МИКРОПРОЦЕССОР МСТ-01

Руководство пользователя

Перечень сокращений:

- **CPU** – центральный процессор на основе RISC-ядра;
- **FPU** – сопроцессором арифметики в формате с плавающей точкой;
- **CRAM** – двухпортовая оперативная память центрального процессора;
- **DMA** – контроллер прямого доступа в память;
- **MPORT** – порт внешней памяти;
- **SWIC** – контроллер канала SpaceWire;
- **LPORT** – линковый порт;
- **UART** – универсальный асинхронный порт;
- **ICACHE** – кэш программ центрального процессора;
- **DCACHE** – кэш данных центрального процессора;
- **IT** – интервальный таймер;
- **WDT** – сторожевой таймер;
- **RTT** – таймер реального времени;
- **A[31:0]** – шина адреса порта внешней памяти;
- **D[31:0]** – шина данных порта внешней памяти;
- **OnCD** – встроенные средства отладки программ;

ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ

1.1 Назначение

Микросхема интегральная МСТ-01 спроектирована как “система на кристалле” на базе IP-ядерной (IP-intellectual property) платформы «МУЛЬТИКОР», разработанной в ГУП НПЦ «ЭЛВИС».

МСТ-01 обеспечивает работу под операционной системой **Linux**, а также под другими операционными системами для встраиваемых применений.

МСТ-01 предназначен для применения в следующих приложениях:

- Системы промышленного контроля;
- Фильтрация, корреляция, быстрая свертка;

1.2 Функциональные параметры и возможности

Микропроцессор МСТ-01 имеет следующие функциональные параметры и возможности:

☐ Центральный процессор (CPU):

- Архитектура – MIPS32;
- 32-х битные шины передачи адреса и данных;
- Кэш команд объемом 16 Кбайт;
- Кэш данных объемом 16 Кбайт;
- Архитектура привилегированных ресурсов в стиле ядра R4000:
  - Регистры Count/Compare для прерываний реального времени;
  - Отдельный вектор обработки исключений по прерываниям;
- Программируемое устройство управления памятью:
  - Два режима работы – с TLB (Translation Lookaside Buffer) и FM (Fixed Mapped);
  - 16 строк в режиме TLB.
- Устройство целочисленного умножения и деления;
- Сопроцессором арифметики в формате с плавающей точкой;
- JTAG IEEE 1149.1, встроенные средства отладки программ;
- Производительность – 100 млн. оп/сек (здесь и далее параметры производительности приведены при тактовой частоте 100 МГц);
- Оперативная память центрального процессора (CRAM) объемом 128 Кбайт;
- 5 внешних запросов прерывания, в том числе немаскируемое прерывание (NMI).

☐ Порт внешней памяти (MPORT):

- Шина данных – 32 разряда, шина адреса – 32 разряда;
- Встроенный контроллер управления статической памятью типа SRAM, FLASH, ROM, а также синхронной памятью типа SDRAM;
- Программное конфигурирование типа блока памяти и его объема;
- Программное задание циклов ожидания;
- Формирование сигналов выборки 4 блоков внешней памяти;
- Обеспечение обслуживания 4 внешних прерываний.

☐ Периферийные устройства:

- 8 - канальный контроллер прямого доступа в память (DMA). 4 внешних запроса прямого доступа; Специальные режимы синхронизации. Поддержка 2-мерной и разрядно-инверсной адресации. Режим передачи Flyby, подобный реализованному в ADSP-TS201: внешнее устройство ⇔ внешняя память;
- два линковых порта (LPORT) совместимые с ADSP21160. Имеется режим работы в качестве портов ввода-вывода общего назначения (GPIO);
- два дуплексных канала SpaceWire с пропускной способностью не менее 200 Мбит/с каждый;
- универсальный асинхронный порт (UART) типа 16550;
- 32-разрядный интервальный таймер (IT);
- 32-разрядный таймер реального времени (RTT);
- 32-разрядный сторожевой таймер (WDT).

☐ Дополнительные возможности и особенности:

- Узел фазовой автоподстройки частоты (PLL) с умножителем/делителем входной частоты;
- Встроенные средства отладки программ (OnCD);
- Порт JTAG в соответствии со стандартом IEEE 1149.1;
- Режимы энергосбережения;
- Поддержка операционной системы Linux;
- Корпус: QFP-240.

1.3 Структурная схема

Структурная схема МСТ-01 приведена на Рисунок 1.1.

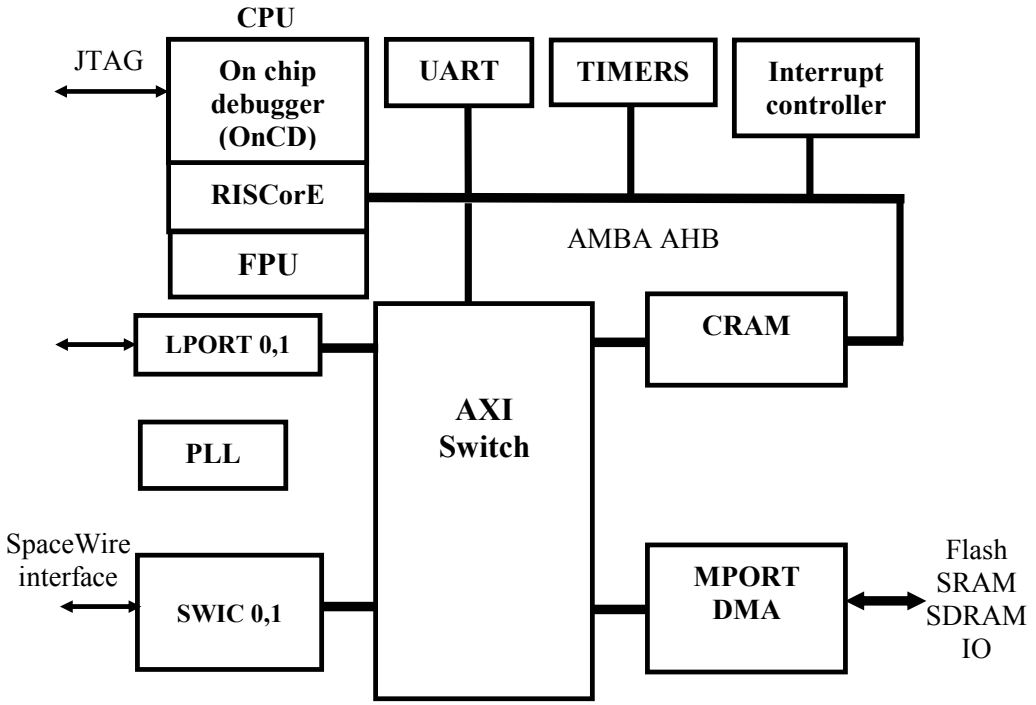


Рисунок 1.1. Структурная схема периферийного контроллера

В состав МСТ-01 входят следующие основные узлы:

- CPU – центральный процессор на основе RISC-ядра и сопроцессора арифметики в формате с плавающей точкой (FPU);
- CRAM – двухпортовая оперативная память центрального процессора;
- MPORT – порт внешней памяти;
- DMA – контроллер прямого доступа в память;
- два линковых порта (LPORT);
- два контроллера интерфейса SpaceWire (SWIC);
- OnCD – встроенные средства отладки программ;
- UART – асинхронный последовательный порт;
- TIMERS – таймеры;
- Контроллер прерываний;
- Коммутатор;
- PLL – умножитель частоты на основе PLL;
- JTAG – отладочный порт.

1.4 Инструментальное программное обеспечение

Для микросхемы МСТ-01 разработана интегрированная среда проектирования программного обеспечения **MCStudio™**, которая обеспечивет полный цикл разработки и отладки программ. MCS является кросс - системой и функционирует на инструментальной машине IBM PC в среде Windows 9x, XP.

Интегрированная среда проектирования включает:

- среду разработки программ для RISC – ядра;
- среду отладки программ в исходных текстах, исполняемых на программном симуляторе, и отладчик для работы с платой отладочного модуля (МСТ-01ЕМ) для микросхемы МСТ-01 или целевым устройством через JTAG;
- средства программного моделирования;
- возможность доступа пользователю ко всем инструментам через один интерфейс.

Среда разработки программ для RISC – ядра включает:

- компилятор с языка Си с препроцессором;
- ассемблер с препроцессором;
- дисассемблер;
- линковщик;
- библиотекарь;
- утилиты подготовки исполняемого кода.

Описание интегрированной среды и инструментального программного обеспечения приведено в документации (см. раздел 1.7 «Дополнительная документация»).

В состав отладочного комплекта МСТ-01ЕМ входят:

- Интегрированная среда разработки и отладки программ MCStudio™;
- Отладочный модуль с JTAG-отладчиком;
- Набор кабелей и источник питания;
- Библиотека прикладных программ для МСТ-01 (*Поставляется как опция).*

Инструментальное программное обеспечение МСТ-01 базируется на архитектуре MIPS32. Вследствие этого, оно поддерживает большой объем свободно распространяемого программного обеспечения для этой архитектуры.

Библиотека прикладных программ для микросхемы МСТ-01 включает:

- элементарные математические функции;
- арифметические операции над матрицами.

1.5 Операционная система для микросхемы МСТ-01.

Linux - свободно распространяемое ядро Unix-подобной операционной системы. Linux обладает всеми свойствами современной Unix-системы, включая полноценную многозадачность, развитую подсистему управления памятью и сетевую подсистему.

Ядро Linux, поставляемое вместе со свободно распространяемыми прикладными и системными программами образует полнофункциональную универсальную операционную систему. Большую часть базовых системных компонент Linux унаследовал

от проекта GNU, целью которого является создание свободной микроядерной операционной системы с лицом Unix.

В качестве дополнительной опции для микросхемы МСТ-01 в составе отладочного модуля МСТ-01ЕМ может быть портировано ядро операционной системы Linux версий 2.4.17, 2.4.25, 2.6, 5.

1.6 Дополнительная документация

Дополнительно при изучении данного руководства рекомендуется использовать следующие документы:

- Процессорное ядро RISCore32. Система команд;
- Интегрированная среда разработки и отладки программ MCStudio Установка среды MCStudio. Руководство системного программиста;
- Интегрированная среда разработки и отладки программ MCStudio. Описание пользовательского интерфейса. Руководство оператора;
- Интегрированная среда разработки и отладки программ MCStudio. Руководство программиста;
- Интегрированная среда разработки и отладки программ MCStudio. Инструменты ядра RISC. Руководство оператора.

2. ЦЕНТРАЛЬНЫЙ ПРОЦЕССОР

2.1 Основные характеристики CPU

- Архитектура – MIPS32;
- 32-х битные пути передачи адреса и данных;
- Кэш команд объемом 16 Кбайт;
- Кэш данных объемом 16 Кбайт;
- Архитектура привилегированных ресурсов в стиле ядра R4000:
  - Регистры Count/Compare для прерываний реального времени;
  - Отдельный вектор обработки исключений по прерываниям;
- Программируемое устройство управления памятью:
  - Два режима работы – с TLB (Translation Look aside Buffer) и FM (Fixed Mapped);
  - 16 строк в режиме TLB;
- Устройство целочисленного умножения и деления;
- Сопроцессором арифметики в формате с плавающей точкой;
- Поддержка отладки JTAG.

2.2 Блок схема

Блок схема процессорного ядра RISCore32 приведена на Рисунок 2.1.

Ядро содержит следующие узлы:

- Устройство исполнения (Execution Core);
- Устройство целочисленного умножения и деления (MDU);
- Системный управляющий сопроцессор (CP0);
- Сопроцессор арифметики в формате с плавающей точкой (FPU);
- Устройство управления памятью (MMU – Memory Management Unit);
- Контроллер кэш (Cache Controller);
- Устройство шинного интерфейса (BIU);
  - Кэш команд (Instruction Cache);
  - Кэш данных (Data Cache);
- Средства отладки программ (OnCD – On Chip Debugger) с JTAG портом.

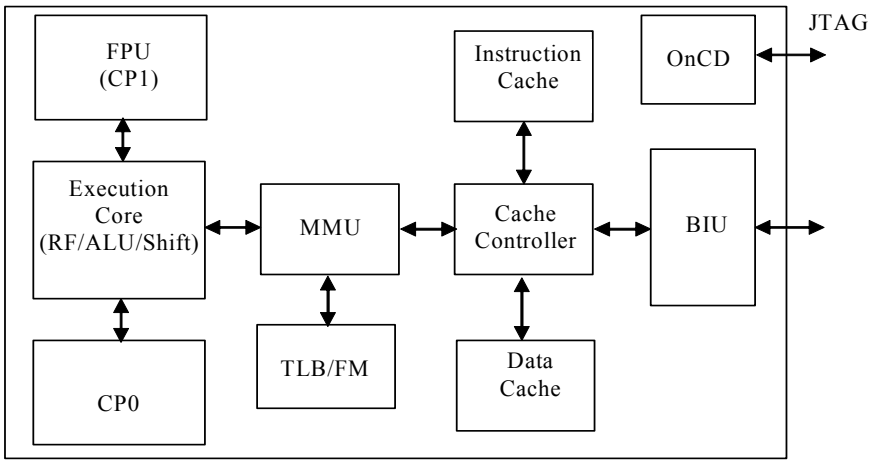


Рисунок 2.1. Блок схема процессорного ядра

2.3 Составляющие логические блоки

В следующих подразделах описываются устройства, входящие в состав процессорного ядра.

2.3.1 Устройство исполнения

Входящее в ядро устройство исполнения реализует архитектуру load-store (загрузка-сохранение) с однотактными операциями арифметического логического устройства (АЛУ) (логические операции, операции сдвига, сложение и вычитание). В ядре имеется тридцать два 32-х битных регистра общего назначения, используемых для скалярных целочисленных операций и вычисления адреса. В регистровом файле есть два порта чтения и один порт записи. Также используются обходные пути передачи данных для минимизации количества остановок конвейера.

В состав устройства исполнения входят:

- 32-х битный сумматор, используемый для вычисления адреса данных;
- Адресное устройство для вычисления адреса следующей команды;
- Логика определения перехода и вычисления адреса перехода;
- Блок выравнивания при загрузке данных;
- Мультиплексоры обходных путей передачи данных для исключения остановок конвейера в тех случаях, когда команды, производящие данные и команды, использующие эти данные, расположены в программе достаточно близко;
- Блок обнаружения Нуля/Единицы для реализации команд CLZ и CLO;
- АЛУ для выполнения побитных операций;
- Сдвигающее устройство и устройство выравнивания при сохранении данных.

2.3.2 Устройство целочисленного умножения и деления (MDU)

Устройство умножения/деления выполняет соответствующие операции. MDU выполняет операции умножения за 17 тактов, операции умножения с накоплением за 18 тактов, операции деления за 33 такта и операции деления с накоплением за 34 такта. Попытка активизировать следующую команду умножения/деления до завершения выпол-

нения предыдущей, так же как и использование результата этой операции до того, как она закончена, вызывает остановку конвейера. В MDU имеется вывод, определяющий формат операции – знаковый или беззнаковый.

2.3.3 Системный управляющий сопроцессор

Сопроцессор отвечает за преобразование виртуального адреса в физический, протоколы кэш, систему управления исключениями, выбор режима функционирования (Kernel/User) и за разрешение/запрещение прерываний. Конфигурационная информация доступна посредством чтения регистров CP0.

2.3.4 Сопроцессор арифметики в формате с плавающей точкой (FPU)

Сопроцессор арифметики в формате с плавающей точкой выполняет операции в соответствии со стандартом ANSI/IEEE Standard 754-1985, "IEEE Standard for Binary Floating-Point Arithmetic." Поддерживаются операции, как с одинарной, так и с двойной точностью. Сопроцессор выполняет дополнительные операции, не определенные стандартом. Сопроцессор содержит 16 64-разрядных регистра для хранения операндов с одинарной и двойной точностью. Сопроцессор также содержит регистры управления и состояния, которые обеспечивают обработку исключений в соответствии с требованиями стандарта.

2.3.5 Устройство управления памятью (MMU)

Процессорное ядро содержит устройство управления памятью (MMU), реализующее интерфейс между исполнительным блоком и контроллером кэш. Ядро может работать как в режиме TLB – с 16-строчной, полностью ассоциативной матрицей TLB, так и в режиме FM (Fixed Mapped), когда используются простые преобразования виртуального адреса в физический адрес.

2.3.6 Контроллер кэш

В данной версии процессора реализованы кэш команд и данных, виртуально индексируемые и контролируемые по физическому тэгу типа direct mapped, что позволяет осуществлять доступ к кэш параллельно с преобразованием виртуального адреса в физический адрес. Объем каждой из кэш составляет 16 Кбайт.

2.3.7 Устройство шинного интерфейса (BIU – Bus Interface Unit)

Устройство шинного интерфейса управляет внешними интерфейсными сигналами в соответствии со спецификацией шины AHB (Advanced High-performance Bus) архитектуры AMBA (Advanced Microcontroller Bus Architecture).

2.3.8 OnCD контроллер

В ядре имеется устройство для отладки программ OnCD с портом JTAG.

2.4 Конвейер

В RISC-ядре процессора реализован конвейер, состоящий из пяти стадий и аналогичный конвейеру ядра R3000. Конвейер дает возможность процессору работать на высо-

кой частоте, при этом минимизируется сложность устройства, а также уменьшается стоимость и потребление энергии.

В этой главе содержатся следующие разделы:

- Раздел 2.1, “Стадии работы конвейера”
- Раздел 2.2, “Операции умножения и деления”
- Раздел 2.3, “Задержка выполнения команд перехода”
- Раздел 2.4, “Обходные пути передачи данных (Data bypass)”
- Раздел 2.5, “Задержка загрузки данных”
- Раздел 2.6, “Особые случаи при выполнении команд (Instruction Hazards)”

2.4.1 Стадии конвейера

Конвейер содержит пять стадий:

- Выборка команды (стадия I- Instruction)
- Дешифрация команды (стадия D - Data)
- Исполнение команды (стадия E - Execution)
- Выборка из памяти (стадия M - Memory)
- Обратная запись (стадия W – Write Back)

На Рисунок 2.2 показаны операции, выполняемые RISC-ядром на каждом этапе конвейера.

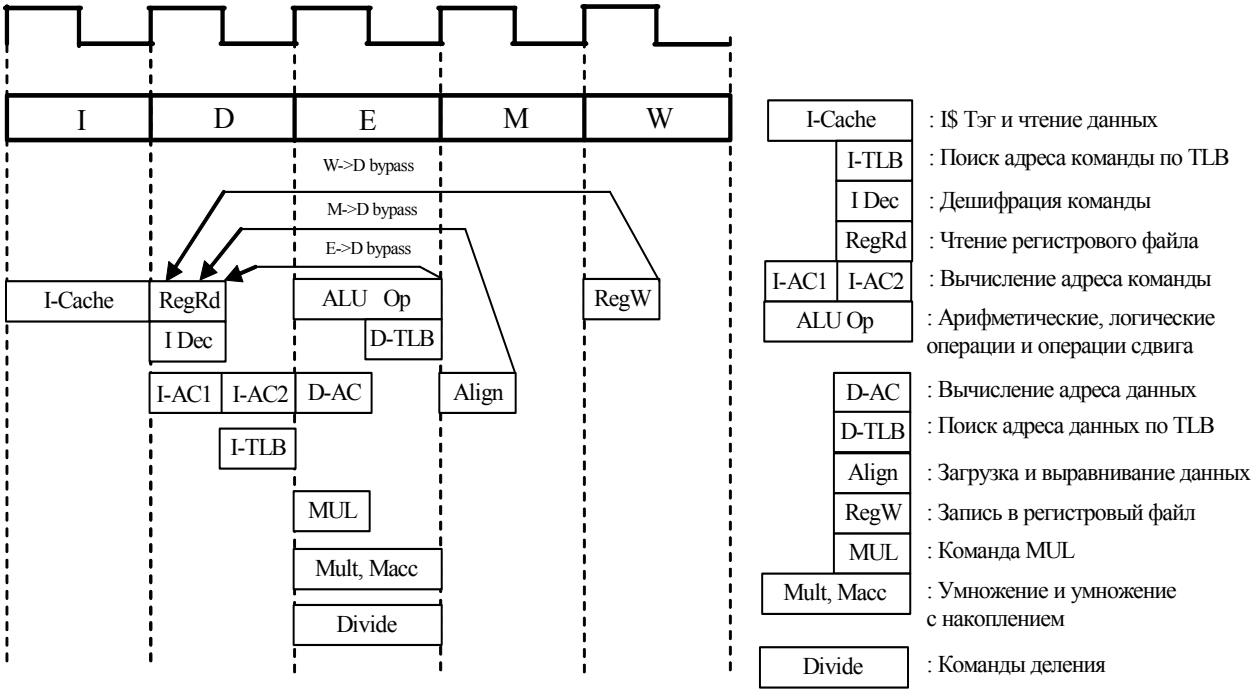


Рисунок 2.2

2.4.1.1 Стадия I: выборка команды

На этой стадии команда выбирается из командного кэш.

2.4.1.2 Стадия D: дешифрация команды

На этой стадии:

- Операнды выбираются из регистрового файла.
- Операнды передаются на эту стадию со стадий E, M и W.
- ALU определяет, выполняется ли условие перехода и вычисляет виртуальный адрес перехода для команд перехода.
- Осуществляется преобразование виртуального адреса в физический адрес.
- Производится поиск адреса команды по TLB и вырабатывается признак hit/miss.
- Командная логика выбирает адрес команды.

2.4.1.3 Стадия E: исполнение

На этой стадии:

- ALU выполняет арифметические или логические операции для команд типа регистр-регистр.
- Производится преобразование виртуального адреса в физический адрес для данных, используемых командами загрузки и сохранения.
- Производится поиск данных по TLB и вырабатывается признак hit/miss.
- Все операции умножения и деления выполняются на этой стадии.

2.4.1.4 Стадия M: выборка из памяти

На этой стадии осуществляется загрузка и выравнивание загруженных данных в границах слова.

2.4.1.5 Стадия W: обратная запись

На этой стадии для команд типа регистр-регистр или для команд загрузки результат записывается обратно в регистровый файл.

2.4.2 Операции умножения и деления

Время выполнения этих операций соответствует 17 тактам для команд умножения и 18 тактам для команд умножения с накоплением, а также 33 тактам для команд деления и 34 тактам для команд деления с накоплением.

2.4.3 Задержка выполнения команд перехода (Jump, Branch)

Конвейер осуществляет выполнение команд перехода с задержкой в один такт. Однотактная задержка является результатом функционирования логики, ответственной за принятие решения о переходе на стадии D конвейера. Эта задержка позволяет использовать адрес перехода, вычисленный на предыдущей стадии, для доступа к команде на следующей D-стадии. Слот задержки перехода (branch delay slot) позволяет отказаться от остановок конвейера при переходе. Вычисление адреса и проверка условия перехода выполняются одновременно на стадии D. Итоговое значение PC (счетчика команд) используется для выборки очередной команды на стадии I, которая является второй командой после перехода. На Рисунок 2.3 показан слот задержки перехода.

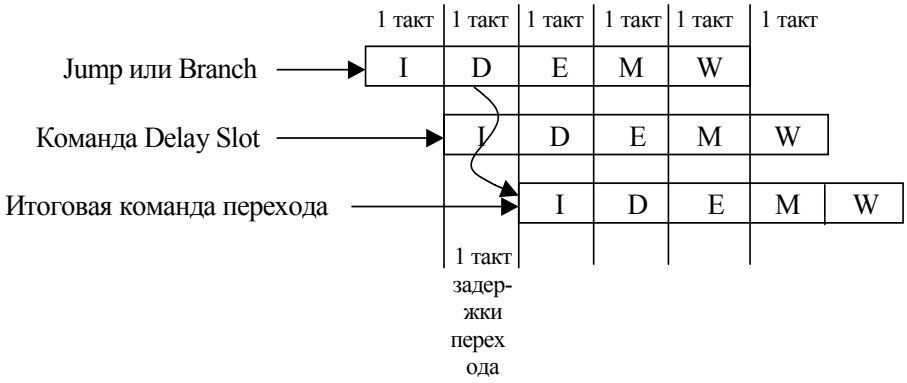


Рисунок 2.3. Слот задержки перехода

2.4.4 Обходные пути передачи данных (Data bypass)

Для большинства команд MIPS32 исходными операндами являются значения, хранящиеся в регистрах общего назначения. Эти операнды выбираются из регистрового файла в первой половине D-стадии. После исполнения на ALU результат, в принципе, готов для использования другими командами. Но запись результата в регистровый файл осуществляется только на стадии W. Это лишает следующую команду возможности использовать результат в течение 3-х циклов, если ее операндом является результат выполнения последней операции, сохраненный в регистровом файле. Для преодоления этой проблемы используются обходные пути передачи данных.

Мультиплексоры обходных путей передачи данных для обоих операндов располагаются между регистровым файлом и ALU (Рисунок 2.4). Они позволяют передавать данные с выхода стадий E, M и W конвейера прямо на стадию D, если один из регистров источника (source) декодируемой команды совпадает с регистром назначения (target) одной из предшествующих команд. Входы мультиплексоров подключены к обходным путям M→D и E→D, а также W→D.

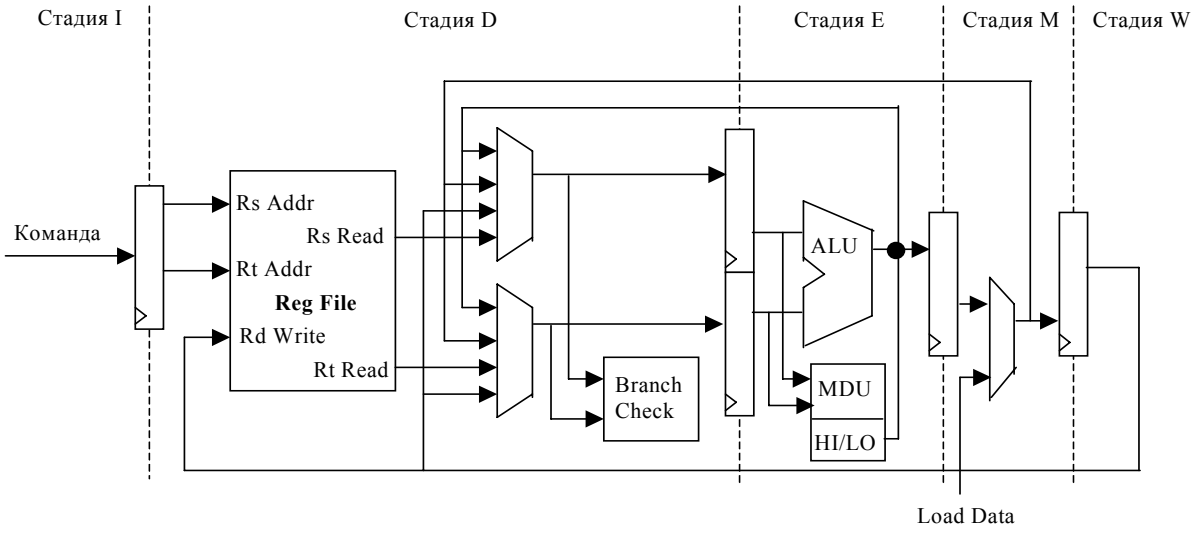


Рисунок 2.4

На Рисунок 2.5 показаны обходные пути передачи данных для команды Add_1, за которой следует команда Sub_2 и затем снова Add_3. Поскольку команда Sub_2 в качестве одного из операндов использует результат операции Add_1, используется обходной путь

E→D. Следующая команда Add3 использует результаты обеих предшествующих операций: Add1 и Sub2. Так как данные команды Add1 в это время находятся на стадии M, используется обходной путь M→D. Кроме того, вновь используется обходной путь E→D для передачи результата операции Sub2 команде Add3.

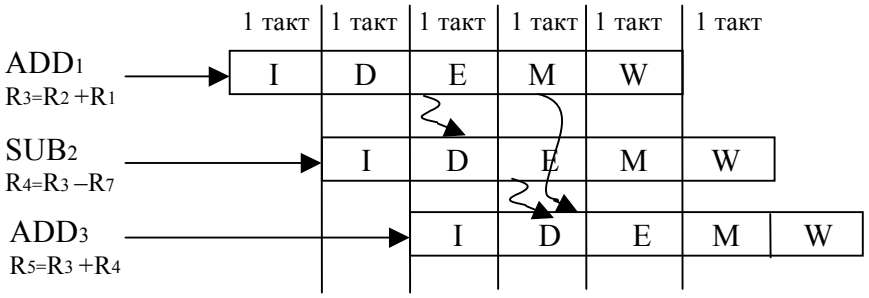


Рисунок 2.5

2.4.5 Задержка загрузки данных

Данные, выбираемые командами загрузки (Load), становятся доступными на конвейере только после выравнивания на стадии M. При этом данные, являющиеся исходными операндами, должны предоставляться командам для обработки уже на стадии D. Поэтому, если сразу за командой загрузки следует команда, для которой один из регистров исходных операндов совпадает с регистром, в который производится загрузка данных, это вызывает приостановку в работе конвейера на стадии D. Эта приостановка осуществляется аппаратной вставкой команды NOP. Во время этой задержки часть конвейера, которая находится дальше стадии D, продолжает продвигаться. Если же команда, использующая загружаемые данные, следует за командой загрузки не сразу, а через одну или через две, то для обеспечения бесперебойной работы конвейера используется один из обходных путей передачи данных: M→D или W→D (Рисунок 2.6).

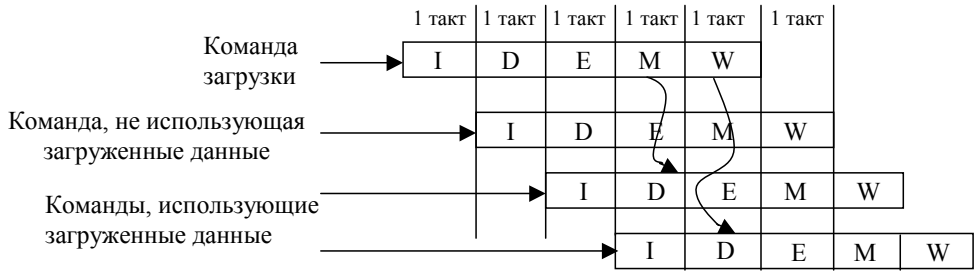


Рисунок 2.6

2.5 Сопроцессор арифметики в формате с плавающей точкой (FPU)

2.5.1 Введение

Сопроцессор арифметики в формате с плавающей точкой выполняет операции в соответствии со стандартом ANSI/IEEE Standard 754-1985, "IEEE Standard for Binary Floating-Point Arithmetic." Поддерживаются операции, как с одинарной, так и с двойной точностью (single- or double-precision). Сопроцессор выполняет дополнительные операции не определенные стандартом. Сопроцессор содержит 16 64-разрядных регистра для хранения операндов с одинарной и двойной точностью. Сопроцессор также содержит регистры управления и состояния, которые обеспечивают обработку исключений в соответствии с требованиями стандарта.

FPU реализован как сопроцессор CP1.

2.5.2 Регистры FPU

2.5.2.1 Типы регистров

В FPU имеется три типа регистров:

- регистры общего назначения (FGR);
- регистры в формате с плавающей точкой (FPR);
- регистры управления (FCR).

32-разрядные регистры FGR являются прямо адресуемыми. FPU содержит 32 таких регистра.

64-разрядные регистры в формате с плавающей точкой FPR являются логическими и используются для хранения данных в процессе выполнения операций в формате с плавающей точкой. Эти регистры образованы конкатенацией двух соседних регистров FGR. В зависимости от операции, FPR содержит величину с одинарной или двойной точностью.

Регистры управления регистры FCR используются для выбора режима округления, обработки исключений и сохранения состояния.

В Таблица 2.1. приведены регистры управления FPU в порядке возрастания нумерации.

Таблица 2.1. Управляющие регистры FPU

Номер регистра	Название Регистра	Функция
0	FIR	Регистр версии и реализации (Implementation and Revision register)
25	FCCR	Регистр кодов условий (Condition Codes register)
26	FEXR	Регистр исключений (Exceptions register)
28	FENR	Регистр разрешения исключений (Enables register)
31	FCSR	Регистр управления и состояния (Control/Status register)

В командах CTC1 и CFC1 регистры FCCR, FEXR и FENR получают доступ к соответствующим частям регистра FCSR, т.е. эти регистры являются отражением соответствующих частей регистра FCSR.

Доступ к регистрам управления FPU не является привилегированным. Любая программа, которая выполняет инструкции с плавающей точкой, имеет доступ к регистрам управления FPU. Доступ к ним осуществляется посредством CTC1 и CFC1 команд.

2.5.2.2 Регистры общего назначения и регистры в формате с плавающей точкой

32 регистра общего назначения (FGR) являются 32-разрядными и могут непосредственно адресоваться. Они используются в операциях в формате с плавающей точкой и индивидуально доступны по командам move, load и store. Перечень регистров FGR приведен в Таблица 2.2.

Таблица 2.2. Регистры FGR и FPR

Номер регистра FGR	Название регистра FGR	Название регистра FPR
0	FGR0	FPR0 (least)
1	FGR1	FPR0 (most)
2	FGR2	FPR2 (least)
3	FGR3	FPR2 (most)
. . .	. . .	. . .
28	FGR28	FPR28 (least)
29	FGR29	FPR28 (most)
30	FGR30	FPR30 (least)
31	FGR31	FPR30 (most)

Регистры в формате с плавающей точкой (FPR) формируются из регистров FGR, посредством их конкатенации. Для адресации этих регистров используется только четный номер. Нечетный номер является недопустимым. В процессе операций с одинарной точностью используется только младшая часть (least) регистра FPR используется.

2.5.2.3 Форматы величин, хранящихся в регистрах FPR

В отличие от процессора целочисленной арифметики, FPU не интерпретирует двоичную кодировку входных операндов и не производит двоичное кодирование результатов каждой операции. Значение, хранящееся в регистре FPR, имеет определенный формат или тип. Этот формат могут использовать только те команды, которые оперируют с ним (этим форматом). Формат может быть неизвестным (не интерпретируемым) либо одним из существующих числовых форматов: формат с плавающей точкой одинарной или двойной точностью, слово или двойное слово с фиксированной точкой.

Числовая величина в регистре FPR всегда установлена, когда она записана в этот регистр:

- при загрузке регистра FPR по команде load в регистр записываются двоичные данные, формат которых не интерпретируется.
- команды вычисления в формате с плавающей точкой или команды move, формируют в регистре FPR результат формата fmt.

Когда регистр FPR с не интерпретируемым значением используется как входной операнд для команды, которая требует значение в формате fmt и рассматривает двоичное содержимое как значение в формате fmt, значение в регистре FPR изменяется к значению в формате fmt. То есть, двоичное содержимое этого регистра не может рассматриваться в другом формате.

Если регистр FPR содержит значение в формате fmt, то вычислительные команды не должны использовать этот регистр как входной операнд другого формата. Если такое происходит, то значение в регистре становится неизвестным и результат команды также является неизвестным значением. Использование FPR регистра с неизвестным значением в качестве входного операнда команды приводит к результату, значение которого также неизвестно.

Формат величины, находящейся в регистре FPR, не изменяется, когда происходит чтение этого регистра командой store. Команда store выводит двоичную кодировку в соответствии со значением, содержащимся в регистре FPR. Если значение в регистре FPR неизвестно, то закодированное двоичное значение, выведенное операцией, неопределенно.

2.5.2.4 Управляющие регистры

2.5.2.4.1 Регистр реализации (FIR, CP1 Control Register 0)

Регистр реализации (Floating Point Implementation Register - FIR) - это 32-битный регистр доступный только на чтение. Он содержит информацию, которая определяет возможности FPU, идентификацию FPU и номер версии FPU. На Рисунок 2.7 показан формат регистра FIR, а в Таблица 2.3 описаны поля этого регистра.

31 18	17	16	15 8	7 0
0	D	S	Processor ID	Revision

Рисунок 2.7. Формат FIR регистра

Таблица 2.3. Описание полей регистра FIR

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	Биты			
-	31:18	Не используется	0	0
D	17	Указывает, реализованы ли тип данных двойной точности (D) и соответствующие инструкции: 0 - не реализованы 1 – реализованы	R	1
S	16	Указывает, реализованы ли тип данных одинарной точности (S) и соответствующие инструкции: 0 – не реализованы 1 - реализованы	R	1
Processor ID	15:8	Идентификация типа процессора вычислений с плавающей точкой (FPU)	R	0000 0000
Revision	7:0	Номер версии FPU. Это поле позволяет программам различать разные версии одного типа FPU.	R	0000 0000

2.5.2.4.2 Регистр управления и состояния (FCSR, CP1 Control Register 31)

Регистр управления и состояния (Floating Point Control and Status Register - FCSR) – это 32-битный регистр, который управляет работой FPU и содержит информацию о состоянии FPU:

- выбор режима округления для арифметических операций;
- выборочное разрешение исключений при возникновении соответствующих условий исключений;
- управление некоторыми опциями обработки денормализованных чисел;
- сообщает о любых IEEE исключениях произошедших во время последней выполненной команды;
- сообщает о IEEE исключениях произошедших в совокупности выполненных команд;
- показывает код условия, который является результатом команд сравнения.

Доступ к регистру *FCSR* не является привилегированным. Любая программа, которая имеет доступ к FPU (если он разрешён в регистре *Status*), может читать из или записывать в регистр *FCSR*. На Рисунок 2.8 представлен формат *FCSR* регистра, в Таблица 2.8 описаны поля этого регистра.

31 25							24	23	22 -18	17 16 15 14 13 12						11 10 9 8 7					6 5 4 3 2					1 0
FCC							FS	FCC	0	Cause						Enables					Flags					RM
7	6	5	4	3	2	1		0		E	V	Z	O	U	I	V	Z	O	U	I	V	Z	O	U	I	

Рисунок 2.8. Формат регистра FCSR

Таблица 2.4. Описание полей регистра FCSR

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	**Биты**			
FCC	31:25, 23	Коды условий. Эти биты содержат результат выполнения FPU команд сравнения и используются в командах условных переходов и в командах условных перемещений данных. Какой FCC бит используется точно определено в команде перехода или перемещения.	R/W	Не определено
FS	24	Сброс в ноль. Когда FS=1, денормализованный результат операции сбрасывается в ноль вместо появления исключения “Нереализованная операция” (Unimplemented Operation).	R/W	Не определено
-	22:18	Не используются	0	0
Cause	17:12	Биты причины. Эти биты показывают условия исключений, которые возникают во время выполнения арифметических команд. Бит устанавливается в 1, если соответствующая исключительная ситуация появилась во время выполнения команды и устанавливается в 0 в противоположном случае. По значениям этих бит можно определить какая исключительная ситуация вызвана выполнением предыдущей арифметической команды. Значение каждого бита данного поля представлено в Таблице 2.5.	R/W	Не определено

Продолжение Таблица 2.4

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	**Биты**			
Enables	11:7	Биты разрешения соответствующего исключения при возникновении любой из пяти IEEE исключительных ситуаций. Исключение происходит в случае, когда соответствующие бит Cause и бит Enables одновременно установлены либо во время выполнения арифметической операции, либо при перемещении нового значения в регистр FCSR или FEXR и FENR по команде move. Заметьте, что бит E в поле Cause не имеет соответствующего бита в поле Enables, так как исключение “Нереализованная Операция” всегда разрешено. Значение каждого бита данного поля представлено в Таблице 2.5.	R/W	Не определено
Flags	6:2	Флаговые биты. Это поле показывает любые исключительные ситуации, вызванные завершившимися командами со времени последнего программного сброса данного поля. Когда при арифметической операции возникает исключительная ситуация, которая не приводит к FPU исключению (соответствующий бит в Enables сброшен), то соответствующий бит (биты) устанавливается в поле Flags. В других ситуациях поле Flags остаётся без изменений. Арифметические операции, которые приводят к возникновению FPU исключения (бит в Enables установлен), не изменяют состояния бит в поле Flags. У этого поля нет аппаратного сброса, оно должно явно сбрасываться программой. Значение каждого бита данного поля представлено в Таблице 2.5.	R/W	Не определено

Продолжение Таблица 2.4

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	**Биты**			
RM	1:0	Режим округления. Обозначает режим округления, который используется большинством операций в формате с плавающей точкой (некоторые операции используют специфический режим округления). Возможные кодировки этот поля представлены в Таблице 2.6.	R/W	Не определено

Поля FCC, FS, Cause, Enables, Flags и RM в регистрах FCSR, FCCR, FEXR и FENR всегда обозначают правильные состояния. Это означает что, если новое значение поля записывается в FCSR регистр, то это новое значение можно прочитать в соответствующем альтернативном регистре FCCR, FEXR или FENR. И наоборот, записав новое значение поля в альтернативный регистр, его можно прочитать в FCSR регистре.

Таблица 2.5. Описание бит в полях Cause, Enables и Flags

Имя бита	Значение бита
E	Нереализованная операция (Unimplemented Operation) Этот бит существует только в поле Cause
V	Недействительная операция (Invalid Operation)
Z	Деление на ноль (Divide by Zero)
O	Переполнение (Overflow)
U	Потеря значимости (Underflow)
I	Неточность (Inexact)

Таблица 2.6. Описание режимов округления

Кодировка поля RM	Описание
0	RN – округление к ближайшему (round to nearest) Округление результата к ближайшему представимому значению. Когда два представимых значения одинаково близки, результат округляется к значению, чей наименее значащий бит равен 0 (чётный)
1	RTZ – округление к нулю (round towards zero) Округление результата к ближайшему значению, величина (модуль) которого не больше величины результата
2	RP – округление к плюс бесконечности (round towards plus infinity) Округление результата к ближайшему значению не меньшему чем сам результат
3	RM – округление к минус бесконечности (round towards minus infinity) Округление результата к ближайшему значению не большему чем сам результат.

2.5.2.4.3 Регистр кодов условий (FCCR, CP1 Control Register 25)

Регистр кодов условий (Floating Point Condition Codes Register - FCCR) является альтернативным регистром для чтения и записи поля кодов условий FCC, которое также хранятся в регистре FCSR. В отличие от FCSR регистра, в регистре FCCR восемь бит поля FCC являются смежными. На Рисунок 2.9 представлен формат *FCSR* регистра, в Таблице 2.7 описаны поля этого регистра.

31 8	7 0							
0000 0000 0000 0000 0000 0000	FCC							
	7	6	5	4	3	2	1	0

Рисунок 2.9. Формат регистра FCCR

Таблица 2.7. Описание полей регистра FCCR

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	**Биты**			
-	31:8	Не используются	0	0
FCC	7:0	Коды условий. Эти биты содержат результат выполнения FPU команд сравнения и используются в командах условных переходов и в командах условных перемещений данных. Какой FCC бит используется точно определено в команде перехода или перемещения. См. описание поля FCC в регистре *FCSR* в Таблице 2.4	R/W	Не определено

2.5.2.4.4 Регистр исключений (FEXR, CP1 Control Register 26)

Регистр исключений (Floating Point Exceptions Register - FEXR регистр) является альтернативным регистром для чтения и записи полей Cause и Flags, которые также хранятся в регистре FCSR. На Рисунке 2.10 представлен формат *FEXR* регистра, в Таблице 2.8 описаны поля этого регистра.

31 18	17 16 15 14 13 12	11 7	6 5 4 3 2	1 0
0	Cause	0	Flags	0
	E V Z O U I		V Z O U I	

Рисунок 2.10. Формат регистра FEXR

Таблица 2.8. Описание полей регистра FEXR

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	**Биты**			
-	31:18, 11:7, 1:0	Не используются	0	0
Cause	17:12	Биты причины. Эти биты показывают исключительные ситуации, которые возникают во время выполнения FPU арифметических команд. См. описание поля Cause в регистре *FCSR* в Таблице 2.4.	R/W	Не определено
Flags	6:2	Флаговые биты. Это поле показывает любые исключительные ситуации вызванные завершившимися командами со времени последнего программного сброса данного поля. См. описание поля Flags в регистре *FCSR* в Таблице 2.4.	R/W	Не определено

2.5.2.4.5 Регистр разрешения исключений (FENR, CP1 Control Register 28)

Регистр разрешения исключений (Floating Point Enable Register - *FENR регистр*) является альтернативным регистром для чтения и записи полей Enables, FS и RM, которые также хранятся в регистре *FCSR*. На Рисунок 2.11 представлен формат *FENR* регистра, в Таблице 2.9 описаны поля этого регистра.

31 12	11 10 9 8 7					6 3	2	1 0
0000 0000 0000 0000 0000	Enables					0000	FS	RM
	V	Z	O	U	I			

Рисунок 2.11. Формат регистра FENR

Таблица 2.9. Описание полей регистра FENR

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	**Биты**			
0	31:12, 6:3	Не используется	0	0
Enables	11:7	Биты разрешения соответствующего исключения при возникновении любой из пяти IEEE исключительных ситуаций. См. описание поля Enables в регистре *FCSR* в Таблице 2.4.	R/W	Не определено
FS	2	Сброс в ноль. Когда FS=1, денормализованный результат операции сбрасывается в ноль вместо появления исключения “Нереализованная операция” (Unimplemented Operation). См. описание поля FS в регистре *FCSR* в Таблице 2.4.	R/W	Не определено
RM	1:0	Режим округления. Обозначает режим округления, который используется большинством операций с плавающей точкой. См. описание поля RM в регистре *FCSR* в Таблице 2.4.	R/W	Не определено

2.5.3 Исключения FPU

2.5.3.1 Формирование исключения

При возникновении исключения команда, вызвавшая его, а также все последующие команды не выполняются и не изменяют содержимого регистров FGR. При необходимости, после обработки исключения выполнение прерванного потока команд может быть возобновлено.

В поле *Cause* содержаться признаки исключений. Оно обновляются при выполнении каждой арифметической операции в формате с плавающей точкой. Признак устанавливается в 1, если возникает соответствующее условие исключения, иначе он устанавливается в 0.

Исключение возникает каждый раз, если одновременно признак поля *Cause* и соответствующий ему бит *Enable* установлены в 1. Это происходит или во время выполнения операции в формате с плавающей точкой или, при передаче данных в регистр FCSR по команде move. Бита *Enable* для Unimplemented Operation не существует, то есть исключение по этому условию возникает всегда.

Содержимое поля *Cause* используется в обработчике исключения. Перед выходом из обработчика исключения по операции в формате с плавающей точкой, или перед установкой бит поля *Cause* по команде move, необходимо сначала обнулить соответствующие биты *Enable*, для того, чтобы предотвратить повторное возникновение исключениия.

Пользовательским программам не доступны биты поля *Cause*. Если эта информация необходима этим программам, то она должна быть доступна им другими путями, а не через регистр *Status*.

Если операция в формате с плавающей точкой устанавливает только неразрешенные биты поля *Cause*, то исключения не происходит, и записывается результат, определяемый стандартом IEEE (см. Таблица 2.10). Когда операция в формате с плавающей точкой не вызывает исключения, программа может контролировать условия исключения, считывая содержимое поля *Cause*.

Поле *Flag* – совокупная накопленная информация по условиям исключений. Команды, которые вызывают исключения, не обновляют биты поля *Flag*. Биты поля *Flag* устанавливаются в 1, если соответствующее условие исключение возникает, иначе биты остаются без изменения. Бита для условия исключения типа Unimplemented Operation в этом поле не предусмотрено. В результате выполнения операции в формате с плавающей точкой биты поля *Flag* никогда не сбрасываются, но могут быть установлены или сброшены (обнулены) при записи данных в регистр FCSR по команде move.

2.5.3.2 Условие исключений

В этом пункте описаны следующие пять условий исключения, определенных стандартом ANSI/IEEE Standard 754-1985:

- исключение по недопустимой операции (Invalid Operation Exception);
- исключение при делении на ноль (Division By Zero Exception);
- исключение по ложному переполнению (Underflow Exception);
- исключение по переполнению (Overflow Exception);
- неточное исключение (Inexact Exception).

Этот пункт также содержит описание исключения по нереализованной операции (unimplemented operation). Оно используется для сообщения о необходимости программной эмуляции команды. Обычно арифметическая операция IEEE может вызывать только одно условие исключения. Единственный случай, когда два исключения могут происходить в то же самое время, это Inexact With Overflow и Inexact With Underflow.

Под управлением программы, условие исключения IEEE может вызывать прерывание (trap) процессора или не вызывать его. Стандарт IEEE определяет результат операции при возникновении условии исключения для случая, когда прерывание процессора по этому исключению не разрешено. Для этого случая результаты операций приведены в Таблица 2.10. При переполнении результат операции зависит от режима округления.

Таблица 2.10. Результаты операций при исключениях

Бит	Описание	Результат операции
V	Invalid Operation	Quiet NaN
Z	Divide by Zero	Properly signed infinity
U	Underflow	Округленный результат (Rounded result)
I	Inexact	Округленный результат. Если это исключение вызвано переполнением (Overflow) при неразрешенном прерывании, то формируется результат с переполнением.
O	Overflow	Зависит от режима округления: 0 (RN) – infinity со знаком промежуточного результата; 1 (RZ) – format's infinity со знаком промежуточного результата; 2 (RP) – при положительном переполнении – positive infinity. При отрицательном переполнении - format's most negative infinity; 3 (RM) - при положительном переполнении – format's largest finite number. При отрицательном переполнении – minus infinity.

2.5.3.3 Исключение по недопустимой операции

Это исключение возникает, если один или оба операнда недопустим для выполняемой операции.

Недопустимые операции:

- Один или оба операнда являются NaN (за исключением не арифметических команд MOV.fmt, MOVT.fmt, MOVF.fmt, MOVN.fmt, и MOVZ. fmt);
- Сложение или вычитание: вычитание бесконечных величин, таких как (+∞) + (-∞) или (-∞) - (-∞);
- Умножение: 0 * ∞, с любыми знаками;
- Деление: 0/0 или ∞ / ∞, с любыми знаками;
- Квадратный корень: операнд меньше чем 0 (-0 является допустимым значением);
- Преобразование числа в формате с плавающей запятой к формату с фиксированной запятой, если возникает переполнение, или значение операнда равное infinity или NaN препятствуют точному представлению данных в необходимом формате;
- Некоторые операции сравнения, в которых один или оба операнда имеют значение QNaN.

2.5.3.4 Исключение при делении на ноль

Это исключение возникает, если делитель равен нулю, а делимое является конечным числом, отличным от нуля. Результат, когда не возникает прерывания, равен бесконечности. Деление (0/0) и (∞/0) не приводят к исключению. При делении (0/0) возникает исключение по недопустимой операции. Результат (∞/0) – бесконечность со знаком.

2.5.3.5 Исключение по ложному переполнению(потеря значимости)

Два связанных события могут повлиять на возникновение ложного переполнения:

- близость результата к нулю (tininess): создание бесконечно малого результата отличного от нуля находящегося в промежутке между $\pm 2^{E\_min}$, который из-за своей малой величины может вызывать впоследствии какое либо другое исключение, например как переполнение при делении;
- потеря точности: экстраординарная потеря точности во время аппроксимации таких малых чисел ненормированными числами.

Стандарт IEEE определяет, что «близость результата к нулю» может быть обнаружена в любой из следующих моментов времени:

- после округления, когда не нулевой результат получен из предположения неограниченности диапазона экспоненты и находится строго между $\pm 2^{E\_min;}$
- пред округлением, когда не нулевой результат получен из предположения неограниченности, как диапазона экспоненты, так и точности, и находится строго между $\pm 2^{E\_min;}$

В FPU близость результата к нулю обнаруживается после округления.

Стандарт IEEE определяет, что потеря точности может быть получена в результате любого из следующих условий:

- нарушение нормализации (denormalization), когда полученный результат отличается от вычисленного без ограничений диапазона экспоненты;
- неточный результат (inexact result), когда полученный результат отличается от вычисленного без ограничений диапазона экспоненты и точности.

В FPU потеря точности формируется, если получен неточный результат.

Если прерывание процессора при ложном переполнении не разрешено, признак U вырабатывается, когда обнаруживается одновременно и близость к нулю и потеря точности. При этом, результат может быть нулевым, ненормализованным или 2^{E_min}.

Если прерывание процессора при ложном переполнении разрешено, признак U вырабатывается, когда обнаруживается только близость к нулю, в не зависимости от потери точности.

2.5.3.6 Исключение при переполнении

Это исключение возникает, когда величина округленного результата в формате с плавающей запятой (где диапазон экспоненты не ограничен) больше, чем наибольшее конечное число результирующего формата (destination format's largest finite number).

Если прерывание процессора при переполнении не разрешено, результат определяется режимом округления и знаком промежуточного результата.

2.5.3.7 Неточное исключение

Неточное исключение возникает, если:

- округленный результат операции не является точным;
- округленный результат операции вызывает переполнение, а прерывание по переполнению не разрешено.

2.5.3.8 Исключение по нереализованной операции

Это исключения не регламентировано стандартом IEEE. Операции, которые не полностью поддерживаются аппаратурой, вызывают исключение, для того, чтобы программное обеспечение могло выполнить соответствующую операцию.

Для этого условия исключения не предусмотрено разрешающего бита, то есть прерывание процессора возникает всегда. После того, как соответствующее эмулирование будет выполнено, прерванная программа возобновляется.

2.5.4 Время выполнения команд FPU

Время выполнения команд в формате с плавающей точкой приведено в Таблице 2.11.

Таблица 2.11. Время выполнения команд FPU

Команда	Время выполнения, такты
BC1F, BC1T, FLOOR, ROUND, TRUNC	1
CFC1, CTC1, MFC1, MOVF	1
CVT.S, CVT.D, CEIL	2
ABS, ADD, SUB, MULL, NEG	3
SQRT.S/SQRT.D	6/15
DIV.S/DIV.D	11/16

2.6 Устройство управления памятью (MMU)

2.6.1 Введение

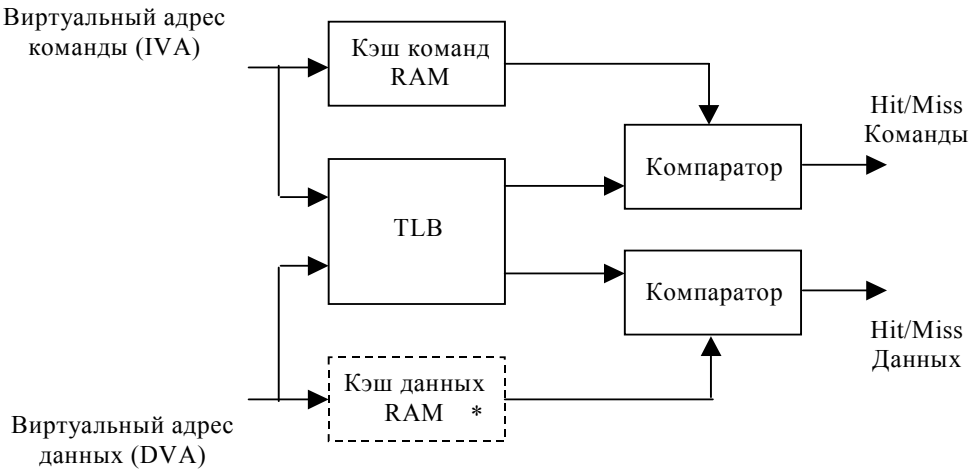
Процессорное ядро содержит устройство управления памятью (MMU), реализующее интерфейс между устройством исполнения и контроллером кэш. MMU преобразует виртуальный адрес в физический прежде, чем посылает запрос контроллеру кэш для сравнения тэга или блоку шинного интерфейса для доступа к внешнему запоминающему устройству. Это преобразование является очень полезным свойством функционирования операционных систем при управлении физической памятью таким образом, чтобы в ней размещались несколько процессов, активных в одной и той же области памяти, и может быть даже на одном виртуальном адресе, но обязательно в различных областях физической памяти. Другие свойства MMU - защита зон памяти и определение протокола кэш.

MMU может выполнять преобразование адресов в двух режимах: в режиме TLB и в режиме FM. Режим преобразования определяется битом FM регистра CSR.

В режиме TLB используется полностью ассоциативная таблица преобразования адресов (TLB), имеющая 16 парных строк (entries). Во время преобразования осуществляется поиск соответствия по TLB. Если искомая строка отсутствует, генерируется прерывание.

