Спецификация локальной шины PCI

Реализация 2.0
30 Апреля, 1993

Реализация	Хронология реализации	Дата
1.0	Оригинальное издание	6/22/92
2.0	Спецификации соединительного разъема и платы расширения	4/30/93

Специальная группа интересов PCI не несет ответственности за использование этого документа и информации, содержащейся в нем, а также за любые ошибки, которые могут возникнуть в этом документе, Специальная группа интересов PCI не берет на себя обязательств по модифицированию приведенной здесь информации.

Для получения последней версии спецификации обращайтесь в Специальную группу интересов PCI.

По всем вопросам и пожеланиям относительно спецификации PCI, либо по поводу вхождения в состав Специальной группы интересов PCI, обращаться:

PCI Special Interest Group
M/S HF3-15A
5200 N.E. Elam Young Parkway
Hillsboro, Oregon 97124-6497
(503)696-2000

Содержание

Глава 1 Введение

Глава 2 Описание сигналов

Глава 3 Функционирование шины

Глава 4 Электрическая спецификация

Глава 5 Конструктивная спецификация

Глава 6 Пространство конфигурации

Приложение A Сообщения специального цикла

Приложение B Конечные автоматы

Приложение C Правила эксплуатации

Глоссарий

Предисловие

Более ранние документы, заменяемые данной спецификацией

Этот документ содержит формальные спецификации протокола, а также электрических и механических особенностей локальной шины PCI (реализация 2.0) в качестве промышленной версии, действующей с 30 апреля 1993 года. Данная спецификация содержит более новые сведения, чем следующие документы:

- *Peripheral Component Interconnect (Спецификация соединений периферийных компонентов)*, реализация 1.0, вступила в силу 22 июня 1992 года.
- *Technical Addendum to PCI Specification (Техническое приложение к спецификации PCI)*, реализация 1.0, действует с сентября 1992 года.
- *Peripheral Component Interconnect, Add-in Board/Connector (ABC) Addendum (Draft) (Взаимосвязи периферийных компонентов, встраиваемые/подсоединяемые приложения -ABC)*. Проект вступил в силу 10 ноября 1992 года.
- *Peripheral Component Interconnect, Add-in Board/Connector (ABC) Addendum (Final Version) (Взаимосвязи периферийных компонентов, встраиваемые/подсоединяемые приложения)*, действует с марта 1992 года.
- *PCI Electrical Definition (Final Version) (Электрическое описание сигналов PCI)*, действует с 19 апреля 1993 года.

Кроме того, следующие документы не отражают текущие требования для данной *Спецификации локальной шины PCI*, реализация 2.0, и будут пересмотрены специальной группой интересов PCI:

- *PCI BIOS Specification (Спецификация PCI BIOS)*, реализация 2.0, введенная в действие с 28 сентября 1992 года фирмой Intel Corporation
- Предварительное руководство по системному проектированию PCI, реализация 0.6, действует с 1 ноября 1992 года.

Объединение запросов по техническим изменениям (ECRs)

Управляющий комитет специальной группы интересов PCI (SIG) постановил, чтобы последующие запросы ECRs, которые находятся в различных стадиях ECR - цикла утверждения, были включены в данную промышленную версию спецификации:

Первые два запроса, которые прошли 30 дней освидетельствования группой SIG без значительных дополнений или изменений, были одобрены управляющим комитетом для учета в спецификации. Они пройдут мандатный период, с учетом поступления протеста от членов SIG в течение срока до 20 мая 1993 года:

1. ″Стандартный доступ к пространству конфигураций″ (″Standardized Access to Configuration Space″)
2. ″Стандартный механизм для генерирования специальных циклов″ (″Standardized Mechanism for Generating Special Cycles″)

Считается, что два из трех предложенных запросов ECRs делают слишком малый акцент на существующие разработки и вряд ли смогут получить сколько-нибудь значимые результаты для компаний, входящих в состав группы SIG.

Таким образом, были включены следующие дополнения:

3. ″Mechanical Modifications to ABC Addendum″ (″Механические поправки для ABC - приложений″) - здесь корректируется ряд механических параметров;
4. ″Fast Back-to-Back Cycles″ (″Быстрые циклы back-to-back″) - точно определяет целевое поведение и обеспечивает дополнительный механизм, в котором можно определять возможности и ″поведение″ шины. Вполне возможно, что большинство целей уже ″неосознанно″ обеспечивают это решение.

Последний предложенный ECR, который не был включен в финальную спецификацию, также не затрагивает значимых проблем:

5. ″Mechanical: ISA Extender Design″ (″Механическая проблема: разработка расширителя ISA″) - корректирует разработку, использующую существующую системную механическую поддержку, и добавляет вентиляционные отверстия на расширитель, чтобы улучшить вентиляцию системы.

Последние три ECRs были представлены 23 апреля 1993 года в SIG, для рассмотрения. Каждый из них пройдет проверку и период ″протеста″ в соответствии с уставом SIG. Если на одном из пяти запросов ECRs будет затронута любая значительная проблема, то компании, входящие в SIG, и зарегистрированные держатели спецификации будут своевременно уведомлены относительно данной проблемы и ее влияния на спецификацию, как определено здесь.

Комментарий к тексту

В этом документе будут использоваться следующие наименования:

Установившийся, неустановившийся — Эти термины относятся к видимому общему состоянию сигнала по фронту синхроимпульса, а не к переходным сигналам.

Фронт, фронт синхросигнала — Термины *фронт и фронт синхросигнала* относятся к положительному фронту сигнала. Для шины PCI имеют значение тактирующие сигналы именно с положительным фронтом.

— Символ # в конце названия сигнала указывает, что активное состояние сигнала - при низком значении напряжения. Отсутствие символа # указывает, что сигнал активен при высоком напряжении.

зарезервировано — Содержание, неопределенные состояния или другая информация в настоящий момент времени еще не определены. Использование любой зарезервированной области в спецификации PCI не разрешается. Все такие области в спецификации PCI могут быть изменены только согласно уставу Специальной группы интересов PCI. Любое использование зарезервированных областей в PCI спецификации приведет к тому, что изделие не будет PCI-совместимо. Функциональные возможности такого изделия нельзя гарантировать для этой или любой другой будущей реализации спецификации PCI.

наименования сигналов — Наименования сигналов отмечаются **полужирным шрифтом.** При первом упоминании о сигнале, его полное имя дается вместе с сокращением в круглых скобках. После этого сигнал уже упоминается с сокращением.

диапазон сигнала За именем сигнала, в скобках, указывается его диапазон, например AD[31:00], представляющий диапазон логически связанных сигналов. Первое число в диапазоне указывает старший бит (msb), а последнее - указывает самый младший бит (lsb).

Сопутствующие документы

PCI Design Guide (Руководство по проектированию PCI), реализация 0.6, 1 ноября 1992 г. (будет пересмотрена в Q3 1993 г.)

PCI BIOS Specification (Спецификация PCI BIOS), реализация 1.0, 28 сентября, 1992 (будет пересмотрена в Q2 1993 г.)

Глава 1
Введение

1.1. Содержание спецификации

Локальная шина PCI - это высокопроизводительная 32-битная или 64-битная шина с мультиплексированными линиями адреса и данных. Она предназначена для использования в качестве связующего механизма между высокоинтегрированными периферийными контроллерами ввода-вывода, периферийными встраиваемыми платами и системами процессор/память.
Спецификация локальной шины PCI, реализация 2.0, включает протокол, электрическую, механическую и конфигурационную спецификации для локальной шины PCI и плат расширения. Описания электрических сигналов приводятся для напряжений питания 3.3В и 5.0В.

Этот документ заменяет собой *спецификацию PCI* реализации 1.0, которая определяет связующий механизм только на уровне компонент. В данной второй реализации поддерживаются спецификации для устройств, разработанных в соответствии с версией 1.0 спецификации. Спецификация 2.0 локальной шины PCI заменяет собой следующие документы:

PCI Specification, Rev. 1.0 (Спецификация PCI, реализация 1.0).

Addendum to PCI Specification, Rev. 1.0 (Приложение к спецификации PCI, реализация 1.0).

PCI Add-in Board/Connector Addendum (Применение плат расширения/разъемов PCI).*

Peripheral Component Interconnect, Add-in Board/Connector (ABC) Addendum (Final Version) (Подсоединение периферийных устройств, применение плат расширения/разъемов - последняя версия).

PCI Electrical Definition, Rev. 1.0 (Описание электрических сигналов PCI, реализация 1.0).*

* Доступно только членам Специальной группы интересов PCI (SIG)

Спецификация локальной шины PCI определяет аппаратное обеспечение PCI. За более подробной информацией относительно руководства по системному проектированию PCI и спецификации PCI BIOS обращайтесь в PCI SIG. За информацией, как вступить в PCI SIG или получить данные документы, обращайтесь к разделу 1.6.

Предварительные замечания

Операционные системы типа Windows и OS/2, ориентированные на графику, привели к появлению "узкого места" при передаче данных между процессором и периферийными устройствами в стандартных архитектурах ввода - вывода PC. Функции пересылки данных периферийных устройств с высокими требованиями к пропускаемой способности, близкой к пропускной способности системного процессора, могут устранить это "узкое место". Реальное повышение эффективности при использовании «локальной шины» наблюдается для графических интерфейсов пользователя (GUIs) и других функций, требовательных к большой ширине диапазона ввода-вывода (например, полноценное «видео», интерфейс SCSI, локальные вычислительные сети и т.д.).
Преимущества, обеспечиваемые отдельными разработками локальной шины, послужили причиной появления нескольких версий ее реализации. Выгоды установления открытого стандарта для шин системного ввода - вывода хорошо видны на примере производства PC. Важно отметить, что новый стандарт для локальных шин установлен для упрощения проектирования, снижения стоимости и расширения ассортимента отдельных компонентов локальной шины, а также плат расширения.

Применения локальной шины PCI

Главной целью разработки локальной шины PCI было установление промышленного стандарта высокоэффективной архитектуры локальной шины, которая ведет к снижению стоимости и допускает дифференциацию. В то время как главным для сегодняшних систем являются новые соотношения цена/производительность, очень важно, чтобы новый стандарт также учитывал будущие требования к системам и был применим на множестве платформ и архитектур. Рисунок 1-1 показывает множество направлений применения локальной шины PCI.

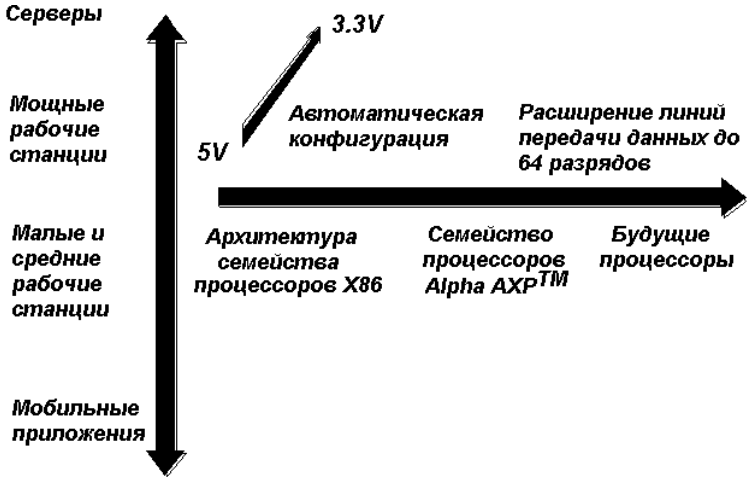


Рисунок 1-1: Применения локальной шины PCI

Несмотря на то, что первоначальной целью применения локальной шины было повышение возможностей настольных систем, она также учитывает требования к "переносным" приложениям и ведомственным серверам. Требования к напряжению питания в 3.3В для "переносных" приложений делают неизбежным учет перехода напряжения питания для настольных систем от 5В к 3.3В. Локальная шина PCI поддерживает оба напряжения питания и описывает методы перехода от одного из них к другому.
PCI-компоненты и интерфейс плат расширения не зависят от процессора, допуская эффективный переход к будущим поколениям процессора и использованию множества процессорных архитектур. Независимость от процессора позволяет оптимизировать локальную шину PCI для функций ввода-

вывода, делает возможным конкурирующую работу локальной шины с подсистемой процессор/память и позволяет использовать для графики множество высокопроизводительных периферийных устройств (видеосигналы изображения, локальная вычислительная сеть, SCSI, FDDI, жесткие диски и т.д.). Переход к расширенным видео- и мультимедийным дисплеям (а именно, к HDTV и с 3-мя измерениями) и другому многоразрядному вводу-выводу продолжит повышать требования к разрядности локальной шины. Имеющееся "прозрачное" расширение 32-разрядных шин данных и адреса до 64 разрядов, удваивает разрядность шины и делая возможным совместимость в прямом и обратном направлении периферии для 32-разрядной и 64-разрядной локальной шины PCI .

Стандарт локальной шины PCI предлагает дополнительные преимущества пользователям PCI-систем. Для PCI-компонентов и плат расширения определены регистры конфигурации. Система со встроенным программным обеспечением автоматической настройки делает легким эксплуатацию системы для ее пользователя, автоматически конфигурируя PCI-платы расширения при включении питания.

Краткий обзор локальной шины PCI

Блок-схема (рисунок 1-2) показывает типичную системную организацию локальной шины PCI. Этот пример не показывает какие-то характерные особенности архитектуры. В этом примере подсистема процессор/кэш 2-го уровня/память соединена с PCI через PCI-мост. Этот мост обеспечивает малое время задержки, за которое процессор может непосредственно обращаться к PCI устройствам, отображенным где-нибудь в адресных пространствах ввода-вывода или памяти. Он также обеспечивает широкую пропускную способность, позволяя управителям PCI напрямую обращаться в главную память. Мост может по необходимости исполнять такие функции, как буферизацию данных/регистрация и главные функции PCI (например, арбитраж).

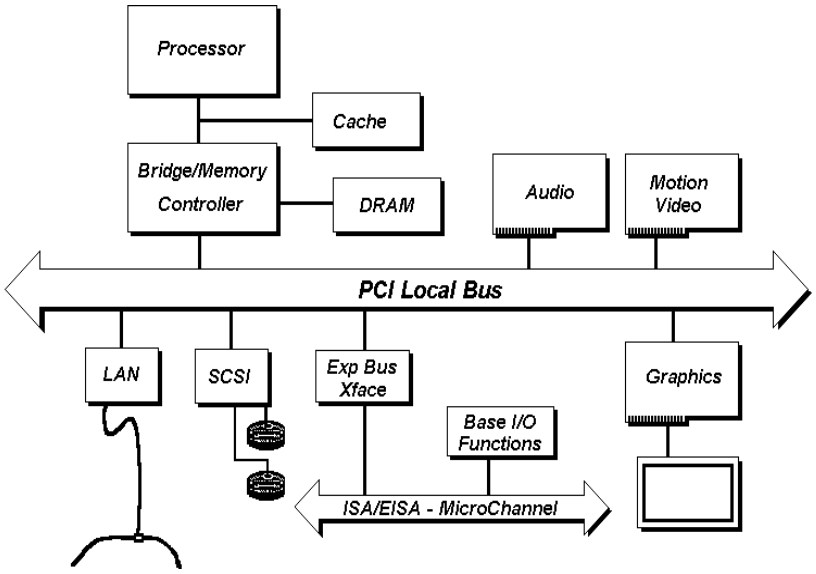


Рисунок 1-2: Блок-схема PCI-системы

Типовые реализации локальной шины PCI могут поддерживать до 3-х разъемов плат расширения, а большие возможности для расширения и не требуются. PCI - разъем платы расширения выполнен в стиле разъема для Micro Channel (MC). Это означает, что эту же самую PCI - плату расширения можно использовать в ISA-, EISA- и MC - системах. PCI - платы расширения применяются с этим разъемом на материнских платах так, что это позволяет разместить разъем типа «папа» параллельно с разъемами системной шины. Для обеспечения быстрого и легкого перехода от питающего напряжения в 5В к напряжению в 3.3В PCI предусматривает два разъема для плат расширения: один - для 5В - технологии, а другой - для 3.3В - технологии.

Существуют PCI - платы расширения двух размеров: со стандартной длиной и укороченные. Системам не требуется поддерживать оба типа плат. Стандартные платы включают в себя ISA/EISA - расширитель, что позволяет использовать ISA/EISA - модули в ISA/EISA - системах. Для использования напряжений питания 5В и 3.3В и для «сглаживания» промежуточного перехода между ними, предусмотрено 3 типа плат расширения: плата «5 volt» (5В), которая вставляется только в разъемы с напряжением 5В; «универсальная» плата, которую можно вставлять как в 5В-, так и в 3.3В - разъемы; плата «3.3 volt» (3.3В), которая вставляется в разъем с напряжением 3.3В.

В настоящее время предусмотрены два типа плат объединения с PCI: ISA/EISA - и MC - совместимые. Оба взаимозаменяемые платы объединения должны быть совместно используемы с любой PCI - платой расширения, чтобы обеспечить использование платы во всех трех типах системы.

Типичная наименьшая разрядность «диапазона пропускания» принимается равной ширине разрядности стандартных плат ввода-вывода, типа ISA, EISA или MC. Одна компонента (или множество таких компонент) может образовывать стандартную шину расширения ввода- вывода на PCI, используемую в системе. В некоторых переносных или пользовательских системах такая стандартная шина расширения может не требоваться.

1.5 Особенности и преимущества локальной шины PCI.

Локальная шина PCI предусматривала создание стандарта для различных поколений подобного изделия, которым она является. Спецификация PCI дает возможность выбора характеристик, что позволяет рассмотреть множество «точек» соотношения «цена/производительность» и много функций, ведущих к рассмотрению на системном уровне и на уровне компонент. Эти особенности разделяются по следующим категориям:

Высокая производительность:

- «Прозрачный» переход от 32-разрядных данных (132 Мб/сек) к 64-разрядным (264 Мб/сек).
- Переменная длина и переключаемый режим как для чтения, так и для записи, повышающие производительность при работе с графикой.
- Произвольный доступ с малым временем задержки (задержка 64нс при записи во «вспомогательные» регистры из «управителя» шины).
- Возможность полного параллелизма подсистемы процессор/память.
- Синхронизация шины частотой до 33 Мгц.
- Скрытый (перекрываемый) центральный арбитраж.

