**19.Universal Asynchronous Receiver/Transmitter (UART)**

**19.1 Введение**

**19.1.1 Фичи режима UART**

Общие характеристики модуля UART/IrDA при работе в режиме UART:

- Совместимость с 16C750.

- Скорость передачи от 300 бит/с до 3,6864 Мбит/с.

- Автопереключение между 1200 бит/с и 115,2 кбит/с.

- Программное/аппаратное управление потоком.

- Программируемые символы Xon/Xoff.

– Программируемый Auto-RTS и Auto CTS.

- Характеристики программируемого последовательного интерфейса.

– 5, 6, 7 или 8-битные символы.

– Генерация четных, нечетных, меток (всегда 1), пробелов (всегда 0) или битов без четности (битовый кадр без четности) и обнаружения.

– Генерация 1, 1,5 или 2 стоповых бита.

- Обнаружение ложного стартового бита.

- Генерация и обнаружение разрыва строки.

- Функции управления модемом (CTS, RTS, DSR, DTR, RI и DCD).

- Управление системой прерываний с полным приоритетом.

- Возможности внутреннего тестирования и замыкания на себя.

**19.1.2 Особенности режима ИК-порта(IrDA)**

Общие характеристики **UART/IrDA** при работе в режиме **IrDA**:

- Поддержка связи **IrDA** 1.4 в медленном инфракрасном диапазоне (**SIR**), среднем инфракрасном диапазоне (**MIR**) и быстром инфракрасном диапазоне (**FIR**).

(очень быстрый инфракрасный порт (**VFIR**) не поддерживается).

- Форматирование кадра: добавление переменных символов **xBOF** и символов **EOF**.

- Генерация/обнаружение **CRC** восходящей/нисходящей линии связи.

- Асинхронная прозрачность (автоматическая вставка символа разрыва).

- **FIFO** состояния на 8 записей (с выбираемыми уровнями срабатывания), доступный для контроля длины кадра и ошибок кадра.

- Ошибка кадрирования, ошибка проверки циклическим избыточным кодом (**CRC**), недопустимый символ (**FIR**), шаблон прерывания (**SIR**, **MIR**).

**19.1.3 Особенности режима CIR**

Общие характеристики **UART**/**IrDA** при работе в режиме **CIR**:

- Поддержка потребительского инфракрасного излучения (**CIR**) для приложений дистанционного управления.

- Передача и получение.

- Свободный формат данных (поддерживает любые частные стандарты удаленного управления).

- Выбираемая скорость передачи данных.

- Настраиваемая несущая частота.

- Рабочий цикл 1/2, 5/12, 1/3 или 1/4 несущей.

**19.1.4 Неподдерживаемые функции UART**

Следующие функции модуля **UART**/**IrDA** не поддерживаются этим устройством:

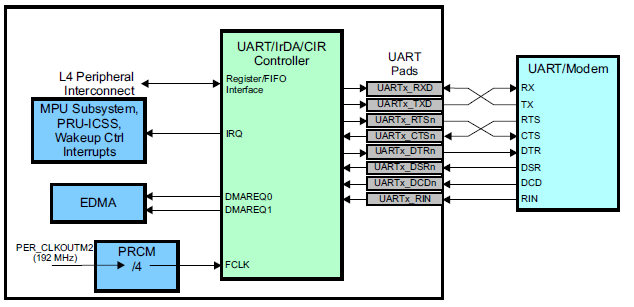
**Таблица 19-1. Неподдерживаемые функции UART**

|  |  |
| --- | --- |
| Feature | Reason |
| Full modem control on UART0 | DCD, DSR, DTR, RI not pinned-out |
| Full modem control on UART2-5 | DCD, DSR, DTR, RI not pinned-out |
| Device wake-up on UART1-5 | Wake-up not supported - no SWake connection |
| DMA mode 2 and 3 | Only DMA mode 0 and 1 is supported with EDMA |

**19.2 Интеграция**

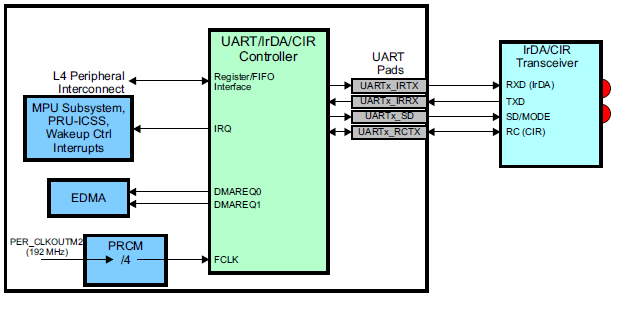
Это устройство содержит 6 экземпляров периферийного устройства **UART**/**IrDA** (**UARTIRDAOCP**). Есть шесть **UART** модулей, они называются **UART0** – **UART5**. **UART0** обеспечивает возможность пробуждения. Только **UART1** обеспечивает управляющие сигналы полноценного модема. Все **UART** поддерживают режимы **IrDA** и **CIR**, а также управление потоком **RTS**/**CTS** (с учетом мультиплексирования контактов конфигурации). На **рис. 19-1** показан пример соединения системы с использованием связи **UART** с аппаратным рукопожатием.

**Рисунок 19-1. Модуль UART/IrDA — Приложение UART**



На рис. 19-2 показан пример подключения системы с помощью инфракрасной связи с дистанционным управлением.(бытовой инфракрасный).

**Рисунок 19-2. Модуль UART/IrDA — Приложение IrDA/CIR**



**19.2.1 Атрибуты подключения UART**

Общие атрибуты подключения для каждого из модулей UART показаны в Таблице 19-2 и Таблице 19-3.

**Таблица 19-2. Атрибуты подключения UART0**

|  |  |
| --- | --- |
| Attributes | Type |
| Power Domain | Wake-Up Domain |
| Clock Domain | PD\_WKUP\_L4\_WKUP\_GCLK (OCP)  PD\_WKUP\_UART0\_GFCLK (Func) |
| Reset Signals | WKUP\_DOM\_RST\_N |
| Idle/Wakeup Signals | Smart Idle / Wakeup |
| Interrupt Requests | 1 interrupt to MPU Subsystem (UART0INT), PRU-ICSS (nirq)  and WakeM3 |
| DMA Requests | 2 DMA requests to EDMA (TX – UTXEVT0, RX – URXEVT0) |
| Physical Address | L4 Wakeup slave port |

**Таблица 19-3. Атрибуты подключения** **UART1-5**

|  |  |
| --- | --- |
| Attributes | Type |
| Power Domain | Peripheral Domain |
| Clock Domain | PD\_PER\_L4LS\_GCLK (OCP)  PD\_PER\_UART\_GFCLK (Func) |
| Reset Signals | PER\_DOM\_RST\_N |
| Idle/Wakeup Signals | Smart Idle |
| Interrupt Requests | UART1-2  1 interrupt per instance to MPU Subsystem (UART1INT,  UART2INT) and PRU-ICSS (nirq)  UART3-5  1 interrupt per instance to only MPU Subsystem (UART3INT,  UART4INT, UART5INT) |
| DMA Requests | 2 DMA requests per instance to EDMA (TX – UTXEVTx, RX –  URXEVTx) |
| Physical Address | L4 Peripheral slave port |

**19.2.2 Управление синхронизацией и сбросом UART**

Модули UART используют отдельные функциональные и шинные интерфейсы тактирования.

**Таблица 19-4. Сигналы тактирования UART0**

|  |  |  |  |
| --- | --- | --- | --- |
| Clock Signal | Max Freq | Reference / Source | Comments |
| CLK  Interface clock  From PRCM | 100 MHz | CORE\_CLKOUTM4 / 2 | pd\_wkup\_l4\_wkup\_gclk |
| FCLK  Functional clock  From PRCM | 48 MHz | PER\_CLKOUTM2 / 4 | pd\_wkup\_uart0\_gfclk |

**Таблица 19-5. UART1-5 Тактовые сигналы**

|  |  |  |  |
| --- | --- | --- | --- |
| Clock Signal | Max Freq | Reference / Source | Comments |
| CLK  Interface clock | 100 MHz | CORE\_CLKOUTM4 / 2 | pd\_per\_l4ls\_gclk  From PRCM |
| FCLK  Functional clock | 48 MHz | PER\_CLKOUTM2 / 4 | pd\_per\_uart\_gfclk  From PRCM |

Для работы UART функциональное тактирование используются для обеспечения скорости передачи данных до 3,6 Мбит/с. Табл. 19-6 показан список настройки скоростей передачи от делителя частоты

**Таблица 19-6. Baud-rate UART и частоты ошибок**

|  |  |  |  |
| --- | --- | --- | --- |
| Baud rate | Over sampling | Divisor | Error (%) |
| 300 | **16** | **10000** | **0** |
| 600 | **16** | **5000** | **0** |
| 1200 | **16** | **2500** | **0** |
| 2400 | **16** | **1250** | **0** |
| 4800 | **16** | **625** | **0** |
| 9600 | **16** | **313** | **0.16** |
| 14400 | **16** | **208** | **0.16** |
| 19200 | **16** | **156** | **0.16** |
| 28800 | **16** | **104** | **0.16** |
| 38400 | **16** | **78** | **0.16** |
| 57600 | **16** | **52** | **0.16** |
| 115200 | **16** | **26** | **0.16** |
| 230400 | **16** | **13** | **0.16** |
| 460800 | **13** | **8** | **0.16** |
| 921600 | **13** | **4** | **0.16** |
| 1843200 | **13** | **2** | **0.16** |
| 3000000 | **16** | **1** | **0** |
| 3686400 | **13** | **1** | **0.16** |

Для работы IrDA внутренний функциональный делитель тактовых сигналов позволяет генерировать скорости SIR, MIR или FIR в бодах как показано в таблице 19-7.

**Таблица 19-7. Скорость передачи в режиме IrDA и частота ошибок**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Baud rate | IR mode | Encoding | Divisor | Error (%) |
| 2400 | SIR | 3/16 | 1250 | 0 |
| 9600 | SIR | 3/16 | 312 | 0.16 |
| 19200 | SIR | 3/16 | 156 | 0.16 |
| 38400 | SIR | 3/16 | 78 | 0.16 |
| 57600 | SIR | 3/16 | 52 | 0.16 |
| 115200 | SIR | 3/16 | 26 | 0.16 |
| 576000 | MIR | 1/4 | 2 | 0 |
| 1152000 | MIR | 1/4 | 1 | 0 |
| 4000000 | FIR | 4PPM | 1 | 0 |

**19.2.3 Список контактов UART**

Контакты интерфейса UART перечислены в таблице 19-8. Функциональность контактов зависит от выбранного режима модуля.

**Таблица 19-8. Список выводов UART**

|  |  |  |
| --- | --- | --- |
| Pin | Type | Description |
| UARTx\_RXD / IRRX / RCRX | I | UART / IrDA / CIR Receive Data |
| UARTx\_TXD / IRTX / RCTX | OZ | UART / IrDA / CIR Transmit Data |
| UARTx\_RTSn / SD | OZ | UART Request to Send / IrDA Mode |
| UARTx\_CTSn | I | UART Clear to Send |
| UARTx\_DTRn(1) | OZ | UART Data Terminal Ready |
| UARTx\_DSRn(1) | I | UART Data Set Ready |
| UARTx\_DCDn(1) | I | UART Data Carrier Detect |
| UARTx\_RIn(1) | I | UART Ring Indicator |

(1) UART1 only

Модуль UART может работать в трех различных режимах на основе MODE\_SELECT битов. Сигнал мультиплексирование на основе этих битов режима показано в таблице 19-9.

**Таблица 19-9. Управление мультиплексированием UART**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UARTx\_TXD / IRTX /  RCTX Function | UARTx\_RXD / IRRX / RCRX  Function | UARTx\_RTSn / SD  Function | UARTx\_CTSn  Function | Mode |
| TXD | RXD | RTSn | CTSn | UART |
| IRTX | IRRX | SD | not used | IrDA (SIR, MIR, FIR) |
| RCTX | RCRX | SD | not used | CIR |

**19.3 Функциональное описание**

**19.3.1 Блок-схема**

Модуль **UART/IrDA/CIR** можно разделить на три основных блока:

- Управление FIFO

- Выбор режима

- Форматирование протокола

Управление FIFO является общим для всех функций и позволяет передавать и получать данные с позиции хост-процессора.

Есть два режима:

- Функциональный режим: направляет данные в выбранную функцию (**UART**, **IrDA** или **CIR**) и включает

механизм, соответствующий выбранной функции

- Режим регистрации: разрешает условный доступ к регистрам.

Для получения дополнительной информации о настройке режима см. Раздел 19.3.7, Выбор режима.

Форматирование протокола имеет три подкатегории:

- Генерация тактового сигнала: Входной тактовый сигнал 48 МГц генерирует все необходимые тактовые сигналы.

- Форматирование данных: каждая функция использует свой конечный автомат, отвечающий за переход

между данными FIFO и данными кадра, связанными с ним.

- Управление прерываниями: в зависимости от выбранной функции генерируются различные типы прерываний:

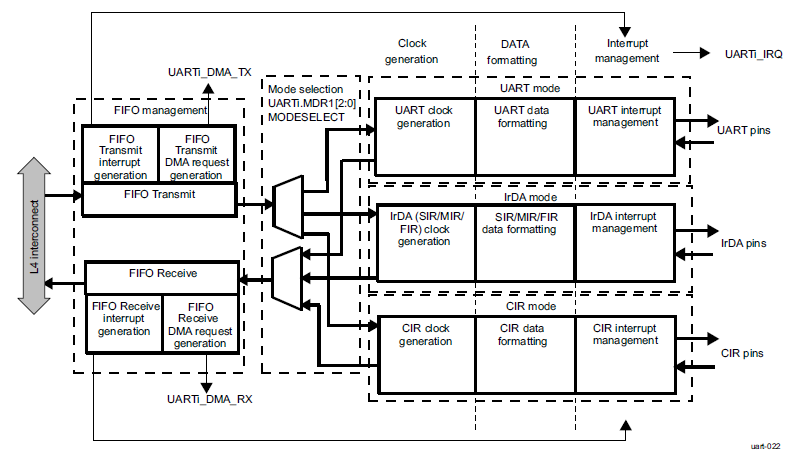
- Прерывания режима **UART**: семь прерываний с приоритетом на шести различных уровнях.

- Прерывания в режиме **IrDA**: восемь прерываний. Линия прерывания активируется при генерации любого прерывания (нет приоритета).

– Прерывания режима **CIR**: используется подмножество существующих прерываний режима **IrDA**.

В каждом режиме при генерации прерывания регистр **UART\_IIR** указывает тип прерывания. Параллельно с этими функциональными блоками для каждой функции существует стратегия энергосбережения. На рис. 19-3 представлена ​​блок-схема **UART/IrDA/CIR**.

**Рис. 19-3. Блок-схема функциональных спецификаций UART/IrDA/CIR**



**19.3.2 Конфигурация тактирования**

Каждый **UART** использует тактовую частоту 48 МГц для своей логики и для генерации сигналов внешнего интерфейса. Каждый **UART** использует интерфейсное тактирование для доступа к регистрам. Модуль **PRCM** генерирует и контролирует все эти тактирования (дополнительную информацию см. в разделе Атрибуты модуля **Clock Domain** в Главе 8, Питание, Сброс и управление тактированием). Процессы бездействия и пробуждения используют протокол квитирования между **PRCM** и **UART** (для описание протокола см. в разделе «Управление тактированием на уровне модуля» в главе 8 «Питание, сброс и управление тактированием"). Битовое поле **UARTi.UART\_SYSC[4:3] IDLEMODE** управляет режимом ожидания **UART**.