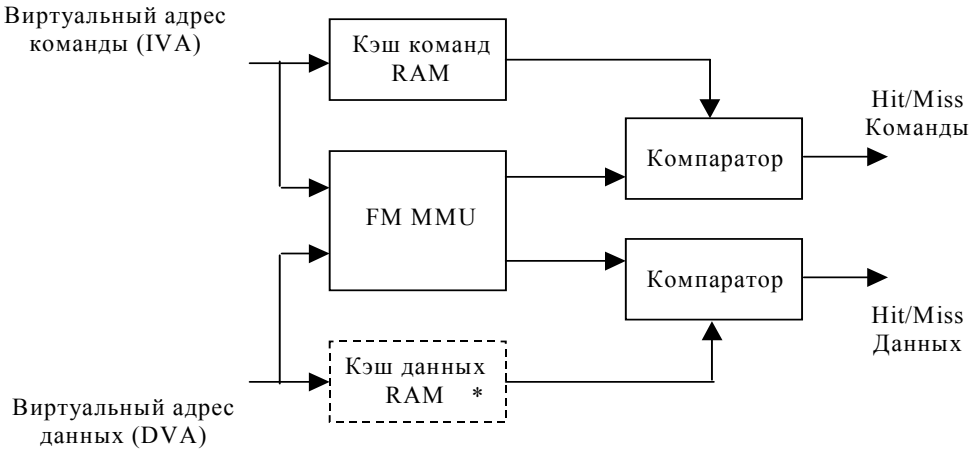
В режиме FM (Fixed Mapped) работа MMU основана на простом алгоритме, обеспечивающем преобразование виртуального адреса в физический посредством механизма фиксированного отображения. Правила преобразования отличаются для различных областей виртуального адресного пространства (useg/kuseg, kseg0, kseg1, kseg2, kseg3).

На Рисунок 2.12 показано, взаимодействие MMU с процедурой доступа к кэш в режиме TLB, а на Рисунок 2.13 – в режиме FM.



* - Кэш данных в данной реализации отсутствует

Рисунок 2.12



* - Кэш данных в данной реализации отсутствует

Рисунок 2.13

2.6.2 Режимы работы.

Процессорное ядро поддерживает два режима работы:

- Режим User (непривилегированный режим)
- Режим Kernel (привилегированный режим)

Режим User в основном используется для прикладных программ. Режим Kernel обычно используется для обработки исключительных ситуаций и привилегированных функций операционной системы, включая управление сопроцессором CP0 и доступ к устройствам ввода-вывода.

Преобразования, выполняемые MMU, зависят от режима работы процессора.

2.6.2.1 Виртуальные сегменты памяти

Виртуальные сегменты памяти, на которые делится адресное пространство, различаются в зависимости от режима работы процессора. На Рисунок 2.14 показана сегментация для 4 Гбайт (2^{32} байт) виртуального адресного пространства, адресуемого 32-разрядным виртуальным адресом для обоих режимов работы.

Ядро входит в режим Kernel после аппаратного сброса или когда происходит исключение. В режиме Kernel программное обеспечение имеет доступ к полному адресному пространству и ко всем регистрам CP0. В режиме User доступ ограничен подмножеством виртуального адресного пространства (0x0000_0000 - 0x7FFF_FFFF) и запрещен доступ к функциям CP0. В режиме User недоступны виртуальные адреса 0x8000_0000 - 0xFFFF_FFFF и обращение к ним вызывает исключение.

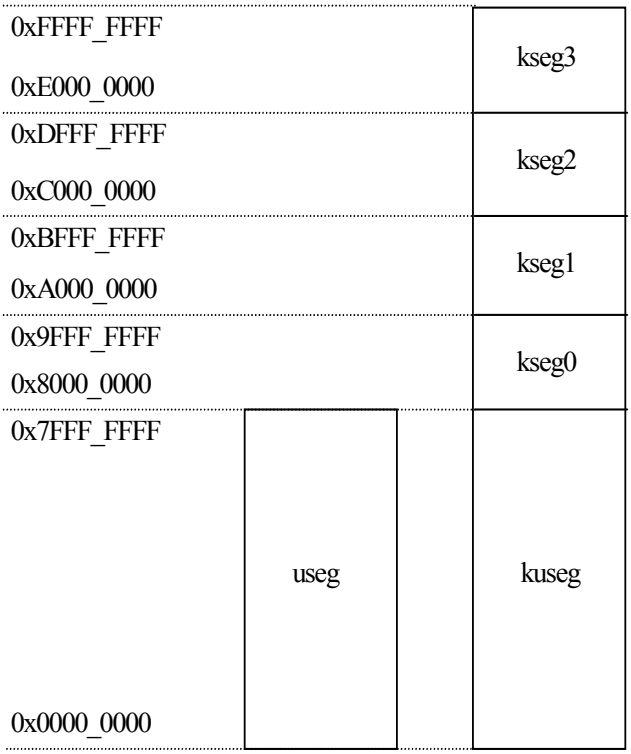


Рисунок 2.14. Карта виртуальной памяти для режимов User и Kernel

Каждый из сегментов, показанных на Рисунок 2.14, является либо отображаемым (mapped), либо неотображаемым (unmapped). Различие объясняется в следующих двух разделах.

2.6.2.1.1 Неотображаемые сегменты

В неотображаемом сегменте механизмы TLB или FM для преобразования виртуального адреса в физический адрес не используются. Особенно важно иметь неотображаемые сегменты памяти после аппаратного сброса, потому что TLB еще не запрограммировано и не может осуществлять преобразования.

Для неотображаемых сегментов преобразование виртуального адреса в физический является фиксированным.

Все неотображаемые сегменты, за исключением kseg0, никогда не кэшируемы. Кэшируемость kseg0 определяется полем K0 регистра Config CP0.

2.6.2.1.2 Отображаемые сегменты

В отображаемом сегменте для преобразования виртуального адреса в физический адрес используются TLB или FM.

В режиме TLB преобразование отображаемых сегментов имеет постраничную основу. При преобразовании выявляется информация о кэшируемости страницы, а также атрибуты защиты, относящиеся к странице.

Для режима FM отображаемые сегменты имеют закрепленное преобразование виртуального адреса в физический. Кэшируемость сегмента определяется значениями полей

K23 и KU регистра Config CP0. При FM-преобразовании невозможна защита сегментов от записи.

2.6.2.2 Режим User

В режиме User доступно однородное виртуальное адресное пространство размером 2 Гбайт (2^{31} байт), называемое сегментом пользователя.

На Рисунок 2.15 показано размещение виртуального адресного пространства режима User.

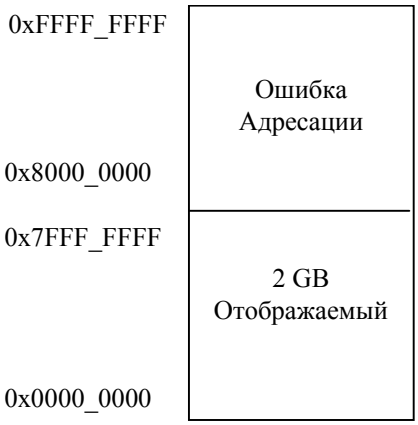


Рисунок 2.15

Сегмент потребителя начинается с адреса 0x0000_0000 и заканчивается адресом 0x7FFF_FFFF. Обращения по всем остальным адресам вызывают прерывания по ошибке адресации.
Процессор находится в режиме User, если в регистре Status CP0 установлены следующие значения разрядов:

- UM = 1
- EXL = 0
- ERL = 0

В Таблица 2.12 приводятся характеристики сегмента useg режима User.

Таблица 2.12

Адрес	Регистр Состояния			Имя сегмента	Диапазон адресов	Размер сегмента
	EXL	ERL	UM			
A(31)=0	0	0	1	useg	0x0000_0000 → 0x7FFF_FFFF	2GB (2^{31} байт)

Для всех допустимых виртуальных адресов режима User старший значащий бит адреса равен нулю, поскольку в режиме User допустимо обращение только к нижней половине карты виртуальной памяти. Любая попытка обращения по адресу со старшим битом, равным 1, в режиме User вызывает прерывание по ошибке адресации.

В режиме TLB виртуальный адрес перед преобразованием расширяется содержимым 8-разрядного поля ASID, образуя уникальный виртуальный адрес. Кэшируемость ссылки для страницы в этом режиме определяется установкой определенных бит строки TLB.

В режиме FM, область виртуальных адресов 0x0000_0000-0x7FFF_FFFF преобразуется в область физических адресов 0x4000_0000-0xBFFF_FFFF. Кэшируемость задается полем KU регистра Config CP0.

2.6.2.3 Режим Kernel

Процессор находится в режиме Kernel, когда регистр Status CP0 содержит хотя бы одно из следующих значений:

- UM = 0
- ERL = 1
- EXL = 1

Когда обнаруживается исключение, биты EXL или ERL устанавливаются, и процессор входит в режим Kernel. При завершении процедуры обработки исключения обычно выполняется команда возвращения из исключения (ERET). Команда ERET осуществляет переход по PC исключения, очищает ERL и EXL (если ERL=0). В результате возможен возврат процессора в режим User.

Виртуальное адресное пространство режима Kernel разделено на области в соответствии со значением старших битов виртуального адреса, как показано на Рисунок 2.16. Кроме того, в Таблица 2.13 содержатся характеристики сегментов режима Kernel.

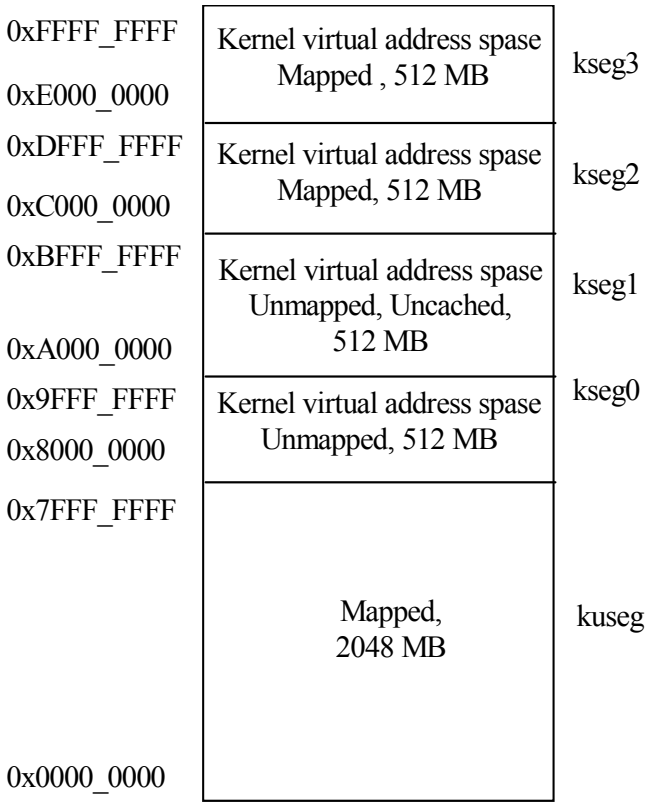


Рисунок 2.16

Таблица 2.13

Адрес	Регистр Состояния			Имя сегмента	Диапазон адресов	Размер сегмента
	EXL	ERL	UM			
A(31)=0	UM = 0 или EXL=1 или ERL=1			kuseg	0x0000_0000 → 0x7FFF_FFFF	2 GB (2^{31})
A(31:29)=100_2				kseg0	0x8000_0000 → 0x9FFF_FFFF	512 MB (2^{29})
A(31:29)=101_2				kseg1	0xA000_0000 → 0xBFFF_FFFF	512 MB (2^{29})
A(31:29)=110_2				kseg2	0xC000_0000 → 0xDFFF_FFFF	512 MB (2^{29})
A(31:29)=111_2				kseg3	0xE000_0000 → 0xFFFF_FFFF	512 MB (2^{29})

2.6.2.3.1 Режим Kernel, Пространство пользователя (kuseg)

Если старший значащий бит виртуального адреса A[31]=0, то выбирается виртуальное адресное пространство kuseg объемом 2 Гбайт, отображенное на адреса 0x0000_0000 - 0x7FFF_FFFF.

При ERL=0 в режиме TLB виртуальный адрес расширяется 8-битным значением поля ASID для образования уникального виртуального адреса. Кэшируемость определяется полем C строки TLB.

При ERL=0 в режиме FM, область виртуальных адресов 0x0000_0000-0x7FFF_FFFF преобразуется в область физических адресов 0x4000_0000-0xBFFF_FFFF. Кэшируемость задается полем KU регистра Config CP0.

При ERL = 1 в режимах TLB и FM, область адресов пользователя становится неотображаемым и некэшируемым адресным пространством. Виртуальный адрес kuseg соответствует тому же физическому адресу и не включает поле ASID. То есть, область виртуальных адресов kuseg соответствует области физических адресов 0x0000_0000-0x7FFF_FFFF.

2.6.2.3.2 Режим Kernel, пространство 0 режима Kernel (kseg0).

Если в режиме Kernel три старших бита виртуального адреса равны 100_2, выбирается виртуальное адресное пространство kseg0. Это область размером 2^{29} байт (512 MB), которая расположена внутри границ, определяемых адресами 0x8000_0000 и 0x9FFF_FFFF.

Вне зависимости от состояния бита ERL и режима работы ссылки к kseg0 не отображаются, а физический адрес получается вычитанием 0x8000_0000 из виртуального адреса. Кэшируемость сегмента kseg0 определяется значением поля K0 регистра Config CP0.

2.6.2.3.3 Режим Kernel, пространство 1 режима Kernel (kseg1)

Если в режиме Kernel три старших бита виртуального адреса равны 101_2, выбирается виртуальное адресное пространство kseg1. Это область размером 2^{29} байт (512 MB), которая расположена внутри границ, определяемых адресами 0xA000_0000 и 0xBFFF_FFFF.

Вне зависимости от состояния бита ERL и режима работы ссылки к kseg1 не отображаются, а физический адрес получается вычитанием 0xA000_0000 из виртуального адреса.

2.6.2.3.4 Режим Kernel, пространство 2 режима Kernel (kseg2)

Если в режиме Kernel три старших бита виртуального адреса равны 110_2, выбирается виртуальное адресное пространство kseg2.

В режиме TLB вне зависимости от состояния бита ERL это виртуальное пространство отображается через TLB и его кэшируемость определяется полем C строки TLB.

В режиме FM вне зависимости от состояния бита ERL это виртуальное пространство зафиксировано в физических адресах 0xC000_0000 - 0xDFFF_FFFF и его кэшируемость определяется полем K23 Регистра Config CP0.

2.6.2.3.5 Режим Kernel, пространство 3 режима Kernel (kseg3)

Если в режиме Kernel три старших бита виртуального адреса равны 111_2, выбирается 32-разрядное виртуальное адресное пространство kseg3.

В режиме TLB вне зависимости от состояния бита ERL это пространство отображается через TLB и его кэшируемость определяется полем C строки TLB.

В режиме FM вне зависимости от состояния бита ERL это виртуальное пространство зафиксировано в физических адресах 0xE000_0000 - 0xFFFF_FFFF и его кэшируемость определяется полем K23 регистра Config.

2.6.3 Буфер быстрого преобразования адреса (TLB)

В этой главе описывается управление памятью с помощью буфера быстрого преобразования адреса (TLB), которое осуществляется в режиме TLB.

В режиме TLB реализуется полностью ассоциативный буфер быстрого преобразования адреса (TLB), содержащий 16 двойных строк, позволяющих отображать 32 виртуальных страницы в соответствующие физические адреса. TLB организовано в виде 16 парных строк – четных и нечетных, содержащих адреса страниц размером от 4 Кбайт до 16 Мбайт, которые хранятся в 4 Гбайтном физическом адресном пространстве. Задача TLB состоит в преобразовании виртуальных адресов и их соответствующего идентификатора адресного пространства (ASID) в физический адрес памяти. Преобразование

выполняется путем сравнения старших разрядов виртуального адреса (вместе с битами поля ASID) с каждой из строк тэговой порции TLB и иначе называется поиском соответствия по TLB (поиском соответствия тэга одной из строк виртуальному адресу на входе TLB).

Буфер TLB организован в виде страничных пар для минимизации общего количества хранящейся информации. Каждая строка тэговой порции соответствует двум физическим строкам данных – строке четных страниц и строке нечетных страниц. Самый старший разряд виртуального адреса, не участвующий в сравнении тэгов, определяет какая строка из двух строк данных используется. Поскольку размер страницы может варьироваться для каждой пары страниц, определение адресных разрядов, участвующих в сравнении и разряда, задающего четность страницы, должно осуществляться динамически при поиске по TLB.

На Рисунок 2.17 показано содержание одной из 16 двойных строк TLB.

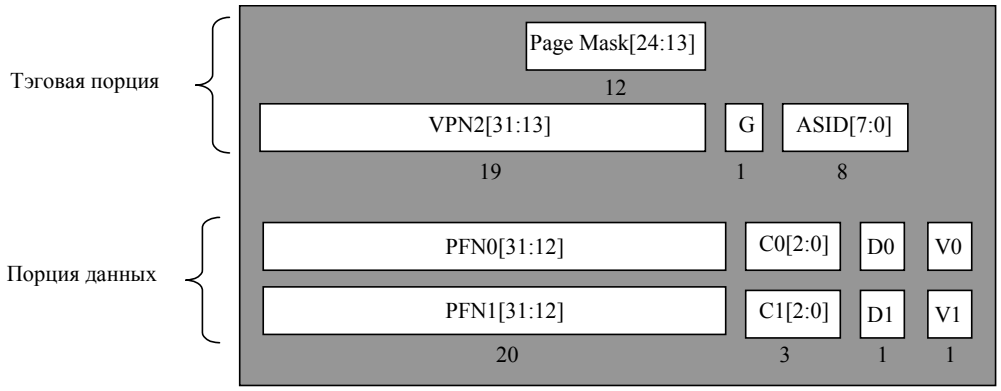


Рисунок 2.17

Описание полей строки TLB приведены в Таблица 2.14.

Таблица 2.14

<table>
<tr><th>Название поля</th><th>Описание</th></tr>
<tr><td>Page Mask[24:13]</td><td>Значение маски размера страницы. Определяет размер страницы маскировкой соответствующих разрядов VPN2, и тем самым исключением их из рассмотрения. Также используется для задания адресного разряда, определяющего четность страницы (PFN0-PFN1). См. следующую таблицу:
<table>
<tr><th>Page Mask[11:0]</th><th>Размер страницы</th><th>Бит определения четности</th></tr>
<tr><td>0000_0000_0000</td><td>4 КБ</td><td>VAddr[12]</td></tr>
<tr><td>0000_0000_0011</td><td>16 КБ</td><td>VAddr[14]</td></tr>
<tr><td>0000_0000_1111</td><td>64 КБ</td><td>VAddr[16]</td></tr>
<tr><td>0000_0011_1111</td><td>256 КБ</td><td>VAddr[18]</td></tr>
<tr><td>0000_1111_1111</td><td>1 МБ</td><td>VAddr[20]</td></tr>
<tr><td>0011_1111_1111</td><td>4 МБ</td><td>VAddr[22]</td></tr>
<tr><td>1111_1111_1111</td><td>16 МБ</td><td>VAddr[24]</td></tr>
</table>
В столбце Page Mask приведены все возможные значения Page Mask. Поскольку каждая пара битов этого поля всегда имеет одинаковое значение, физическая строка в TLB содержит сокращенную версию Page Mask, содержащую только 6 бит. Однако для программы это значение всегда преобразуется в 12-битное.
Следует иметь в виду, что при кэшируемых ссылках, страницы размером 4 Кбайт использовать нельзя.</td></tr>
<tr><td>VPN2[31:13]</td><td>Виртуальный номер страницы, поделенный на 2. Данное поле содержит старшие разряды виртуального номера страницы. Виртуальный номер разделен на 2 потому, что он соответствует паре страниц TLB. Разряды 31:25 всегда участвуют в сравнении. Участие в сравнении разрядов 24:13 зависит от размера страницы, задаваемого полем Page Mask.</td></tr>
<tr><td>G</td><td>Бит глобальности. Если он установлен, данная строка является глобальной для всех процессов и подпроцессов, и таким образом, поле ASID исключается из рассмотрения.</td></tr>
<tr><td>ASID[7:0]</td><td>Идентификатор адресного пространства. Определяет процесс или подпроцесс, с которым ассоциируется данная строка TLB.</td></tr>
</table>

Продолжение Таблица 2.14.

<table>
<tr><th>Название поля</th><th>Описание</th></tr>
<tr><td>PFN0[31:12],
PFN1[31:12]</td><td>Физический номер кадра. Задает старшие разряды физического адреса. Для страниц размером более 4 Кбайт используется подмножество этого поля.</td></tr>
<tr><td>C0[2:0],
C1[2:0]</td><td>Кэшируемость. Содержит закодированное значение атрибута кэшируемости и определяет должна ли страница помещаться в кэш или нет. Поле кодируется следующим образом:
<table>
<tr><th>C[2:0]</th><th>Атрибуты когерентности</th></tr>
<tr><td>000</td><td>При записи преобразуется в код 011</td></tr>
<tr><td>001</td><td>При записи преобразуется в код 011</td></tr>
<tr><td>010</td><td>Некэшируемая страница</td></tr>
<tr><td>011</td><td>Кэшируемая страница</td></tr>
<tr><td>100</td><td>При записи преобразуется в код 011</td></tr>
<tr><td>101</td><td>При записи преобразуется в код 011</td></tr>
<tr><td>110</td><td>При записи преобразуется в код 011</td></tr>
<tr><td>111</td><td>При записи преобразуется в код 010</td></tr>
</table></td></tr>
<tr><td>D0,
D1</td><td>"Dirty" (Грязная страница) – бит разрешения записи. Показывает, что в страницу была сделана запись и/или разрешена запись в данную страницу. Если этот бит установлен, разрешены операции сохранения в данной странице. Если не установлен, сохранения в данной странице будут вызывать исключения модификации.</td></tr>
<tr><td>V0,
V1</td><td>Бит валидности. Показывает, что данная строка TLB и, соответственно, отображение виртуальной страницы, действительны. Если этот бит установлен, то обращения к данной странице разрешены. Если не установлен, то обращения к странице будут вызывать исключения TLB (TLB invalid).</td></tr>
</table>

Для заполнения строки TLB используются команды TLBWI и TLBWR (см. документ “Процессорное ядро RISCore32. Система команд ”). Перед запуском этих команд нужно обновить некоторые регистры CP0, записав в них значения, которые будут затем помещены в строку TLB.

- Значение Page Mask задается в регистре Page Mask CP0.
- Значения VPN2 и ASID задаются в регистре EntryHi CP0.
- Значения PFN0, C0, D0, V0 и G задаются в регистре EntryLo0 CP0.
- Значения PFN1, C1, D1, V1 и G задаются в регистре EntryLo1 CP0.

Биты глобальности G входят в оба регистра EntryLo0 и EntryLo1. Бит G строки TLB является результатом логической операции "И", проведенной над битами глобальности из EntryLo0 и EntryLo1. Более подробно эти регистры описаны в разделе 2.7 “Регистры CP0”.

Наличие идентификатора адресного пространства (ASID) дает возможность уменьшить частоту попаданий при поисках по TLB на контекстной основе. Это определяет возможность одновременного существования нескольких процессов как в TLB, так и в кэш команд. Значение ASID хранится в регистре EntryHi и сравнивается со значением ASID каждой строки.

2.6.4 Преобразование виртуального адреса в физический в режиме TLB.

Преобразование виртуального адреса в физический начинается со сравнения полученного виртуального адреса с виртуальными адресами, хранящимися в TLB. Соответствие имеет место, если виртуальный номер страницы (VPN) адреса совпадает с полем VPN строки TLB с учетом маски, хранящейся в этой строке, а также выполняется одно из двух условий:

- Установлен бит глобальности (G) для четных и нечетных страниц в строке TLB;
- Поле ASID виртуального адреса совпадает с полем ASID строки TLB.

Это соответствие называется попаданием TLB. Если не имеется ни одного соответствия, возникает исключение промаха TLB и программному обеспечению дается возможность пополнить TLB из расположенной в памяти таблицы страниц виртуальных /физических адресов. На Рисунок 2.18 показана логика преобразования виртуального адреса в физический.

На этом рисунке виртуальный адрес расширяется 8-разрядным идентификатором адресного пространства (ASID), который уменьшает частоту попаданий при просмотрах TLB на контекстной основе. Это 8-разрядное поле ASID содержит номер, присвоенный процессу, и хранится в регистре EntryHi CP0.

1. Виртуальный адрес (VA), представленный виртуальным номером страницы (VPN), сравнивается с тэгом из строки TLB (VPN2) с учетом маски (PageMask).

2. Если имеется соответствие, номер страничного кадра (PFN0 или PFN1, в зависимости от значения бита четности – самого старшего бита, не участвующего в сравнении) извлекается и помещается в старшие разряды физического адреса (PA)

3. В младшие разряды физического адреса помещается смещение (Offset), не участвующее в сравнении.

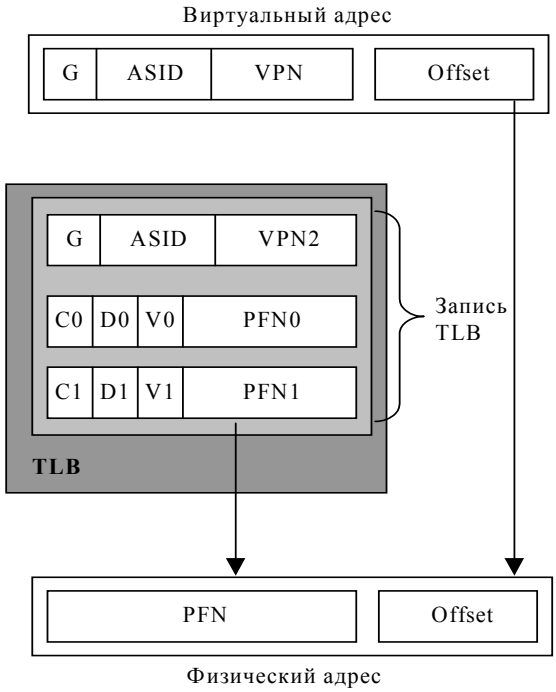


Рисунок 2.18

Когда происходит совпадение виртуальных адресов при поиске по TLB, физический номер кадра (PFN) извлекается из соответствующей физической порции строки TLB и дополняется смещением, взятым из виртуального адреса, формируя, таким образом, физический адрес. Смещение представляет собой адрес в пределах пространства страничного кадра. Как показано на рисунке, смещение не пропускается через TLB.

На Рисунок 2.19 показана блок-схема процесса преобразования адреса. В верхней части рисунка показан виртуальный адрес для страницы размером 4 Кбайт. Ширина поля смещения определяется размером страницы.

В нижней части рисунка показан виртуальный адрес для страницы размером 16 Мбайт.

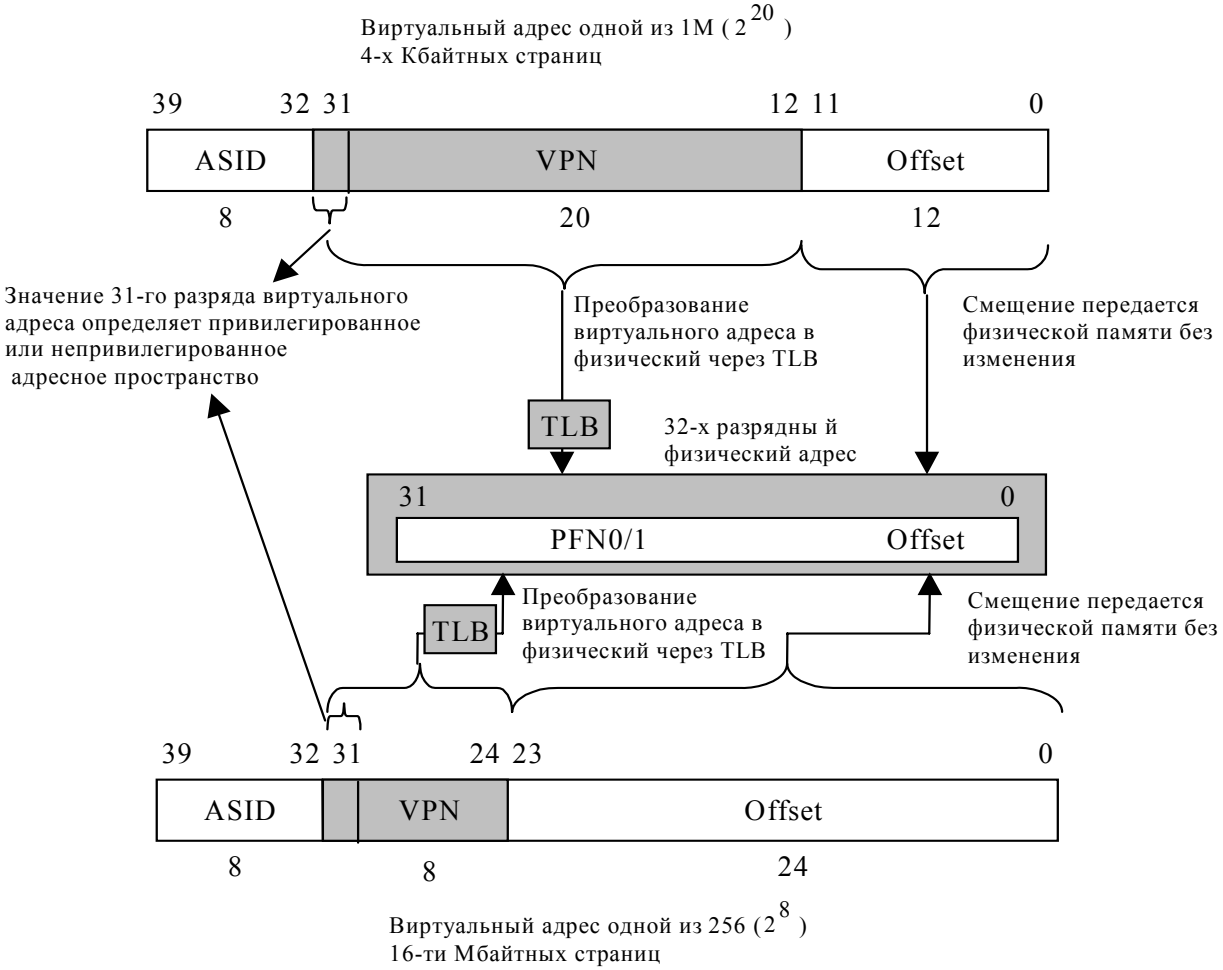


Рисунок 2.19

2.6.4.1 Попадания (hits), промахи (misses), и множественные попадания (multiple matches)

Каждая строка TLB содержит тэг и два поля данных. Если найдено соответствие, старшие разряды виртуального адреса заменяются физическим номером кадра (PFN), хранящимся в соответствующей строке массива данных TLB. Способ разбиения памяти при отображении определяется в терминах TLB-страниц. TLB поддерживает страницы различных размеров в пределах от 4 КБ до 16 МБ с шагом по степеням 4. Если соответствие найдено, но строка является запрещенной (т.е., бит V в поле данных равен 0), вырабатывается исключение TLB Invalid.

Если соответствие не найдено, возникает исключение TLB Refill, и программное обеспечение пополняет TLB из таблицы страниц, находящейся в памяти. На Рисунок 2.20 показан алгоритм преобразования и условия возникновения исключений TLB.

Программное обеспечение может делать записи в конкретные строки TLB или использовать аппаратный механизм записи в случайно выбранные строки. Регистр Random определяет, в какую строку будет сделана запись командой TLBWR. Этот регистр декрементируется на каждом такте продвижения конвейера, возвращаясь к максимальному значению после достижения величины, равной значению регистра Wired. Таким обра-

зом, строки TLB, чей номер меньше значения регистра Wired, не затрагиваются командой TLBWR, что позволяет зарезервировать TLB-отображения первостепенной важности.

В режиме TLB также реализован механизм сравнения при записи с целью предотвращения возникновения нескольких соответствий (множественных попаданий). Работает он следующим образом. При выполнении операции записи в TLB, поле VPN2 сравнивается с одноименными полями всех строк TLB. Если будет найдено соответствие, возникнет аппаратно обрабатываемое исключение, которое установит бит TS регистра Status CP0 и прервет эту операцию. Подробно исключения описаны в п. 2.7. В каждой строке TLB имеется скрытый бит, обнуляемый при аппаратном сбросе. Устанавливается этот бит при записи в данную строку, разрешая просмотр этой строки при поисках соответствий. Поэтому непроинициализированные строки не вызывают неадекватные преобразования адресов.

Замечание: этот скрытый бит инициализации приводит все строки TLB к запрещенному состоянию после аппаратного сброса, что делает ненужной процедуру очистки (flush) TLB. Но для совместимости с другими MIPS – процессорами рекомендуется заполнять значения тэгов уникальными величинами и обнулять бит валидности (V).

Очистить строку TLB (вывести ее из рассмотрения при поиске) можно, записав в нее значение с неотображаемым через TLB адресом.

Смена размера маски или других переменных строки TLB не приводит к исключению, если она не вводит в противоречие данной строки с другими строками. Например, увеличение размера страницы расширением маски в одной строке TLB может привести к перекрытию данной страницы с другими страницами TLB.

2.6.4.2 Размеры страниц и алгоритм замещения

Для управления общим количеством отображаемого адресного пространства и характеристиками замещения в различных областях памяти ядро обеспечивает два механизма. Первый заключается в том, что размер страницы может быть задан относительно каждой строки TLB, что позволяет отображать страницы размером от 4 Кбайт до 16 Мбайт (по степеням 4). В регистр Page Mask CP0 загружается требуемый размер страницы, который при выполнении операции записи попадает в очередную строку TLB. Таким образом, операционная система может задавать отображения особых назначений. Например, характерный кадровый буфер (frame buffer) может быть отображен на память всего одной строкой TLB.

Второй механизм управляет замещением, когда возникает промах при просмотре TLB. Для выбора строки TLB, в которую будет записано новое отображение, в процессорном ядре предусмотрен алгоритм случайного замещения. Но существует также способ программно предотвратить случайное замещение зарезервированных отображений, количество которых определяется значением регистра Wired CP0. (см. также п. 2.8.3.6).

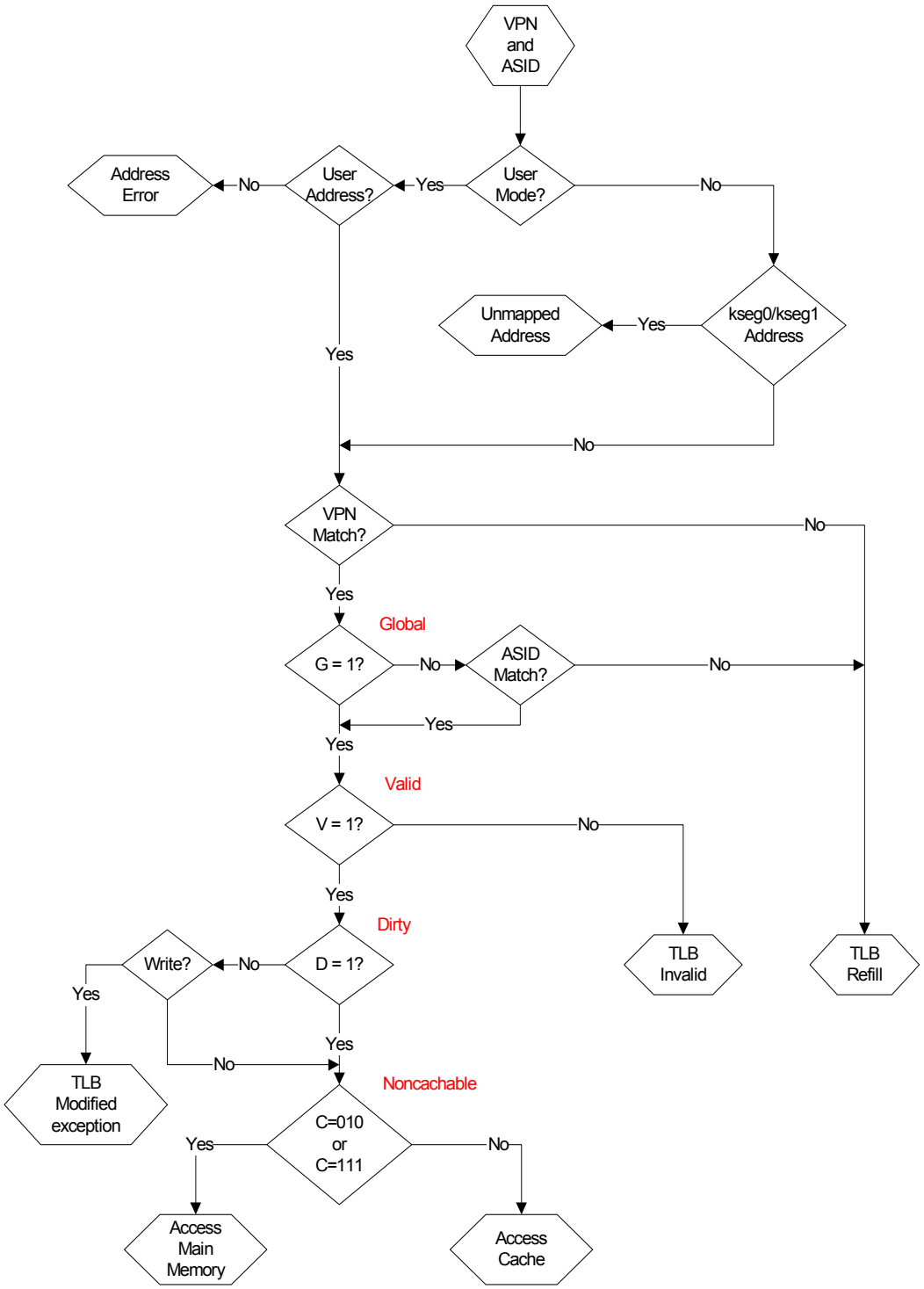


Рисунок 2.20. Алгоритм преобразования адреса через TLB.

2.7 Исключения

Процессорное ядро способно принимать исключения от ряда источников, в том числе промах буфера преобразования адресов (TLB), арифметические переполнение, прерывание ввода-вывода, и системные вызовы. Обнаружив одно из этих исключений, CPU приостанавливает нормальную последовательность исполнения команд и процессор входит в режим Kernel.

В режиме Kernel ядро отключает прерывания и вынуждает процессор запустить программу обработчика исключений, расположенную в фиксированных адресах памяти. Обработчик сохраняет контекст процессора – содержимое счетчика команд, текущий режим процессора и статус разрешения прерываний. Таким образом, контекст может быть восстановлен по завершению обработки исключения.

При возникновении исключения в регистр Exception Program Counter (EPC) загружается адрес, начиная с которого исполнение команд может возобновиться после завершения обработки исключения. В регистр EPC помещается адрес команды, вызвавшей исключение или, если команда находилась в слоте задержки перехода, адрес команды перехода, предшествующей слоту задержки. Чтобы различить эти ситуации, программное обеспечение должно проанализировать бит BD (branch delay) в регистре Cause CP0.

2.7.1 Условия исключений

Исключения обрабатываются на стадии M конвейера. Когда исключительная ситуация обнаруживается, команда, находящаяся на стадии M, и все команды, следующие за ней на конвейере, отменяются. Соответственно, все условия остановки конвейера, относящиеся к этой команде, а также условия последующих исключений, которые также могут относиться к ней, игнорируются, поскольку обслуживание приостановок для отмененной команды не приносит выигрыша.

Когда условие исключения обнаруживается на стадии M, процессор заполняет необходимые регистры CP0 значениями, относящимися к состоянию исключения, изменяет счетчик команд (PC) на адрес соответствующего вектора обработки исключения и очищает признаки исключения, относящиеся к более ранним стадиям конвейера.

Такая реализация позволяет завершить исполнение команды, находящейся на стадии W, и запретить завершение последующих команд. Таким образом, значения, сохраненного в регистре EPC (в случае ошибок – в Error PC), достаточно для возобновления исполнения. Это также обеспечивает поступление исключений в соответствии с порядком исполнения команд – команда, вызывающая исключение, может быть уничтожена командой с более поздней стадии конвейера, также вызвавшей исключение.

2.7.2 Приоритеты исключений

В Таблица 2.15. перечислены все возможные исключения со своими относительными приоритетами от высшего к низшему. Некоторые из этих исключений могут случаться одновременно, в этом случае вызывается исключение с наивысшим приоритетом.

Таблица 2.15

Исключение	Описание
Reset	Аппаратный сброс
NMI	Внешнее немаскируемое прерывание и прерывание от таймера WDT (см. табл. 7.2).
TLB_Ri, TLB_Ii	Промах TLB при выборке команды, Попадание в запрещенную страницу TLB (V=0) при выборке команды
AdELi	Ошибка выравнивания адреса при выборке команды; Ссылка на адрес режима Kernel при работе в режиме User при выборке команды
MCheck Sys Bp CpU RI Ov Tr AdELd AdES	Запись в TLB, создающая конфликт с существующей строкой TLB Выполнение команды SYSCALL Выполнение команды BREAK Выполнение команды сопроцессора в режиме User Выполнение зарезервированной команды Переполнение в арифметической команде Выполнение trap (когда условие trap истинно) Ошибка выравнивания адреса при загрузке данных; Ссылка на адрес режима Kernel при работе в режиме User при загрузке данных Ошибка выравнивания адреса при сохранении данных; Попытка сохранения по адресу Kernel в режиме User
TLB_Rd, TLB_Id	Промах TLB при загрузке данных; Попадание в запрещенную страницу TLB (V=0) при загрузке данных
TLB_M	Сохранение в TLB-странице с D=0
Interrupt	Установка немаскируемых HW или SW - прерываний

2.7.3 Расположение векторов исключений

Векторы исключений аппаратного сброса и NMI всегда находятся по адресу 0xBFC_0000. Адреса всех других исключений являются комбинациями векторных смещений и базового адреса. В Таблице 2.16 приведены базовые адреса как функции исключения и состояния бита BEV Регистра Status. В Таблице 2.17 приведены смещения от базового адреса как функции исключения. В Таблице 2.18 эти две таблицы сведены в одну таблицу, содержащую все возможные адреса векторов исключений как функции состояний, влияющих на выбор этих векторов.

Таблица 2.16.

Исключение	$Status_{BEV}$	
	0	**1**
Reset, NMI	0xBFC0_0000	
Остальные исключения	0x8000_0000	0xBFC0_0200

Таблица 2.17. Базовые адреса векторов исключений

Исключение	Смещение вектора
TLB Refill, EXL = 0	0x000
Reset, NMI	0x000
Исключения общего характера (General Exeptions)	0x180
Interrupt, $Cause_{IV}$ = 1	0x200

Таблица 2.18. Векторы исключений

Исключение	BEV	EXL	IV	Вектор
Reset, NMI	-	-	-	0xBFC0_0000
TLB Refill	0	0	-	0x8000_0000
TLB Refill	0	1	-	0x8000_0180
TLB Refill	1	0	-	0xBFC0_0200
TLB Refill	1	1	-	0xBFC0_0380
Interrupt	0	0	0	0x8000_0180
Interrupt	0	0	1	0x8000_0200
Interrupt	1	0	0	0xBFC0_0380
Interrupt	1	0	1	0xBFC0_0400
Остальные	0	-	-	0x8000_0180
Остальные	1	-	-	0xBFC0_0380

2.7.4 Обработка общих исключений

Кроме исключений аппаратного сброса и NMI, которые обслуживаются особым образом, обработка всех остальных исключений происходит в соответствии со следующим основным маршрутом:

- Если бит EXL Регистра Состояния (Status) очищен, в регистр EPC загружается значение PC, по которому выполнение программы будет перезапущено, и при необходимости устанавливается бит BD в Регистре Причины (Cause). Если команда не находится в слоте задержки перехода, бит BD в Регистре Причины будет очищен, а в регистр EPC загружается значение, соответствующее текущему PC. Если же команда находится в слоте задержки перехода, бит BD в Регистре Причины устанавливается в "1", и в EPC загружается значение, равное PC - 4. Если бит EXL в Регистре Состояния установлен, в регистр EPC ничего не загружается, и бит BD в Регистре Причины не модифицируется.
- В поля CE и ExcCode Регистра Причины загружаются значения, соответствующие исключению.
- Устанавливается бит EXL в Регистре Состояния (Status).
- Процессор стартует с вектора исключения.

Значение, загруженное в EPC, представляет собой адрес возврата из исключения и в обычной ситуации программе обработки исключения не требуется его модифицировать. Программе также не нужно просматривать бит BD в Регистре Причины, если не возникает потребность определить действительный адрес команды, вызвавшей исключение.

Operation:

```
if StatusEXL == 0 then
if InstructionInBranchDelaySlot then
EPC <= PC - 4
CauseBD <= 1
else
EPC <= PC
CauseBD <= 0
endif
if (ExceptionType == TLBRefill) then
vectorOffset <= 0x000
elseif (ExceptionType == Interrupt) and
(CauseIV == 1) then
vectorOffset <= 0x200
else
vectorOffset <= 0x180
endif
else
vectorOffset <= 0x180
endif
CauseCE <= FaultingCoprocessorNumber
CauseExcCode <= ExceptionType
StatusEXL <= 1
if (StatusBEV == 1) then
PC <= 0xBFC0_0200 + vectorOffset
else
PC <= 0x8000_0000 + vectorOffset
endif
```

2.7.5 Исключения

В следующих разделах описаны все исключения в порядке, соответствующем табл.2.4.

2.7.5.1 Исключение по аппаратному сбросу (Reset Exception)

Это немаскируемое исключение, которое происходит при установке сигнала аппаратного сброса. Когда возникает исключение аппаратного сброса, процессор выполняет полную начальную инициализацию, то есть приводит автоматы к начальному состоянию и переводит процессор в состояние, из которого он может начать запуск команд, находящихся в некэшируемой и неотображаемой области. После возникновения исключения аппаратного сброса состояние процессора не определено, за исключением следующего:

- Регистр Random устанавливается в значение, равное количеству строк TLB - 1.
- Регистр Wired устанавливается в 0.
- Регистр Config устанавливается в свое начальное состояние (boot state).
- Поля BEV, TS, NMI и ERL Регистра Status устанавливаются в заданные значения.
- В PC загружается значение 0xBFC0_0000 (виртуальный адрес).

Вектор исключения:

Reset (0xBFC0_0000)

Operation:

Random <= TLBEntries – 1
Wired <= 0
Config <= ConfigurationState
$Status_{BEV}$ <= 1
$Status_{TS}$ <= 0
$Status_{NMI}$ <= 0
$Status_{ERL}$ <= 1
PC <= 0xBFC0_0000

2.7.5.2 Исключение по немаскируемому прерыванию (Non Maskable Interrupt – NMI Exception)

Немаскируемое прерывание возникает по положительному фронту входного сигнала NMI или при срабатывании сторожевого таймера WDT. Исключение NMI происходит только в пределах границ команды, поэтому оно не вызывает сброса или другую переинициализацию аппаратных средств. Состояние кэш, памяти, а также другие состояния процессора остаются неизменными. Значения регистров также сохраняются за исключением следующего:

- Поля BEV, TS, NMI и ERL регистра Status принимают заданные значения.
- В регистр ErrorEPC загружается значение PC - 4, если прерывание произошло на фоне команды в слоте задержки перехода. В противном случае в регистр ErrorEPC загружается значение PC.
- В PC загружается значение 0xBFC0_0000.

Вектор исключения:

Reset (0xBFC0_0000)

Operation:

StatusBEV <= 1
StatusTS <= 0
StatusNMI <= 1

StatusERL <= 1
if InstructionInBranchDelaySlot then
ErrorEPC <= PC - 4
else
ErrorEPC <= PC
endif

PC <= 0xBFC0_0000

2.7.5.3 Исключение по обновлению TLB — выборка команды или доступ к данным (TLB Refill Exception – Instruction Fetch or Data Access)

Исключение TLB Refill происходит во время выборки команды или доступа к данным, если в TLB нет ни одной строки, соответствующей ссылке к отображенному адресному пространству, и бит EXL в регистре Status равен 0.

Значение поля ExcCode регистра Cause:

TLBL: Произошла ссылка по загрузке данных или выборке команды

TLBS: Произошла ссылка по сохранению данных

Дополнительно сохраняемые состояния:

Таблица 2.19

Состояние регистра	Значение
BadVAddr	ошибочный адрес
Context	поле BadVPN2 содержит $VA_{31:13}$ ошибочного адреса
EntryHi	поле VPN2 содержит $VA_{31:13}$ ошибочного адреса; поле ASID содержит ASID отсутствующей ссылки

Вектор исключения:

Вектор TLB Refill (смещение 0x000)

2.7.5.4 Исключение TLB Invalid — выборка команды или доступ к данным (TLB Invalid Exception – Instruction Fetch or Data Access)

Исключение TLB Invalid происходит во время выборки команды или доступа к данным в одном из следующих случаев:

- В TLB нет ни одной строки, соответствующей ссылке к отображенному адресному пространству, и бит EXL в регистре Status равен 1.
- Строка TLB соответствует ссылке к отображенному адресу, но ее бит валидности выключен.

Значение поля ExcCode регистра Cause:

TLBL: Произошла ссылка по загрузке данных или выборке команды

TLBS: Произошла ссылка по сохранению данных

Дополнительно сохраняемые состояния:

Таблица 2.20

Состояние регистра	Значение
BadVAddr	ошибочный адрес
Context	поле BadVPN2 содержит $VA_{31:13}$ ошибочного адреса
EntryHi	поле VPN2 содержит $VA_{31:13}$ ошибочного адреса; поле ASID содержит ASID отсутствующей ссылки

Вектор исключения:

Общий Вектор исключения (смещение 0x180)

2.7.5.5 Исключение по ошибке адресации — выборка команды / доступ к данным (Address Error Exception – Instruction Fetch / Data Access)

Исключение по ошибке адресации во время доступа к команде или данным возникает при попытке выполнить одно из следующих действий:

- Выбрать команду, загрузить или сохранить слово данных, если они не выровнены в границах слова
- Загрузить или сохранить половину слова, если оно не выровнено в границах половины слова
- Обратиться по адресу пространства Kernel при работе в режиме User

Значение поля ExcCode регистра Cause:

ADEL: Произошла ссылка по загрузке данных или выборке команды

ADES: Произошла ссылка по сохранению данных

Дополнительно сохраняемые состояния:

Таблица 2 21

Состояние регистра	Значение
BadVAddr	ошибочный адрес

Вектор исключения:

Общий Вектор исключения (смещение 0x180)

2.7.5.6 Исключение по аппаратному контролю (Mcheck – Machine Check Exception)

Данное исключение возникает, если при выполнении команды записи в TLB (TLBWI или TLBWR) обнаруживается, что поле виртуального адреса записываемой строки соответствует такому же полю одной из строк, уже хранящихся в TLB.

При возникновении данной ситуации запись в TLB не выполняется и устанавливается бит TS в регистре Status. Этот бит является статусным и не влияет на функционирование процессорного ядра. Сбрасывается он программно после разрешения данной ситуации, осуществляемого очисткой конфликтных строк в TLB.

Значение поля ExcCode регистра Cause:

Mcheck

Дополнительно сохраняемые состояния:

Нет

Вектор исключения:

Общий Вектор исключения (смещение 0x180)

2.7.5.7 Исключение исполнения – системный вызов (System Call Exception)

Исключение System Call является одним из шести исключений исполнения. Все такие исключения имеют одинаковый приоритет. Исключение System Call возникает при исполнении команды SYSCALL.

Значение поля ExcCode регистра Cause:

Sys

Дополнительно сохраняемые состояния:

Нет

Вектор исключения:

Общий Вектор исключения (смещение 0x180)

2.7.5.8 Исключение исполнения — Breakpoint (Execution Exception – Breakpoint)

Исключение Breakpoint является одним из шести исключений исполнения. Все такие исключения имеют одинаковый приоритет. Исключение Breakpoint возникает при исполнении команды BREAK.

Значение поля ExcCode регистра Cause:

Bp

Дополнительно сохраняемые состояния:

Нет

Вектор исключения:

Общий Вектор исключения (смещение 0x180)

2.7.5.9 Исключение исполнения — зарезервированная команда (Execution Exception – Reserved Instruction)

Исключение зарезервированной команды является одним из шести исключений исполнения. Все такие исключения имеют одинаковый приоритет. Исключение зарезервированной команды вызывается при исполнении команды с неопределенным кодом операции или полем функции.

Значение поля ExcCode регистра Cause:

RI

Дополнительно сохраняемые состояния:

Нет

Вектор исключения:

Общий Вектор исключения (смещение 0x180)

2.7.5.10 Исключение исполнения — недоступен сопроцессор (Execution Exception – Coprocessor Unusable)

Исключение недоступности сопроцессора является одним из шести исключений исполнения. Все такие исключения имеют одинаковый приоритет. Исключение недоступности сопроцессора вызывается при попытке исполнения команды сопроцессора CP0 в режиме User.

Значение поля ExcCode регистра Cause:

CpU

Дополнительно сохраняемые состояния:

Нет

Вектор исключения:

Общий Вектор исключения (смещение 0x180)

2.7.5.11 Исключение исполнения — целочисленное переполнение (Execution Exception – Integer Overflow)

Исключение целочисленного переполнения является одним из шести исключений исполнения. Все такие исключения имеют одинаковый приоритет. Исключение целочисленного переполнения вызывается, когда выбранные целочисленные команды приводят к переполнению в двоичном коде.

Значение поля ExcCode регистра Cause:

Ov

Дополнительно сохраняемые состояния:

Нет

Вектор исключения:

Общий Вектор исключения (смещение 0x180)

2.7.5.12 Исключение исполнения — Trap (Execution Exception – Trap)

Исключение Trap является одним из шести исключений исполнения. Все такие исключения имеют одинаковый приоритет. Исключение Trap вызывается, если условие команды trap истинно (TRUE).

Значение поля ExcCode регистра Cause:

Tr

Дополнительно сохраняемые состояния:

Нет

Вектор исключения:

Общий Вектор исключения (смещение 0x180)

2.7.5.13 Исключение сохранения в запрещенной области (TLB Modified Exception)

Это исключение возникает при обращении по записи данных к отображенному адресу, если выполняется следующее условие:

- Найденная строка TLB действительна, но страница запрещена для записи.

Значение поля ExcCode регистра Cause:

Mod

Дополнительно сохраняемые состояния:

Таблица 2.22

Состояние регистра	Значение
BadVAddr	Ошибочный адрес
Context	Поля BadVPN2 содержат $VA_{31:13}$ ошибочного адреса
EntryHi	Поле VPN2 содержит $VA_{31:13}$ ошибочного адреса; поле ASID содержит ASID отсутствующей ссылки

Вектор исключения:

Общий Вектор исключения (смещение 0x180)

2.7.5.14 Исключение прерывания (Interrupt Exception)

Исключение прерывания возникает, когда сигнал одного или более разрешенных регистром Status прерываний устанавливается на входе процессора.

Значение поля ExcCode регистра Cause:

Int

Дополнительно сохраняемые состояния:

Таблица 2.23

Состояние регистра	Значение
$Cause_{IP}$	Указывает код прерывания

Вектор исключения:

Общий Вектор исключения (смещение 0x180), если бит IV регистра Cause равен 0;

Вектор прерывания (смещение 0x200), если бит IV регистра Cause равен 1.

2.7.6 Алгоритмы обработки исключений

В этом разделе приведены алгоритмы обработки следующих исключений:

- Общие исключения;
- Исключения пропуска при поиске по TLB;
- Исключения Reset и NMI;

Исключения аппаратно обрабатываются, а затем программно обслуживаются.

Алгоритмы обработки исключений приведены на Рисунок 2.21, Рисунок 2.22, Рисунок 2.23.