Низкая стоимость

- Оптимизация для прямых «межкремниевых» соединений компонент, т. е. отсутствие «склеивающей логики». Электрические, частотные спецификации и спецификации драйверов (например, полная загрузка) реализованы в соответствии со стандартными ASIC -технологиями и другими типовыми процессами.
- Мультиплексированная архитектура снижает количество выводов (47 - для сигналов подчиненного устройства, 49- для управителя) и уменьшает размеры корпуса PCI-компонент или обеспечивает дополнительные функции в корпусе заданного размера.
- Одна плата расширения работает в ISA-, EISA- и MC - системах (с минимальными изменениями для «шасси» существующих разработок), что снижает капиталовложения и бережет нервы пользователю.

Легкость в использовании

- Возможна поддержка полной автоконфигурации плат расширения и компонент локальной шины PCI. PCI - устройства имеют регистры, в которых хранится информация об устройстве, требуемая для конфигурирования.

Долговечность	• Независимость от процессора. Поддерживает множество семейств процессоров, в том числе и будущие их поколения (через переходники-«мосты» или путем прямой интеграции). • Поддержка 64-разрядной адресации. • Поддержка использования как напряжения в 5В, так и 3.3В для сигналов. • Специальные возможности делают переход в использовании напряжения 5В на 3.3В в промышленности более плавным.
Возможности к взаимодействию / надежность	• Малый размер плат расширения. • Наличие сигналов, позволяющих оптимизировать напряжение питания для ожидаемой степени загруженности системы путем отслеживания работы плат расширения, что позволяет добиться от системы максимальной эффективности работы. • Более 2000 часов электрического моделирования в пакете PSPICE с коррекцией аппаратной модели. • Прямая и обратная совместимость с 32 - разрядными и 64 - разрядными платами расширения и компонентами. • Повышенная надежность и операционные возможности взаимодействия с платами расширения путем повышения требований к загрузке и частоте локальной шины на уровне компонент, уничтожение буферов и «склеивающей логики». • MC - тип разъемов расширения.
Гибкость	• Возможности по полному управлению множеством PCI-мастеров, обеспечивающие одно-ранговый доступ любого PCI-«управителя» к любому другому PCI-«управителю» / управляемому устройству. • Общедоступный слот как для платы стандарта ISA, EISA или MC, так и для PCI - платы расширения (см. Главу 5 для дополнительных сведений).
Целостность данных	• PCI обеспечивает контроль по четности как для данных, так и для адреса, позволяя создавать устойчивые пользовательские платформы.
Программная совместимость	• PCI - компоненты могут быть полностью совместимы с существующим программным обеспечением и драйверами устройств и переноситься для различных классов платформ.

1.6. Управление

Данный документ поддерживается группой поддержки PCI - SIG. PCI SIG - независимая ассоциация членов индустрии микроЭВМ - создана для контроля и дальнейшей разработки локальной шины PCI тремя путями. Уставом PCI SIG предусматривается:

- Поддержка прямой совместимости всех разработок локальной шины PCI или приложений на ее основе.
- Поддержка спецификации локальной шины PCI как простой, легкой в использовании и устойчивой технологии в духе всего проекта.
- Способствуйте учреждению локальной шины PCI, в качестве широкого промышленного стандарта, и технически долговечной архитектуры локальной шины PCI.

Членство в SIG доступно всем претендентам в производстве микроЭВМ. Выгоды от этого следующие:

- Представление на рассмотрение изменений спецификации и предложений по приложениям.
- Участие в рассмотрении изменений спецификации и предложений по приложениям.
- Автоматическое получение изменений и приложений.
- Права голоса при выборе членов в Комитет Управления.
- Назначение идентификационных номеров (ID) продавцам.
- Техническая поддержка PCI.
- Поддержка документации и материалов по PCI.
- Участие во программе встреч, конференций, спонсором которых является SIG, и других акциях, связанных с локальной шиной PCI.

За информацией относительно того, как стать членом SIG или для получения документации по локальной шине PCI, обращайтесь по адресу:

PCI Special Interest Group
M/S HF3-15A
5200 NE Elam Young Parkway
Hillsboro, OR 97124-6497

Phone (503) 696-2000
Fax (503) 693-0929

Глава 2
Описание сигналов

Для обработки данных, адресации, управления интерфейсом, арбитража и некоторых системных функций интерфейсу PCI требуется как минимум[1] 47 выводов для целевого устройства и 49 выводов - для «управителя». На рисунке 2-1 показаны функциональные группы выводов: слева указаны необходимые выводы, а справа - необязательные. Указание на рисунке 2-1 направления сигналов подразумевает комбинацию ведущего/целевого устройства.

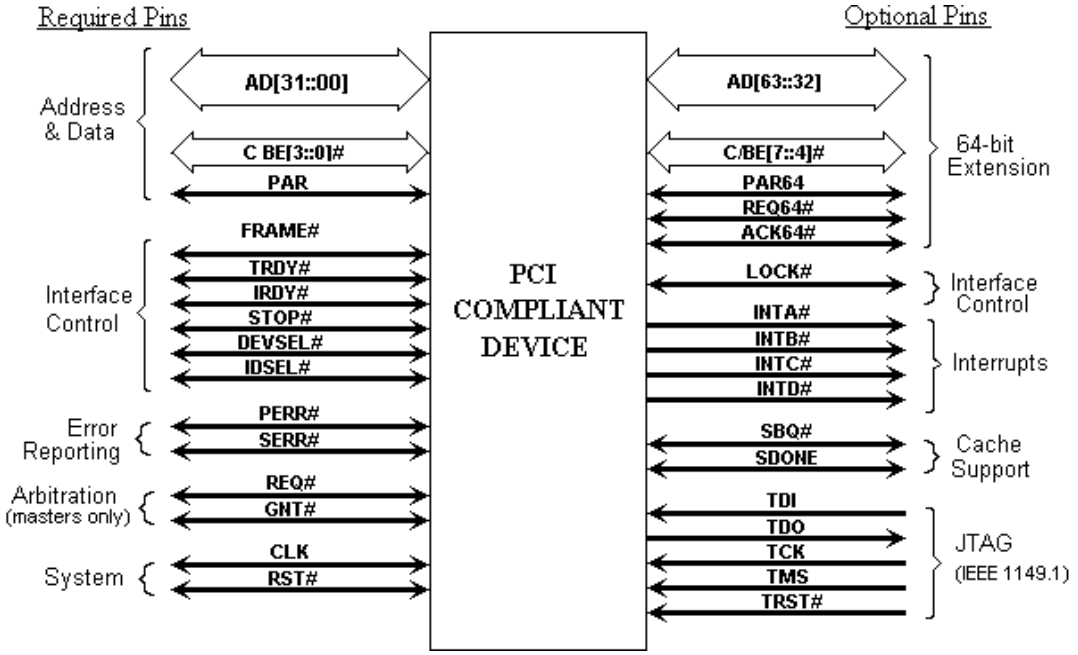


Рисунок 2-1: Список выводов PCI

[1] Минимальное число выводов для планарного устройства: для целевого - 45, для «управителя»- 47.

2.1. Описание типов сигналов

Input - стандартный входной сигнал .

Totem Pole Output - стандартный активный драйвер.

Tri-State® - это двунаправленный, с тремя состояниями, входной/выходной вывод.

Sustained Tri-State - подчиненный активный низкий сигнал с тремя состояниями, управляемый одним и только одним агентом в одно и то же времени. Агент, который управляет низким уровнем выводов s/t/s, должен сделать его «высоким» хотя бы один раз перед тем, как оставить в свободном состоянии. Новый агент не может начать управлять сигналом s/t/s, пока не пройдет один такт после того, как предыдущий «владелец» сигнала переведет его в свободное состояние. Повышение уровня требуется для поддержания неактивного состояния, пока другой агент не начнет управлять сигналом, что обеспечивается центральным ресурсом.

Open Drain - открытый коллектор - позволяет использовать количество устройств путем объединения их по «ИЛИ».

2.2. Функциональные группы выводов

Описания выводов PCI объединены в функциональные группы, которые показаны на рисунке 2-1. Символ # в конце наименования сигнала показывает, что активное состояние сигнал имеет при низком уровне напряжения. Когда символ # отсутствует, сигнал активен при высоком уровне напряжения. Способ подачи сигнала, используемый для каждого вывода, показан после названия сигнала.

2.2.1. Системные выводы

CLK входной

Clock обеспечивает синхронизацию всех транзакций на PCI, а также является входным для каждого PCI - устройства. Все другие сигналы PCI, за исключением RST#, IRQ#, IRQB#, IRQC# и IRQD#, являются дискретными по фронту CLK, а другие временные параметры определяются относительно этой границы. PCI функционирует при частоте до 33 MHz, а в общем случае минимальная частота составляет 0 Гц; тем не менее, в главе 4 описаны специфичные для отдельных компонент ограничения (обращайтесь к разделу «Спецификация синхронизации»).

RST# входной *Reset* используется для приведения специфичных для PCI регистров, секвенсоров и сигналов к соответствующему состоянию. К какому же эффекту приводит сигнал RST# для устройства, если PCI-секвенсор не поддерживает спецификацию PCI, за исключением начальных состояний регистров, которые требуются для конфигурации PCI? В любое время, когда присутствует сигнал RST#, необходимо привести все выходные сигналы PCI в нужное состояние. В общем случае это означает, что они должны быть тристабильными. Далее изменяется сигнал SERR# (открытый коллектор). Сигналы SBO# и SDONE[2] можно установить в логически низкий уровень при условии, что выходы с тремя состояниями не поддерживаются. Сигналы REQ# и GNT# оба должны быть тристабильными (во время сброса ими нельзя управлять по высокому или низкому уровню). Для предотвращения изменения сигналов AD, C/BE# и PAR центральное устройство может управлять этими линиями в течение инициализации шины, но только по логическому низкому уровню - по высокому уровню управление невозможно. Сигнал REQ64# получает значение в конце инициализации, так, как это описано в разделе 4.3.2.
RST# может становиться активным или неактивным асинхронно по отношению к сигналу CLK. Несмотря на асинхронность, приведение сигнала в неактивное состояние гарантируется для «чистого» фронта, свободного от биений (искажений). За исключением случая, когда требуется доступ для конфигурации, после инициализации могут «откликаться» только те устройства, которым требуется перезагрузить систему.

2.2.2. Адресные выводы и выводы данных

AD[31::00] t/s Адрес и данные мультиплексированы на одних и тех же выводах PCI. Транзакция шины состоит из фазы адреса[3] , сопровождаемой одним или большим количеством фаз данных. PCI поддерживает как чтение блоками, так и запись. Фаза адреса - это временной цикл, в котором активен FRAME#. В течение фазы адреса в AD[31::00] содержится физический адрес (32 бита). При вводе-выводе это - адрес байта, для конфигурации и памяти это - адрес двойного слова (DWORD). Когда идут фазы данных, AD[07::00] содержит младший значащий байт (lsb), а в AD[31::24] содержится старший значащий байт(msb). Записываемые данные «устойчивы» и правильны, когда активен сигнал IRDY#, а читаемые данные «устойчивы» и правильны, когда активен TRDY#. Данные передаются во время активности сигналов IRDY# и TRDY#.

C/BE[3::0] t/s Выводы *Bus Command* и *Byte Enables* («команды шины и разрешение байта») мультиплексированы на одних и тех же выводах PCI. Во время фазы адреса транзакции, C/BE[3::0]# определяет команду шины (смотрите раздел 3.1 для уточнения). В течение фазы данных C/BE[3::0]# используется в качестве *Byte Enable*. *Byte Enable* допустим для всей фазы данных и определяет, какие части байта несут значимые данные. C/BE[0]# применяется к байту 0 (lsb), а C/BE [3]# применяется к байту 3 (msb).

[2] Сигналы SDONE и SBO# не имеют никакого значения, пока не будет активен сигнал FRAME#, показывая начало транзакции.
[3] ЦАП использует две фазы адреса для передачи 64-разрядного адреса.

PAR t/s Parity - это контроль по четности[4] по линиям AD[31::00] и C/BE[3::0]#. Для Генерирование контрольного кода по четности требуется для всех агентов PCI. Сигнал PAR стабилен и допустим в течение одного такта после фазы адреса. Для фаз данных PAR стабилен и допустим в течение такта после того, как будет активен IRDY# - при транзакции записи, или TRDY# - при транзакции чтения. Если присутствует только сигнал PAR, то это остается в силе только в течение одного такта после завершения текущей фазы данных. (PAR имеет ту же самую синхронизацию, что и AD[31::00], но с задержкой на один такт). «Хозяин» шины управляет сигналом PAR для фаз адреса и фаз данных при записи; подчиненное же устройство управляет сигналом PAR для фаз данных при чтении.

2.2.3. Интерфейсные управляющие выводы

FRAME# s/t/s *Cycle Frame* (циклический временной интервал) управляется текущим «управителем» для указания начала и продолжительности доступа. FRAME# становится активным, когда надо указать начало транзакции шины. Пока FRAME# активен, идет передача данных. Когда сигнал FRAME# становится неактивным, транзакция переходит в заключительную фазу данных.

IRDY# s/t/s *Initiator Ready* (готовность инициализации) показывает способность агента инициализации («управителя» шины) завершить текущую фазу транзакции данных. IRDY# используется вместе с TRDY#. Фаза данных завершается в любой момент времени, когда активны IRDY# и TRDY#. Во время записи, IRDY# показывает, что на линиях AD[31::00] присутствуют достоверные данные. При чтении это показывает, что мастер готов к приему данных. Циклы ожидания вставляются до тех пор, пока активны IRDY# и TRDY#.

TRDY# s/t/s *Target Ready* (целевое устройство готово) показывает способность целевого агента (выбранного устройства) завершить текущую фазу данных транзакции. Сигнал TRDY# используется совместно с IRDY#. Фаза данных завершается в любом такте, когда активны оба сигнала TRDY# и IRDY#. При чтении TRDY# указывает, что на линиях AD[31::00] присутствуют достоверные данные. Во время записи это означает готовность целевого устройства к принятию данных. Циклы ожидания вставляются до тех пор, пока активны оба IRDY# и TRDY#.

STOP# s/t/s *Stop* показывает, что текущее подчиненное устройство посылает «управителю» запрос на останов текущей транзакции.

[4] Если на линиях AD[31::00], C/BE[3::0]# и PAR появились «1», то это означает равенство по четному значению.

LOCK# s/t/s *Lock* показывает элементарную операцию, которой для завершения требуется множество транзакций. Неисключительные транзакции при активном LOCK# могут выполняться с адресом, который в текущий момент не блокирован. Разрешение исполнения транзакции на шине PCI не гарантирует контроля над LOCK#. Контроль над LOCK# можно получить в его собственном протоколе и при наличии GNT#. В то время, как единственный мастер монопольно управляет выводом LOCK#, возможно использование шины PCI различными агентами. Если устройство реализует исполняющую память (Executable Memory), то оно также должно установить LOCK# и гарантировать полное исключение доступа в этой памяти. Целевое устройство для доступа, поддерживающее LOCK#, должно обеспечить исключение минимум 16 байтов (с учетом выравнивания). Для главных интерфейсов, находящихся после системной памяти, также необходимо выполнить LOCK#.

IDSEL входной *Initialization Device Select* (выбор устройства инициализации) используется для выбора кристалла при транзакциях чтения конфигурации и записи.

DEVSEL# s/t/s Когда активным выводом *Device Select* (выбор устройства) управляют, он показывает, что управляющее устройство дешифрировало данный адрес как цель текущего доступа. DEVSEL# в качестве входа показывает, было ли выбрано на шине какое-то устройство.

2.2.4. Арбитражные выводы (только для мастеров шины)

REQ# t/s Сигнал *Request* (запрос) показывает арбитру, что данному агенту требуется поработать с шиной. Этот сигнал - типа «от одного пункта к другому». Каждый мастер имеет свой собственный вывод REQ#.

GNT# t/s Сигнал *Grant* (разрешение) показывает агенту, что разрешен доступ к шине. Этот сигнал типа «от одного пункта к другому». Каждый мастер имеет свой собственный вывод GNT#.

2.2.5. Выводы для сообщения об ошибках

Выводы для сообщения об ошибках требуются[5] всем устройствам:

PERR# s/t/s Вывод *Parity Error* (ошибка контроля по четности) предназначен только для сообщения об ошибках контроля по четности во время всех транзакций PCI, за исключением специального цикла (Special Cycle). Вывод PERR# - три-стабильный и должен активно управляться агентом, получающим данные в течение двух тактов, после того, как обнаружена ошибка контроля данных по четности. Минимальная продолжительность PERR# - один такт для любой фазы данных, у которой обнаружена ошибка контроля данных по четности (если идут последовательно несколько фаз данных, каждая из которых имеет ошибку контроля данных по четности, то сигнал PERR# будет установлен за более, чем один такт). PERR# должен быть установлен в высокое состояние за один такт прежде, перед тем, как он перейдет в третье состояние со всеми соответствующими тристабильными сигналами. Не существует никаких специальных условий для случая, когда теряется ошибка контроля данных по четности или сообщается об отсроченной ошибке. Агент не может установить PERR#, пока он не разрешил доступ, установив DEVSEL# и завершив фазу данных.