**19.3.3 Программный сброс**

Бит **UARTi.UART\_SYSC[1] SOFTRESET** управляет программным сбросом; установка этого бита в **1** запускает программный сброс функционально эквивалентеный аппаратному сбросу.

**19.3.4 Управление питанием**

**19.3.4.1 Управление питанием в режиме UART**

**19.3.4.1.1 Энергосбережение модуля**

В режимах **UART** спящий режим включается установкой бита **UARTi.UART\_IER[4] SLEEP\_MODE** в **1** (когдабит **UARTi.UART\_EFR[4] ENHANCED\_EN** установлен в 1).

Переход в спящий режим осуществляется при соблюдении всех следующих условий:

- Линия последовательного ввода данных, **uarti\_rx**, не занята.

- **TX** **FIFO** и сдвиговый регистр **TX** пусты.

- **RX** **FIFO** пуст.

- Единственными ожидающими прерываниями являются прерывания **THR**.

Спящий режим — хороший способ снизить энергопотребление **UART**, но это состояние может быть достигнуто только если **UART** установлен в режим модема. Поэтому, даже если **UART** не задействован, он должен инициализироваться в функциональном режиме, чтобы использовать спящий режим. В спящем режиме тактирование модуля и тактирование битрейта передачи останавливаются . Поскольку большинство регистров тактируется этими источниками, это значительно снижает энергопотребление. Модуль просыпается при изменении  на линии **uarti\_rx**, при записи данных в **TX FIFO** и при изменении состояние входных контактов модема. Прерывание может быть сгенерировано по событию пробуждения, установив **UARTi.UART\_SCR[4] RX\_CTS\_WU\_EN** бит в **1**. Чтобы понять, как управлять прерыванием, см. Раздел 19.3.5.2, Прерывание пробуждения.

***ПРИМЕЧАНИЕ. Не следует записывать что либо в защелки делителя, UARTi.UART\_DLL и UARTi.UART\_DLH, для установки тактирования битрейта (BCLK) в спящем режиме. Желательно сначала отключить спящий режим с помощью бита UARTi.UART\_IER[4] SLEEP\_MODE перед записью в регистр UARTi.UART\_DLL или регистр UARTi.UART\_DLH.***

**19.3.4.1.2 Энергосбережение системы**

Спящий режим и автоматический режим ожидания — это встроенные функции энергосбережения. Методы снижения мощности могут применяется на системном уровне путем отключения определенного внутреннего тактирования и доменов питания устройства. **UART** поддерживает протокол квитирования запроса/подтверждения бездействия, используемый на системном уровне для отключения **UART** синхронизируется чистым и контролируемым образом, а также переключает **UART** из режима генерации прерываний в режим генерации пробуждения для немаскированных событий (см. бит **UARTi.UART\_SYSC[2] ENAWAKEUP** и регистр **UARTi.UART\_WER**).

Дополнительные сведения см. в разделе «Управление чтактированием на уровне модуля» в главе 8 «Питание, сброс и управление тактированием".

**19.3.4.2 Управление питанием в режиме IrDA/CIR**

**19.3.4.2.1 Энергосбережение модуля**

В режимах **IrDA**/**CIR** спящий режим включается установкой бита **UARTi.MDR[3] IR\_SLEEP** в **1**.Переход в спящий режим осуществляется при соблюдении всех следующих условий:

- Линия ввода последовательных данных, uarti.rx\_irrx, не занята.

- TX FIFO и сдвиговый регистр TX пусты.

- RX FIFO пуст.

- Нет ожидающих прерываний, кроме прерываний THR.

Модуль просыпается при обнаружении изменения в строке **uarti\_rx\_irrx** или при записи данных в **TX FIFO**.

**19.3.4.2.2 Энергосбережение системы**

Энергосбережение системы для режима **IrDA**/**CIR** имеет ту же функцию, что и для режима **UART** (см.Раздел 19.3.4.1.2, Энергосбережение системы).

**19.3.4.3 Локальное управление питанием**

В Таблице 19-10 описаны функции управления питанием, доступные для **UART**.

***ПРИМЕЧАНИЕ. Информацию о стробировании исходного тактового сигнала и описании переходов в спящий/пробуждающий режим см.***

***Управление тактированием на уровне модуля в Главе 8, Управление питанием, сбросом и тактированием.***

**Таблица 19-10. Локальные функции управления питанием**

|  |  |  |
| --- | --- | --- |
| Feature | Registers | Description |
| Clock autogating | UART\_SYSC[0] AUTOIDLE | Этот бит обеспечивает локальную оптимизацию мощности в модуле посредством стробирования UARTi\_ICLK интерфейсного тактирования или стробирования UARTi\_FCLK функционального тактирования внутренней активности. |
| Slave idle modes | UART\_SYSC[4:3] IDLEMODE | Доступны: режим форсированного простоя, режим отсутствия простоя, режим умного простоя и режим пробуждения с возможностью умного простоя. |
| Clock activity | N/A | Функция недоступна |
| Master standby  modes | N/A | Функция недоступна |
| Global wake-up  enable | UART\_SYSC[2] ENAWAKEUP | Этот бит активизирует функцию пробуждения на уровне модуля. |
| Wake-Up sources  enable | N/A | Функция недоступна |

**19.3.5 Запросы на прерывание**

Модуль **UART** **IrDA** **CIR** генерирует прерывания. Все прерывания могут быть разрешены/запрещены путем записи соотвествтующего бита в регистр разрешения прерываний (**IER**). Статус прерывания устройства можно проверить в любое время, читая регистр идентификации прерывания (**IIR**). Режимы **UART**, **IrDA** и **CIR** имеют

разные прерывания в модуле **UART** **IrDA** **CIR** и, следовательно, имеют разные отображения **IER** и **IIR** в соответствии с выбранным режимом.

**19.3.5.1 Управление прерываниями режима UART**

**19.3.5.1.1 Прерывания UART**

Режим **UART** включает семь возможных прерываний с шестью уровнями приоритета. Когда генерируется прерывание, регистр идентификации прерывания (**UARTi.UART\_IIR**) устанавливает **UARTi.UART\_IIR[0]** бит **IT\_PENDING** равен **0**, чтобы указать, что прерывание ожидает обработки, и указывает тип прерывания.

прерывание через битовое поле **UARTi.UART\_IIR[5:1]**. Таблица 19-11 суммирует управление фунциями перываний.

**Таблица 19-11. Прерывания режима UART**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UART\_IIR[5:0] | Priority Level | Interrupt Type | Interrupt Source | Interrupt Reset Method |
| 0x000001 | None | None | None | None |
| 0x000110 | 1 | Receiver line  status | OE, FE, PE, or BI errors occur in  characters in the RX FIFO. | FE, PE, BI: Read the UART\_RHR  register. OE: Read the UART\_LSR  register. |
| 0x001100 | 2 | RX time-out | Stale data in RX FIFO | Read the UART\_RHR register. |
| 0x000100 | 2 | RHR interrupt | DRDY (data ready) (FIFO  disable)  RX FIFO above trigger level  (FIFO enable) | Read the UART\_RHR register until the  interrupt condition disappears. |
| 0x000010 | 3 | THR interrupt | TFE (UART\_THR empty) (FIFO  disable)  TX FIFO below trigger level (FIFO  enable) | Write to the UART\_THR until the  interrupt condition disappears. |
| 0x000000 | 4 | Modem status | See the UART\_MSR register. | Read the UART\_MSR register. |
| 0x010000 | 5 | XOFF interrupt/special  character  interrupt | Receive XOFF characters/special  character | Receive XON character(s), if XOFF  interrupt/read of the UART\_IIR register,  if special character interrupt. |
| 0x100000 | 6 | CTS, RTS, DSR | RTS pin or CTS pin or DSR  change state from active (low) to  inactive (high). | Read the UART\_IIR register. |

Для прерывания по линии **RX** бит **RX\_FIFO\_STS (UARTi.UART\_LSR[7])** генерирует прерывание. Для прерывания **XOFF**, если символ потока **XOFF** вызвал прерывание, прерывание сбрасывается только если обнаружен символ потока **XON**. Если обнаружение специального символа вызвало прерывание, прерывание

очищается чтением регистра **UARTi.UART\_IIR**.

**19.3.5.2 Прерывание побудки**

Пробуждающее прерывание — это специальное прерывание, которое работает иначе, чем другие прерывания. Это прерывание включено, когда бит **UARTi.UART\_SCR[4] RXCTSDSRWAKEUPENABLE** установлен в **1**. При этом регистр **UARTi.UART\_IIR** не изменяется;В регистре **UARTi.UART\_SSR[1]** бит **RXCTSDSRWAKUPSTS** должен быть проверен для обнаружения события пробуждения. Когда происходит прерывание пробуждения, его можно сбросить только путем сброса бита **RXCTSDSRWAKEUPENABLE** в **UARTi.UART\_SCR[4]** . Этот бит должен быть повторно включен (установлен в **1**) после текущего пробуждения. Прерывание обрабатывается для обнаружения следующего входящего события пробуждения. Прерывание пробуждения также может произойти, если бит **WER[7] TXWAKEUPEN** установлен в **1** и выполняется одно из следующих условий:

имеет место:

- Произошло прерывание **THR**, если оно разрешено (опущено, если включен запрос **TX DMA**).

- Возник запрос **TX DMA**, если он включен.

- Возникло сообщение **TX\_STATUS\_IT**, если оно включено (только режимы **IrDA** и **CIR**). Нельзя использовать с прерыванием от **THR**.

***ВНИМАНИЕ***

***Реализация интерфейса пробуждения в режиме IrDA основана на***

***UARTi\_SIDLEACK переходе от низкого уровня к высокому вместо просто проверки состояния UARTi\_SIDLEACK. Это не гарантирует генерацию события пробуждения, как ожидается, при настройке в интеллектуальном режиме ожидания, и система на короткое время выходит из спящего режима.***

**19.3.5.3 Управление прерываниями в режиме IrDA**

**19.3.5.3.1 Прерывания ИК-порта**

Функция **IrDA** генерирует прерывания. Все прерывания могут быть разрешены и запрещены путем записи в соответствующий бит в регистре разрешения прерывания (**UARTi.UART\_IER**). Состояние прерывания устройства может быть проверено чтением регистра идентификации прерывания (**UARTi.UART\_IIR**). Режимы **UART**, **IrDA** и **CIR** имеют разные прерывания в модуле **UART/IrDA/CIR** и, следовательно, различные отображения **UARTi.UART\_IER** и **UARTi.UART\_IIR**, в зависимости от выбранного режима.

Режимы **IrDA** имеют восемь возможных прерываний (см. Таблицу 19-12). Линия прерывания активируется при любом генерируется прерывание (нет приоритета).

**Таблица 19-12. Прерывания режима IrDA**

|  |  |  |  |
| --- | --- | --- | --- |
| UART\_IIR Bit | Interrupt Type | Interrupt Source | Interrupt Reset Method |
| 0 | RHR interrupt | DRDY (data ready) (FIFO  disable)  RX FIFO above trigger level  (FIFO enable) | Read the UART\_RHR register until the  interrupt condition disappears. |
| 1 | THR interrupt | TFE (UART\_THR empty)  (FIFO disable)  TX FIFO below trigger level  (FIFO enable) | Write to the UART\_THR until the interrupt  condition disappears. |
| 2 | Last byte in RX FIFO | Last byte of frame in RX FIFO  is available to be read at the  RHR port | Read the UART\_RHR register. |
| 3 | RX overrun | Write to the UART\_RHR  register when the RX FIFO is  full. | Read UART\_RESUME register |
| 4 | Status FIFO interrupt | Status FIFO triggers level  reached. | Read STATUS FIFO. |
| 5 | TX status | 1. UART\_THR empty before  EOF sent. Last bit of  transmission of the IrDA  frame occurred, but with  an underrun error.  OR  2. Transmission of the last bit  of the IrDA frame  completed successfully. | 1. Read the UART\_RESUME register.  OR  2. Read the UART\_IIR register |
| 6 | Receiver line status interrupt | CRC, ABORT, or frame-length  error is written into the  STATUS FIFO. | Read the STATUS FIFO (read until empty  - maximum of eight reads required). |
| 7 | Received EOF | Received end-of-frame | Read the UART\_IIR register. |

**19.3.5.4 Управление прерываниями  в режиме CIR**

**19.3.5.4.1 Прерывания CIR**

Функция **CIR** генерирует прерывания, которые можно разрешать и запрещать записью в соответствующий бит в регистр разрешения прерывания (**UARTi.UART\_IER**). Статус прерывания устройства можно проверить с помощью чтение регистра идентификации прерывания (**UARTi.UART\_IIR**). Режимы **UART, IrDA и CIR** имеют разные прерывания в модуле **UART/IrDA/CIR** и, следовательно, различные отображения **UARTi.UART\_IER** и **UARTi.UART\_IIR**, в зависимости от выбранного режима.

В Таблице 19-13 перечислены поддерживаемые режимы прерывания. В режиме **CIR** единственная цель бит **UARTi.UART\_IIR[5]** указывает, что последний бит инфракрасных данных был передан на вывод **uart\_cts\_rctx**.