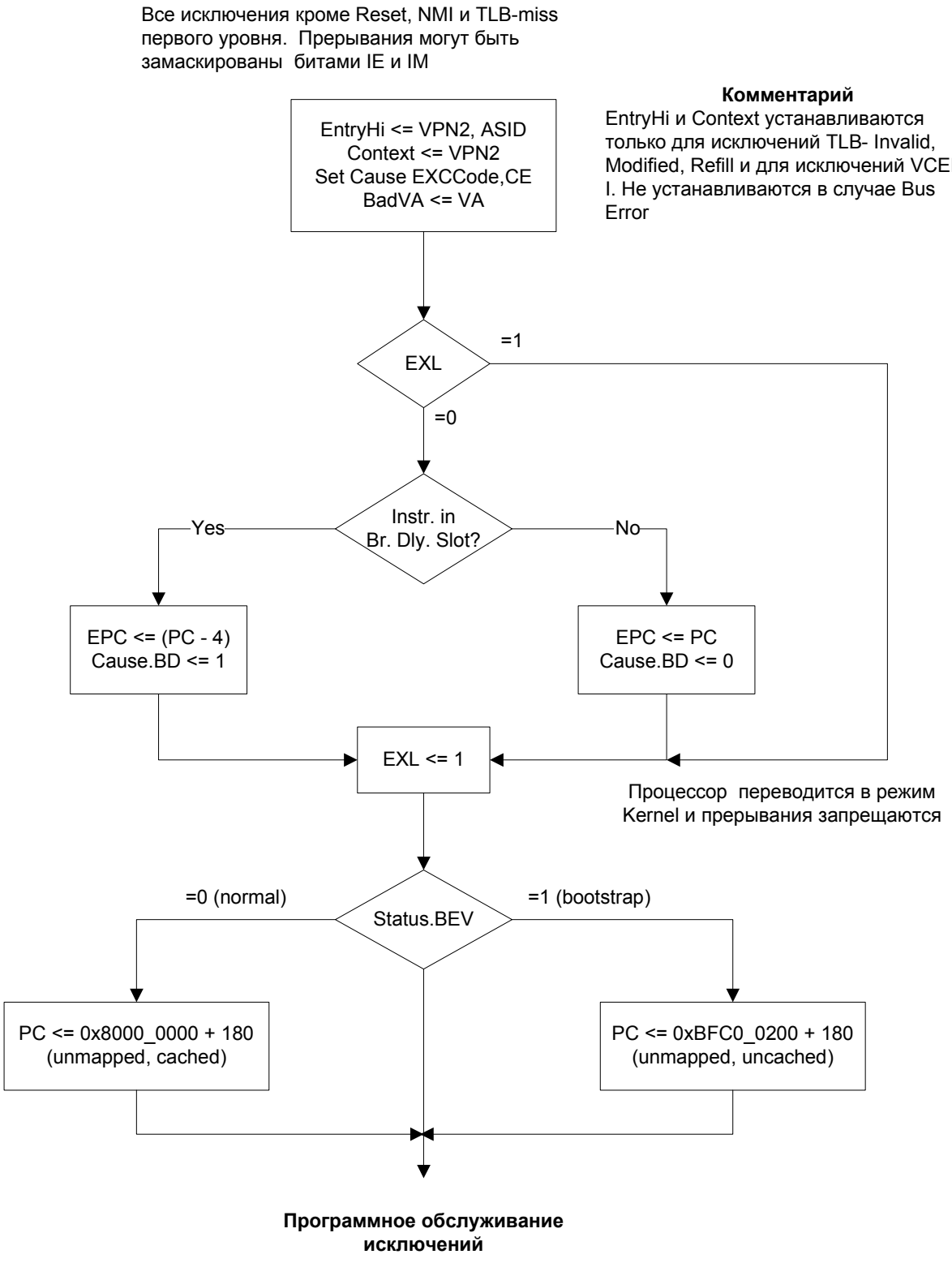


Рисунок 2.21 Обработка общих исключений

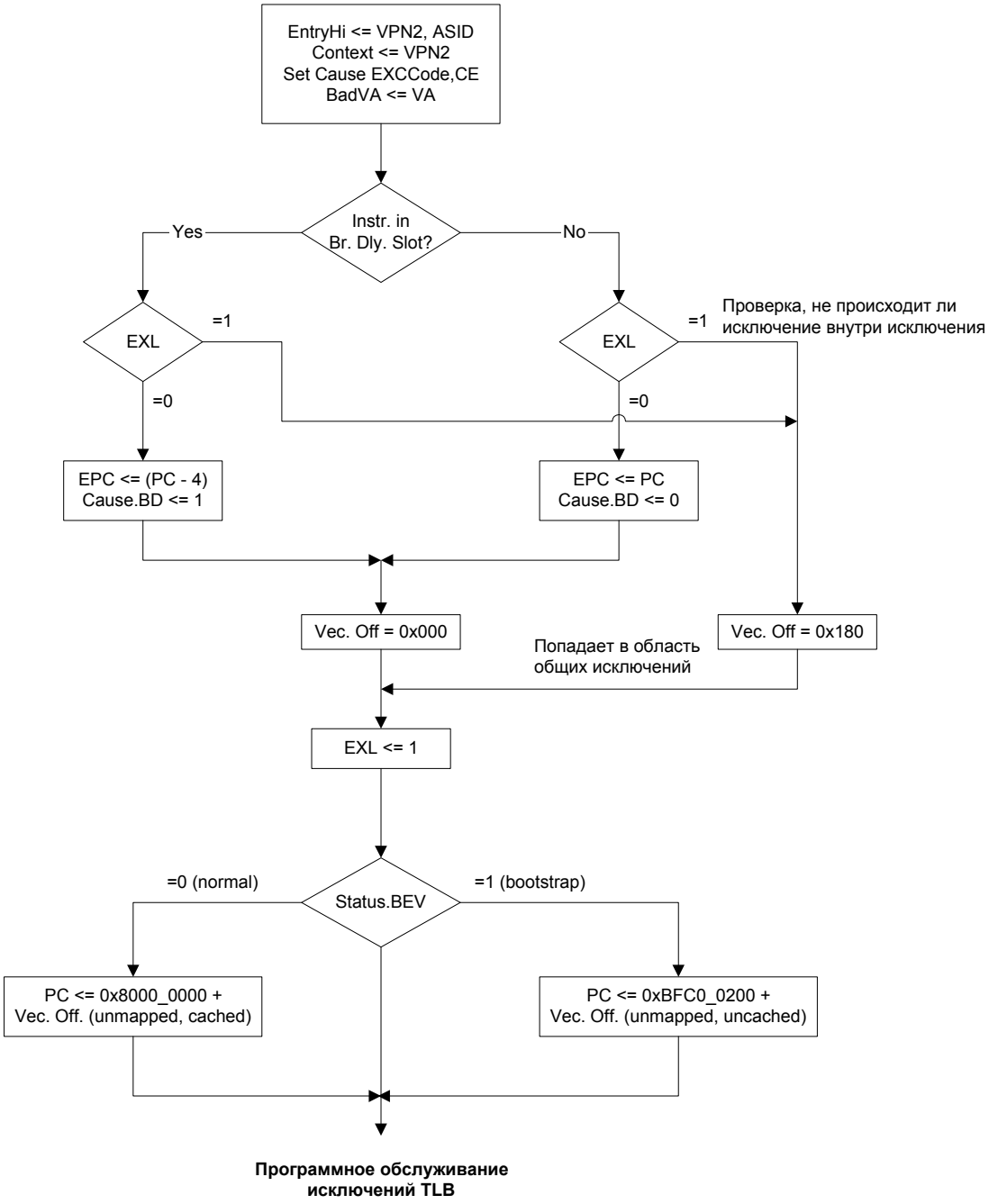


Рисунок 2.22 Обработка исключений TLB Refill и TLB Invalid

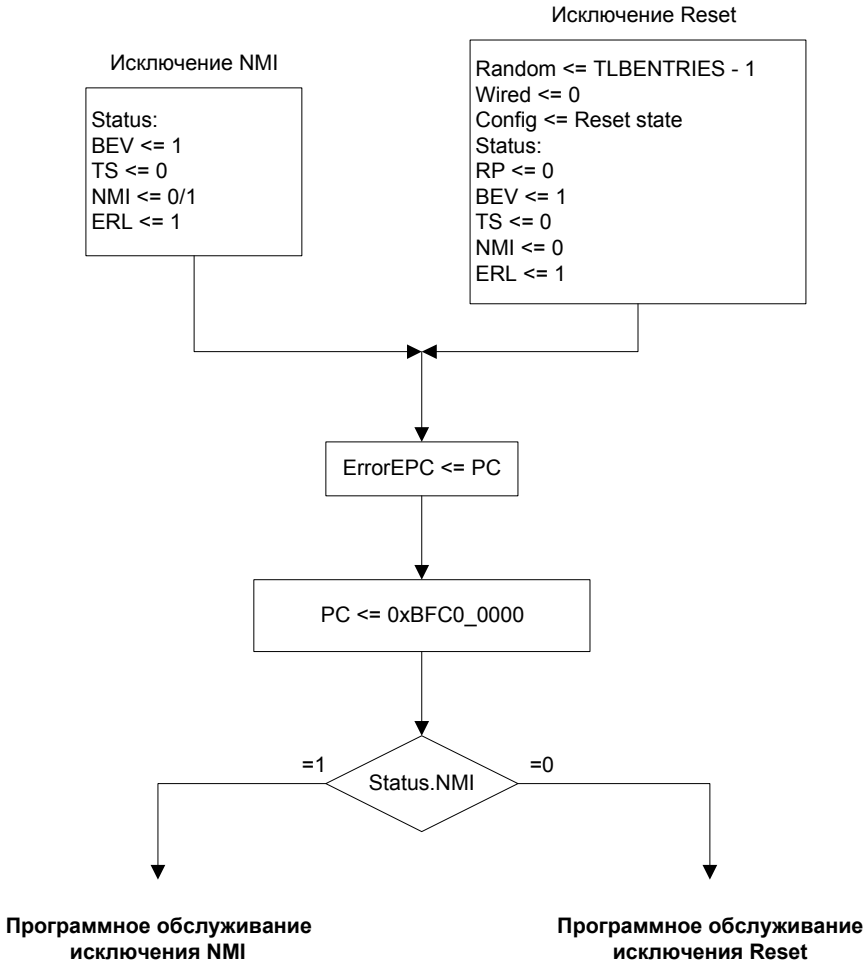


Рисунок 2.23. Обработка исключений Reset и NMI

2.8 Регистры CP0

2.8.1 Назначение

Системный Управляющий Сопроцессор (CP0) обеспечивает регистровый интерфейс с процессорным ядром MIPS32 и поддерживает управление памятью, преобразование адреса, обработку исключений и другие привилегированные операции. Каждому регистру CP0 соответствует определяющий его уникальный номер; этот номер называется *номером регистра*. Например, регистру PageMask соответствует 5-й номер регистра.

После записи нового значения в регистр CP0 (с помощью команды MTC0), его обновление происходит не сразу, а по прошествии периода от 0 и более команд. Этот период называется периодом особой ситуации.

2.8.2 Обзор регистров CP0

В Таблица 2.24. приведены все регистры CP0 в порядке возрастания нумерации. В разделе 5.3 каждый из этих регистров описан отдельно.

Таблица 2.24. Регистры CP0

Номер регист-ра	Название Регистра	Функция
0	Index[1]	Индекс матрицы TLB (режим TLB)
1	Random[1]	Случайным образом сгенерированный индекс для буфера TLB (режим TLB)
2	EntryLo0[1]	Младшая часть строки TLB для виртуальных страниц с четными номерами (режим TLB)
3	EntryLo1[1]	Младшая часть строки TLB для виртуальных страниц с нечетными номерами (режим TLB)
4	Context[2]	Указатель на строку в таблице страниц памяти (режим TLB)
5	PageMask[1]	Управление переменным размером страниц строк TLB (режим TLB)
6	Wired[1]	Управление количеством закрепленных "привязанных" строк TLB (режим TLB)
7	Reserved	Резерв
8	BadVAddr[2]	Содержит адрес, вызвавший последнее связанное с адресацией исключение
9	Count[2]	Счетчик процессорных циклов
10	EntryHi[1]	Старшая часть строки TLB (режим TLB)
11	Compare[2]	Управление прерыванием таймера
12	Status[2]	Состояние и управление процессором
13	Cause[2]	Причина последнего исключения
14	EPC[2]	Значение счетчика команд во время последнего исключения
15	PRId	Идентификация и ревизия процессора
16	Config/Config1	Конфигурационный регистр
17	LLAddr	Загрузка адреса сопряжения
18-19	Не реализованы	
20-22	Reserved	Резерв
23-24	Не реализованы	
25-27	Reserved	Резерв
28-29	Не реализованы	
30	ErrorEPC[2]	Значение счетчика команд при последней ошибке
31	Не реализован	

[1]Регистры, используемые при управлении памятью.

[2]Регистры, используемые при обработке исключений.

2.8.3 Регистры CP0

Регистры CP0 обеспечивают интерфейс между системой команд (ISA) и архитектурой процессора. Каждый регистр, описанный в этом разделе, представлен своим порядковым номером и значением поля select.

Все поля описанных регистров характеризуются свойствами записи / чтения, а также значением после аппаратного сброса. Свойства записи / чтения охарактеризованы в Таблица 2.25.

Таблица 2.25

Свойства записи/чтения	Аппаратная интерпретация	Программная интерпретация
R/W	Поле, в котором все биты программно и аппаратно доступны по записи и чтению. Аппаратное обновление этого поля доступно для программы при чтении программой. Программное обновление этого поля доступно для процессора при чтении процессором. Если значение поля после сброса не определено, программа или процессор должны проинициализировать это поле, чтобы первое чтение возвратило предсказуемое значение.	
R	Поле, значение которого постоянно или обновляется только процессором. Значение поля после начальной установки восстанавливается также при включении питания. Если значение поля не определено после начальной установки, процессор обновляет его только при условиях, определенных при описании поля.	Поле, для которого значение, записанное программой, процессором игнорируется. Программное прочтение этого поля возвращает последнее обновленное процессором значение. Если значение поля не определено после начальной установки, программное прочтение этого поля возвратит непредсказуемое значение кроме тех случаев, когда произошло обновление процессором значения этого поля по возникновению условий, определенных в описании поля условий.
0	Поле, значение которого процессором не обновляется и всегда равно нулю.	Программное чтение всегда возвращает нуль.

2.8.3.1 Регистр Index (Регистр 0 CP0, Select 0).

Регистр Index является 32-х разрядным регистром, доступным для чтения и записи. Он содержит индекс доступа к TLB для команд TLBP, TLBR и TLBWI. Ширина поля индекса зависит от количества строк TLB и равна 4.

Функционирование процессора НЕОПРЕДЕЛЕНО, если в регистр Index записано значение большее или равное количеству строк TLB.

Формат регистра Index

31 30 4 3 0		
P	0	Index

Таблица 2.26. Описание полей регистра Index

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	Биты			
P	31	Неудачная проба. Устанавливается в 1, если предыдущей командой TLBProbe (TLBP) не было найдено соответствия в TLB.	R	Не определено
0	30:4	При чтении возвращается нуль	0	0
Index	3:0	Индекс строки TLB, к которой относятся команды TLBRead и TLBWrite	R/W	Не определено

2.8.3.2 Регистр Random (Регистр CP0 1, Select 0).

Регистр Random доступен только для чтения, и его значение используется как индекс TLB для команды TLBWR. Ширина поля Random определяется таким же образом, как для регистра Index.

Значение этого регистра изменяется между верхней и нижней границами следующим образом:

- Нижняя граница определяется количеством строк TLB , зарезервированных для использования операционной системой (содержимое регистра Wired). Строка, чей индекс равен значению Wired, является первой из доступных для записи командой TLB Write Random (TLBWR).
- Верхняя граница равна общему количеству строк TLB минус 1.

Регистр Random уменьшается на 1 при продвижении конвейера RISC, возвращаясь к максимальному значению по достижению величины, равной значению регистра Wired.

Процессор инициализирует регистр Random значением, равным верхней границе по возникновению исключения Reset и по записи в регистр Wired.

Формат регистра Random

31 4 3 0	
0	Random

Таблица 2.27. Описание полей регистра Random

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	Биты			
0	31:4	При чтении возвращается нуль	0	0
Random	3:0	Случайный индекс строки TLB	R	TLB Entries - 1

2.8.3.3 EntryLo0, EntryLo1 (Регистры 2 и 3 CP0, Select 0)

Пара регистров EntryLo действует как интерфейс между TLB и командами TLBR, TLBWI, TLBWR.

В режиме TLB EntryLo0 содержит строки для четных страниц TLB, а EntryLo1 – для нечетных страниц.

После ошибки адресации и возникновения исключений TLB refill, TLB invalid и TLB modified, содержимое регистров EntryLo0 и EntryLo1 не определено.

Формат регистров EntryLo0, EntryLo1

31 30 29 26 25 6 5 3 2 1 0						
R	0	PFN	C	D	V	G

Таблица 2.28. Описание полей регистров EntryLo0 и EntryLo1

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	Биты			
R	31:30	Резервные. При чтении возвращается нуль	R	0
0	29:26	При чтении возвращается нуль	R	0
PFN	25:6	Номер страничного кадра. Соответствует битам 31:12 физического адреса.	R/W	Не определено
C	5:3	Атрибут когерентности страницы. См. табл.2.18.	R/W	Не определено
D	2	“Dirty” – бит, разрешающий запись. Указывает на то, что в страницу была сделана запись, и/или страница открыта для записи. Если этот бит равен 1, разрешается сохранение в этой странице. Если он равен 0, сохранение в этой странице вызывает исключение TLB Modified.	R/W	Не определено
V	1	Бит валидности. Указывает, на то, что строка TLB и, соответственно, отображение виртуальной страницы, является действительным. Если этот бит равен 1, доступ к странице разрешается. Если этот бит равен 0, доступ к странице вызывает исключение TLB Invalid.	R/W	Не определено
G	0	Бит глобальности. При записи в TLB битом G в строке TLB становится логическое “И“ битов G EntryLo0 и EntryLo1. Если бит G строки TLB равен 1, результат сравнения полей ASID игнорируется при поиске по TLB. При чтении строки TLB биты G EntryLo0 и EntryLo1 отражают состояние бита G TLB.	R/W	Не определено

В Таблица 2.29. приведена кодировка для поля C регистров EntryLo0 и EntryLo1 и полей K0, K23 и KU регистра Config.

Таблица 2.29. Атрибуты когерентности Кэш

Значение C[5:3]	Описание
0, 1, 3*, 4, 5, 6	Кэшируемая, некогерентная область
2*, 7	Некэшируемая область
* - Архитектура MIPS32 предусматривает только эти два значения. Остальные значения не используются и отображаются в используемые значения. Например, 0, 1, 4, 5 и 6 отображается в 3, а 7 – в 2.	

2.8.3.4 Регистр Context (Регистр 4 CP0, Select 0)

Регистр Context доступен для чтения и записи, и содержит указатель на строку в матрице PTE (page table entry). Эта матрица является структурой данных операционной системы, в которой содержатся преобразования виртуального адреса в физический. При возникновении промаха TLB, операционная система загружает в TLB недостающее преобразование из матрицы PTE. Регистр Context дублирует часть информации, содержащейся в регистре BadVAddr, но организован таким образом, что операционная система может прямо ссылаться к 8-байтной матрице PTE в памяти.

При возникновении исключения TLB (TLB Refill, TLB Invalid, или TLB Modified) биты $VA_{31:13}$ виртуального адреса записываются в поле BadVPN2 регистра Context. Поле PTEBase записывается и используется операционной системой.

После возникновения исключения ошибки адресации значение поля BadVPN2 регистра Context не определено.

Формат регистра Context

31 23	22 4	3 0
PTEBase	BadVPN2	

Таблица 2.30. Описание полей регистра Context

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	Биты			
PTEBase	31:23	Это поле используется операционной системой и обычно содержит значение, позволяющее операционной системе использовать регистр Context в качестве указателя на текущую матрицу PTE в памяти.	R/W	Не определено
BadVPN2	22:4	Это поле заполняется процессором при промахе TLB. Оно содержит биты $VA_{31:13}$ пропущенного виртуального адреса	R	Не определено
0	3:0	При чтении возвращается нуль	0	0

2.8.3.5 Регистр PageMask (Регистр 5 CP0, Select 0)

Регистр PageMask доступен для чтения и записи, и используется для чтения TLB и записи в TLB. Он содержит маску сравнения, которая устанавливает переменную размера страниц для каждой строки TLB, как показано вТаблица 2.32. Если значение регистра отлично от значений, приведенных в таблице, поведение процессора при поиске по TLB не определено.

Формат регистра PageMask

31	**25**	**24**	**13**	**12**	**0**
0		Mask		0	

Таблица 2.31. Описание полей регистра PageMask

Поля		**Описание**	**Чтение/ Запись**	**Начальное состояние**
Имя	**Биты**			
Mask	24:13	Бит маски, содержащий “1”, указывает на то, что соответствующий бит виртуального адреса не должен принимать участие при поиске соответствия по TLB	R/W	Не определено
0	31:25, 12:0	При чтении возвращается нуль	0	0

Таблица 2.32. Таблица возможных значений поля Mask регистра PageMask.

Размер страницы	**Бит**											
	24	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**
4 КБАЙТ	0	**0**	0	0	0	0	0	0	0	0	0	0
16 Кбайт	0	0	0	0	0	0	0	0	0	0	1	1
64 Кбайт	0	0	0	0	0	0	0	0	1	1	1	1
256 Кбайт	0	0	0	0	0	0	1	1	1	1	1	1
1 Мбайт	0	0	0	0	1	1	1	1	1	1	1	1
4 Мбайт	0	0	1	1	1	1	1	1	1	1	1	1
16 Мбайт	1	1	1	1	1	1	1	1	1	1	1	1

2.8.3.6 Регистр Wired (Регистр 6 CP0, Select 0)

Регистр Wired доступен для чтения и записи. Этот регистр определяет границу между случайными и “привязанными” строками TLB, как показано на Рисунок 2.24. Ширина поля Wired определяется так же, как для описанного выше регистра Index. “Привязанные ” строки зафиксированы, то есть они не являются удаляемыми и не могут быть перезаписаны командой TLBWR. Эти строки могут быть перезаписаны только командой TLBWI.

Регистр Wired устанавливается в нулевое состояние исключением по аппаратному сбросу (Reset). Запись в регистр Wired вызывает установку регистра Random в значение, равное его верхней границе.

Если значение, записанное в регистр Wired, больше или равно числу строк TLB, операция процессора не определена.

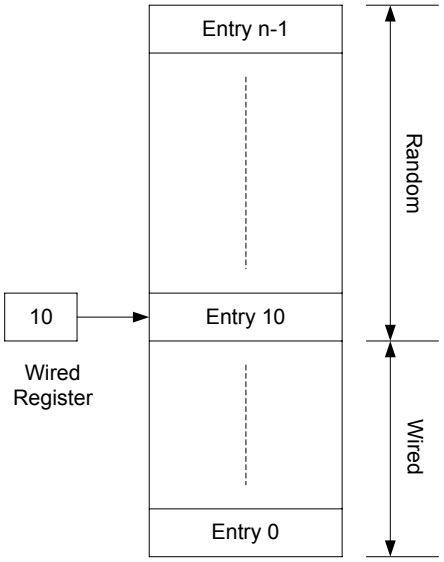


Рисунок 2.24. “Привязанные” и случайные строки TLB

Формат регистра Wired

31 **4 3** **0**	
0	Wired

Таблица 2.33. Описание полей регистра Wired

Поля		**Описание**	**Чтение/ Запись**	**Начальное состояние**
Имя	**Биты**			
0	31:4	При чтении возвращается нуль	0	0
Wired	3:0	Граница между “привязанными” и случайными строками TLB.	R/W	0

2.8.3.7 Регистр BadVAddr (Регистр 8 CP0, Select 0)

Регистр BadVAddr доступен только для чтения и содержит последний виртуальный адрес, вызвавший одно из следующих исключений:

- Ошибка адреса (AdEL или AdES)
- TLB Refill
- TLB Invalid
- TLB Modified

Формат регистра BadVAddr

31 0
BadVAddr

Таблица 2.34. Описание полей регистра BadVAddr

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	Биты			
BadVAddr	31:0	Виртуальный адрес, вызвавший исключение	R	Не определено

2.8.3.8 Регистр Count (Регистр 9 CP0, Select 0)

Регистр Count действует как таймер, увеличивающий свое значение каждый такт.

Регистр Count может быть записан в функциональных или диагностических целях, включая установку или синхронизацию процессора.

Формат регистра Count

31 0
COUNT

Таблица 2.35. Описание полей регистра Count

Поля		ОПИСАНИЕ	Чтение/ Запись	Начальное состояние
Имя	Биты			
Count	31:0	Счетчик	R/W	Не определено

2.8.3.9 Регистр EntryHi (Регистр 10 CP0, Select 0)

Регистр EntryHi содержит информацию соответствия виртуального адреса, использующуюся при чтении, записи и операциях доступа к TLB.

При возникновении исключений TLB (TLB Refill, TLB Invalid или TLB Modified) биты $VA_{31:13}$ виртуального адреса записываются в поле VPN2 регистра EntryHi. В поле ASID, которое используется в процессе сравнения при поиске по TLB, программно записывается идентификатор текущего адресного пространства.

Поле VPN2 регистра EntryHi не определено после прерывания по ошибке адресации.

Формат регистра EntryHi

31 0		
VPN2	0	ASID

Таблица 2.36. Описание полей регистра EntryHi

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	Биты			
VPN2	31:13	Разряды $VA_{31:0}$ виртуального адреса (виртуальный номер страницы, деленный на 2). Это поле записывается аппаратно при исключении TLB или при чтении TLB, и программно перед записью в TLB.	R/W	Не определено
0	12:8	При чтении возвращается нуль	0	0
ASID	7:0	Идентификатор адресного пространства. Это поле записывается аппаратно при чтении TLB, и программно при установке текущего значения ASID для записи в TLB и для сравнения при поиске по TLB с соответствующими полями ASID в строках TLB.	R/W	Не определено

2.8.3.10 Регистр Compare (Регистр 11 CP0, Select 0)

Регистр Compare действует совместно с регистром Count с целью реализации функции таймера и прерывания по таймеру. Прерывание по таймеру является выходным сигналом процессора.

Результат сравнения регистров Count и Compare заведен на 19 разряд регистра QSTR. Когда значение регистра Count равняется значению регистра Compare, этот бит имеет единичное состояние. Он остается в этом состоянии, пока в регистр Compare не будет произведена запись.

Для диагностических целей регистр Compare доступен для чтения и записи. Однако при нормальном функционировании регистр Compare используется только для записи. При записи значения в регистр Compare в качестве побочного эффекта происходит очистка бита прерывания по таймеру.

Формат регистра Compare

31 **0**
Compare

Таблица 2.37. Описание полей регистра Compare

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	Биты			
Compare	31:0	Период счета таймера	R/W	Не определено

2.8.3.11 Регистр Status (Регистр 12 CP0, Select 0)

Регистр Status (SR) является регистром, доступным для чтения и записи. Он содержит поля рабочего режима, разрешения прерываний и диагностические состояния процессора. Для задания режимов функционирования процессора, поля этого регистра объединяются следующим образом:

Разрешение прерываний: Прерывания разрешаются, когда истинны все следующие условия:

- IE = 1
- EXL = 0
- ERL = 0

Если эти условия выполнены, прерывания разрешаются установкой битов IM.

Рабочие режимы: Процессор всегда находится в одном из двух режимов – Kernel или User. Режим задается установкой следующих битов регистра Status CPU.

- Режим User: UM = 1, EXL = 0, and ERL = 0
- Режим Kernel: UM = 0 или EXL = 1 или ERL = 1

Формат Status регистра

31 28	**27**	**26**	**23**	**22**	**21**	**20**	**19**	**18 16**	**15 8**	**7**	**5**	**4**	**3 2**	**1**	**0**
CU3-CU0	0	0	BEV	TS	0	NMI	0	IM7-IM0	0	UM	0	ERL	EXL	IE	

Таблица 2.38. Описание полей регистра Status

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	Биты			
CU3-CU0	31:28	Не используются	R/W	Не определено
-	27	Не используется	0	0
-	26:23	При чтении возвращается нуль	0	0
BEV	22	Управление размещением векторов исключения: 0: Нормальный 1: Начальная загрузка	R/W	1
TS	21	TLB-закрытие системы. Этот бит устанавливается, если при выполнении команд TLBWI или TLBWR образуется команда, которая приводит к условию закрытия, если оно разрешено. Программа может записывать в этот разряд только 0, чтобы очистить его, и не может вызвать переход этого бита из 0 в 1.	R/W	0
NMI	19	Указывает, что вход в вектор исключения начальной установки был осуществлен по причине возникновения NMI. 0: Не NMI (Аппаратный сброс) 1: NMI Программное обеспечение может записывать в этот бит только 0, чтобы очистить его, и не может записать 1.	R/W	1 для NMI, иначе 0
-	18:16	При чтении возвращается нуль	0	0
IM[7:0]	15:8	Маска прерываний: управление разрешением внешних, внутренних и программных прерываний. Прерывание принимается в случае, если установлен бит IE регистра Status и установлены соответствующие биты как в поле IM[7:0] регистра Status, так и в поле IP[7:0] регистра Cause. 0: Запрос на прерывание не разрешен. 1: Запрос на прерывание разрешен.	R/W	Не определено

Продолжение Таблица 2.38. Описание полей регистра Status

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	Биты			
-	7:5	При чтении возвращается нуль	0	0
UM	4	Указывает на то, что процессор работает в непривилегированном режиме (User): 0: Процессор работает в привилегированном режиме (Kernel) 1: Процессор работает в непривилегированном режиме (User) Замечание: процессор может также находиться в режиме Kernel, если установлены биты EXL или ERL. Это условие не влияет на состояние бита UM.	R/W	Не определено
-	3	При чтении возвращается нуль	0	0
ERL	2	Уровень ошибки. Устанавливается процессором при возникновении исключений Reset и NMI. 0: Нормальный уровень 1: Уровень ошибки Когда бит ERL установлен: Процессор находится в режиме Kernel. Прерывания запрещены. Команда ERET использует адрес возврата, содержащийся в ErrorEPC вместо EPC. kuseg используется как неотображаемая и некэшируемая область. Это позволяет иметь доступ к главной памяти при ошибках кэш. Поведение процессора не определено, если бит ERL установлен при выполнении кода из useg/kuseg.	R/W	1

Продолжение Таблица 2.38. Описание полей регистра Status

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	Биты			
EXL	1	Уровень Исключения. Устанавливается процессором при возникновении любого исключения, кроме Reset и NMI. 0: Нормальный уровень 1: Уровень исключения Когда бит EXL установлен: Процессор переходит в привилегированный режим (Kernel). Прерывания запрещены. Исключения TLB Refill используют общий вектор исключения вместо вектора TLB Refill. Если происходит другое исключение, EPC не модифицируется.	R/W	Не определено
IE	0	Разрешение Прерывания. 0: Отключает прерывания 1: Разрешает прерываниям	R/W	Не определено

2.8.3.12 Регистр Cause (Регистр 13 CP0, Select 0)

Регистр Cause, в основном, описывает причину последнего исключения. Кроме того, поля регистра управляют запросами на программные прерывания и определяют вектор, которым обрабатываются прерывания. Все поля регистра Cause, за исключением IP[1:0], IV и WP, доступны только для чтения.

Формат регистра Cause

31	30 24	23	22 16	15 10	9 8	7	6 2	1 0
BD	0	IV	0	IP[7:2]	IP[1:0]	0	Exc Code	0

Таблица 2.39. Описание полей регистра Cause

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	**Биты**			
BD	31	Указывает на то, что последнее исключение произошло в слоте задержки перехода: 0: Не в слоте задержки 1: В слоте задержки Замечание: бит BD не модифицируется на новом исключении, если установлен бит EXL.	R	Не определено
0	30:24	При чтении возвращается нуль	0	0
IV	23	Указывает, какой вектор используется для обслуживания исключений прерывания – общий или специальный вектор прерываний: 0: Используется общий вектор исключения (0x180) 1: Используется специальный вектор прерываний (0x200)	R/W	Не определено
0	22:16	При чтении возвращается нуль	0	0
IP[7:2]	15:10	Указывает, какое прерывание установлено: 15: все внутренние прерывания от DMA и устройств микроконтроллера (объединены по ИЛИ); 14: не используется, всегда имеет нулевое состояние; 13: внешнее прерывание nIRQ[3]; 12: внешнее прерывание nIRQ[2]; 11: внешнее прерывание nIRQ[1]; 10: внешнее прерывание nIRQ[0].	R	Не определено
IP[1:0]	9:8	Управляет запросами программных прерываний (посредством записи «1» в данные разряды): 9: Запрос программного прерывания 1; 8: Запрос программного прерывания 0.	R/W	Не определено
ID	7	Прерывание от встроенных средств отладки программ (OnCD).	R/W	0
Exc Code	6:2	**Код исключения — см.** Таблица 2.40		
0	1:0	При чтении возвращается нуль	0	0

Таблица 2.40. Описание поля Exc Code регистра Cause

Значение Exc Code	Мнемоника	Описание
0	Int	Прерывание
1	Mod	TLB-исключение модификации
2	TLBL	TLB-исключение (загрузка или вызов команды)
3	TLBS	TLB-исключение (сохранение)
4	AdEL	Прерывание по ошибке адресации (загрузка или вызов команды)
5	AdES	Прерывание по ошибке адресации (сохранение)
6-7	-	Не используются
8	Sys	Системное исключение
9	Bp	Исключение Breakpoint
10	RI	Исключение зарезервированной команды
11	CpU	Исключение недоступности сопроцессора
12	Ov	Исключение целочисленного переполнения
13	Tr	Исключение Trap
14	-	Не используются
15	FPE	Исключение от сопроцессора арифметики в формате с плавающей точкой (FPU)
16-23	-	Не используются
24	MCheck	Аппаратный контроль
25-31	-	Не используются

2.8.3.13 Регистр EPC (Регистр 14 CP0, Select 0)

Программный счетчик исключения (EPC) является регистром, доступным для чтения и записи. EPC содержит адрес, начиная с которого возобновляется исполнение программы после завершения обработки исключения. Все биты регистра EPC значимы и должны перезаписываться.

Для синхронных (точных) исключений, EPC содержит одно из следующего:

- Виртуальный адрес команды, которая была прямой причиной исключения;
- Виртуальный адрес команды перехода (Branch или Jump), непосредственно предшествующей исключению, если команда, вызвавшая исключение, находится в слоте задержки перехода и установлен бит BD в регистре Cause.

Если установлен бит EXL в регистре Status, процессор не записывает адрес в регистр EPC при возникновении новых исключений. Однако, новое значение можно записать в EPC командой MTC0.

Формат регистра EPC

31 **0**
EPC

Таблица 2.41. Описание полей регистра EPC

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	Биты			
EPC	31:0	Программный счетчик исключения	R/W	Не определено

2.8.3.14 Регистр PRId (Регистр 15 CP0, Select 0)

Регистр идентификации процессора (PRId) – это 32-х разрядный регистр, доступный только для чтения. Он содержит информацию, идентифицирующую изготовителя, опции изготовителя, идентификацию процессора, и версию процессора.

Формат регистра PRId

31 24	23 16	15 8	7 0
R	Company ID	Processor ID	Revision

Таблица 2.42. Описание полей регистра PRId

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	Биты			
R		При чтении возвращается нуль	R	0
Company ID	23:16	Идентификация компании, которая проектировала или изготовляла процессор.	R	1010
Processor ID	15:8	Идентификация типа процессора.	R	10010
Revision	7:0	Номер версии процессора. Позволяет программам различать разные версии одного типа процессора.	R	0

2.8.3.15 Регистр Config (Регистр 16 CP0, Select 0)

Регистр Config определяет различную конфигурационную информацию, а также информацию о возможностях процессора. Большинство полей регистра Config инициализируется аппаратно при выполнении исключения Reset или имеет постоянное значение, и только поле K0 должно быть проинициализировано программно обработчиком исключения Reset.

Формат регистра Config

31	30 28	27 25	24 21	20	19 18	17 16	15 14	13 12	10 9	7 6 3 2 0
M	K23	KU	0	MDU	R	MM	BM	BE	AT	AR	MT	0	K0

Таблица 2.43. Описание полей регистра Config

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	Биты			
M	31	Этот бит аппаратно устанавливается в высокий уровень, указывая на наличие регистра Config1	R	1
K23	30:28	Это поле управляет кэшируемостью адресных сегментов kseg2 и kseg3 в режиме FM. В режиме TLB не используется. См. табл.2.33.	FM:R/W	FM:010
			TLB:R	TLB:000
KU	27:25	Это поле управляет кэшируемостью адресных сегментов kuseg и useg в режиме FM. В режиме TLB не используется. См. табл.2.33.	FM:R/W	FM:010
			TLB:R	TLB:000
0	24:21	Не используются	0	0
MDU	20	Тип MDU: итеративный умножитель и делитель	R	1
R	19	При чтении возвращается нуль	0	0
MM	18:17	Режим No Merging для 32 bit collapsing write buffer	R	0
BM	16	Тип передачи Burst: последовательный	R	0
BE	15	Режим endian: Little endian	R	0
AT	14:13	Тип архитектуры, реализованной процессором: MIPS32.	R	0
AR	12:10	Номер версии: 1	R	0
MT	9:7	Тип MMU: 1: Стандартный TLB (FM = 0) 3: Фиксированное отображение (FM = 1) 0, 2, 4-7: зарезервированы	R	TLB: 01 FM: 11
R	6:3	При чтении возвращается нуль	0	0
K0	2:0	Алгоритм когерентности для kseg0, см. Таблица 2.29.	R/W	010

Таблица 2.44. Атрибуты когерентности кэш

Значение C[5:3]	
0, 1, 3*, 4, 5, 6	Кэшируемая, некогерентная область
2*, 7	Некэшируемая область
* - Архитектура MIPS32 предусматривает только эти два значения. Остальные значения не используются и отображаются в используемые значения. Например, 0, 1, 4, 5 и 6 отображается в 3, а 7 – в 2.	

2.8.3.16 Регистр Config1 (Регистр 16 CP0, Select 1)

Регистр Config1 является дополнением к регистру Config и кодирует дополнительную информацию о возможностях процессора. Все поля регистра Config1 доступны только для чтения.

Формат регистра Config1

31	30 25	24 22	21 19	18 16	15 13	12 10	9 7	6	5	4	3 2	1	0
R	MMUSize	IS	IL	IA	DS	DL	DA	R	PC	WR	CA	EP	FP

Таблица 2.45. Описание полей Config1 регистра

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	Биты			
R	31	При чтении возвращается нуль	0	0
Размер MMU	30:25	Это поле содержит количество строк TLB минус 1. В режиме TLB возвращается код 15 в десятичном формате, в режиме Fixed Mapping – 0.	R	001111 (FM =0) 000000 (FM =1)
IS	24:22	Количество наборов кэш команд: резервная опция	R	111
IL	21:19	Размер строки кэш команд: 16 байт	R	011
IA	18:16	Тип кэш команд: Direct mapped	R	0
DS	15:13	Нет кэш данных	R	0
DL	12:10	Нет кэш данных	R	0
DA	9:7	Нет кэш данных	R	0
R	6:5	При чтении возвращается нуль	0	0
PC	4	Нет регистра Performance Counter	R	0
WR	3	Нет регистра WATCH	R	0
CA	2	Не реализовано	R	0
EP	1	EJTAG не реализован	R	0
FP	0	Нет плавающей арифметики	R	0

2.8.3.17 Регистр LLAddr – Load Linked Address (Регистр 17 CP0, Select 0)

Регистр LLAddr содержит физический адрес последней команды Load Linked (LL). Этот регистр используется только для диагностических целей.

Формат LLAddr регистра

31 28	27 0
0	Paddr[31:4]

Таблица 2.46. Описание полей LLAddr регистра

Поля		ОПИСАНИЕ	Чтение/ Запись	Начальное состояние
Имя	**Биты**			
0	**31:28**	При чтении возвращается нуль	0	0
Paddr[31:4]	27:0	Физический адрес последней команды LL	R	Не определено

2.8.3.18 Регистр ErrorEPC (Регистр 30 CP0, Select 0)

Доступный для чтения и записи, регистр ErrorEPC полностью подобен регистру EPC, но используется при возникновении исключений ошибок. Все биты регистра ErrorEPC значимы и должны перезаписываться. Регистр ErrorEPC также используется для сохранения значения счетчика команд при возникновении исключений Reset и немаскируемого прерывании (NMI).

Регистр ErrorEPC содержит виртуальный адрес, начиная с которого может возобновиться исполнение программы после обработки ошибочной ситуации.

Этот адрес может быть:

- Виртуальным адресом команды, вызвавшей исключение;
- Виртуальным адресом команды перехода (Branch или Jump), непосредственно предшествующей исключению, если команда, вызвавшая ошибку, находится в слоте задержки перехода.

В отличие от регистра EPC, для регистра ErrorEPC не имеется соответствующего признака слота задержки перехода.

Формат регистра ErrorEPC

31 **0**
ErrorEPC

Таблица 2.47. Описание полей регистра ErrorEPC

Поля		Описание	Чтение/ Запись	Начальное состояние
Имя	Биты			
ErrorEPC	31:0	Счетчик команд при исключении ошибки	R/W	Не определен

Регистры WatchLo, WatchHi, Debug, DEPC, TagLo, DataLo, DeSave не реализованы

2.9 Кэш

2.9.1 Введение

В данной версии процессора реализован виртуально индексируемый и контролируемый по физическому тэгу кэш команд и данных типа direct mapped. Это позволяет осуществлять доступ к кэш параллельно с преобразованием виртуального адреса в физический. Объем каждой из кэш составляет 16 Кбайт.

Загрузка кэш (операция Refill) выполняются посредством пачки (burst), состоящей из 4 команд. Адрес, по которому начинается burst, выровнен по 16-байтной границе. До получения критического слова кэш блокируется.

2.9.2 Протокол кэш

2.9.2.1 Организация кэш

Кэш команд состоит из двух массивов – массива тэгов и массива данных. Кэш индексируется виртуально, поскольку для выбора соответствующей строки в обоих массивах используется виртуальный адрес. Контроль осуществляется по физическому тэгу, так-так массив тэгов содержит физический, а не виртуальный адрес.

На Рисунок 2.25 представлен формат каждой строки массивов тэгов и данных. Тэговая строка содержит 18 старших бита физического адреса (биты [31:14]) и бит валидности.

Строка данных содержит 4 32-х разрядных слова – всего 16 байт.

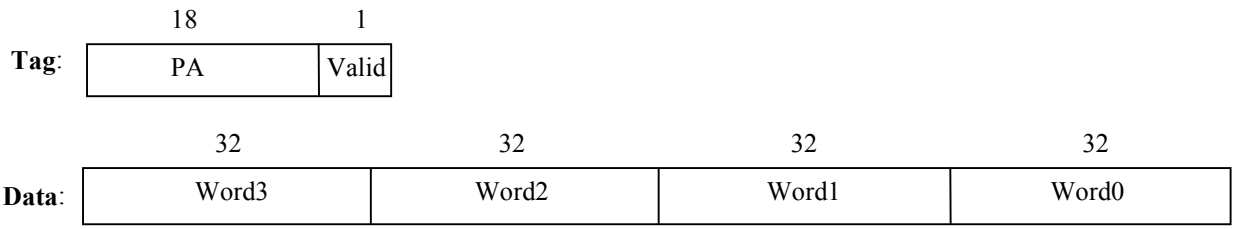


Рисунок 2.25 Формат массива кэш

2.9.2.2 Атрибуты кэшируемости.

В данной версии реализовано только два атрибута. Область может быть либо кэшируемой, либо некэшируемой (см. Таблица 2.44)

2.10 Карта памяти CPU

Карта физической памяти CPU приведена в Таблица 2.48. Здесь и далее, если это не оговорено специально, коды адреса и данных указаны в шестнадцатеричной системе счисления. Объемы областей памяти указаны с учетом ее дальнейшего расширения.

Таблица 2.48. Карта физической памяти CPU

Диапазон адресов	Название области	Объем области, Мбайт
FFFF_FFFF 2000_0000	Внешняя память	3584
1FFF_FFFF 1C00_0000	Внешняя память (ПЗУ)	64
1BFF_FFFF 1800_0000	Внутренняя память	64
17FF_FFFF 0000_0000	Внешняя память	384

Вся внешняя память доступна через порт внешней памяти (MPORT).

Для CPU все адресное пространство памяти является 32-разрядным. Память CRAM, а также внешняя память, могут адресоваться с точностью до байта.

При DMA обменах вся память является словной (32 разряда).

Карта внутренней памяти МСТ-01 приведена в Таблица 2.49.

Таблица 2.49. Карта внутренней памяти

Диапазон адресов	Название области	Объем области, Кбайт
1BFF_FFFF 1830_0000	Резерв	56000
182F_FFFF 182F_0000	Регистры периферийных устройств	64
182E_FFFF 1802_0000	Резерв	3000
1801_FFFF 1800_0000	Память CRAM	128

Перечень программно доступных регистров для CPU приведен в Таблица 2.50.

Таблица 2.50

Условное обозначение регистра	Название регистра	Адрес регистра
Регистры DMA линковых портов		
CSR_LpCh0	Регистр управления и состояния канала LpCh0	182F_0400
CP_LpCh0	Регистр указателя цепочки канала LpCh0	182F_0408
IR_LpCh0	Индексный регистр памяти канала LpCh0	182F_040C
OR_LpCh0	Регистр смещения памяти канала LpCh0	182F_0410
Y_LpCh0	Регистр параметров направления Y при двухмерной адресации памяти канала LpCh0	182F_0414
CSR_LpCh1	Регистр управления и состояния канала LpCh1	182F_0500
CP_LpCh1	Регистр указателя цепочки канала LpCh1	182F_0508
IR_LpCh1	Индексный регистр памяти канала LpCh1	182F_050C
OR_LpCh1	Регистр смещения памяти канала LpCh1	182F_0510
Y_LpCh1	Регистр параметров направления Y при двухмерной адресации памяти канала LpCh1	182F_0514
Регистры линковых портов		
LTx0	Буфер передачи порта LPORT0	182F_7000
LRx0	Буфер приема порта LPORT0	182F_7000
LCSR0	Регистр управления и состояния порта LPORT0	182F_7004
LDIR0	Регистр управления порта ввода-вывода LPORT0	182F_7008
LDR0	Регистр данных порта ввода-вывода LPORT0	182F_700C
LTx1	Буфер передачи порта LPORT1	182F_8000
LRx1	Буфер приема порта LPORT1	182F_8000
LCSR1	Регистр управления и состояния порта LPORT1	182F_8004
LDIR1	Регистр управления порта ввода-вывода LPORT1	182F_8008
LDR1	Регистр данных порта ввода-вывода LPORT1	182F_800C

Продолжение Таблица 2.50.

Условное обозначение регистра	Название регистра	Адрес регистра
Регистры DMA типа память-память		
CSR_MemCh0	Регистр управления и состояния канала MemCh0	182F_0800
IOR_MemCh0	Регистр индекса и смещения внутренней памяти канала MemCh0	182F_0804
CP_MemCh0	Регистр указателя цепочки канала MemCh0	182F_0808
IR_MemCh0	Индексный регистр внешней памяти канала Memh0	182F_080C
OR_MemCh0	Регистр смещения внешней памяти канала MemCh0	182F_0810
Y_MemCh0	Регистр параметров направления Y при двухмерной адресации внешней памяти канала MemCh0	182F_0814
Run0	Псевдорегистр управления состоянием бита RUN регистра CSR_MemCh0	182F_0818
CSR_MemCh1	Регистр управления и состояния канала MemCh1	182F_0900
IOR_MemCh1	Регистр индекса и смещения внутренней памяти канала MemCh1	182F_0904
CP_MemCh1	Регистр указателя цепочки канала MemCh1	182F_0908
IR_MemCh1	Индексный регистр внешней памяти канала MemCh1	182F_090C
OR_MemCh1	Регистр смещения внешней памяти канала MemCh1	182F_0910
Y_MemCh1	Регистр параметров направления Y при двухмерной адресации внешней памяти канала MemCh1	182F_0914
Run1	Псевдорегистр управления состоянием бита RUN регистра CSR_MemCh1	182F_0918

Продолжение Таблица 2.50.

Условное обозначение регистра	Название регистра	Адрес регистра
Регистры DMA типа память-память		
CSR_MemCh2	Регистр управления и состояния канала MemCh2	182F_0A00
IOR_MemCh2	Регистр индекса и смещения внутренней памяти канала MemCh2	182F_0A04
CP_MemCh2	Регистр указателя цепочки канала MemCh2	182F_0A08
IR_MemCh2	Индексный регистр внешней памяти канала Memh2	182F_0A0C
OR_MemCh2	Регистр смещения внешней памяти канала MemCh2	182F_0A10
Y_MemCh2	Регистр параметров направления Y при двухмерной адресации внешней памяти канала MemCh2	182F_0A14
Run2	Псевдорегистр управления состоянием бита RUN регистра CSR_MemCh2	182F_0A18
CSR_MemCh3	Регистр управления и состояния канала MemCh3	182F_0B00
IOR_MemCh3	Регистр индекса и смещения внутренней памяти канала MemCh3	182F_0B04
CP_MemCh3	Регистр указателя цепочки канала MemCh3	182F_0B08
IR_MemCh3	Индексный регистр внешней памяти канала Memh3	182F_0B0C
OR_MemCh3	Регистр смещения внешней памяти канала MemCh3	182F_0B10
Y_MemCh3	Регистр параметров направления Y при двухмерной адресации внешней памяти канала MemCh3	182F_0B14
Run3	Псевдорегистр управления состоянием бита RUN регистра CSR_MemCh3	182F_0B18

Продолжение Таблица 2.50.

Условное Обозначение регистра	Название регистра	Адрес регистра
Регистры контроллеров SWIC0		
HW_VER0	Регистр аппаратной версии контроллера	182F_5000
STATUS0	Регистр состояния	182F_5004
RX_TIME0	Регистр принятого маркера времени	182F_5008
MODE_CR0	Регистр управления режимом работы	182F_500C
TX_SPEED0	Регистр управления скоростью передачи	182F_5010
TX_TIME0	Регистр передаваемого маркера времени	182F_5014
CNT_TX_PACK 0	Регистр счетчика переданных пакетов	182F_5018
RX_SPEED0	Регистр измерителя скорости приема	182F_501C
ADDR_RxChDes 0	Регистр адреса блока памяти канала RxChDes	182F_5020
ADDR_RxChDat 0	Регистр адреса блока памяти канала RxChDat	182F_5024
ADDR_TxChDes 0	Регистр адреса блока памяти канала TxChDes	182F_5028
ADDR_TxChDat 0	Регистр адреса блока памяти канала TxChDat	182F_502C
SIZE_RxChDes0	Регистр размера блока памяти канала RxChDes	182F_5030
SIZE_RxChDat0	Регистр размера блока памяти канала RxChDat	182F_5034
SIZE_TxChDes0	Регистр размера блока памяти канала TxChDes	182F_5038
SIZE_TxChDat0	Регистр размера блока памяти канала TxChDat	182F_503C
CR_DMA0	Регистр управления каналами DMA	182F_5040
MAX_TR_SIZE0	Регистр максимального размера транзакции	182F_5044
PTR_RxChDat0	Регистр адреса цепочки канала RxChDat	182F_5048
PTR_TxChDat0	Регистр адреса цепочки канала TxChDat	182F_504C
SR_DMA0	Регистр состояния каналов DMA	182F_5050

Продолжение Таблица 2.50.

Условное Обозначение регистра	Название регистра	Адрес регистра
Регистры контроллеров SWIC1		
HW_VER1	Регистр аппаратной версии контроллера	182F_6000
STATUS1	Регистр состояния	182F_6004
RX_TIME1	Регистр принятого маркера времени	182F_6008
MODE_CR1	Регистр управления режимом работы	182F_600C
TX_SPEED1	Регистр управления скоростью передачи	182F_6010
TX_TIME1	Регистр передаваемого маркера времени	182F_6014
CNT_TX_PACK 1	Регистр счетчика переданных пакетов	182F_6018
RX_SPEED1	Регистр измерителя скорости приема	182F_601C
ADDR_RxChDes 1	Регистр адреса блока памяти канала RxChDes	182F_6020
ADDR_RxChDat 1	Регистр адреса блока памяти канала RxChDat	182F_6024
ADDR_TxChDes 1	Регистр адреса блока памяти канала TxChDes	182F_6028
ADDR_TxChDat 1	Регистр адреса блока памяти канала TxChDat	182F_602C
SIZE_RxChDes1	Регистр размера блока памяти канала RxChDes	182F_6030
SIZE_RxChDat1	Регистр размера блока памяти канала RxChDat	182F_6034
SIZE_TxChDes1	Регистр размера блока памяти канала TxChDes	182F_6038
SIZE_TxChDat1	Регистр размера блока памяти канала TxChDat	182F_603C
CR_DMA1	Регистр управления каналами DMA	182F_6040
MAX_TR_SIZE1	Регистр максимального размера транзакции	182F_6044
PTR_RxChDat1	Регистр адреса цепочки канала RxChDat	182F_6048
PTR_TxChDat1	Регистр адреса цепочки канала TxChDat	182F_604C
SR_DMA1	Регистр состояния каналов DMA	182F_6050

Продолжение Таблица 2.50.

Условное обозначение регистра	Название регистра	Адрес регистра
DMA контроллера SWIC0		
CSR_UCh0	Регистр управления и состояния канала	182F_0200
CP_UCh0	Регистр указателя цепочки канала	182F_0208
IR_UCh0	Индексный регистр внешней памяти канала	182F_020C
OR_UCh0	Регистр смещения внешней памяти канала	182F_0210
RUN_UCh0	Псевдорегистр управления состоянием бита RUN регистра CSR_UCh0	182F_0218
CSR_UCh1	Регистр управления и состояния канала	182F_0240
CP_UCh1	Регистр указателя цепочки канала	182F_0248
IR_UCh1	Индексный регистр внешней памяти канала	182F_024C
OR_UCh1	Регистр смещения внешней памяти канала	182F_0250
RUN_UCh1	Псевдорегистр управления состоянием бита RUN регистра CSR_UCh1	182F_0258
CSR_UCh2	Регистр управления и состояния канала	182F_0280
CP_UCh2 IR_UCh2	Регистр указателя цепочки канала Индексный регистр внешней памяти канала	182F_0288 182F_028C
OR_UC2	Регистр смещения внешней памяти канала	182F_0290
RUN_UCh2	Псевдорегистр управления состоянием бита RUN регистра CSR_UCh2	182F_0298
CSR_UCh3	Регистр управления и состояния канала	182F_02C0
CP_UCh3	Регистр указателя цепочки канала	182F_02C8
IR_UCh3	Индексный регистр внешней памяти канала	182F_02CC
OR_UC3	Регистр смещения внешней памяти канала	182F_02D0
RUN_UCh3	Псевдорегистр управления состоянием бита RUN регистра CSR_UCh3	182F_02D8

Продолжение Таблица 2.50.

Условное обозначение регистра	Название регистра	Адрес регистра
DMA контроллера SWIC1		
CSR_UCh0	Регистр управления и состояния канала	182F_0300
CP_UCh0	Регистр указателя цепочки канала	182F_0308
IR_UCh0	Индексный регистр внешней памяти канала	182F_030C
OR_UCh0	Регистр смещения внешней памяти канала	182F_0310
RUN_UCh0	Псевдорегистр управления состоянием бита RUN регистра CSR_UCh0	182F_0318
CSR_UCh1	Регистр управления и состояния канала	182F_0340
CP_UCh1	Регистр указателя цепочки канала	182F_0348
IR_UCh1	Индексный регистр внешней памяти канала	182F_034C
OR_UCh1	Регистр смещения внешней памяти канала	182F_0350
RUN_UCh1	Псевдорегистр управления состоянием бита RUN регистра CSR_UCh1	182F_0358
CSR_UCh2	Регистр управления и состояния канала	182F_0380
CP_UCh2 IR_UCh2	Регистр указателя цепочки канала Индексный регистр внешней памяти канала	182F_0388 182F_038C
OR_UC2	Регистр смещения внешней памяти канала	182F_0390
RUN_UCh2	Псевдорегистр управления состоянием бита RUN регистра CSR_UCh2	182F_0398
CSR_UCh3	Регистр управления и состояния канала	182F_03C0
CP_UCh3	Регистр указателя цепочки канала	182F_03C8
IR_UCh3	Индексный регистр внешней памяти канала	182F_03CC
OR_UC3	Регистр смещения внешней памяти канала	182F_03D0
RUN_UCh3	Псевдорегистр управления состоянием бита RUN регистра CSR_UCh3	182F_03D8

Продолжение Таблица 2.50.