SERR# o/d *System Error* предназначен для выдачи сообщений об ошибках контроля по четности для адреса, по команде Special Cycle (специальный цикл), или любых других системных ошибках, когда результаты могут оказаться катастрофическими. Если агенту не требуется генерирование немаскируемого прерывания (NMI), то необходим механизм для сообщения о разных событиях. SERR# представляет собой открытый коллектор и управляется в течение единственного такта PCI, когда агент сообщает об ошибке. Установление SERR# синхронизировано во времени, при этом требуется время на установку и «замораживание» всех сигналов на шине. Однако установка SERR# в неактивное состояние происходит при небольшом повышении уровня напряжения (до той же величины, что и для тристабильных сигналов), и это должно обеспечиваться системным разработчиком, а не агентом или центральным ресурсом. Такое повышение напряжения может занимать от двух до трех временных интервалов до полного восстановления SERR#. Агент, который посылает операционной системе сигналы SERR#, делает это в любой момент времени, когда установлен сигнал SERR#.

[5] Для устройств, которым разрешены некоторые исключительные ситуации (смотрите раздел 3.7.2. для дополнительной информации по данному вопросу).

2.2.6. Выводы прерывания (необязательно)

Прерывания на PCI произвольны и определяются как «чувствительные к уровню», т.е. устанавливаются по низкому уровню (отрицательное «истинно»), при этом для устройств используется выход с открытым коллектором. Переход сигналов INTx# в активное состояние и обратно асинхронно по отношению к CLK. PCI предусматривает одну линию прерываний для устройства с одной функцией, и до четырех линий прерывания - для многофункциональных[6] устройств или соединителя. Для одно-функционального устройства может использоваться только линия INTA#, в то время как три других линий прерывания не имеют никакого значения.

INTA# o/d *Interrupt A* - используется для запроса прерывания.

INTB# o/d *Interrupt B* - используется для запроса прерывания и имеет значение только для многофункционального устройства.

INTC# o/d *Interrupt C* - используется для запроса прерывания и имеет значение только для многофункционального устройства.

INTD# o/d *Interrupt D* - используется для запроса прерывания и имеет значение только для многофункционального устройства.

Любая функция на многофункциональном устройстве может быть соединена с любой линией INTx#. Регистр вывода прерывания определяет, какая из линий INTx# используется для запроса прерывания. Если устройство реализует единственную линию INTx#, то она называется INTA#; если реализуются две строки, то они называются INTA# и INTB#; и т.д. Все функции многофункционального устройства могут использовать одну и ту же линию INTx#, либо каждая функция может иметь собственную линию (по максимальному количеству функций), либо любую комбинацию такого набора. Одна и та же функция не может генерировать прерывание более, чем на одной линии INTx#.

Поставщик системы свободен в выборе способа объединения различных сигналов INTx# из разъема PCI, для их соединения с контроллером прерываний. Они могут быть объединены по «ИЛИ», либо переключаться электроникой под управлением программы, либо как-то иначе, путем комбинации вышеперечисленных способов. Это означает, что драйвер устройства не может делать какие-то «предположения» относительно совместного использования прерываний. Все драйверы PCI - устройств должны обладать способностью к совместному использованию прерываний (цепочек прерываний) с любым другим логическим устройством, включая устройства в этом же самом многофункциональном модуле.

[6] Когда несколько независимых функций объединены в одном устройстве, то это будет называться, в целом, многофункциональным устройством. Каждая функция в таком устройстве обладает собственным пространством конфигураций.

2.2.7. Выводы поддержки кэша (необязательно)

Кэшируемая память PCI должна реализовывать оба вывода поддержки кэширования в качестве входных, чтобы разрешить работу как с кэшем сквозной записи, так и с кэшем обратной записи. Если кэшируемая память размещена на PCI, то интерфейс, соединяющий кэш обратной записи и PCI, должен реализовывать оба вывода в качестве выходных; а интерфейс, соединяющий PCI и кэш сквозной записи, может реализовывать только один вывод, как это описано в разделе 3.8.

SBO# **вх/вых** *Snoop Backoff* указывает удачную попытку для изменения состояния линии. Когда сигнал SBO# неактивный, и активен SDONE, то это означает успешный результат «вмешательства».

SDONE **вх/вых** *Snoop Done* указывает состояние «вмешательства» для текущего доступа. Когда сигнал неактивен, то это показывает, что результат «вмешательства» все еще ожидается. Когда сигнал активен, то это показывает, что «вмешательство» завершено.

2.2.8. Выводы расширения шины до 64-бит (необязательно)

Выводы расширения до 64 бит в общем случае не обязательны. Это означает, что если расширение до 64 бит используется, то задействованы все выводы в этой секции.

AD[63::32] **t/s** *Address* и *Data* (адрес и данные) мультиплексированы на одних и тех же выводах и обеспечивают 32 дополнительных разряда. В течение фазы адреса (когда используются команды ЦАП и когда активен REQ64#) передаются старшие 32 бита 64-разрядного адреса; в противном случае, эти биты резервные[7] , но при этом они устойчивы и не определены. В течение фазы данных, когда активны REQ64# и ACK64#, передаются дополнительные 32 бита данных.

C/BE [7::4]# **t/s** *Bus Command* и *Byte Enables* (команды шины и разрешение байта) мультиплексированы на одних и тех же выводах. В течение фазы адреса (когда используются команды ЦАП и когда активен REQ64#) передается фактическая команда шины по линиям C/BE[7::4]#; в противном случае, эти биты зарезервированы и не определены. В течение фазы данных, по линиям C/BE[7::4]# передается байт, который показывает, какой байт содержит значимые данные, при условии, что активны оба сигнала REQ64# и ACK64#. Сигнал C/BE[4]# применяется к байту 4, а C/BE[7]# применяется к байту 7.

REQ64# **s/t/s** Когда *Request 64-bit Transfer* управляется текущим «управителем» шины, то он показывает, что тот желает передать данные, использую для пересылки 64 бита. REQ64# имеет такие же временные параметры, что и FRAME#. REQ64# получает значение в конце сброса, как это описано в разделе 4.3.2.

ACK64# **s/t/s** Когда *Acknowledge 64-bit Transfer* управляется устройством, которое успешно дешифрировало данный адрес в качестве агента текущего доступа, то он показывает, что агент желает передать данные, используя при этом 64 бита. ACK# имеет такие же временные параметры, как и DEVSEL#.

[7] Это означает резервирование данных разрядов Управляющим Комитетом PCI SIG для будущего использования. Зарезервированные биты не должны использоваться каким - либо устройством.

PAR64 t/s *Parity Upper DWORD* - это бит контроля по четности, который защищает линии AD[63::32] и C/BE[7::4]. PAR64 идет в течение одного такта после фазы начального адреса, когда активен REQ64, и по линии C/BE[3::0] поступает команда ЦАП. Также PAR64 идет в течение такта после второй фазы адреса команды ЦАП.
PAR64 устойчив и корректен для тех фаз данных, когда активны REQ64# и ACK64#, в течение одного такта, когда активен IRDY# при транзакции записи, или активен TRDY# при транзакции чтения. PAR64 допустим один раз, это остается в течение такта после завершения фазы данных. (PAR64 имеет те же временные параметры, что и AD[63::32], но с задержкой на один такт). «Мастер» управляет PAR64 во время фазы адреса и фазы записи данных; агент управляет PAR64 во время фаз чтения данных.

2.2.9. Выводы JTAG / периферийного сканирования (необязательно)

Стандарт IEEE 1149.1, *Порт для тестирования и архитектура периферийного сканирования* («*Test Access Port and Boundary Scan Architecture*»), включен в качестве необязательного интерфейса для PCI устройств. Стандарт IEEE 1149.1 определяет правила и ограничения для проектирования ИС (интегральных схем) в соответствии с 1149.1. Включение в состав устройства порта для тестирования (TAP - *Test Access Port*) позволяет использовать периферийное сканирование для проверки устройства и платы, на которой данное устройство установлено. TAP состоит из четырех выводов (в общем случае - из пяти), которые используются для организации последовательного интерфейса с контроллером TAP внутри PCI- устройства.

TCK in *Test Clock* используется для синхронизации ввода собранной информации и данных в устройство и их вывода во время работы с TAP.

TDI in *Test Data Input* используется для последовательного ввода в устройство тестирующих данных и команд при работе с TAP.

TDO out *Test Output* используется для последовательного вывода тестирующих данных и команд из устройства при работе с TAP.

TMS out *Test Mode Select* используется для управления в устройстве состоянием контроллера TAP.

TRST# in *Test Reset* обеспечивает асинхронную инициализацию контроллера TAP. Этот сигнал по стандарту IEEE 1149.1 необязателен.

Данные выводы TAP должны работать при тех же электрических условиях (5В или 3.3В), что и буферы ввода-вывода PCI - интерфейса устройств. Кроме того, управление выводом TDO необязательно должно быть таким же, как это делается для стандартных выводов шины PCI. Способ управления TDO должен быть указан в техническом паспорте устройства.
Поставщик системы ответственен за проектирование и функционирование в системе последовательных цепочек стандарта 1149.1 («кольца»). Дополнительные к шине PCI сигналы не используются в «многоточечном» режиме. Обычно «кольцо» по стандарту 1149.1 создается путем соединения вывода TDO одного устройства с выводом TDO другого, чтобы получить последовательную цепочку устройств. В этом случае микросхемы получают одни и те же сигналы TCK, TMS и необязательные сигналы TMS#. Все кольца по стандарту 1149.1 соединены либо с тестирующим разъемом материнской платы для целей тестирования, либо к ИС резидентного контроллера по стандарту 1149.1.

Спецификация PCI поддерживает платы расширения с разъемом, который предусматривает сигналы периферийного сканирования. Устройства на плате расширения можно соединять в цепочку на материнской плате.

Методы соединения и использования системы колец по стандарту 1149.1 с платами расширения включают:

- Использование кольца по стандарту 1149.1 на плате расширения только во время тестирования этой платы расширения на производстве. В этом случае, кольцо по стандарту 1149.1 на материнской плате не должно контактировать с сигналами по стандарту 1149.1 для плат расширения. Материнская плата должна самотестироваться непосредственно в ходе производства.

- Создание для каждой платы расширения в системе независимых колец, по стандарту 1149.1, на материнской плате. Например, если на плате есть два разъема расширения, то на материнской плате должны оставаться свободные кольца по стандарту 1149.1.

- Инициализацию ИС, которая допускает иерархичную многоточечную адресацию по стандарту 1149.1. Это позволит обрабатывать множество колец по стандарту 1149.1 и разрешит многоточечную адресацию и операции.

- Платы расширения, не поддерживающие стандарт интерфейса IEEE 1149.1, должны осуществлять переход от вывода TDI платы к выводу TDO.

За более подробной информацией относительно использования JTAG / Периферийного сканирования в PCI - системе обращайтесь к *PCI System Design Guide (Руководству по системному проектированию PCI).*

2.3. Остальные сигналы

Обеспечиваются все основные механизмы пересылки, в основном, универсальные, а также множество мастеров. Однако это не препятствует повышению эффективности изделия за счет вспомогательных сигналов (*sideband*). В качестве таких сигналов могут использоваться любые сигналы, которые не входят в спецификацию PCI, но которые соединяют два или более PCI - агентов и имеют значение только для них. Данные сигналы могут использоваться для одного или большего количества устройств с целью объединения их специфичных состояний и обеспечения максимальной эффективности использования в системе шины PCI. Эти сигналы предусматриваются в разъеме PCI. Отсюда следует, что на них действуют ограничения, связанные с планарным расположением. Кроме того, данные сигналы могут нарушать специфицированный протокол для определенных сигналов PCI, либо приводить к таким нарушениям протокола.

2.4. Функции центрального ресурса

Вне этой спецификации термин *центральный ресурс* используется для описания опорных выводов шины, обеспечиваемых главной системой, обычно для PCI - интерфейса, либо стандартного интерфейса. Эти функции могут включать следующее (но этим не ограничиваться):

- Центральный арбитраж.

- Приведение требуемых сигналов в активное состояние, как это описано в разделе 4.3.3.

- Вычитающее дешифрирование. Только один агент на шине PCI может использовать вычитающее дешифрирование и, обычно, предоставлять интерфейс к стандартной шине расширения (смотрите раздел 3.2.2.).

- Генерирование индивидуальных сигналов IDSEL для каждого устройства в целях конфигурации системы.

- Управление сигналом REQ64# во время инициализации.

Глава 3
Функционирование шины

3.1. Операции на шине

Операции на шине показывают агенту тип транзакции, которая требуется мастеру. Эти операции кодируются в течение фазы адреса на линиях C/BE[3::0]#.

3.1.1. Описание операций

Код операций на PCI - шине и их типы приведены ниже, с кратким описанием каждого из них. Обратите внимание, что коды команд приведены в таком виде, в каком они присутствуют на шине («1» показывает высокий уровень напряжения, «0» - низкий уровень). *Byte Enables (побайтовый доступ или, иначе, разрешение байта)* активен при низком уровне напряжения («0»).

C/BE[3::0]#	Тип операции
0000	Interrupt Acknowledge (подтверждение прерывания)
0001	Special Cycle (специальный цикл)
0010	I/O Read (чтение при вводе - выводе)
0011	I/O Write (запись при вводе - выводе)
0100	Зарезервировано
0101	Зарезервировано
0110	Memory Read (чтение памяти)
0111	Memory Write (запись в память)
1000	Зарезервировано
1001	Зарезервировано
1010	Configuration Read (чтение конфигурации)
1011	Configuration Write (запись конфигурации)
1100	Memory Read Multiple (множественное чтение памяти)
1101	Dual Address Cycle (двойной цикл адреса)
1110	Memory read Line (линия чтения памяти)
1111	Memory Write and Invalidate (запись в память и недействительные данные)

Команда *Interrupt Acknowledge* представляет собой неявное обращение к системному контроллеру прерываний. Биты адреса в течение фазы адреса не имеют логического значения и показывают длину возвращаемого вектора.

Команда S*pecial Cycle* обеспечивает простой механизм передачи сообщений по шине PCI. Он используется в качестве альтернативного для физических сигналов, когда необходимо организовать связь по *sideband* - сигналам. Данный механизм полностью описан в разделе 3.6.2.

Команда *I/O Read* используется для чтения данных, поступающих от агента, отображенного в адресном пространстве ввода - вывода. По линиям AD[31::00] поступает адрес байта. Должны дешифрироваться все 32 бита. *Byte Enables* указывает размер передачи, при этом он должен соответствовать адресу байта.

Команда *I/O Write* используется при передаче данных агенту, отображенному в адресном пространстве ввода - вывода. Должны дешифрироваться все 32 бита. *Byte Enables* указывает размер передачи и должен соответствовать адресу байта.

Зарезервированные коды команд предназначены для будущего использования. PCI - устройства не должны использовать эти коды с остальными командами, а также не должны на них отвечать. Если в интерфейсе используются зарезервированные коды, то доступ должен быть завершен аварийным прекращением работы мастера.

Команда *Memory Read* используется для чтения данных от агента, отображенного в пространстве адресов памяти. Целевое устройство свободно в выборе, выполнять или нет упреждающее чтение для этой команды, при условии гарантии, что при таком чтении не будет никаких побочных эффектов. Кроме того, для данной PCI - транзакции целевое устройство должно убедиться в синхронизации данных, хранящихся во временных буферах. Эти буферы должны быть отменены перед любыми событиями синхронизации (например, обновление регистра состояния ввода-вывода или флажка памяти), при обращении через данный путь доступа.

Команда *Memory Write* используется при передаче данных агенту, отображенному в пространстве адресов памяти. Когда целевое устройство возвращает признак готовности, это подразумевает готовность данных и актуальность их значения. Это может обеспечиваться как при выполнении данной команды полностью синхронным способом, так и при проверке любого программного буфера, прежде, чем его содержимое будет «сброшено» перед каким-то событием синхронизации (например, при обновлении регистра состояния ввода - вывода или флажка памяти), через данный путь доступа. В данном случае подразумевается, что мастер может создать событие синхронизации сразу после использования этой команды.

Команда *Configuration Read* используется для чтения пространства конфигурации каждого агента. Агент выбран, когда активен его сигнал IDSEL, и на линии AD[1::0] присутствует 00. Во время фазы адреса цикла или цикла конфигурации, AD[7::2] адресует одно из 64 двойных слов DWORD (и внутри каждого DWORD допускается адрес байта или байтов); конфигурации с информацией о каждом устройстве, а AD[31::11] - логически это не предусматривает. AD[10::08] показывает, что каждый элемент многофункционального агента адресован.

Команда *Configuration Write* используется для передачи данных в пространство конфигураций агента. Агент выбран, когда активен его сигнал IDSEL, и на AD[1::0] присутствует 00. Во время фазы адреса цикла конфигурации линии AD[7::2] адресуют 64 двойных слова DWORD (и внутри каждого двойного слова допускается адрес байта или байтов) конфигурации с информацией о каждом устройстве, а AD[31::11] - логически это не предусматривают. AD[10::08] показывает, что каждый элемент многофункционального агента адресован.

Команда *Memory Read Multiple* является семантически идентичной команде *Memory Rea*d, за исключением того, что в ней дополнительно указывается возможность выбора мастером более, чем одной строки кэширования, перед отсоединением. Контроллер памяти должен продолжать конвейерную обработку памяти, пока активен сигнал FRAME#. Эта команда предназначена для работы по передаче больших последовательностей данных, когда можно повысить эффективность системы памяти (и запрашивающего мастера), при последовательном чтении дополнительной строки кэша, когда программной установленный буфер доступен для временного хранения.

Команда *Dual Address Cycle (DAC)* используется для передачи 64-разрядного адреса в устройства, поддерживающие 64-битную адресацию. Те же устройства, которые поддерживают только 32-битную адресацию, должны обработать эту команду, как зарезервированную, и никогда не отвечать на текущую транзакцию.

