-ю масками и флагами прерываний, определенными в регистрах **I2C\_IRQENABLE\_SET** и **I2C\_IRQSTATUS\_RAW** соответственно. Обратите внимание, что все эти 12 событий прерывания используют одну и ту же линию аппаратного прерывания.

- Прерывание **Addressed As Slave** (**AAS**) генерируется для информирования локального хоста о том, что внешний мастер обратился к модулю как к ведомому. Когда это прерывание происходит, CPU может проверить регистр **I2C\_ACTOA** состояния для проверки того, какой из 4 собственных адресов использовался внешним мастером для доступа к модуль.

- Прерывание **Bus Free (BF)** генерируется для информирования локального хоста о том, что шина I2C освободилась (когда cостояние остановки обнаружено на шине), и модуль может инициировать свою собственную транзакцию **I2C**.

- Прерывание по условию запуска **Start Condition** (**STC**) генерируется после того, как модуль, находящийся в режиме ожидания, обнаружил (синхронно или асинхронно) возможное состояние запуска на шине (сигнализируется с помощью **WakeUp**).

- Прерывание по ошибке доступа **Access Error** (AERR) генерируется, если доступ для чтения данных выполняется, когда **RX FIFO** пуст или доступ для записи данных выполняется, когда **TX FIFO** заполнен.

- Прерывание при потере арбитража **Arbitration lost** (AL) генерируется при потере процедуры арбитража I2C.

- Прерывание без подтверждения **No-acknowledge** (NACK) генерируется, когда ведущий I2C не получает подтвержденя от приемника.

- Прерывание общего вызова **General call** (GC) генерируется, когда устройство обнаруживает адрес из всех нулей (8 бит).

- Прерывание готовности к доступу к регистрам **Registers-ready-for-access** (ARDY) генерируется I2C, когда ранее запрограммированный адрес, данные и команда выполнены, а биты состояния обновлены. Это прерывание используется, чтобы сообщить ЦП, что регистры I2C готовы к доступу.

- Прерывание/состояние приема **Receive interrupt/status** (RRDY) генерируется, когда полученные данные готовы для чтения ЦП из регистра **I2C**\_**DATA** (см. подраздел «Управление FIFO» для полного описания необходимые условия для генерации прерывания). В качестве альтернативы ЦП может опросить этот бит, чтобы прочитать полученный данные из регистра **I2C\_DATA**.

- Прерывание/статуса передачи **Transmit interrupt/status** (XRDY) генерируется, когда ЦП необходимо поместить больше данных в **I2C\_DATA** регистр после того, как переданные данные были смещены на вывод SDA (см. раздел «Управление FIFO» для полного описания необходимых условий для генерации прерывания). ЦП может в качестве альтернативы опросить этот бит, чтобы записать следующие передаваемые данные в регистр **I2C\_DATA**.

- Прерывание по приему данных **Receive draining** (RDR) генерируется, когда длина передачи не кратна пороговому значению, чтобы сообщить ЦП, что он может прочитать количество данных, оставшихся для передачи, и разрешить механизм слива (дополнительную информацию см. в разделе 21.3.14.4 «Функция слива»).

- Прерывание слива передачи **Transmit draining** (XDR) генерируется, когда длина передачи не кратна пороговому значению, чтобы сообщить ЦП, что он может считать количество данных, оставшихся для записи, и включить механизм слива (дополнительную информацию см. в разделе 21.3.14.4 «Функция слива»).

Когда активируется сигнал прерывания, локальный хост должен прочитать регистр **I2C\_IRQSTATUS\_RAW**, чтобы определить тип прерывания, обработать запрос, а затем записать в эти регистры правильное значение для очистки флага прерывания.

**21.3.12 События прямого доступа к памяти**

Модуль I2C может генерировать два события запроса DMA: чтения (**I2C\_DMA\_RX**) и записи (**I2C\_DMA\_TX**), которые могут использоваться контроллером DMA для синхронного чтения полученных данных из **I2C\_DATA** или записи переданных данныч в регистр **I2C\_DATA**. Запросы чтения и записи DMA генерируются аналогичным образом как **RRDY** и **XRDY** соответственно. Сигналы запроса **DMA I2C (I2C\_DMA\_TX** и **I2C\_DMA\_RX**) активируются в соответствии с подразделом Управление FIFO.

**21.3.13 Прерывания и события DMA**

I2C имеет два канала прямого доступа к памяти (Tx и Rx).