**Таблица 19-13. Прерывания режима CIR**

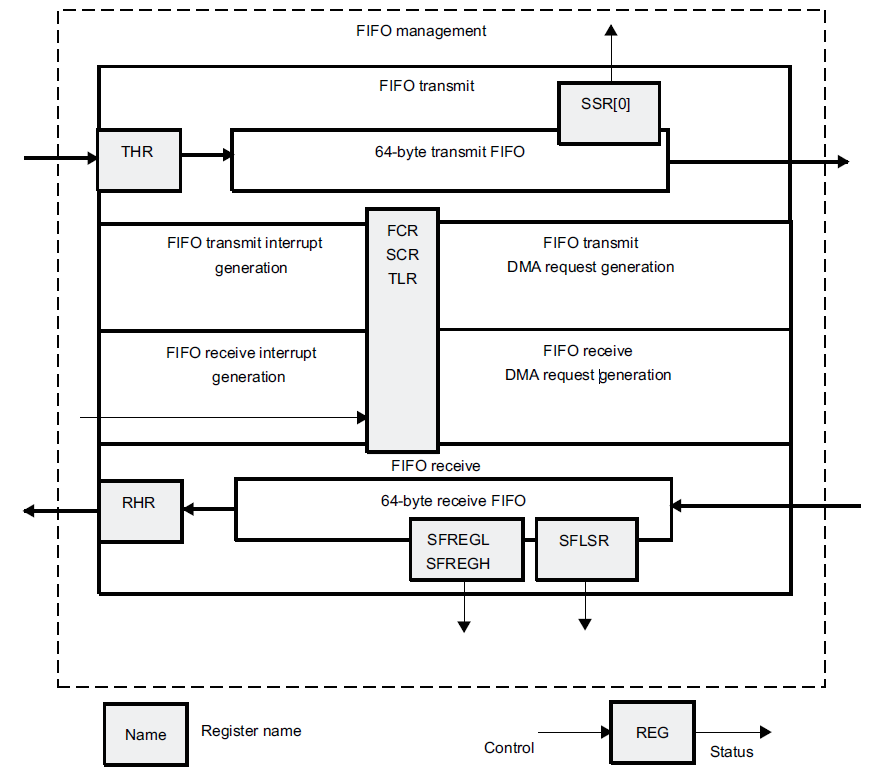
|  |  |  |  |
| --- | --- | --- | --- |
| UART\_IIR Bit  Number | Interrupt Type | Interrupt Source | Interrupt Reset Method |
| 0 | RHR interrupt | DRDY (data ready) (FIFO disable)  RX FIFO above trigger level (FIFO  enable) | Read UART\_RHR until interrupt condition  disappears. |
| 1 | THR interrupt | TFE (UART\_THR empty) (FIFO  disable)  TX FIFO below trigger level (FIFO  enable) | Write to the UART\_THR register until the  interrupt condition disappears. |
| 2 | RX\_STOP\_IT | Receive stop interrupt (depending  on value set in the BOF Length  Register (UART\_EBLR). | Read IIR |
| 3 | RX overrun | Write to RHR when RX FIFO is full. | Read RESUME register. |
| 4 | N/A for CIR mode | N/A for CIR mode | N/A for CIR mode |
| 5 | TX status | Transmission of the last bit of the  frame is complete successfully. | Read the UART\_IIR register. |
| 6 | N/A for CIR mode | N/A for CIR mode | N/A for CIR mode |
| 7 | N/A for CIR mode | N/A for CIR mode | N/A for CIR mode |

**19.3.6 Управление FIFO**

Доступ к **FIFO** осуществляется путем чтения и записи регистров **UARTi.UART\_RHR** и **UARTi.UART\_THR**. Управление параметрами осуществляется с помощью регистра управления **FIFO** (**UARTi.UART\_FCR**) и дополнительного управления регистр (**UARTi.UART\_SCR**). Чтение бита **UARTi.UART\_SSR[0] TX\_FIFO\_FULL** в **1** означает, что **FIFO** полон. Регистр **UARTi.UART\_TLR** управляет уровнем запуска **FIFO**, который включает **DMA** и генерацию прерывания. После сброса **FIFO** для передачи (**TX**) и приема (**RX**) отключены; таким образом, триггерный уровень значение по умолчанию 1 байт. На рис. 19-4 показаны регистры управления **FIFO**.

***ПРИМЕЧАНИЕ. Данные в регистре UARTi.UART\_RHR не перезаписываются при переполнении.***

***ПРИМЕЧАНИЕ. Состояние UARTi.UART\_SFLSR, UARTi.UART\_SFREGL и UARTi.UART\_SFREGH регистры используются только в режиме IrDA. Для использования см. Раздел 19.3.8.2.6, Форматирование данных ИК-порта.***

**Рис. 19-4. Регистры управления FIFO** 

**19.3.6.1 Триггер FIFO**

**19.3.6.1.1 Триггер передачи FIFO**

В Таблице 19-14 перечислены настройки уровня запуска TX FIFO.

**Таблица 19-14. Сводка настроек уровня триггера TX FIFO**

|  |  |  |
| --- | --- | --- |
| UART\_SCR[6] | UART\_TLR[3:0] | TX FIFO Trigger Level |
| 0 | =0x0 | Определяется битовым полем **UARTi.UART\_FCR[5:4] TX\_FIFO\_TRIG** (8,16, 32 или 56 пробелов) |
| 0 | !=0x0 | Определяется битовым полем **UARTi.UART\_TLR[3:0] TX\_FIFO\_TRIG\_DMA** (от 4 до 60 пробелов с детализацией в  4 пробела) |
| 1 | value | Определяется конкатенированным значением **RX\_FIFO\_TRIG\_DMA** и  **RX\_FIFO\_TRIG** (от 1 до 63 символов с точностью до 1 символа) *Примечание*. *Комбинация RX\_FIFO\_TRIG\_DMA = 0x0 и RX\_FIFO\_TRIG =*  *0x0 (все нули) не поддерживается (требуется минимум один символ). Все нули могут привести к непредсказуемому поведению.* |

**19.3.6.1.2 Триггер приема FIFO**

В Таблице 19-15 перечислены настройки уровня срабатывания **RX FIFO**.

**Таблица 19-15. Сводка настроек уровня триггера RX FIFO**

|  |  |  |
| --- | --- | --- |
| UART\_SCR[7] | UART\_TLR[7:4] | RX FIFO Trigger Level |
| 0 | =0x0 | Определяется битовым полем **UARTi.UART\_FCR[7:6] RX\_FIFO\_TRIG** (8,16, 56 или 60 символов) |
| 0 | !=0x0 | Определяется битовым полем **UARTi.UART\_TLR[7:4] RX\_FIFO\_TRIG\_DMA** (от 4 до  60 символов с точностью до 4 символов) |
| 1 | value | Определяется конкатенированным значением **RX\_FIFO\_TRIG\_DMA** и  **RX\_FIFO\_TRIG** (от 1 до 63 символов с точностью до 1 символа)  *Примечание. Комбинация RX\_FIFO\_TRIG\_DMA = 0x0 и RX\_FIFO\_TRIG =*  *0x0 (все нули) не поддерживается (требуется минимум один символ). Все нули могут привести к непредсказуемому поведению.* |

Порог приема программируется с помощью **UARTi.UART\_TCR[7:4] RX\_FIFO\_TRIG\_START** и битовыми полями **UARTi.UART\_TCR[3:0] RX\_FIFO\_TRIG\_HALT**:

- Уровни запуска от 0 до 60 байт доступны с шагом 4 (уровень запуска = 4 x [4-битный величина регистра]).

- Чтобы обеспечить правильную работу устройства, убедитесь, что **RX\_FIFO\_TRIG\_HALT** **RX\_FIFO\_TRIG**  включены во время режима авто-RTS.

Задержка = [4 + 16 x (1 + CHAR\_LENGTH + Parity + Stop 0.5)] x Baud\_rate + 4 x FCLK

*ПРИМЕЧАНИЕ. Сигнал* ***RTS*** *сбрасывается после того, как модуль UART получает данные по* ***RX\_FIFO\_TRIG\_HALT****. Задержка означает, сколько времени требуется модулю UART для отмены подтверждения* ***RTS*** *сигнал после достижения* ***RX\_FIFO\_TRIG\_HALT****.*

- В режиме прерывания **FIFO** с управлением потоком убедитесь, что уровень срабатывания для остановки передачи больше или равен уровню срабатывания **RX FIFO** ( это бит **UARTi.UART\_TCR[7:4] RX\_FIFO\_TRIG\_START** или бит **UARTi.UART\_FCR[7:6] RX\_FIFO\_TRIG**); в противном случае операция **FIFO** остановится. В **FIFO** режима с **DMA**  управлением потоком этого понятия не существует, потому что запрос **DMA** отправляется, когда байт получен.

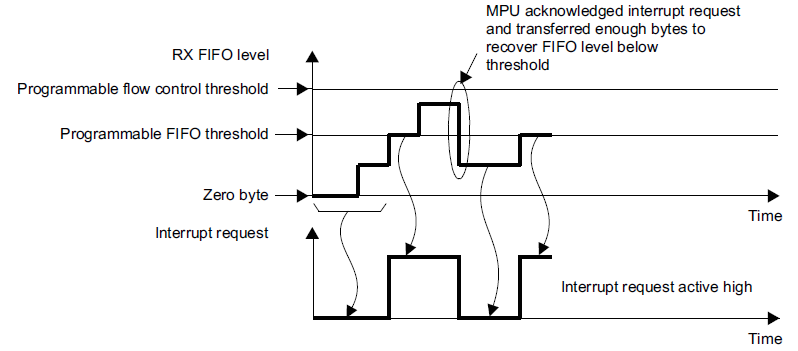
**19.3.6.2 Режим прерывания FIFO**

В режиме прерывания **FIFO** (в регистре управления **FIFO** **UARTi.UART\_FCR[0]** бит **FIFO\_EN** устанавливается в **1** и соответствующие прерывания разрешены регистром **UARTi.UART\_IER**), сигнал прерывания информирует процессор состояния приемника и передатчика. Эти прерывания возникают, когда порог **RX/TX FIFO** достигается  **(**в битовом поле **UARTi.UART\_TLR[7:4] RX\_FIFO\_TRIG\_DMA** и **UARTi.UART\_TLR[3:0**] бит **TX\_FIFO\_TRIG\_DMA** или в битовом поле **UARTi.UART\_FCR[7:6] RX\_FIFO\_TRIG** и **UARTi.UART\_FCR[5:4]** бит **TX\_FIFO\_TRIG** соответственно). Сигналы прерывания предписывают **MPU** передать данные в пункт назначения (от **UART** в режиме приема).

и/или из любого источника в **UART FIFO** в режиме передачи). Когда управление потоком **UART** включено с возможностью прерывания, порог **FIFO** управления потоком **UART** (значение битового поля **UARTi.UART\_TCR[3:0] RX\_FIFO\_TRIG\_HALT**) должно быть больше или равно порогу **RX FIFO** .

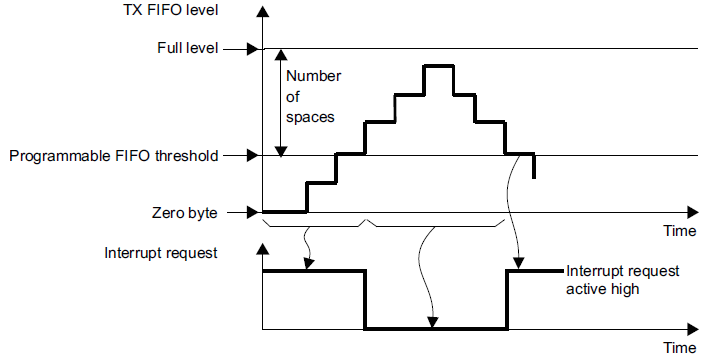
На рис. 19-5 показано генерирование запроса прерывания **RX FIFO**.

**Рис. 19-5. Генерация запроса прерывания FIFO RX**



В режиме приема прерывание не генерируется до тех пор, пока **RX FIFO** не достигнет своего порога. После низкого уровня прерывание может быть снято только тогда, когда **MPU** обработает достаточно байтов, чтобы поместить уровень **FIFO** ниже порогового значения. Порог управления потоком устанавливается на более высокое значение, чем порог **FIFO**.

На рис. 19-6 показано генерирование запроса прерывания **TX FIFO**.

**Рис. 19-6. Генерация запроса прерывания TX FIFO** 

В режиме передачи запрос на прерывание автоматически утверждается, когда TX FIFO пуст. Этот запрос сбрасывается, когда TX FIFO пересекает пороговый уровень. Линия прерывания снимается до тех пор, пока передается достаточное количество элементов, чтобы опуститься ниже порога TX FIFO.

**19.3.6.3 Работа в режиме опроса FIFO**

В режиме опроса **FIFO** (бит **UARTi.UART\_FCR[0] FIFO\_EN** установлен в **0** и соответствующие прерывания отключен регистром **UARTi.UART\_IER**), состояние приемника и передатчика можно проверить с помощью опроса регистра состояния линии (**UARTi.UART\_LSR**). Этот режим является альтернативой режиму прерывания **FIFO**, в котором состояние приемника и передатчик определяется автоматически путем отправки прерываний на **MPU**.

**19.3.6.4 Работа в режиме FIFO с DMA**

На этом устройстве поддерживаются только режимы **DMA** **0** и **1**.

- Когда бит **UARTi.UART\_SCR[0] DMA\_MODE\_CTL** установлен в **0**, установка в регистре **UARTi.UART\_FCR[3]** бита **DMA\_MODE** в **0** включает режим **DMA** **0**. Установка бита **DMA\_MODE** в **1** включает режим **DMA** **1**.

- Когда бит **DMA\_MODE\_CTL** установлен в **1**, битовое поле **UARTi.UART\_SCR[2:1]DMA\_MODE\_2** определяет режим **DMA** от **0** до режима **3** на основе  описания дополнительного регистра управления (**UART\_SCR**) .

Например:

- Если операция **DMA** не требуется, установите бит **DMA\_MODE\_CTL** в **1**, а битовое поле **DMA\_MODE\_2** в **0x0**. (Бит **DMA\_MODE** отбрасывается.)

- Если требуется режим **DMA** **1**, установите бит **DMA\_MODE\_CTL** в **0**, а бит **DMA\_MODE** в **1** или установите бит **DMA\_MODE\_CTL** в **1**, а битовое поле **DMA\_MODE\_2** — **0x01**. (Бит **DMA\_MODE** отбрасывается.)

Если **FIFO** отключены (бит **UARTi.UART\_FCR[0] FIFO\_EN** установлен в **0**), **DMA** происходит в режиме однократной отправки символов.

Когда запрограммирован режим **0** прямого доступа к памяти, сигналы, связанные с операцией прямого доступа к памяти, не активны. В зависимости от **UART\_MDR3[2]** **SET\_DMA\_TX\_THRESHOLD** порог может быть запрограммирован разными способами:

- **SET\_TX\_DMA\_THRESHOLD** **= 1**:

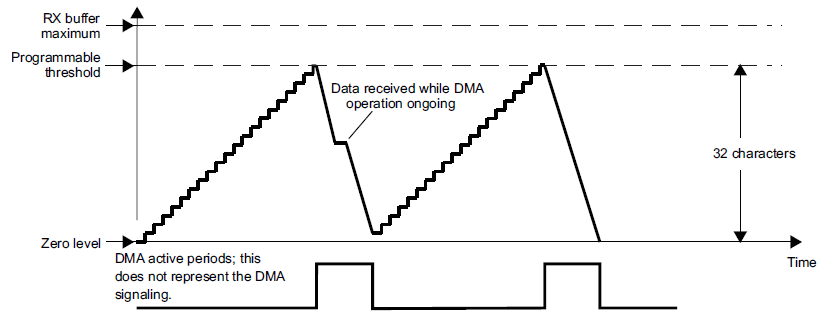
Пороговым значением будет значение регистра **UART\_TX\_DMA\_THRESHOLD**. Если **SET\_TX\_DMA\_THRESHOLD** + 64 **TX**  триггерных пробела, тогда используется метод порога по умолчанию: пороговое значение = размер **TX** **FIFO**.

- **SET\_TX\_DMA\_THRESHOLD = 0**:

Пороговое значение = размер **TX** **FIFO** - количество триггерных пробелов **TX**. Линия **TX** **DMA** устанавливается, если уровень **TX** **FIFO** ниже порога. Он остается активным до тех пор, пока триггер **TX** не пропустит количество байтов записываемых в **FIFO**. Затем линия **DMA** сбрасывается, и уровень **FIFO** сравнивается с пороговым значением.