Команда *Memory read Line* является семантически идентичной команде *Memory Read*, за исключением того, что в ней дополнительно указывается возможность завершения мастером более, чем двух 32-битных фаз данных PCI. Эта команда предназначена для работы с большими последовательностями передаваемых данных, когда можно повысить производительность системы памяти (и запрашивающего мастера), при чтении строки кэша до конца в случае, если время реакции на запрос менее одного цикла памяти. Как и в случае с командой *Memory Read*, буферы выборки должна быть отменены прежде, через данный путь доступа будут инициированы события синхронизации.

Команда *Memory Write and Validate* семантически идентична команде *Memory Write*, за исключением того, что она дополнительно гарантирует передачу как минимум одной полной строки кэша; например, это требуется, когда мастеру необходимо записать все байты внутри адресованной строки кэша за одну транзакцию PCI. Мастер может позволить транзакции «перейти» границу строки кэша только в случае, если это предполагает передачу и всей последующей строки. Для выполнения этой команды требуется наличие у мастера регистра конфигурации, в котором был бы указан размер строки кэша (за более подробной информацией обращайтесь к разделу 6.2.4). Это позволит повысить производительность памяти, путем отмены строки в кэше обратной записи, без осуществления фактического цикла обратной записи, и, таким образом, сокращая время доступа. Обращайтесь к разделу 3.3.3.1. за информацией относительно блокировок по времени обработки.

3.1.2. Правила использования операций

Всем устройствам PCI требуются реагировать на команды конфигурации (чтения и записи) в качестве целевых устройств. Все остальные команды необязательные. Порядок исполнения команд на шине PCI гарантирован для операций ввода - вывода (чтения и записи). Устройствам PCI, которые содержат перегружаемые функции или регистры, требуется осуществить их отображение в пространстве памяти через регистры конфигурации. Данное действие обеспечивает возможность использования устройства в конфигурациях, в случае, если пространство ввода-вывода недоступно. Когда выполнено такое отображение, то проектировщик системы может гарантировать порядок выполнения команды, когда устройство используется при вводе - выводе или в пространстве памяти. Команды записи в память и чтения из нее, для отображенного в память устройства, осуществляют «ввод - вывод при отображении в память» («memory mapped I/O»).

При необходимости мастер может выполнять необязательные команды. Целевое устройство также может, при необходимости, выполнять эти команды, но если оно реализует основные операции с памятью, то необходимо поддерживать все подобные операции, включая Memory Write (запись в память) и Invalidate (отмена), Memory Read Line (чтение строки памяти), Read Multiple (множественное чтение). При неполной реализации, данные команды повышения производительности должны быть совмещены с основными командами работы с памятью. Например, целевое устройство может не реализовывать команду Memory Read

Line; тем не менее, оно должно принять запрос (если адрес дешифрирован как для доступа к памяти) и обработать его как команду чтения из памяти (Memory Read). Аналогично, целевое устройство может не реализовывать команду Memory Write и Invalidate, но оно обязано принять запрос (если адрес дешифрирован как для доступа к памяти) и обработать его как команду Memory Write (запись в память).
Для передача блоков данных в системную память и обратно мастеру рекомендуется поддерживать команды Memory Write and Invalidate и Read Memory Line. Команды Memory Read или Memory Write можно использовать, если по некоторым причинам мастер не способен использовать команды, повышающие производительность.
Мастера, использующие команды чтения из памяти данных любой длины (Memory Read), будут работать для всех команд, однако их основное применение показано ниже. Несмотря на то, что команда чтения и отмены (Write and Invalidate) - единственная команда, которая требует наличия регистра длины строки кэша, настоятельно рекомендуется, чтобы ею пользовались команды чтения памяти (Memory Read). Во всех остальных случаях, интерфейс не гарантирует правильности любых «неявных» данных. Основное использование команды показано как для случая с использованием регистра длины строки кэша.

Основное использование команд с применением *регистра длины строки кэша*:

Команда *Memory Read*	Используется в случае блочной передачи половины строки кэша или менее.
Команда *Memory Read Line*	Используется в случае блочной передачи от половины строки кэша до трех строк кэша.
Команда *Memory Read Multiple*	Используется при блочной передаче более трех строк кэша.

Основное использование команд без применения *регистра длины строки кэша*:

Команда *Memory Read*	Используется при блочной передаче, если имеется две и менее пересылок данных.
Команда *Memory Read Line*	Используется при блочной передачи, если количество пересылок данных составляет от 3 до 12.
Команда *Memory Multiple Read*	Используется при передаче больших блоков данных (до 13 и более пересылок).

3.2. Основы протокола для PCI

Основным механизмом передачи данных на шине PCI является блочный механизм. Блок состоит из фазы адреса и одной или более фаз данных. PCI поддерживает передачу блоками как для адресного пространства памяти, так и для адресного пространства ввода - вывода. Основной интерфейс (между главным процессором и шиной PCI) может объединять запросы по записи в память в единую транзакцию, при условии, что не будет никаких побочных эффектов. В этом случае устройство устанавливает бит предварительной выборки в регистре базового адреса (чтобы разрешить чтение данных и объединение данных для записи в любом порядке). Интерфейс может различать, когда такое объединение разрешено, а когда - нет, при помощи анализа адресного диапазона, данные по которому должно предоставлять программное обеспечение при инициализации. Прекращение объединения данных в буфер (и сброс содержимого буфера) должно происходить во время следующей записи или чтения, которые заранее не предусмотрены (для любого диапазона). Транзакции записи после любого из этих двух событий могут быть объединены с последующими транзакциями, но это не относится к ранее объединенным данным, если они попали в «предсказанный» диапазон.
Основной интерфейс может комбинировать последовательности двойных слов записываемых в память данных (DWORD), которые генерируются процессором в блоки, при условии сохранения порядка следования адресов (ассоциированного с каждым двойным словом). Например, интерфейс может записывать следующую последовательность - DWORD 0, DWORD 2 и DWORD 3. Тогда последовательность на PCI может быть следующей: DWORD 0, DWORD 1 (байты не допускаются), DWORD 2 и в конце блока - DWORD 3. Такое объединение допускается во всех случаях, когда последующий адрес DWORD более значим, чем предыдущий. Интерфейс может преобразовывать одиночные запросы по чтению памяти, поступающие от процессора, в пакетный запрос (чтение производится с выборкой, до прихода данных в процессор), при условии, что при этом чтении не будут возникать побочные эффекты в адресуемом целевом устройстве.

Если запросы от процессора для ввода-вывода нельзя объединить, то они будут иметь единственную фазу - фазу данных. В настоящее время нет таких процессоров или «управителей» шины, которые бы генерировали пакетные запросы в пространстве ввода - вывода. Однако, если в будущем некоторое новое устройство будет генерировать значимые пакетные запросы на ввод-вывод, (например, для доступа к порту FIFO), то они не будут игнорироваться. Для пакетных запросов по вводу-выводу адресация не подразумевается. В этом случае адресация должна быть установлена целевым устройством и мастером после выполнения запросов по вводу-выводу. PCI - устройства, которые не работают с фазами данных при множественном вводе - выводе, должны разрывать связь после первой фазы данных. Чтобы убедиться, что устройства ввода - вывода функционируют правильно, интерфейсы не должны объединять или комбинировать в одиночный запрос доступа к PCI или блок последовательности запросов ввода - вывода. Все запросы на ввод - вывод должны появляться на шине PCI точно так, как их сгенерировал процессор. (Если адресуется целевое устройство, для которого *Byte Enables* показывает, что объем передаваемых данных больше, чем оно поддерживает, то это устройство аварийно завершает свою работу).

Значение всех сигналов проверяется по фронту синхроимпульса[1] . Каждый сигнал имеет свою установочную апертуру и апертуру неактивного состояния относительно положительного фронта синхроимпульса, при которых не позволяется переход в другое состояние. Величины сигналов выше этой апертуры не имеют никакого значения. Эта апертура имеет значение только на «подходящих» фронтах синхроимпульсов для сигналов AD[31::00], [63::32], PAR[2] , PAR64 и IDSEL[3] и на любом фронте синхроимпульса для LOCK#, IRDY#, TRDY#, FRAME#, DEVSEL#, STOP#, REQ#, GNT#, REQ64#, ACK#64, SBO#, SDONE, SERR# (только на срезе), а для сигналов PERR#, C/BE[3::0]#, C/BE[7::4]# (так как это команды шины) существует ограничение по положительному фронту синхроимпульса, при условии активности сигнала FRAME#. Сигналы C/BE[3::0], C/BE[7::4]# (с разрешением байта) ограничены на любом положительном фронте синхроимпульса, после завершения фазы адреса или фазы данных. RST*, IRQA*, IRQB*, IRQC*, и IRQD* не ограничены и асинхронны.

[1] Единственным исключением являются RST#, INTA#, INTB#, INTC# и INTD#, которые обсуждаются в разделе 2.2.1.
[2] PAR и PAR64 обрабатываются по одноименным линиям, с задержкой на один такт.
[3] Замечания по ограничениям сигналов AD и IDSEL полностью приведены в разделе 3.6.3.

3.2.1. Общее управление передачей информации

Все основные пересылки данных на шине PCI управляются тремя сигналами. За более подробной информацией обращайтесь к рисунку 3-1.

FRAME#	Управляется мастером для того, чтобы он мог указать начало и конец транзакции.
IRDY#	Управляется мастером, чтобы он мог инициировать циклы ожидания.
TRDY#	Управляется целевым устройством, чтобы оно могло инициировать циклы ожидания.

Когда неактивны сигналы FRAME# и IRDY#, интерфейс находится в ожидании - состояние IDLE. Первый фронт синхроимпульса, на котором активизируется сигнал FRAME# - это фаза адреса, в которую передаются адрес и команда шины. По следующему фронту синхроимпульса начинается первая фаза данных или более, в течение которой передаются данные между мастером и целевым устройством по фронту синхроимпульса, для которого активны сигналы IRDY# и TRDY#. Циклы ожидания могут быть инициированы в фазе данных мастером либо целевым устройством, с сигналами IRDY# и TRDY#, соответственно.
Когда данные корректны, для независимой ни от чего установки сигнала xRDY# требуется источник данных (IRDY# - для транзакции записи, TRDY# - для транзакции чтения). Получение данных может привести к установлению в активное состояние сигналов xRDY#, конкретно - в зависимости от того, какой сигнал выбран.
Если мастер установил сигнал IRDY#, то он не может изменять состояние сигналов IRDY# или FRAME# до тех пор, пока не завершится текущая фаза данных, независимо от состояния TRDY#. Если целевое устройство один раз уже установило сигнал TRDY# или STOP#, то оно не может изменять состояние сигналов DEVSEL#, TRDY# или STOP# до тех пор, пока не завершится текущая фаза данных. Ни мастер, ни целевое устройство не могут изменять свое состояние, пока не завершится передача данных.
Когда мастер предполагает завершить одну большую передачу данных (это может произойти сразу после фазы адреса), то сигнал FRAME# переходит в неактивное, а IRDY# - в активное, показывая готовность мастера. После того, как целевое устройство показывает, что данная передача - последняя (сигнал TRDY# - активен), интерфейс возвращается в состояние ожидания IDLE, с активными сигналами FRAME# и IRDY#.

3.2.2. Адресация

Все определено три физических адресных пространства. Пространства адресов памяти и ввода-вывода объединены. Адресное пространство конфигураций было введено для обеспечения аппаратной конфигурации PCI. Работа с этим пространством описана далее, в разделе 3.6.4.1.
Дешифрирование адреса на шине PCI распределено; это означает, что оно выполняется на каждом устройстве. Это устраняет проблемы для центральной дешифрирующей логики, а также для сигналов выбора устройства, независимо от их использования для конфигурации. Каждый агент отвечает только на свой дешифрированный адрес. PCI поддерживает два режима дешифрирования адреса: суммирующий и вычитающий. Суммирующее дешифрирование быстрее, так как устройство ищет запрос в своем адресном интервале. Вычитающее дешифрирование может выполняться только одним устройством на шине, так как это подразумевает запрос, который не может быть положительно дешифрирован каким - то другим агентом. Данный механизм дешифрирования медленнее, так как он должен давать всем остальным агентам на шине «первое право по отказу» при запросе.
Тем не менее, это может оказаться полезным для такого агента, как стандартная шина расширения, которая должна работать в высоко-фрагментированном адресном пространстве. Целевые устройства,

которые выполняют как положительное, так и отрицательное дешифрирование, не должны реагировать на зарезервированные команды шины (путем установки сигнала DEVSEL#).

Информация, содержащаяся в двух младших битах адреса (AD[1::0]), определяется адресным пространством. Чтобы обеспечить полный адрес байта, в адресном пространстве ввода - вывода используются все 32 линии. Это позволяет агенту, требующему разрешение адресации на уровне байта, завершить дешифрирование адреса и начать свой цикл[4] без дополнительного цикла ожидания разрешения адресации байта (таким образом осуществляется задержка всех циклов вычитающего дешифрирования дополнительным тактом). AD[1::0] используются только для генерирования сигнала DEVSEL#, указывающего при передаче последний допустимый байт. Например, если BE0# были активны, то AD[1::0] будет ″00″; если были активны только BE3#, то AD[1::0] будет ″11″. Как только целевое устройство установило запрос на ввод - вывод (используя AD[1::0]), оно может определить, можно ли завершить запрос, как это указано разрешением байта. Если все выбранные байты оказались вне выбранного целевым устройством адресного интервала, то данный запрос не может быть завершен. В этом случае целевое устройство не может передать никакие данные, но прекращает свою работу с аварийным завершением. Таблица ниже подводит итог по дешифрирования AD[1::0].

AD1	**AD0**	**C/BE3#**	**C/BE2#**	**C/BE1#**	**C/BE0#**
0	0	X	X	X	0
0	1	X	X	0	1
1	0	X	0	1	1
1	1	0	1	1	1

Примечание:
1 = неактивное состояние
0 = активное состояние
X = 1 либо 0
Любая другая комбинация в таблице запрещена и завершается с аварийным прекращением работы.

Всем целевым устройствам требуется проверять линии AD[1::0] в течение транзакции команды при обращении к памяти, а также обеспечивать требуемую последовательность передаваемого блока, либо прерывать связь с целевым устройством в первую фазу данных или по ее завершении. Реализация линейного порядка передаваемого блока требуется всеми устройствами, которые его не могут обеспечить. Выполнение переключения строки кэша при этом не требуется. В адресном пространстве памяти запросы дешифрируются в адрес DWORD с использованием линий AD[31::02]. В линейном режиме приращения инкрементирование адреса осуществляется путем инкрементирования двойного слова DWORD (четыре байта) после каждой фазы данных, до тех пор, пока не завершится транзакция.

[4] Стандартные значения адресов в пространстве ввода - вывода PC таковы, что разные физические устройства могут совместно использовать одни и те же адреса DWORD. В некоторых случаях это означает, что для установления запроса устройству требуется полный адрес байта (установленный в активное состояние сигнала DEVSEL#).

При использовании команд для работы с памятью, AD[1::0] принимает следующие значения:

AD1	AD0	Режим в блоке
	0	Линейное приращение
	1	Режим переключения строки кэша (порядок заполнения кэша - как у Intel486™/Pentium™/и т.д.)
	X	Зарезервировано (разрыв связи после первой фазы данных)

Запросы для адресных пространств конфигураций дешифрируются в адрес DWORD, который используется на AD[7::2]. Когда команда дешифрирована, агент определяет целевое устройство запроса (активен сигнал DEVSEL#), активен IDSEL и на AD[1::0] присутствует "00". В противном случае агент игнорирует текущую транзакцию. Интерфейс определяет, что запрос конфигурации предназначен для устройства, находящегося перед ним, дешифрируя команду конфигурации и номер интерфейса, на линиях AD[1::0] - "01". За более подробными сведениями относительно запросов конфигурации обращайтесь к разделу 3.6.4.1.2.

3.2.3. Выравнивание байта

Переключение линий на PCI не выполняется до тех пор, пока все PCI - устройства не будут подсоединены к 32 битам адрес/данных с целью дешифрирования адреса. Тем не менее, переключение DWORD осуществляется мастерами, что обеспечивает для данных поддержку 64-разрядных линий передачи. Это означает, что байты будут всегда приходить по своему «родному» маршруту, который базируется на адресе байта.
В то же время, PCI никогда не обеспечивает автоматическое установление «размеров» шины. В общем случае, программное обеспечение очень чувствительно к характеристикам целевого устройства и поэтому выдает только соответствующую длину запроса; исключением являются 64-разрядные линии передачи данных.

Побайтовый доступ используется только для того, чтобы определить, какие байты несут значимые данные. Можно свободно изменять его во время фаз данных, но при этом данное состояние должно быть корректным по положительному фронту синхроимпульса, которым начинается каждая фаза данных, и оно должно остаться таким в течение всей фазы данных. Как показано на рисунке 3-1, фазы данных начинаются в тактах 3, 5 и 7. (Изменение режима побайтового доступа допускается во время транзакции блока данных, в общем случае это не требуется, хотя и разрешено). Мастер может изменять режим побайтового доступа при каждой новой фазе данных (хотя это не показано на диаграмме чтения). Если мастер изменяет его при транзакции чтения, то это делается с синхронизацией, как при транзакции записи. Если побайтовый доступ необходим для целевого устройства при транзакции чтения, то устройство должно подождать, когда наступит данное состояние в течение фазы данных, перед завершением передачи; в противном случае устройство должно вернуть все байты.

Если текущая транзакция чтения относится к кэшируемой памяти, то должны быть возвращены все байты, при условии, что осуществляется управление разрешением байта. Поэтому требуется агент, который определяет способность целевого устройства возвратить все байты. Если возможность кэширования определена самим инициатором, то он должен убедиться в том, что установлены все разрешенные, так как целевое устройство должно возвратить требуемые данные. Если возможность кэширования определена целевым устройством, то оно должно игнорировать разрешение байта (за исключением генерирования сигнала PAR) и вернуть все слова DWORD. Кэшируемое устройство должно также вернуть всю кэшируемую строку или только первые затребованные данные.
Целевое устройство не должно поддерживать кэширование, но должно обеспечивать предварительную выборку (установкой бита в регистре базового адреса - смотрите раздел 6.2.5.l.), а также должно возвращать все данные, независимо от состояния разрешения байта. В режиме, когда имеются побочные эффекты (разрушение данных или изменение состояния из - за появления запроса), целевое устройство может только управлять.