Условное обозначение регистра	Название регистра	Адрес регистра
Регистры ADC		
CSR_ADC0	Регистр управления и состояния ADC0	182F_B000
CSR_ADC1	Регистр управления и состояния ADC1	182F_B004
CSR_ADC2	Регистр управления и состояния ADC2	182F_B008
CSR_ADC3	Регистр управления и состояния ADC3	182F_B00C
FIFO_ADC0	Буфер FIFO ADC0	182F_B010
FIFO_ADC1	Буфер FIFO ADC1	182F_B014
FIFO_ADC2	Буфер FIFO ADC2	182F_B018
FIFO_ADC3	Буфер FIFO ADC3	182F_B01C
Регистры DMA ADC		
CSR_Ach0	Регистр управления и состояния канала ACh0	182F_0000
CP_Ach0	Регистр указателя цепочки канала ACh0	182F_0008
IR_Ach0	Индексный регистр памяти канала ACh0	182F_000C
OR_Ach0	Регистр смещения памяти канала ACh0	182F_0010
RUN_Ach0	Псевдорегистр управления состоянием бита RUN регистра CSR_Ach0	182F_0018
CSR_Ach1	Регистр управления и состояния канала ACh1	182F_0040
CP_Ach1	Регистр указателя цепочки канала ACh1	182F_0048
IR_Ach1	Индексный регистр памяти канала ACh1	182F_004C
OR_Ach1	Регистр смещения памяти канала ACh1	182F_0050
RUN_Ach1	Псевдорегистр управления состоянием бита RUN регистра CSR_Ach1	182F_0058
CSR_Ach2	Регистр управления и состояния канала ACh2	182F_0080
CP_Ach2	Регистр указателя цепочки канала ACh2	182F_0088
IR_Ach2	Индексный регистр памяти канала ACh2	182F_008C
OR_Ach2	Регистр смещения памяти канала ACh2	182F_0090
RUN_Ach2	Псевдорегистр управления состоянием бита RUN регистра CSR_Ach2	182F_0098
CSR_Ach3	Регистр управления и состояния канала ACh3	182F_00C0
CP_Ach3	Регистр указателя цепочки канала ACh3	182F_00C8
IR_Ach3	Индексный регистр памяти канала ACh3	182F_00CC
OR_Ach3	Регистр смещения памяти канала ACh3	182F_00D0
RUN_Ach3	Псевдорегистр управления состоянием бита RUN регистра CSR_Ach3	182F_00D8

Продолжение Таблица 2.50.

Условное обозначение регистра	Название регистра	Адрес регистра
Регистры DAC		
CSR_DAC0	Регистр управления и состояния DAC0	182F_C000
CSR_DAC1	Регистр управления и состояния DAC1	182F_C004
CSR_DAC2	Регистр управления и состояния DAC2	182F_C008
CSR_DAC3	Регистр управления и состояния DAC3	182F_C00C
FIFO_DAC0	Буфер FIFO DAC0	182F_C010
FIFO_DAC1	Буфер FIFO DAC1	182F_C014
FIFO_DAC2	Буфер FIFO DAC2	182F_C018
FIFO_DAC3	Буфер FIFO DAC3	182F_C01C
Регистры DMA DAC		
CSR_Dch0	Регистр управления и состояния канала DCh0	182F_0100
CP_Dch0	Регистр указателя цепочки канала DCh0	182F_0108
IR_Dch0	Индексный регистр памяти канала DCh0	182F_010C
OR_Dch0	Регистр смещения памяти канала DCh0	182F_0110
RUN_Dch0	Псевдорегистр управления состоянием бита RUN регистра CSR_Dch0	182F_0118
CSR_Dch1	Регистр управления и состояния канала DCh1	182F_0140
CP_Dch1	Регистр указателя цепочки канала DCh1	182F_0148
IR_Dch1	Индексный регистр памяти канала DCh1	182F_014C
OR_Dch1	Регистр смещения памяти канала DCh1	182F_0150
RUN_Dch1	Псевдорегистр управления состоянием бита RUN регистра CSR_Dch1	182F_0158
CSR_Dch2	Регистр управления и состояния канала DCh2	182F_0180
CP_Dch2	Регистр указателя цепочки канала DCh2	182F_0188
IR_Dch2	Индексный регистр памяти канала DCh2	182F_018C
OR_Dch2	Регистр смещения памяти канала DCh2	182F_0190
RUN_Dch2	Псевдорегистр управления состоянием бита RUN регистра CSR_Dch2	182F_0198
CSR_Dch3	Регистр управления и состояния канала DCh3	182F_01C0
CP_Dch3	Регистр указателя цепочки канала DCh3	182F_01C8
IR_Dch3	Индексный регистр памяти канала DCh3	182F_01CC
OR_Dch3	Регистр смещения памяти канала DCh3	182F_01D0
RUN_Dch3	Псевдорегистр управления состоянием бита RUN регистра CSR_Dch3	182F_01D8

Продолжение Таблица 2.50.

Условное обозначение регистра	Название регистра	Адрес регистра
Регистры UART		
RBR	Приемный буферный регистр	182F_3000
THR	Передающий буферный регистр	182F_3000
IER	Регистр разрешения прерываний	182F_3004
IIR	Регистр идентификации прерывания	182F_3008
FCR	Регистр управления FIFO	182F_3008
LCR	Регистр управления линией	182F_300C
MCR	Регистр управления модемом	182F_3010
LSR	Регистр состояния линии	182F_3014
MSR	Регистр состояния модемом	182F_3018
SPR	Регистр Scratch Pad	182F_301C
DLL	Регистр делителя младший	182F_3000
DLM	Регистр делителя старший	182F_3004
SCLR	Регистр предделителя (scaler)	182F_3014
Регистры интервального таймера (IT)		
ITCSR	Регистр управления	182F_D000
ITPERIOD	Регистр периода работы таймера	182F_D004
ITCOUNT	Регистр счетчика	182F_D008
ITSCALE	Регистр предделителя	182F_D00C
Регистры WDT		
WTCSR	Регистр управления	182F_D010
WTPERIOD	Регистр периода работы таймера	182F_D014
WTCOUNT	Регистр счетчика	182F_D018
WTSCALE	Регистр предделителя	182F_D01C
Регистры RTT		
RTCSR	Регистр управления	182F_D020
RTPERIOD	Регистр периода работы таймера	182F_D024
RTCOUNT	Регистр счетчика	182F_D028

Продолжение Таблица 2.50.

Условное обозначение регистра	Название регистра	Адрес регистра
Регистры порта внешней памяти		
CSCON0	Регистр конфигурации сегмента 0 внешней памяти	182F_1000
CSCON1	Регистр конфигурации сегмента 1 внешней памяти	182F_1004
CSCON2	Регистр конфигурации сегмента 2 внешней памяти	182F_1008
CSCON3	Регистр конфигурации сегмента 3 внешней памяти	182F_100C
CSCON4	Регистр конфигурации внешней памяти, не вошедшей в сегменты 3-0	182F_1010
SDRCON	Регистр конфигурации памяти SDRAM	182F_1014
CKE_CTR	Регистр управления состоянием вывода CKE микросхемы	182F_1018
Системные регистры		
MASKR	Регистр маски	182F_4000
QSTR	Регистр заявок	182F_4004
CSR	Регистр управления	182F_4008

3. СИСТЕМНОЕ УПРАВЛЕНИЕ

3.1 Система синхронизации

МСТ-01 имеет два входа синхронизации:

- Вход системной частоты XTI/XTO. Сюда может подключаться кварцевый резонатор или внешний генератор;
- Вход частоты реального времени RTCXTI.

Схема синхронизации узлов МСТ-01 приведена на Рисунке 3.1.

Для синхронизации работы узлов МСТ-01 используется умножитель частоты на основе схемы фазовой автоподстройки частоты PLL. Управление PLL осуществляется при помощи полей CLK_SEL[4:0] (выбор коэффициента умножения/деления входной частоты) и CLKEN (разрешение формирование частоты) регистра CSR, а также при помощи внешнего вывода PLL_EN:

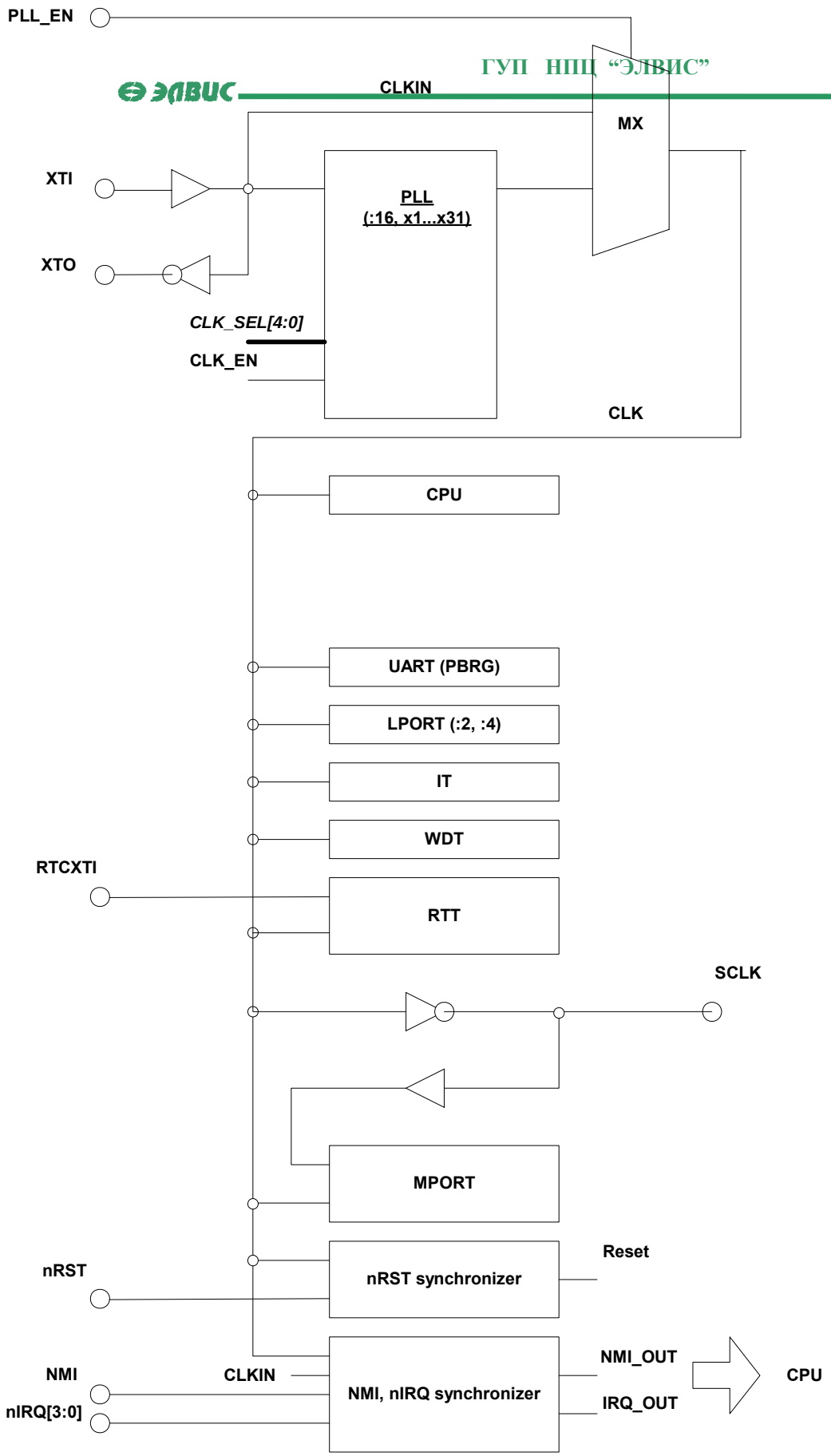
при PLL_EN=0 системная тактовая частота микроконтроллера равна входной частоте XTI;

при PLL_EN=1 системная тактовая частота микроконтроллера поступает из PLL и равна входной частоте XTI, умноженной на коэффициент умножения/деления.

CPU, IT, WDT, MPORT работают на частоте CLK.

Частота передачи данных линковыми портами (LPORT) – от CLK/2 до CLK/4.

Частота передачи данных UART определяется коэффициентом деления частоты CLK, который содержится в регистрах программируемого делителя (PBRG).



Reset - установка исходного состояния

CLK - системная тактовая частота

CLKIN - входная тактовая частота

NMI_OUT, IRQ_OUT - сигналы прерывания, поступающие на вход CPU

nRST, NMI, nIRQ synchronizer - схемы синхронизации входных сигналов

Рисунок 3.1. Схема синхронизации узлов МСТ-01

3.2 Отключение и включение тактовой частоты

В МСТ-01 имеется два режима энергосбережения:

- уменьшение внутренней тактовой частоты CLK;
- отключение внутренней тактовой частоты CLK.

Уменьшение внутренней тактовой частоты CLK выполняется при записи необходимого кода в поле CLK_SEL[4:0] регистра CSR. При этом значение тактовой частоты изменится через время не более чем 2 мс.

Отключение внутренней тактовой частоты выполняется следующим образом:

- программа CPU должна выполняться из кэш программ или из внутренней памяти CRAM;
- UART, DMA должны быть в неактивном состоянии;
- записать 1 в 31 разряд регистра SDRCON (поле RFR не должно быть изменено). По данной операции SDRAM деактивизируется (выполняется команда PRECHARGE);
- произвести запись нулей по адресу 182F_1018 (установка выходного сигнала CKE в нулевое состояние);
- произвести запись 0 в разряд CLKEN регистра CSR. По этой операции внутренняя тактовая частота отключается. За этой командой должна стоять команда NOP.

При отключении внутренней тактовой частоты энергопотребление уменьшается не менее чем в 100 раз.

Включение внутренней тактовой частоты осуществляется по любому внешнему прерыванию nIRQ[3:0] или NMI. Обработка исключения по данным прерываниям в этом случае должна выполняться следующим образом:

- для определения факта того, что прерывание произошло при выключенной частоте, можно опросить состояние бита CLKEN=0;
- записать 1 в бит CLKEN;
- произвести запись всех единиц по адресу 182F_1018 (установка сигнала CKE в единичное состояние);
- ожидание не менее 10 тактов.

3.3 Системные регистры

3.3.1 Регистр управления и состояния CSR

Формат регистра CSR приведен в Таблица 3.1.

Таблица 3.1

Номер разряда	Условное Обозначение	Назначение	Доступ	Исходное состояние
0	FM	Режим преобразования виртуальных адресов CPU в физические адреса: 0 – с использованием TLB; 1 – Fixed Mapped (FM).	R/W	1
3:1	-	Резерв	-	0
8:4	CLK_SEL[4:0]	Управление PLL: выбор коэффициента умножения/деления входной частоты: 0 – 1/16; 1 – 1 2 – 2; … 29 – 29; 30 – 30; 31 – 31.	R/W	1
11:9	-	Резерв	-	0
12	FLUSH	При записи 1 в данный разряд кэш команд CPU останавливается в исходное состояние, то есть ее содержимое девалидируется. Эта процедура может использоваться для обеспечения когерентности кэш при работе DMA.	W	0
15:13	-	Резерв	-	0
16	CLKEN	Управление PLL: разрешение формирование тактовой частоты: 1 – частота включена; 0 – частота выключена.	R/W	1
31:17	-	Резерв	-	0

Номерация разрядов регистров МСТ-01 соответствует нумерации разрядов памяти CPU. Если разряды регистров МСТ-01 доступны только по записи или не используются (резерв), то при чтении из них считываются нули. Если разряды регистров МСТ-01 доступны только по чтению или не используются, то при записи в них необходимо указывать нули.

3.3.2 Регистр запросов прерывания QSTR

Все сигналы внутренних прерываний поступают на вход псевдорегистра QSTR, формат которого приведен в Таблица 3.2

Данный регистр не имеет элементов памяти и доступен только по чтению.

Каждый разряд регистра QSTR содержит запрос прерывания от внутренних узлов МСТ-01 в не зависимости от состояния соответствующих разрядов регистра MASKR:

0 – нет запроса;

1 – есть запрос.

Сигналы внутренних прерываний формируются в соответсвующих устройствах при выполнении определенных условий. В процессе обслуживания прерывания необходимо проанализировать состояние устройства для определения причины его возникновения. Сброс прерывания осуществляется в момент исключения причины возникновения данного прерывания. Например, прерывание от LPORT (при неактивизированном DMA) сбрасывается при записи данных в буфер LTx или при чтении данных из буфера LRx.

Все незамаскированные прерывания объединяются по «или» и поступают в разряд IP[7] регистра Cause CPU.

Исходное состояние регистра QSTR – нули.

Таблица 3.2

Номер Разряда	**Условное обозначение прерывания**	**Название прерывания**
0	ADC_Ch0	Прерывание от канала DMA ADC_Ch0
1	ADC_Ch1	Прерывание от канала DMA ADC_Ch1
2	ADC_Ch2	Прерывание от канала DMA ADC_Ch2
3	ADC_Ch3	Прерывание от канала DMA ADC_Ch3
4	Uart	Прерывание от UART
5	SWIC0	Прерывание от SWIC0
6	SWIC1	Прерывание от SWIC1
7	LTRx0	Прерывание от порта LPORT0 при обмене данными или от канала DMA LportCh0
8	LSrq0	Запрос обслуживания от порта LPORT0
9	LTRx1	Прерывание от порта LPORT1 при обмене данными или от канала DMA LportCh0
10	LSrq1	Запрос обслуживания от порта LPORT1
11	SWIC0_Ch0	Прерывание от канала DMA SWIC0_Ch0
12	SWIC0_Ch1	Прерывание от канала DMA SWIC0_Ch1
13	SWIC0_Ch2	Прерывание от канала DMA SWIC0_Ch2
14	SWIC0_Ch3	Прерывание от канала DMA SWIC0_Ch3

Продолжение Таблица 3.2

Номер Разряда	Условное обозначение прерывания	Название прерывания
15	SWIC1_Ch0	Прерывание от канала DMA SWIC1_Ch0
16	SWIC1_Ch1	Прерывание от канала DMA SWIC1_Ch1
17	SWIC1_Ch2	Прерывание от канала DMA SWIC1_Ch2
18	SWIC1_Ch3	Прерывание от канала DMA SWIC1_Ch3
19	Compare	Прерывание от таймера CPU
20	-	Резерв
21	MemCh0	Прерывание от канала DMA MemCh0
22	MemCh1	Прерывание от канала DMA MemCh1
23	MemCh2	Прерывание от канала DMA MemCh2
24	MemCh3	Прерывание от канала DMA MemCh3
25	DAC_Ch0	Прерывание от канала DMA DAC_Ch0
26	DAC_Ch1	Прерывание от канала DMA DAC_Ch1
27	DAC_Ch2	Прерывание от канала DMA DAC_Ch2
28	DAC_Ch3	Прерывание от канала DMA DAC_Ch3
29	Timer	Прерывание от таймеров IT, WDT, RTT
31:30	-	Резерв

Таблица 3.3

Номер Разряда	Условное обозначение прерывания	Название прерывания
0	ADC_Ch0	Прерывание от канала DMA ADC_Ch0
1	ADC_Ch1	Прерывание от канала DMA ADC_Ch1
2	ADC_Ch2	Прерывание от канала DMA ADC_Ch2
3	ADC_Ch3	Прерывание от канала DMA ADC_Ch3
4	Uart	Прерывание от UART
5		
6		
7	LTRx0	Прерывание от порта LPORT0 при обмене данными или от канала DMA LportCh0
8	LSrq0	Запрос обслуживания от порта LPORT0
9	LTRx1	Прерывание от порта LPORT1 при обмене данными или от канала DMA LportCh0
10	LSrq1	Запрос обслуживания от порта LPORT1
11	SW_LINK0	Прерывание от SWIC0
12	SW_ERR0	Прерывание от SWIC0
13	SW_TIME0	Прерывание от SWIC0
14	SW_DMA0	Прерывание от SWIC0
15	SW_LINK1	Прерывание от SWIC1
16	SW_ERR1	Прерывание от SWIC1
17	SW_TIME1	Прерывание от SWIC1
18	SW_DMA1	Прерывание от SWIC1
19	Compare	Прерывание от таймера CPU
20	-	Резерв
21	MemCh0	Прерывание от канала DMA MemCh0
22	MemCh1	Прерывание от канала DMA MemCh1
23	MemCh2	Прерывание от канала DMA MemCh2
24	MemCh3	Прерывание от канала DMA MemCh3
25	DAC_Ch0	Прерывание от канала DMA DAC_Ch0
26	DAC_Ch1	Прерывание от канала DMA DAC_Ch1
27	DAC_Ch2	Прерывание от канала DMA DAC_Ch2
28	DAC_Ch3	Прерывание от канала DMA DAC_Ch3
29	Timer	Прерывание от таймеров IT, WDT, RTT
31:30	-	Резерв

3.3.3 Регистр маски MASKR

Каждое внутреннее прерывание маскируется при помощи 32-разрядного регистра маски MASKR, формат которого аналогичен формату регистра QSTR. Исходное состояние данного регистра – нули (все внутренние прерывания запрещены). Регистр доступен по записи и чтению.

3.4 Процедура начальной загрузки

После снятия сигнала nRST выполняется следующее:

- Все устройства МСТ-01 устанавливаются в исходное состояние;
- в CPU возникает исключение, вектор которого расположен по физическому адресу 0x1FC0_0000 внешней памяти. В этой области, как правило, расположено постоянное запоминающее устройство (ПЗУ) или, например память типа Flash.

В зависимости от состояния сигнала на выводе BYTE ПЗУ может быть 8 – или 32 – разрядным.

В ПЗУ может находиться или только программа начальной загрузки или все программы МСТ-01. В первом случае основная программа может быть загружена например через линковые порты.

Программа начальной загрузки должна обеспечивать конфигурирование всех устройств МСТ-01.

4. ИНТЕРВАЛЬНЫЙ ТАЙМЕР

4.1 Назначение

Интервальный таймер (IT), предназначен для выработки периодических прерываний на основе деления тактовой частоты CPU. Основные характеристики интервального таймера:

- Число разрядов основного делителя – 32;
- Число разрядов предделителя – 8;
- Программное управление стартом и остановкой таймера;
- Доступ ко всем регистрам обеспечивается в любой момент времени.

4.2 Структурная схема

Структурная схема интервального таймера приведена на Рисунок 4.1.

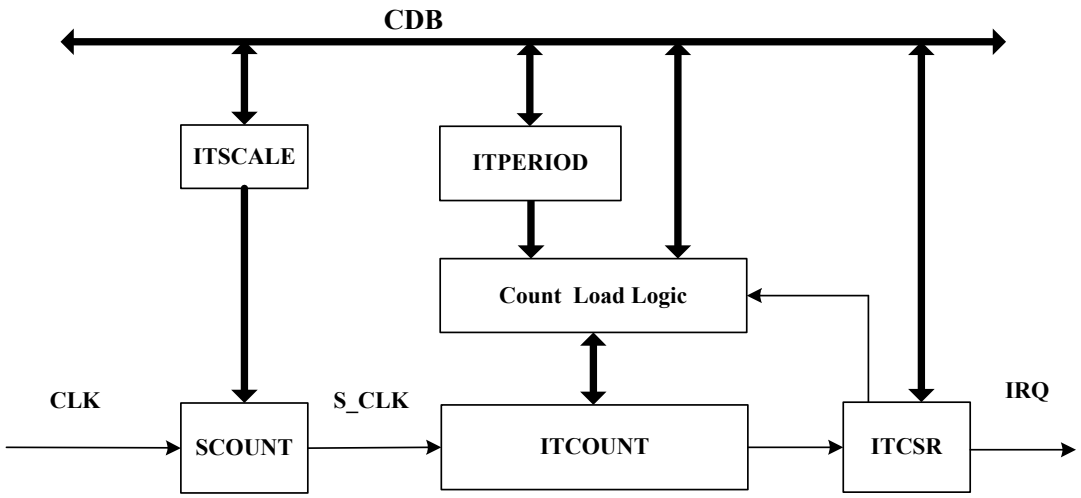


Рисунок 4.1. Структурная схема IT.

В состав интервального таймера входят следующие основные узлы:

- ITCSR - регистр управления и состояния;
- ITCOUNT - счетчик основного делителя;
- ITPERIOD - регистр периода основного делителя;
- ITSCALE - регистр предделителя;
- SCOUNT – счетчик предделителя;
- Count Load Logic - логика загрузки счетчика основного делителя.

На структурной схеме интервального таймера использованы следующие обозначения:

- CDB – шина данных CPU;
- CLK – тактовая частота работы CPU;
- S_CLK – выходная частота предделителя;
- IRQ – запрос на прерывание от интервального таймера.

4.3 Регистры интервального таймера

Перечень программно-доступных регистров интервального таймера приведен в Таблица 4.1.

Таблица 4.1. Перечень программно-доступных регистров интервального таймера.

Условное обозначение регистра	Название регистра	Тип доступа	Исходное состояние
ITCSR[2:0]	Регистр управления и состояния	W/R	0
ITPERIOD[31:0]	Регистр периода	W/R	FFFF_FFFF
ITCOUNT[31:0]	Регистр счетчика основного делителя частоты	W/R	0000_0000
ITSCALE[7:0]	Регистр предделителя частоты	W/R	0000

Формат регистра ITCSR приведен в Таблица 4.2.

Таблица 4.2. Формат регистра ITCSR.

Номер разряда	Условное обозначение	Описание
0	EN	Разрешение работы таймера: 0 – запрещение работы (неактивное состояние таймера); 1 – разрешение работы (активное состояние таймера).
1	INT	Признак срабатывания таймера. Состояние данного разряда транслируется в бит Timer регистра QSTR (на входе этого регистра он объединяется по логическому "или" с одноименными разрядами регистров управления и состояния таймеров WDT и RTT). Сбрасывается при записи нуля в этот разряд.

8-разрядный регистр ITSCALE используется для задания коэффициента предделения тактовой частоты CPU (CLK), которая поступает на вход счетчика SCOUNT.

32-разрядные регистр ITPERIOD используется для задания периода работы основного делителя.

32-разрядный счетчик основного делителя частоты ITCOUNT работает в режиме декремента. На вход этого счетчика поступает частота (S_CLK) с выхода счетчика предделителя.

4.4 Программирование IT.

Перед началом работы с интервальным таймером необходимо загрузить значение периода в регистр ITPERIOD и значение коэффициента предделения частоты в регистр ITSCALE.

Для активизации таймера необходимо в бит EN регистра ITCSR записать 1. В момент этой записи содержимое регистров ITSCALE и ITPERIOD переписывается в счетчики SCOUNT и ITCOUNT соответственно. После этого оба счетчика начинают работать в режиме декремента. При этом предделитель работает от частоты CLK, а счетчик ITCOUNT – от частоты S_CLK, формируемой предделителем.

Когда оба счетчика SCOUNT и ITCOUNT достигают нулевого состояния, в регистре ITCSR устанавливается бит INT и формируется запрос на прерывание QSTR[29] (бит TIMER), а содержимое регистров ITSCALE и ITPERIOD опять переписывается в счетчики SCOUNT и ITCOUNT соответственно. Далее таймер работает аналогичным образом.

Запрос на прерывание формируется каждые {(itperiod + 1)*(itscale + 1)} тактов работы CPU, где itperiod и itscale – содержимое регистров ITPERIOD и ITSCALE соответственно.

При необходимости, в любой момент времени в ITCOUNT и ITPERIOD можно произвести запись новых данных и тем самым изменить значение отрабатываемого временного интервала.

5. ТАЙМЕР РЕАЛЬНОГО ВРЕМЕНИ

5.1 Назначение

Таймер реального времени (RTT) предназначен для выработки периодических прерываний на основе деления внешней тактовой частоты RTCXTI. Основные характеристики таймера реального времени:

- Число разрядов делителя – 32;
- Программное управление стартом и остановкой таймера;
- Доступ ко всем регистрам обеспечивается в любой момент времени.

5.2 Структурная схема RTT

Структурная схема RTT представлена на Рисунок 5.1.

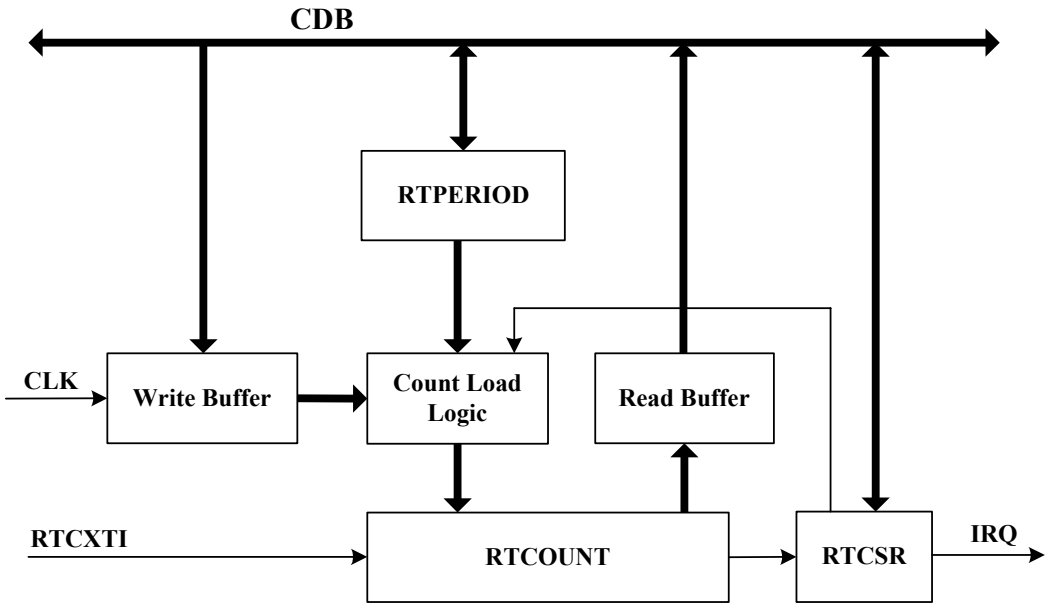


Рисунок 5.1. Структурная схема RTT.

В состав таймера реального времени входят следующие основные узлы:

- RTCSR - регистр управления и состояния;
- RTCOUNT - счетчик основного делителя;
- RTPERIOD - регистр периода основного делителя;
- Count Load Logic - логика загрузки счетчика основного делителя;
- Write Buffer – буфер записи;
- Read Buffer – буфер чтения.

На структурной схеме интервального таймера использованы следующие обозначения:

- CDB – шина данных CPU;
- CLK – тактовая частота работы CPU;
- RTCXTI – внешняя тактовая частота;
- IRQ – запрос на прерывание от таймера реального времени.

На вход таймера реального времени поступает внешняя тактовая частота RTCXTI. Для правильной работы RTT должно выполняться соотношение: $f_{RTCXTI} \leq \frac{f_{CLK}}{7}$, где f_{RTCXTI} и f_{CLK} значения частот RTCXTI и CLK соответственно. Как правило, RTCXTI имеет частоту 32,768 кГц.

5.3 Описание регистров таймера реального времени

В Таблица 5.1. приведен перечень программно-доступных регистров RTT.

Таблица 5.1. Перечень регистров RTT.

Условное обозначение регистра	Название регистра	Тип доступа	Исходное состояние
RTCSR[1:0]	Регистр управления и состояния	W/R	0
RTPERIOD[31:0]	Регистр периода	W/R	0000_7FFF
RTCOUNT[31:0]	Регистр счетчика делителя	W/R	0000_0000

Формат регистра RTCSR приведен в Таблица 5.2.

Таблица 5.2. Формат регистра RTCSR.

Номер Разряда	Условное обозначение	Описание
0	EN	Разрешение работы таймера: 0 – запрещение работы (неактивное состояние таймера); 1 – разрешение работы (активное состояние таймера).
1	INT	Признак срабатывания таймера. Состояние данного разряда транслируется в бит Timer регистра QSTR (на входе этого регистра он объединяется по логическому «или» с одноименными разрядами регистров управления и состояния таймеров WDT и IT). Сбрасывается при записи нуля в этот разряд.

32-разрядные регистр RTPERIOD используется для задания периода работы таймера. Если RTPERIOD = 0000_7FFF, а частота RTCXTI = 32,768 кГц, то таймер реального времени формирует прерывание каждую секунду.

32-разрядный счетчик RTCOUNT работает в режиме декремента от частоты RTCXTI.

5.4 Программирование RTT.

Перед началом работы с таймером необходимо загрузить данные в регистр RTPERIOD.

Для активизации таймера необходимо в бит EN регистра RTCSR записать 1. В момент этой записи содержимое регистра RTPERIOD переписывается в счетчик RTCOUNT, который начинает работать в режиме декремента. Когда счетчик RTCOUNT достигнет нулевого состояния, в регистре RTCSR устанавливается бит INT и формируется запрос на прерывание QSTR[29] (бит TIMER), а содержимое регистра RTPERIOD опять переписывается в счетчик RTCOUNT. Далее таймер работает аналогичным образом.

При необходимости, в любой момент времени в RTPERIOD и RTCOUNT можно произвести запись новых данных и тем самым изменить значение, отрабатываемого временного интервала.

Следует отметить, что при записи в RTCOUNT, обновление его содержимого происходит с задержкой, равной периоду RTCXTI.

6. СТОРОЖЕВОЙ ТАЙМЕР

6.1 Назначение

Сторожевой таймер (WDT) предназначен для:

- вывода системы из зависания, если программное обеспечение зациклилось и не формирует соответствующих управляющих воздействий;
- выработки прерываний на основе деления тактовой частоты CPU.

Основные характеристики таймера:

- число разрядов основного делителя – 32;
- число разрядов предделителя – 8;
- программное управление стартом и остановкой таймера;
- два режима работы: режим сторожевого таймера (WDM) и режим интервального таймера (ITM);
- два режима отработки временных интервалов: однократный и периодический;
- доступ ко всем регистрам обеспечивается в любой момент времени.

6.2 Структурная схема

Структурная схема сторожевого таймера приведена на Рисунок 6.1.

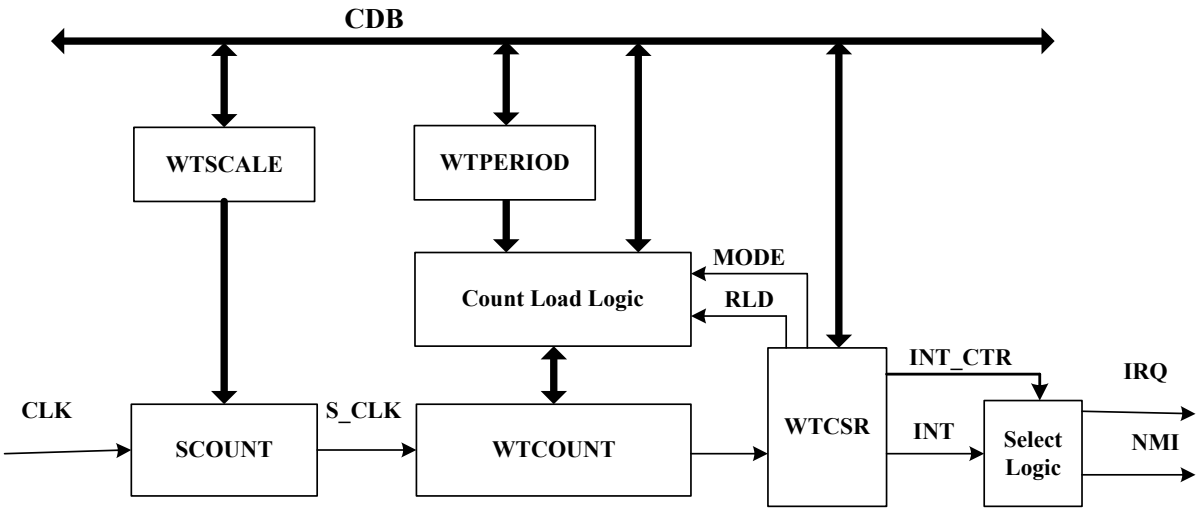


Рисунок 6.1. Структурная схема сторожевого таймера.

В состав сторожевого таймера входят следующие основные узлы:

- WTCSR - регистр управления и состояния;
- WTCOUNT - счетчик основного делителя;
- WTPERIOD - регистр периода основного делителя;
- WTSCALE - регистр предделителя;
- SCOUNT – счетчик предделителя;
- Count Load Logic - логика загрузки счетчика основного делителя.

На структурной схеме интервального таймера использованы следующие обозначения:

- CDB – шина данных CPU;
- CLK – тактовая частота работы CPU;
- S_CLK – выходная частота предделителя;
- IRQ – запрос на прерывание от интервального таймера;
- NMI – немаскируемое прерывание.

6.3 Описание регистров WDT

В Таблице 6.1 приведен перечень программно-доступных регистров WDT.

Таблица 6.1. Перечень программно-доступных регистров WDT.

Условное обозначение регистра	Название регистра	Тип доступа	Исходное состояние
WTCSR[14:0]	Регистр управления и состояния	W/R	0000
WTPERIOD[31:0]	Регистр периода	W/R – в неактивном состоянии; R – в активном состоянии.	FFFF_FFFF
WTCOUNT[31:0]	Регистр счетчика основного делителя частоты	W/R – в неактивном состоянии; R – в активном состоянии.	0000_0000
WTSCALE[15:0]	Регистр предделителя частоты	W/R – в неактивном состоянии; R – в активном состоянии.	0000

8-разрядный регистр WTSCALE используется для задания коэффициента предделения тактовой частоты CPU (CLK), которая поступает на вход счетчика SCOUNT.

32-разрядные регистр WTPERIOD используется для задания периода работы основного делителя.

32-разрядный счетчик основного делителя частоты WTCOUNT работает в режиме декремента. На вход этого счетчика поступает частота S_CLK с выхода счетчика предделителя.

Формат регистра WTCSR приведен в Таблице 6.2.

Таблица 6.2. Формат регистра WTCSR.

Номер разряда	Условное обозначение	Описание
7: 0	KEY	Поле для записи ключей. Запись в это поле последовательности кодов A0 (ключ KEY1) и F5 (ключ KEY2) приводит к переключению таймера из режима сторожевого таймера (WDM) в режим интервального таймера (ITM). Поле доступно по чтению и записи. Поле доступно по записи только в режиме WDM: когда EN=1 или когда таймер находится в состоянии Timeout. Сбрасывается в ноль при переводе таймера из режима ITM в режим WDM. Значение в исходном состоянии – 0.
8	EN	Разрешение работы таймера: 0 – запрещение работы (неактивное состояние таймера); 1 – разрешение работы (активное состояние таймера). Доступен по чтению и записи. Запись нуля в этот бит при работе таймера в режиме WDM не имеет эффекта. Значение в исходном состоянии – 0.
9	INT	Признак срабатывания таймера. В зависимости от содержимого поля INT_CTR состояние данного разряда транслируется или в бит Timer регистра QSTR (на входе этого регистра он объединяется по логическому «или» с одноименными разрядами регистров управления и состояния таймеров RTT и IT), или в немаскируемое прерывание (NMI). Сбрасывается при записи нуля в этот разряд, а также при переводе таймера из режима ITM в режим WDM. Доступен по чтению и записи в режиме ITM и только по чтению в режиме WDM. Значение в исходном состоянии – 0.

Продолжение Таблицы 6.2

Номер разряда	Условное обозначение	Описание
10	MODE	Режим работы таймера: 0 – режим сторожевого таймера (WDM); 1 – режим обычного таймера (ITM). Доступен по чтению и записи при EN=0 и только по чтению при EN=1. Значение в исходном состоянии – 0.
11	RLD	Бит управления перезагрузкой SCOUNT и WTCOUNT при работе в режиме ITM: 0 – таймер однократно отрабатывает временной интервал и останавливается; 1 – таймер отрабатывает заданный временной интервал периодически. После отработки очередного временного интервала содержимое WTSCALE и WTPERIOD загружается в SCOUNT и WTCOUNT соответственно. Доступен по чтению и записи при EN=0 и только по чтению при EN=1. Значение в исходном состоянии – 0.
13: 12	INT_CTR	Управления типом прерывания, которое формируется таймером WDT: 00 – прерывание не формируется; 01 – обычное прерывание (QSTR[29]). Как правило, используется в режиме ITM; 10 – немаскируемое прерывание (NMI). Как правило, используется в режиме WDM. 11 – прерывание не формируется. Формируется внешний сигнал WDT (см. табл. 15.2). Поле доступно по чтению и записи при EN=0 и только по чтению при EN=1. Значение в исходном состоянии – 0.

6.4 Программирование WDT

Диаграмма состояний WDT приведена на рис 6.2.

В исходном состоянии WDT находится в режиме сторожевого таймера. Для перевода его в режим интервального таймера необходимо записать 1 в бит MODE регистра WTCSR. Следует отметить, что смена режима работы таймера посредством записи в бит MODE возможна, если таймер не активен (EN=0).

Перед началом работы с таймером WDT необходимо загрузить значение периода в регистр WTPERIOD и значение коэффициента предделения частоты в регистр WTSCALE.

Для активизации таймера необходимо в бит EN регистра WTCSR записать 1. В момент этой записи содержимое регистров WTSCALE и WTPERIOD переписывается в счетчики SCOUNT и WTCOUNT соответственно. После этого оба счетчика начинают работать в режиме декремента. При этом предделитель работает от частоты CLK, а счетчик WTCOUNT – от частоты S_CLK, формируемой предделителем.

После активизации таймера, WTCOUNT, WTPERIOD, WTSCALE, а также поля INT_CTR, MODE, RLD регистра WTCSR, становятся не доступными по записи.

Сторожевой таймер в режиме WDM необходимо периодически обслуживать. То есть, если он был активизирован в режиме WDM, то для того, чтобы не возникло состояния Timeout необходимо периодически выполнять следующую последовательность действий:

- переключить таймер из режима WDM в режим ITM посредством последовательной записи в поле KEY регистра WTCSR кодов A0 и F5;
- остановить таймер посредством записи 0 в бит EN регистра WTCSR;
- установить MODE=0;

Если вслед за значением A0 в поле KEY будет записано значение ≠ F5, то таймер перейдет в состояние Timeout.

Если после активизации таймера в режиме WDM, он не будет переведен в режим ITM, то, когда оба счетчика SCOUNT и WTCOUNT достигнут нулевого значения, таймер перейдет в состояние Timeout.

В состоянии Timeout таймер формирует признак INT и останавливается, а запись в какой-либо из его регистров блокируется. Для вывода WDT из состояния Timeout необходимо его переключить в режим ITM посредством последовательной записи в поле KEY регистра WTCSR кодов A0 и F5.

При переключении таймера из неактивного состояния в режиме ITM в режим WDM путем записи 0 в поле MODE регистра WTCSR происходит обнуление полей KEY и INT.
При работе таймера в режиме ITM при RLD=0 он однократно отрабатывает заданный временной интервал, устанавливает INT=1 и останавливается (когда оба счетчика SCOUNT и WTCOUNT достигают нулевого состояния). Если RLD=1, то каждый раз после достижения счетчиками нулевого состояния и установки INT=1, происходит перезагрузка значений периода и коэффициента предделения частоты. То есть, таймер

отрабатывает заданный временной интервал периодически до тех пор, пока он не будет остановлен.

Запрос на прерывание формируется каждые {(wtperiod + 1)*(wtscale + 1)} тактов работы CPU, где wtperiod и wtscale – содержимое регистров WTPERIOD и WTSCALE соответственно.

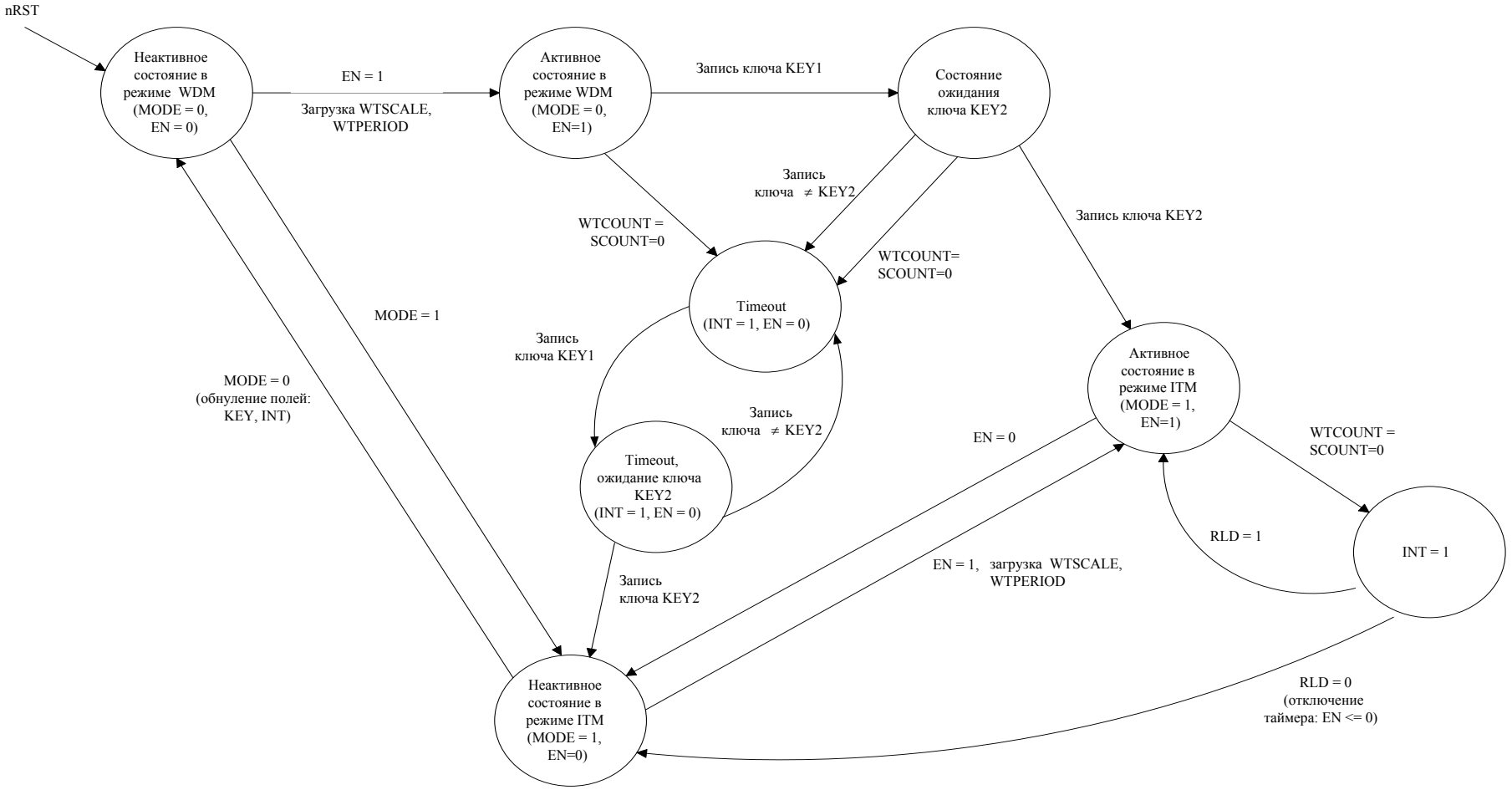


Рисунок 6.2. Диаграмма состояний WDT.

7. КОНТРОЛЛЕР ИНТЕРФЕЙСА SPACE WIRE

7.1 Введение

Контроллер интерфейса SpaceWire (далее по тексту - SWIC) предназначен для обеспечения аппаратной поддержки функций внутрисистемных коммуникаций с использованием протокола SpaceWire, Блок контроллера канала SWобеспечивает дуплексную прием-передачу последовательных данных по стандарту SpaceWire.

7.2 Особенности

- Контроллер разработан в соответствии с международным стандартом ECSS-E-50-12;
- Обеспечивает функционирование одного дуплексного канала связи со скоростью от 2 до 400 Мбит/с в каждую сторону;
- Реализация контроллера охватывает уровни стека протоколов SpaceWire, от сигнального до сетевого (частично) уровня;
- Аппаратное детектирование ошибок связи: рассоединение, ошибки четности;
- Встроенные LVDS приемопередатчики в соответствии со стандартом стандарта ANSI/TIA/EIA-644(LVDS);
- Встроенные в приемник LVDS резисторы-терминаторы;
- Контроллер имеет интерфейс с шиной AMBA AHB согласно документу "AMBA Specification" ver.2.0;
- Содержит в своем составе SLAVE и MASTER устройства;
- Содержит десятиразрядный регистр управления синтезатором частоты передачи;
- Четыре канала DMA (два канала данных и два канала дескрипторов пакетов);
- Обмен данными через DMA с памятью словами по 32 бита;
- Четыре линии прерываний;
- Встроенный буфер передаваемых данных на шестнадцать 32-х разрядных слов;
- Встроенный буфер принимаемых данных на четыре 32-х разрядных слова;

7.3 Структура контроллера

Структура контроллера коммуникационного канала по стандарту SpaceWire приведена на Рисунок 7.1. Основой контроллера канала SW является DS-макроячейка, реализующая функции кодера/декодера SpaceWire. Кодер/декодер SpaceWire через драйверы LVDS подключен к физическим линиям связи.

Контроллер канала SW взаимодействует с центральным процессором и системной оперативной памятью через шину AMBA AHB. На этой шине он представлен интерфейсами ведущего и ведомого устройства. Через интерфейс ведомого устройства ЦП может осуществлять чтение и запись регистров контроллера для определения его состояния и настройки параметров работы. Через интерфейс ведущего устройства контроллер выполняет запись данных в оперативную память хост-системы, полученных из

канала SW и чтение из системной памяти данных, которые необходимо передать в канал.

Блок управления по командам центрального процессора задает режимы работы приемопередатчика SpaceWire. В этом блоке содержатся программно управляемый регистр, содержащий коэффициент скорости передачи данных, и доступный программному обеспечению на чтение регистр, в который записывается коэффициент скорости приема данных. В стандарте SpaceWire поддерживается передача маркеров системного времени; состояние последнего полученного извне маркера хранится в соответствующем регистре блока управления и может быть считано хост-системой.

В контроллере обеспечивается промежуточное запоминание в коммуникационной буферной памяти принимаемых и передаваемых данных через DS-макроячейку, что позволяет согласовать скорость приема-передачи данных в канале с пропускной способностью шины AMBA. Блок буферной памяти реализуется на основе двухпортового ОЗУ. Управление приемом данных в буфер из DS-макроячейки и передачей данных из буфера в DS-макроячейку осуществляют соответственно блоки приема и передачи, которые взаимодействуют с блоками двухпортового ОЗУ.

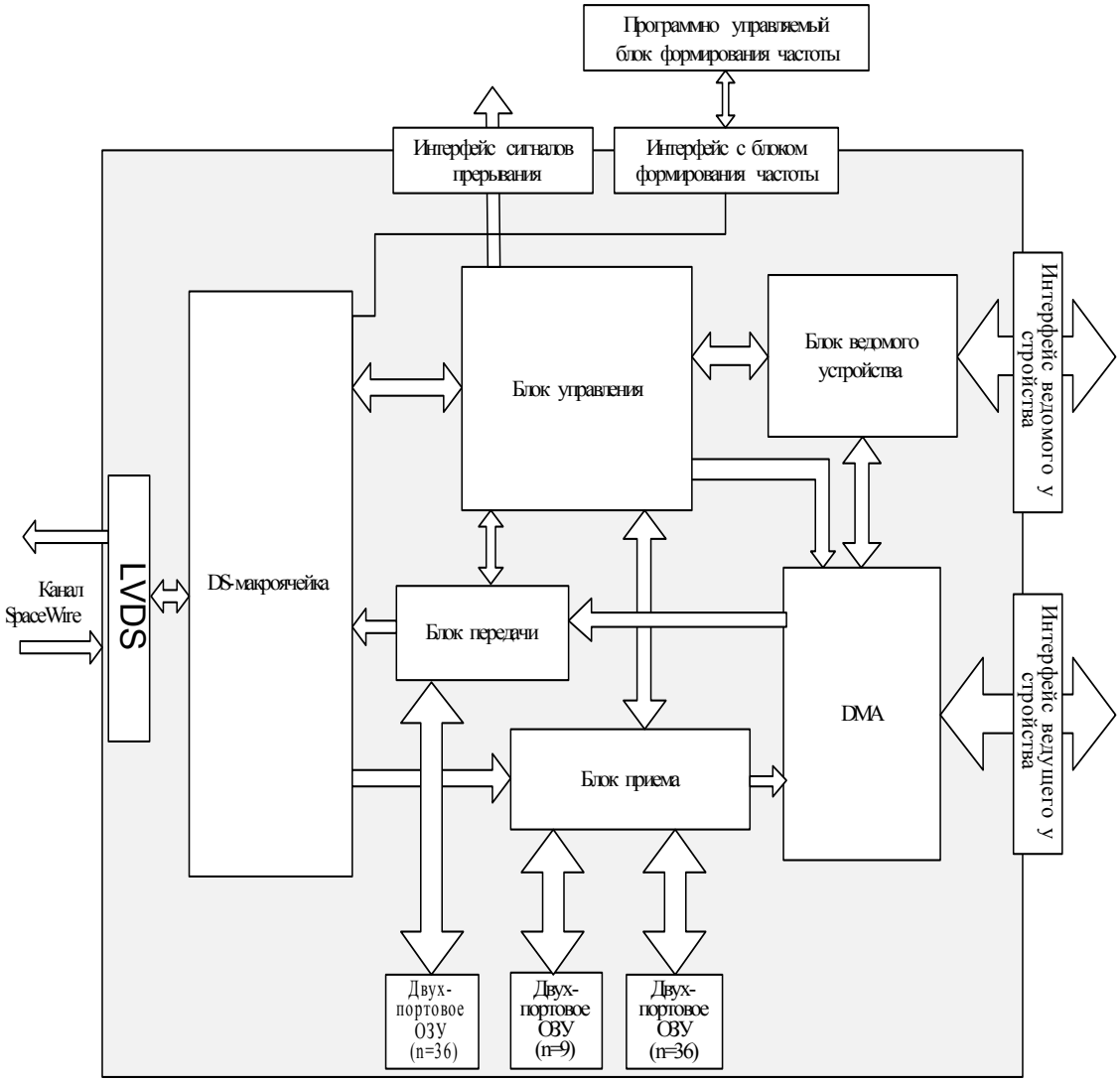


Рисунок 7.1 Структура контроллера SWIC

Буфер приема имеет конвейерную организацию и состоит из двух ступеней. Сначала буферизируются восьмиразрядные данные, принимаемые от DS-макроячейки (один разряд служебный). Затем для более эффективного использования пропускной способности шины AMBA в буфере формируются 32-разряные слова данных (и дополнительно четыре служебных бита). В буфер передачи с помощью канала передаваемых данных DMA с шины AMBA записываются 32-разряные слова данных (четыре разряда также используются для хранения формируемых блоком передачи служебных битов). Данные из буфера передачи в DS-макроячейку выдаются побайтно.

Каналы DMA через интерфейс ведущего устройства с шиной AMBA реализуют обмен данными между оперативной памятью хост-системы и коммуникационной буферной памятью контроллера.

Используются четыре канала DMA:

- TxChDes - канал дескрипторов передаваемых пакетов;
- TxChDat - канал данных передаваемых пакетов;
- RxChDes - канал дескрипторов принимаемых пакетов;
- RxChDat - канал данных принимаемых пакетов.

Для эффективной поддержки механизмов, обеспечивающих управление процессов обмена данными в соответствии со стандартом, заложенным в архитектуре виртуального интерфейса (VIA – Virtual Interface Architecture), в контроллере содержится четыре канала DMA – по два в каждом направлении (два канала на чтение из памяти и два канала на запись).

7.4 Регистры SWIC

Управление контроллером осуществляется через набор регистров, доступных для чтения процессору. Перечень программно-доступных регистров блока приведен в Таблица 3.1.

Таблица 7.1 Перечень регистров SWIC.

Условное обозначение регистра	Назначение
HW_VER[31:0]	Номер аппаратной версии контроллера
STATUS[11:0]	Регистр состояния
RX_TIME[7:0]	Регистр маркера времени из сети
MODE_CR[6:0]	Регистр режима работы
TX_SPEED[9:0]	Регистр коэффициента скорости передачи
TX_TIME[7:0]	Регистр маркера времени для передачи в сеть
CNT_TX_PACK[31:0]	Регистр счетчика переданных пакетов
RX_SPEED[7:0]	Регистр коэффициента скорости приема
ADDR_RxChDes[31:0]	Регистр адреса блока памяти канала RxChDes
ADDR_RxChDat[31:0]	Регистр адреса блока памяти канала RxChDat
ADDR_TxChDes[31:0]	Регистр адреса блока памяти канала TxChDes
ADDR_TxChDat[31:0]	Регистр адреса блока памяти канала TxChDat
SIZE_RxChDes[15:0]	Регистр размера блока памяти канала RxChDes
SIZE_RxChDat[15:0]	Регистр размера блока памяти канала RxChDat
SIZE_TxChDes[15:0]	Регистр размера блока памяти канала TxChDes
SIZE_TxChDat[15:0]	Регистр размера блока памяти канала TxChDat
CR_DMA[5:0]	Регистр управления каналами DMA
MAX_TR_SIZE[31:0]	Регистр максимального размера транзакции

Условное обозначение регистра	Назначение
PTR_RxChDat[31:0]	Регистр указателя на текущий адрес цепочки канала RxChDat
PTR_TxChDat[31:0]	Регистр указателя на текущий адрес цепочки канала TxChDat
SR_DMA[9:0]	Регистр состояния каналов DMA

7.4.1 Регистр HW_VER

Регистр номера версии [31..0]. При чтении этого регистра выводится номер версии аппаратной реализации SWIC. В данной микросхеме применен SWIC версии 0x0000 0002.