I2C имеет одну линию прерывания для всех запросов на прерывание.

Номера событий и прерываний см. в техническом описании конкретного устройства.

**21.3.14 Управление ФИФО**

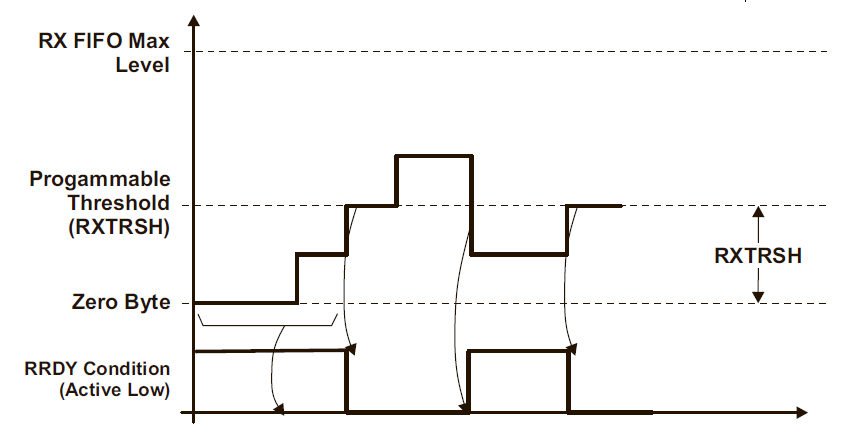
Модуль I2C реализует два внутренних 32-байтовых FIFO с двойной синхронизацией для режимов RX и TX. Глубинe FIFO можно настроить при интеграции с помощью общего параметра, который также будет отражен в регистре **I2C\_IRQSTATUS\_RAW.FIFODEPTH**.

**21.3.14.1 Работа в режиме прерывания FIFO**

В режиме прерывания FIFO (соответствующие прерывания разрешены через регистр **I2C\_IRQENABLE\_SET**) процессор информируется о состоянии приемника и передатчика сигналом прерывания. Эти прерывания вызываются при достижении порога **FIFO** приема/передачи (определяемого параметрами **I2C\_BUF.TXTRSH** или **I2C\_BUF.RXTRSH**); сигналы прерывания предписывают локальному хосту передать данные в пункт назначения (из модуля I2C в в режиме приема и/или из любого источника в **FIFO I2C** в режиме передачи).

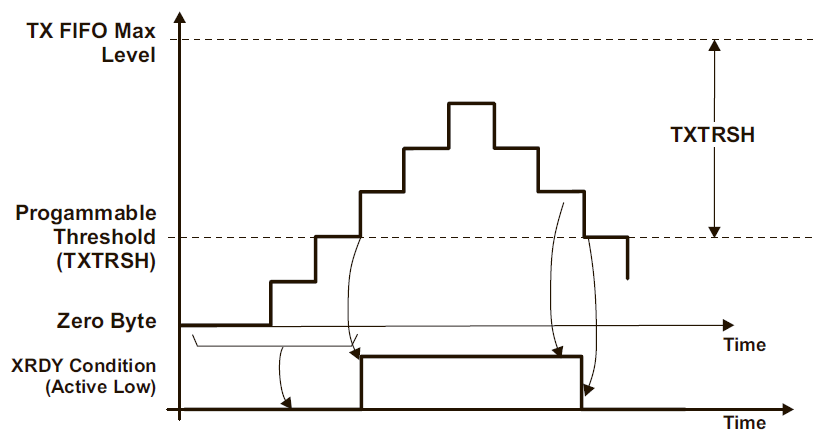
Рисунок 21-11 и Рисунок 21-12, соответственно, иллюстрируют операции приема и передачи из FIFO с точки зрения управления.

**Рисунок 21-11. Получение запроса на прерывание FIFO**



Обратите внимание, что на рис. 21-11 условие RRDY показывает, что достигается условие для создания RRDY прерывания. Запрос на прерывание генерируется, когда этот сигнал активен, и его можно сбросить только через ЦП, записав 1 в соответствующий флаг прерывания. Если состояние сохраняется после очистки предыдущего прерывания будет сгенерирован другой запрос прерывания. В режиме приема прерывания RRDY не генерируется до тех пор, пока FIFO не достигнет своего порога приема. Пока он низкий, прерывание может быть отменено только тогда, когда локальный хост обработает достаточно байтов, чтобы сделать FIFO уровень ниже порога. Для каждого прерывания локальный хост может быть настроен на чтение определенного количества байтов равного значению порога RX FIFO + 1.