PCI допускает любую непрерывную или состоящую из нескольких несмежных участков комбинацию разрешенных байтов. Если не установлено разрешение байта, то запрашиваемое целевое устройство должно завершить транзакцию путем установки TRDY#, и обеспечивая контроль по четности, если это был запрос по чтению. Когда не допускается разрешение байта, запрашиваемое целевое устройство должно завершить текущую фазу данных без каких-то изменений. Для транзакции чтения это означает, что данные или состояние не изменяются. Если завершение запроса не оказывает воздействия на данные или состояние, то целевое устройство может завершать запрос с обеспечением данных либо без их обеспечения. Целевое устройство (при чтении) должно обеспечить контроль по четности для линий AD[31::0] и C/BE[3::0]#, независимо от состояния разрешения байта. При транзакции записи данные не сохраняются, и имеет значение сигнал PAR.

Тем не менее, некоторые целевые устройства могут быть неспособны правильно интерпретировать структуры, состоящие из нескольких несмежных участков (например, интерфейсы шины расширения как интерфейс 8- и l6-разрядных подчиненные устройства). Если это все же происходит, то целевое устройство (интерфейс шины расширения) может необязательно объявить запрещенную структуру как асинхронную ошибка (SERR#) или, по возможности, разделить транзакцию в две 16 - разрядные транзакции, которые являются допустимыми для предназначенного агента. При запросе на ввод - вывод, целевому устройству требуется сигнал, чтобы сообщить о своем аварийном прекращении работы в случае, если устройство неспособно завершить все запросы, определенные разрешающим байтом.

3.2.4. Управление шиной и оборотный цикл

Оборотный цикл требуется для всех сигналов, которые могут управляться более, чем одним агентом. Этот цикл также требуется, чтобы избежать конкуренции, когда один агент прекращает управление сигналом, а другой агент начинает. Цикл обозначен на диаграммах синхронизации как две стрелки, указывающие одна в хвост другой. Оборотный цикл происходит в разное время для различных сигналов. Например, сигналы IRDY#, TRDY#, DEVSEL#, STOP# и ACK64# используют фазу адреса в качестве своего оборотного цикла. FRAME#, REQ64#, C/BE[3::0]#, C/BE[7::4]#, AD[31::00] и AD[63::32] используют в качестве своего оборотного цикла цикл ожидания IDLE между транзакциями. Оборотный цикл для LOCK# занимает один такт после того, как его освобождает текущий владелец. Сигнал PERR# имеет оборотный цикл на четвертом такте после последней фазы данных, которая занимает три такта после оборотного цикла для адресных линий. Цикл ожидания наступает, когда не установлены оба сигнала FRAME# и IRDY# (например, 9-й такт на рисунке 3-1).

Все адресные линии (включая [63::32], в случае, если мастер поддерживает 64-разряда для линий данных) должны быть приведены в устойчивое состояние в течение каждой фазы адреса и фазы данных. Четные линии передачи байта, не участвующие в текущей передаче данных, должны передавать на шину одни и те же данные (хотя и не имеющие значение). Смысл этого заключается в вычислении суммы контроля по четности и пересылке входных буферов по линиям передачи байта, не участвующих в пересылках при переключении на пороговом уровне, а также для обеспечения быстрой метастабильной и свободной блокировки. В энерго - чувствительных приложениях, в целях уменьшения энергопотребления при переключениях шины, рекомендуется, чтобы линии передачи байта, не используемые в текущей фазе шины, управлялись с теми же самыми данными, что и в предыдущей фазе шины. В приложениях, которые энерго - нечувствительны, агент, управляющий линиями адреса, может управлять чем - угодно по неиспользуемым линиям передачи байта. Контроль по четности должен делаться для всех байтов, независимо от разрешения байта.

3.3. Транзакции на шине

На диаграммах синхронизации показана связь значимых сигналов, участвующих в 32-разрядных транзакциях. Если сигнал нарисован сплошной линией, то это означает, что он в данный момент управляется текущим мастером или целевым устройством. Если сигнал нарисован пунктирной линией, то им никто не управляет. Тем не менее, он все еще может принимать постоянное значение, пока пунктирная линия находится в верхнем положении. На рисунке также обозначены три - стабильные сигналы, которые имеют неопределенное значение, когда пунктирная линия находится между двумя положениями (например, линии адреса AD или строками C/BE#). Когда сплошная линия переходит в пунктирную, то это означает, что сигнал, который активно управлялся, теперь перешел в третье состояние. Когда сплошная линия переходит из нижнего положения в верхнее, а затем становится пунктирной, то показывает, что сигнал активно управлялся по высокому уровню до установки на шине, и затем перешел в третье состояние. Циклы до и после каждой транзакции будут обсуждены в разделе, посвященному арбитражу.

3.3.1. Транзакция чтения

Рисунок 3-1 показывает транзакцию чтения и начинается фазой адреса, которая происходит во 2-ом такте, когда впервые устанавливается сигнал FRAME#. В течение фазы адреса AD[31::00] содержит допустимый адрес, а C/BE[3::0]# - допустимую команду шины.

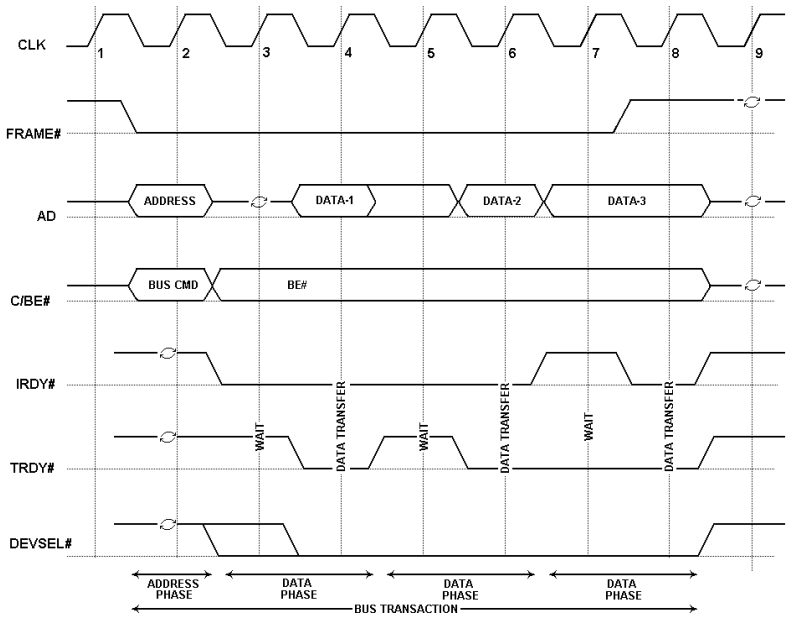


Рисунок 3-1: Базовая операция чтения

Такт 3 является первым тактом для первой фазы данных. В течение фазы данных C/BE# показывает, какие линии передачи байта участвуют в текущей фазе данных. Фаза данных может состоять из циклов передачи данных и циклов ожидания. Выходные буферы C/BE# должны оставаться доступными (и для чтения, и для записи) от первого такта фазы данных до конца транзакции. Необходимо убедиться, что линии C/BE# не будут изменяться для длинных интервалов времени.

При транзакции чтения первой фазе данных требуется оборотный цикл (инициированный целевым устройством через сигнал TRDY#). В этом случае адрес будет достоверен во 2-ом такте, а затем мастер завершит управление адресными линиями. Самый ранний такт, в котором целевое устройство может гарантировать достоверные данные - это такт 4. Целевое устройство должно управлять

адресными линиями после оборотного цикла, когда установлен сигнал DEVSEL#. По возможности, буферы вывода должны оставаться доступными до конца транзакции. (Это гарантирует, адресные линии не будут изменяться в течение длительного времени).
Фаза данных завершается, когда данные перемещены, при этом сигналы IRDY# и TRDY# установлены на том же самом фронте синхроимпульса (сигналом TRDY# нельзя управлять, пока установлен DEVSEL#). Если состояние неактивное, то вставляется цикл ожидания, и не передаются никакие данные. Как отмечено в диаграмме, данные успешно пересылаются в тактах 4, 6 и 8, а циклы ожидания вставлены в такты 3, 5 и 7. Первая фаза данных завершается для транзакции чтения за минимальное время. Вторая фаза данных расширяется в 5-ом такте, так как неактивен сигнал TRDY#. Последняя же фаза данных расширена, из-за того, что IRDY# был неактивен в 7-ом такте.
В 7-ом такте мастер знает, что следующая фаза данных - последняя. Тем не менее, сигнал FRAME# остается активным, так как мастер не готов завершить последнюю пересылку (IRDY# в такте 7 - неактивен). Только тогда, когда IRDY# станет активным, FRAME# может перейти в неактивное состояние, что и происходит в такте 8.

3.3.2. Транзакция записи

На рисунке 3-2 показана транзакция записи. Транзакция начинается в такте 2, когда впервые устанавливается в активное состояние сигнал FRAME#. Транзакция записи подобна транзакции чтения, за исключением того, что после фазы адреса не требуется оборотный цикл, так мастер обеспечивает и адрес, и данные. Фазы данных для транзакции записи такие, как и для транзакции чтения.

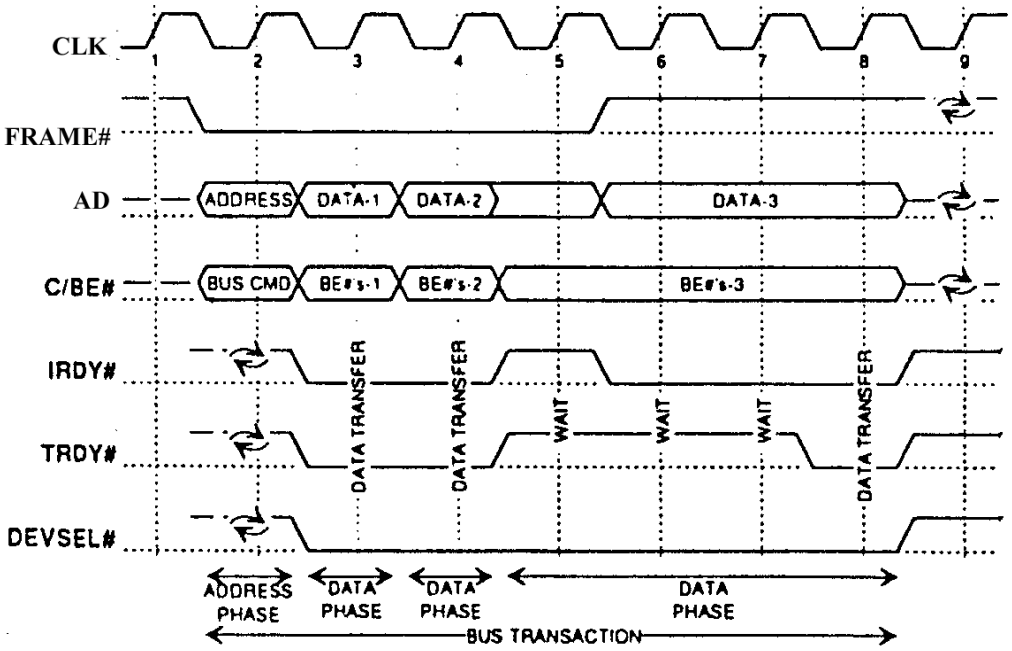


Рисунок 3-2: Базовая операция записи

На рисунке 3-2 первая и вторая фазы данных завершаются нулевыми циклами ожидания. Однако третья фаза данных имеет три цикла ожидания, вставленных целевым устройством.

Обратите внимание, что агент вставляет оба цикла ожидания в такте 5. IRDY# должен быть активен, когда FRAME# неактивный, показывая, таким образом, последнюю фазу данных. В такте 5 передача данных была отсрочена мастером, так как был неактивный сигнал IRDY#. Хотя это позволяет мастеру задержать данные, задержка разрешение байта запрещена. Последняя фаза данных указывается мастером в такте 6, но она не завершается до 8-го такта.

3.3.3. Завершение транзакции

Окончание транзакции PCI может быть инициировано как мастером, так и целевым устройством. Остановить транзакцию односторонне фактически нельзя, поэтому мастер постепенно завершает управление, приводя все транзакции к упорядоченному и систематическому выводу, чем вызывается само завершение. Все транзакции завершаются, когда оба сигнала FRAME# и IRDY# неактивны, показывая таким образом цикл ожидания (например, такт 9 на рисунке 3-2).

3.3.3.1. Завершение транзакции, инициированное мастером

Механизм, используемый мастером, инициирует завершение транзакции, когда сигнал FRAME# - неактивный, а IRDY# - активный. Это сигнализирует целевому устройству, что идет заключительная фаза данных. Новая передача данных происходит, когда активны оба сигнала IRDY# и TRDY#. Транзакция завершается, когда оба сигнала FRAME# и IRDY# неактивны (шина при этом ожидает завершения транзакции).

Мастер может инициировать завершение, использующее этот механизм, по одной из следующих двух причин:

Завершение

Завершение инициируется, когда мастер завершает соответствующую транзакцию. Это - наиболее типичная причина для завершения транзакции.

Тайм - аут

Ведет к завершению, когда сигнал GNT# мастера - неактивный, и истекло время ожидания по внутреннему таймеру. Соответствующая транзакция не обязательно завершается. Время может закончиться из-за задержки ожидания целевого устройства, либо из-за того, что соответствующая операция очень длинная. За описанием работы с внутренним таймером обращайтесь к разделу 3.4.4.1.

Таймер времени ожидания не управляет транзакцией записи в память и отмены (Memory Write and Invalidate). Мастер, который инициирует транзакцию командой Memory Write and Invalidate, игнорирует таймер времени ожидания, пока не будет достигнута граница строки кэша. Когда же это произойдет, и истечет время ожидания (и сигнал GNT# - неактивный), мастер должен завершить транзакцию. Если транзакция Memory Write and Invalidate завершается целевым устройством, то мастер завершает транзакцию (остальная часть строки кэша), как только это станет возможным (с соблюдением по протоколу сигнала STOP#), используя команду записи в память Memory Write (так как условия, при которых можно выдать команду Memory Write and Invalidate, станут недействительны).

Модифицированная версия этого механизма завершения позволяет мастеру завершать транзакцию, когда никакое целевое устройство не отвечает. Такое аварийное завершение называется завершением, инициированным мастером (*master-initiated*). Хотя это может вызывать фатальную

ошибку для приложения, изначально запросившего транзакцию, транзакция завершается элегантно, поддерживая таким образом нормальную работу PCI для других агентов.

На рисунке 3-3 показаны два примера нормального завершения транзакции. Заключительная передача данных происходит, когда неактивен сигнал FRAME#, и активны оба сигнала IRDY# и TRDY#, а это происходит в такте 3. Условие ожидания на шине вступает в силу, когда сигнал IRDY# - неактивный, это происходит в такте 4. Так как транзакция завершилась, то в такте 4 также неактивен TRDY#. Обратите внимание, что в такте 3 не требуется, чтобы TRDY# был активным, но тогда должна обеспечиваться задержка заключительной передачи данных (и задержка окончания транзакции), пока не будет обеспечена готовность путем отсрочки заключительной установки сигнала TRDY#. Если же это сделает целевое устройство, то мастеру потребуется сохранять IRDY# активным, пока не завершится заключительная передача данных.

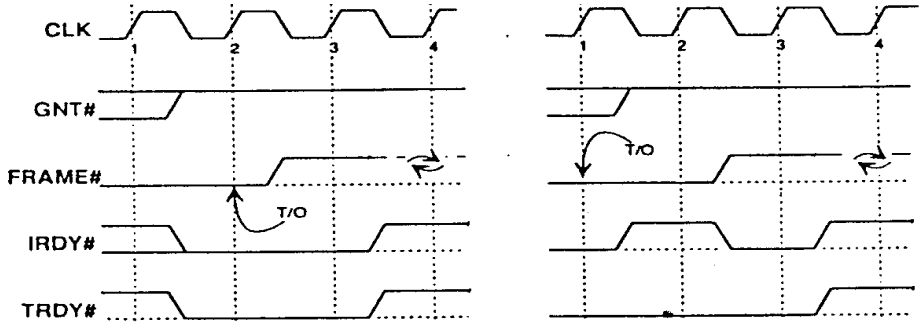


Рисунок 3-3: Завершение транзакции, инициированное мастером

На обеих частях рисунка 3-3 ситуация могла бы завершиться тайм - аутом. Как показано на левой части рисунка, сигнал FRAME# неактивен в такте 3, так как истекает время по тайм-ауту. GNT# переходит в неактивное состояние, и мастер становится готов (IRDY# активен) к заключительной передаче. Так как GNT# неактивен по истечении времени, то дальнейшее использование шины не разрешается, за исключением случая использования команды Memory Write and Invalidate, которая должна завершиться при достижении границы строки кэша. Далее завершение происходит нормальным образом. Если TRDY# - неактивен в такте 2, то фаза данных будет продолжаться, пока он не станет активным. FRAME# и IRDY# должны оставаться активными, пока не завершится фаза данных.
На правой части рисунка показана ситуация, когда время истекает в такте 1. Так как мастер не готов к передаче данных (IRDY# перешел в неактивное состояние в такте 2), то требуется, чтобы сигнал FRAME# оставался активным. FRAME# - неактивный в такте 3, так как мастер готов (IRDY# - активный) к завершению транзакции в такте 3. Мастер должен управлять достоверными данными (при записи), либо быть способным получить данные (при чтении) всякий раз, когда активен IRDY#. Данная задержка завершения не должна быть больше 2-х или 3-х циклов. Также обратите внимание, что транзакция не должна завершаться по истечении времени, если только GNT# не перейдет в неактивное состояние.