7.4.2 Регистр STATUS

Регистр состояния блока SWIC предназначен для оперативного контроля состояния фаз работы контроллера. Регистр доступен как на чтение, так и на запись. Заполнение регистра выполняется побитно по сигналам от DS-макроячейки, блока приема данных из канала SpaceWire, блока передачи данных в канал SpaceWire.

Таблица 7.2 Назначение разрядов регистра STATUS

Номер разряда	Условное обозначение	Назначение
0	DC_ERR	Признак ошибки рассоединения (DisconnectError): "1" – Ошибка произошла "0" – Нет ошибки (после сигнала сброса) Запись "1" в этот разряд сбрасывает этот разряд в "0"
1	P_ERR	Признак ошибки четности: "1" – Ошибка произошла "0" – Нет ошибки (после сигнала сброса) Запись "1" в этот разряд сбрасывает этот разряд в "0"
2	ESC_ERR	Признак ошибки в ESC последовательности: "1" – Ошибка произошла "0" – Нет ошибки (после сигнала сброса) Запись "1" в этот разряд сбрасывает этот разряд в "0"

Номер разряда	Условное обозначение	Назначение
3	CREDIT_ERR	Признак ошибки кредитования: "1" – Ошибка произошла "0" – Нет ошибки (после сигнала сброса) Запись "1" в этот разряд сбрасывает этот разряд в "0"
4	RX_BUF_FULL	Буфер приема полон: "1" – Заполнен полностью "0" – Не полон или пуст (после сигнала сброса)
5	RX_BUF_EMPTY	Буфер приема пуст "1" – Пуст (после сигнала сброса) "0" – В буфере есть данные
6	DSM_ERR_RST	DS-макроячейка находится в состоянии ErrorReset "1" – Состояние ErrorReset (после сигнала сброса) "0" – Иное состояние макроячейки
7	TX_BUF_FULL	Буфер передачи полон "1" – Заполнен полностью "0" – Не полон или пуст (после сигнала сброса)
8	TX_BUF_EMPTY	Буфер передачи пуст "1" – Пуст (после сигнала сброса) "0" – В буфере есть данные
9	CONNECTED	Состояние установки соединения "1" – Соединение установлено (DS-макроячейка находится в состоянии Run) "0" – Иное состояние DS-макроячейки (после сигнала сброса) Запись "1" в этот разряд сбрасывает этот разряд в "0"
10	GOT_EOP	Принят конец пакета "1" – Принят символ конца пакета (EOP или EEP) "0" – Символ конца пакета не принят (после сигнала сброса) Запись "1" в этот разряд сбрасывает этот разряд в "0"

Номер разряда	Условное обозначение	Назначение
11	GOT_TIME	Принят маркер времени из сети "1" – Принят маркер времени "0" – Марке времени не принят (после сигнала сброса) Запись "1" в этот разряд сбрасывает этот разряд в "0"
12..31	-	Резерв. Оставлено для будущих применений. Содержит "0..0"

7.4.3 Регистр RX_TIME

Регистр принятого из сети маркера времени. Биты регистра [0:5] собственно принятый маркер времени. Биты [31:6] читаются как "0". После начального сброса содержимое регистра 0x00000000. Содержимое регистра сохраняется до принятия следующего маркера времени и обновляется автоматически. Процессор может прочитать содержимое этого регистра после получения прерывания INT_TIME или в любой другой момент времени.

7.4.4 Регистр MODE_CR

Таблица 7.3 Назначение разрядов регистра MODE_CR

Номер разряда	Условное обозначение	Назначение
0	LinkDisabled	Установка LinkDisabled для блока DS-кодирования
1	AutoStart	Установка Autostart для блока DS-кодирования
2	LinkStart	Установка LinkStart для блока DS-кодирования
3	RX_RST	Сброс блока приема "1" – предустановка в исходное состояние (по сигналу сброса) "0" – Разрешение работы
4	TX_RST	Сброс блока передачи "1" – предустановка в исходное состояние (по сигналу сброса) "0" – Разрешение работы

Номер разряда	Условное обозначение	Назначение
5	DSM_RST	Сброс DS-макроячейки “1” – предустановка в исходное состояние (по сигналу сброса) “0” – Разрешение работы
6	DMA_RST	Сброс DMA “1” – предустановка в исходное состояние (по сигналу сброса) “0” – Разрешение работы
7	SWCORE_RST	Программный сброс контроллера “1” – предустановка в исходное состояние (по сигналу сброса) “0” – Разрешение работы
8	WORK_TYPE	Тип режима работы (‘1’ – рабочий, ‘0’ – тестовый)
9..31	-	Резерв. Оставлено для будущих применений

Рекомендуется разрешать AutoStart и/или LinkStart только после того, как настроены каналы DMA. После того, как в результате разрешения AutoStart или LinkStart блок DS-кодирования установил соединение, буфер передачи в сеть начинает читать данные из канала DMA TXD_Ch. Если из DMA прочитаны все данные, то далее в сеть передаются Null. Соединение при этом не прекращается. Соединение прекращается, если процессор осуществляет запись единицы в бит LinkDisabled.

В рабочем режиме (WORK_TYPE=1), согласно стандарту SpaceWire, ошибки, возникающие до установки соединения, процессору не выдаются (не возникают прерывания, не устанавливаются биты регистра состояния (STATUS)). При тестовом режиме работы (WORK_TYPE=0), почти все ошибки, возникающие до установки соединения, выдаются процессору (возникают прерывания, устанавливаются биты регистра состояния (STATUS)). В данной версии контроллера тестовый режим реализован не полностью. На ошибки типа ‘принят неожиданный символ’, ‘истёк тайм-аут 12.8 мкс’ в тестовом режиме не выдаются прерывания и не устанавливаются биты в регистре состояния (STATUS).

7.4.5 Регистр TX_SPEED

Регистр управления блоком PLL для установления скорости передачи.

Таблица 7.4 Назначение разрядов регистра TX_SPEED

Номер разряда	Условное обозначение	Назначение
0:7	TX_SPEED	Определяет скорость передачи данных
8:9	PLL_CTR	Разрешение работы PLL. "11" – работа PLL разрешена. "00" – работа PLL запрещена (по сигналу сброса)
10..31	-	Резерв. Оставлено для будущих применений

Скорость передачи (Мбит/сек) определяется как произведение множителя [7:0]TX_SPEED на опорную частоту PLL 2МГц, деленную пополам. Например, для получения скорости передачи 10Мбит/сек в регистр TX_SPEED необходимо записать константу 0x0000003A. Заказанная скорость передачи установится за время не более 100мС.

7.4.6 Регистр TX_TIME

Регистр маркера времени для передачи в канал [7..0]. При передаче маркера времени разряды 7 и 6 должны быть равными '0'. Сразу же после записи в регистр TX_TIME начинается передача в DS-макроячейку и далее в канал содержимого этого регистра.

7.4.7 Регистр CNT_TX_PACK

Регистр счетчика переданных пакетов [31..0]. Регистр предназначен для определения количества пакетов, полностью переданных в сеть. Значение регистра увеличивается на 1 каждый раз, когда в DS макроячейку передается символ конца пакета.

При записи, значение регистра обнуляется. Процессор может обнулить содержимое этого регистра для того, чтобы начать счет пакетов заново. Рекомендуется выполнять сброс регистра каждый раз при выполнении новой настройки DMA для передачи данных в сеть.

7.4.8 Регистр RX_SPEED

Регистр коэффициента скорости приема [8..0]. Значение регистра обновляется каждые 200 тактов базовой частоты в 200 МГц. В течение 200 тактов базовой частоты подсчитывается число тактов входной частоты и обновляется значение регистра. Если установка скорости придётся в середине этого замера в 200 тактов, то к концу 200 тактов значение скорости в регистре не будет соответствовать действительности (была посчитана часть старой частоты и часть новой). Значение регистра окончательно обновится через следующие 200 тактов. Таким образом, новое значение скорости окажется в регистре за время 1-2 мкс. Скорость приема соответствует считанному значению регистра RX_SPEED. Например, из регистра считано значение 0x00000014, что соответствует скорости прием а 20Мбит/сек.

7.4.9 Регистры ADDR_RxChDes, ADDR_RxChDat, ADDR_TxChDes и ADDR_TxChDat

Четыре 32-х разрядных регистра имеющих одинаковое предназначение для четырех каналов DMA. Регистры предназначены для установки начального адреса блока памяти, выделенного для передаваемых или принимаемых данных, в соответствии с обозначением регистра. По мере чтения/записи слов данных из/в память каналом DMA, значение этих регистров инкрементируется каналом. Регистры доступны на чтение для контроля процессов передачи и приема данных.

ADDR_RxChDes - регистр начальной установки адреса блока памяти, выделенного для чтения каналу DMA RxChDes.

ADDR_RxChDat - регистр начальной установки адреса блока памяти, выделенного для чтения каналу DMA RxChDat.

ADDR_TxChDes - регистр начальной установки адреса блока памяти, выделенного для записи каналу DMA TxChDes.

ADDR_TxChDat - регистр начальной установки адреса блока памяти, выделенного для записи каналу DMA TxChDat.

7.4.10 Регистры SIZE_RxChDes, SIZE_RxChDat, SIZE_TxChDes и SIZE_RxChDat

В эти регистры производится запись размеров блоков памяти выделенных процессором для работы соответствующих каналов DMA. По мере чтения (записи) слов данных значение этих регистров декрементируется каналом. Когда значение регистра равно нулю – блок памяти соответствующего канала DMA полностью обработан.

SIZE_RxChDes - регистр начальной установки размера блока памяти, выделенного для чтения каналу DMA RxChDes.

SIZE_RxChDat - регистр начальной установки размера блока памяти, выделенного для чтения каналу DMA RxChDat.

SIZE_TxChDes - регистр начальной установки размера блока памяти, выделенного для записи каналу DMA TxChDes.

SIZE_TxChDat - регистр начальной установки размера блока памяти, выделенного для записи каналу DMA TxChDat.

7.4.11 Регистр CR_DMA

Регистр управления всеми каналами DMA [5..0]. Состояние регистра определяется процессором через интерфейс ведомого устройства. Назначение разрядов регистра управления сведено в Таблица 7.5.

Таблица 7.5 Назначение разрядов регистра CR_DMA

Номер разряда	Условное обозначение	Назначение
0	EN_RxChDes	Разрешение работы канала DMA RxChDes
1	EN_RxChDat	Разрешение работы канала DMA RxChDat
2	EN_TxChDes	Разрешение работы канала DMA TxChDes
3	EN_TxChDat	Разрешение работы канала DMA TxChDat
4	S_INIT_RxChDat	Разрешение самоинициализации канала RxChDat
5	S_INIT_ TxChDat	Разрешение самоинициализации канала TxChDat
6-31	Резерв	Оставлено для будущих применений

7.4.12 Регистр MAX_TR_SIZE

Регистр максимального размера транзакции [31..0]. Регистр используется для ограничения максимального размера передачи данных через шину AMBA. При записи значения 0x0000.0000 в этот регистр, устройства DMA будут обмениваться с памятью пакетами неограниченной длинны. Этот режим является рекомендуемым.

7.4.13 Регистры PTR_RxChDat, PTR_TxChDat

Запись начального адреса цепочки осуществляется процессором через интерфейс ведомого устройства. По мере обработки блоков значение регистров инкрементируется.

Регистр PTR_RxChDat – указатель на текущий адрес цепочки канала RxChDat [31..0]

Регистр PTR_TxChDat – указатель на текущий адрес цепочки канала TxChDat [31..0]

7.4.14 Регистр SR_DMA

Регистр состояния каналов DMA [9..0]. Регистр предназначен для определения состояния каналов DMA и определения причин возникновения прерывания INT_DMA.

Таблица 7.6 Назначение разрядов регистра SR_DMA

Номер разряда	Условное обозначение	Назначение
0	RxChDes_END	Признак окончания обработки блока данных каналом DMA RxChDes, 1 – Обработка завершена
1	RxChDat_END	Признак окончания обработки блока данных каналом DMA RxChDat 1 – Обработка завершена

Номер разряда	Условное обозначение	Назначение
2	TxChDes_END	Признак окончания обработки блока данных каналом DMA TxChDes 1 – Обработка завершена
3	TxChDat_END	Признак окончания обработки блока данных каналом DMA TxChDat 1 – Обработка завершена
4	CHAIN_RxChDat_END	Признак окончания обработки цепочки блоков данных каналом RxChDat 1 – Обработка завершена
5	CHAIN_TxChDat_END	Признак окончания обработки цепочки блоков данных каналом TxChDat 1 – Обработка завершена
6	RxChDat_MEM_ERR	Признак ошибки обращения канала RxChDat к памяти 1 – Ошибка произошла
7	RxChDes_MEM_ERR	Признак ошибки обращения канала RxChDes к памяти 1 – Ошибка произошла
8	TxChDat_MEM_ERR	Признак ошибки обращения канала TxChDat к памяти 1 – Ошибка произошла
9	TxChDes_MEM_ERR	Признак ошибки обращения канала TxChDes к памяти 1 – Ошибка произошла
10-31	-	Резерв.

7.5 Логика работы SWIC

7.5.1 Передача данных из канала SW в системную память

Маршрут принимаемых данных и схема их обработки приведены на Рисунок 7.2. Из DS-линков в DS-макроячейку символы (байты) данных поступают последовательно (побитно). DS-макроячейка выделяет из последовательности приходящих символов символы данных и символы концов пакетов и передает их в буфер приема. По DS-линку байты данных передаются младшими разрядами вперед.

Передача всех разрядов символа (9 разрядов, из них 8 используется для представления собственно байта данных, девятый бит является дополнительным и указывает, является ли этот байт символом данных или символом конца пакета) от DS-макроячейки в буфер приема осуществляется в параллельном коде.

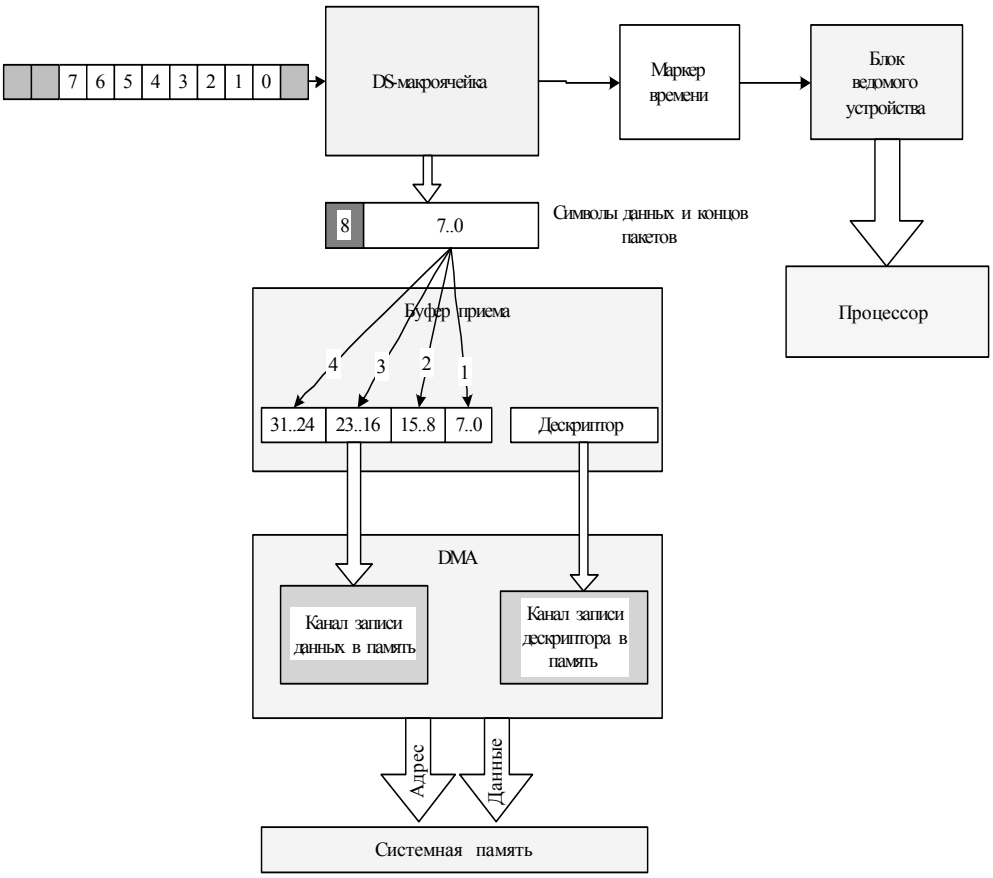


Рисунок 7.2 Передача данных из DS-линков в память MC-24R.

В буфере приема из байтов данных формируются слова разрядности 32 (в соответствии с разрядностью памяти и разрядностью шины AMBA AHB). При формировании слов первый поступивший байт размещается в разрядах 7:0, второй – в разрядах 15:8, третий – в разрядах 23:16 и четвертый – в разрядах 31:24.

Для того чтобы сократить загрузку процессора в ходе последующей обработки пакетов данных, в этом блоке выполняется выравнивание границ пакетов по границам слов и формирование дескрипторов пакетов, позволяющих процессору распознать границы отдельных пакетов.

Собственно пакеты данных и дескрипторы пакетов могут храниться в различных областях памяти. Местоположение этих областей в памяти определяется процессором при настройке каналов DMA. Дескрипторы пакетов записываются в память друг за другом и логически организованы в очередь.

7.5.2 Выравнивание границ пакетов по границам слов

Если очередное слово данных сформировано не полностью (действительными данными заполнены один, два или три байта слова), а следующий символ в последовательности – символ конца пакета, то заполнение данного слова прекращается. Первый символ следующего пакета будет записан в первый байт нового слова. Действительный размер пакета в байтах записывается в дескриптор пакета. Это позволяет процессору при обработке пакета исключить из рассмотрения “лишние” байты – байты, добавленные для выравнивания пакетов по границам слов. В дескриптор заносится также ин-

формация о типе конца пакета (нормальный конец пакета – EOP, или признак завершения пакета с ошибкой – EEP).

7.5.3 Формат дескриптора пакета

Дескриптор пакета имеет следующую структуру:

- 31 – признак заполнения дескриптора действительными данными. Бит учитывается только при приёме пакетов (позволяет процессору идентифицировать конец очереди дескрипторов в памяти). При передаче пакетов этот бит не учитывается (DMA вычитывает всю область дескрипторов, заданную процессором). До запуска приёма, все 31-е биты дескрипторов области приёма должны быть обнулены программно; DMA не обнуляет 31-е биты не принятых дескрипторов, DMA только записывает ‘1’ в 31-е биты принятых дескрипторов.
- 30:29 – тип конца пакета: 01 – EOP, 10 – EEP;
- 28:25 – зарезервировано («0000»)
- 24:0 – размер пакета в байтах.

Слова данных из буфера приема передаются в канал DMA записи данных в память. Дескрипторы из блока приема передаются в канал записи дескриптора в память DMA. Блок DMA записывает данные и дескрипторы в системную память в соответствии с настройками, выполненными процессором.

Процессор для канала записи дескрипторов в память определяет начальный адрес блока памяти и размер блока памяти. Для записи собственно пакетов данных в память может быть задан один блок памяти (так же, как и для канала записи дескриптора в память) или последовательность блоков памяти, физически расположенных в разных местах памяти.

При передаче данных в канал SpaceWire передается число дескрипторов (и пакетов), указанное в регистре размера блока дескрипторов на передачу.

7.5.4 Передача данных из системной памяти в канал SW

Процесс передачи пакетов данных из системной памяти в канал через контроллер, а также преобразование форматов данных показаны на Рисунок 7.3. Пакеты данных загружаются из системной памяти в буфер передачи через каналы DMA чтения данных из памяти и чтения дескриптора из памяти. Чтение выполняется 32-х разрядными словами (в соответствии с разрядностью памяти и шины AMBA AHB).

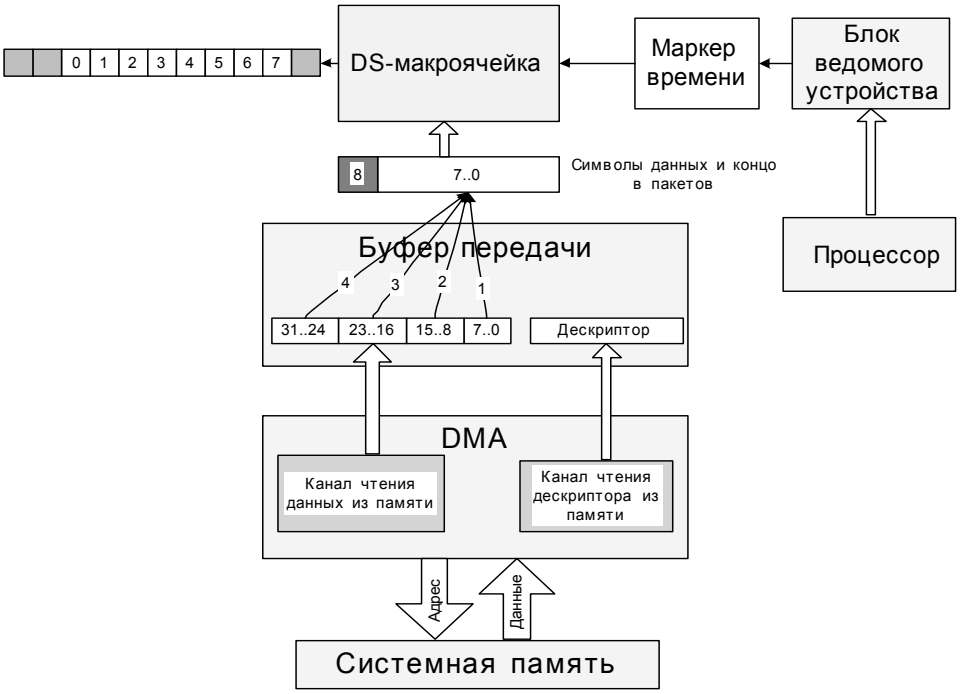


Рисунок 7.3 Передача данных из системной памяти в DS-линк

Буфер передачи разбивает слова на отдельные байты. При этом из последовательности байтов в соответствии с информацией, содержащейся в дескрипторе, удаляются “лишние” байты – байты, добавленные для выравнивания пакетов по границам слов, и вставляются символы концов пакетов EOP или EEP. Если в DS-линк передаются пакеты, сгенерированные в данном узле, то предполагается, что они всегда должны заканчиваться символом EOP. Однако пакеты могут проходить через данный процессорный модуль транзитом. В этом случае они могут заканчиваться символом EEP. Коды маркеров EOP или EEP формируются контроллером аппаратно, на основании кодов дескриптора пакета на передачу (разряды 29-30 дескриптора пакета). Сами дескрипторы пакетов на передачу в сеть из основной памяти формируются программно.

Распаковка 32-разрядного слова в последовательность из 4 байт при передаче из контроллера выполняется по правилу, согласованному с правилом упаковки байтов при приеме данных из канала в контроллер.

Буфер передачи вначале передает в DS-макроячейку байт данных, находящийся в разрядах 7:0 слова, затем байт, находящийся в разрядах 15:8, затем байт, находящийся в разрядах 23:15, затем байт из разрядов 31:24 тридцатидвухразрядного слова.

Символы данных и концов пакетов передаются блоком передачи в блок DS-макроячейки. DS-макроячейка преобразует полученные символы в соответствии с алгоритмом DS кодирования и передает их в канал. Символы передаются младшими разрядами вперед.

7.5.5 Представление данных в памяти.

Рассмотрим пример представления данных в системной памяти, если для данных выделен один сегмент памяти (Рисунок 7.4). Пусть в системную память из канала SpaceWire было записано 3 пакета. Первый пакет имеет размер 10 байт и заканчивается символом EOP. Второй пакет имеет размер 8 байт и заканчивается символом EEP. Третий пакет имеет размер 11 байт и заканчивается символом EOP. Собственно пакеты хранятся в сегменте памяти, выделенном процессором для записи данных. Первый и третий пакет дополнены двумя и одним байтом соответственно, для выравнивания по границам 32-х разрядных слов.

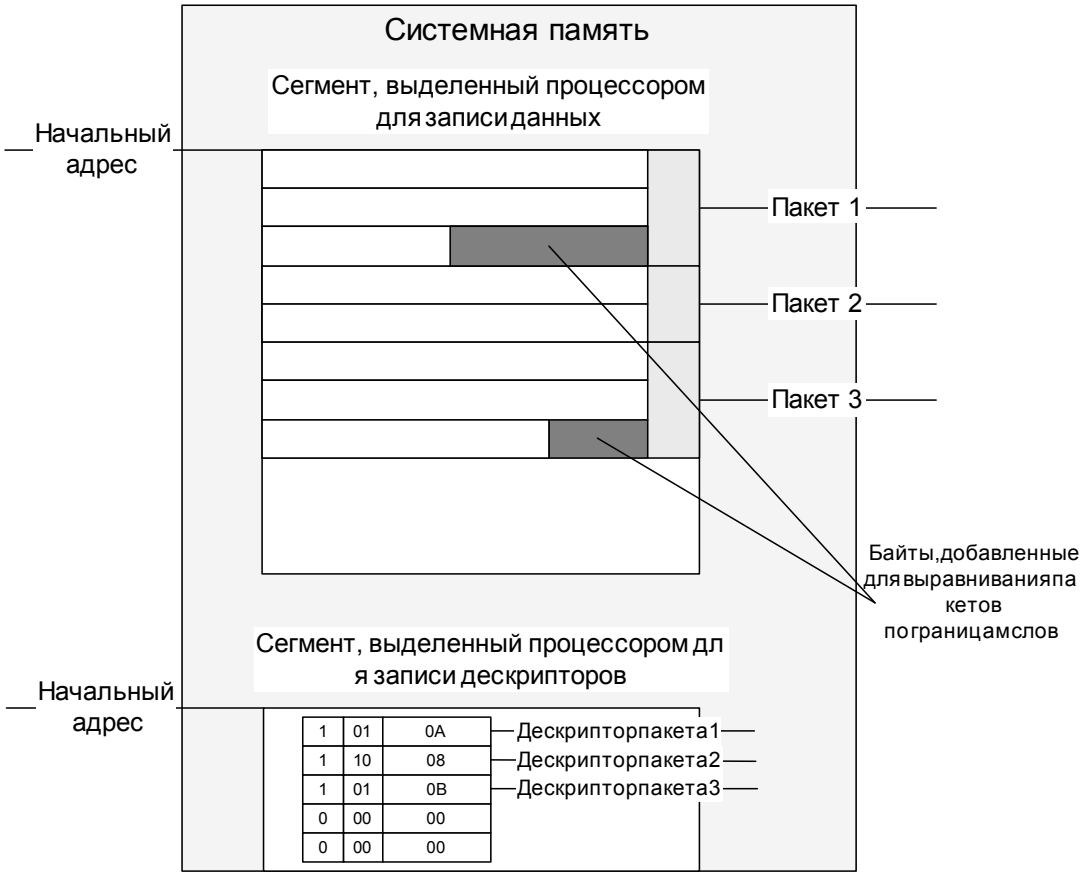


Рисунок 7.4 Расположение принятых данных

Дескрипторы хранятся в сегменте памяти, выделенном процессором для записи дескрипторов. В дескрипторе указаны размеры пакетов в байтах – 0Ah, 08h и 0Bh соответственно. В дескрипторах хранится так же информация о типе конца пакета. В разряд 31 дескриптора записывается 1, что указывает процессору на то, что дескриптор заполнен действительными данными.

7.5.6 Схема обработки данных процессором

В данном примере пакеты могут быть обработаны процессором в соответствии со следующей схемой. Процессор прочитывает первое слово из блока, выделенного для дескрипторов – первый дескриптор. По дескриптору он определяет тип конца пакета, в соответствии с этим решает, как его обрабатывать. По дескриптору он определяет дейст-

вительный размер пакета и извлекает данные, относящиеся к пакету 1. Для того чтобы вычислить начальный адрес второго пакета к начальному адресу блока данных добавляется размер первого пакета и выполняется округление до границы ближайшего слова. После того, как первый пакет полностью обработан, процессор прочитывает дескриптор второго пакета. Обработка остальных пакетов выполняется аналогично. Процесс обработки очереди пакетов заканчивается, когда 31 разряд очередного дескриптора равен 0.

7.5.7 Организация цепочек

Рассмотрим, как организуется цепочка сегментов памяти для хранения данных (Рисунок 7.5). Пусть цепочка включает в себя три блока. В соответствии с начальным адресом цепочки прочитывается первый элемент цепочки, в котором определяется начальный адрес и размер первого блока. После того, как первый блок обработан, выполняется чтение второго элемента цепочки и так далее, до тех пор, пока не будет достигнут конец цепочки.

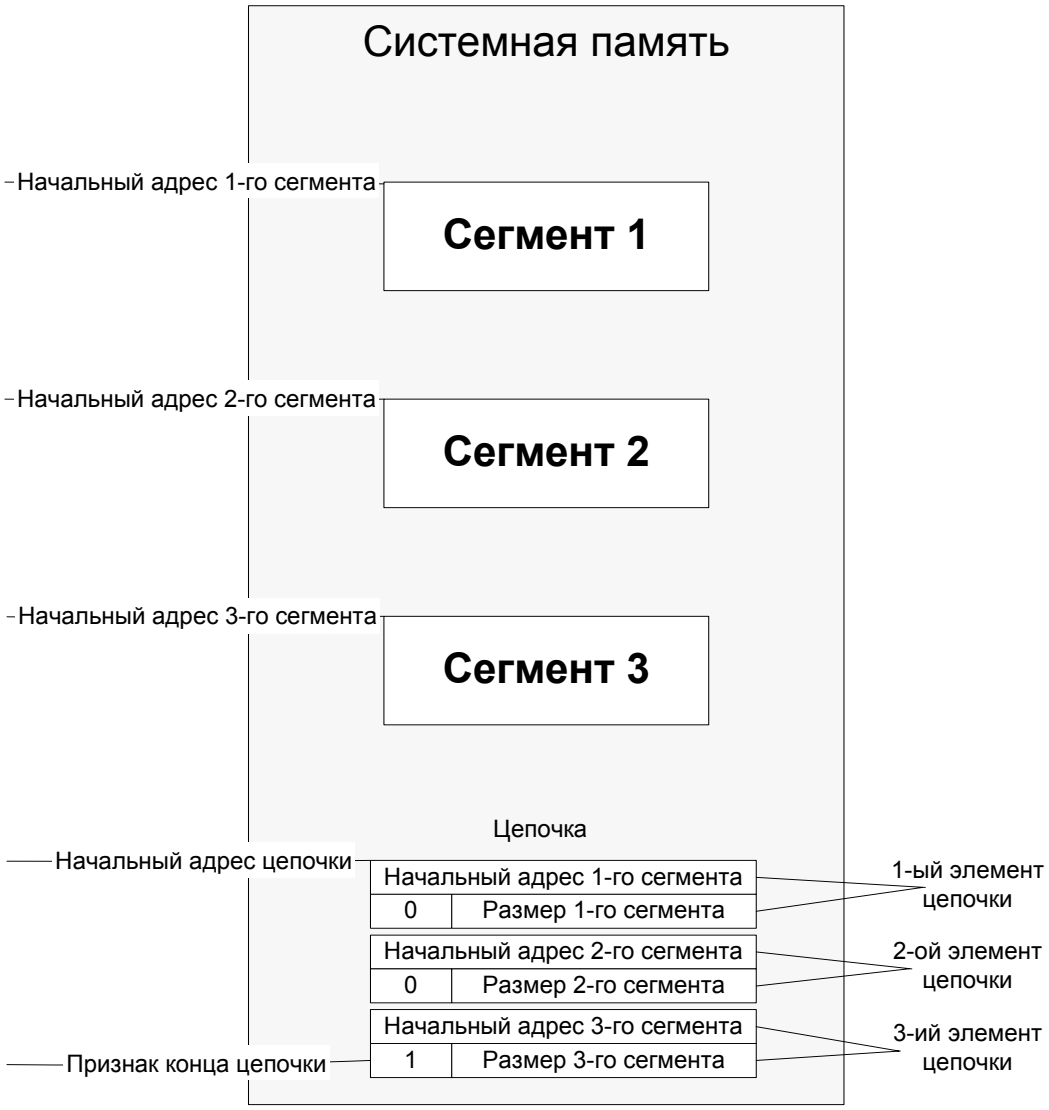


Рисунок 7.5 Пример цепочки блоков памяти

Конец цепочки определяется следующим образом: если в 31 бите второго элемента конца цепочки 1, то это рассматривается как признак конца цепочки (последний элемент в цепочке).

7.5.8 Маркеры времени

Маркеры времени – системная функция стандарта Space Wire. Они предназначены для синхронизации системных часов взаимодействующих систем. При передаче данных маркеры времени имеют наивысший приоритет. Маркер времени записывается в регистр TX_TIME. После записи DS-макроячейка дожидается окончания передачи символа данных или служебного символа и начинает передачу маркера времени, после окончания передачи маркера времени продолжается передача потока данных. В канале приема маркер времени выделяется из потока данных и при безошибочном приеме заносится в регистр RX_TIME с выставлением соответствующего прерывания. В механизм передачи маркера времени внесена упрощенная защита от неверных маркеров. Каждый последующий маркер времени должен быть на «1» больше предыдущего.

7.6 Прерывания SWIC

7.6.1 Прерывание INT_LINK

Соединение установлено или из сети принят пакет. Если прерывание устанавливается вследствие установки соединения, то причина прерывания – первый символ FCT полученный из сети.

Если прерывание устанавливается вследствие принятия пакета из сети, то причина прерывания – факт записи дескриптора пакета в память.

Какая из этих причин вызвала установку сигнала INT_LINK, можно определить в результате чтения регистра STATUS (“1”в разряде CONNECTED указывает на установку соединения, “1” в разряде GOT_EOP указывает на принятие пакета). Причина прерывания может быть определена также и по контексту работы: если в регистр режима были записаны биты, соответствующие установке соединения, то первое появление этого сигнала прерывания указывает на установку соединения, последующие – на прием пакета. Если возникло INT_ERR – разрыв соединения, то также первое появление INT_LINK указывает на установку соединения, а последующие – на прием пакетов.

Для сброса этого сигнала прерывания (например, для того, чтобы иметь возможность отследить получение следующего конца пакета) необходимо записать ‘1’ в разряд CONNECTED или (и) GOT_EOP регистра STATUS. При этом соответствующий разряд регистра состояния сбрасываются в “0”

7.6.2 Прерывание INT_ERR

Причинами возникновения этого прерывания являются ошибка рассоединения, ошибка четности, ошибка в ESC-последовательности, ошибка кредитования.

Процессор может определить конкретную причину прерывания путем чтения регистра состояния STATUS и анализа разрядов D_ERR, P_ERR,ESC_ERR, CREDIT_ERR. Разряд, в котором обнаружится «1» укажет на конкретную ошибку. Для того чтобы иметь возможность отследить возникновение следующей ошибки соединения, необходимо

сбросить сигнал прерывания в “0”, для чего записать “1” в разряд в котором была обнаружена «1». Допускается после выяснения типа ошибки записывать «1» во все разряды – флаги ошибок для снятия прерывания.

Может возникнуть ситуация, когда после возникновения INT_LINK вследствие установки соединения и вскоре после этого будет установлено INT_ERR, так как на противоположной стороне символ FCT получен не будет.

7.6.3 Прерывание INT_TIME

Прерывание возникает при получение маркера времени из сети. Для того чтобы иметь возможность отслеживать получение последующих маркеров времени из сети, необходимо сбросить прерывание, записав ‘1’ в разряд GOT_TIME регистра состояния STATUS.

7.6.4 Прерывание INT_DMA

Данное прерывание возникает при завершение работы DMA или при ошибках при работе DMA.

Причины возникновения:

- блок памяти, выделенный одному из каналов DMA, полностью обработан;
- цепочка блоков памяти, выделенная одному из каналов DMA, полностью обработана;
- при обращении к памяти получено подтверждение типа ERROR.
- Причина прерывания может быть определена в результате чтения регистра состояния DMA SR_DMA. Для сброса прерывания необходимо произвести остановку канала DMA, вызвавшего прерывание, записью в соответствующий разряд регистра CR_DMA, настроить этот канал заново и разрешить его работу.

7.7 Рекомендации по применению

7.7.1 Установка скорости передачи данных

Для включения блока синтезатора частоты в разряды PLL_CTR регистра TX_SPEED необходимо записать "11". Для снижения тока потребления при неиспользуемом контроллере в эти разряды нужно записывать "00" для отключения синтезатора частоты. Перед установкой соединения в регистр TX_SPEED[7:0] необходимо записать код, соответствующий скорости передачи 10 МБит/сек. В соответствии со стандартом Space Wire установление соединения необходимо производить на этой скорости.

Изменение рабочей скорости передачи разрешается только после установления соединения с удаленным контроллером. Рекомендуется применять адаптивный метод определения максимальной скорости передачи. После разрыва соединения в соответствии со стандартом SpaceWire необходимо перед повторным соединением установить скорость передачи 10 МБит/сек.

7.7.2 Установление соединения

Для разрешения процесса установки соединения необходимо записать лог "0" в разряд LinkDisabled и “1” в разряд LinkStart регистра режима работы MODE_CR – для запуска канала, WORK_TYPE = “1”.

Критерием успешного установления соединения является прохождение прерывания INT_LINK и отсутствие прерывания INT_ERR. После обнаружения прерывания INT_LINK, необходимо считать регистр STATUS и проверить биты DC_ERR, P_ERR, ESC_ERR, CREDIT_ERR на равенство «0». Бит CONNECTED должен быть равен «1». Соединение установлено.

Для активации функции автоматического восстановления соединения после обрыва связи дополнительно в разряд AutoStart записывается «1». В этом случае после рассоединения из-за ошибок будет выставлено прерывание INT_ERR и система будет производить повторное установление соединения.

7.7.3 Определение скорости приема данных

Оценка скорости приема выполняется постоянно при разрешенной работе канала. Скорость приема данных отображается в регистре RX_SPEED[7:0]. После установления соединения скорость должна составлять 10±1 Мбит/сек при этом регистр RX_SPEED[7:0] должен быть равен 0x0000000A ±1 единица младшего разряда.

7.7.4 Настройка каналов DMA

Для приёма/передачи данных необходимо настроить один из вариантов одновременной работы каналов DMA:

- канал дескриптора + канал данных
- канал дескриптора + канал данных + канал самоинициализации

Настройка любого канала DMA должна производиться в состоянии, когда его работа запрещена. Запретить работу канала можно через регистр управления каналами DMA CR_DMA.

Канал дескрипторов нужно настроить на область памяти, предназначенную для дескрипторов. Для этого используются регистры базового адреса (ADDR_RxChDes, ADDR_TxChDes) и регистр размера области памяти (SIZE_RxChDes, SIZE_TxChDes).

Канал данных нужно настроить на область памяти, предназначенную для пакетов. Для этого используются регистры базового адреса (ADDR_RxChDat, ADDR_TxChDat) и регистры размера блока памяти (SIZE_RxChDat, SIZE_TxChDat).

Канал самоинициализации нужно настроить на цепочку, описывающую блоки памяти, с которыми будет работать канал данных. Для этого используются регистры базового адреса цепочки (PTR_RxChDat, PTR_TxChDat).

При использовании канала самоинициализации канал данных можно не настраивать (тогда он сразу начнёт работу с блоками памяти из цепочки). Если при использовании канала самоинициализации настроить канал данных, то сначала обработается блок памяти, настроенный в канале данных, а затем будут использоваться блоки памяти из цепочки.

Настраивать регистры адресов и длин можно в любой последовательности.

В каналах DMA память адресуется байтами. При настройке *адреса* сегмента любого канала DMA, записываемое значение должно быть кратно 4, так как блок DMA оперирует 4-байтными словами данных, и адресами, кратными 4.

При настройке *длины* сегментов указывается длина в количестве 32-разрядных слов. Например, если под размер сегмента выделяется область 1024 байт, то в качестве размера сегмента следует указывать число 256.

Запрещается записываеть в качестве размера сегмента значение 0 (в этом случае работа DMA будет не корректна).

Необходимо определить значение регистра *максимального размера транзакции* по шине AMBA. Если значение этого регистра установить в 0, то DMA как ведущее устройство сможет выполнять на шине транзакции неограниченной длины. При использовании канала самоинициализации размер транзакции нельзя устанавливать менее 2 (иначе работа DMA не начнётся).

После того, как настройки выполнены, необходимо разрешить работу каналов, участвующих в обмене данными, через регистр CR_DMA.

Если соединение установлено, работа необходимых каналов DMA разрешена, и приемный и передающий буферы не находятся в состоянии reset, в канале SpaceWire начинается обмен данными.

Для того, чтобы в процессе работы канала определить, какая часть блока уже обработана каналом DMA, необходимо прочитать содержимое регистра адреса блока памяти (ADDR_RxChDes, ADDR_TxChDes, ADDR_RxChDat, ADDR_TxChDat, PTR_RxChDat, PTR_TxChDat). Содержимое этих регистров увеличивается в процессе обработки блока.

Возможна настройка канала самоинициализации при работающем канале данных в том случае, если канал самоинициализации не настроен. В этом случае канал данных после окончания обработки текущего сегмента данных произведет самоинициализацию, если она разрешена, и продолжит работать с новыми сегментами данных из сегмента самоинициализации.

Если необходимо, чтобы данные начали поступать, начиная с области данных, указанных в новой настройке канала самоинициализации, то необходимо дождаться окончания работы каналов данных и самоинициализации, или подать сигнал reset на блок DMA и затем произвести настройку этих каналов.

Перенастройку канала DMA следует проводить только после того, как он закончил работу с выделенным ему блоком памяти и запрашивает новую инициализацию (устанавливается прерывание). Если необходимо изменить настройки канала DMA в процессе его работы (до того, как выделенный ему блок памяти или цепочка полностью обработана), необходимо приостановить работу канала DMA (выполнить соответствующую запись в регистр управления DMA) и выдержать паузу не менее 17 тактов шины AMBA (чтобы дать возможность каналу DMA закончить текущую транзакцию по шине). После этого можно перезаписать значения регистров, относящихся к этому каналу DMA. Если работа канала DMA не будет приостановлена перед перезаписью значений регистров, то значения этих регистров не изменятся.

7.7.5 Определение числа переданных пакетов

После установления связи и начала передачи данных удаленной стороне, имеется возможность отслеживать число переданных пакетов данных через регистр CNT_TX_PACK. Данный регистр позволяет контролировать прогресс передачи заданного числа пакетов. Обнуление счетчика переданных пакетов выполняется программно путем записи в регистр CNT_TX_PACK 32-х разрядного слова, содержащего 0 значение. Значение счетчика увеличивается аппаратно на "1" каждый раз при передаче в DS-макроячейку символа конца пакета и отображается в регистре.

7.7.6 Разрыв соединения

Для принудительного разрыва соединения производится запись в регистр MODE_CR следущих констант: LinkDisabled = 1, AutoStart = 0, LinkStart = 0.

7.7.7 Программный сброс

Сброс всего контроллера выполняется безусловно при активизации сигнала сброса шины AMBA AHB.

Кроме того, контроллер или его компоненты по отдельности (DS-макроячейка, буфер приема, буфер передачи , блок DMA) могут быть переведены в исходное состояние программно. Для обнуления состояния всего контроллера необходимо записать «1» в разряд [7] SWCORE_RST регистра MODE_CR. Для обнуления отдельных блоков контроллера используются разряды 3 – 6 регистра MODE_CR. В соответствующий разряд должна быть записана “1”. После записи «1» в регистр и до записи «0» в этот –же разряд, контроллер будет находиться в состоянии «ОСТАНОВ». В этом состоянии контроллер находится по сигналу системного сброса. Для перевода в рабочий режим, в биты 3 – 7 регистра MODE_CR необходимо записать нули.

В результате сброса буфера приема его внутренняя память очищается (регистры управления в блоке приема обнуляются). В результате сброса буфера передачи его внутренняя память также очищается (регистры управления в блоке передачи обнуляются). В результате сброса блока DMA во все его внутренние регистры (относительные адреса 8-20) записываются нулевые значения, что приводит, в том числе, к запрещению работы всех его каналов и установке прерывания 4.

Если выполняется сброс Контроллера в целом (в разряд 7 регистра режима работы записана 1), то для DS-макроячейки, блока приема, блока передачи и блока DMA выполняются вышеописанные действия. Кроме того, обнуляются все внутренние регистры блока управления (относительные адреса 0 – 7). В этом случае через интерфейс ведомого устройства на запись будет доступен только этот регистр. В нем будет возможность изменять разряды 3-6, тем самым выйти из режима сброса и продолжить работу.

При завершении работы со SWIC его необходимо перевести в режеим «ОСТАНОВ» записав в регистр MODE_CR[3:7] единичные значения разрядов.

8. КОНТРОЛЛЕР ПРЯМОГО ДОСТУПА В ПАМЯТЬ (DMA)

8.1 Общие положения

8.1.1 Типы каналов

Контроллер DMA имеет 12 каналов следующих типов:

- каналы обмена данными между линковыми портами и внутренней (CRAM, PMEM, XMEM, YRAM) или внешней памятями;
- каналы обмена данными между внутренней памятью (CRAM, PMEM, XMEM, YRAM) и внешней памятью.

Перечень каналов DMA МСТ-01 приведен в Таблица 8.1.

Таблица 8.1. Каналы DMA

Условное Обозначение Канала	Назначение канала	Приоритет каналов DMA и CPU
CPU	-	4
LportCh3 – LportCh0	Обмен данными между буферами данных линковых портов и памятью (внешней или внутренней)	8-5
MemCh3 – MemCh0	Обмен данными между внешней памятью и внутренней памятью.	12-9 (изменяется циклически)

Если при работе DMA изменяется программный код в памяти, то когерентность кэш программ CPU (ICACHE) аппаратно не обеспечивается. В этом случае для обеспечения когерентности используется бит FLUSH в регистре CSR.

8.1.2 Приоритет каналов DMA и CPU

CPU по шине CDB без конфликтов с DMA обменивается с памятью CRAM, с системными регистрами CSR, MASKR, QSTR, и с регистрами таймеров IT, WDT, RTT. CPU без конфликтов с DMA обменивается с регистрами MPORT и внешней памятью, если нет DMA передач.

При передаче данных каналы DMA конфликтуют между собой всегда. Каналы DMA конфликтуют с CPU, если CPU и DMA одновременно запрашивают шину DDB.

Приоритет каналов DMA указан в правой колонке Таблица 8.1 (0 – наивысший приоритет). Если несколько каналов DMA одновременно запрашивают шину DDB, то ее занимает канал, приоритет которого самый высокий.

Взаимный приоритет каналов MemCh изменяется циклически следующим образом. Исходное распределение приоритетов между каналами MemCh (в порядке их убывания): MemCh0, MemCh1, MemCh2, MemCh3. Далее, после каждой DMA передачи распределение приоритетов изменяется циклическим сдвигом влево, таким образом, что приоритет канала, который выполнил DMA передачу, становится самым низким. Например, если после исходного состояния передал канал MemCh0, то приоритеты распределятся следующим образом: MemCh1, MemCh2, MemCh3, MemCh0. Далее, если передал

канал MemCh3, то приоритеты распределятся следующим образом: MemCh0, MemCh1, MemCh2, MemCh3 и т.д.

8.1.3 Темп передачи

DMA передача одного 32-разрядного слова данных между внутренней памятью и LPORT выполняется за время TCLK (период частоты CLK).

Время DMA передачи одного 32-разрядного слова данных между внешней памятью и LPORT или внутренней памятью, равно:

- для асинхронной внешней памяти – 2*TCLK + TCLK*N, где N – число тактов ожидания (код в поле WS регистров CSCON, увеличенный на 1).
- для синхронной внешней памяти - TCLK.

Каналы линковых портов за один цикл занятия шины DDB передают одно слово данных. После передачи этого слова шина DDB данным каналом освобождается.

Каналы MemCh за один цикл занятия шины DDB передают пачку данных. Размер пачки задается полем WN в регистре CSR соответствующего канала DMA и определяется системными требованиями по передаче данных. Если после передачи пачки данных нет запросов от других каналов DMA или CPU, то данный канал без перерыва начинает передавать следующую пачку данных и т.д.

CPU за один цикл занятия шины DDB выполняет одну из следующих операций (после этого шина освобождается):

- чтение одного слова данных по команде Load;
- запись одного слова данных по команде Store;
- выборка команды из внешней памяти;
- процедура Refill (загрузка из внешней памяти в ICACHE 4 команды), если адрес команды CACHED, а ее нет в ICACHE (ситуация MISS).

8.1.4 Регистры DMA

Для управления работой каждого канала DMA имеются следующие регистры:

- регистр управления и состояния (CSR);
- набор регистров индекса (адрес памяти) и смещения (IOR, IR, OR, Y);
- регистр начального адреса блока параметров DMA передачи (CP).

Следует отметить, что индексные регистры IR и IOR содержат физические адреса памяти.

Для эффективной передачи двумерных массивов (матриц W[m;n]) все каналы DMA используют регистр Y, в котором хранятся смещение и число строк в направления Y.

Разные типы каналов содержат разный набор регистров.

Исходное состояние регистров CSR: разряды 15:0 – нули, а состояние разрядов 31:16 не определено. Исходное состояние остальных регистров не определено.

Индексный регистр содержит адрес 32-разрядного слова в памяти (младшие два разряда адреса должны быть равны нулю).

Регистр смещения задает приращение адреса. Содержимое регистра смещения, аппаратно умноженное на 4, прибавляется к индексу после передачи каждого слова данных. Если по каналам MemCh выполняется обмен данными с SDRAM, то смещение прибавляется после передачи каждой пачки 32-разрядных слов, которая передается в режиме “Burst”. То есть, при обмене данными с SDRAM по каналам MemCh, величина смещения в регистре OR должна быть не меньше, чем размер пачки, указанный в поле WN регистра CSR (WN=0, OR>=1; WN=1, OR>=2 и т.д.).

8.1.5 Прерывания DMA

Канал DMA формирует прерывание (при условии, если установлен соответствующий бит в регистре MASKR и бит IM[7] в регистре STATUS RISC-ядра):

- при единичном состоянии бита DONE;
- при единичном состоянии битов END и IM.

Обнуление битов DONE и END (и снятие соответствующего прерывания) выполняется посредством чтения содержимого регистра CSR. Обнуление бита DONE может быть выполнено также записью нуля в него.

8.2 Процедура самоинициализации

Все каналы DMA могут выполнять процедуру самоинициализации (выполнение цепочки передач DMA).

Для выполнения самоинициализации в каналах имеется 16-разрядный регистр CP, в котором хранится начальный адрес блока параметров очередного DMA обмена. Эти параметры при самоинициализации аппаратно загружаются в соответствующие регистры канала DMA. Процедура этой загрузки ничем не отличается от обычного DMA обмена. Блок параметров может размещаться только во внутренней памяти MEM.

Блоки параметров, размещаемых в памяти, имеют следующую структуру (в порядке возрастания адресов):

- каналы линковых портов – IR, OR, Y, CP, CSR;
- каналы MemCh – IOR, IR, OR, Y, CP, CSR.

Параметры, соответствующие 16-разрядным регистрам, размещаются в младших разрядах памяти. В слове памяти, соответствующем регистру CSR должно быть: RUN=1, DONE=0. Если необходимо продолжить цепочку команд, то необходимо указать CHEN=1.

Для запуска работы канала DMA в режиме с самоинициализацией необходимо в регистр CP записать адрес первого блока параметров DMA передачи. При этом 31 разряд записываемых данных должен содержать 1 (признак пуска самоинициализации). В результате этого, соответствующий канал загрузит в свои регистры параметры DMA передачи и начнет обмен данными.

После окончания передачи данного блока данных устанавливается в единичное состояние бит END в регистре CSR и выдается прерывание, если бит IM = 1. После этого канал проверяет состояние бита CHEN. Если он равен 1, то будет загружен следующий блок параметров DMA передачи и т.д. В противном случае цепочка DMA обменов закончится и в регистре CSR бит DONE установится в единичное состояние.

При необходимости каналы DMA могут инициализироваться программно. Для этого RISC должен загрузить все необходимые регистры индекса и смещения, а затем регистр CSR. При загрузке регистра CSR бит RUN необходимо установить в единичное состояние. Следует отметить, что бит RUN может быть использован для приостановки канала DMA. Для этого в любой момент времени в него необходимо записать 0.

Следует иметь в виду, что если биты END или DONE имеют единичное состояние, то после считывания содержимого регистра CSR эти биты автоматически обнуляются.

8.3 Каналы DMA линковых портов

Для обслуживания линковых портов имеется 4 канала DMA: LportCh0, LportCh1, LportCh2, LportCh3.

Формат регистров управления и состояния CSR_Lp0, CSR_Lp1, CSR_Lp2, CSR_Lp3 каналов DMA линковых портов приведен в Таблице 8.2.

Таблица 8.2. Формат регистров управления и состояния DMA линковых портов

Номер разряда	Условное Обозначение	Назначение
0	RUN	Состояние работы канала DMA: 0 – состояние останова; 1 – состояние обмена данными.
1-8	-	Резерв
9	2D	Режим модификации адреса памяти: 0 – одномерный режим; 1 – двухмерный режим.
11,10	-	Резерв
12	CHEN	Признак разрешения самоинициализации (выполнения цепочки DMA передач)
13	IM	Маска прерывания при окончании передачи блока данных: 0 – прерывание запрещено; 1 – прерывание разрешено.
14	END	Признак окончания передачи блока данных
15	DONE	Признак завершения передачи цепочки блоков данных. Аппаратно устанавливается в 1 после завершения передачи данных (при CHEN=0), при этом бит RUN сбрасывается. Доступен по записи и чтению. Состояние данного бита дублируется в соответствующий бит регистра QSTR
31:16	WCX	Счетчик слов при одномерной адресации. Счетчик числа слов в строке при двухмерной адресации.

Для задания адреса памяти (внутренней или внешней) каналы DMA последовательных портов содержат два регистра:

32-разрядный индексный регистр памяти IR;

16-разрядный регистр смещения памяти OR.

16-разрядный регистр OR содержит код смещения (приращения) памяти в 32-разрядных словах для перехода к следующему элементу массива. Он используется всегда. При адресации в двухмерном режиме он указывает приращение в направлении X. Приращение рассматривается как число со знаком в диапазоне от –32768 до +32767.

При работе каналов последовательных портов память (внутренняя или внешняя) может адресоваться в двухмерном режиме. Для этого имеется 32-разрядный регистр Y, формат которого приведен в Таблице 8.3.

Таблица 8.3. Формат регистра Y

Номер разряда	Условное Обозначение	Назначение
15:0	OY	Смещение (приращение) адреса памяти в 32-разрядных словах по направлению Y. Используется только при двухмерной адресации.
31:16	WCY	Число строк по Y направлению. Используется только при двухмерной адресации.

При двухмерном режиме адресации поле WCX регистра CSR содержит число слов в строке (X направление), а поле WCY регистра Y содержит число строк (Y направление). Пересылка каждого слова данных осуществляется по индексному регистру IR с его последующей инкрементацией на величину, соответствующую содержимому регистра смещения или поля OY регистра Y. Двухмерная адресация выполняется следующим образом:

Содержимое счетчика WCX сохраняется в буферном регистре;

1 цикл. Индексный регистр внешней памяти модифицируется с использованием смещения OR_MEM. Счетчик WCX декрементируется. Если он равен 0, то переход ко второму циклу.

2 цикл. Состояние счетчика WCX восстанавливается из буферного регистра. Индексный регистр внешней памяти модифицируется с использованием смещения OY. Счетчик WCY декрементируется. Если он не равен 0, то переход к первому циклу. Если он равен 0, то работа канала завершается.