**Рисунок 21-12. Генерация запроса на прерывание передачи FIFO**



Обратите внимание, что на рис. 21-12 условие XRDY показывает, что достигается условие для создания XRDY прерывания. Запрос на прерывание генерируется при достижении этого условия (когда TX FIFO опутошен, или пороговое значение TX FIFO не достигнуто, и в TX все еще есть байты данных для передачи FIFO), и он может быть очищен только процессором путем записи 1 в соответствующий флаг прерывания после передачи сконфигурированного количества байтов. Если условие все еще присутствует после сброса предыдущего прерывания, будет сгенерирован другой запрос на прерывание.

Обратите внимание, что в режиме прерывания модуль предлагает два варианта обработки прерываний приложением ЦП:

- При обнаружении запроса на прерывание (типа XRDY или RRDY) ЦП может записать/прочитать один байт данных в/из FIFO, а затем очистить прерывание. Модуль не будет подтверждать прерывание до тех пор, пока условие прерывания не выполнится.

- При обнаружении запроса прерывания (типа XRDY или RRDY) ЦП можно запрограммировать на запись/чтение количества байтов данных, указанное соответствующим порогом FIFO (I2C\_BUF.TXTRSH + 1 или I2C\_BUF.RXTRSH + 1). В этом случае условие прерывания будет очищено, и будет выполнено следующее прерывание, когда снова будет выполнено условие XRDY или RRDY.

Если используется второй подход к обслуживанию прерываний, используется дополнительный механизм (функция слива) реализован для случая, когда длина передачи не кратна порогу FIFO (см.Раздел 21.3.14.4, Функция слива).

В ведомом режиме TX функция слива не может быть использована, так как длина передачи неизвестна на время конфигурации, и внешний мастер может завершить передачу в любой момент, не подтвердив байт данных.

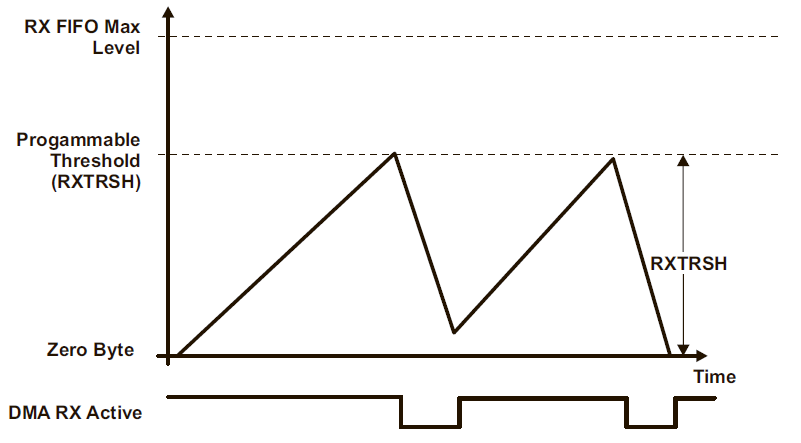
**21.3.14.2 Работа в режиме опроса FIFO**

В режиме опроса FIFO **(I2C\_IRQENABLE\_SET.XRDY\_IE** и **I2C\_IRQENABLE\_SET.RRDY\_IE** отключены и DMA отключен), состояние модуля (приемник или передатчик) можно проверить, опросив **XRDY** и регистры состояния **RRDY** (**I2C\_IRQSTATUS\_RAW**) (**RDR** и **XDR** также могут быть опрошены, если необходимо использовать функцию слива). Флаги XRDY и RRDY точно отражают условия прерывания, упомянутые в режимах прерывания. Этот режим является альтернативой режиму прерывания FIFO, в котором состояние приемника и передатчика автоматически узнаются с помощью прерываний, отправляемых в ЦП.