**19.3.6.4.1 Передачи DMA (режим DMA 1, 2 или 3)**

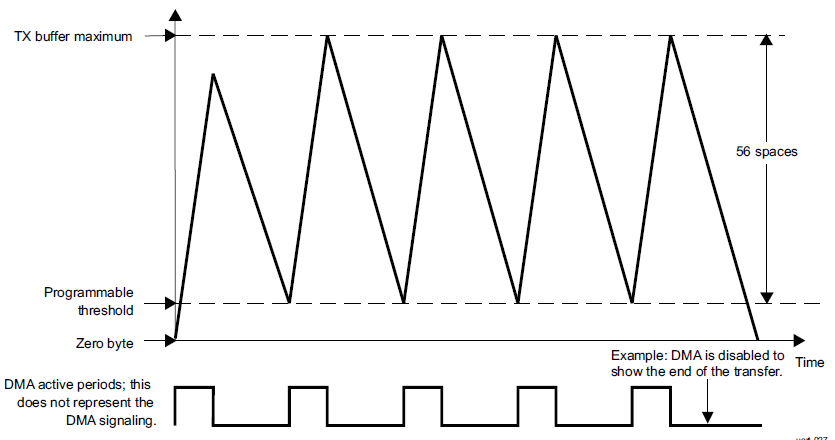
Рисунок 19-7 — Рисунок 19-10 показывают поддерживаемые операции прямого доступа к памяти.

**Рис. 19-7. Генерация запроса FIFO DMA (32 символа)** 

В режиме приема запрос **DMA** генерируется, когда **RX** **FIFO** достигает своего порогового уровня, определенного в регистре уровня триггера (**UARTi.UART\_TLR**). Этот запрос отменяется, когда число байтов, определяемое пороговым уровнем будет полностью отправлено с помощью **EDMA**.

В режиме передачи запрос **DMA** автоматически назначается, когда опустошается **TX** **FIFO** . Этот запрос сбрасывается, когда количество байтов определенное количеством пробелов в регистре **UARTi.UART\_TLR** записывается с помощью **EDMA** в буфер. Если в буфер записано недостаточное количество символов, запрос **DMA** остается активным.

**Рис. 19-8. Генерация запроса передачи FIFO DMA (56 пробелов)**



Запрос **DMA** снова утверждается, если **FIFO** может получить количество байтов, определенное в регистр **UARTi.UART\_TLR**.

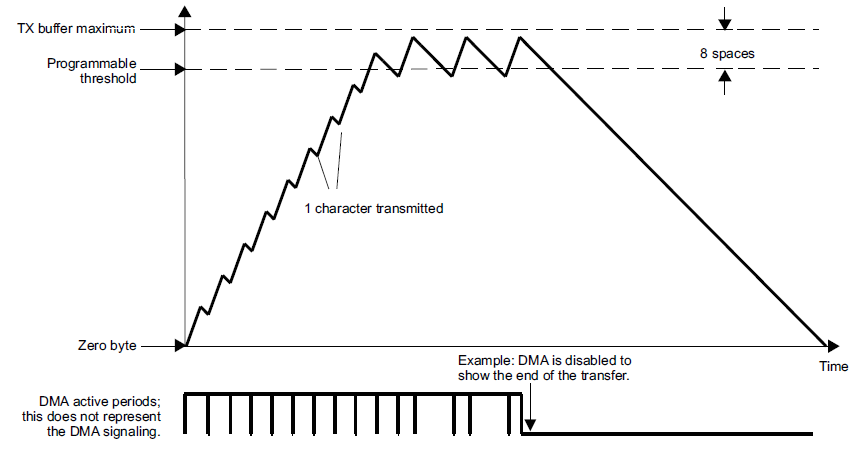
Порог можно запрограммировать несколькими способами. На рис. 19-8 показана передача **DMA**, работающая с

установкой 56 пробелов, которая может возникнуть из-за использования автоматических настроек вбитовом поле **UARTi.UART\_FCR[5:4]**

**TX\_FIFO\_TRIG** или битовое поле **UARTi.UART\_TLR[3:0] TX\_FIFO\_TRIG\_DMA**, объединенное с битовым полем **TX\_FIFO\_TRIG**.

Настройка 56 пробелов в модуле **UART/IrDA/CIR** должна коррелировать с настройками **EDMA**, потомучто буфер не переполняется (запрограммируйте размер запроса **DMA** контроллера **EDMA**, чтобы он равнялся количество пробелов в модуле **UART/IrDA/CIR**). На рис. 19-9 показан пример с восемью пробелами, показывающий, как уровень буфера пересекает порог пробела. Настройки контроллера **EDMA** должны соответствовать настройкам модуля **UART/IrDA/CIR**.

**Рис. 19-9. Генерация запроса передачи FIFO DMA (8 пробелов)**

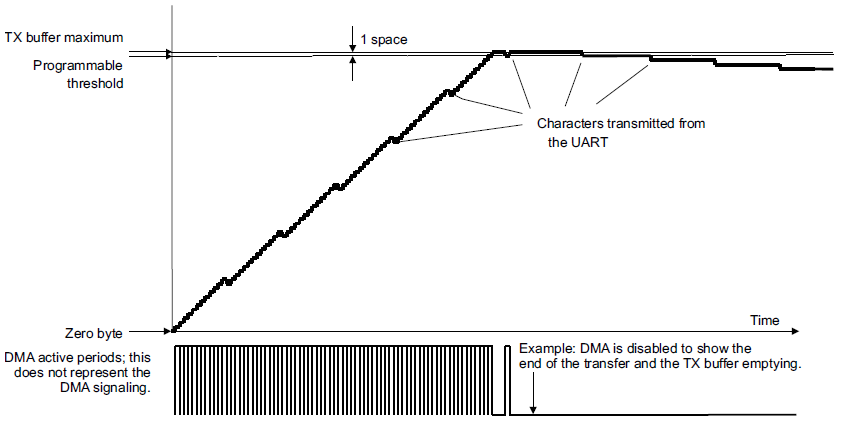


В следующем примере показана настройка одного пробела, использующая **DMA** для каждой передачи одного символа в буфер передачи (см. рис. 19-10). Буфер заполняется быстрее, чем  данные передаются на вывод **TX**. В конце концов, буфер полностью заполняется, и операции **DMA** прекращают передачу данных в буфер передачи.

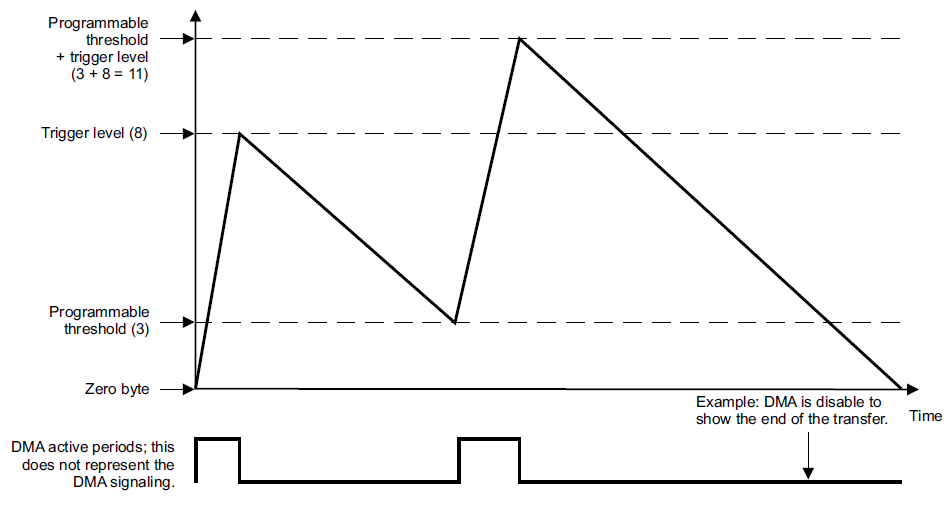
В двух случаях буфер содержит максимальное количество слов данных; вскоре после этого DMA отключается, чтобы показать более медленную передачу слов данных на вывод TX. В конце концов, буфер опустошается со скоростью, заданной настройками скорости передачи в регистрах **UARTi.UART\_DLL** и **UARTi.UART\_DLH**.

Настройки **DMA** должны соответствовать настройкам системного контроллера **EDMA**, чтобы обеспечить правильную работу этой логики.

**Рис. 19-10. Генерация запроса FIFO DMA передачи (1 пробел)**

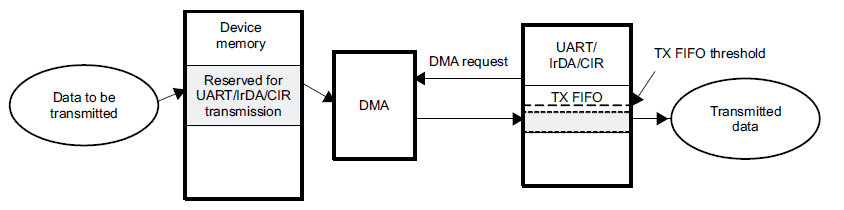


В последнем примере показана установка восьми пробелов, но через задание порога **TX DMA** напрямую путем установки бита **UART\_MDR3[1]SET\_DMA\_RX\_THRESHOLD** в регистре **UART\_TX\_DMA\_THRESHOLD** (см. рис. 19-11). В примере порог опустошения буфера **UART\_TX\_DMA\_THRESHOLD[2:0]TX\_DMA\_THRESHOLD** равен **3** а уровень триггера заполнения буфера равен **8**. Буфер заполняется быстрее, чем скорость передачи данных на вывод **TX**. Буфер заполняется **8** байтами, и операции DMA прекращают передачу данных в буфер передачи. Когда буфер опустошается до порогового уровня при передаче, операция **DMA** снова активируется чтобы заполнить буфер **8** байтами. В конце концов, буфер окончательно опустошается со скоростью, заданной настройками скорости передачи в регистрах **UART\_DLL**  **UART\_DLH**. Если выбранный пороговый уровень плюс уровень срабатывания превышают максимальный размер буфера, исходный пороговый метод  **TX DMA** используется для предотвращения переполнения **TX**, независимо от значения бита **UART\_MDR3[1]SET\_DMA\_RX\_THRESHOLD**. Настройки **DMA** должны соответствовать настройкам локального хост-контроллера **DMA** системы, чтобы обеспечить корректную работу этой логики.

**Рисунок 19-11. Генерация запроса FIFO DMA передачи с использованием прямого программирования порога DMA TX. (Порог = 3; Пробелы = 8)**

**19.3.6.4.2 Передача DMA**

На рис. 19-12 показана передача DMA.

**Рис. 19-12. Передача DMA** 

1. Данные для передачи помещаются в память устройства, зарезервированную для  **UART/IrDA/CIR** для передачи по **DMA**:

а. Пока не будет достигнут уровень срабатывания **TX FIFO**, генерируется запрос **DMA**.

б. Элемент (1 байт) передается из **SDRAM** в **TX FIFO** при каждом запросе **DMA** (происходит синхронизация элементов с DMA  ).

2. Данные в **TX FIFO** передаются автоматически.

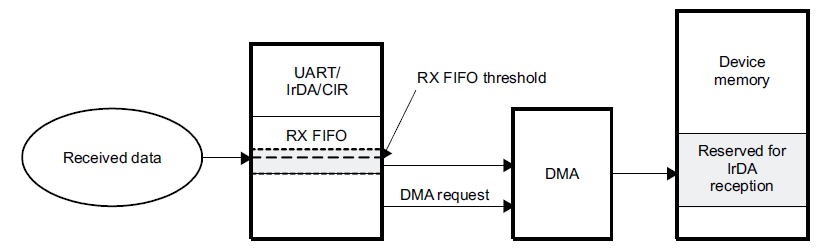
3. Окончание передачи сигнализируется пустым **UARTi.UART\_THR** (**TX** **FIFO** пуст).

***ПРИМЕЧАНИЕ. В режиме IrDA передача не заканчивается сразу после опустошения буфера TX FIFO, потому что так же  должны быть переданы последний байт данных, поле CRC и стоп-флаг; таким образом, конец передачи происходит через несколько миллисекунд после опустошения регистра UARTi.UART\_THR .***

**19.3.6.4.3 Прием DMA**

На рис. 19-13 показан прием DMA.

**Рис. 19-13. Прием DMA**



1. Включить прием.

2. Полученные данные помещаются в **RX** **FIFO**.

3. Данные передаются из **RX** **FIFO** в память устройства по **DMA**:

а. С каждым принятым байтом достигается срабатывание триггера уровеня  **RX FIFO** (один символ) и генерируется запрос DMA .

б. Элемент (1 байт) передается из **RX FIFO** в **SDRAM** при каждом запросе **DMA** (синхронизация элементов  с **DMA**).

4. Окончание приема сигнализируется прерыванием **EOF**.

**19.3.7 Выбор режима**

**19.3.7.1 Режимы доступа к регистру**

**19.3.7.1.1 Рабочий режим и режимы конфигурации**

Доступ к регистру зависит от режима доступа к регистру, хотя режимы доступа к регистру не коррелируют с  функциональным режимом. Доступны три разных режима:

- Режим работы

- Режим конфигурации А

- Режим конфигурации B

Рабочий режим — это выбранный режим, когда функция активна; последовательная передача данных может быть выполнена в этом режиме.

Режим конфигурации A и режим конфигурации B используются на этапах инициализации модуля. Эти режимы разрешают доступ к конфигурационным регистрам, скрытым в рабочем режиме. Режимы используются когда модуль неактивен (последовательная передача данных не обрабатывается) и нужны только для нициализации или реконфигурации модуля. Значение регистра **UARTi.UART\_LCR** определяет режим доступа к регистру (см. Таблицу 19-16).

**Таблица 19-16. Программирование режима доступа к регистру UART/IrDA/CIR (с использованием UART\_LCR)**

|  |  |
| --- | --- |
| Mode | Condition |
| Configuration mode A | UART\_LCR[7] = 0x1 and UART\_LCR[7:0] != 0xBF |
| Configuration mode B | UART\_LCR[7] = 0x1 and UART\_LCR[7:0] = 0xBF |
| Operational mode | UART\_LCR[7] = 0x0 |

**19.3.7.1.2 Подрежим доступа к регистру**

В каждом режиме доступа к регистру (рабочий режим или режим конфигурации A/B) некоторые обращения к регистру зависят  от программирования подрежима (**MSR\_SPR, TCR\_TLR** и **XOFF**).

Таблицы с 19-17 по 19-19 обобщают подрежимы доступа к регистрам.

**Таблица 19-17. Сводка режима подконфигурации A**

|  |  |
| --- | --- |
| Mode | Condition |
| MSR\_SPR | (UART\_EFR[4] = 0x0 or UART\_MCR[6] = 0x0) |
| TCR\_TLR | UART\_EFR[4] = 0x1 and UART\_MCR[6] = 0x1 |

**Таблица 19-18. Сводная информация о подконфигурации режима B**

|  |  |
| --- | --- |
| Mode | Condition |
| TCR\_TLR | UART\_EFR[4] = 0x1 and UART\_MCR[6] = 0x1 |
| XOFF | (UART\_EFR[4] = 0x0 or UART\_MCR[6] = 0x0) |

**Таблица 19-19. Сводная информация о подконфигурационном режиме**

|  |  |
| --- | --- |
| Mode | Condition |
| MSR\_SPR | UART\_EFR[4] = 0x0 or UART\_MCR[6] = 0x0 |
| TCR\_TLR | UART\_EFR[4] = 0x1 and UART\_MCR[6] = 0x1 |

**19.3.7.1.3 Регистры, доступные для режимов доступа к регистру**

В таблице 19-20 перечислены имена битов регистра в каждом режиме доступа к регистру. Серая заливка указывает на то, что регистр не зависит от режима доступа  (доступен во всех режимах).