Функционально двухмерная адресация эквивалентна следующему двойному циклу:

```
for  ( i=0;   i < WCY;  i++ )  {  /* Внешний цикл, выполняется WCY раз  */
  for  ( k=0;   k < WCX - 1;  k++ )  {/* Внутренний цикл, выполняется WCX-1  раз  */
          переслать  слово по указателю *(IR) ++ OR;  /* Постинкремент  указателя*/
      };                                                /* на  OR слов
        */
      переслать  слово по указателю *(IR) ++ OY;       /*  Постинкремент  указателя */
};                                                      /*  на OY слов              */
```

8.4 Каналы обмена данными между внутренней и внешней памятью

Четыре канала MemCh0:MemCh3 обеспечивают обмен данными между внутренней памятью МСТ-01 CRAM и внешней памятью.

Формат регистров состояния и управления этих каналов приведен в Таблице 8.4.

Таблица 8.4. Формат регистра управления и состояния каналов MemCh

Номер разряда	Условное обозначение	Назначение
0	RUN	Состояние работы канала DMA: 0 – состояние останова; 1 – состояние обмена данными.
1	DIR	Направление обмена данными: 0 – внутренняя память => внешняя память; 1 – внутренняя память <= внешняя память.
5:2	WN	Число слов данных (пачка), которое передается за одно предоставление прямого доступа: 0 – 1 слово, F – 16 слов. Посредством этого параметра можно плавно изменять приоритет каналов DMA относительно RISC и относительно друг друга.
6:7	-	Резерв
8	MODE	Режим модификации адреса внутренней памяти: 0 – линейный режим; 1 – режим с реверсивным переносом.
10	MASK	Маска внешнего запроса прямого доступа nDMAR: 0 – запрос запрещен; 1 – запрос разрешен. Если разряд равен нулю, то канал работает только под управлением бита RUN. Если разряд равен 1, то для инициализации канала необходимо также наличие запроса nDMAR (низкий уровень).
11	FLYBY	Признак выполнения обмена между внешней памятью и внешним устройством.

Продолжение Таблицы 8.4

Номер разряда	Условное обозначение	Назначение
12	CHEN	Признак разрешения самоинициализации (выполнения цепочки DMA передач)
13	IM	Маска прерывания при окончании передачи блока данных: 0 – прерывание запрещено; 1 – прерывание разрешено.
14	END	Признак окончания передачи блока данных
15	DONE	Признак завершения передачи цепочки блоков данных. Аппаратно устанавливается в 1 после завершения передачи цепочки блоков данных (при CHEN=0), при этом бит RUN сбрасывается. Доступен по записи и чтению. Состояние данного бита дублируется в соответствующий бит регистра QSTR
31:16	WCX	Счетчик слов при одномерной адресации. Счетчик числа слов в строке при двухмерной адресации.

Следует иметь в виду, что при обмене с внешней памятью типа SDRAM в поле WN допускается указывать только числа 0, 1, 3, 7, 15. При этом начальный адрес массива, предназначенного для передачи при помощи DMA, должен быть кратен WN+1. В противном случае обмен данными будет произведен неправильно.

Состоянием разряда 0 регистра CSR можно управлять, используя адрес псевдорегистра Run. При этом остальные разряды этого регистра не изменяются. Эта процедура может быть использована для временной приостановки канала DMA.

Для задания адресов обмена данными каналы MemCh содержат три регистра:

- 32-разрядный регистр индекса и смещения адреса внутренней памяти IOR;
- 32-разрядный индексный регистр внешней памяти IR;
- 16-разрядный регистр смещения внешней памяти OR.

Формат регистра индекса и смещения IOR_MEM приведен в Таблице 8.5.

Таблица 8.5. Формат регистра индекса и смещения каналов MemCh

Номер разряда	Условное Обозначение	Назначение
23:0	ADDR	Адрес внутренней памяти
31:24	OFFSET	Смещение (приращение) адреса внутренней памяти в 32-разрядных словах после передачи каждого слова данных

Смещение, задаваемое полем OFFSET, имеет диапазон от –128 до +127.

При инверсном режиме модификации адреса внутренней памяти смещение, задаваемое полем OFFSET, имеет диапазон от 0 до 255.

Поле ADDR в регистре IOR_MEM указывает адрес внутренней памяти относительно базового адреса 1800_0000.

16-разрядный регистр OR содержит код смещения внешней памяти в 32-разрядных словах. Он используется всегда. При адресации в двухмерном режиме он указывает смещение (приращение) в направлении X для перехода к следующему элементу строки. Смещение рассматривается как число со знаком в диапазоне от –32768 до +32767.

При работе каналов MemCh внешняя память может адресоваться в двухмерном режиме аналогично каналам последовательных портов.

Работа по внешним запросам.

Каждый канал MemCh[3-0] имеет внешний сигнал запроса передачи (nDMAR[3-0] соответственно), позволяющий организовывать эффективный обмен данными с внешними устройствами. Для работы по внешним запросам необходимо сначала настроить канал DMA (в том числе установить бит MASK регистра CSR_MemCh в «1»), а затем активизировать внешнее устройство на формирование сигналов nDMAR.

По каждому переходу сигнала nDMAR из «1» в «0» DMA выполняет процедуру передачи одной пачки слов размером в соответствии с полем WN регистра CSR_MemCh. Внешнее устройство может снять сигнал nDMAR в начале этой пачки или выдавать сигнал nDMAR в виде отрицательного импульса длительностью не менее 1,5 периодов системной тактовой частоты CLK (частота, на которой работает CPU).

Следует иметь в виду, что факт перехода сигнала nDMAR из «1» в «0» запоминается в DMA на триггере. Это триггер сбрасывается в момент представления данному каналу права на передачу в соответствии с его текущим приоритетом.

Необходимо также учитывать то, что факт перехода сигнала nDMAR из «1» в «0» запоминается в DMA при MASK=1 вне зависимости от состояния бита RUN. Если в процессе работы в DMA будет запомнен «лишний» факт перехода сигнала nDMAR из «1» в «0», то его можно сбросить, выполнив фиктивный DMA обмен.

Работа в режиме FLYBY.

Режим FLYBY используется для передачи данных между внешним устройством ввода-вывода и внешней памятью (как асинхронной, так и синхронной). Например, контроллер DMA может быть запрограммирован для передачи данных из аналого-цифрового преобразователя в SDRAM. Для выполнения передачи данных в этом режиме в соответствующем регистре CSR_MemCh необходимо установить бит FLYBY.

При передаче данных в режиме FLYBY МСТ-01 отключается от шины данных, и активизирует внешнюю память и внешнее устройство ввода-вывода одновременно. Память управляется как обычно, а устройство ввода-вывода – при помощи сигналов nFLYBY (признак данного режима) , nOE (активизация выходных формирователей устройства ввода-вывода) и nCSIO[3:0] (выбор устройства ввода-вывода).

Каждому каналу MemCh может соответствовать свое устройство ввода-вывода. Выбор устройства ввода-вывода осуществляется посредством сигналов nCSIO[3:0]. Каналу MemCh0 соответствует низкий уровень на выводе nCSIO[0], каналу MemCh1 соответствует низкий уровень на выводе nCSIO[1], и так далее.

В режиме FLYBY можно использовать сигналы nDMAR[3:0].

Временные диаграммы работы МСТ-01 в режиме FLYBY приведены в разделе 9.

9. ПОРТ ВНЕШНЕЙ ПАМЯТИ

9.1 Введение

Порт внешней памяти (MPORT) позволяет организовать интерфейс с широким набором устройств памяти и периферии, асинхронной и синхронной памятью. Внешний интерфейс порта обеспечивает подключение без сложной дополнительной логики синхронной памяти типа SDRAM, а также асинхронной памяти, например EPROM и FLASH.

Порт памяти имеет следующие основные характеристики:

- Шина данных внешней памяти – 32 разряда;
- Шина адреса внешней памяти – 32 разряда;
- программное конфигурирование областей внешней памяти;
- интерфейс с синхронной памятью типа SDRAM;
- интерфейс с асинхронной памятью (SRAM, EPROM, FLASH, FIFO и т.д.);
- режим передачи данных Flyby;
- управление числом тактов ожидания при обмене с асинхронной памятью при помощи внешнего входного сигнала nACK и поля WS регистров CSCON.

9.2 Регистры порта внешней памяти

Перечень регистров порта внешней памяти приведен в Таблица 9.1.

Таблица 9.1. Регистры порта внешней памяти

Условное обозначение регистра	Название регистра
CSCON0	Регистр конфигурации 0.
CSCON1	Регистр конфигурации 1.
CSCON2	Регистр конфигурации 2.
CSCON3	Регистр конфигурации 3.
CSCON4	Регистр конфигурации 4.
SDRCON	Регистр конфигурации памяти типа SDRAM
CKE_CTR	Регистр управления состоянием вывода CKE

9.2.1 Регистр конфигурации CSCON0

Регистр CSCON0 предназначен для конфигурирования блока внешней памяти, подключаемого к выводу nCS[0].

Формат регистра приведен в Таблица 9.2.

Таблица 9.2. Назначение разрядов регистра CSCON0

Номер разряда	Условное Обозначение	Описание
7-0	CSMASK	Разряды маски 31:24 при определении базового адреса блока. Младшие разряды маски равны нулю.
15-8	CSBA	Разряды 31:24 базового адреса блока. Младшие разряды базового адреса равны нулю.
19-16	WS	Число тактов ожидания при обращении к блоку памяти, если она является асинхронной
20	E	Разрешение формирования сигнала nCS[0]: 0 – запрещено; 1 – разрешено.
21	T	Тип памяти данного блока: 0 – асинхронная; 1 – синхронная.
22	AE	Разрешение ожидания сигнала nACK: 0 – запрещено; 1 – разрешено.
31-23	-	Резерв

Регистр CSCON0 доступен по записи и чтению. Исходное состояние регистра – 000F_0000.

Сигнал nCS[0] формируется, если PHA & CSMASK = CSBA, где PHA – 32-разрядный физический адрес. Минимальный размер блока – 16 Мбайт (при CSMASK = FF). Для увеличения размера блока в младшие разряды поля CSMASK необходимо записать соответствующее число нулей. Например, для блока размером в 128 Мбайт, разряды 2-0 CSMASK должны быть равны нулю.

Регистры CSCON должны быть сконфигурированы таким образом, чтобы определяемые ими области памяти занимали уникальные адресные пространства. Если эти области перекрываются, то результат обмена данными будет непредсказуем.

В поле WS этого регистра задается количество тактов ожидания в тактах частоты CLK, которое необходимо добавить в цикл шины при обращении к несинхронной внешней памяти. Во время аппаратного сброса процессора во все эти поля записывается значение F (15 тактов).

Управление длительностью циклов обмена с асинхронной памятью осуществляется сигналом nACK и полем тактов ожидания WS. Сигнал nACK позволяет вставлять такты ожидания непосредственно в начатый цикл обмена данными. Количество вставленных тактов ожидания равно максимальному количеству дополнительных тактов, заданных полем WS и сигналом nACK.

9.2.2 Регистр конфигурации CSCON1

Регистр CSCON1 предназначен для конфигурирования блока памяти, подключаемого к выводу nCS[1]. Формат регистра приведен в Таблице 9.3.

Таблица 9.3. Назначение разрядов регистра CSCON1

Номер разряда	Условное Обозначение	Описание
7-0	CSMASK	Разряды маски 31:24 при определении базового адреса блока. Младшие разряды маски равны нулю.
15-8	CSBA	Разряды 31:24 базового адреса блока. Младшие разряды базового адреса равны нулю.
19-16	WS	Число тактов ожидания при обращении к блоку памяти, если она является асинхронной
20	E	Разрешение формирования сигнала nCS[1]: 0 – запрещено; 1 – разрешено.
21	T	Тип памяти данного блока: 0 – асинхронная; 1 – синхронная.
22	AE	Разрешение ожидания сигнала nACK: 0 – запрещено; 1 – разрешено.
31-23	-	Резерв

Регистр CSCON1 доступен по записи и чтению. Исходное состояние регистра – 000F_0000.

9.2.3 Регистр конфигурации CSCON2

Регистр CSCON2 предназначен для конфигурирования блока памяти, подключаемого к выводу nCS[2].

Формат регистра приведен в Таблица 9.4.

Таблица 9.4. Назначение разрядов регистра CSCON2

Номер разряда	Условное Обозначение	Описание
7-0	CSMASK	Разряды маски 31:24 при определении базового адреса блока. Младшие разряды маски равны нулю.
15-8	CSBA	Разряды 31:24 базового адреса блока. Младшие разряды базового адреса равны нулю.
19-16	WS	Число тактов ожидания при обращении к блоку памяти.
20	E	Разрешение формирования сигнала nCS[2]: 0 – запрещено; 1 – разрешено.
21	-	Резерв
22	AE	Разрешение ожидания сигнала nACK: 0 – запрещено; 1 – разрешено.
31-23	-	Резерв

Регистр CSCON2 доступен по записи и чтению. Исходное состояние регистра – 000F_0000.

Память, подключаемая к выводу nCS[2], может быть только асинхронной.

9.2.4 Регистр конфигурации CSCON3

Регистр CSCON3 предназначен для конфигурирования блока памяти, подключаемого к выводу nCS[3].

Формат регистра приведен в Таблица 9.5.

Таблица 9.5. Назначение разрядов регистра CSCON3

Номер разряда	Условное обозначение	Описание
15-0	-	Резерв
19-16	WS	Число тактов ожидания при обращении к блоку памяти.
22-20	-	Резерв
23	BYTE	Разрядность памяти: 0 – 32 разряда; 1 – 8 разрядов. Исходное состояние данного разряда соответствует состоянию сигнала на входе BYTE микросхемы во время аппаратного сброса.
24	OVER	Признак того, что при обмене данными с асинхронной памятью блоков 0, 1, 2, 4 от нее не поступил сигнал nACK в течение 256 периодов частоты CLK.
31-25	-	Резерв

Регистр CSCON3 доступен по записи и чтению. Исходное состояние регистра – 000F_0000, или 008F_0000, в зависимости от состояния сигнала на выводе BYTE микросхемы.

Область памяти, определяемая регистром CSCON3, размещается в диапазоне физических адресов от 1C00_0000 до 1FFF_FFFF (64 Мбайт). Память данного блока может быть только асинхронной. Доступ к данному блоку памяти всегда разрешен. При обмене данными с этим блоком сигнал nACK безразличен.

Как правило, к выводу nCS[3] подключается блок памяти программ, реализованный на FLASH, PROM, EEPROM и т.д. Этот блок, в зависимости от состояния сигнала на выводе микросхемы BYTE может быть 8 – или 32 – разрядным.

8-разрядная память подключается к выводам D[7:0] микросхемы МСТ-01. Шину адреса A[31:0] к этой памяти необходимо подключать, начиная с 0 разряда (к 32-разрядной памяти адрес подключается, начиная со 2 разряда). 32-разрядное слово из 8-разрядной памяти считывается байтами, причем сначала считывается младший байт. Запись данных в 8-разрядную память выполняется побайтно в соответствии с рекомендациями п. 9.4.2.

Признак OVER формируется, если в соответствующем регистре CSCON бит AE=1, а от памяти не поступил сигнал nACK в течение 256 тактов CLK. В этом случае операция обмена данными заканчивается обычным образом, за исключением того, что считываемые данные не определены, а записываемые данные теряются. Состояние бита OVER не влияет на выполнение последующих операций обмена данными.

9.2.5 Регистр конфигурации CSCON4

Регистр CSCON4 предназначен для конфигурирования внешней памяти, не вошедшей в области, определяемые регистрами CSCON3-CSCON0.

Формат регистра приведен в Таблице 9.6.

Таблица 9.6. Назначение разрядов регистра CSCON4

Номер разряда	Условное Обозначение	Описание
15-0	-	Резерв
19-16	WS	Число тактов ожидания при обращении к памяти.
21:20	-	Резерв
22	**AE**	Разрешение ожидания сигнала nACK: 0 – запрещено; 1 – разрешено.
31-23	-	Резерв

Регистр CSCON4 доступен по записи и чтению. Исходное состояние регистра – 000F_0000.

Данная область памяти может быть только асинхронной. Доступ к ней всегда разрешен.

9.2.6 Регистр управления работой с памятью SDRAM

Формат регистра приведен в Таблица 9.7. Исходное состояние – нули.

Таблица 9.7. Формат регистра SDRCON

Номер разряда	Условное Обозначение	Описание
3:0	PS	Размер страницы микросхем SDRAM, подключенных к порту внешней памяти: 0 – 512; 1 – 1024; 2 – 2048; 3 – 4096. Число банков SDRAM – 4.
15:4	RFR	Период регенерации SDRAM в тактах частоты CLK
18:16	BL	Длина burst (двоичный код): 000 – 1; 001 – 2; 010 – 4; 011 – 8; 100:110 – резерв; 111 – Full Page.
19	WBM	Режим записи: 0 – Программируемая длина burst; 1 – Одиночная запись.
20	CL	Задержка чтения (CAS latency): 0 – 2; 1 – 3.
30:21	-	Резерв
31	INIT	При выполнении процедуры записи 1 в данный разряд выполняется процедура инициализации SDRAM. Время инициализации – не более 2 мкс. В SDRAM устанавливаются следующие режимы работы: Burst Length – поле BL; Burst Type – последовательный; CAS latency – бит CL; Режим записи – бит WBM.

Регистр SDRCON доступен по записи и чтению. Исходное состояние регистра – 0. 31 разряд регистра SDRCON доступен только по записи, при чтении всегда 0.

Для работы со SDRAM ее необходимо инициализировать со следующими параметрами:

- PS (размер страницы) - в соответствии с параметрами SDRAM;
- RFR (период регенерации) – в соответствии с параметрами SDRAM. Например, при тактовой частоте SMK 100 МГц для обеспечения 8 192 цикловой регенерации за 64 мс необходимо в поле RFR записать код 30D, что соответствует 7, 81 мкс на строку;

- BL = 111 (Full page). Остальные значения используются только при тестировании микросхемы;
- WBM = 0 (программируемая длина burst);
- CL (задержка чтения) - в соответствии с параметрами SDRAM;

Выполнение инициализации SDRAM осуществляется посредством записи в регистр SDRCON соответствующего кода с единицей в 31 разряде. Следует отметить, что перед выполнением процедуры инициализации SDRAM необходимо сконфигурировать регистры CSCON0, CSCON1.

Для прекращения burst Full Page и тем самым задания реального числа передаваемых слов данных, используется команда «BURST TERMINATE», которая формируется портом внешней памяти аппаратно.

9.2.7 Регистр CKE_CTR

Регистр CKE_CTR предназначен для управления состояния вывода CKE микросхемы.

Формат регистра приведен в Таблица 9.8.

Таблица 9.8. Назначение разрядов регистра CKE_CTR

Номер разряда	Условное Обозначение	Описание
0	CKE	Состояние вывода CKE микросхемы: 0 – низкий уровень; 1 – высокий уровень.
1-7	-	Резерв.
8	INIT_DONE	Признак окончания выполнения процедуры инициализации SDRAM: 0 – инициализация завершена; 1 – инициализация не проводилась.
31-9	-	Резерв.

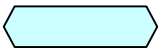
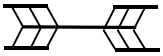
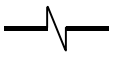
Регистр CKE_CTR доступен по записи и чтению. Исходное состояние регистра – 0000_0101.

9.3 Временные диаграммы обмена данными

9.3.1 Общие положения

При описании временных диаграмм используются условные обозначения в соответствии с **Таблица 9.9**.

Таблица 9.9. Условные обозначения

Условное обозначение	Описание
	Стабильное значение
	Возможное значение
	область изменения из «0» в «1»
	область изменения из «1» в «0»
	Достоверное значение
	Для входов: Не воспринимается, допустимо любое переключение Для выходов: состояние не определено
	Переключение выхода из (в) высокоимпедансного (е) состояния (е) (центральная линия)
	Повторение сигнала в течение неопределенного времени
Ti	i = 1, 2, … фаза обмена на временной диаграмме
n	Число дополнительных тактов ожидания, задаваемых полем WS регистров CSCON
w	Число тактов ожидания поступления сигнала nACK
nCSx	Один из четырёх сигналов nCS[3:0]
nCSIOx	Один из четырёх сигналов nCSIO[3:0]

9.3.2 Обмен данными с асинхронной памятью

Временные диаграммы записи данных в асинхронную память приведены на Рисунок 9.1 - Рисунок 9.3.

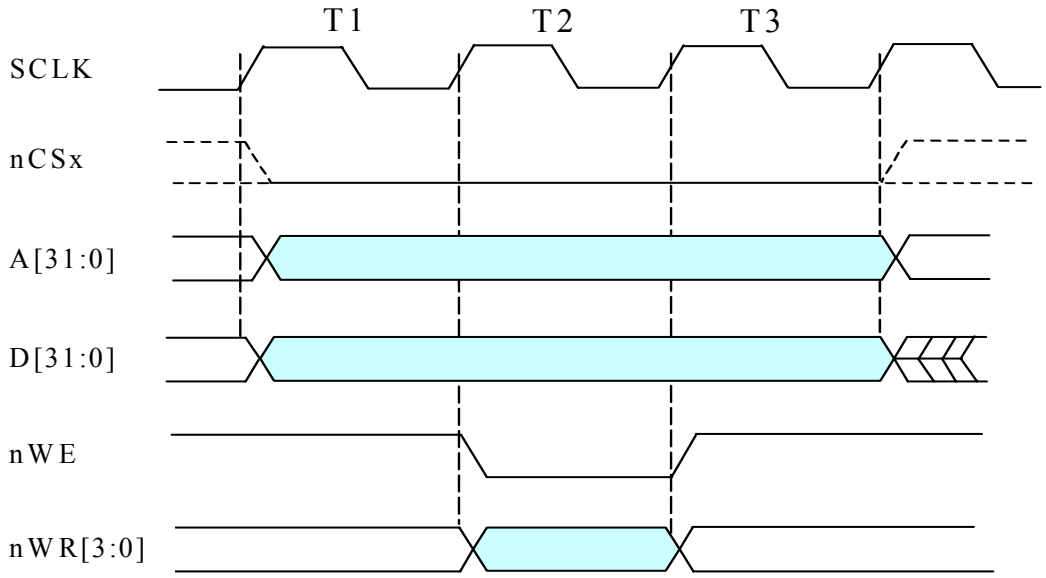


Рисунок 9.1 .Запись в асинхронную память без дополнительных тактов ожидания.

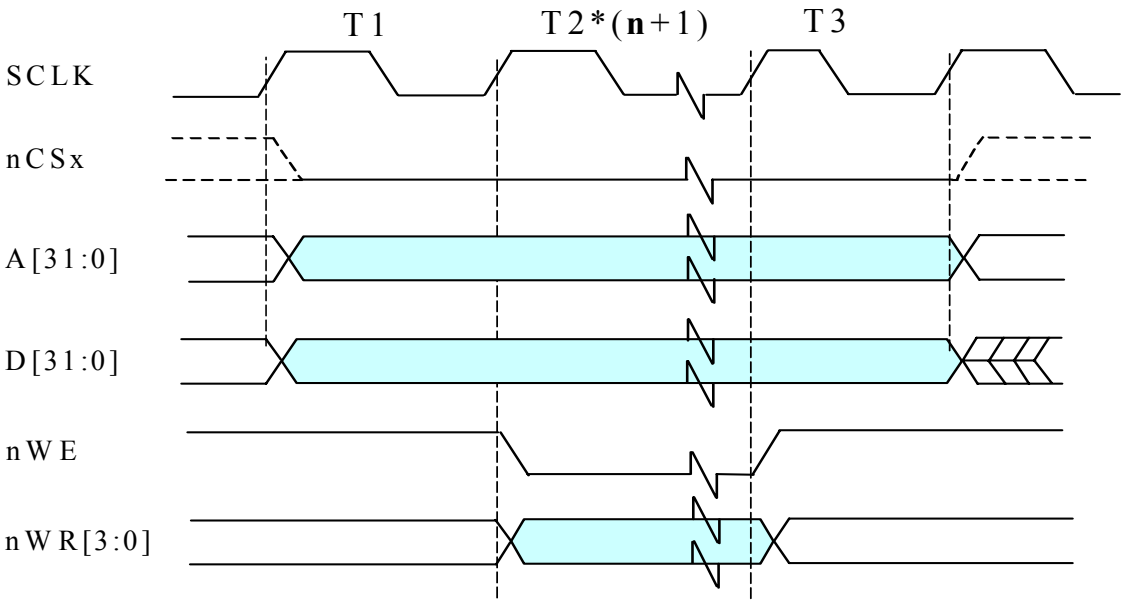


Рисунок 9.2. Запись в асинхронную память с n дополнительными тактами ожидания.

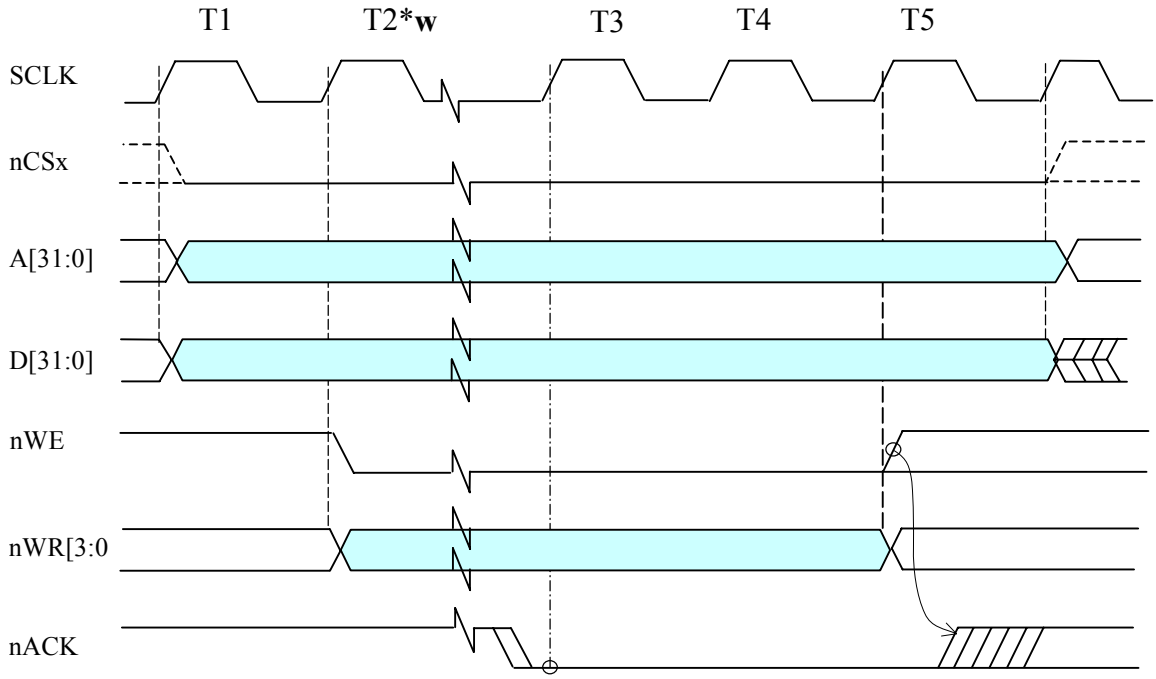


Рисунок 9.3. Запись в асинхронную память с ожиданием сигнала nACK.

Временные диаграммы чтения данных из асинхронной памяти приведены на Рисунок 9.4 - Рисунок 9.6.

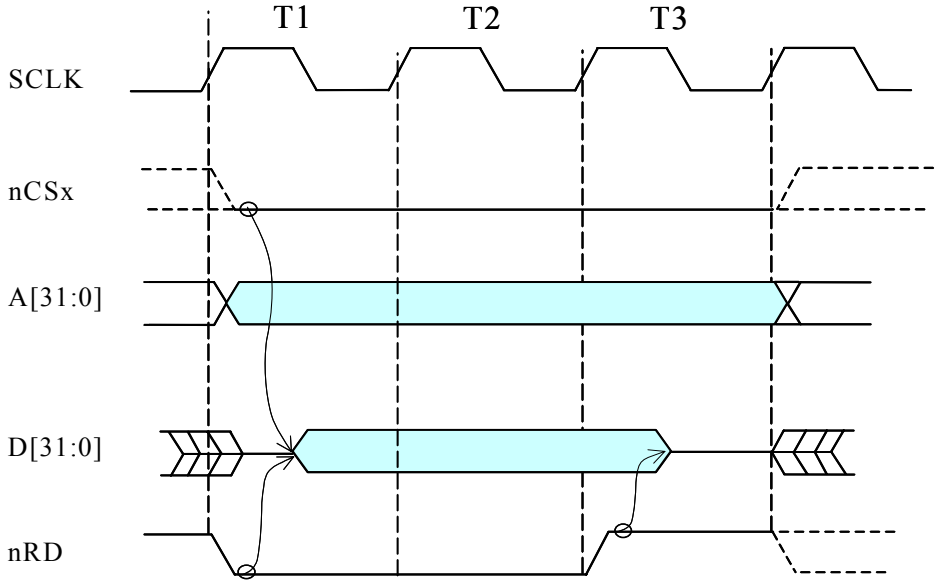


Рисунок 9.4.Чтение асинхронной памяти без дополнительных тактов ожидания.

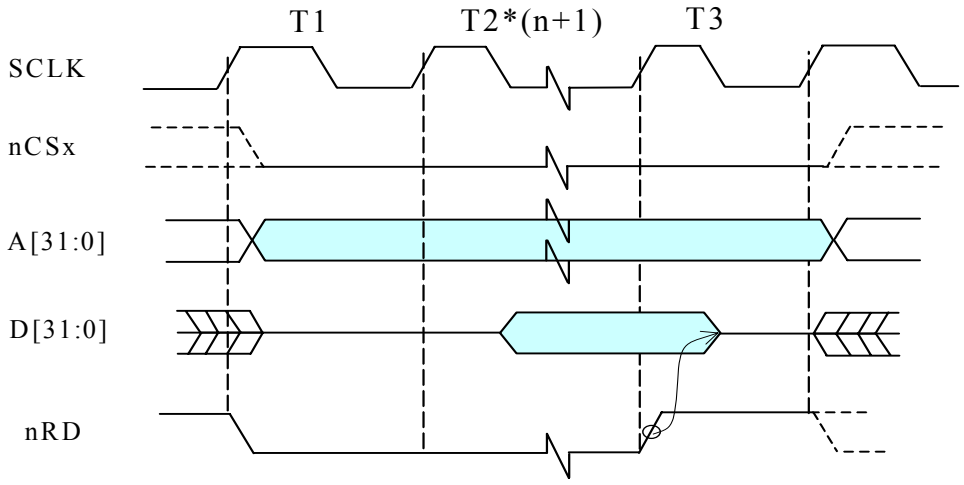


Рисунок 9.5. Чтение асинхронной памяти с n дополнительными тактами ожидания.

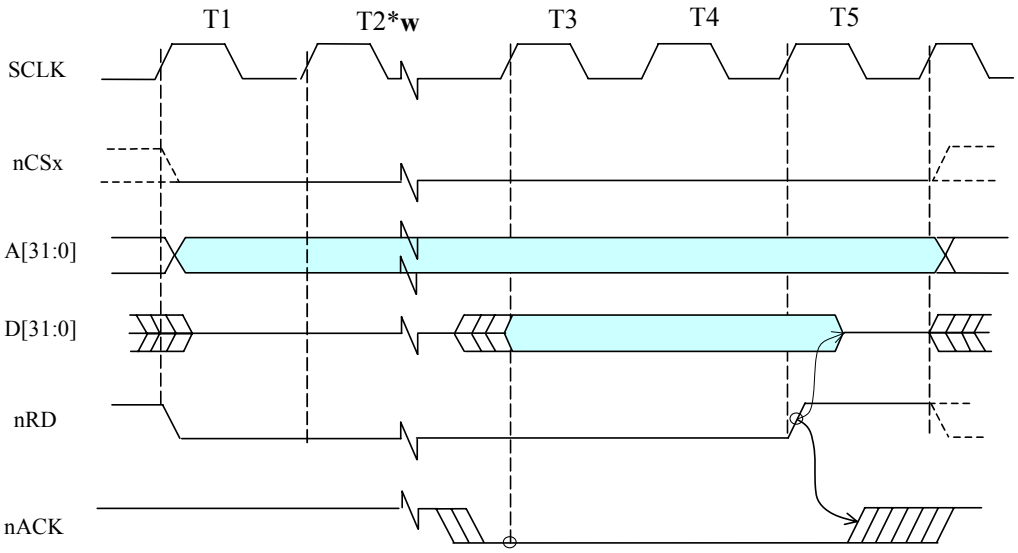


Рисунок 9.6. Чтение данных из асинхронной памяти с ожиданием сигнала nACK.

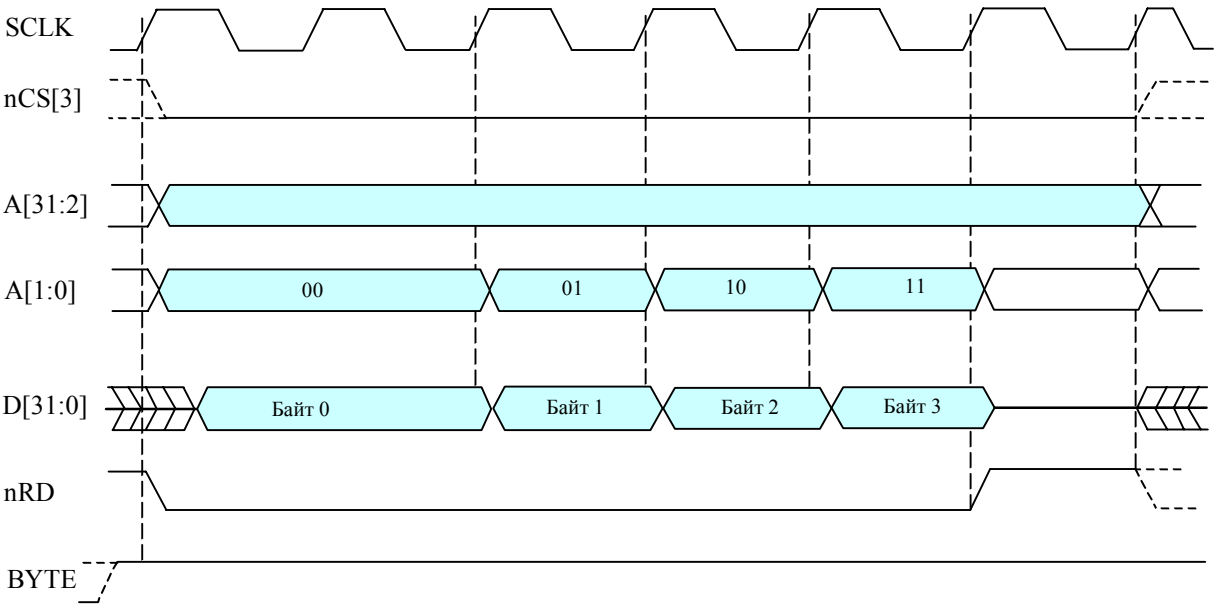


Рисунок 9.7. Чтение 32-разрядного слова из 8-разрядной асинхронной памяти (BYTE = 1, n = 0).

Если CPU выполняет программу из кэшируемой области внешней памяти, то загрузка строки кэш (процедура Refill) выполняются посредством чтения 4 слов в режиме burst. Адрес, по которому начинается burst, выровнен по 16-байтной границе. На Рисунок 9.8 приведена временная диаграмма выполнение процедуры Refill из 32-разрядной асинхронной памяти. На Рисунок 9.9 приведена временная диаграмма выполнение процедуры Refill из 8-разрядного ПЗУ.

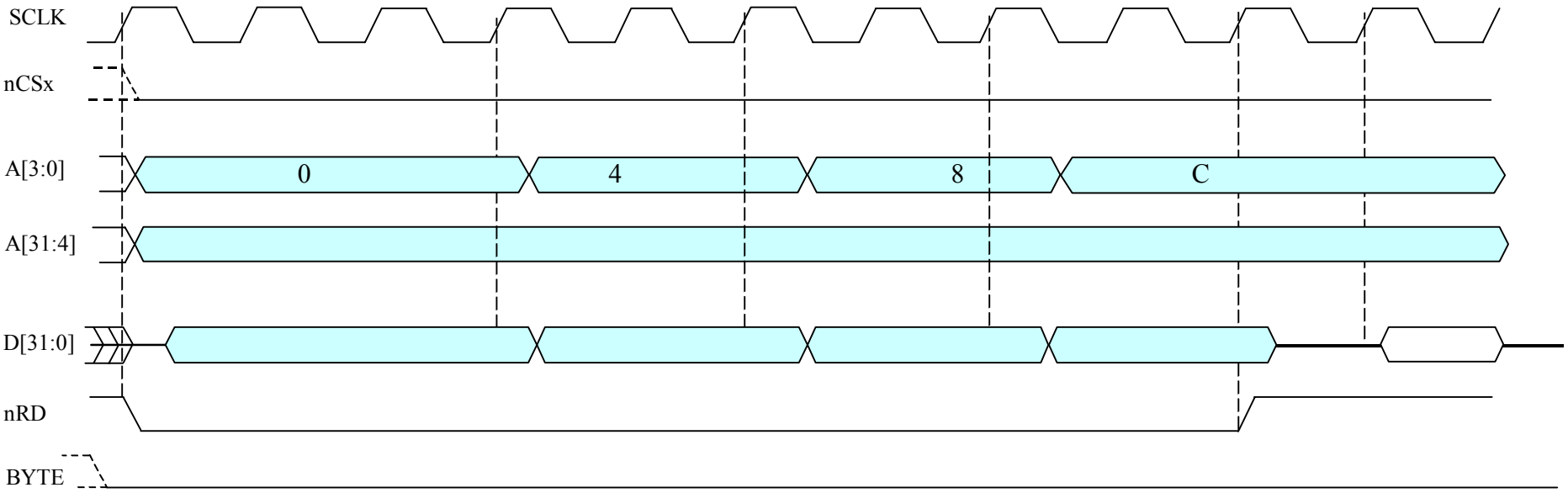


Рисунок 9.8. Выполнение процедуры Refill из 32-разрядной асинхронной памяти (BYTE = 0, n = 0).

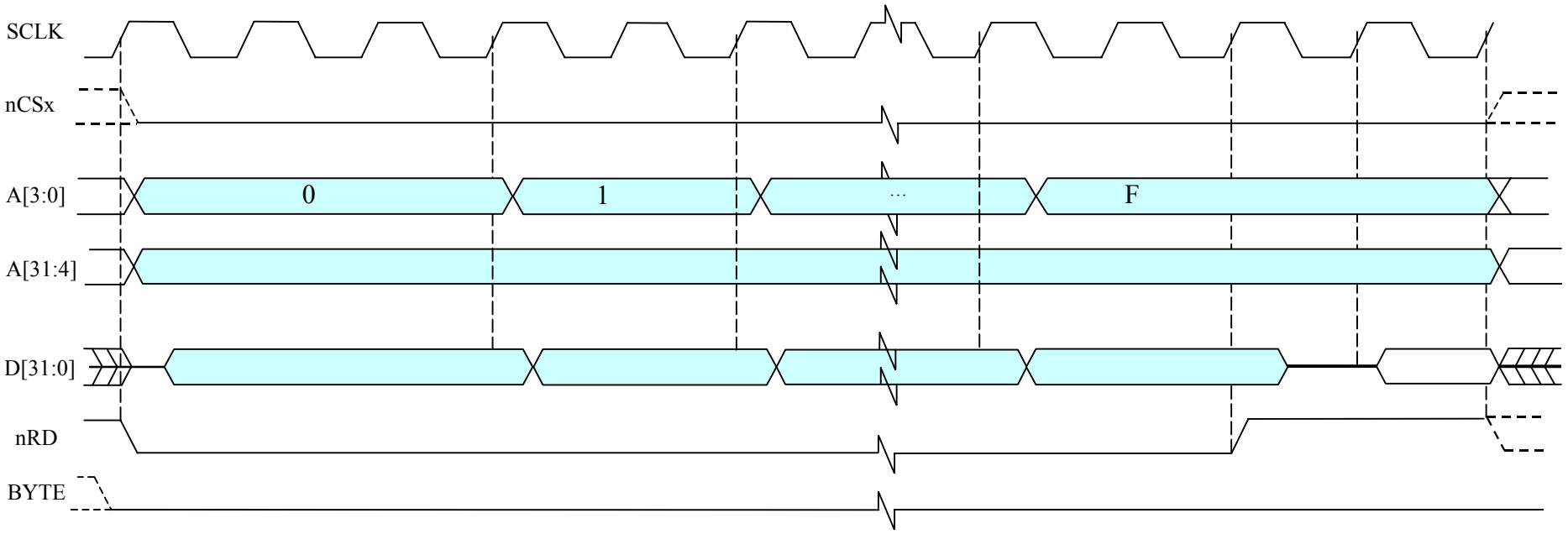


Рисунок 9.9. Выполнение процедуры Refill из 8-разрядной асинхронной памяти (BYTE = 1, n = 0).

9.3.3 Обмен данными с синхронной памятью

Временные диаграммы с синхронной памятью приведены на Рисунок 9.10 -Рисунок 9.16. Временные диаграммы инициализации и регенерации SDRAM приведены на Рисунок 9.17, Рисунок 9.18 соответственно.

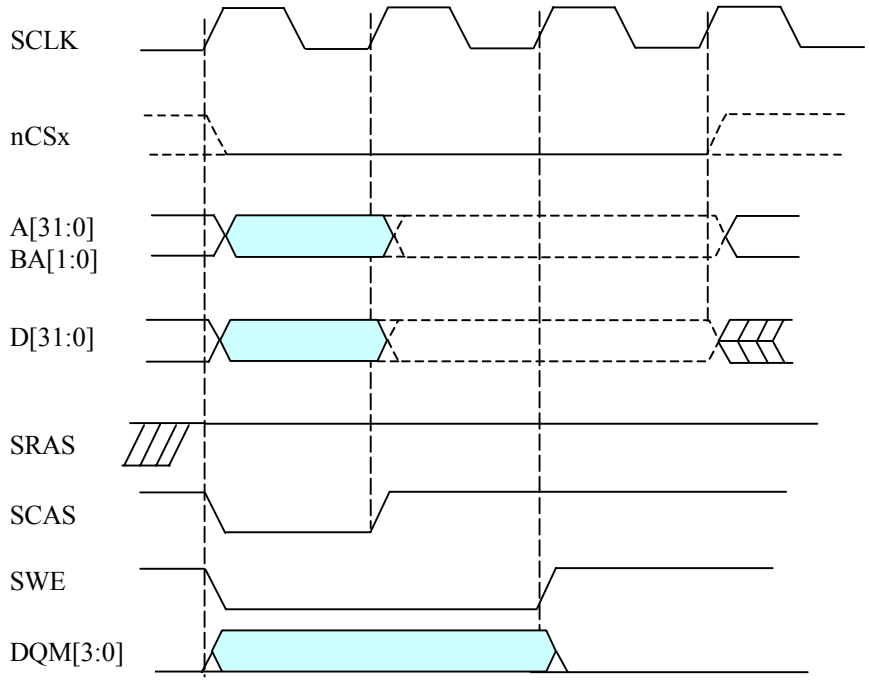


Рисунок 9.10. Запись одного слова данных в синхронную память.

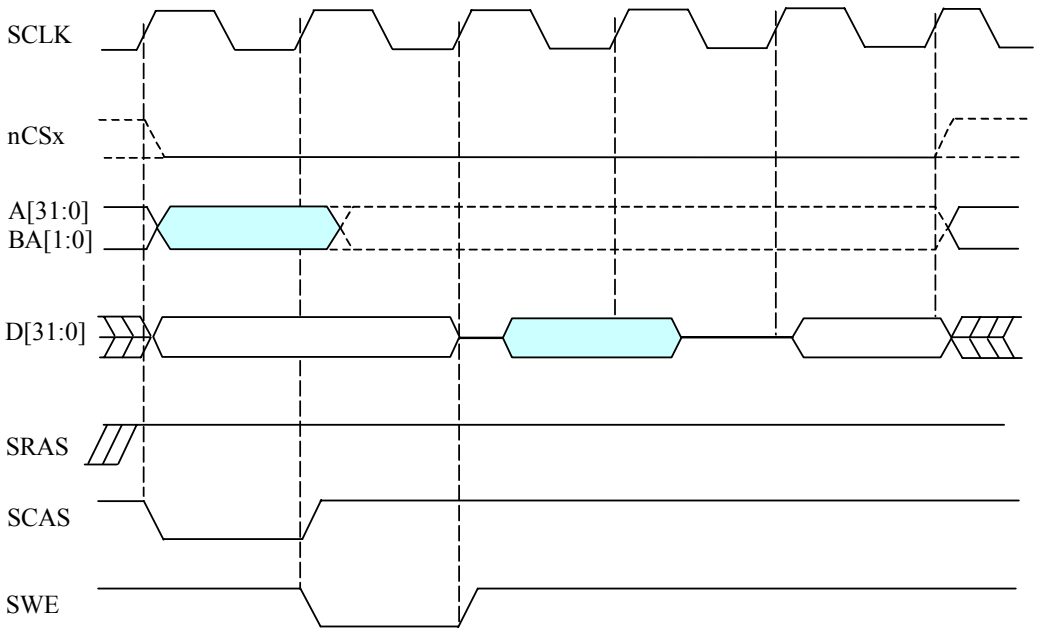


Рисунок 9.11. Чтение одного слова данных из синхронной памяти (здесь и далее CAS latency = 2)

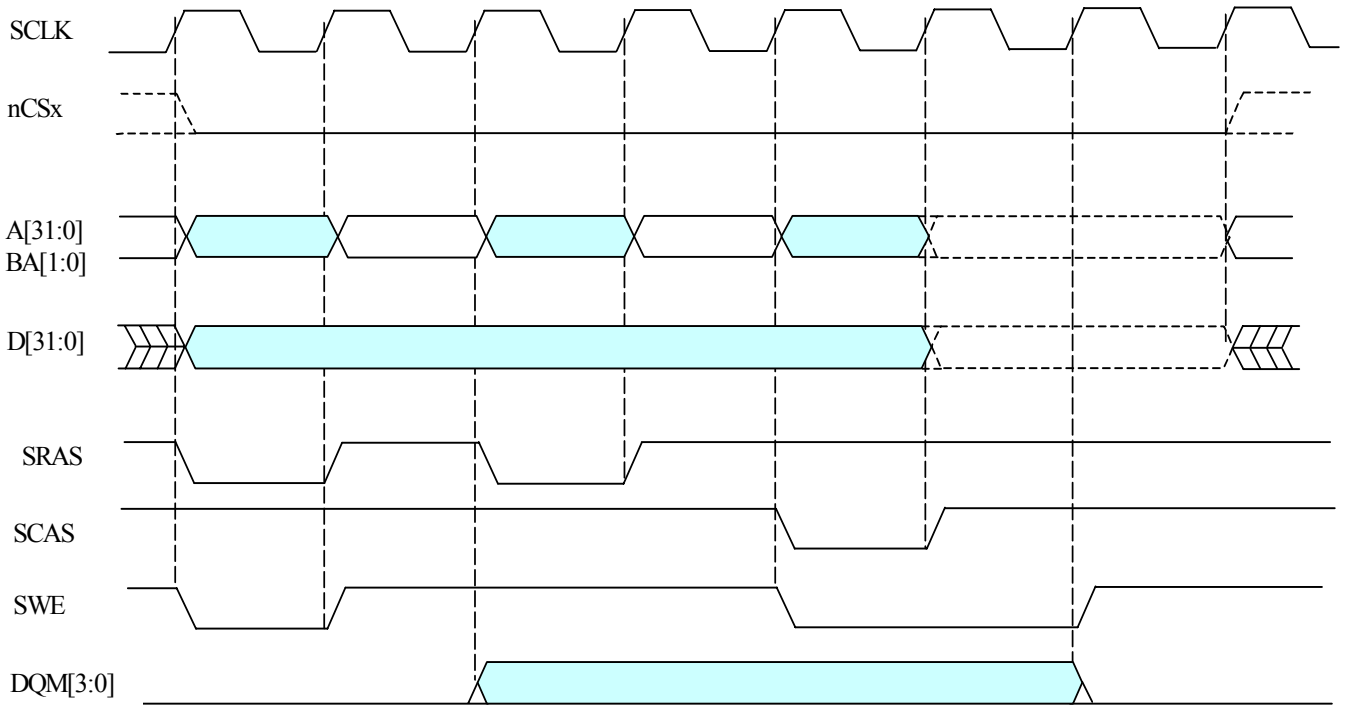


Рисунок 9.12. Запись одного слова данных в синхронную память с деактивизацией строки

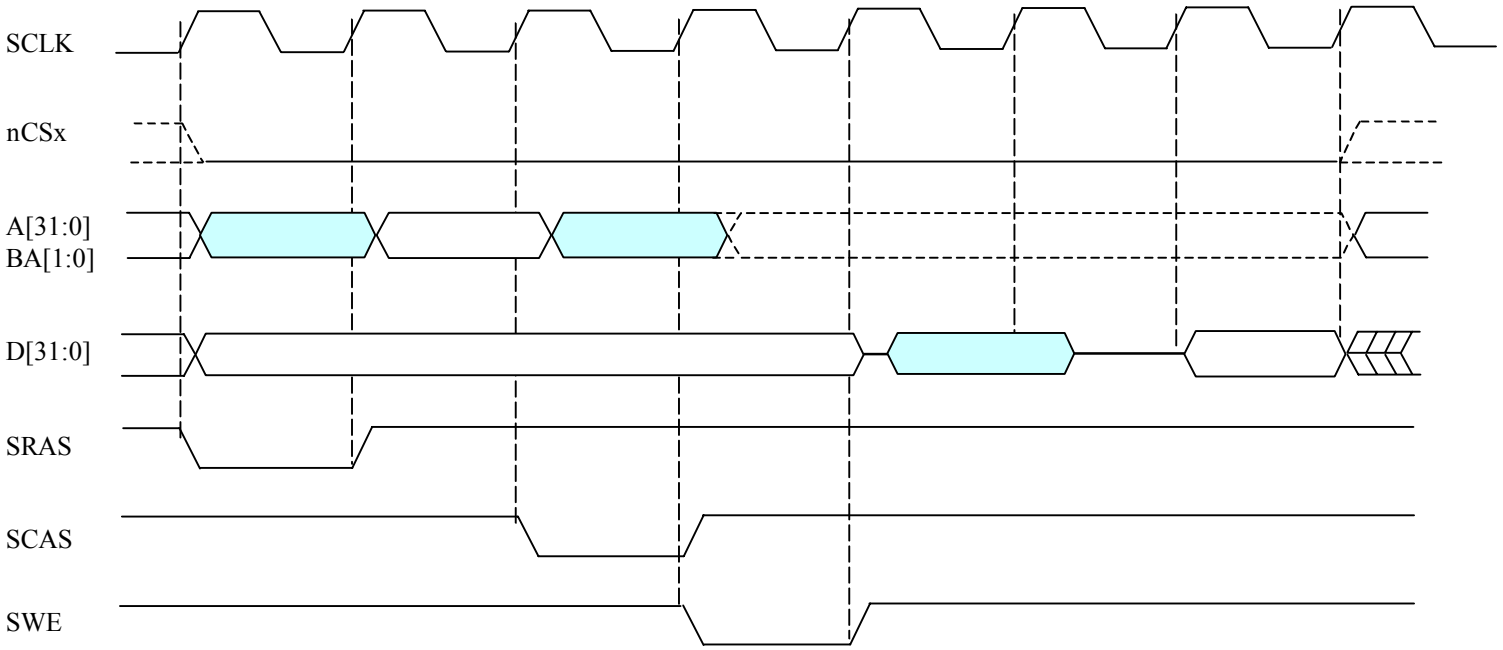


Рисунок 9.13. Чтение одного слова данных из синхронной памяти с активизацией строки.

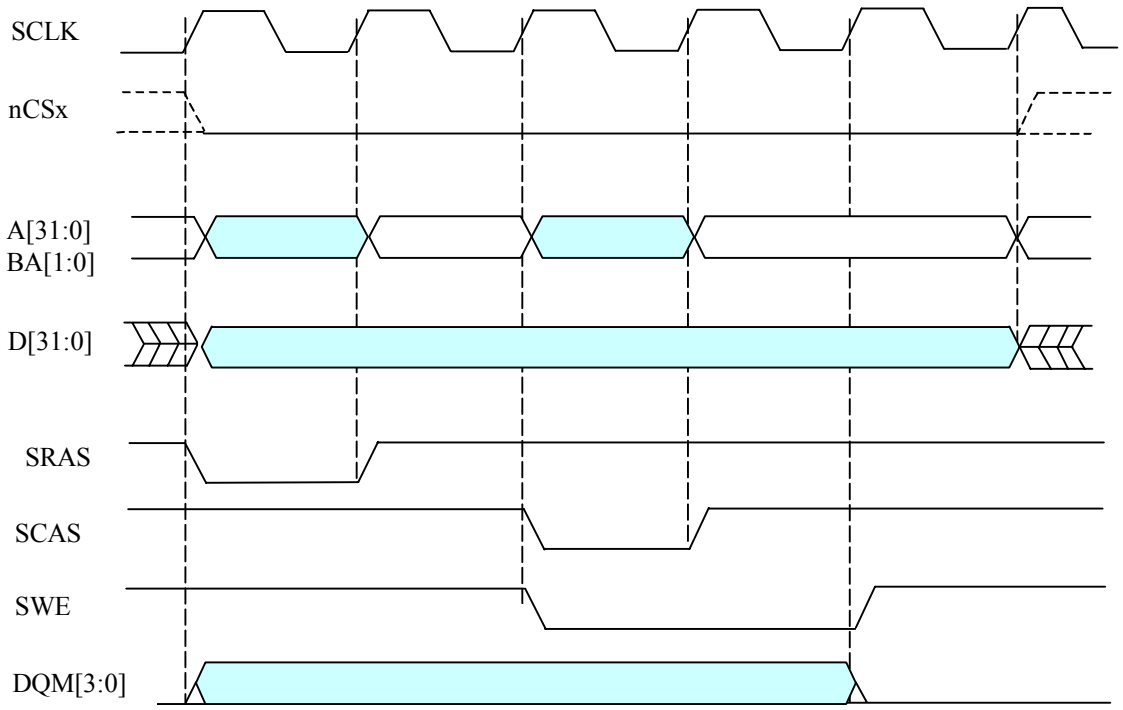


Рисунок 9.14. Запись одного слова данных в синхронную память с активизацией строки.

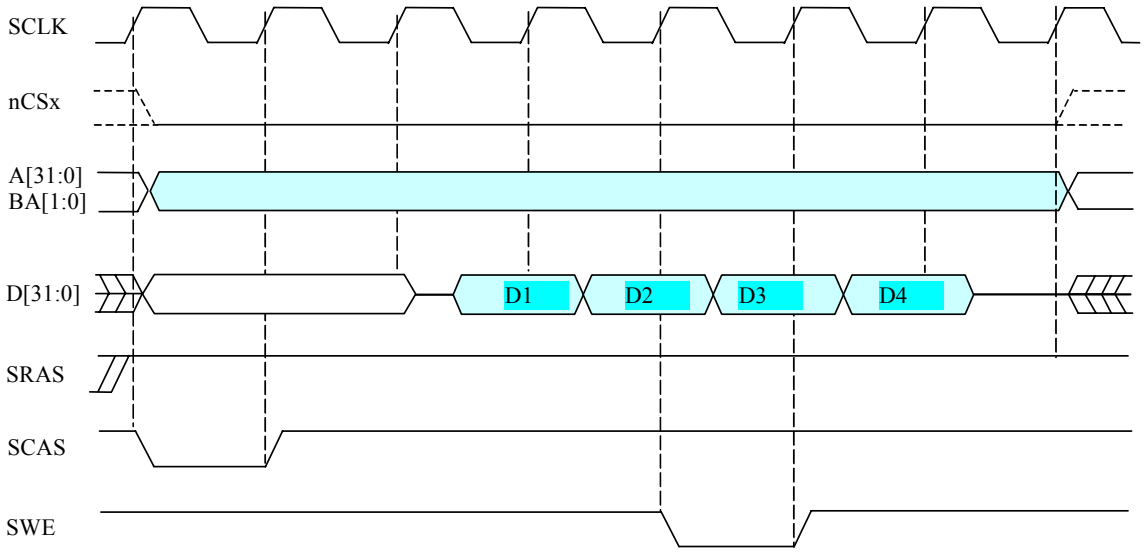


Рисунок 9.15. Чтение 4-х слов данных из синхронной памяти в режиме “burst“.