**21.3.14.3 Работа в режиме FIFO DMA**

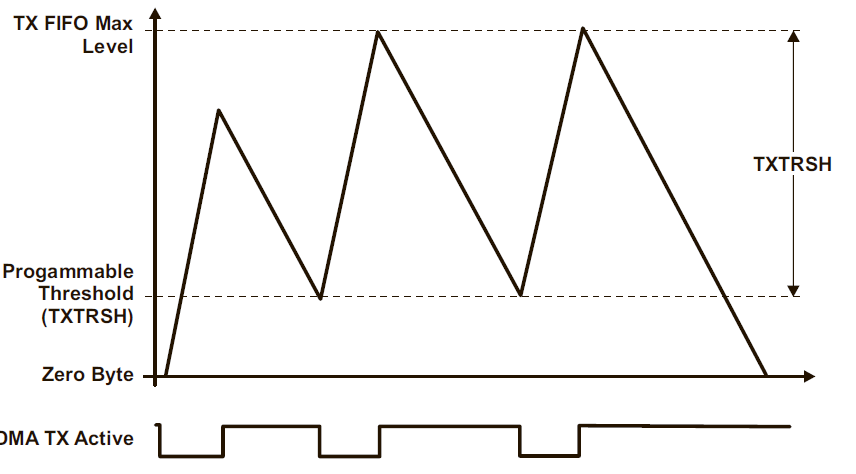
В режиме приема запрос DMA генерируется, как только приемный FIFO превышает пороговый уровень , определенный в регистре порогового уровня (**I2C\_BUF.RXTRSH +1**). Этот запрос должен быть отменен, когда количество байт, определенное пороговым уровнем, было прочитано DMA, путем установки поля **I2C\_DMARCXENABLE\_CLR.DMARX\_ENABLE\_CLEAR**.

**Рисунок 21-13. Прием генерации запроса FIFO DMA**

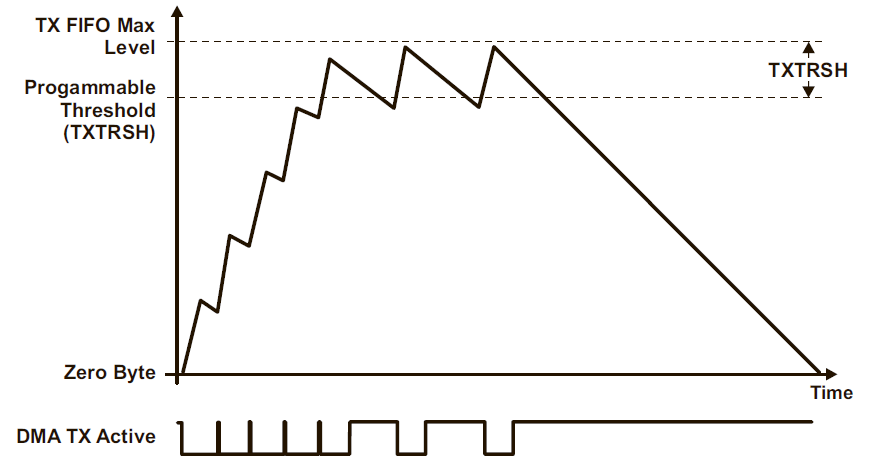


В режиме передачи запрос DMA автоматически утверждается, когда буфер передачи FIFO пуст. Этот запрос должно быть отменен, когда количество байтов, определяемое числом в пороговом регистре **(I2C\_BUF.TXTHRS+1)** был записан **DMA** в FIFO путем установки битового поля **DMATX\_ENABLE\_CLEAR** в **I2C\_DMATXENABLE\_CLR**. Если недостаточно символов записано, то запрос DMA останется активным. Рисунок 21-14 и Рисунок 21-15 иллюстрируют **DMA TX** передачу с разными значениями для **TXTRSH**.

**Рисунок 21-14. Генерация запроса DMA FIFO передачи (высокий порог)**



**Рисунок 21-15. Генерация запроса FIFO DMA на передачу (низкий порог)**



Обратите внимание, что также в режиме DMA возможна передача, длина которой не кратна сконфигурированному порогу FIFO. В этом случае функция слива DMA также используется для передачи дополнительных байтов переноса (дополнительную информацию см. в Разделе 21.3.14.4, Функция слива).

В соответствии с желаемым режимом работы программист должен установить пороги FIFO в соответствии со следующей таблицой (обратите внимание, что можно запрограммировать только пороговые значения на стороне интерфейса/OCP; на стороне I2C пороги по умолчанию равны 1). Обратите внимание, что пороговые значения должны быть установлены в соответствии с длиной канала DMA.

.

В режиме I2C Slave RX локальный хост может запрограммировать порог RX с желаемым значением и использовать функция очистки FIFO в конце передачи I2C для извлечения из FIFO оставшихся байтов, если пороговое значение не достигнуто (дополнительные сведения см. в Разделе 21.3.14.4, Функция слива).