Аварийное прекращение работы мастера, как показано в рисунке 3-4, является аварийным случаем (кроме случая конфигурации или команд специального цикла) завершения, инициированного мастером. Мастер определяет, что если не будет никаких запросов на транзакцию, то DEVSEL# останется неактивным в такте 6 (за полным описанием работы с сигналом DEVSEL# обращайтесь к разделу 3.6.1.). Мастер должен установить, что либо целевое устройство, подавшее запрос, неспособно к работе с запрошенной транзакцией, либо адрес был неправильным. Как только мастер обнаружил отсутствие DEVSEL# (такт 6 в данном примере), сигнал FRAME# переходит в неактивное состояние в такте 7, а IRDY# - в такте 8. Самое начальный момент, когда мастер может завершать транзакцию с аварийным прекращением своей работы - это 5 тактов после того, как впервые был установлен сигнал FRAME#, а это происходит, когда мастер делает попытку

одиночной передачи данных. Однако мастеру может потребоваться больше времени для перевода сигнала FRAME# в неактивное состояние и завершения запроса. Мастер должен поддерживать связь FRAME# - IRDY# на всех транзакциях, которые включают в себя аварийное прекращение работы мастера. FRAME# не может перейти в неактивное состояние прежде, чем IRDY# станет активным, а IRDY# должен остаться активным по крайней мере один такт после того, как FRAME# станет неактивным, даже когда транзакция заканчивается с аварийным прекращением работы мастера.
В качестве альтернативы, IRDY# мог бы стать неактивным в такте 7, при условии, что FRAME# также был неактивным, как в случае транзакции с одной фазой данных. В нормальной ситуации мастер не будет повторять данный запрос (смотрите раздел 3.7.2.2.). Обратите внимание, что если DEVSEL# был активным в тактах 3, 4, 5 или 6 в этом примере, то это показывает, что запрос был подтвержден событием, и завершение с аварийным прекращением работы мастера недопустимо.

Главный интерфейс шины, в PC - совместимых системах, должен возвратить все единицы при транзакции чтения и отменить данные при транзакции записи, когда завершение происходит с аварийным прекращением работы мастера. Интерфейс требуется для установления бита аварийного прекращения работы мастера в регистре состояния. Другие мастер - устройства могут сообщать об этом состоянии как об ошибке, подачей сигнала SERR#, когда мастер не может сообщить об ошибке через драйвер устройства. Интерфейс PCI - PCI должен обеспечивать совместимость для PC, которая описывается для главного интерфейса шины. Когда интерфейс PCI - PCI используется в другой системе, то он ведет себя аналогично с ее мастер - устройствами. Предварительная выборка интерфейсом данных при чтении должна быть полностью понятна для системы. Это означает, что когда предварительная транзакция завершена с аварийным прекращением работы мастера, то интерфейс должен просто остановить транзакцию и продолжать нормальную работу без сообщения. Это происходит, когда транзакция не запрашивается целевым устройством.

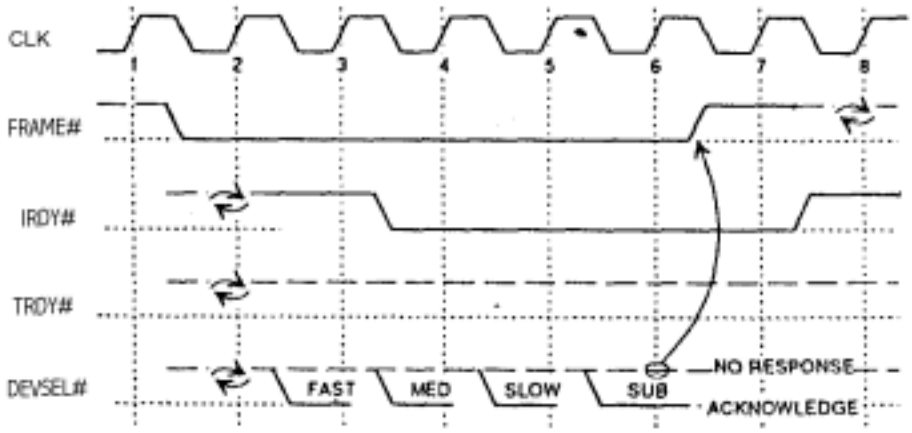


Рисунок 3-4: Завершение с аварийном прекращением работы мастера

Подводя итог, можно перечислить следующие общие правила управления сигналами FRAME# и IRDY# во всех транзакциях PCI.

1. FRAME# и соответствующий IRDY# определяют состояние шины занято/ожидание (BUSY/IDLE); когда оба сигнала активные, шина занята; когда оба - неактивные, шина находится в ожидании.
2. Как только FRAME# стал неактивным, то он не может быть переустановлен в течение этой же самой транзакции.

3. FRAME# не может быть установлен в неактивное состояние, пока IRDY# - неактивный (IRDY# должен всегда устанавливаться на первом фронте синхроимпульса, когда сигнал FRAME# становится неактивным).
4. Как только мастер установил сигнал IRDY#, он не должен изменять сигналы IRDY# или FRAME#, пока не завершится текущая фаза данных.

3.3.3.2. Завершение транзакции, инициированное целевым устройством

Механизм, используемый при завершении транзакции, инициированном целевым устройством, использует STOP#. Целевое устройство устанавливает STOP# для запроса к мастеру, чтобы он завершил транзакцию. Если STOP# однажды уже был установлен, то он остается активным, пока не станет неактивным сигнал FRAME#. Связь между сигналами IRDY# и TRDY# не зависит от связи между STOP# и FRAME#. Это означает, что во время запроса целевым устройством завершения транзакции могут передаваться (но необязательно) данные; это зависит только от состояния сигналов IRDY# и TRDY#. Тем не менее, когда STOP# - активный, а TRDY# - неактивный, то показывает, что целевое устройство больше не будет передавать данные, поэтому мастер не будет ожидать заключительной передачи данных, как это было бы при завершающем прекращении транзакции.
Целевое устройство может инициализировать завершение, используя этот механизм, по одной из следующих двух причин:

Повтор — ведет к завершению транзакции, так как целевое устройство находится в состоянии, которое не позволяет ему начать транзакцию. Данная ситуация может включать в себя состояние тупика, некоторые условия занятости не - PCI устройств, либо условие блокировки эксклюзивного запроса. Повтор означает, что целевое устройство завершает транзакцию, и никакие данные не были переданы.

Разрыв связи — ведет к завершению транзакции, так как целевое устройство неспособно ответить в пределах задержки, установленной, установленной для PCI (она равна 8 тактам). Обратите внимание, что это действие нетипично для первой фазы данных (смотрите раздел 3.4.4.1.). Разрыв связи означает, что целевое устройство прекращает транзакцию, когда данные получены, либо во время их получения. Кэшируемые целевые устройства не должны разрывать связь для команды Memory Write and Invalidate, за исключением случая, когда достигнута граница строки кэша, если осуществляется кэширование. Следовательно, заинтересованный агент может всегда предполагать, что команда Memory Write and Invalidate завершится без разрыва связи, в случае, когда осуществляется запрос с обращением в диапазон кэшируемой памяти.

Модифицированная версия этого механизма позволяет целевому устройству завершать транзакцию, в которой произошла фатальная ошибка, или если на эту транзакцию целевое устройство не будет способно ответить. Такое аварийное завершение называется как завершение, инициированное целевым устройством (target - abort). Хотя это может вызвать фатальную ошибку для приложения, первоначально запросившего транзакцию, транзакция завершается элегантно, таким образом сохраняется нормальная работа PCI для других агентов.
Большинству целевых устройств может потребоваться возможность повтора, но все остальные случаи инициированного целевым устройством завершения транзакции необязательны для таких устройств. Мастер - устройства должны обладать способностью к вызову всех этих функций. Повтор также необязателен для очень простых целевых устройств, которые: 1) не поддерживают исключительные запросы (с блокировкой), 2) не могут обнаружить возможную тупиковую ситуацию и 3) не могут войти в состояние, в котором они им потребовалось бы отклонить запрос.

Три примера разрыва связи показаны на рисунке 3-5. Каждый пример отражает одну и ту же связь между сигналами STOP# и FRAME#, а именно:

Разрыв связи наблюдается, когда активен STOP#, и он остается таким, пока FRAME# не станет неактивным.

Сигнал FRAME# становится неактивным как можно раньше, после того, как становится активным STOP#. Пример С показывает появление внешнего цикла, из-за того, что IRDY# не мог стать активным сразу после того, как сигнал STOP# был установлен в активное состояние.

STOP# завершает цикл сразу после того, как FRAME# перешел в неактивное состояние.

Кроме того, данные три примера разрыва связи показывают, что DEVSEL# всегда активный, когда активен STOP#; в противном случае наблюдается аварийное прекращение работы целевого устройства.

Эти три примера также показывают три различные возможности для передачи данных при разрыве связи. Обратите внимание, что целевое устройство может определять, переданы данные или нет, после того, как STOP# стал активным. Передача данных происходит в каждом цикле, в котором активны сигналы IRDY# и TRDY#, независимо от состояния STOP#. Если целевое устройство хочет осуществить еще одну передачу данных, а затем остановиться, то оно в это же самое время устанавливает в активное состояние TRDY# и STOP#.

На рисунке 3-5, *примеры А и В* показывают два разных способа разрыва связи, когда данные передаются после установления в активное состояние сигнала STOP#. В обоих случаях, целевое устройство заявляет о своем намерении осуществить еще одну передачу данных при наличии активного TRDY# , когда активен сигнал STOP#. В примере А данные передаются того, как FRAME# стал неактивным (в такте 3), так как мастер не был к этому готов (IRDY# перешел в неактивное состояние в такте 2).

В примере В данные передаются перед установкой FRAME# в неактивное состояние (в такте 2). Если TRDY# был активным, когда активным был и сигнал STOP#, то TRDY# должен перейти в неактивное состояние, когда завершится текущая фаза данных. Целевое устройство не может установить в неактивное состояние STOP# и продолжать транзакцию. Мастер перезапускает любую транзакцию, завершившуюся с повтором либо разрывом связи, после запуска синхроимпульсов с адресами следующих, не переданных, данных. После того, как данные будут переданы, целевое устройство уберет сигнал TRDY#, так как предполагается, что данные больше не будут передаваться. Обратите внимание, что в заключительной фазе данные не передаются. Если целевое устройство сохраняло TRDY# активным в течение такта 3, и задержало активизацию сигнала STOP# до такта 3, после чего данные будут передаваться в тактах 2 и 3. Тем не менее, целевое устройство не может завершить более, чем одну передачу данных после того, как устанавливается в активное состояние сигнал STOP#, как это показано в примере А.

Будучи один раз установленным, сигнал STOP# должен остаться активным, пока не станет неактивным сигнал FRAME#. Если целевому устройству требуется цикла ожидания в последней фазе данных, то оно должно отсрочить переход STOP# в активное состояние, пока устройство не станет готово завершить транзакцию.

Пример С показывает случай, в котором после перехода STOP# в активное состояние данные не передаются, так как сигнал TRDY# - неактивный. Обратите внимание, что в этом примере, установка сигнала FRAME# в неактивное состояние отсрочена, до перехода IRDY# в активное состояние. Этот пример показывает повтор, который является фактически частным случаем, разрыва связи, когда вообще не происходит никакой передачи данных. Обобщенный пример повтора - это когда целевое устройство в текущий момент времени блокировано для монопольного запроса со стороны другого мастера. Другой пример - это когда целевое устройство должно осуществить запрос к некоторым другим не - PCI ресурсам перед разрешением транзакции (в этом случае требуется соблюдать осторожность, чтобы убедиться, что какие-либо условия, когда повтор непосредственно вел бы к порождению тупикового состояния). Когда текущая транзакция завершена целевым

устройством, мастер должен установить в неактивное состояние свой сигнал REQ#. Мастер должен это сделать за минимальное время из двух тактов PCI; один такт идет, когда шина переходит в состояние ожидания (в конце транзакции, когда активен STOP#), и еще один такт - перед или после состояния ожидания. Если мастер предполагает завершать транзакцию, то должен подтвердить это установкой REQ# немедленно после двух тактов, когда он перешел в неактивное состояние, или если появилось возможное состояние "зависания". Если мастер не предполагает завершать транзакцию (так как это могла быть выборка, или должен быть обслужен более приоритетный внутренний запрос), то агент только устанавливает в активное состояние REQ# всякий раз, когда требуется вновь использовать интерфейс.

Нижний правый пример на рисунке 3-5 показывает завершение транзакции по инициативе целевого устройства, а это происходит при активном STOP# и неактивном DEVSEL#. Таким образом показывается, что целевое устройство требует завершения транзакции, и что оно не желает ее повтора. К тому же, если в текущей транзакции были переданы какие-то данные, то они могут быть разрушены (смотрите раздел 3.7.2.2.). DEVSEL# должен быть активен в течение одного или большего количества временных циклов, а TRDY# должно стать неактивным прежде, чем может произойти аварийное прекращение работы, инициированное целевым устройством.

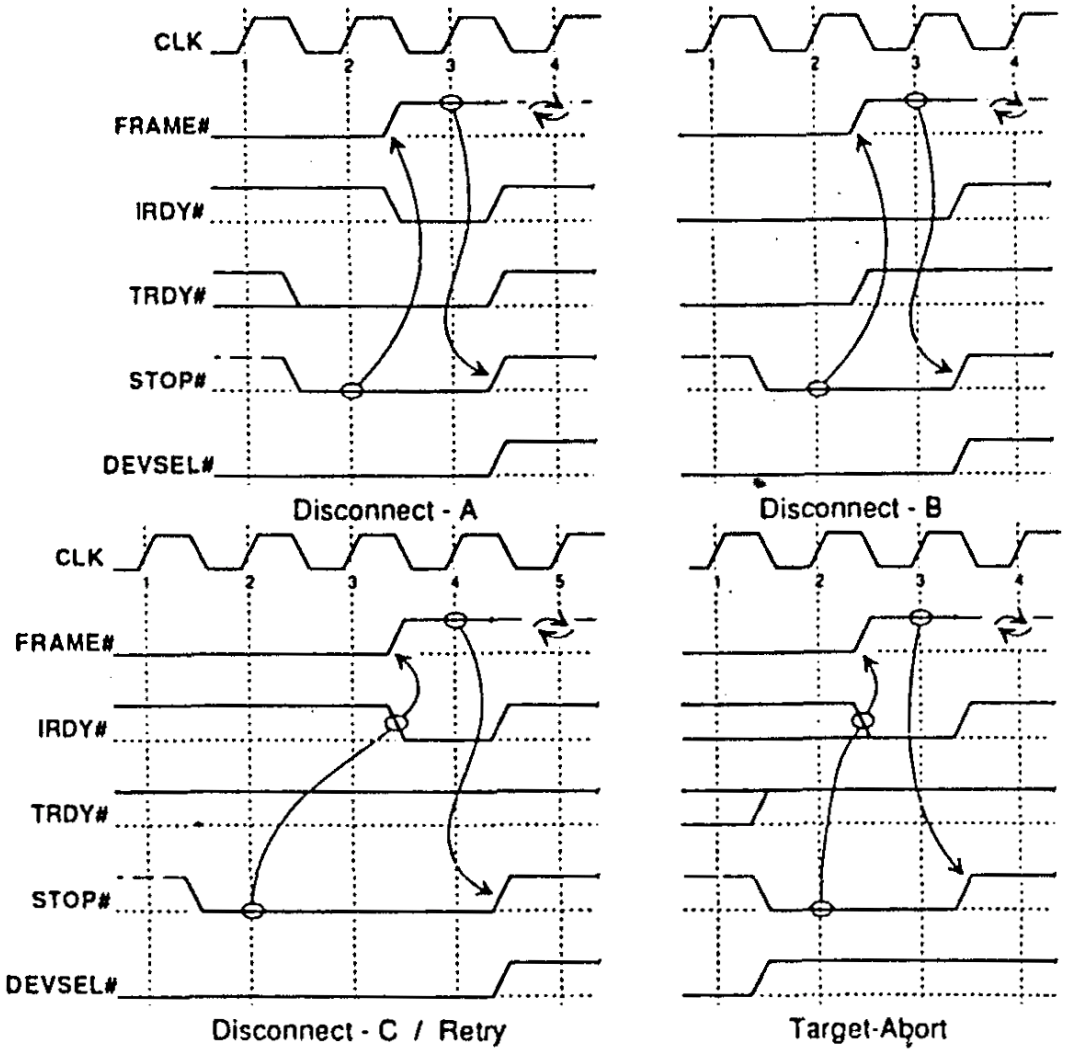


Рисунок 3-5: Завершение транзакции, инициированное целевым устройством

В заключение, перечислим следующие общие правила при управлении сигналами FRAME#, IRDY#, TRDY# и STOP# во всех транзакциях PCI:

1. Всякий раз, когда сигнал STOP# становится активным, FRAME# должен перейти в неактивное состояние как можно скорее, в соответствии с правилами такого установления (то есть IRDY# должен быть активным). Сигнал FRAME# должен стать активным после перехода STOP# в неактивное состояние как можно скорее, предпочтительно за два или три цикла. Целевое устройство должно игнорировать любые попытки установить связь в интервалах времени между переходом сигнала STOP# в активное состояние, а сигнала FRAME# - в неактивное но при этом оно должно сохранять активным сигнал STOP#, пока FRAME# не станет неактивным. Когда мастер обнаружил, что активен STOP#, он должен установить в неактивное состояние сигнал FRAME# в первом цикле, начиная с которого стал активным IRDY#. Установка в активное состояние IRDY# (а, следовательно, установка FRAME# в неактивное состояние) может произойти как следствие нормального поведения сигнала IRDY# мастера (при условии, что транзакция не была прервана целевым устройством), и может быть задержана нулевым или большим количеством циклов, в зависимости от того, когда мастер будет готов завершить передачу данных. В качестве альтернативы, мастер может установить сигнал IRDY# в активное состояние немедленно (даже без подготовки к завершению передачи данных), при условии, что TRDY# - неактивный, таким образом показывая, что больше не будет никаких дальнейших передач данных.
2. Будучи установленным, STOP# должен остаться активным до тех пор, пока FRAME# не перейдет в неактивное состояние, после чего STOP# также должен стать неактивным.
3. В течение заключительной фазы данных транзакции (FRAME# неактивный и IRDY# активен), любой фронт синхроимпульса, на котором сигналы STOP# и TRDY# переходят в активное состояние, становится последним циклом транзакции, а сигнал IRDY# становится неактивным по следующему положительному фронту (таким образом, создается цикл ожидания и определяется конец транзакции).
4. Мастер должен повторить запрос, который был завершен целевым устройством (за исключением аварийного прекращения работы целевым устройством) по адресам следующих еще не переданных данных, если он, конечно, предполагает полностью завершить запрос. Если устройство не желает, то мастер может не повторять запрос.
5. Если целевое устройство установило в активное состояние сигналы TRDY# или STOP#, то оно не должно изменять DEVSEL#, TRDY# или STOP# до тех пор, пока не завершится текущая фаза данных.