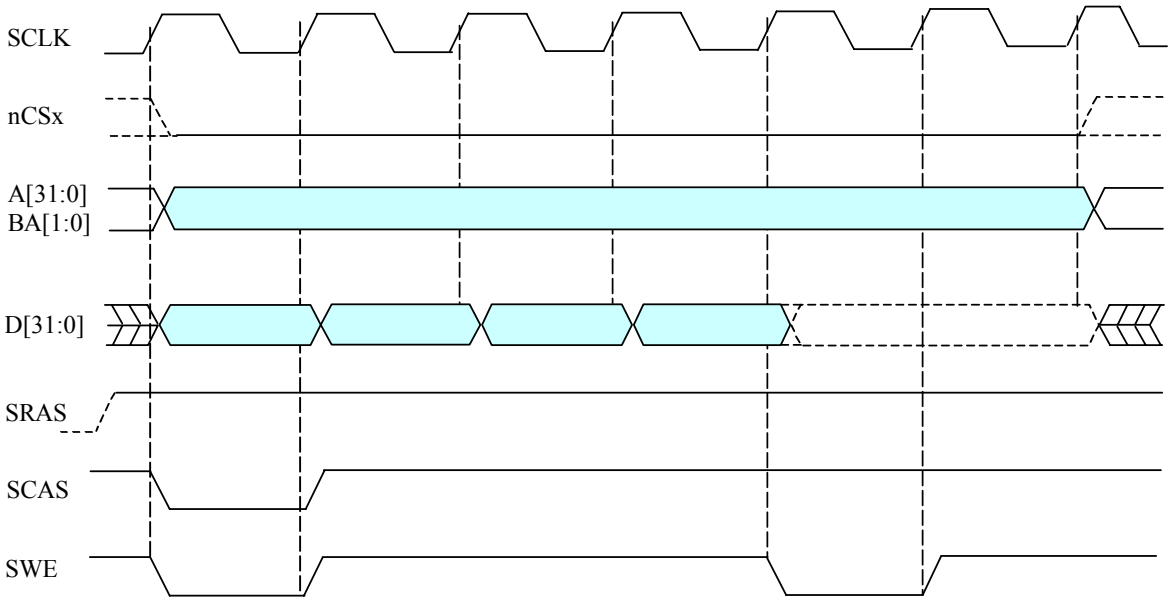


Рисунок 9.16. Запись 4-х слов данных в синхронную память в режиме “burst“.

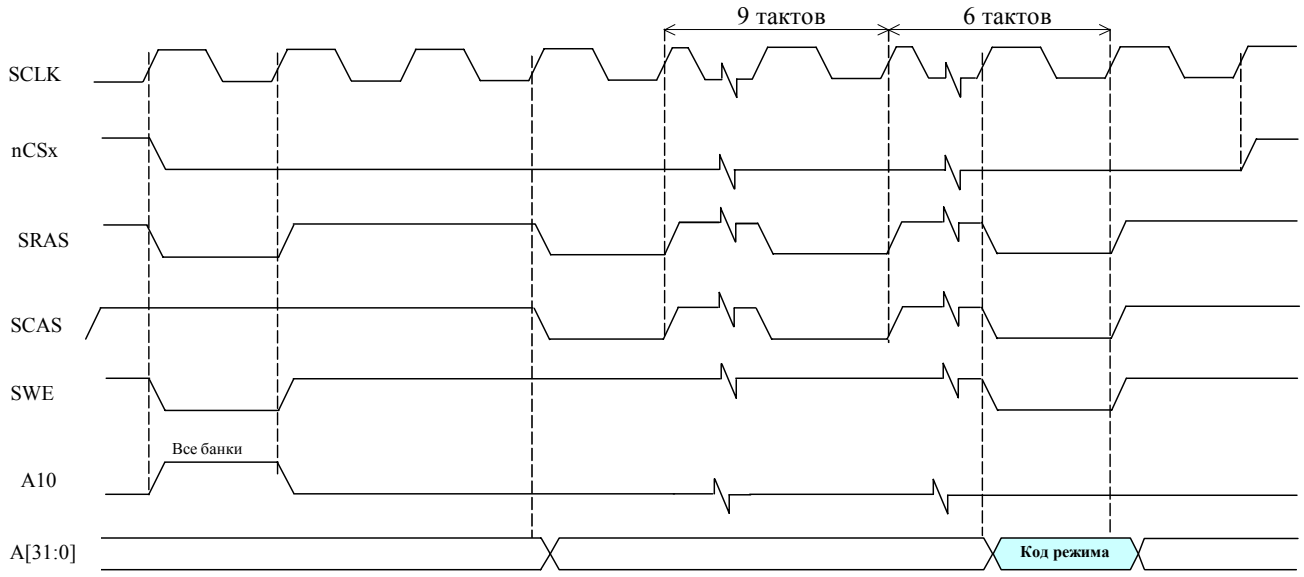


Рисунок 9.17. Инициализация синхронной памяти

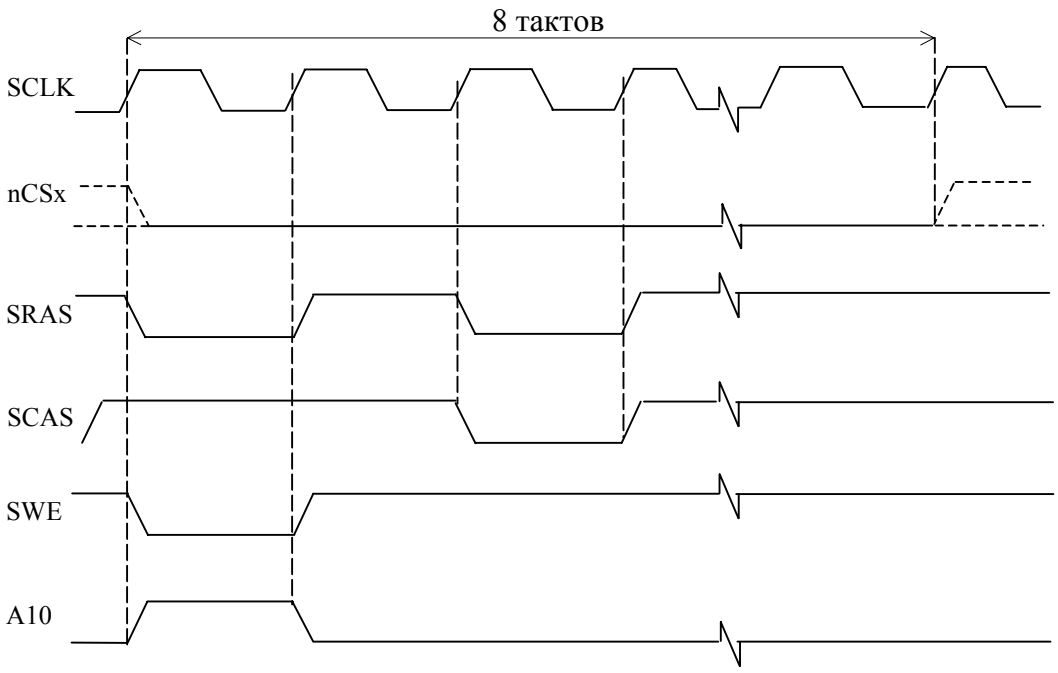


Рисунок 9.18. Временная диаграмма регенерация синхронной памяти.

9.3.4 Обмен данными в режиме Flyby

Режим Flyby используется контроллером DMA (каналы MemCh) для передачи данных между внешним устройством ввода-вывода и внешней памятью (как асинхронной, так и синхронной). Например, контроллер DMA может быть запрограммирован для передачи данных из аналого-цифрового преобразователя в SDRAM. Для выполнения передачи данных в режиме Flyby в соответствующем регистре CSR_MemCh необходимо установить бит 11.

При передаче данных в режиме Flyby MCT-01 отключается от шины данных, и активизирует внешнюю память и внешнее устройство ввода-вывода одновременно. Память управляется как обычно, а устройство ввода-вывода – при помощи сигналов nFLYBY (признак данного режима) , nOE (активизация выходных формирователей устройства ввода-вывода) и nCSIO[3:0] (выбор устройства ввода-вывода).

Каждому каналу MemCh может соответствовать свое устройство ввода-вывода. Выбор устройства ввода-вывода осуществляется посредством сигналов nCSIO[3:0]. Каналу MemCh0 соответствует низкий уровень на выводе nCSIO[0], каналу MemCh1 соответствует низкий уровень на выводе nCSIO[1], и так далее.

При работе с медленными внешними устройствами можно использовать сигнал nACK следующим образом. Если nFLYBY=1, то nACK=0. По сигналу nFLYBY=0 nACK переводится в «1» и удерживается в этом состоянии необходимое время. Для завершения обмена nACK переводится в состояние «0».

Временные диаграммы обмена данными в режиме Flyby приведены на Рисунок 9.19 - Рисунок 9.24 (WS=0, AE=0, CL=0).

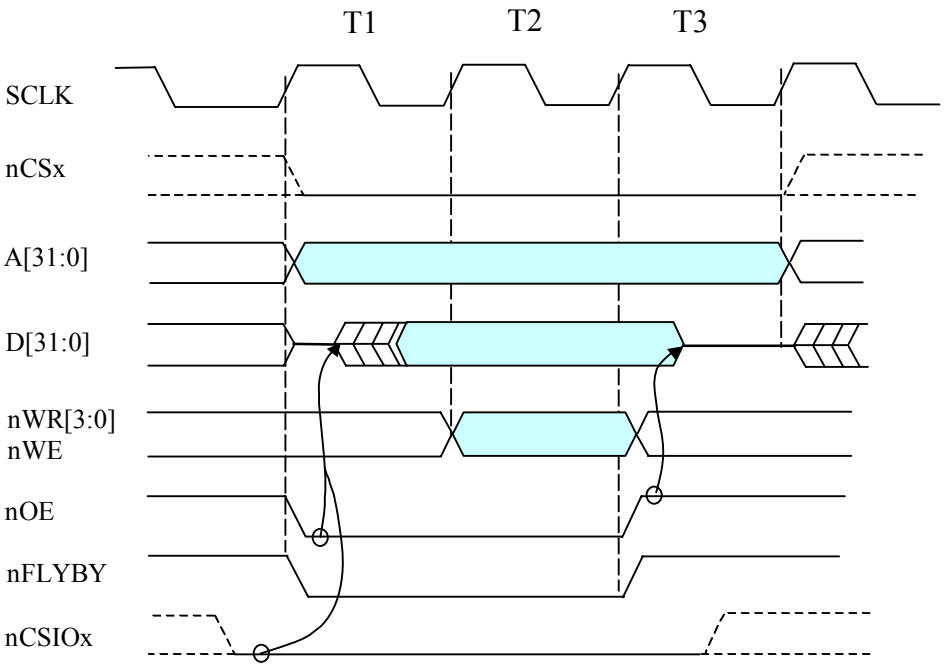


Рисунок 9.19. Передача одного слова данных из устройства ввода-вывода в асинхронную память.

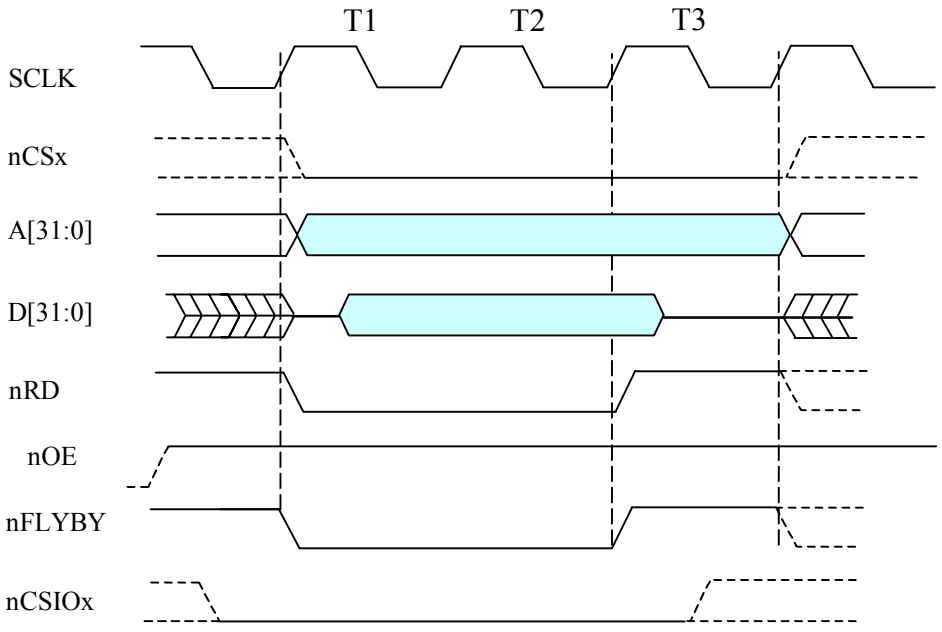


Рисунок 9.20. Передача одного слова данных из асинхронной памяти в устройство ввода-вывода.

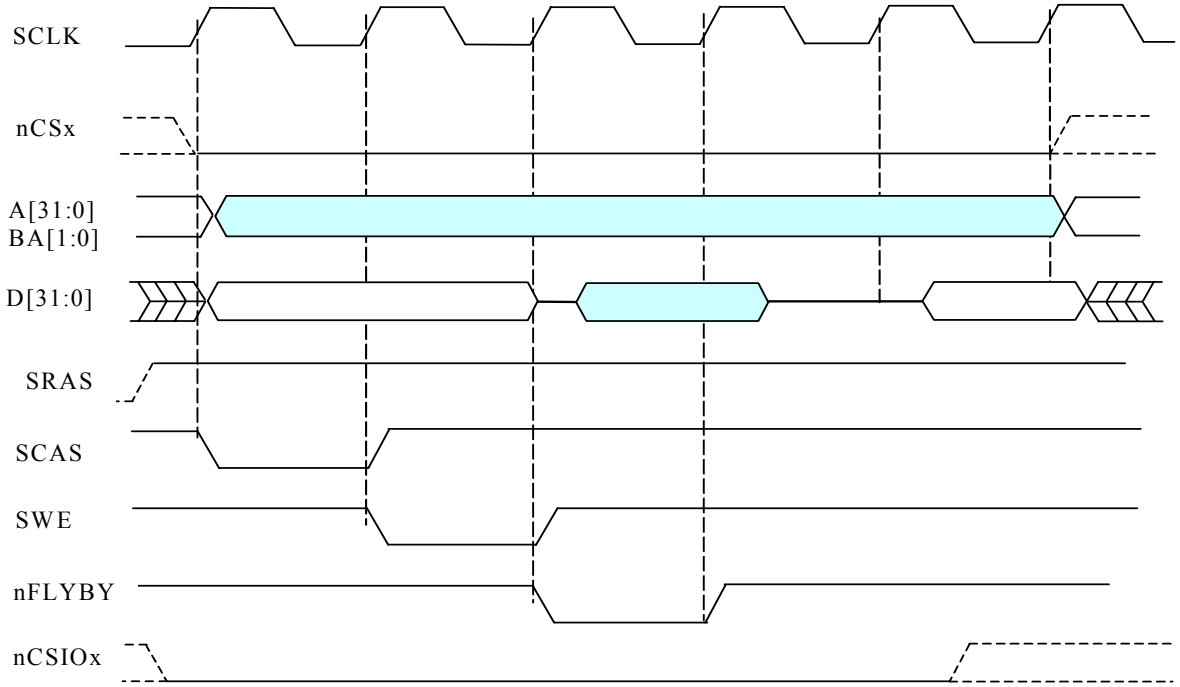


Рисунок 9.21. Передача одного слова данных из синхронной памяти в устройство ввода-вывода.

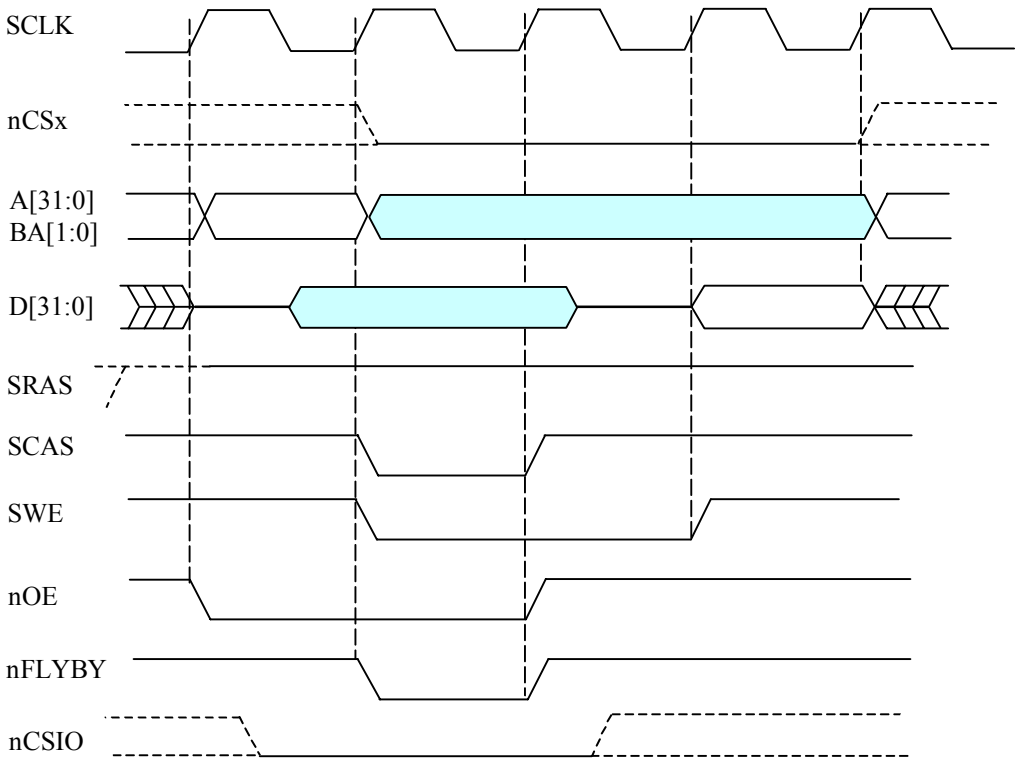


Рисунок 9.22. Передача одного слова данных из устройства ввода-вывода в синхронную память.

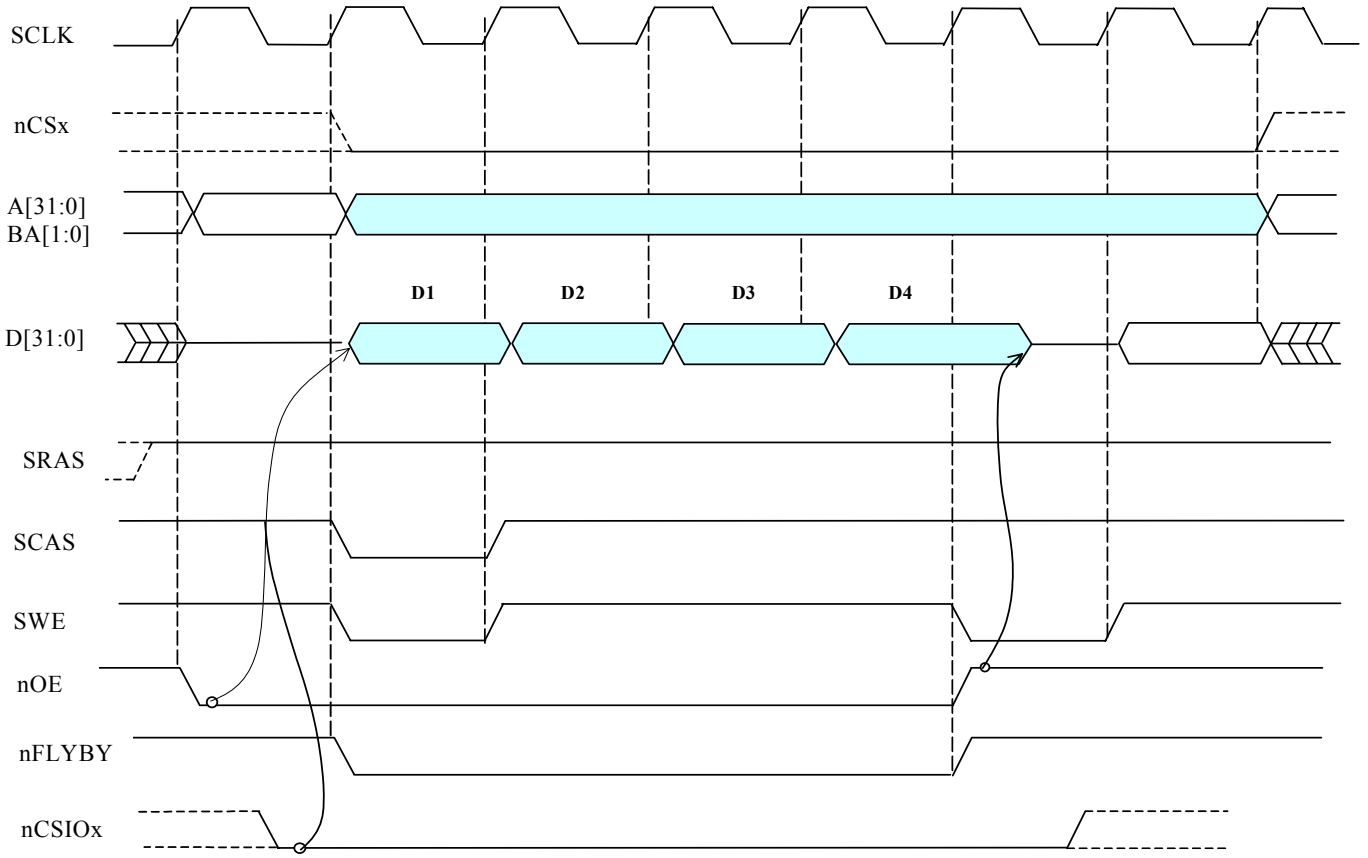


Рисунок 9.23. Передача 4-х слов данных из устройства ввода-вывода в синхронную память.

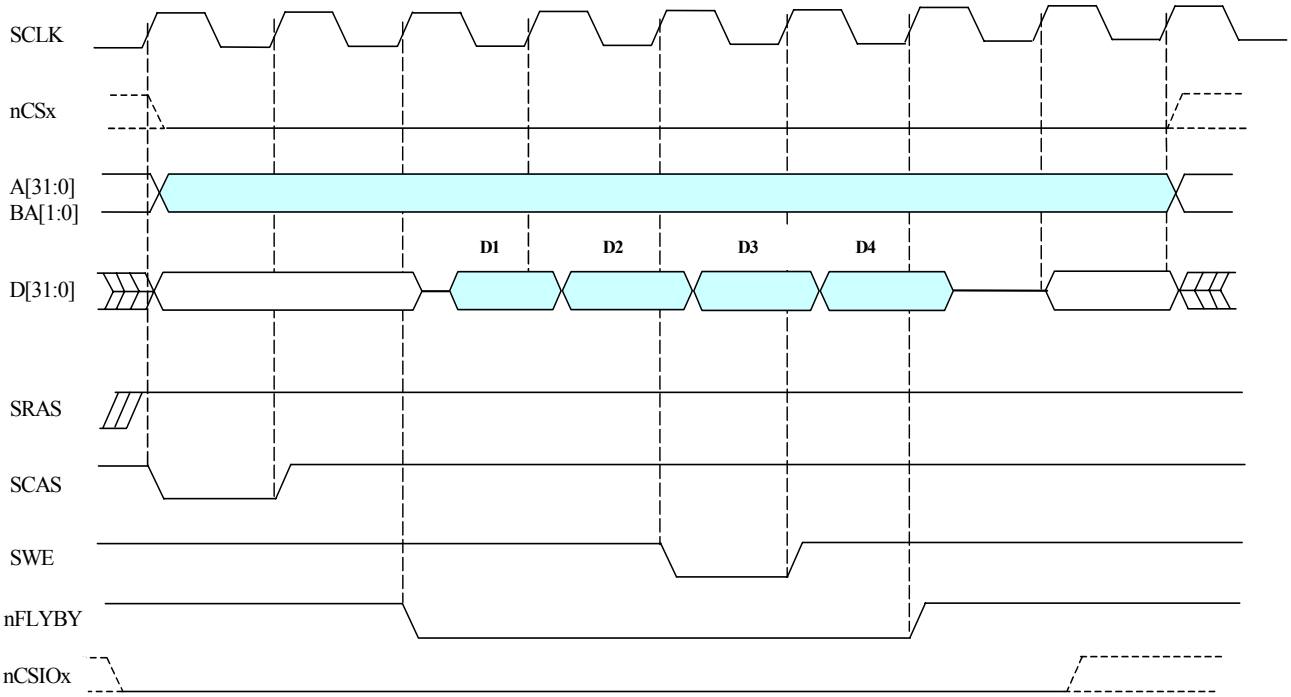


Рисунок 9.24. Передача 4-слов данных из синхронной памяти в устройство ввода-вывода.

9.4 Рекомендации по подключению внешней памяти

9.4.1 Память типа SDRAM

Выводы адреса микросхем типа SDRAM подключаются к выводам шины адреса порта внешней памяти следующим образом:

- номер банка SDRAM – к выводам BA[1:0];
- адрес A[12:0] SDRAM – к выводам A[14:13], A10, A[11:2] соответственно.

9.4.2 Память типа Flash

К микросхеме МСТ-01 можно подключать 32-разрядную или 8-разрядную память типа Flash.

32-разрядная память Flash подключается к МСТ-01 аналогично статической памяти. Как правило, она подключается к сигналу выборки памяти nCS[3] и используется для старта МСТ-01. Но при необходимости, 32-разрядная память Flash может быть подключена к любому из 4-х сигналов выборки памяти nCS[3:0].

8-разрядная память Flash подключается только к сигналу выборки памяти nCS[3], а на вход BYTE МСТ-01 необходимо подать высокий уровень. Выходную адресную шину МСТ-01 необходимо подключать к памяти Flash, начиная с 0 разряда (к 32-разрядной памяти адрес подключается, начиная со 2 разряда).

При использовании памяти типа Flash возможны два варианта ее программирования:

1. Микросхемы этой памяти программируется на программаторе и потом распаивается на плату или устанавливаются в контактирующее устройство.
2. Микросхемы этой памяти программируются на плате через порт JTAG микросхемы МСТ-01. Для процесса программирования необходим специальный драйвер, который не входит в состав MC Studio.

Если используется 8-разрядная память Flash и требуется ее программирование в составе платы через порт JTAG МСТ-01, то при ее проектировании необходимо иметь в виду следующую особенность микросхемы МСТ-01. В этой микросхеме разряды адреса A[1:0] изменяются только при чтении из 8-разрядной памяти, а при записи в память (8- или 32-разрядную) они имеют постоянно нулевое состояние. Поэтому, для обеспечения записи в 8-разрядную память Flash через порт JTAG разряды адреса A[1:0] от МСТ-01 при помощи внешней логики необходимо объединить по логическому ИЛИ с двумя сигналами, при помощи которых можно перебрать все состояния шины адреса микросхемы памяти Flash.

10. УНИВЕРСАЛЬНЫЙ АСИНХРОННЫЙ ПОРТ (UART)

10.1 Общие положения

Универсальный асинхронный порт (далее UART) имеет следующие характеристики:

- по архитектуре совместим с UART 16550;
- частота приема и передачи данных – от 50 до 1 Мбод;
- FIFO для приема и передачи данных имеют объем по 16 байт;
- полностью программируемые параметры последовательного интерфейса: длина символа от 5 до 8 бит; генерация и обнаружение бита четности; генерация стопового бита длиной 1, 1.5 или 2 бита;
- диагностический режим внутренней петли;
- эмуляция символьных ошибок;
- функция управления модемом (CTS, RTS, DSR, DTR, RI, DCD).

Структурная схема порта UART приведена на Рисунок 10.1.

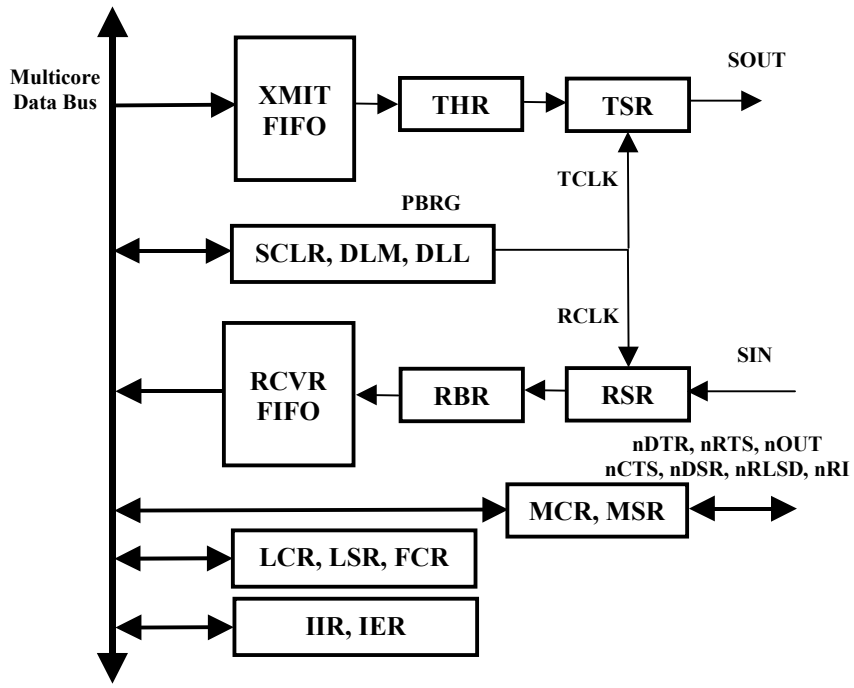


Рисунок 10.1. Структурная схема UART.

Передаваемые данные записываются в регистр THR, а затем аппаратно переписываются в передающий сдвигающий регистр (TSR), если он пуст. После этого в регистр THR могут быть записаны следующие данные.

После приема данных в приемный сдвигающий регистр (RSR) данные переписываются в регистр RBR, если он не занят.

Назначение внешних выводов UART приведено в Таблица 10.1.

Таблица 10.1. Внешние выводы UART.

Название вывода	Тип вывода	Описание
SIN	I	Вход последовательных данных
SOUT	O	Выход последовательных данных
nDTR	O	Готовность UART к установлению связи (Data Terminal Ready)
nRTS	O	Готовность UART к обмену данными (Request To Send)
nOUT1	O	Выход общего назначения
nOUT2	O	Выход общего назначения
nCTS	I	Готовность модема к обмену данными (Clear To Send)
nDSR	I	Готовность модема к установлению связи (Data Set Ready)
nDCD	I	Признак обнаружения модемом несущей частоты (Receiver Line Signal Detect)
nRI	I	Признак обнаружения модемом телефонного звонка (Ring Indicator)

10.2 Регистры UART

10.2.1 Общие положения

Перечень регистров UART приведен в Таблица 10.2.

Таблица 10.2. Перечень регистров UART

Условное Обозначение регистра	Название регистра	Смещение	Доступ (R-чтение, W-запись)
RBR	Приемный буферный регистр	0 (DLAB=0)	R
THR	Передающий буферный регистр	0 (DLAB=0)	W
IER	Регистр разрешения прерываний	1 (DLAB=0)	R/W
IIR	Регистр идентификации прерывания	2	R
FCR	Регистр управления FIFO	2	W
LCR	Регистр управления линией	3	R/W
MCR	Регистр управления модемом	4	R/W

Условное Обозначение регистра	Название регистра	Смещение	Доступ (R-чтение, W-запись)
LSR	Регистр состояния линии	5	R
MSR	Регистр состояния модема	6	R/W
SPR	Регистр Scratch Pad	7	R/W
DLL	Регистр делителя младший	0 (DLAB=1)	R/W
DLM	Регистр делителя старший	1 (DLAB=1)	R/W
SCLR	Регистр предделителя (scaler)	5	W

10.2.2 Регистр LCR

Формат регистра LCR приведен в Таблица 10.3.

Таблица 10.3. Формат регистра LCR

Номер бита	Условное Обозначение	Назначение
1:0	WLS (Word Length Select)	Количество бит данных в передаваемом символе: 00 -5 бит, 01 -6 бит, 10 -7 бит, 11 -8 бит.
2	STB (Number Stop Bits)	Количество стоп-бит: 0 - 1 стоп-бит, 1 - 2 стоп-бита (для 5-битного символа стоп-бит имеет длину 1,5 бита). Приемник анализирует только первый стоп бит.
3	PEN (Parity Enable)	Разрешение генерации (передатчик) или проверки (приемник) контрольного бита: 1 – контрольный бит (паритет или постоянный) разрешен, 0 – запрещен.
4	EPS (Even Parity Select)	Выбор типа контроля (при PEN=1): 0 – нечетность, 1 – четность.
5	STP (Stick Parity)	Принудительное формирование бита паритета: 0 – контрольный бит генерируется в соответствии с паритетом выводимого символа, 1 – постоянное значение контрольного бита: при EPS=1 - нулевое, при EPS=0 – единичное.
6	SBC (Set Break Control)	Формирование обрыва линии: 0 – нормальная работа; 1 – на выходе SOUT устанавливается низкий уровень (Spacing level). Это влияет только на выход SOUT, а не на логику передачи символа.
7	DLAB (Divisor Latch Access bit)	Управление доступом к регистрам: 0 – разрешен доступ к регистрам RBR, THR, IER; 1 – разрешен доступ к регистрам DLL, DLM

Исходное состояние регистра LCR – нули.

Бит SBC используется как признак «Внимание» для приемного терминала, подключенному к выходу UART. Для того чтобы не было передано ошибочного символа при использовании бита SBC, необходимо выполнять следующую последовательность действий:

- Загрузить в регистр THR все нули по признаку THRE=1;
- Установить SBC=1 по следующему THRE=1;
- Дождаться TEMT=1.

Для восстановления нормальной передачи необходимо установить SBC=0.

10.2.3 Регистр FCR

Формат регистра FCR приведен в Таблица 10.4.

Таблица 10.4. Формат регистра FCR

Номер бита	Условное Обозначение	Назначение
0	FEWO (FIFO Enable)	Разрешение работы XMIT и RCVR FIFO: 0 – символьный режим; 1 – режим FIFO. При изменении состояния этого бита, данные из FIFO, не удаляются. Запись в биты RFR, TFR, RFTL выполняется, если FEWO=1.
1	RFR (Receiver FIFO Reset)	Установка RCVR FIFO в исходное состояние. Регистр RSR не обнуляется. После записи 1 в этот бит он автоматически сбрасывается.
2	TFR (Transmitter FIFO Reset)	Установка XMIT FIFO в исходное состояние. Регистр TSR не обнуляется. После записи 1 в этот бит он автоматически сбрасывается.
5:3	-	Резерв
7:6	RFTL (RCVR FIFO Trigger Level)	Порог заполнения RCVR FIFO (в байтах), при котором формируется прерывание: 00 – 1; 01 – 4; 10 – 8; 11 – 14.

Исходное состояние регистра FCR – нули.

10.2.4 Регистр LSR

Формат регистра LSR приведен в Таблица 10.5.

Таблица 10.5. Формат регистра LSR

Номер бита	Условное Обозначение	Назначение
0	RDR (Receiver Data Ready)	Готовность данных. Устанавливается после приема символа данных и передачи его в регистр RBR или FIFO. Сбрасывается после чтения регистра RBR (в символьном режиме) или чтения всего содержимого RCVR FIFO (в режиме FIFO)
1	OE (Overrun Error)	Ошибка переполнения. Устанавливается, если содержимое регистра RBR не было прочитано, в сдвигающий регистр принят следующий символ и начат прием очередного символа. При этом новый символ записывается в сдвигающий регистр вместо старого. В режиме FIFO устанавливается, если после перехода порогового (trigger) уровня FIFO заполнено до конца, во входной сдвигающий регистр полностью принят следующий символ и начат прием очередного символа. При этом в FIFO ничего не передается. Бит сбрасывается при чтении содержимого регистра LSR.
2	PE (Parity Error)	Ошибка контрольного бита (паритета или фиксированного). В режиме FIFO этот бит показывает на ошибку в символе, находящемся наверху FIFO. Бит сбрасывается при чтении содержимого регистра LSR.
3	FE (Framing Error)	Ошибка кадра. Устанавливается, если стоп-бит равен нулю (Spacing level). В режиме FIFO этот бит показывает на ошибку в символе, находящемся наверху FIFO. После этой ошибки UART пересинхронизируется. Бит сбрасывается при чтении содержимого регистра LSR.
4	BI (Break Interrupt)	Обрыв линии. Устанавливается, если вход приема данных находится в состоянии 0 (Spacing level) не менее чем время передачи всего символа. В режиме FIFO этот бит показывает на ошибку в символе, находящемся наверху FIFO. При возникновении этой ситуации, в FIFO загружается только один нулевой символ. Прием следующих символов разрешается после того, как вход приема данных перейдет в единичное состояние (Marking state) и будет принят действительный стартовый бит. Бит сбрасывается при чтении содержимого регистра LSR.

Продолжение Таблица 10.5

Номер бита	Условное Обозначение	Назначение
5	THRE (Transmitter Holding Register Empty)	Передающий буферный регистр пуст. Показывает, что UART готов принять следующий символ для передачи. Устанавливается, когда содержимое регистра THR передается в передающий сдвигающий регистр. Одновременно с этим генерируется прерывание THREI, если оно разрешено. Бит сбрасывается при записи символа в регистр THR. В режиме FIFO этот бит устанавливается, когда XMIT FIFO пусто, и сбрасывается, если в XMIT FIFO записывается хотя бы один символ.
6	TEMT (Transmitter Empty)	Передатчик пуст. Устанавливается, если регистры THR и TSR пусты. Имеет нулевое состояние, если хотя бы один из регистров THR и TSR не пуст. В режиме FIFO этот бит устанавливается, если нет символов ни в XMIT FIFO, ни в регистре TSR.
7	EIRF (Error in RCVR FIFO)	Наличие хотя бы одного признака ошибки в FIFO. В символьном режиме этот бит всегда равен нулю. Бит сбрасывается при чтении содержимого регистра LSR, если в FIFO нет больше признаков ошибок.

Исходное состояние бит THRE, TEMT – 1, остальных – 0.

Установка бит OE, PE, FE, BI приводит к формированию прерыванию по состоянию входа приема данных (Receiver Line Status Interrupt), если это прерывание разрешено.

10.2.5 Регистр IER

Формат регистра IER приведен в Таблице 10.6. Исходное состояние регистра IER – нули.

Таблица 10.6. Формат регистра IER

Номер бита	Условное Обозначение	Назначение
0	ERBI	Разрешение прерывания по наличию принятых данных (RDAI), а также по таймауту (CTI)
1	ETBEI	Разрешение прерывания по отсутствию данных в регистре THR (THREI)
2	ERLSI	Разрешение прерывания по статусу приема данных (RLSI)
3	EMSI	Разрешение прерывания по статусу модема (MSI)
7:4	-	Резерв

10.2.6 Регистр IIR

Формат регистра IIR приведен в Таблица 10.7.

Таблица 10.7. Формат регистра IIR

Номер бита	Условное Обозначение	Назначение
0	IP (Interrupt Pending)	Признак наличия прерывания: 0 – есть прерывание; 1 – нет прерывания.
3:1	IID[2:0]	Код идентификации прерывания в соответствии с Таблица 10.8.
5:4	-	Резерв
7:6	FE	Признак разрешения работы RCVR и XMIT FIFO

Исходное состояние бита IP – 1, остальных – 0.

Таблица 10.8. Идентификация прерываний

Код Поля ID[2:0]	Уровень Приоритета (1 – наивысший)	Тип Прерывания	Причина прерывания	Условие Сброса Прерывания
011	1	Статус приема данных (RLSI – Receiver Line Status Interrupt)	OE - Overrun Error; PE - Parity Error; FE - Framing Error; BI - Break Interrupt.	Чтение содержимого регистра LSR. Чтение из FIFO символа, по которому сформировано это прерывание. Обнуление FIFO.
010	2	Наличие принятых данных (RDAI – Received Data Available Interrupt)	Наличие данных в регистре RBR или достижение заданного порога FIFO	Чтение содержимого регистра RBR. Считывание данных из FIFO до уровня ниже порогового.
110	2	Таймаут (CTI – Character Timeout Interrupt)	С момента приема последнего символа в RCVR FIFO прошло время, равное длительности передачи 4-х символов и не было ни чтения FIFO, ни приема очередного символа.	Чтение содержимого регистра RBR. Прием очередного символа. Сброс FIFO.
001	3	Регистр THR пуст (THREI – Transmitter Holding Register Empty Interrupt)	Регистр THR пуст	Запись символа в регистр THR
000	4	Статус модема (MSI – Modem Status Interrupt)	Изменение состояния сигналов на входах порта nCTS, nDSR, nRI, nDCD	Чтение содержимого регистра MSR.

10.2.7 Регистр MCR

Формат регистра MCR приведен в Таблица 10.9.

Таблица 10.9. Формат регистра MCR

Номер бита	Условное Обозначение	Назначение
0	DTR	Управление выходом nDTR: 0 – на выходе высокий уровень; 1 – на выходе низкий уровень.
1	RTS	Управление выходом nRTS: 0 – на выходе высокий уровень; 1 – на выходе низкий уровень.
2	Out 1	Управление выходом OUT1: 0 – на выходе высокий уровень; 1 - на выходе низкий уровень.
3	Out 2	Управление выходом OUT1: 0 – на выходе высокий уровень; 1 - на выходе низкий уровень.
4	LOOP	Режим петли. Используется для тестирования UART. При установке этого бита в 1 выполняется следующее: На выходе SOUT UART устанавливается высокий уровень; Вход SIN UART отключается от внешнего вывода; Выход регистра TSR подключается к входу регистра RSR; На выходах nDTR, nRTS, nOUT1, nOUT2 устанавливаются высокие уровни; Входы nCTS, nDSR, nDCD, nRI UART отключаются от внешних выводов; Выходы разрядов DTR, RTS, Out 1, Out 2 регистра MCR подключаются к входам разрядов DSR, CTS, RI, DCD регистра MSR соответственно. В режиме петли передаваемые данные немедленно принимаются. В режиме петли все прерывания формируются как обычно.
7:5	-	Резерв

Исходное состояние регистра MCR – нули.

10.2.8 Регистр MSR

Формат регистра MSR приведен в Таблица 10.10.

Таблица 10.10. Формат регистра MCR

Номер бита	Условное Обозначение	Назначение
0	DCTS	Признаки любого изменения состояния входного сигнала CTS. Бит устанавливается в единичное состояние, если сигнал CTS изменил свое состояние после последнего считывания содержимого регистра MSR. Одновременно с этим формируется прерывание MSI, если оно разрешено. Бит сбрасывается при чтении содержимого регистра MSR.
1	DDSR	Признаки любого изменения состояния входного сигнала DSR. Бит устанавливается в единичное состояние, если сигнал DSR изменил свое состояние после последнего считывания содержимого регистра MSR. Одновременно с этим формируется прерывание MSI, если оно разрешено. Бит сбрасывается при чтении содержимого регистра MSR.
2	TERI	Признаки перехода входного сигнала RI с низкого уровня на высокий уровень. Бит устанавливается в единичное состояние, если сигнал RI изменил свое состояние после последнего считывания содержимого регистра MSR. Одновременно с этим формируется прерывание MSI, если оно разрешено. Бит сбрасывается при чтении содержимого регистра MSR.
3	DDCD	Признаки любого изменения состояния входного сигнала nDCD. Бит устанавливается в единичное состояние, если сигнал nDCD изменил свое состояние после последнего считывания содержимого регистра MSR. Одновременно с этим формируется прерывание MSI, если оно разрешено. Бит сбрасывается при чтении содержимого регистра MSR.
4	CTS	Состояние сигнала на входе nCTS: 0 – на входе высокий уровень; 1 – на входе низкий уровень.
5	DSR	Состояние сигнала на входе nDSR: 0 – на входе высокий уровень; 1 – на входе низкий уровень.
6	RI	Состояние сигнала на входе nRI: 0 – на входе высокий уровень; 1 – на входе низкий уровень.
7	DCD	Состояние сигнала на входе nDCD: 0 – на входе высокий уровень; 1 – на входе низкий уровень.

Исходное состояние бит 3:0 регистра MSR – нули. Биты 7:4 следуют за инверсией состояния соответствующих входных сигналов.

10.2.9 Программируемый генератор скорости обмена

В UART имеется программируемый генератор скорости обмена данными (PBRG – Programmable Baud Rate Generator). Он состоит из 8-разрядного предделителя и 16-разрядного основного делителя частоты. На вход предделителя поступает системная тактовая частота CLK, на которой работает CPU, UART и другие устройства (см. рис. 4.1). Выходная частота предделителя поступает на вход основного делителя. Выходная частота генератора PBRG в 16 раз больше частоты обмена последовательными данными.

Коэффициент деления предделителя задается 8-разрядным регистром SCLR таким образом, чтобы частота на выходе предделителя соответствовала одной из трех стандартных частот (см. Таблица 10.11, Таблица 10.12, Таблица 10.13). Значение частоты на выходе предделителя равно CLK/(SCLR + 1). Коэффициент деления основного делителя задается 16-разрядным регистром, который является конкатенацией регистров DLM и DLL. Для получения одной из стандартных частот передачи значение этого коэффициента выбирается из Таблица 10.11, Таблица 10.12, Таблица 10.13.

Таблица 10.11 Скорости обмена и значения делителей для входной частоты 1,8432 МГц

Требуемая скорость обмена (бод)	**Делитель для получения частоты 16 * бод**	**Ошибка в процентах Разница между требуемой и действительной скоростью**
50	2304	-
75	1536	-
110	1047	0.026
134.5	857	0.058
150	768	-
300	384	-
600	192	-
1200	96	-
1800	64	-
2000	58	0.690
2400	48	-
3600	32	-
4800	24	-
7200	16	-
9600	12	-
19200	6	-
38400	3	-
56000	2	2.860

Таблица 10.12 Скорости обмена и значения делителей для входной частоты 3,072 МГц

Требуемая скорость обмена (бод)	Делитель для получения частоты 16 * бод	Ошибка в процентах Разница между требуемой и действительной скоростью
50	3840	-
75	2560	-
110	1745	0.026
134.5	1428	0.034
150	1280	-
300	640	-
600	320	-
1200	160	-
1800	107	0.312
2000	96	-
2400	80	-
3600	53	0.628
4800	40	-
7200	27	1.230
9600	20	-
19200	10	-
38400	5	-
56000	3	14.285

Таблица 10.13 Скорости обмена и значения делителей для входной частоты 8,0 МГц

Требуемая скорость обмена (бод)	Делитель для получения частоты 16 * бод	Ошибка в процентах Разница между требуемой и действительной скоростью
50	10000	-
75	6667	0.005
110	4545	0.010
134.5	3717	0.013
150	3333	0.010
300	1667	0.020
600	833	0.040
1200	417	0.080
1800	277	0.080
2000	250	-
2400	208	1.160
3600	139	0.080
4800	104	1.160
7200	69	0.644
9600	52	1.160
19200	26	1.160
38400	13	1.160
56000	9	0.790
128000	4	2.344
256000	2	2.344

Период частот передачи и приема (TCLK и RCLK) UART вычисляется по формуле:

CLK/(SCLR + 1) /((конкатенация содержимого регистров DLM и DLL) *16). Минимальная величина, которая может быть записана в регистры {DLM, DLL}, равна 1.

Исходное состояние регистров DLL, DLM, SCLR – нули.

10.3 Работа с FIFO по прерыванию

Если установлен режим работы с FIFO (EFWO=1 в регистре FCR) и разрешены прерывания по приему (бит ERI=1 в регистре IER), то в процессе приема:

- формируется прерывание, если число символов в RCVR FIFO достигло запрограммируемого порога. Это прерывание сбрасывается, если при чтении из FIFO число символов оставшихся в нем, станет меньше запрограммируемого порога;
- одновременно с этим в регистре IIR устанавливается индикатор наличия принятых данных RDAI. Индикатор обнуляется, при чтении из FIFO до снижения запрограммируемого порога;
- может возникнуть прерывание по статусу приема данных (RLSI), приоритет которого выше, чем RDA;
- бит RDR в регистре LSR устанавливается в момент передачи символа из регистра RSR в RCVR FIFO. Этот бит обнуляется при считывании из FIFO всех символов данных.

Если установлен режим работы с FIFO (EFWO=1 в регистре FCR) и разрешены прерывания по приему (ERI=1 в регистре IER), то генерируется прерывание по таймауту, если с момента приема последнего символа в RCVR FIFO прошло время, равное длительности передачи 4-х символов и за это время не было:

- ни чтения RCVR FIFO;
- ни приема в RCVR FIFO очередного символа.

При 12-битном символе и скорости передачи 300 бод, прерывание по этой причине возникнет через 160 мс.

При возникновении прерывания по таймауту оно обнуляется при считывании символа из RCVR FIFO. При этом обнуляется и таймер, генерирующий данное прерывание. Если прерывание по таймауту не возникло, то таймер таймаута обнуляется при приеме нового символа или при считывании символа из RCVR FIFO.

Если установлен режим работы с FIFO (EFWO=1 в регистре FCR) и разрешены прерывания по передаче данных (бит ETI=1 в регистре IER), то генерируется прерывание по передаче следующим образом:

- формируется прерывание THREI, если XMIT FIFO пусто. Это прерывание обнуляется, как только выполняется запись символа в регистр THR (при приеме данного прерывания в XMIT FIFO можно записать от 1 до 16 символов);
- индикатор TEMT в регистре LSR установится в единичное состояние через время равное длительности одного символа минус последний стоп бит, после установки THRE=1. Первое прерывание по передаче (если оно разрешено) формируется немедленно после установки FEWO=1.

10.4 Работа с FIFO по опросу

Если установлен режим работы с FIFO (EFWO=1 в регистре FCR) и запрещены прерывания, то обмен данными выполняется по опросу, а управление FIFO приема и передачи (RCVR, XMIT) выполняется раздельно.

В этом режиме опрос состояния RCVR и XMIT FIFO осуществляется программно, посредством считывания содержимого регистра LSR:

- бит RDR=1, пока есть данные в RCVR FIFO;
- биты OE, PE, FE, BI указывают на ошибки. Эти ошибки обрабатываются так же, как и при работе по прерыванию;
- бит THRE=1, если XMIT FIFO пусто;
- бит TEMT=1, если в XMIT FIFO и TSR нет данных.

При работе по опросу нет индикации таймаута и факта достижения порога RCVR FIFO. Однако оба RCVR и XMIT FIFO могут хранить символы данных.

11. ЛИНКОВЫЙ ПОРТ

11.1 Архитектура линкового порта

Линковый порт имеет следующие основные характеристики:

- частота передачи данных – CLK/4, CLK/2 (CLK – тактовая частота МСТ-01);
- использована двойная буферизация передаваемых и принимаемых данных;
- выполняет однословный обмен данными по прерываниям под управлением RISC-ядра;
- выполняет обмен блоками данных при помощи DMA;
- по внешнему интерфейсу линковый порт совместим с ADSP-21160.

Структурная схема линкового порта приведена на Рисунок 11.1.

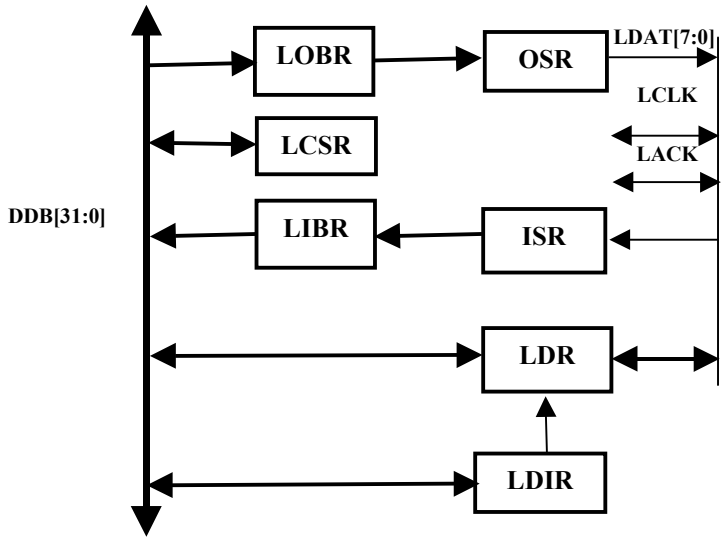


Рисунок 11.1. Структурная схема линкового порта

Передаваемые 32-разрядные данные записываются в выходной буферный регистр (OBR), а затем аппаратно переписываются в передающий сдвигающий регистр (OSR), если он пуст. После этого, в выходной буферный регистр могут быть записаны очередные данные. Из передающего сдвигающего регистра данные выдаются во внешнюю шину данных тетрадами или байтами.

Из внешней шины данные поступают в приемный сдвигающий регистр (ISR) тетрадами или байтами. После набора 32-разрядного слова он переписывается во входной буферный регистр (IBR).

Данные передаются, начиная со старшей тетрады или старшего байта.

Если LPORT неактивизирован (LEN=0), внешние линии LDAT[7:0], LCLK, LACK можно использовать как 10-разрядный двунаправленный порт ввода-вывода.

В Таблица 11.1. описаны внешние выводы линкового порта.

Таблица 11.1. Выводы линковых портов

Название вывода	Тип вывода	Описание
LDAT[3:0]/[7:0]	IO	Внешняя шина данных. Данные по этой шине передаются по положительному фронту сигнала LCLK.
LCLK	IO	Частота передачи данных
LACK	IO	Подтверждение приема

11.2 Регистры

11.2.1 Общие положения

Перечень регистров порта приведен в Таблица 11.2.

Таблица 11.2

Условное Обозначение регистра	**Название регистра**
LTx	Буфер передачи данных
LRx	Буфер приема данных
LCSR	Регистр управления и состояния
LDIR	Регистр управления направлением выводов порта ввода-вывода
LDR	Регистр данных порта ввода-вывода

11.2.2 Буфер передачи LTx

Буфер передачи LTx является буфером FIFO на два 32-разрядных слова и состоит из выходного буферного регистра и передающего сдвигающего регистра. Два 32-разрядных слова могут быть сразу записаны в буфер LTx, если он был до этого пуст.

Буфер передачи LTx генерирует прерывание (бит LportTx в регистре QSTR) при следующих условиях:

- бит LTRAN=1;
- выходной регистр данных пуст;
- соответствующий канал DMA не активизирован;
- данное прерывание не замаскировано.

Данное прерывание формируется в момент активизации линкового порта на передачу при пустом буфере LTx, или в момент переписи содержимого выходного регистра данных в выходной сдвигающий регистр. Прерывание, генерируемое буфером передачи, сигнализирует о том, что буфер LTx готов принять следующее слово. Прерывание от буфера передачи сбрасывается в момент записи в него данных.

Загрузка данных в порт возможна только при активизации порта на передачу.

11.2.3 Буфер приема LRx

Буфер приема LRx является буфером FIFO на два 32-разрядных слова и состоит из входного регистра данных и входного буферного регистра. Одно принятое 32-разрядное слово может храниться в буфере LRx, пока вдвигается второе слово.

В момент окончания приема в буфер LRx 32-разрядного слова данных, генерируется прерывание, если оно разрешено, а соответствующий канал DMA не активизирован. Данное прерывание сбрасывается при чтении данных из буфера приема.

Считывание данных из буфера приема возможно только при активизации порта на прием.

11.2.4 Регистр управления и состояния LCSR

Формат регистра LCSR приведен в Таблица 11.3.

Таблица 11.3. Формат регистра LCSR

Номер разряда	Условное обозначение	Назначение
0	LEN	Разрешение работы порта: 0 – все выводы порта находятся в высокоимпедансном состоянии; 1 – порт работает в соответствии с состоянием бита LTRAN.
1	LTRAN	Режим работы порта: 0 – приемник; 1 – передатчик.
2	LCLK	Управление частотой работы порта: 0 – CLK/4; 1 – CLK/2.
4:3	LSTAT	Состояние буферов Tx или Rx: 00 – буфер пуст; 10 – буфер содержит одно слово данных; 11 – буфер полон.
5	LRERR	Ошибка приема данных: 0 – приняты все биты данных; 1 – приняты не все биты данных.