Обратите внимание, что в режиме передачи I2C Slave пороговое значение TX FIFO должно быть установлено на 1 (I2C\_BUF.TXTRSH=0, значение по умолчанию), поскольку продолжительность передачи может быть неизвестна во время настройки. Таким образом, прерывание (или, соответственно, DMA) запросы будут генерироваться для каждого байта, запрошенного удаленным мастером I2C для передачи по шине I2C. Эта конфигурация не позволит ядру I2C запрашивать дополнительные данные от ЦП или с контроллера DMA (используя IRQ или DMA), данные, которые в конечном итоге не будут извлечены из FIFO внешним мастером (который может использовать неподтверждение в любое время для завершения передачи). Если порог TX не установлен на 1, модуль будет генерировать прерывание или устанавливать DMA только тогда, когда внешний master запрашивает байт, а FIFO пуст. Однако в этом случае TX FIFO потребуется очистить в конце передачи.

Модуль I2C предлагает пользователю возможность очистить RX или TX FIFO. Это достигается за счет регистра **I2C\_BUF.RXFIFO\_CLR** и **I2C\_BUF.TXFIFO\_CLR**, которые действуют как программный сброс для FIFO.

В режиме **DMA** эти биты также сбрасывают конечные автоматы **DMA**.

Функцию очистки FIFO можно использовать, когда модуль сконфигурирован как передатчик, внешний получатель отвечает NACK в середине передачи, и в TX FIFO все еще есть данные, ожидающие отправки.

В функциональном домене (I2C) пороги всегда можно считать равными 1. Это означает, что ядро I2C может начать передачу данных по шине I2C всякий раз, когда у него есть данные в FIFO (FIFO не пуст).

**21.3.14.4 Функция слива**

Функция слива реализована ядром I2C для обработки конца передачи, длина которой не кратно пороговому значению FIFO и предлагает возможность передать оставшееся количество байтов (поскольку порог не достигнут).

Обратите внимание, что эта функция не позволяет ЦП или контроллеру прямого доступа к памяти предпринимать больше попыток доступа к FIFO, чем необходимо (например, сгенерировать в конце передачи запрос DMA RX, имеющий в FIFO меньше байтов, чем настроенная длина передачи DMA). В противном случае будет сгенерировано прерывание по ошибке доступа (см. прерывание I2C\_IRQSTATUS\_RAW.AERR).

Механизм слива генерирует прерывание (**I2C\_IRQSTATUS\_RAW.RDR** или **I2C\_IRQSTATUS\_RAW.XDR**) в конце передачи, информируя ЦП о необходимости проверки объема данных, оставшихся для передачи (I2C\_BUFSTAT.TXSTAT или RXSTAT) и для включения Draining Feature контроллера DMA при включенном режиме DMA (путем перенастройки длины передачи DMA в соответствии с этим значением), либо выполнять только необходимое количество обращений к данным, если отключен режим DMA.

В режиме приема (ведущий или подчиненный), если порог RX FIFO не достигнут, но передача завершена на шине I2C, а в FIFO еще остались данные (меньше порогового значения), прерывание (**I2C\_IRQSTATUS\_RAW.RDR**) будет установлено, чтобы сообщить локальному хосту, что он может прочитать объем данных в FIFO (**I2C\_BUFSTAT.RXSTAT**). ЦП выполнит ряд операций чтения данных доступ равеного значению RXSTAT (в режиме прерывания или опроса) или переконфигурет контроллер DMA с помощью необходимого значение для опустошения FIFO.

В режиме основной передачи, если пороговое значение TX FIFO не достигнуто, но количество данных, оставшихся для записанное в FIFO меньше, чем TXTRSH, прерывание слива передачи (I2C\_IRQSTATUS\_RAW.XDR) будет утверждено, чтобы сообщить локальному хосту, что он может прочитать объем данных, оставшихся для записи в TX FIFO (I2C\_BUFSTAT.TXSTAT). ЦП должен будет записать требуемое количество байтов данных (указанных по значению **TXSTAT**) или перенастроить DMA-контроллер на требуемое значение, чтобы передать последний байт в FIFO.

Обратите внимание, что в ведущем режиме ЦП может альтернативно пропускать проверку значений TXSTAT и RXSTAT поскольку он может получить эту информацию внутренне (путем вычисления DATACOUNT по модулю TX/RXTHRSH).