**Таблица 19-20. Обзор режима доступа к регистру UART/IrDA/CIR**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Address  Offset |  |  | Registers |  |  |  |
|  | **Configuration Mode A** | | **Configuration Mode B** | | **Operational Mode** | |
|  | **Read** | **Write** | **Read** | **Write** | **Read** | **Write** |
| 0x000 | UART\_DLL | UART\_DLL | UART\_DLL | UART\_DLL | UART\_RHR | UART\_THR |
| 0x004 | UART\_DLH | UART\_DLH | UART\_DLH | UART\_DLH | UART\_IER | UART\_IER |
| 0x008 | UART\_IIR | UART\_FCR | UART\_EFR | UART\_EFR | UART\_IIR | UART\_FCR |
| 0x00C | UART\_LCR | UART\_LCR | UART\_LCR | UART\_LCR | UART\_LCR | UART\_LCR |
| 0x010 | UART\_MCR | UART\_MCR | UART\_XON1\_ADD  R1 | UART\_XON1\_AD  DR1 | UART\_MCR | UART\_MCR |
| 0x014 | UART\_LSR | - | UART\_XON2\_ADD  R2 | UART\_XON2\_AD  DR2 | UART\_LSR | - |
| 0x018 | UART\_MSR  (1)/UART\_TCR  (2) | UART\_TCR (2) | UART\_TCR  (2)/UART\_XOFF1 (3) | UART\_TCR  (2)/UART\_XOFF1  (3) | UART\_MSR  (1)/UART\_TCR (2) | UART\_TCR (2) |
| 0x01C | UART\_SPR  (1)/UART\_TLR  (2) | UART\_SPR  (1)/UART\_TLR (2) | UART\_TLR  (2)/UART\_XOFF2 (3) | UART\_TLR  (2)/UART\_XOFF2  (3) | UART\_SPR  (1)/UART\_TLR (2) | UART\_SPR  (1)/UART\_TLR  (2) |
| 0x020 | UART\_MDR1 | UART\_MDR1 | UART\_MDR1 | UART\_MDR1 | UART\_MDR1 | UART\_MDR1 |
| 0x024 | UART\_MDR2 | UART\_MDR2 | UART\_MDR2 | UART\_MDR2 | UART\_MDR2 | UART\_MDR2 |
| 0x028 | UART\_SFLSR | UART\_TXFLL | UART\_SFLSR | UART\_TXFLL | UART\_SFLSR | UART\_TXFLL |

(1) Режим MSR\_SPR включен (см. раздел 19.3.7.1.2, Подрежим доступа регистра)

(2) Режим TCR\_TLR включен (см. раздел 19.3.7.1.2, Подрежим доступа регистра)

(3) Включен режим XOFF (см. раздел 19.3.7.1.2, Подрежим доступа регистра)

**Таблица 19-20. Обзор режима доступа к регистру UART/IrDA/CIR (продолжение)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Address  Offset | Registers | | | | | |
|  | **Configuration Mode A** | | **Configuration Mode B** | | **Operational Mode** | |
|  | **Read** | **Write** | **Read** | **Write** | **Read** | **Write** |
| 0x02C | UART\_RESUM  E | UART\_TXFLH | UART\_RESUME | UART\_TXFLH | UART\_RESUME | UART\_TXFLH |
| 0x030 | UART\_SFREG  L | UART\_RXFLL | UART\_SFREGL | UART\_RXFLL | UART\_SFREGL | UART\_RXFLL |
| 0x034 | UART\_SFREG  H | UART\_RXFLH | UART\_SFREGH | UART\_RXFLH | UART\_SFREGH | UART\_RXFLH |
| 0x038 | UART\_UASR | - | UART\_UASR | - | UART\_BLR | UART\_BLR |
| 0x03C | - | - | - | - | UART\_ACREG | UART\_ACREG |
| 0x040 | UART\_SCR | UART\_SCR | UART\_SCR | UART\_SCR | UART\_SCR | UART\_SCR |
| 0x044 | UART\_SSR | - | UART\_SSR | - | UART\_SSR | - |
| 0x048 | - | - | - | - | UART\_EBLR | UART\_EBLR |
| 0x050 | UART\_MVR | - | UART\_MVR | - | UART\_MVR | - |
| 0x054 | UART\_SYSC | UART\_SYSC | UART\_SYSC | UART\_SYSC | UART\_SYSC | UART\_SYSC |
| 0x058 | UART\_SYSS | - | UART\_SYSS | - | UART\_SYSS | - |
| 0x05C | UART\_WER | UART\_WER | UART\_WER | UART\_WER | UART\_WER | UART\_WER |
| 0x060 | UART\_CFPS | UART\_CFPS | UART\_CFPS | UART\_CFPS | UART\_CFPS | UART\_CFPS |
| 0x064 | UART\_RXFIFO  \_LVL | UART\_RXFIFO  \_LVL | UART\_RXFIFO  \_LVL | UART\_RXFIFO  \_LVL | UART\_RXFIFO  \_LVL | UART\_RXFIFO  \_LVL |
| 0x068 | UART\_TXFIFO  \_LVL | UART\_TXFIFO  \_LVL | UART\_TXFIFO  \_LVL | UART\_TXFIFO  \_LVL | UART\_TXFIFO  \_LVL | UART\_TXFIFO  \_LVL |
| 0x06C | UART\_IER2 | UART\_IER2 | UART\_IER2 | UART\_IER2 | UART\_IER2 | UART\_IER2 |
| 0x070 | UART\_ISR2 | UART\_ISR2 | UART\_ISR2 | UART\_ISR2 | UART\_ISR2 | UART\_ISR2 |
| 0x074 | UART\_FREQ\_  SEL | UART\_FREQ\_  SEL | UART\_FREQ\_  SEL | UART\_FREQ\_  SEL | UART\_FREQ\_  SEL | UART\_FREQ\_  SEL |
| 0x080 | UART\_MDR3 | UART\_MDR3 | UART\_MDR3 | UART\_MDR3 | UART\_MDR3 | UART\_MDR3 |
| 0x084 | UART\_TX\_DM  A\_THRESHOL  D | UART\_TX\_DM  A\_THRESHOL  D | UART\_TX\_DM  A\_THRESHOL  D | UART\_TX\_DM  A\_THRESHOL  D | UART\_TX\_DM  A\_THRESHOL  D | UART\_TX\_DM  A\_THRESHOL  D |

**19.3.7.2 Выбор режима UART/IrDA (SIR, MIR, FIR)/CIR**

Для выбора режима установите битовое поле **UARTi.UART\_MDR1[2:0] MODESELECT** (см. Таблицу 19-21).

**Таблица 19-20. Обзор режима доступа к регистру UART/IrDA/CIR (продолжение)**

|  |  |
| --- | --- |
| Value | Mode |
| 0x0 | UART 16x mode |
| 0x1 | SIR mode |
| 0x2 | UART 16x auto-baud |
| 0x3 | UART 13x mode |
| 0x4 | MIR mode |
| 0x5 | FIR mode |
| 0x6 | CIR mode |

**MODESELECT** действует, когда модуль находится в рабочем режиме (см. Раздел 19.3.7.1, Регистрация режимов доступа).

**19.3.7.2.1 Регистры, доступные для функции UART**

Для функции **UART** используются только регистры, перечисленные в таблице 19-22.

**Таблица 19-22. Обзор регистра режима UART (1) (2)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Address  Offset | Registers | | | | | |
|  | **Configuration Mode A** | | **Configuration Mode B** | | **Operational Mode** | |
|  | **Read** | **Write** | **Read** | **Write** | **Read** | **Write** |
| 0x00 | UART\_DLL | UART\_DLL | UART\_DLL | UART\_DLL | UART\_RHR | UART\_THR |
| 0x04 | UART\_DLH | UART\_DLH | UART\_DLH | UART\_DLH | UART\_IER | UART\_IER |
| 0x08 | UART\_IIR | UART\_FCR | UART\_EFR | UART\_EFR | UART\_IIR | UART\_FCR |
| 0x0C | UART\_LCR | UART\_LCR | UART\_LCR | UART\_LCR | UART\_LCR | UART\_LCR |
| 0x10 | UART\_MCR | UART\_MCR | UART\_XON1\_ADD  R1 | UART\_XON1\_AD  DR1 | UART\_MCR | UART\_MCR |
| 0x14 | UART\_LSR | - | UART\_XON2\_ADDR2 | UART\_XON2\_ADDR2 | UART\_LSR | - |
| 0x18 | UART\_MSR  (1)/UART\_TCR  (2) | UART\_TCR (2) | UART\_TCR  (2)/UART\_XOFF1 (3) | UART\_TCR  (2)/UART\_XOFF1  (3) | UART\_MSR  (1)/UART\_TCR (2) | UART\_TCR (2) |
| 0x1C | UART\_SPR  (1)/UART\_TLR  (2) | UART\_SPR  (1)/UART\_TLR (2) | UART\_TLR  (2)/UART\_XOFF2 (3) | UART\_TLR  (2)/UART\_XOFF2  (3) | UART\_SPR  (1)/UART\_TLR (2) | UART\_SPR  (1)/UART\_TLR  (2) |
| 0x20 | UART\_MDR1 | UART\_MDR1 | UART\_MDR1 | UART\_MDR1 | UART\_MDR1 | UART\_MDR1 |
| 0x24 | UART\_MDR2 | UART\_MDR2 | UART\_MDR2 | UART\_MDR2 | UART\_MDR2 | UART\_MDR2 |
| 0x28 | - | - | - | - | - | - |
| 0x2C | - | - | - | - | - | - |
| 0x30 | - | - | - | - | - | - |
| 0x34 | - | - | - | - | - | - |
| 0x38 | UART\_UASR | - | UART\_UASR | - | - | - |
| 0x3C | - | - | - | - | - | - |
| 0x40 | UART\_SCR | UART\_SCR | UART\_SCR | UART\_SCR | UART\_SCR | UART\_SCR |
| 0x44 | UART\_SSR | - | UART\_SSR | - | UART\_SSR | - |
| 0x48 | - | - | - | - | - | - |
| 0x50 | UART\_MVR | - | UART\_MVR | - | UART\_MVR | - |
| 0x54 | UART\_SYSC | UART\_SYSC | UART\_SYSC | UART\_SYSC | UART\_SYSC | UART\_SYSC |
| 0x58 | UART\_SYSS | - | UART\_SYSS | - | UART\_SYSS | - |
| 0x5C | UART\_WER | UART\_WER | UART\_WER | UART\_WER | UART\_WER | UART\_WER |
| 0x60 | - | - | - | - | - | - |
| 0x64 | UART\_RXFIFO  \_LVL | UART\_RXFIFO\_LVL | UART\_RXFIFO  \_LVL | UART\_RXFIFO  \_LVL | UART\_RXFIFO  \_LVL | UART\_RXFIFO  \_LVL |
| 0x68 | UART\_TXFIFO  \_LVL | UART\_TXFIFO  \_LVL | UART\_TXFIFO  \_LVL | UART\_TXFIFO  \_LVL | UART\_TXFIFO  \_LVL | UART\_TXFIFO  \_LVL |
| 0x6C | UART\_IER2 | UART\_IER2 | UART\_IER2 | UART\_IER2 | UART\_IER2 | UART\_IER2 |
| 0x70 | UART\_ISR2 | UART\_ISR2 | UART\_ISR2 | UART\_ISR2 | UART\_ISR2 | UART\_ISR2 |
| 0x74 | UART\_FREQ\_  SEL | UART\_FREQ\_  SEL | UART\_FREQ\_  SEL | UART\_FREQ\_  SEL | UART\_FREQ\_  SEL | UART\_FREQ\_  SEL |
| 0x80 | UART\_MDR3 | UART\_MDR3 | UART\_MDR3 | UART\_MDR3 | UART\_MDR3 | UART\_MDR3 |
| 0x84 | UART\_TX DMA\_THRESHOLD | UART\_TX DMA\_THRESHOLD | UART\_TX DMA\_THRESHOLD D | UART\_TX DMA\_THRESHOLD | UART\_TX DMA\_THRESHOLD | UART\_TX DMA\_THRESHOLD |

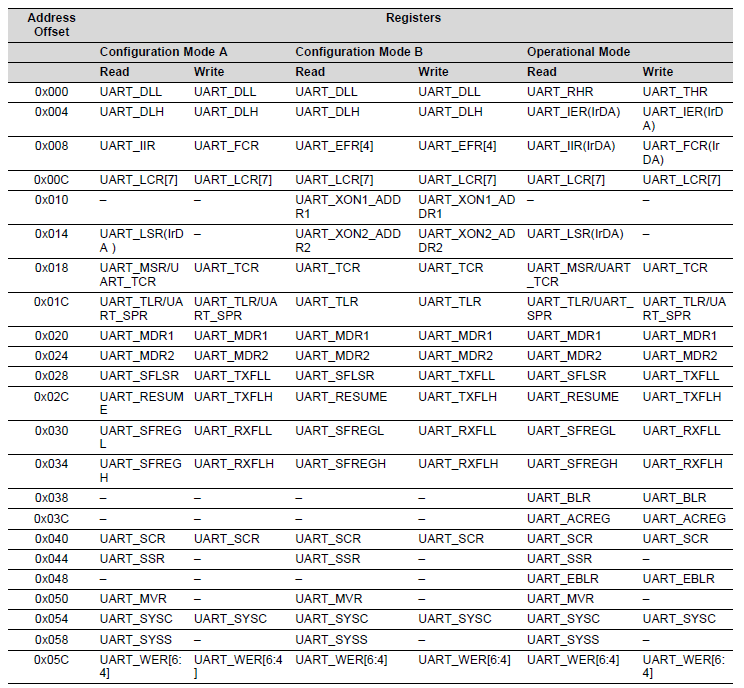
(1) Нотация REGISTER\_NAME (UART) указывает, что регистр существует для других функций (IrDA или CIR), но поля имеют разные

значения для других функций.

(2) REGISTER\_NAME[m:n] обозначает, что к функции UART применяются только биты регистров, пронумерованные от m до n.

**19.3.7.2.2 Регистры, доступные для функции IrDA**

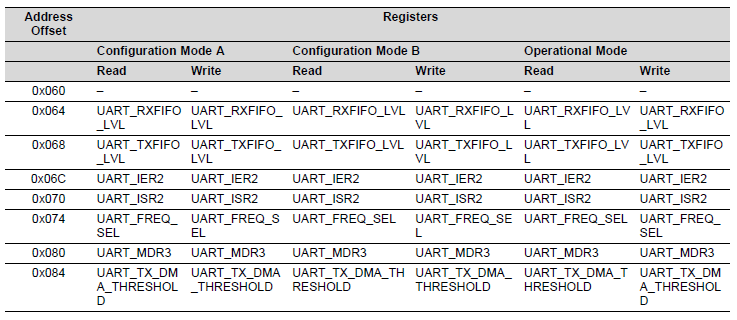
Для функции IrDA используются только регистры, перечисленные в таблице 19-23.

**Таблица 19-23. Обзор регистра режима IrDA (1) (2)**

(1) Нотация REGISTER\_NAME (UART) указывает, что регистр существует для других функций (IrDA или CIR), но поля имеют разные значения для других функций.