Продолжение Таблица 11.3

Номер разряда	Условное обозначение	Назначение
6	LDW	Разрядность внешней шины данных: 0 - 4-разряда (32-разрядное слово передается за 8 посылок); 1 - 8-разряда (32-разрядное слово передается за 4 посылки).
7	SRQ_TX	Признак запроса обслуживания на передачу данных
8	SRQ_RX	Признак запроса обслуживания на прием данных
31:9	-	Резерв

Исходное состояние регистра LCSR – нули. Биты LEN, LTRAN, LCLK доступны по записи и чтению, а LSTAT, LRERR – только по чтению.

Биты LSTAT, LRERR сбрасываются при LEN=0.

11.2.5 Регистры порта ввода-вывода

10-разрядный регистр данных порта ввода-вывода (LDR) предназначен для реализации гибкого интерфейса с внешними устройствами. Внешние выводы порта ввода-вывода совмещены с внешними выводами линкового порта.

Соответствие разрядов регистра LDR и внешних линий линкового порта приведено в Таблица 11.4.

Таблица 11.4

Номер разряда Регистра LDR	Внешние выводы LPORT
0	**LACK**
1	**LCLK**
9:2	**LDAT[7:0]**

Настройка направления выводов порта ввода-вывода осуществляется программно при помощи 10-разрядного регистра LDIR. Если разряд этого регистра имеет нулевое состояние, то соответствующий разряд порта ввода-вывода является входом и наоборот. Линии порта ввода-вывода могут быть выходами, если LEN=0.

Исходное состояние регистров LDR, LDIR – нули.

11.3 DMA линковых портов

С каждым линковым портом связан канал DMA LportCh. Направление передачи DMA определяется битом LTRAN.

11.4 Прерывания от линковых портов

11.4.1 Прерывания при приеме и передаче данных

Линковый порт формирует прерывания по приему и передаче данных.

Если обмен данными по линковому порту выполняется программно без использования DMA, то прерывания формируются по завершению передачи или приема каждого 32-разрядного слова данных. При этом, биты RUN, DONE и END регистра CSR соответствующего канала DMA должны иметь нулевое состояние.

Если обмен данными по линковому порту выполняется с использованием DMA, то прерывания формируются в соответствии с п. 8.1.5.

11.4.2 Прерывания по запросу обслуживания

Если линковый порт не активизирован (LEN=0), он формирует прерывание по запросу обслуживания, если:

- на внешней шине выставлены данные на прием (активное состояние сигнала LCLK);
- из внешней шины поступил запрос на выдачу данных (активное состояние сигнала LACK).

Данное прерывание сбрасывается после установки LEN=1.

11.5 Временная диаграмма работы линкового порта

Временная диаграмма работы линкового порта приведена на Рисунок 11.2.

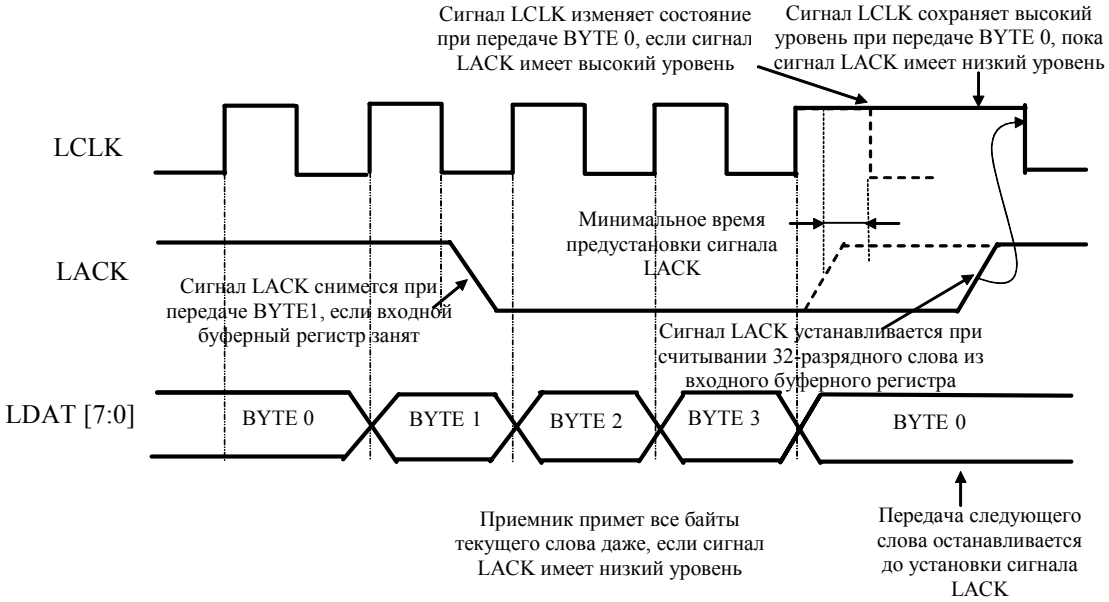


Рисунок 11.2. Временная диаграмма работы линкового порта (LDW=1)

При LDW=0 передача 32-разрядного слово выполняется за 8 посылок, а при LDW=1 - за 4 посылки. Передатчик изменяет данные LDAT по положительному фронту LCLK, а приемник защелкивает данные по отрицательному фронту.

Исходное состояние сигнала LACK – высокий уровень. Сигнал LACK снимется приемником по заднему фронту LCLK при передаче BYTE1, если его входной буферный регистр занят. При этом приемник примет все байты текущего 32-разрядного слова даже, если сигнал LACK имеет низкий уровень. Сигнал LACK устанавливается при считывании 32-разрядного слова из входного буферного регистра.

Передатчик после выставления BYTE0 анализирует состояние сигнала LACK. Если LACK=1, то LCLK продолжает изменять свое состояние и после BYTE 0 передается BYTE 1 и так далее. Если LACK=0, то LCLK сохраняет высокий уровень при передаче BYTE 0, пока сигнал LACK имеет низкий уровень.

Если линковый порт деактивизирован (LEN=0) сигналы LDAT, LCLK LACK являются входами. Поэтому эти сигналы необходимо привязывать к земле через резисторы 10 кОм. Если порт настроен как передатчик, LDAT и LCLK становятся выходами, а LACK – входом. Если порт настроен как приемник, LDAT и LCLK становятся входами, а LACK – выходом.

12. ПОРТ JTAG И ВСТРОЕННЫЕ СРЕДСТВА ОТЛАДКИ ПРОГРАММ

12.1 Введение

В МСТ-01 встроен тестовый порт JTAG, реализованный в соответствии со стандартом IEEE 1149.1 (IEEE Standard Test Access Port and Boundary-Scan Architecture). Данный порт предназначен для тестирования МСТ-01 в составе изделия в объеме, предусмотренном стандартом, а также для доступа к встроенным средствам отладки программ (далее модуль OnCD). В данном разделе используются условные обозначения и термины, определенные стандартом IEEE 1149.1.

Порт JTAG состоит из входного порта доступа (TAP), имеющего пять сигнальных выводов, TAP-контроллера управления на 16 состояний, интерпретирующего последовательно вводимую информацию синхронно с частотой TCK, регистра команд (IR) и обходного регистра Bypass.

Тестовая логика порта реализует следующие функции:

- выполнение обязательных команд, определенных стандартом IEEE 1149.1: EXTEST, BYPASS, SAMPLE/PRELOAD;
- перевод МСТ-01 в режим отладки (команда DEBUG_REQUEST);
- подключение к выводам TDI, TDO порта JTAG модуля OnCD (команда DEBUG_ENABLE).

Модуль OnCD обеспечивает:

- выполнение остановки программы CPU по контрольным точкам (Breakpoint);
- выполнение заданного числа команд CPU (трассы) в реальном масштабе времени или пошаговое выполнение команд;
- доступ к адресуемым регистрам и памяти МСТ-01.

Структурная схема порта JTAG и модуля OnCD приведена на Рисунок 12.1.

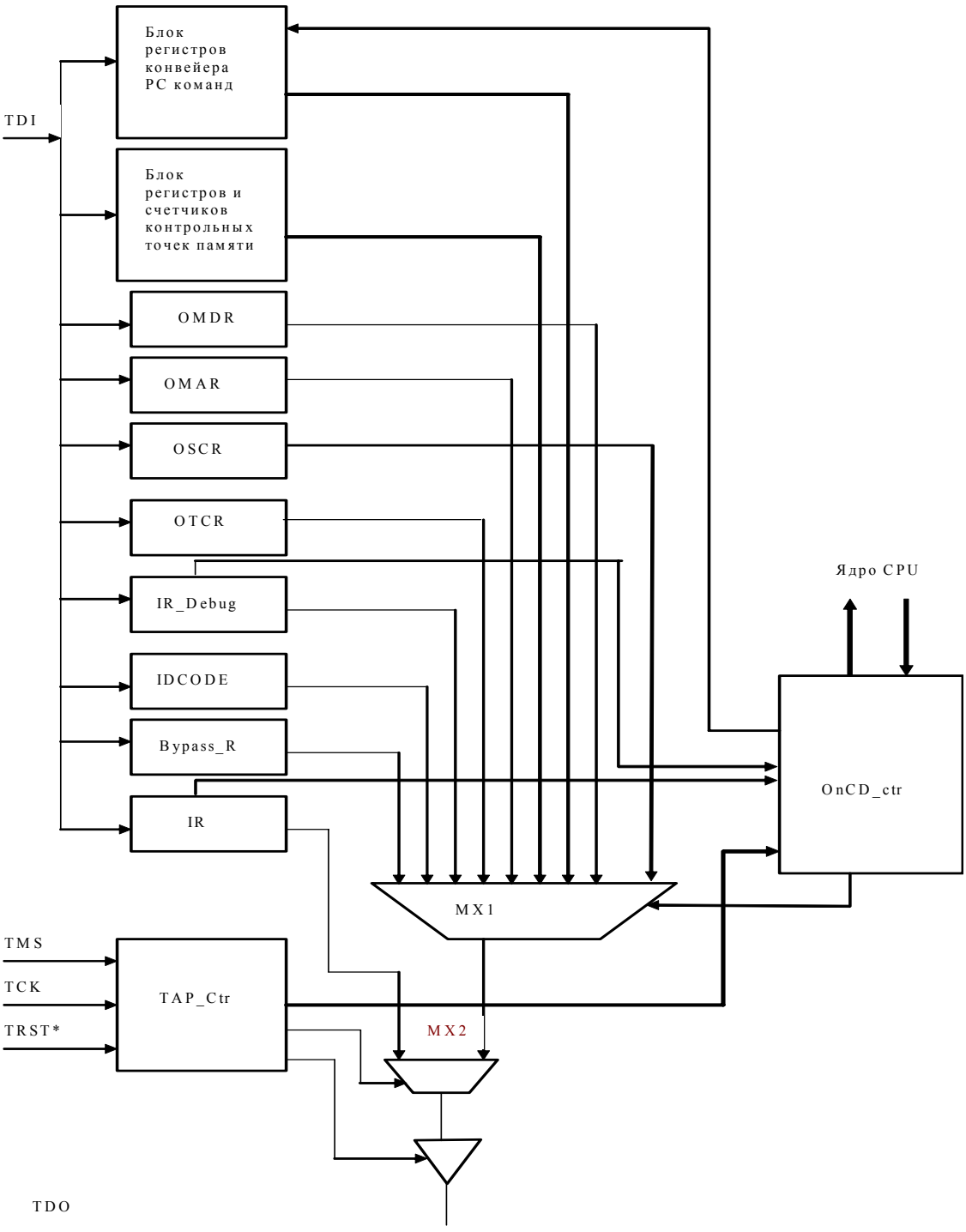


Рисунок 12.1. Порт JTAG и модуль OnCD

12.2 Порт JTAG

12.2.1 Внешние выводы порта

Порт JTAG имеет следующие внешние выводы:

- вход тестовой тактовой частоты – TCK;
- вход управления выборкой вида тестового воздействия – TMS;
- вход последовательного ввода тестовых команд и данных – TDI;
- вход инициализации контроллеров порта JTAG и OnCD – nTRST;
- выход последовательного вывода тестовых команд и данных – TDO.

Протокол обмена по порту JTAG подробно изложен в стандарте IEEE 1149.1 и здесь не приводится.

12.2.2 Контроллер порта (TAP)

TAP-контроллер порта является синхронным автоматом на 16 состояний, который распознает последовательный код по входу TMS и выполняет:

- ввод-вывод в регистр IR команд;
- ввод-вывод в один из регистров данных и операции над ними согласно команде JTAG, содержащейся в регистре IR.

12.2.3 Регистр команд (IR)

Регистр команд IR порта JTAG предназначен для хранения команд, относящихся к обязательным, согласно стандарту IEEE 1149.1, а так же команд, связанных с обслуживанием аппаратуры модуля OnCD. Обмен информацией с регистром IR выполняется посредством пути Select-IR-Scan TAP-контроллера порта JTAG.

Регистр команд имеет 4 разряда. Исходное состояние регистра IR по аппаратному сбросу – все единицы. Во время состояния Capture-IR в данный регистр по его параллельным входам загружается:

- в два младших разряда 1-0 – код 01;
- в два старших разряда 3-2 – биты состояния МСТ-01 OS[1:0].

Перечень команд порта JTAG МСТ-01 приведен в Таблица 12.1.

Таблица 12.1. Команды порта JTAG

Код IR[3:0]	Название команды
0000	EXTEST
0001	SAMPLE/PRELOAD
0010	Зарезервировано
0011	Зарезервировано
0100	DEBUG_REQUEST
0101	DEBUG_ENABLE
0110-1110	Зарезервировано
1111	BYPASS

Команды EXTEST, BYPASS и SAMPLE/PRELOAD выполняются в соответствии со стандартом IEEE 1149.1.

Команда DEBUG_REQUEST переводит МСТ-01 из рабочего состояния в состояние отладки. Выполнение данной команды приводит к тому, что все активные компоненты МСТ-01 (CPU, DMA, DSP) после выполнения текущей команды (операции) переходят в режим останова (stall), а адресуемые ресурсы - в распоряжение OnCD. Переход МСТ-01 в состояние отладки определяется циклической подачей команды DEBUG_ENABLE и сканированием состояния МСТ-01 на выводе TDO по битам OS[1:0]. Комбинация OS[1:0] = 11 сигнализирует о том, что МСТ-01 находится в состоянии отладки.

Команда DEBUG_ENABLE обеспечивает подключение к выводам TDI и TDO регистра команды IRd и регистров данных модуля OnCD. Выбор конкретного регистра данных определяется текущей командой модуля OnCD. Обмен данными с регистрами модуля OnCD (в том числе и с регистром команд) выполняется посредством пути Select-DR-Scan TAP-контроллера порта JTAG. Следует отметить, что команда OnCD и данные передаются за один цикл Select-DR-Scan: сначала через порт JTAG передается команда, а затем данные регистра, определяемого вводимой командой OnCD.

12.2.4 Регистр Bypass.

Регистр Bypass является одноразрядным сдвиговым регистром, с помощью которого обеспечивается кратчайший путь между выводами TDI и TDO при выполнении команды BYPASS.

12.3 Модуль встроенных средств отладки программ (OnCD)

Модуль OnCD позволяют взаимодействовать с аппаратурой МСТ-01 и иметь доступ к его адресуемым регистрам и памяти.

Модуль OnCD управляется через порт JTAG.

После загрузки команды DEBUG_ENABLE тестовые последовательности, поступающие по порту JTAG посредством пути Select-DR-Scan TAP-контроллера порта JTAG, интерпретируются модулем OnCD, который разделяет информацию на команды и данные, последними заполняются регистры, описанные ниже.

12.3.1 Регистры модуля OnCD

12.3.1.1 Регистр команд OnCD (IRd).

Доступ к регистру IRd через порт JTAG (подключение к выводам TDI и TDO) разрешается после выполнения команды DEBUG_ENABLE. Запись данных в регистр IRd выполняется посредством первой фазы пути Select-DR-Scan TAP-контроллера порта JTAG, а данные в регистр OnCD - в течение второй фазы. Таким образом, команда OnCD и данные передаются за один цикл Select-DR-

Scan: сначала порт JTAG передает команду (8 разрядов), а затем данные заданной разрядности (прием и передача).

Формат регистра IRd приведен в Таблица 12.2.

Таблица 12.2

Номер разряда	Условное обозначение	Описание
3-0	RS[3:0]	Определяет, какой регистр данных модуля OnCD является источником/приемником для команд запись/чтение. См. табл.13.3
4	EX	Указывает, что должен делать МСТ-01 после выполнения данной команды: 0 – оставаться в режиме отладки; 1 – выйти из режима отладки и возобновить нормальную работу. Данное действие выполняется, если GO=1 и RS=1111.
5	GO	Указывает, что должен исполнять МСТ-01 после выполнения данной команды: 0 – никаких действий не выполняется; 1 – выйти из режима отладки и выполнить одну команду, адрес которой в PC, если EX=0, и возобновить нормальную работу, если EX=1. Данные действия выполняются, если записывается RS=1111.
6	W_R	Определяет вид обращения к регистрам PC, Irdec, OTC, OSCR со стороны OnCD: если W_R=1, то указанные регистры доступны только по чтению, в противном случае – по чтению и записи, т.е. при чтении этих регистров обязательно производится их запись.
7	-	Резерв

Регистр IRd доступен по записи с чтением его предыдущего состояния. Исходное состояние по аппаратному сбросу – нули.

Перечень регистров данных модуля OnCD приведен в Таблице 12.3.

Таблица 12.3

Содержимое поля RS[3:0]	Регистр данных модуля OnCD
0000	Регистр управления и состояния OSCR
0001	Счетчик контрольных точек OMBC
0010	Регистр 0 границы адреса OMLR0
0011	Регистр 1 границы адреса OMLR1
0100	Регистр управления остановом при обращении к памяти или по PC OBCR
0101	Регистр команд CPU
0110	Счетчик трассы OTC
0111	Регистр адреса команды CPU, находящейся на стадии декодирования (DECODE)
1000	Регистр адреса команды CPU, находящейся на стадии выполнения (EXECUTE)
1001	Регистр адреса команды CPU, находящейся на стадии обмена с памятью (MEMORY)
1010	Программный счетчик PC
1011	Регистр адреса при обращении к памяти OMAR
1100	Регистр данных при обращении к памяти OMDR
1101	Команда непосредственного обмена данными с памятью МСТ-01 (псевдорегистр) –En_MEM
1110	Команда подтверждения выполнения операции с памятью – En_XX
1111	Команда выхода из состояния отладки En_GO

В Таблице 12.3. и далее, под программным счетчиком PC понимается программный счетчик CPU.

12.3.1.2 Регистр состояния и управления (OSCR)

Регистр OSCR служит для управления процессом отладки, а также для фиксации событий, возникающих в МСТ-01. Разряды 4-0 регистра OSCR доступны по записи и чтению, а 20-16 – только по чтению. Исходное состояние регистра OSCR – нули. Формат регистра приведен в Таблице 12.4.

Таблица 12.4

Номер разряда	Условное обозначение	Описание
0	SlctM	Разрешение проведения операции обращения к памяти CPU
1	RWm	Вид обращения к памяти: 0 - чтение; 1- запись
2	TME	Разрешение режима трассировки. 0 – запрещение режима трассировки; 1 – разрешение режима трассировки. В данном режиме МСТ-01 выполняет число команд, заданное в счетчике трассировки OMBC.
3	IME	Разрешает выдачу прерывания DI, если IME=1, которое формируется при выходе из режима отладки
4	MPE	Если установлен, то при выходе из режима отладки производится очистка конвейера CPU, в противном случае нет.
5	RDYm	Устанавливается в 1, если при обращении к памяти сигнал RDY = 1.
6	MBO	Устанавливается в 1, если выполнился останов по обращению в память или по PC. Он используется внешним контроллером команд, чтобы определить причину перехода МСТ-01 в режим отладки. Этот бит устанавливается в 0 при выходе из режима отладки.
7	TO	Устанавливается в 1, если переход в режим отладки был вызван уменьшением до нуля содержимого счетчика трассировки OTC и был разрешен режим трассировки. Этот бит устанавливается в 0 при выходе из режима отладки.
8	SWO	Признак программного перехода в режим отладки. Этот бит устанавливается в 0 при выходе из режима отладки.
10:9	OS[1:0]	Состояние МСТ-01: 00 – штатное выполнение команд; 01 – резерв; 10 – резерв; 11 – режим отладки.
16:11	-	Не используются. Считываются нули.

12.3.1.3 Счетчик трассы ОТС

Для реализации режима трассировки (выполнения заданного числа команд) имеется 16-разрядный счетчик ОТС, который позволяет перед возвращением в режим отладки выполнить более одной команды CPU. Это счетчик дает пользователю возможность выполнять в реальном времени до 2**16 команд без перехода в режим отладки.

Для того, чтобы включить режим трассировки, в счетчик ОТС загружается требуемое число (если необходимо выполнить N команд, то в ОТС необходимо загрузить число N-1), в PC CPU загружается адрес первой команды отрезка программы, который должен быть выполнен (или остается старое PC), в разряд TME регистра OSCR записывается 1, а затем МСТ-01 выводится из режима отладки, выполняя команду OnCD: Регистр IRd = {EX=1, GO=1, EnGO}.

После выхода из режима отладки счетчик ОТС уменьшается на 1 после выполнения каждой команды. При этом все исключения обрабатываются. Когда значение счетчика достигнет нуля, процессор после выполнения очередной команды снова перейдет в режим отладки, при этом в единичное состояние установятся разряды TO и OS[1:0] регистра OSCR, как индикация того, что МСТ-01 перешел в режим отладки и ожидает обслуживания.

Начальное значение счетчика ОТС по аппаратному сбросу равно нулю.

12.3.1.4 Логическая организация контрольных точек останова

Контрольные точки останова могут быть расставлены по адресам обращения к памяти и/или по содержимому PC.

Логическая схема контрольных точек содержит регистры для хранения двух контрольных адресов OMLR0 и OMLR1, компараторы и счетчик OMBC. Каждый из контрольных адресов может сравниваться на своем компараторе с адресом памяти или содержимым PC. Таким образом, имеется возможность отнести контрольные адреса только к памяти или только к PC или к тому и другому одновременно. Контрольные адреса могут быть организованы как границы заданного адресного пространства или как независимые адреса сравнения.

Останов МСТ-01 может осуществляется, когда адрес выбранной ячейки памяти находится в пределах, заданных содержимым регистров адресов нижней и верхней границ памяти, или равен одному из двух заданных адресов. Эту возможность можно отнести таким же образом только к PC или на равенство по одному адресу к PC и памяти одновременно.

16-разрядный счетчик контрольных точек OMBC загружается числом, которое на единицу меньше числа обращений к памяти/к PC, которое должно быть выполнено до останова МСТ-01. При каждом доступе к памяти/ к PC, соответствующей установленным контрольным точкам, счетчик контрольных точек уменьшается на 1. Когда счетчик достигает нулевого состояния и выполняется еще один доступ к контрольной точке, МСТ-01 переходит в режим отладки. Управление контрольными точками производится с помощью регистра управления OBCR.

Формат регистра OBCR приведен в Таблице 12.5.

Таблица 12.5

Номер разряда	Условное обозначение	Описание
0-1	MBS[1:0]	Управление коммутацией адреса на входы компараторов. MBS[1]: 0-на вход компаратора 1 подается адрес PC, 1-на вход компаратора 1 подается адрес памяти. MBS[0]: 0-на вход компаратора 0 подается адрес PC, 1-на вход компаратора 0 подается адрес памяти.
2-3	RW0[1:0]	RW0 устанавливает вид обращения для контроля 00-точка останова 0 запрещена, 01-точка останова 0 при доступе по записи, 10 -точка останова 0 при доступе по чтению, 11 -точка останова 0 при доступе по чтению или по записи
4-5	CC0[1:0]	CC0 устанавливает условия сравнения между текущим адресом обращения и содержимым регистра OMLR0: 00-останов по не равно, 01 -останов по равенству, 10-останов, если меньше, 11- останов, если больше.
6-7	RW1[1:0]	RW1 устанавливает вид обращения для контроля 00-точка останова 1 запрещена, 01-точка останова 1 при доступе по записи, 10-точка останова 1 при доступе по чтению, 11-точка останова 1 при доступе по чтению или по записи
8-9	CC1[1:0]	CC1 устанавливает условия сравнения между текущим адресом обращения и содержимым регистра OMLR1: 00-останов по не равно, 01-останов по равенству, 10-останов, если меньше, 11- останов, если больше.
10	BT	Определяет контроль последовательности возникновения точек останова: 0 -счетчик декрементируется, если произошло сравнения для обеих точек останова. 1 - счетчик декрементируется, если произошло сравнения для любой одной или обеих точек останова

12.3.1.5 Информация о конвейере CPU

При помощи модуля OnCD обеспечивается доступ к PC и трем 32-разрядным регистрам, содержащим адреса трех предыдущих команд, выполняемых CPU: команд, находящихся на стадиях декодирования, исполнения и обмена с памятью.

Кроме того, для модуля OnCD доступен 32-разрядный регистр команды CPU IRdec, в котором запоминается последняя выбранная команда перед переходом МСТ-01 в режим отладки. Регистр доступен по чтению и записи. Этот регистр может изменяться операциями, выполняемыми в режиме отладки, и может быть восстановлен контроллером команд порта JTAG при возврате МСТ-01 в нормальный режим работы.

12.3.1.6 Обмен данными с памятью МСТ-01

Модуль OnCD обеспечивает возможность непосредственного обмена данными с адресуемыми регистрами и памятью МСТ-01 без участия CPU. Для этого имеются 32-разрядные регистры OMAR и OMDR.

Запись данных в память МСТ-01 выполняется следующим образом:

- в регистр OMAR записывается адрес памяти;
- в регистр OMDR записываются данные;
- выполняется команда непосредственного обмена данными с памятью МСТ-01, по которой содержимое регистра OMDR переписывается в память МСТ-01 по адресу, который содержится в регистре OMAR.

Чтение данных из памяти МСТ-01 выполняется следующим образом:

- в регистр OMAR записывается адрес памяти;
- выполняется команда непосредственного обмена данными с памятью МСТ-01, по которой в регистр OMDR записывается содержимое памяти;
- производится чтение содержимого регистра OMDR.

12.3.2 Способы перевода МСТ-01 в режим отладки

12.3.2.1 Внешнее требование входа в режим отладки при действии сигнала nRST

При инициализации МСТ-01 сигналом nRST можно войти в режим отладки, выполнив команду DEBUG_REQUEST**,** не снимая сигнал nRST. После обнаружения внешним контроллером JTAG факта вхождения МСТ-01 в режим отладки сигнал nRST должен быть снят и может быть начат процесс отладки.

12.3.2.2 Внешнее требование входа в режим отладки во время нормальной работы МСТ-01

Во время нормальной работы МСТ-01 войти в режим отладки можно, выполнив команду DEBUG_REQUEST.

12.3.2.3 Программный вход в режим отладки

Переход в режим отладки программным способом МСТ-01 производится посредством выполнения команды BREAK с полем code=1.

12.3.2.4 Переход в режим отладки после выполнения заданной трассы программы

Если выбран режим трассировки и содержимое счетчика трассировки ОТС не равно 0, то последний уменьшается на 1 при выполнении каждой команды CPU. МСТ-01 переходит в режим отладки после выполнения очередной команды, если на момент ее начала ОТС был равен нулю.

12.3.2.5 Вход в режим отладки после останова по контрольной точке в памяти и /или по PC

Если установлен режим останова по контрольной точке в памяти и/или по PC, то при достижении счетчиком ОМВС нулевого состояния МСТ-01 перейдет в режим отладки, ожидая выполнения команд от внешнего контроллера порта JTAG.

13. ЭЛЕКТРИЧЕСКИЕ И ВРЕМЕННЫЕ ПАРАМЕТРЫ

13.1 Электропитание

Номинальное значение напряжения электропитания микросхемы:

- U_{CC1}=3,3 В (периферия);
- U_{CC2}=2,5 В (ядро).

Допустимые отклонения напряжения электропитания микросхемы от номинального значения - не более ±5%.

Напряжения электропитания U_{CC1} и U_{CC2} необходимо подавать и снимать одновременно с разбросом не более 1 мс. Фронт нарастания напряжений электропитания должен быть не более 1 мс.

Для фильтрации напряжений электропитания микросхемы, необходимо подключить к каждому источнику (U_{CC1} и U_{CC2}) не менее шести высокочастотных конденсаторов номиналом 0,1 мкФ типа CC 0603 Y5V 0,1 uF Z 25V. Конденсаторы необходимо разместить по возможности равномерно по периметру корпуса микросхемы между выводами PVDD и GND, а так же CVDD и GND. При этом расстояние между контактами микросхемы и площадками подсоединения конденсаторов должно быть не более 3 мм.

13.2 Электрические параметры

Электрические параметры микросхемы приведены в Таблица 13.1.

Таблица 13.1. Электрические параметры микросхемы

Наименование параметров, единица измерения, режим измерения	Буквенное обозначение	Норма		Температура °C
		не менее	не более	
Ток потребления статический по цепи PVDD, мА при U_{CC1}=3,47В, U_{CC2}=2,63В, XTI=0	I_{CC1}	-	3	от -60 до +85
Ток потребления статический по цепи CVDD, мА при U_{CC1}=3,47В, U_{CC2}=2,63В, XTI=0	I_{CC2}	-	10	от -60 до +85
Ток потребления динамический по цепи CVDD, мА, при U_{CC1}=3,47В, U_{CC2}=2,63В и рабочей частоте 80 МГц	I_{OCC2}	-	300	от -60 до +85
Ток утечки высокого и низкого уровня на входе, мкА при U_{CC1}=3,47В и U_{CC2}=2,63В	I_{IL}	-	2	от -60 до +85
Выходное напряжение низкого уровня, В при I_{OL}= 4 мА, U_{CC1}=3,47В	U_{OL}	-	0,4	от -60 до +85
Выходное напряжение высокого уровня, В при I_{OH} =-2,8 мА, U_{CC1} =3,13В	U_{OH}	2,4		от -60 до +85
Входная емкость, пФ	C_I	-	10	25 ± 10
Емкость входа/выхода, пФ	$C_{I/O}$	-	10	25 ± 10
Выходная емкость, пФ	C_O	-	15	25 ± 10

13.3 Динамическая потребляемая мощность

Динамическая потребляемая мощность микросхемы имеет две составляющие: потребление ядра (по цепи CVDD) и потребление выходных драйверов (по цепи PVDD).

Мощность, потребляемая ядром микросхемы по цепи CVDD, зависит от последовательности выполняемых процессорными ядрами команд, от операндов, а также от активности DMA и периферийных устройств. Максимальный ток, потребляемый ядром микросхемы, не превышает 300 мА при внутренней частоте синхронизации 80 МГц.

Мощность, потребляемая выходными драйверами по цепи PVDD, зависит от следующих параметров:

- Число выходных драйверов (O);

- Максимальная частота, на которой выходные драйверы переключаются (F);
- Емкости нагрузки выходных драйверов (C);
- Величина напряжения электропитания выходных драйверов (U_{CC1}).

Мощность, потребляемая выходными драйверами по цепи PVDD, определяется следующим уравнением:

$$Pext = O*C* U_{CC1}^2*F.$$

Рассмотрим для примера расчет мощности, потребляемой выходными драйверами при непрерывной записи данных в память типа SRAM (при U_{CC1}. = 3,3 В). Максимальная частота обмена данными со SRAM = CLK/4, где CLK – внутренняя тактовая частота микросхемы (80 МГц). При обращении по произвольным адресам можно предположить, что с частотой CLK/4 изменяются 50% разрядов адреса. Также можно допустить, что каждый цикл изменяются 50% разрядов шины данных. Данные для расчета потребляемой мощности приведены в Таблица 9.9

Таблица 13.2

Название драйвера	Число драйверов	Емкость нагрузки	F, МГц	U_{CC1}^2	Pext, мВт
A[31:0]	16	30	20	10,9	100
nWR[3:0]	4	30	20	10,9	25
D[31:0]	16	30	20	10,9	100
SCLK	1	30	80	10,9	25
Итого:					250

То есть, при тактовой частоте 80 МГц и C=30 пФ при непрерывной записи данных в SRAM потребление составляет 250 мВт. При чтении данных из SRAM выходные драйверы не активизируются. Поэтому, если запись данных в SRAM чередуется с чтением, то реальное энергопотребление микросхемы будет существенно меньше.

Оценим мощность, потребляемую драйверами линкового порта при передаче данных. Максимальная частота передачи данных по линковому порту равна 40 МГц. Потребление по LCLK составляет 12 мВт, а потребление по данным (изменяется 50% 8-разрядных данных с частотой 20 МГц) - 24 мВт. Суммарно – 36 мВт.

13.4 Предельно-допустимые и предельные электрические режимы эксплуатации

Значения предельно-допустимых и предельных электрических режимов эксплуатации микросхемы приведены в Таблица 13.3.

Таблица 13.3. Значения предельно-допустимых и предельных электрических режимов эксплуатации

Наименование параметра, единица измерения	Буквенное обозначение	Норма			
		Предельно допустимый режим		Предельный Режим	
		не менее	не более	не менее	не более
Напряжение питания периферии, В	U_{CC1}	3,13	3,47	-	3,9
Напряжение питания ядра, В	U_{CC2}	2,37	2,63	-	3,0
Входное напряжение высокого уровня, В	U_{IH}	2,0	U_{CC1}+0,2	-	U_{CC1}+0,3
Входное напряжение низкого уровня, В	U_{IL}	0,0	0,7	-0,3	-
Напряжение, прикладываемое к выходу микросхемы в состоянии «выключено», В	U_{OZ}	0,0	U_{CC1}+0,1	-0,3	U_{CC1}+0,3
Емкость нагрузки каждого выхода, пФ	C_L	-	30	-	50

13.5 Временные параметры

13.5.1 Обмен данными с внешней памятью и устройствами

Временные параметры при обмене данными с внешней памятью и устройствами приведены в Таблица 13.4.

Таблица 13.4. Временные параметры при обмене данными с внешней памятью и устройствами

Наименование параметра, единица измерения	Буквенное обозначение	Норма		Температура °C
		не менее	не более	
Время задержки выходных сигналов A, D, nWR, nWE, nRD, nCS, SRAS, SCAS, SWE, DQM, CKE, A10, BA, nFLYBY, nOE после переднего фронта частоты SCLK, нс	t_{DOSC}	2	5	от -60 до +85
Время предустановки считываемых данных из асинхронной памяти перед задним фронтом частоты SCLK, нс	t_{SDSC}	6	-	от -60 до +85
Время удержания считываемых данных из асинхронной памяти после фронта снятия сигнала nRD, нс (t_{CLK} – период частоты CLK)	t_{HDRD}	0	0,5 t_{CLK}	от -60 до +85
Время предустановки считываемых данных из синхронной памяти перед передним фронтом частоты SCLK, нс	t_{SDSC}	5	-	от -60 до +85
Время удержания считываемых данных из синхронной памяти после переднего фронта частоты SCLK, нс	t_{HDSC}	0	0,5t_{CLK}	от -60 до +85

Временная диаграмма при чтении данных из асинхронной памяти приведена на Рисунок 9.4. Считываемые данные фиксируются в микросхеме по заднему фронту частоты SCLK перед снятием сигнала nRD.

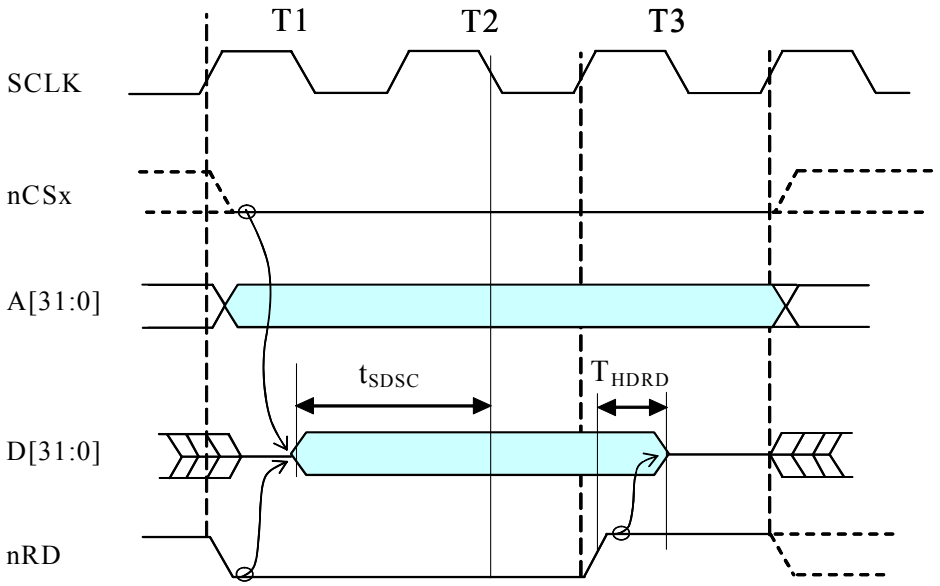


Рисунок 13.1. Чтение асинхронной памяти без дополнительных тактов ожидания.

13.5.2 Прием и передача данных по линковому порту

Временные параметры при приеме данных по линковому порту приведены в Таблица 13.5 и Рисунок 13.2.

Таблица 13.5. Временные параметры при приеме данных по линковому порту

Наименование параметра, единица измерения	Буквенное обозначение	Норма		Температура °C
		не менее	не более	
Время предустановки данных перед задним фронтом частоты LCLK, нс	t_{SLDCL}	5	-	от -60 до +85
Время удержания данных после заднего фронта частоты LCLK, нс	t_{HLDCL}	3	-	от -60 до +85
Время задержки переключения сигнала LACK с высокого на низкий уровень после заднего фронта частоты LCLK, нс	t_{DLALC}	5	15	от -60 до +85
Период частоты LCLK	t_{LCLK}	2,05 * t_{CLK}	-	от -60 до +85

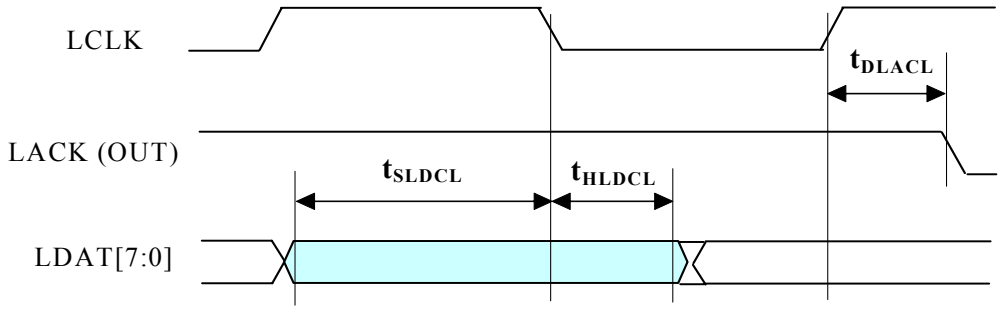


Рисунок 13.2. Прием данных по линковому порту

Временные параметры при передаче данных по линковому порту приведены в Таблица 13.6 и Рисунок 13.3.

Таблица 13.6. Временные параметры при передаче данных по линковому порту

Наименование параметра, единица измерения	Буквенное обозначение	Норма		Температура °C
		не менее	не более	
Время задержки данных после переднего фронта частоты LCLK, нс	t_{DLDCH}	-	10	от -60 до +85
Время удержания данных после переднего фронта частоты LCLK, нс	t_{HLDCH}	0	-	от -60 до +85
Время задержки переключения частоты LCLK в низкий уровень, после переключения сигнала LACK с низкого уровня на высокий, нс	t_{DLACLK}	5	t_{CLK} + 5	от -60 до +85

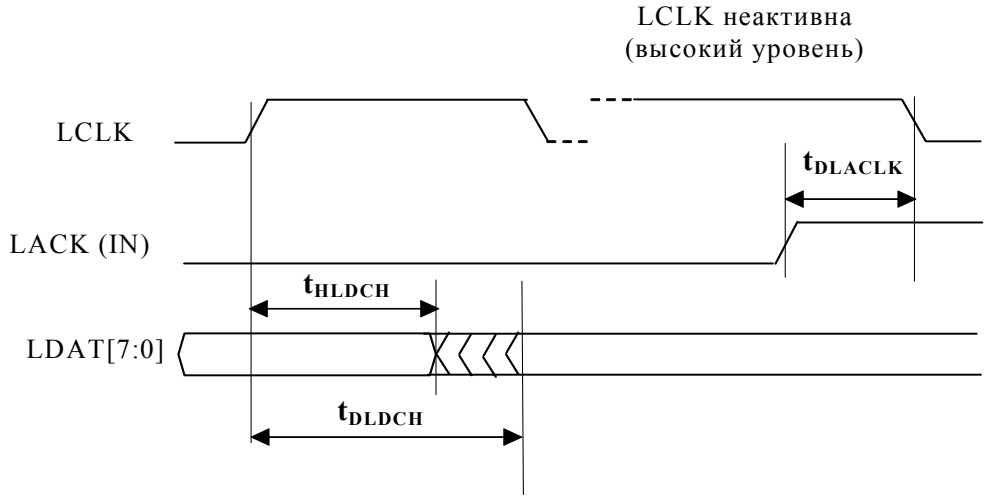


Рисунок 13.3. Передача данных по линковому порту

13.6 Рекомендации по подключению кварцевого резонатора.

Схема подключения кварцевого резонатора к микросхеме приведена на **Рисунок 13.4**.

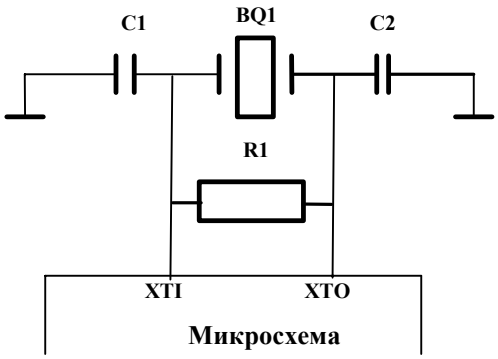


Рисунок 13.4. Схема подключения кварцевого резонатора к микросхеме

Частота кварцевого резонатора должна быть от 10 до 20 МГц. Ориентировочные величины: R1=1 мОм, C1=C2=7 пФ. Конкретная величина конденсаторов и резистора указывается в документации на резонатор.

14. ОПИСАНИЕ ВНЕШНИХ ВЫВОДОВ

Микросхема МСТ-01 имеет следующие выводы:

- порт внешней памяти – 100;
- управление – 22;
- 2 порта SpaceWire – 16;
- 2 линковых порта – 20;
- UART – 2;
- электропитание – 37.

Описание выводов приведено в табл. 1-6.

Таблица 14.1. Порт внешней памяти

Название Вывода	Количество	Тип	Назначение
A[29:0]	32	O	Шина адреса.
D[31:0]	32	IO	Шина данных
nWR[3:0]	4	O	Запись байтов асинхронной памяти
nWE	1	O	Запись асинхронной памяти
nRD	1	O	Чтение асинхронной памяти
nWRS[3:0]	4	O	Запись синхронной памяти
nRDS	1	O	Чтение синхронной памяти
nACK	1	I	Готовность асинхронной памяти
nCS[4:0]	5	O	Разрешение выборки банков памяти
SRAS	1	O	Строб адреса строки
SCAS	1	O	Строб адреса колонки
SWE	1	O	Разрешение записи
DQM[3:0]	4	O	Маска выборки байта
SCLK	1	O	Тактовая частота работы
CKE	1	O	Разрешение частоты
A10	1	O	10 разряд адреса
BA[1:0]	2	O	Номер банка
nFLYBY[3:0]	4	O	Признак режима передачи DMA “Flyby”
nOE[3:0]	4	O	Разрешение чтения внешнего устройства
nOES	1	O	Технологический. Должен быть свободным.
Всего 100 вывод			

Таблица 14.2. Управление

Название вывода	Количество	Тип	Назначение
nDMAR[3:0]	4	I	Запрос канала DMA
NMI	1	I	Немаскируемое прерывание
nIRQ[3:0]	4	I	Запросы прерывания
BYTE	1	I	Разрядность шины данных 6 банка внешней памяти: 0 – 32 разряда; 1 – 8 разрядов.
PLL_EN	1	I	Разрешение работы PLL: 0 – системная тактовая частота микроконтроллера равна входной частоте XTI (см. рис. 4.1); 1 - системная тактовая частота микроконтроллера поступает из PLL и равна входной частоте XTI, умноженной на коэффициент умножения/деления. (поле CLK_SEL регистра CSR).
XTI, XTO	2	I	Выводы для подключения внешнего кварцевого резонатора частотой от 10 до 20 МГц. На вывод XTI можно подать частоту от внешнего генератора, при этом вывод XTO должен быть незадействованным.
RTC_XTI	1	I	Сигналы частоты реального времени
XTI10	1	I	Сигнал тактовой частоты 10 МГц.
WDT	1	O	Признак срабатывания сторожевого таймера. Этот сигнал формируется, если в программе произошел сбой. Его можно подать на системный контроллер, который будет принимать решение, что делать в данной ситуации.
nRST	1	I	Сигнал установки исходного состояния
TCK	1	I	Тестовый тактовый сигнал (JTAG)
TRST	1	I	Установка исходного состояния (JTAG)
TMS	1	I	Выбор режима теста (JTAG)
TDI	1	I	Вход данных теста (JTAG)
TDO	1	O	Выход данных теста (JTAG)
Всего 22 вывода			

Таблица 14.3. Линковые порты (2 штуки)

Наименование Сигнала	Количество	Тип	Назначение
LDAT	8	IO	Шина данных.
LCLK	1	IO	Синхронизация
LACK	1	IO	Подтверждение
Всего 10*2=20 выводов			

Таблица 14.4. UART

Наименование сигнала	Количество	Тип	Назначение
SIN	1	I	Вход последовательных данных
SOUT	1	O	Выход последовательных данных
Всего 2 вывода			

Таблица 14.5. Порты SpaceWire

Название вывода	Количество	Тип	Назначение
DINp0, DINp1	2	I	Вход данных положительный
DINn0, DINn1	2	I	Вход данных отрицательный
SINp0, SINp1	2	I	Вход строба положительный
SINn0, SINn1	2	I	Вход строба отрицательный
DOUTp0, DOUTp1	2	O	Выход данных положительный
DOUTn0, DOUTn1	2	O	Выход данных отрицательный
SOUTp0, SOUTp1	2	O	Выход строба положительный
SOUTn0, SOUTn1	2	O	Выход строба отрицательный
Всего 16 выводов			

Таблица 14.6. Электропитание

Название вывода	Количество	Назначение
CVDD	9	Напряжение электропитания ядра и PLL
PVDD	10	Напряжение электропитания входных и выходных драйверов
GND	18	Земля ядра, PLL, входных и выходных драйверов
Всего 37 выводов		

Нумерация выводов микросхемы МСТ-01 в корпусе QFP-240 приведена в таблице 15.7. Символом NC обозначены выводы, которые при использовании должны быть незадействованными.

Таблица 14.7. Нумерация выводов МСТ-01 в корпусе QFP-240

№ вывода корпуса	Тип вывода	Условное обозначение	№ вывода корпуса	Тип вывода	Условное обозначение
1	O	A_10	61	O	nFLYBY[0]
2	I	TDI	62	O	nFLYBY[1]
3	I	TMS	63	O	nFLYBY[2]
4	I	TRST	64	O	nFLYBY[3]
5		GND	65	O	nCS[3]
6		PVDD	66	O	DQM[0]
7	I	nIRQ[3]	67	O	DQM[1]
8	I	nIRQ[2]	68	O	DQM[2]
9	I	nIRQ[1]	69		CVDD
10	I	nIRQ[0]	70		GND
11	O	WDT	71	O	SOUT
12		NC	72	I	XTI10
13		NC	73	I	RTCXTI
14		NC	74		PVDD
15		NC	75		GND
16	I	NMI	76	O	SW0_SOUTp
17	O	TDO	77	O	SW0_SOUTn
18	I	TCK	78	O	SW0_DOUTn
19	O	nRD	79	O	SW0_DOUTp
20	O	CKE	80	I	SW0_SINp
21	O	A[0]	81	I	SW0_SINn
22	O	A[1]	82	I	SW0_DINp
23	O	A[2]	83	I	SW0_DINn
24	O	A[3]	84	O	SW1_SOUTp
25	O	A[4]	85	O	SW1_SOUTn
26	O	A[5]	86	O	SW1_DOUTn
27		GND	87	O	SW1_DOUTp
28		CVDD	88	I	SW1_SINp
29	O	A[6]	89	I	SW1_DINn
30	O	A[7]	90	I	SW1_SINn
31	O	A[8]	91	I	SW1_DINp
32	O	A[9]	92		GND
33	O	A[10]	93		CVDD
34	O	A[11]	94	I	PLL_EN
35	O	A[12]	95	I	XTI
36	O	A[13]	96	O	XTO
37	O	A[14]	97	O	SCLK
38	O	A[15]	98	I	nRST
39	O	A[16]	99	O	nSC[0]
40	O	A[17]	100	O	nSC[1]
41	O	A[18]	101	O	nSC[2]
42	O	A[19]	102	IO	D[0]
43	O	A[20]	103	IO	D[1]
44	O	A[21]	104	IO	D[2]
45	O	A[22]	105	IO	D[3]
46	O	A[23]	106		CVDD
47	O	A[24]	107		GND
48	O	A[25]	108	IO	D[4]
49	O	A[26]	109	IO	D[5]
50	O	A[27]	110	IO	D[6]
51		GND	111	IO	D[7]
52		PVDD	112		CVDD
53	O	A[28]	113		GND
54	O	A[29]	114		PVDD
55	O	nWR[3]	115		GND
56	O	nWR[2]	116	IO	D[8]
57	O	nWR[1]	117	IO	D[9]
58	O	nWR[0]	118	IO	D[10]
59	O	DQM[3]	119	IO	D[11]
60	O	SWE	120	IO	D[12]

Продолжение Таблица 14.7

№ вывода корпуса	Тип вывода	Условное обозначение	№ вывода корпуса	Тип вывода	Условное обозначение
121	IO	D[13]	181	IO	LDAT1[4]
122	IO	D[14]	182	IO	LDAT1[3]
123	IO	D[15]	183	IO	LDAT1[2]
124	IO	D[16]	184	IO	LDAT1[1]
125	IO	D[17]	185	IO	LDAT1[0]
126	IO	D[18]	186	IO	LACK1
127	IO	D[19]	187	IO	LACK0
128	IO	D[20]	188		GND
129	IO	D[21]	189		PVDD
130	IO	D[22]	190		NC
131	IO	D[23]	191		NC
132	IO	D[24]	192		NC
133	IO	D[25]	193		NC
134	IO	D[26]	194		NC
135	IO	D[27]	195		NC
136	IO	D[28]	196		NC
137	IO	D[29]	197		NC
138	IO	D[30]	198		GND
139	IO	D[31]	199		CVDD
140	O	nRDS	200		NC
141	I	BYTE	201		NC
142	O	nWE	202		NC
143	I	nACK	203		NC
144	O	SCAS	204		NC
145	O	SRAS	205		NC
146	O	BA[0]	206		NC
147	O	BA[1]	207		NC
148	O	nWRS[0]	208		NC
149	O	nWRS[1]	209		NC
150	O	nWRS[2]	210		CVDD
151	O	nWRS[3]	211		GND
152	O	nOE[0]	212		NC
153	O	nOE[1]	213		NC
154	O	nOE[2]	214		NC
155		CVDD	215		NC
156		GND	216		PVDD
157		PVDD	217		PVDD
158		GND	218		GND
159	O	nOE[3]	219		NC
160		NC	220		NC
161	IO	LCLK0	221		NC
162	IO	LCLK1	222		NC
163	IO	LDAT0[0]	223		NC
164	IO	LDAT0[1]	224		NC
165	IO	LDAT0[2]	225		NC
166	IO	LDAT0[3]	226		NC
167	IO	LDAT0[4]	227		GND
168	IO	LDAT0[5]	228		CVDD
169	IO	LDAT0[6]	229		NC
170	IO	LDAT0[7]	230		NC
171	I	nDMAR[0]	231		NC
172	I	nDMAR[1]	232		NC
173	I	nDMAR[2]	233		NC
174	I	nDMAR[3]	234		NC
175	IO	LDAT1[7]	235		NC
176	IO	LDAT1[6]	236		NC
177	IO	LDAT1[5]	237		NC
178	I	SIN	238		NC
179		PVDD	239		GND
180		GND	240		PVDD

Чертеж корпуса QFP-240 приведен на Рисунок 14.1

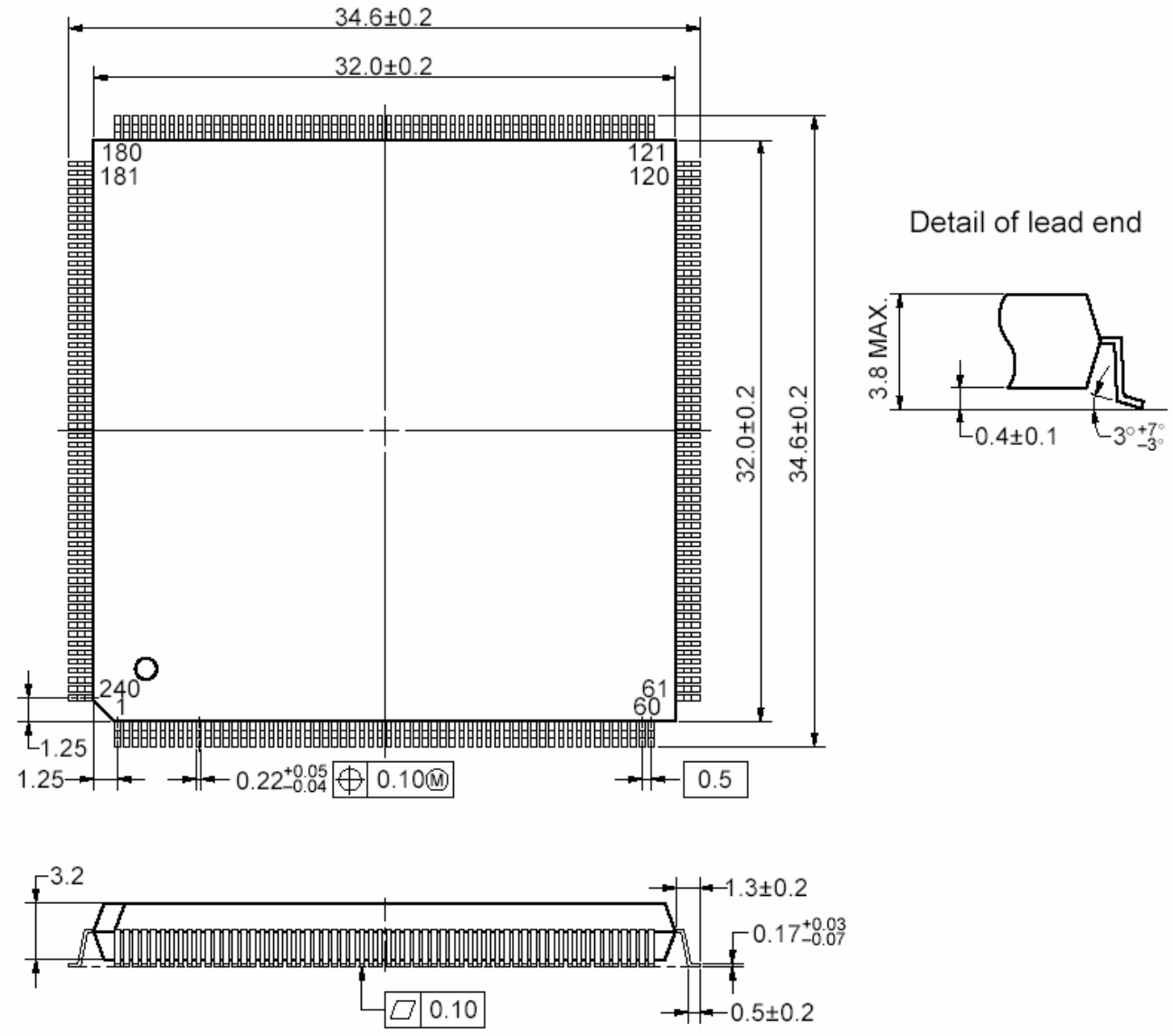


Рисунок 14.1. Чертеж корпуса QFP-240

15. ИСТОРИЯ ИЗМЕНЕНИЙ