3.4. Арбитраж

Для того, чтобы уменьшить время ожидания запроса, при арбитраже PCI используется такой подход: запрос должен завершаться за определенное время. Это означает, что мастер шины должен осуществлять арбитраж для каждого запроса, который выполняется на шине. PCI использует центральную схему арбитража, где каждый агент мастера имеет уникальный сигнал запроса (REQ#) и сигнал разрешения (GNT#). Для получения доступа к шине используется простой прием «рукопожатие», типа запрос - ответ. Арбитраж на шине - "скрытый", это означает, что он осуществляется в течение предыдущего запроса таким образом, чтобы для арбитража не использовались никакие циклы шины PCI, за исключением случая, когда шина находится в состоянии ожидания.

Указанный алгоритм арбитража должен выполняться центральным арбитром, например, при вращении приоритета, точно и т.д. При определении алгоритма арбитража следует установить значение для гарантирования заданного времени ожидания в наихудшем случае. Тем не менее, так как алгоритм арбитража не является частью спецификации шины, то проектировщики системы могут по выбору модифицировать его, но при этом они должны обеспечить соблюдение требований для времени ожидания выбранных ими контроллеров ввода - вывода, а также для плат расширения. За информацией относительно значений для времени задержки обращайтесь к разделу 3.4.4.3. Шина позволяет агенту осуществлять back-to-back транзакции, а также позволяет арбитру определенную гибкость в распределении запросов по приоритетам и весам. Арбитр может реализовывать схему любой сложности, при этом в любом такте должен быть активен только один сигнал GNT#.

3.4.1. Протокол сигналов арбитража

Агент запрашивает шину путем установки сигнала REQ#. Агенты могут использовать сигнал REQ#, чтобы указать, что им действительно требуется использовать шину. Агент никогда не должен использовать сигнал REQ#, чтобы "закрепить" себя на шине. Если же закрепление выполнено, то арбитр должен определить мастера по умолчанию. Когда арбитр решает, что агент может использовать шину, то он устанавливает в активное состояние сигнал GNT# агента.

Арбитр может установить в неактивное состояние сигнал GNT# агента в любом такте. Если агент желает начать транзакцию, он должен убедиться, что GNT# активен по положительному фронту синхроимпульса. Если GNT# неактивный, то транзакция не должна продолжаться. Если GNT# был установлен в активное состояние, то он может перейти в неактивное состояние в соответствии со следующими правилами.

1. Если GNT# - неактивный, и активен FRAME#, то транзакция шины допустима и продолжается.
2. Один сигнал GNT# может быть установлен в неактивное состояние, когда соответствующий сигнал GNT# активен, причем шина не должна находиться в состоянии ожидания. В противном случае, потребуется один такт задержки в период, когда GNT# устанавливается в неактивное состояние, а следующий сигнал GNT# переходит в активное состояние, либо если будет конкуренция на адресных линиях AD и линии PAR.
3. Пока сигнал FRAME# - неактивный, GNT# может быть установлен в неактивное состояние в любой момент времени в порядке обслуживания высокоприоритетным[5] мастером, или при отклике на соответствующий сигнал REQ#, который перешел в неактивное состояние.

Рисунок 3-6 иллюстрирует основной арбитраж. Используются два агента, чтобы показать, как арбитр может чередовать запросы шины.

[5] Высокий приоритет здесь не подразумевает фиксированный приоритетный арбитраж, это относится к агенту, который бы своевременно выиграл арбитраж в установленное время.

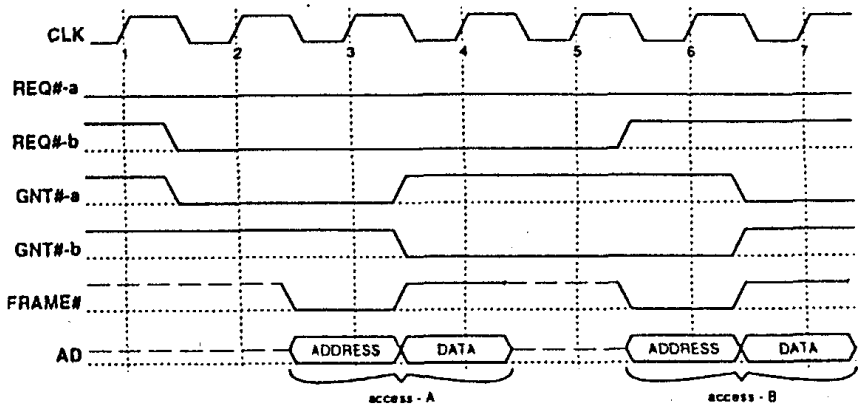


Рисунок 3-6: Основной арбитраж

REQ# активен до такта 1 или в течение его, если поступил запрос на использование интерфейса. Агент A получает разрешение к шине, так как сигнал GNT#-a устанавливается в активное состояние в такте 2. Агент A может начать транзакцию в такте 2, так как сигналы FRAME# и IRDY# были установлены в неактивное состояние, а сигнал GNT# установлен в активное состояние. Транзакция агента A начинается, когда FRAME# устанавливается в активное состояние в такте 3. Так как агент A решает выполнить другую транзакцию, то он оставляет сигнал запроса REQ#-a активным.
Когда сигнал FRAME# активен в такте 3, арбитр определяет, что далее следует агент B, и поэтому устанавливает в такте 4 сигнал GNT#-b - в активное состояние, а сигнал GNT#-a - в неактивное.
Шина освобождается, когда агент A завершает свою транзакцию в такте 4. Все агенты PCI могут определить конец текущей транзакции по неактивным сигналам FRAME# и IRDY#. Агент B захватывает шину в такте 5 (так как FRAME# и IRDY# - неактивные) и завершает свою транзакцию в такте 7.

Обратите внимание, что неактивный REQ#-b и активный FRAME# в такте 6 показывают, что агенту B требуется единственная транзакция. Арбитр разрешает следующую транзакцию агенту A, так как REQ# все еще активен.

Текущий мастер шины сохраняет REQ# активным, когда ему требуются дополнительные транзакции. Если больше нет никаких других активных запросы, или текущий мастер имеет самый высокий приоритет, арбитр продолжает предоставлять шину этому текущему мастеру.

Сигнал GNT# необходим агенту для запроса на шине одиночной транзакции. Если агент желает сделать другой запрос, он должен сохранять активным сигнал REQ#. Агент может установить сигнал REQ# в неактивное состояние в любой момент времени, но арбитр может интерпретировать это, как ситуацию, в которой агенту больше не требуется использования шины, и установить в неактивное состояние сигнал GNT#. Агент должен установить REQ# в неактивное состояние в том же самом такте, когда активен FRAME#, при условии, что агенту требуется выполнить одиночную транзакцию. Когда транзакция завершена целевым устройством (активен STOP#), мастер должен установить в неактивное состояние REQ# менее, чем за два такта PCI, первый из них - когда шина переходит в состояние ожидания (в конце транзакции, когда становится активным сигнал STOP#) , и еще один - на такт раньше или позже, после состояния ожидания. Если мастер предполагает завершать транзакцию, то должен еще раз установить сигнал REQ# после того, как REQ# перейдет в неактивное состояние, или может произойти потенциальное состояние "зависания". Если мастер не предполагает завершение (так как была предвыборка, или должен быть обслужен более приоритетный внутренний запрос), агент должен устанавливать REQ# всякий раз, когда ему может потребоваться использование интерфейса. Это позволяет другому агенту использовать интерфейс, пока предыдущее целевое устройство готовится к следующему запросу.

Арбитр может принять решение, что текущий мастер «разрушен», если тот не начал запрос после того, как был установлен в активное состояние сигнал GNT# (REQ# также активен), и шина находилась в ожидании 16 тактов PCI. Тем не менее, арбитр может убрать сигнал GNT# в любое время, для обслуживания более приоритетного агента.

3.4.2. Быстрые back-to-back транзакции

Имеются два типа быстрых back-to-back транзакций, которые могут быть инициированы тем же самым мастером, который осуществляет запрос агент, обратившегося к нему. Быстрые back-to-back транзакции разрешаются на шине PCI, когда надо избежать конкуренции для сигналов TRDY#, DEVSEL# или STOP#. Первый тип быстрой back-to-back транзакции возлагает ответственность за устранение конкуренции на мастере, в то время как второй возлагает эту ответственность на все потенциальные целевые устройства. Мастер может удалять цикл ожидания между транзакциями, если это будет гарантировать отсутствие конкуренции. Данное условие может быть выполнено, когда вторая транзакция мастера производится для того же самого целевого устройства, которое было в первый раз, и сама транзакция - это транзакция записи. Для этого типа быстрой back-to-back транзакции требуется, чтобы мастер знал границы адресов потенциального целевого устройства, иначе может происходить конкуренция. Этот тип быстрых back-to-back транзакций необязателен для мастера, но должен дешифрироваться целевым устройством.

Второй тип быстрой back-to-back транзакции возлагает ответственность за отсутствие какой-либо конкуренции на все потенциальные целевые устройства. Бит возможности быстрой back-to-back транзакции в регистре статуса может быть аппаратно установлен в логическую единицу (высокий уровень), если только устройство, которое является целевым устройством шины, отвечает следующим двум требованиям:

1. Целевое устройство не должно пропустить начало транзакции шины или потерять адрес, когда данная транзакция начинается без состояния ожидания на шине после предшествующей транзакции. Другими словами, целевое устройство должно обладать способностью «следовать» за состоянием шины, от заключительной передачи данных (высокий FRAME#, низкий IRDY#) непосредственно к фазе адреса (низкий FRAME#, высокий IRDY#) при последовательных циклических тактах. Обратите внимание, что хотя можно выбирать целевое устройство (а можно и не выбирать) на любой из этих транзакций или на обеих сразу, тем не менее целевое устройство должно отслеживать состояния шины[6] .
2. Целевое устройство должна избегать конфликтов для сигналов DEVSEL#, TRDY# и STOP#. Если целевое устройство не реализует самое минимальное, по возможности, время установления DEVSEL# в активное состояние, то это уже гарантируется. Если целевое устройство реализует состояние дешифрирования с нулевым временем ожидания, то оно должно отсрочить установление в активное состояние этих трех сигналов на один такт, за исключением случая, когда выполняется любое из следующих условий:
   a. Текущей транзакции шины предшествовало состояние ожидания. То есть это - не back-to-back транзакция, или
   b. Текущее целевое устройство управляло сигналом DEVSEL# на предыдущей транзакции шины. То есть это - back-to-back транзакция, предусматривающая то же самое целевое устройство, что и в предыдущей транзакции.

Для мастеров, которые хотят выполнять быстрые back-to-back транзакции, обеспечиваемые механизмом целевого устройства, требуется бит *Fast Back-to-Back Enable* в регистре команд (этот бит имеет значение только для устройств, которые функционируют как владельцы шины, и полностью необязателен). Данный бит реализуется в качестве записываемого/считываемого бита. Когда он установлен в единицу (высокий) уровень, мастер шины может начинать транзакцию PCI, используя back-to-back синхронизацию без указания адреса целевого устройства, взяв его из предыдущей транзакции записи, осуществленной текущим мастером шины.

[6] Это рекомендуется выполнять путем возвращения механизма состояний целевого устройства (см. приложение B) из состояния B_BUSY в состояние ожидания так скоро, как только возможно, поскольку FRAME# - неактивный, и отсутствует ожидание на шине состояния IDLE (IRDY# - неактивный).

Если этот бит установлен в ноль (низкий уровень) или вообще не установлен, мастер может выполнять быстрые back-to-back транзакции только, если будет гарантировано, что новая транзакция будет осуществляться к тем же самым целевым устройствам, что и в предыдущий раз (механизм, основанный на мастере).

Данный бит можно установить подпрограммой конфигурации системы, после того, как будет гарантировано, что все целевые устройства на этой шине имеют установленный бит возможности быстрой back-to-back транзакции (Fast Back-to-Back Capable Bit).

Обратите внимание, что механизм быстрых back-to-back транзакций, основанный на мастере, не позволяет осуществлять такие «быстрые» циклы для отдельных целевых устройств, в то время механизм, основанный на целевом устройстве, позволяет это делать.

Если целевое устройство неспособно обеспечить оба вышеуказанных условия, то оно не должно реализовывать данный бит вообще, тогда при чтении статусного регистра автоматически будет возвращаться ноль.

Обратите внимание, что главная польза от реализации такого решения - это получение преимущества при повышении эффективности в конфигурациях систем типа «low, end», которые используют шину PCI для исполнения программ (например, осуществляется связь процессор - основная память). Рекомендуется, чтобы во всех новых целевых устройствах предусматривалось возможное использование их в таких конфигурациях, особенно, если стоимость реализации незначительна. Тем не менее, не рекомендуется, чтобы существующие части устройств были «пропущены» через цикл перепроектирования исключительно для реализации этой особенности, поскольку реально это не будет приносить пользу от части всего решения, если, конечно, данное устройство не будет специально разработано для такой системы.

Во всех других состояниях, мастер должен вставлять минимум одно состояние ожидания на шине (по крайней мере, всегда имеется хотя бы одно состояние ожидания шины между транзакциями, осуществляемых различными мастерами). Обратите внимание, что многопортовые целевые устройства должны блокировать себя, когда они действительно заняты в течение быстрых back-to-back транзакций (за более подробной информацией обращайтесь к разделу 3.5.).

Во время быстрой back-to-back транзакции, мастер начинает следующую транзакцию немедленно, без состояния ожидания шины. Последняя фаза данных завершается, когда FRAME# - неактивный, а IRDY# и TRDY# - активны. Текущий мастер начинает следующую транзакцию в том же самом такте, в котором передавались последние данные в предыдущей транзакции.

Очень важно заметить, что агенты, не включенные в последовательность быстрых back-to-back транзакций, не могут (и вообще не нуждаться) немедленно определять промежуточные границы транзакции, используя только сигналы FRAME# и IRDY# (цикл ожидания шины отсутствует). Только во время быстрых back-to-back транзакций мастера и целевые устройства испытывают потребность определять эти границы. Когда завершена последняя транзакция, все агенты «увидят» цикл ожидания. Тем не менее, те устройства, которые поддерживают механизм, основанный на целевом устройстве, должны быть способны определить завершение всех транзакции PCI, а также обнаруживать все фазы адреса.

На рисунке 3-8, мастер завершает запись в такте 3, и фаза адреса следующей транзакции происходит в такте 4. Целевое устройство должно начать отслеживать FRAME# в такте, следующем после завершения текущей транзакции данных. Целевые устройства должны быть способны дешифрировать back-to-back операции, в то время как мастер может, хотя и необязательно, также поддерживать эту функцию. Целевое устройство может по выбору повторить запрос, после того, как был потребован монопольный доступ, путем установки сигнала DEVSEL#.

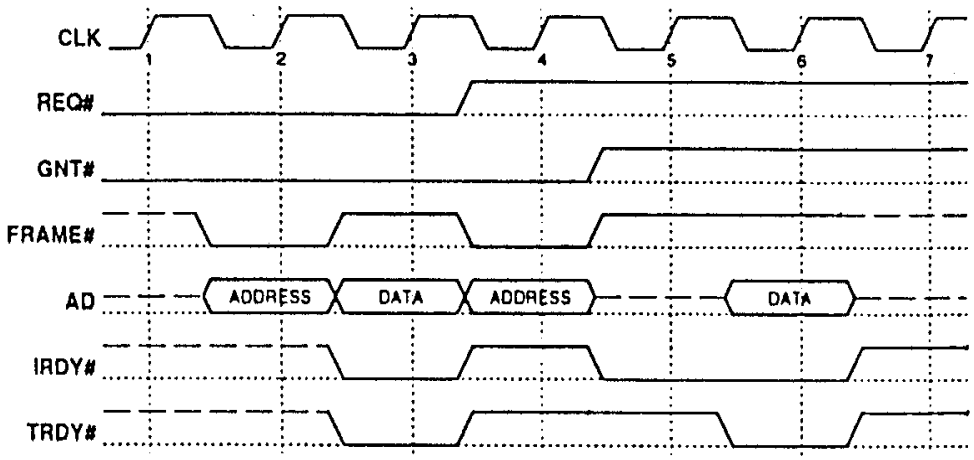


Рисунок 3-8: Арбитраж для back-to-back запроса

3.4.3. Фиксирование арбитража

Термин *парковка* подразумевает разрешение арбитру установки в активное состояние сигнала GNT# для выбранного агента, когда в данный момент времени никакой агент не использует и не запрашивает шину. Арбитр может выбирать заданного по умолчанию мастера, любым требуемым способом (фиксированного мастера, последнего используемого мастера, и т.д.), либо может вообще никого не выбирать (обозначая себя в качестве заданного по умолчанию мастера). Когда арбитр устанавливает активное состояние для сигнала GNT# агента, и шина находится в состоянии ожидания, то этот агент должен разрешить свои линии AD[3l::00], C/BE[3::0]# и (одним тактом позже) буферы вывода PAR в течение восьми тактов PCI (требуется), в то время как рекомендуется два - три такта (смотрите обсуждение контроля по четности и описание временных соотношений между PAR и AD). Агент не обязан разрешить все буферы за один такт. Просто это требование гарантирует, что арбитр может без риска закрепить шину за некоторым агентом и знать, что шина будет работать нормально (если арбитр не закрепляет шину за каким - либо агентом, то шиной будет управлять устройство центрального ресурса, в которое встроен арбитр).