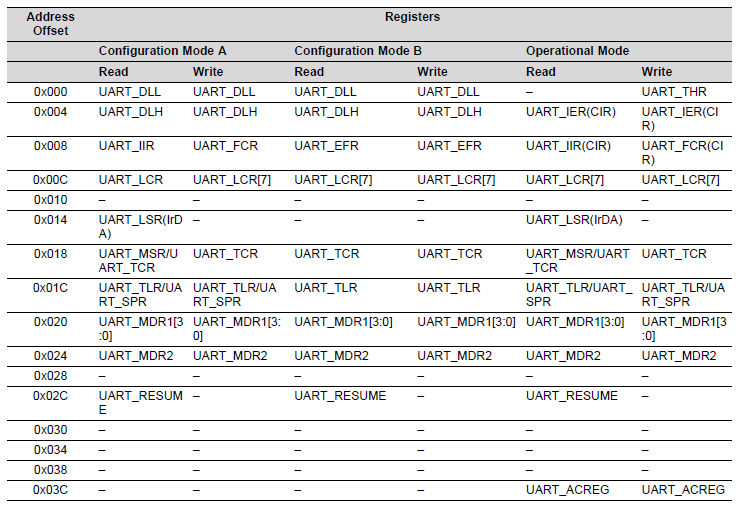
(2) REGISTER\_NAME[m:n] обозначает, что к функции UART применяются только биты регистров, пронумерованные от m до n.

**Таблица 19-23. Обзор регистра режимов IrDA (1) (2) (продолжение)**



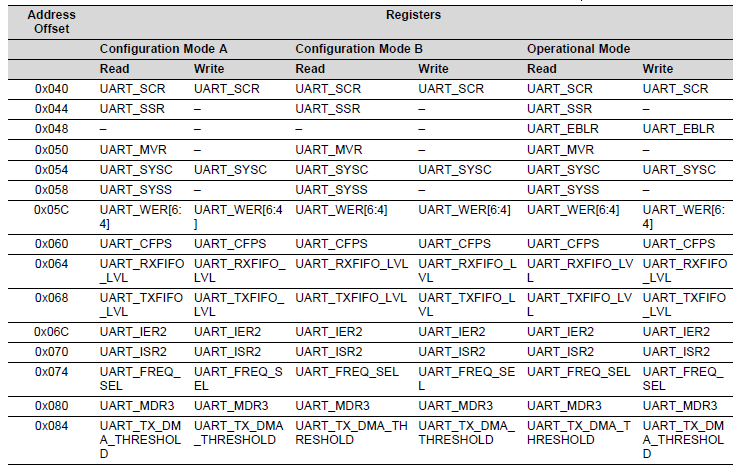
**19.3.7.2.3 Регистры, доступные для функции CIR**

Для функции CIR используются только регистры, перечисленные в таблице 19-24.

**Таблица 19-24. Обзор регистра режимов CIR (1) (2)**

(1) Нотация REGISTER\_NAME (UART) указывает, что регистр существует для других функций (IrDA или CIR), но поля имеют разные значения для других функций.

(2) REGISTER\_NAME[m:n] обозначает, что к функции UART применяются только биты регистров, пронумерованные от m до n.

**Таблица 19-24. Обзор регистра режимов CIR (1) (2) (продолжение)**

**19.3.8 Форматирование протокола**

Модуль **UART/IRDA** может работать в семи различных режимах:

1. Режим UART 16x (≤230,4 Кбит/с), UART16x ≤460 Кбит/с, если установлен MDR3[1]

2. Режим UART 16x с автопередачей (≥1200 бит/с и ≤115,2 Кбит/с), если не установлен MDR3[1]

3. Режим UART 13x (≥460,8 Кбит/с), если не установлен MDR3[1]

4. Режим IrDA SIR (≤115,2 Кбит/с), если не установлен MDR3[1]

5. Режим IrDA MIR (0,576 и 1,152 Мбит/с), если не установлен MDR3[1]

6. Режим IrDA FIR (4 Мбит/с), если не установлен MDR3[1]

7. Режим CIR (программируемая частота модуляции для приложений дистанционного управления), если MDR3[1] не установлен

Модуль выполняет последовательно-параллельное преобразование полученных символов данных и параллельно-последовательное преобразование.

преобразование передаваемых символов данных процессором. Полный статус каждого канала модуль и каждый принятый символ/кадр могут быть прочитаны в любое время во время функциональной работы через линию регистр состояния (**LSR**).

Модуль можно перевести в альтернативный режим (режим **FIFO**), чтобы избавить процессор от избыточной нагрузки накладными расходами программного обеспечения за счет буферизации получаемых/отправляемых символов.

**FIFO** приемника и передатчика могут хранить до 64 байтов данных (плюс три дополнительных бита ошибки состояния на каждый байт **RX FIFO**) и имеют выбираемые уровни запуска. И прерывания, и DMA режим могут управлять потоком данных  между LH и модулем.

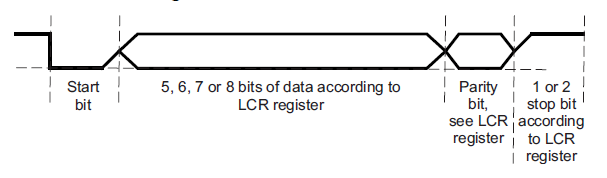
**19.3.8.1 Режим UART**

**UART** использует проводной интерфейс для последовательной связи с удаленным устройством.

Модуль **UART** функционально совместим с **UART TL16C750**, а также функционально совместим к более ранним конструкциям, таким как **TL16C550**. Модуль UART может использовать аппаратное или программное управление потоком для управления передачей и приемом. Аппаратное управление потоком данных значительно снижает накладные расходы на программное обеспечение и повышает эффективность системы за счет автоматического управления последовательным потоком данных с использованием входов RTS и CTS  . Программное управление потоком данных автоматически управляет потоком данных с помощью программируемых символов **XON/XOFF**.

Модуль модема **UART** дополнен функцией автоматической настройки скорости, которая в режиме управления позволяет автоматически устанавливается скорость, количество бит на символ, выбранную четность.

**Рис. 19-14. Формат данных UART**

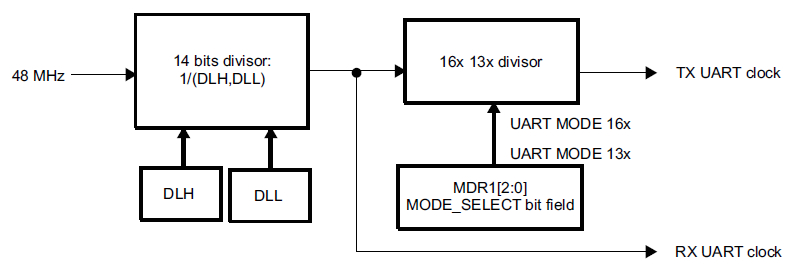


**19.3.8.1.1 Генерация тактирования UART: генерация скорости передачи**

Функция UART содержит программируемый генератор бод и набор фиксированных делителей, которые делят 48-МГц тактовый сигнал до ожидаемой скорости передачи в бодах.

На рис. 19-15 показан генератор скорости передачи в бодах и соответствующие средства управления.

**Рис. 19-15. Формирование скорости передачи в бодах**



*Внимание!!!*

*Перед инициализацией или изменением элементов управления параметрами тактирования (UARTi.UART\_DLH,UARTi.UART\_DLL), MODE\_SELECT = DISABLE (UARTi.UART\_MDR1[2:0])должен быть установлен в 0x7. Несоблюдение этого правила может привести к непредсказуемому поведению модуля.*

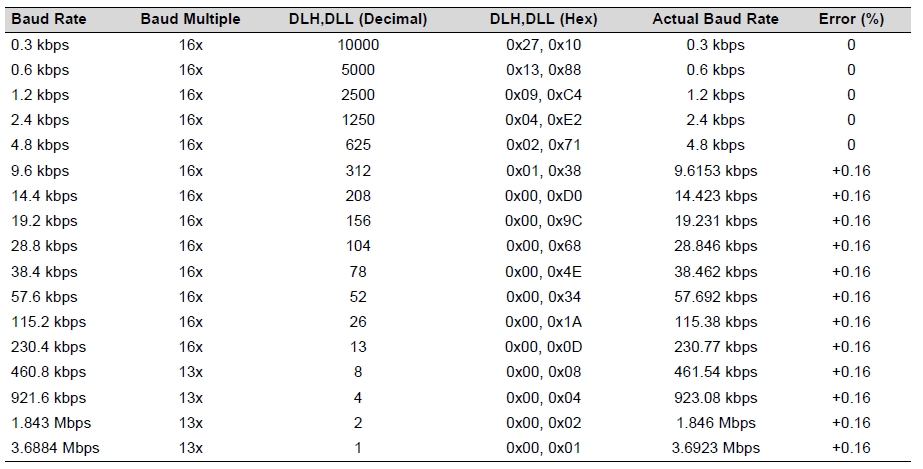
**19.3.8.1.2 Выбор подходящего значения делителя**

Два значения делителя:

- Режим UART 16x: значение делителя = рабочая частота/(скорость передачи 16x)

- Режим UART 13x: значение делителя = рабочая частота/(13-кратная скорость передачи данных)

Таблица 19-25 описывает настройки скорости передачи данных UART.

**Таблица 19-25. Настройки скорости передачи данных UART (тактовый генератор 48-MHz)** 

**19.3.8.1.3 Форматирование данных UART**

**UART** может использовать аппаратное управление потоком для управления передачей и приемом. Аппаратное управление потоком значительно снижает нагрузку на программное обеспечение и повышает эффективность системы за счет автоматического управленияпоток данных с использованием выходных сигналов **RTS** и  **CTS**.

**UART** дополнен функцией автоматической настройки скорости. В режиме управления автопереключение позволяет изменять скорость, количество битов на символ, а выбранная четность устанавливается автоматически.

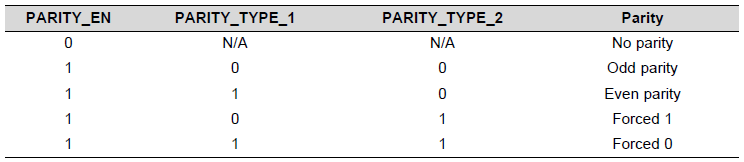
**19.3.8.1.3.1 Форматирование кадра**

Когда автободинг не используется, атрибуты формата кадра должны быть определены в регистре **UARTi.UART\_LCR**. Длина символа указывается с помощью битового поля **UARTi.UART\_LCR[1:0] CHAR\_LENGTH**.

Количество стоп-битов указывается с помощью бита **UARTi.UART\_LCR[2] NB\_STOP**.

Бит четности программируется с использованием **UARTi.UART\_LCR[5:3] PARITY\_EN, PARITY\_TYPE\_1** и битового поля **PARITY\_TYPE\_2** (см. Таблицу 19-26).

**Таблица 19-26. Кодирование битов четности UART**



**19.3.8.1.3.2 Аппаратное управление потоком**

Аппаратное управление потоком состоит из auto-CTS и auto-RTS. Auto-CTS и auto-RTS могут быть включены и отключается независимо путем программирования UARTi.UART\_EFR[7:6] AUTO\_CTS\_EN и битового поля AUTO\_RTS\_EN соответственно. При использовании auto-CTS uarti\_cts должен быть активен, прежде чем модуль сможет передавать данные.Auto-RTS активирует выход uarti\_rts только тогда, когда в **RX FIFO** достаточно места для приема данных. Это деактивирует выход **uarti\_rts**, когда **RX FIFO** достаточно заполнен. Уровни срабатывания **HALT** и **RESTORE** в регистре **UARTi.UART\_TCR** определяют уровни, на которых активируется и деактивируется **uarti\_rts**.

Если включены auto-CTS и auto-RTS, передача данных не происходит, если **RX FIFO** имеет пустое  место. Таким образом, исключаются ошибки переполнения при аппаратном управлении потоком. Если **auto-CTS** и **auto-RTS**

не включены, ошибки переполнения возникают, если скорость передачи данных превышает задержку RX FIFO.

• Авто-RTS:

Управление потоком данных Auto-RTS происходит в блоке приемника. Уровни срабатывания **RX FIFO**, используемые в авто-RTS хранятся в регистре **UARTi.UART\_TCR**. **uarti\_rts** активен, если уровень **RX FIFO** ниже

уровня срабатывания **HALT** в битовом поле **UARTi.UART\_TCR[3:0] RX\_FIFO\_TRIG\_HALT**. Когда **RX FIFO** заполнен до уровеня срабатывания **HALT**, **uarti\_rts** деактивируется. Передающее устройство (например, другой **UART**) может отправить дополнительный байт после достижения уровня срабатывания **HALT**, поскольку он может не распознать снятие подтверждения **RTS** до тех пор, пока он не начнет посылать дополнительный байт. **uarti\_rts** автоматически переназначается, когда **RX FIFO** достигает уровня триггера **RESUME**,запрограмированного в битовом поле **UARTi.UART\_TCR[7:4] RX\_FIFO\_TRIG\_START**. Это переназначение просит отправляющее устройство возобновить передачу. В этом случае **uarti\_rts** является активным низким сигналом.

• Авто-CTS:

Схема передатчика проверяет **uarti\_cts** перед отправкой следующего байта данных. Когда **uarti\_cts** активен,

передатчик посылает следующий байт. Чтобы передатчик не отправил следующий байт, **uarti\_cts** должен

сбрасывается до середины последнего отправляемого в данный момент стоп-бита. Функция **auto-CTS** снижает количество прерываний в хост-системе. Когда автоматическое управление потоком **CTS** включено, изменения состояния **uarti\_cts** не должны вызывать прерывания хоста, поскольку устройство автоматически

управляет собственным передатчиком. Без автоматического CTS передатчик отправляет любые данные, присутствующие в **TX FIFO**, что может привести к ошибке переполнения приемника. В этом случае **uarti\_cts** срабатывает при активном-низком сигнале.

**19.3.8.1.3.3 Программное управление потоком данных**

Управление потоком программного обеспечения включается через регистр расширенных функций (**UARTi.UART\_EFR**) и регистр управления модемом (**UARTi.UART\_MCR**). Различные комбинации программного управления потоком могут быть включается путем установки различных комбинаций битового поля **UARTi.UART\_EFR[3:0]** (см. Таблицу 19-27).

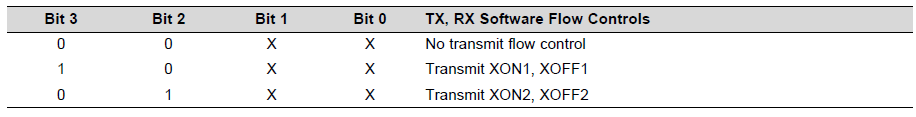
Две другие расширенные функции относятся к программному управлению потоком данных:

- **XON** любая функция **(UARTi.UART\_MCR[5]**): работа возобновляется после получения любого символа после того как символ XOFF распознается. Если обнаружение специального символа включено и получен специальный символ после **XOFF1** передача не возобновляется. Специальный символ хранится в **RX FIFO**.