Функция слива по умолчанию отключена, и ее можно включить с помощью **I2C\_IRQENABLE\_SET.XDR\_IE** или врегистре **I2C\_IRQENABLE\_SET.RDR\_IE** (по умолчанию отключен) только для передач неравной длины с пороговым значением.

**21.3.15 Как запрограммировать I2C**

**21.3.15.1 Настройка модуля перед включением модуля**

1. Запрограммируйте предварительный делитель для получения тактовой частоты модуля I2C приблизительно 12 МГц (I2C\_PSC = x; это значение рассчитывается и зависит от тактовой частоты системы).

2. Запрограммируйте часы I2C для получения 100 Кбит/с или 400 Кбит/с (SCLL = x и SCLH = x; эти значения рассчитываются и зависят от тактовой частоты системы).

3. Настроить собственный адрес (I2C\_OA = x) - только в случае режима работы I2C (режим F/S).

4. Вывести модуль I2C из состояния сброса (I2C\_CON:I2C\_EN = 1).

**21.3.15.2 Процедура инициализации**

1. Настройте биты регистра режима I2C (I2C\_CON).

2. Включить маски прерываний (I2C\_IRQENABLE\_SET), если прерывание используется для передачи/приема данных.

3. Включите DMA (I2C\_BUF и I2C\_DMA/RX/TX/ENABLE\_SET) и запрограммируйте контроллер DMA) - только в случае режима работы I2C (режим F/S), если для передачи/приема данных используется DMA.

**21.3.15.3 Настройка адреса ведомого устройства и регистров счетчика данных**

В ведущем режиме настройте адрес ведомого (I2C\_SA = x) и количество байтов, связанных с передачей (I2C\_CNT = x).

**21.3.15.4 Инициировать передачу**

Опросите бит занятости шины (BB) в регистре состояния I2C (I2C\_IRQSTATUS\_RAW). Если он сброшен на 0 (шина не занята), настройте START/STOP (I2C\_CON: STT / I2C\_CON: условие STP для инициирования передачи) - только в случае режима работы I2C (режим F/S).

**21.3.15.5 Получение данных**

Опросите бит флага прерывания готовности к приему данных (RRDY) в регистре состояния I2C (I2C\_IRQSTATUS\_RAW), прерывание RRDY (установлен I2C\_IRQENABLE\_SET.RRDY\_IE) или прерывание DMA RX (I2C\_BUF.RDMA\_EN устанавливается вместе с I2C\_DMARCXENABLE\_SET) для чтения полученных данных в регистре приема данных (I2C\_DATA). Используйте функцию слива (I2C\_IRQSTATUS\_RAW.RDR включенную I2C\_IRQENABLE\_SET.RDR\_IE)) если длина передачи не равна порогу FIFO.

**21.3.15.6 Передача данных**

Опросите бит флага прерывания готовности передачи данных (XRDY) в регистре состояния I2C (I2C\_IRQSTATUS\_RAW), используйте прерывание XRDY (установлено I2C\_IRQENABLE\_SET.XRDY\_IE) или прерывание DMA TX (I2C\_BUF.XDMA\_EN устанавливается вместе с I2C\_DMATXENABLE\_SET) для записи данных в регистр передачи данных (I2C\_DATA). Используйте функцию слива (I2C\_IRQSTATUS\_RAW.XDR включена I2C\_IRQENABLE\_SET.XDR\_IE)) при передаче длины не равной порогу FIFO.

**21.3.16 Поведение I2C во время эмуляции**

Чтобы настроить I2C для остановки во время событий приостановки эмуляции (например, точек останова отладчика), настройте I2C и подсистема отладки:

1. Установите I2C\_SYSTEST.FREE=0. Это позволит сигналу Suspend\_Control от подсистемы отладки (Глава 27), чтобы остановить и запустить I2C. Обратите внимание, что если FREE=1, сигнал Suspend\_Control игнорируется и I2C работает независимо от любого события приостановки отладки. Этот БЕСПЛАТНЫЙ бит дает локальное управление из перспективы модуля для блокировки сигнала приостановки, поступающего из подсистемы отладки.

2. Установите соответствующий регистр xxx\_Suspend\_Control = 0x9, как описано в Разделе 27.1.1.1, Приостановка поддержку периферийных устройств во время отладки. Выберите регистр, соответствующий периферийному устройству, которое вы хотите приостановить во время события приостановки.