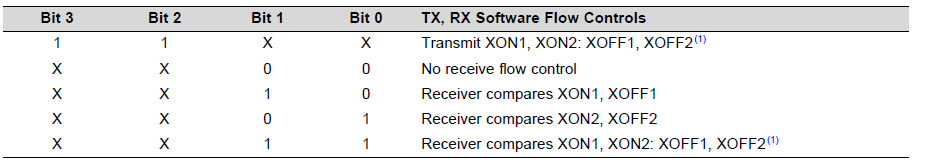
*ПРИМЕЧАНИЕ. Символ XON-any записывается в RX FIFO, даже если он является программным символом потока.*

- Специальный символ (**UARTi.UART\_EFR[5]**): входящие данные сравниваются с XOFF2. Когда спец символ обнаружен, устанавливается прерывание **XOFF** (**UARTi.UART\_IIR[4]**), но оно не останавливает передачу.

Прерывание XOFF очищается чтением **UARTi.UART\_IIR**. Специальный символ переносится в **RXFIFO**. Специальный символ не работает с **XON2**, **XOFF2** или последовательными **XOFF**.

**Таблица 19-27. UART\_EFR[3:0] Параметры управления потоком программного обеспечения**

**Таблица 19-27. UART\_EFR[3:0] Параметры управления потоком программного обеспечения (продолжение)**



(1) В этих случаях символы XON1 и XON2 или символы XOFF1 и XOFF2 должны передаваться/приниматься последовательно

с XON1/XOFF1, за которыми следуют XON2/XOFF2.XON1 определяется в поле UARTi.UART\_XON1\_ADDR1[7:0] XON\_WORD1 бит. XON2 определяется в UARTi.UART\_XON2\_ADDR2[7:0] XON\_WORD2 битовое поле. XOFF1 определяется в поле UARTi.UART\_XOFF1[7:0] XOFF\_WORD1 бит. XOFF2 определяется в UARTi.UART\_XOFF2[7:0] XOFF\_WORD2 битовое поле.

**19.3.8.1.3.3.1 Прием (RX)**

Когда операция программного управления потоком включена, UART сравнивает входящие данные с запрограммированными символами **XOFF1/2**.  (в некоторых случаях **XOFF1** и **XOFF2** должны приниматься последовательно). Когда принимаются правильный символ **XOFF**, передача прекращается после завершения передачи текущего символа. Обнаружение **XOFF** также устанавливает бит **UARTi.UART\_IIR[4]** (если разрешен **UARTi.UART\_IER[5]**) и вызывает низкий уровень на линии прерывания.

Для возобновления передачи должен быть принят символ **XON1/2** (в некоторых случаях должны быть последовательно приняты символы **XON1** и **XON2**). При получении правильных символов **XON** бит **UARTi.UART\_IIR[4]** очищается, и прерывание **XOFF** исчезает.

*ПРИМЕЧАНИЕ. При возникновении ошибки четности, кадрирования или прерывания при получении символа программного управления потоком , этот символ обрабатывается как обычные данные и записывается в RX FIFO.*

Когда обнаружение XON-любого и специального символа отключено, а программное управление потоком включено, недействительные символы **XON** или **XOFF** записываются в **RX FIFO**. Например, когда **UARTi.UART\_EFR[1:0] = 0x2**, если принимаются символы **XON1** и **XOFF1**, они не записываются в **RX FIFO**.

Когда пары символов программного потока запрограммированы на последовательный прием (**UARTi.UART\_EFR[1:0] = 0x3**), символы программного потока не записываются в **RX FIFO**, если они

получены последовательно. Однако принятые символы **XON1/XOFF1** должны быть записаны в **RX FIFO**, если

последующий символ не **XON2/XOFF2**.

**19.3.8.1.3.3.2 Передача (TX)**

Два символа **XOFF1** передаются, когда **RX FIFO** проходит уровень запуска, запрограммированный **UARTi.UART\_TCR[3:0]**. Как только **RX FIFO** достигает уровня срабатывания, запрограммированного в

**UARTi.UART\_TCR[7:4]**, отправляются два символа **XON1**, поэтому передача данных восстанавливается.

*ПРИМЕЧАНИЕ. Если программное управление потоком отключено после отправки символа XOFF, модуль передает*

*символы XON автоматически, чтобы включить нормальную передачу.*

Передача **XOFF(s)/XON(s)** следует тому же протоколу, что и передача обычного байта из **TX FIFO**. Это означает, что даже если длина слова составляет 5, 6 или 7 символов, 5, 6 или 7 младших битов **XOFF1/2** и **XON1/2** все равно передаются. 5, 6 или 7 бит символа передаются редко, но эта функция включена для обеспечения совместимости с более ранними разработками. Предполагается, что программное управление потоком и аппаратное управление потоком никогда не включаются одновременно.

**19.3.8.1.3.4 Режимы автоматической настройки**

В автоматическом режиме UART может извлекать характеристики передачи (скорость, длину и четность) из

AT комманд (код ASCII). Эти характеристики используются для приема данных после AT и для отправки данных.

Допустимы следующие AT-команды:

**AT     DATA <CR>**

**at     DATA <CR>**

**A/**

**a/**

Разрыв строки во время сбора последовательности AT не распознается, а функция echo не реализована аппаратно.**A/** и **a/** не используются для извлечения признаков, но их необходимо распознавать из-за их особого значения.  **A/** или **a/** используется для указания программному обеспечению повторить последнюю полученную AT-команду; следовательно, **а/** всегда следует за **A**T, и не ожидается, что передаточные характеристики изменятся между **AT** и **a/**. Когда получен действительный **AT**, **AT** и все последующие данные, включая окончательный <CR> **(0x0D)**, сохраняются в **RXFIFO**. Конечный автомат **autobaud** ожидает следующей действительной AT-команды. Если **a/ (A/)** получена, **a/ (A/)** сохраняется в **RX FIFO**, и конечный автомат ожидает следующей допустимой AT-команды.

При первом успешном обнаружении скорости передачи **UART** активирует прерывание, чтобы показать, что обнаружена последовательность **AT**(верхний или нижний регистр) . Регистр **UARTi.UART\_UASR** отражает правильные настройки для обнаруженной скорости передачи данных. Прерывание может продолжаться таким же образом, когда последующий символ получен. Поэтому рекомендуется, чтобы программное обеспечение разрешало прерывание **RHR** при использовании режим автобода.

Следующие настройки обнаруживаются в режиме автобода с тактовой частотой модуля 48 МГц:

**- Скорость:**

– 115,2 Кбод

– 57,6 Кбод

– 38,4 Кбод

– 28,8 Кбод

– 19,2 Кбод

– 14,4 Кбод

– 9,6 Кбод

– 4,8 Кбод

– 2,4 Кбод

– 1,2 Кбод

**- Длина: 7 или 8 бит**

**- Четность: нечетный, четный или пробел**

*ПРИМЕЧАНИЕ. Комбинация 7-битного символа и четности пробела не поддерживается.*

Режим Autobauding выбирается, когда в битовом поле **UARTi.UART\_MDR1[2:0] MODE\_SELECT** установлено значение **0x2**.В режиме автоматической настройки UART настройки битовых полей **UARTi.UART\_DLL**, **UARTi.UART\_DLH** и **UARTi.UART\_LCR[5:0]** не используются; вместо этого **UART\_UASR** обновляется конфигурацией, обнаруженной логикой автобаудинга.

**Использование регистра статуса автобода UART\_UASR.**

Этот регистр используется для настройки передачи в соответствии с характеристиками предыдущего приема.

вместо регистров **UARTi.UART\_LCR, UARTi.UART\_DLL** и **UARTi.UART\_DLH**, когда **UART** в режиме автобаудинга.

Чтобы сбросить аппаратное обеспечение автобода (чтобы начать обнаружение нового **AT**) или установить **UART** в стандартный режим (без **autobaud**), битовое поле **UARTi.UART\_MDR1[2:0] MODE\_SELECT** должно быть установлено в состояние сброса (**0x7**), а  затем в режим автопередачи (**0x2**) или в стандартный режим (**0x0**).

Ограничения в использовании:

- Только 7- и 8-битные символы (5- и 6-битные не поддерживаются)

- 7-битный символ с пробелом не поддерживается

- Скорость передачи от 1200 до 115 200 бит/с (10 вариантов)

**19.3.8.1.3.5 Обнаружение ошибок**

При чтении регистра **UARTi.UART\_LSR** битовое поле **UARTi.UART\_LSR[4:2]** отражает биты ошибок

(BI: условие разрыва, FE: ошибка кадра, PE: ошибка четности) символов верхней части RX FIFO (следующий символ для чтения). Следовательно, чтение регистра **UARTi.UART\_LSR**, а затем чтение регистра **UARTi.UART\_RHR** идентифицирует ошибки в символе. Чтение регистра **UARTi.UART\_RHR** обновляет биты **BI**, **FE** и **PE** (см. таблицу 19-11 прерывания режима UART ).

Бит **UARTi.UART\_LSR[7] RX\_FIFO\_STS** устанавливается при наличии ошибки в **RX FIFO** и сбрасывается

только когда в **RX FIFO** не осталось ошибок.

*ПРИМЕЧАНИЕ. Чтение регистра UARTi.UART\_LSR не приводит к приращению указателя чтения RX FIFO . Указатель чтения RX FIFO увеличивается путем чтения UARTi.UART\_RHR регистра.*

Чтение регистра UARTi.UART\_LSR очищает бит OE, если он установлен (см. таблицу 19-11 прерывания режима UART).

**19.3.8.1.3.6 Переполнение во время приема**

Переполнение во время приема происходит, если конечный автомат RX пытается записать данные в RX FIFO, когда он уже полный. Когда происходит переполнение, устройство прерывает **MPU** с помощью установки битового поля **UARTi.UART\_IIR[5:1]** **IT\_TYPE** в **0x3** (ошибка состояния линии приемника) и отбрасывает оставшуюся часть кадра.

Переполнение также приводит к установке внутреннего флага, запрещающего дальнейший прием. Прежде чем следующий кадр может быть получен, **MPU** должен:

- сбросить RX FIFO.

- Прочитать регистр **UARTi.UART\_RESUME**, который очищает внутренний флаг.

**19.3.8.1.3.7 Условия тайм-аута и перерыва**

**19.3.8.1.3.7.1 Счетчик времени ожидания**

Состояние ожидания RX обнаруживается, когда линия приемника (**uarti\_rx**) имеет высокий уровень в течение времени, равного 4-кратной запрограммированной длине слова + 12 бит. **uarti\_rx** сэмплируется в середине каждого бита.

Для спящего режима счетчик сбрасывается при активности на **uarti\_rx**.

Для прерывания по тайм-ауту счетчик считает только тогда, когда в **RX FIFO** есть данные, и счетчик

сбрасывается при наличии активности на **uarti\_rx** или при чтении регистра **UARTi.UART\_RHR**.

**19.3.8.1.3.7.2 Состояние обрыва**

Когда возникает состояние обрыва, **uarti\_tx** становится низким. Условие останова активируется установкой бита

**UARTi.UART\_LCR[6] BREAK\_EN** . Условие разрыва не выровнено по словесному потоку (разрыв условие может возникнуть в середине символа). Единственный способ отправить условие останова при полном символе:

1. сбросьте **TX FIFO** (если он включен).

2. подождите, пока сдвиговый регистр передачи не опустеет (бит **UARTi.UART\_LSR[6] TX\_SR\_E** установлен в **1**).

3. Возьмите защитное время в соответствии с определением стоп-бита.

4. Установите бит **BREAK\_EN** в 1.

Условие прерывания устанавливается, пока бит **BREAK\_EN** установлен в 1.

Счетчик тайм-аута и условие прерывания применяются только к работе модема **UART**, но не к режиму IrDA/CIR.

**19.4 Базовая модель программирования UART/IrDA/CIR**

**19.4.1 Модель программирования UART**

**19.4.1.1 Быстрый старт**

В этом разделе описывается процедура работы с **FIFO UART** через **DMA** или прерывания. Эта процедура состоящая из трех частей обеспечивает быстрый запуск **UART**. Раздел не охватывает все функции UART.

Первая модель программирования охватывает программный сброс UART. Вторая модель программирования

описывает конфигурацию FIFO и DMA. Последняя модель программирования описывает конфигурацию протокола, скорость передачи и прерывания.

*ПРИМЕЧАНИЕ. Каждую модель программирования можно использовать независимо от двух других; например,*

*перенастройка только параметров FIFO и DMA. Каждая модель программирования может быть выполнена, начиная с любого режима доступа к регистру UART. (режимы регистров, подрежимы и другие зависимости регистров). Однако, если*

*режим доступа к регистру UART известен перед выполнением модели программирования, некоторые шаги, которые активируют или разрешают доступ к регистру восстановления не является обязательными. Для получения дополнительной информации см. Раздел 19.3.7.1, Регистрация режимов доступа.*

**19.4.1.1.1 Программный сброс**

Чтобы очистить регистры UART, выполните следующие действия:

1. Инициируйте программный сброс: Установите бит **UARTi.UART\_SYSC[1] SOFTRESET** в **1**.

2. Дождитесь окончания операции сброса: Опрашивайте бит **UARTi.UART\_SYSS[0] RESETDONE**, пока он не станет равным 1.

**19.4.1.1.2 настройки FIFO и DMA**

Чтобы включить и настроить приемные и передающие **FIFO** и запрограммировать режим DMA, выполните

следующие шаги:

1. Переключитесь в **режим конфигурации регистров B**, чтобы получить доступ к регистру **UARTi.UART\_EFR**:

а. Сохраните текущее значение регистра **UARTi.UART\_LCR**.

б. Установите значение регистра **UARTi.UART\_LCR** на **0x00BF**.

2. Включите подрежим регистра **TCR\_TLR** для доступа к регистру **UARTi.UART\_TLR** (часть 1 из 2):

а. Сохраните значение **UARTi.UART\_EFR[4] ENHANCED\_EN**.

б. Установите бит **UARTi.UART\_EFR[4] ENHANCED\_EN** в **1**.

3. Переключитесь в **режим конфигурации регистров A**, чтобы получить доступ к регистр**у UARTi.UART\_MCR**:

Установите значение регистра **UARTi.UART\_LCR** на **0x0080**.

4. Включите подрежим регистра **TCR\_TLR** для доступа к регистру **UARTi.UART\_TLR** (часть 2 из 2):

а. Сохраните значение **UARTi.UART\_MCR[6] TCR\_TLR**.

б. Установите бит **UARTi.UART\_MCR[6] TCR\_TLR** в **1**.

5. Включите **FIFO**; загрузите новые триггеры **FIFO** (часть 1 из 3) и новый режим **DMA** (часть 1 из 2):

Установите следующие биты в желаемые значения:

- **UARTi.UART\_FCR[7:6] RX\_FIFO\_TRIG**

- **UARTi.UART\_FCR[5:4] TX\_FIFO\_TRIG**

- **UARTi.UART\_FCR[3] DMA\_MODE**

- **UARTi.UART\_FCR[0] FIFO\_ENABLE** (0: отключить **FIFO**; 1: включить **FIFO**)

*ПРИМЕЧАНИЕ. Регистр UARTi.UART\_FCR недоступен для чтения.*

6. Переключитесь в **режим конфигурации регистров B**, чтобы получить доступ к регистру **UARTi.UART\_EFR**:

Установите значение регистра **UARTi.UART\_LCR** на **0x00BF**.

7. Загрузите новые триггеры **FIFO** (часть 2 из 3):

Установите следующие биты в желаемые значения:

- **UARTi.UART\_TLR[7:4] RX\_FIFO\_TRIG\_DMA**

- **UARTi.UART\_TLR[3:0] TX\_FIFO\_TRIG\_DMA**

8. Загрузите новые триггеры **FIFO** (часть 3 из 3) и новый режим **DMA** (часть 2 из 2):

Установите следующие биты в желаемые значения:

- **UARTi.UART\_SCR[7] RX\_TRIG\_GRANU1**

- **UARTi.UART\_SCR[6] TX\_TRIG\_GRANU1**

- **UARTi.UART\_SCR[2:1] DMA\_MODE\_2**

- **UARTi.UART\_SCR[0] DMA\_MODE\_CTL**

9. Восстановите значение **UARTi.UART\_EFR[4] ENHANCED\_EN**, сохраненное на шаге 2a.

10. Переключитесь в **режим конфигурации регистра A**, чтобы получить доступ к регистру **UARTi.UART\_MCR**:

Установите значение регистра **UARTi.UART\_LCR** на **0x0080**.

11. Восстановите значение **UARTi.UART\_MCR[6] TCR\_TLR**, сохраненное на шаге 4a.

12. Восстановите значение **UARTi.UART\_LCR**, сохраненное на шаге 1a.

Триггеры используются для генерации прерываний и запросов **DMA**. См. Раздел 19.3.6.1.1, Триггер передачи FIFO,

выбрать следующие значения:

- **UARTi.UART\_FCR[5:4] TX\_FIFO\_TRIG**

- **UARTi.UART\_TLR[3:0] TX\_FIFO\_TRIG\_DMA**

- **UARTi.UART\_SCR[6] TX\_TRIG\_GRANU1**

Триггеры используются для генерации прерываний и запросов **DMA**. См. Раздел 19.3.6.1.2, Получение триггера **FIFO**, выбрать следующие значения:

- **UARTi.UART\_FCR[7:6] RX\_FIFO\_TRIG**

- **UARTi.UART\_TLR[7:4] RX\_FIFO\_TRIG\_DMA**

- **UARTi.UART\_SCR[7] RX\_TRIG\_GRANU1**

Режим **DMA** разрешает запросы **DMA**. См. Раздел 19.3.6.4, Работа в режиме DMA FIFO, чтобы выбрать

следующие значения:

- **UARTi.UART\_FCR[3] DMA\_MODE**

- **UARTi.UART\_SCR[2:1] DMA\_MODE\_2**

- **UARTi.UART\_SCR[0] DMA\_MODE\_CTL**

**19.4.1.1.3 Протокол, скорость передачи данных и параметры прерывания**

Чтобы запрограммировать протокол, скорость передачи данных и параметры прерывания, выполните следующие действия:

1. Отключить UART для доступа к регистрам **UARTi.UART\_DLL** и **UARTi.UART\_DLH**:

Установите битовое поле **UARTi.UART\_MDR1[2:0] MODE\_SELECT** на **0x7**.

2. Переключитесь в **режим конфигурации регистров B**, чтобы получить доступ к регистру **UARTi.UART\_EFR**:

Установите значение регистра **UARTi.UART\_LCR** на **0x00BF**.

3. Включите доступ к битовому полю **UARTi.UART\_IER[7:4]**:

а. Сохраните значение **UARTi.UART\_EFR[4] ENHANCED\_EN**.

б. Установите бит **UARTi.UART\_EFR[4] ENHANCED\_EN** в **1**.

4. Переключитесь в рабочий режим регистрации для доступа к регистру **UARTi.UART\_IER**:

Установите значение регистра **UARTi.UART\_LCR** на **0x0000**.

5. Очистите регистр **UARTi.UART\_IER** (установите бит **SLEEP\_MODE UARTi.UART\_IER[4]** в **0**, чтобы изменить

регистры **UARTi.UART\_DLL** и **UARTi.UART\_DLH**). Установите значение регистра **UARTi.UART\_IER** в **0x0000**.

6. Переключитесь в **режим конфигурации регистрации B**, чтобы получить доступ к регистрам **UARTi.UART\_DLL** и **UARTi.UART\_DLH** :

Установите значение регистра **UARTi.UART\_LCR** на **0x00BF**.

7. Загрузите новое значение делителя:

Установите битовые поля **UARTi.UART\_DLL[7:0] CLOCK\_LSB** и **UARTi.UART\_DLH[5:0] CLOCK\_MSB** в

желаемые значения.

8. Переключитесь в рабочий режим регистрации для доступа к регистру **UARTi.UART\_IER**:

Установите значение регистра **UARTi.UART\_LCR** на **0x0000**.

9. Загрузите новую конфигурацию прерывания (0: запретить прерывание; 1: разрешить прерывание):

Установите следующие биты в желаемые значения:

- **UARTi.UART\_IER[7] CTS\_IT**

- **UARTi.UART\_IER[6] RTS\_IT**

- **UARTi.UART\_IER[5] XOFF\_IT**

- **UARTi.UART\_IER[4] SLEEP\_MODE**

- **UARTi.UART\_IER[3] MODEM\_STS\_IT**

- **UARTi.UART\_IER[2] LINE\_STS\_IT**

- **UARTi.UART\_IER[1] THR\_IT**

- **UARTi.UART\_IER[0] RHR\_IT**

10. Переключитесь в **режим конфигурации регистров B**, чтобы получить доступ к регистру **UARTi.UART\_EFR**:

Установите значение регистра **UARTi.UART\_LCR** на **0x00BF**.

11. Восстановите значение **UARTi.UART\_EFR[4] ENHANCED\_EN**, сохраненное на шаге 3a.

12. Загрузите новое форматирование протокола (четность, стоп-бит, длина символа) и переключитесь на регистрацию рабочего режима:

Установите бит **UARTi.UART\_LCR[7] DIV\_EN** в **0**.

Установите бит **UARTi.UART\_LCR[6] BREAK\_EN** в **0**.

Установите следующие биты в желаемые значения:

- **UARTi.UART\_LCR[5] PARITY\_TYPE\_2**

- **UARTi.UART\_LCR[4] PARITY\_TYPE\_1**

- **UARTi.UART\_LCR[3] PARITY\_EN**

- **UARTi.UART\_LCR[2] NB\_STOP**

- **UARTi.UART\_LCR[1:0] CHAR\_LENGTH**

13. Загрузите новый режим **UART**:

Установите битовое поле **UARTi.UART\_MDR1[2:0] MODE\_SELECT** на желаемое значение.

См. Раздел 19.3.8.1.2, Выбор подходящего значения делителя, чтобы выбрать следующие значения:

- **UARTi.UART\_DLL[7:0] CLOCK\_LSB**

- **UARTi.UART\_DLH[5:0] CLOCK\_MSB**

- **UARTi.UART\_MDR1[2:0] MODE\_SELECT**

См. Раздел 19.3.8.1.3.1, Форматирование фрейма, чтобы выбрать следующие значения:

- **UARTi.UART\_LCR[5] PARITY\_TYPE\_2**

- **UARTi.UART\_LCR[4] PARITY\_TYPE\_1**

- **UARTi.UART\_LCR[3] PARITY\_EN**

- **UARTi.UART\_LCR[2] NB\_STOP**

- **UARTi.UART\_LCR[1:0] CHAR\_LENGTH**

**19.4.1.2 Конфигурация аппаратного и программного управления потоком данных**

В этом разделе описываются шаги программирования для включения и настройки контроля аппаратного и программного потока. Аппаратное и программное управление потоком нельзя использовать одновременно.

*ПРИМЕЧАНИЕ. Каждая модель программирования может выполняться, начиная с любого режима доступа к регистру UART. (режимы регистров, подрежимы и другие зависимости регистров). Однако, если регистр UART режим доступа известен перед выполнением модели программирования, некоторые шаги, которые разрешают или доступ к реестру восстановления не является обязательным. Для получения дополнительной информации см. Раздел 19.3.7.1, Регистрация режмов доступа.*

**19.4.1.2.1 Конфигурация аппаратного управления потоком**

Чтобы включить и настроить аппаратное управление потоком, выполните следующие действия:

1. Переключитесь в **режим конфигурации регистра A**, чтобы получить доступ к регистру **UARTi.UART\_MCR**:

а. Сохраните текущее значение регистра **UARTi.UART\_LCR**.

б. Установите значение регистра **UARTi.UART\_LCR** на **0x0080**.

2. Включите подрежим регистра **TCR\_TLR** для доступа к регистру **UARTi.UART\_TCR** (часть 1 из 2):

а. Сохраните значение **UARTi.UART\_MCR[6] TCR\_TLR**.

б. Установите бит **UARTi.UART\_MCR[6] TCR\_TLR** в **1**.

3. Переключитесь в **режим конфигурации регистров B**, чтобы получить доступ к регистру **UARTi.UART\_EFR**:

Установите значение регистра **UARTi.UART\_LCR** на **0x00BF**.

4. Включите подрежим регистра **TCR\_TLR** для доступа к регистру **UARTi.UART\_TCR** (часть 2 из 2):

а. Сохраните значение **UARTi.UART\_EFR[4] ENHANCED\_EN**.

б. Установите бит **UARTi.UART\_EFR[4] ENHANCED\_EN** в **1**.

5. Загрузите новые значения запуска и остановки для аппаратного управления потоком:

Установите следующие биты в желаемые значения:

- **UARTi.UART\_TCR[7:4] AUTO\_RTS\_START**

- **UARTi.UART\_TCR[3:0] AUTO\_RTS\_HALT**

6. Включите или отключите режим аппаратного управления потоком приема и передачи и восстановите

Значение **UARTi.UART\_EFR[4] ENHANCED\_EN**, сохраненное на шаге 4a.

Установите следующие биты в желаемые значения:

- **UARTi.UART\_EFR[7] AUTO\_CTS\_EN** (0: Отключено; 1: Включено)

- **UARTi.UART\_EFR[6] AUTO\_RTS\_EN** (0: Отключено; 1: Включено)

Восстановите бит **UARTi.UART\_EFR[4] ENHANCED\_EN** до сохраненного значения.

7. Переключитесь в **режим конфигурации регистра A**, чтобы получить доступ к регистру **UARTi.UART\_MCR**:

Установите значение регистра **UARTi.UART\_LCR** на **0x0080**.

8. Восстановите значение **UARTi.UART\_MCR[6] TCR\_TLR**, сохраненное на шаге 2a.

9. Восстановите значение **UARTi.UART\_LCR**, сохраненное на шаге 1a.

См. Раздел 19.3.8.1.3.2, Аппаратное управление потоком, чтобы выбрать следующие значения:

- **UARTi.UART\_EFR[7] AUTO\_CTS\_EN**

- **UARTi.UART\_EFR[6] AUTO\_RTS\_EN**

- **UARTi.UART\_TCR[7:4] AUTO\_RTS\_START**

- **UARTi.UART\_TCR[3:0] AUTO\_RTS\_HALT**

**19.4.1.2.2 Конфигурация программного управления потоком данных**

Чтобы включить и настроить программное управление потоком, выполните следующие действия:

1. Переключитесь в **режим конфигурации регистров B**, чтобы получить доступ к регистру **UARTi.UART\_EFR**.

а. Сохраните текущее значение регистра **UARTi.UART\_LCR**.

б. Установите значение регистра **UARTi.UART\_LCR** на **0x00BF**.

2. Включите подрежим регистрации **XOFF** для доступа к регистрам **UARTi.UART\_XOFF1** и **UARTi.UART\_XOFF2**:

а. Сохраните значение **UARTi.UART\_EFR[4] ENHANCED\_EN**.

б. Установите бит **UARTi.UART\_EFR[4] ENHANCED\_EN** в **0**.

3. Загрузите новые символы программного управления потоком:

Установите следующие биты в желаемые значения:

- **UARTi.UART\_XON1\_ADDR1[7:0] XON\_WORD1**

- **UARTi.UART\_XON2\_ADDR2[7:0] XON\_WORD2**

- **UARTi.UART\_XOFF1[7:0] XOFF\_WORD1**

- **UARTi.UART\_XOFF2[7:0] XOFF\_WORD2**

4. Включите доступ к битовому полю **UARTi.UART\_MCR[7:5]** и включите подрежим регистра **TCR\_TLR** для

получить доступ к регистру **UARTi.UART\_TCR** (часть 1 из 2):

Установите бит **UARTi.UART\_EFR[4] ENHANCED\_EN** в **1**.

5. Переключитесь в **режим конфигурации регистров A,** чтобы получить доступ к регистру **UARTi.UART\_MCR**:

Установите значение регистра **UARTi.UART\_LCR** на **0x0080**.

6. Включите подрежим регистра **TCR\_TLR** для доступа к регистру **UARTi.UART\_TCR** (часть 2 из 2) и включите

или отключите любую функцию **XON**:

а. Сохраните значение **UARTi.UART\_MCR[6] TCR\_TLR**.

б. Установите бит **UARTi.UART\_MCR[6] TCR\_TLR** в **1**.

в. Установите бит **UARTi.UART\_MCR[5] XON\_EN** на желаемое значение (0: отключено; 1: включено).

7. Переключитесь в **режим конфигурации регистров B**, чтобы получить доступ к регистру **UARTi.UART\_EFR**:

Установите значение регистра **UARTi.UART\_LCR** на **0x00BF**.

8. Загрузите новые значения запуска и остановки для программного управления потоком данных:

Установите следующие биты в желаемые значения:

- **UARTi.UART\_TCR[7:4] AUTO\_RTS\_START**

- **UARTi.UART\_TCR[3:0] AUTO\_RTS\_HALT**

9. Включите или отключите функцию специального символа и загрузите новый режим управления потоком программного обеспечения и восстановите значение **UARTi.UART\_EFR[4] ENHANCED\_EN**, сохраненное на шаге 2а:

Установите следующие биты в желаемые значения:

- **UARTi.UART\_EFR[5] SPEC\_CHAR** (0: Отключить; 1: Включить)

- **UARTi.UART\_EFR[3:0] SW\_FLOW\_CONTROL**

Восстановите бит **UARTi.UART\_EFR[4] ENHANCED\_EN** до сохраненного значения.

10. Переключитесь в **режим конфигурации регистра A**, чтобы получить доступ к регистру **UARTi.UART\_MCR**:

Установите значение регистра **UARTi.UART\_LCR** на **0x0080**.

11. Восстановите значение бита **UARTi.UART\_MCR[6] TCR\_TLR**, сохраненное на шаге 6a.

12. Восстановите значение **UARTi.UART\_LCR**, сохраненное на шаге 1a.

См. Раздел 19.3.8.1.3.3, Программное управление потоком данных, чтобы выбрать следующие значения:

- **UARTi.UART\_EFR[5] SPEC\_CHAR**

- **UARTi.UART\_EFR[3:0] SW\_FLOW\_CONTROL**

- **UARTi.UART\_TCR[7:4] AUTO\_RTS\_START**

- **UARTi.UART\_TCR[3:0] AUTO\_RTS\_HALT**

- **UARTi.UART\_XON1\_ADDR1[7:0] XON\_WORD1**

- **UARTi.UART\_XON2\_ADDR2[7:0] XON\_WORD2**

- **UARTi.UART\_XOFF1[7:0] XOFF\_WORD1**

- **UARTi.UART\_XOFF2[7:0] XOFF\_WORD2**