Во всех случаях, когда шина находится в состоянии ожидания, а арбитр убирает сигнал GNT# агента, агент теряет запрос к шине, за исключением одного случая. В этом случае арбитр переводит в неактивное состояние сигнал GNT# агента, установившего FRAME#, и мастер продолжает транзакцию. В противном случае, агент должен перевести в третье состояние линии AD[3l::00], C/BE# [3::0] и (одним тактом позже) PAR. В отличие от вышеупомянутого варианта, агент должен отключить все буферы за один такт, чтобы избежать возможной конкуренции со следующим мастером шины.
Подводя итог вышесказанному, минимальные временные параметры при арбитраже, на шине PCI в состоянии ожидания, должны быть следующими:

- Агент закреплен: ноль тактов - для закрепленного агента, два такта - для остальных.
- Агент не закреплен: один такт для любого агента.

Когда шина закреплена за агентом, ему разрешается начать транзакцию без установленного сигнала REQ# (мастер может начинать транзакцию, когда шина находится в состоянии ожидания, и активен сигнал GNT#). Если агенту требуется сделать множество транзакций, то он должен установить сигнал REQ#, чтобы сообщить арбитру о своих намерениях. Когда мастеру требуется только одна транзакция, то он не должен устанавливать в активное состояние сигнал REQ#; в противном случае, арбитр может продолжать устанавливать его сигнал GNT#, когда использование шины не требуется.

3.4.4. Временные задержки

Здесь подразумевается наибольшая задержка во времени при высокой производительности шины ввода-вывода. В данном разделе описываются механизмы PCI, помогающие предсказывать и управлять временем задержки в наихудшем случае. Используя эти механизмы, можно предсказать возможные задержки для окружения PCI и эффективного интерфейса основной памяти с достаточно высокой точностью. Применение стандартной шины расширения (ISA, EISA и т.д.) осложняет предсказание задержек во времени. Если есть шина расширения, то наихудший случай времени ожидания получается за счет самой шины расширения, либо из-за "неблагополучных" характеристик адаптера.

3.4.4.1. Временные задержки на PCI

Рисунок 3-9 описывает различные составляющие временных параметров, которые происходят для устройства, осуществляющего запрос к шине. Первая часть - это время задержки арбитража, т. е. время, которое ожидает мастер между установкой запроса REQ# и получением разрешения в виде установленного GNT#. Данное время является временным параметром при арбитраже, когда назначаются приоритеты запрашивающим устройствам в процессе эксплуатации система. Для самого «быстрого» устройства это время, обычно составляет два такта PCI. Следующая составляющая - это задержка при ожидании шины - то есть, как долго устройство должно ожидать освобождения шины. Следующая составляющая - это время задержки целевого устройства, то есть количество времени, которое необходимо целевому устройству, чтобы установить в активное состояние сигнал FRAME# для первой передачи данных.

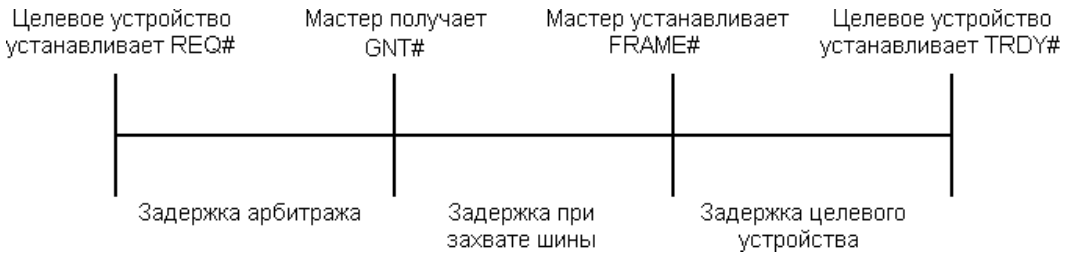


Рисунок 3-9: Компоненты времени задержки при запросе

Проверка может работать неопределенно длительное время, поскольку целевое устройство может посылать / принимать данные, и нет агента, запрашивающего шину. Тем не менее, PCI предусматривает два механизма, предусматривающих задержку мастера, при условии, что присутствуют другие запросы, так, чтобы можно было предсказать время на захват шины. Эти механизмы определяются следующим образом:

Таймер задержки мастера (Master Latency Timer - LT): LT каждого мастера обнуляется и приостанавливается всякий раз, когда не может быть установлен сигнал FRAME#. Когда мастер устанавливает FRAME# в активное состояние, то это дает возможность запустить LT. Если мастер установил в неактивное состояние FRAME# до окончания счета, то работа LT бессмысленна. В противном случае, как только истекает время (счетчик = "T" тактов), мастер должен инициировать завершение транзакции (смотрите раздел 3.3.3.1.) так скоро, как только GNT# будет установлен в неактивное состояние. В сущности, "T" представляет минимально гарантируемый интервал времени («отмеряемые» такты PCI) для мастера, после тот должен прекратить ожидание, как только будет установлен в неактивное состояние GNT# (фактическое завершение работы не происходит, пока целевое устройство находится в состоянии готовности).

Завершение транзакции, инициированное целевым устройством (специальный разрыв связи): целевое устройство должно манипулировать сигналами TRDY# и STOP#, чтобы завершить транзакцию при окончании фазы данных - за "N" тактов (где N=1,2,3...), если возрастающее время задержки для фазы данных "N + 1" больше, чем число прошедших тактов PCI. Например, когда мастер PCI производит чтение из шины расширения, это занимает минимум 15 тактов. При использовании правила N=1, время задержки для фазы данных равно 2 - 15 тактам, таким образом, целевое устройство должно завершить работу после окончания фазы данных (например, «медленное» целевое устройство должно прекратить попытки работу на границах фазы данных).

Обратите внимание, что ни один механизм не ограничивает время задержки первой фазой данных (это обсуждается в разделе 3.4.4.3.). Например, в отдельной системе нет целевых устройств настолько медленных, который бы задерживали TRDY# более, чем T+8 тактов для завершения первой фазы данных. С таким предположением и вышеперечисленными механизмами, можно показать, что самая длинная транзакция, которую может выполнить мастер, составит T+8 тактов (при условии, что GNT# будет установлен в неактивное состояние прежде, чем LT закончит счет). Если принять T=40, то максимальная транзакция займет 48 тактов (1.44 мкс при 33 МГц).

В действительности, параметр "T" характеризует соотношение между производительностью (относительно высокие величины) и временем задержки (малые величины). Например, T=40 означает, что блок состоит из 32 фаз данных (байты 128/256 в 32/64-разрядной транзакции), при условии, что мастер и целевое устройство готовы к 0 циклам ожидания (восемь тактов времени задержки для первых данных). Уменьшение T до 20 тактов может прервать блок из 12-14 пересылок, но зато ограничить максимальную транзакцию до 28 тактов (0.84 мкс при 33 МГц).

3.4.4.2. Определение времени задержки

Время задержки запроса меняется от одной реализации системы к другой, в зависимости от устройств, используемых в данной системе. Но разработчики PCI - устройств должны иметь некоторый эталон времени задержки при типичном запросе, так что они могут обеспечить достаточно неплохую аппаратную буферизацию для минимизирования временных параметров. На рисунке 3-9 каждую составляющую времени задержки при запросе можно рассмотреть индивидуально. Время задержки арбитража может отдать два такта для устройства с наивысшим приоритетом. Низкоприоритетные устройства будут иметь более длинные времена задержки, в зависимости от числа высокоприоритетных устройств и коэффициента использования устройства.

Время задержки при захвате шины определяется одним из следующих двух источников. Это либо наибольшая задержка первой фазы данных в системе, либо значение в таймере времени задержки, при условии, что все времена задержки первых данных минимальные. Задержка первой фазы данных (то есть время ожидания первых данных) - это время от установки сигнала FRAME# до установки сигнала TRDY#.

Третья составляющая времени задержки при запросе - это время задержки целевого устройства, которое обычно равно времени ожидания первых данных. Но устройства организующие интерфейс с центральным процессором (ЦП) и осуществляющие буферизацию, не могут увеличивать время задержки целевого устройства. Например, если устройство делает запрос в оперативную память через буферизированный интерфейс с ЦП, то интерфейс может сбросить свои буферы перед разрешением окончания запроса. Интерфейс сообщит о повторе запрашивающему устройству, сбросит буферы и затем будет готов ответить на запрос. Это может привести к большой величине времени задержки целевого устройства, в зависимости от того, где осуществляется сброс (например, на шине расширения) и каков размер буфера.

Следующие пункты показывают несколько различных значений времени задержки для разных последовательностей событий на шине PCI. Они могут служить примером при вычислении времен задержек при разных запросов. Такие вычисления сделаны со следующими предположениями:

- Шина PCI работает на частоте 33 МГц.
- Время задержки при ожидании первых данных составляет 16 тактов (0.5 мкс) для всех устройств, которые не являются шинами расширения.
- Время между последовательностями фаз данных при блочной передаче - восемь тактов.
- Побайтовый доступ на шине расширения занимает 1.5 мкс, причем время задержки первых данных равно 6 мкс при 32-битном доступе через интерфейс шины расширения.

Сценарий 1: Мастер-устройство желает сделать запрос, когда нет запросов от других агентов, и шина находится в состоянии ожидания. Обычно время задержки арбитража занимает два такта. Время

задержки при захвате шины равняется нулю, так как шина находится в состоянии ожидания. Время задержки целевого устройства - 16 тактов, таким образом общее время ожидания составляет l8 тактов (0.54 мкс).

Сценарий 2: Мастер-устройство желает сделать запрос, когда другое устройство только что начало транзакцию на шине PCI. Мастер, использующий шину, хочет осуществить блочную передачу из l6 фаз данных. Счетчики времени задержки в системе устанавливаются на 66 тактов, что эквивалентно 2 мкс.
Время задержки арбитража обычно равняется двум тактам, если только нет каких-то отложенных запросов. Время задержки при захвате шины равняется 72 тактам, что означает 2 мкс перед тем, как LT завершит отсчет, плюс восемь тактов, чтобы позаботиться о последней фазе данных. В это время происходит восемь пересылок данных (l6 тактов для первой фазы данных, шесть пересылок данных по восемь тактов на каждую, пока LT не завершит счет, и затем еще восемь тактов для последней фазы данных). Время задержки целевого устройства - 16 тактов. Обратите внимание, что в этом случае время задержки арбитража и время ожидания при захвате шины перекрываются так, что общее время задержки достигает 88 тактов (2.7 мкс).

Сценарий 3: Мастер-устройство желает запрос через интерфейс ЦП, которому необходимо сбросить буферы. Буферы сбрасываются через шину расширения, и размер буфера - четыре байта.
Время задержки арбитража равняется двум тактам, а время задержки при захвате шины нулевое. Когда устройство делает свой запрос, интерфейс ЦП сообщит о повторе, вынуждая запрашивающее устройство вернуться к начальному времени ожидания. Интерфейс осуществит арбитраж для шины (два такта), а затем выполнит сброс буферов. Это займет 6 мкс. В это время, для первоначального запрашивающего устройства будет выполнен повторный арбитраж шины, и оно будет ждать, пока шина не перейдет в состояние ожидания. Как только завершится сброс буферов, мастер-устройство осуществит запрос, задержка целевого устройства составит 16 тактов. Общее время задержки - составит 222 такта (два такта для арбитража, четыре такта - для передачи сигналов о повторе, 200 тактов для сброса буферов и l6 тактов - для заключительного времени задержки целевого устройства). Это - 6.7 мкс при частоте шины в 33 МГц.

3.4.4.3. Рекомендации для величины времени ожидания

Необходимо согласовывать устройства, чувствительные как производительности, так и к задержке во времени. Например, во время связанной с ЦП модификации изображения на экране, трафик при передаче буфера изображения может занимать большую часть от ширины диапазона передаваемых по шине данных. Иногда возможны необычайно длинные запросы, если типичная задержка очень мала. Такие особенности PCI, как пересылка блоков данных с неограниченной длиной (широкий диапазон блочного ввода-вывода; FDDI) и закрепление шины за агентом (высокая пропускная способность; например, при передаче буфера изображения) отвечают требованиям устройств с высокой производительностью. С другой стороны, сетевые устройства (LAN - адаптеры) со скоростью 10 Мбит/сек занимают лишь небольшую часть пропускной способности шины PCI. Чтобы сохранить экономичные характеристики таких устройств (требования по оптимизации буферов), такие особенности PCI, как основанный на запросах арбитраж и счетчики задержек мастер - устройства (LT), позволяют ограничить наихудший случай времени ожидания. Например, счетчик задержки может вынудить устройство с большим буфером FDDI, обычно реализующее за один запрос блок длиной в 128 двойных слов (DWORD), разбить передачу на отдельные малые части, осуществляя таки образом поверхностную буферизацию. Устройства, чувствительные ко времени ожидания, обращаются к шине наиболее часто. Высокая производительность и малое время ожидания часто дополняют друг друга. Соответственно, очень важно, чтобы системы с PCI компонентами разрабатывались очень тщательно, чтобы минимизировать риск, связанный с временем ожидания при взаимодействии систем, и облегчать приемлемые соотношения цена/производительность.

У большинства PCI - систем типичное время задержки очень мало (около 2 мкс), и оно легко определяется. Тем не менее, в наихудшем случае (который, в свою очередь, достаточно редок), время задержки не может быть слишком большим, но такие ситуации очень трудно предсказать. Например, задержка для стандартной платы расширения (ISA/EISA/MC) относительно интерфейса часто зависит от самого адаптера, а не от шины PCI (это особенно проблематично, так как некоторые существующие адаптеры не обеспечивают временные параметры задержки, определенные стандартом шины). Для того, чтобы это компенсировать, мастер - устройства, для которых надо гарантировать

наихудший случай задержки, должны обеспечить адекватную буферизацию за 30 мкс. Это подразумевает буферизацию минимум около 50 байтов для LAN - адаптеров со скоростью 10 Мбит/сек, и около 500 байтов - для LAN - адаптера со скоростью 100 Мбит/сек (если буферы организованы в виде строк (например, с 16- или с 32-битным выравниванием) для наилучшего использования PCI и памяти целевого устройства, то минимальный размер буфера желательно увеличить). Несмотря на некоторую неопределенность в наихудшем случае, для реализуемых системных разработок 30 микросекунд должны обеспечить достаточно широкое поле деятельности. Даже при самых жестких требованиях, 30 микросекунд должно хватить в наихудшем случае. Для снижения стоимости проектирования в наихудшем случае мастер - устройство может уменьшить задержку для наихудшего случая, если допускаются случайные нарушения величины задержки без создания ошибочных условий. Например, LAN - адаптер должен иметь соответствующую инфраструктуру (аппаратное и программное обеспечение), чтобы можно было определить, когда происходят ошибки при буферизации, и инициировать повторную передачу блока данных (как в случае, когда блоки искажаются из-за наводок в линиях или коллизий). Секторы на диске можно повторно считать или записать. Передача в реальном времени потока данных (например, аудио- данных) может допускать случайные типовые потери, используя при этом предыдущую выборку данных. Устройство, достаточно устойчивое к сбоям и нарушениям задержки, изменяет значение задержки из обычного состояния в более «благоприятное» для соотношения цена/производительность (то есть, в случае обработки звука мы получим шумы). Например, дешевый LAN - адаптер для клиентских станций (с небольшой загрузкой шины PCI) проектируют с задержкой 10 мкс в наихудшем случае и принимают, что частота повторения пакетов слишком мала, чтобы повлиять на снижение эффективности. Устройства, разрабатываемые для серверов, должны иметь большие буферы (при высокой стоимости) для уменьшения повторений в случае больших нагрузок (если принимать во внимание необходимость разработки очень устойчивых приложений, то устройства, спроектированные для задержки 30 мкс и более, должны самовосстанавливаться во всех случаях нарушения задержки). Некоторые приложения (например, встраиваемые контроллеры, мультимедиа и т. д.) могут принимать на себя более устойчивое управление задержками, чем то, которое обеспечивается обычной (общего назначения) главной шиной PCI (такие условия могут быть реализованы на вторичной шине PCI и подсоединены к host - машине через интерфейс PCI - PCI). Этому типу приложений требуется, чтобы у целевых устройств было малое и предсказуемое время задержки. Для обеспечения такого проектирования, и, в общем случае, для поощрения хороших разработок, при разработке интерфейса целевых устройств рекомендуется пользоваться принципами, перечисленными ниже. Далее, термином «одноуровневый» (single-layer) обозначается целевое PCI - устройство, которое обеспечивает немедленный доступ к выбранному ресурсу. В качестве примера могут служить одно-портовая динамическая память (DRAM) и буфер изображения видеопамяти (VRAM). «Многоуровневым» называется целевое устройство, осуществляющее арбитраж ресурса, выбранного другим ресурсом, для которого арбитраж не осуществляется. В качестве примера можно указать интерфейсы «PCI - шина расширения» (например, PCI - ISA/EISA/MC; PCI - PCI; PCI - основная память и т. д.), а также двух-портовую DRAM.

1. Одноуровневое целевое устройство должно отсрочить первую фазу данных на 16 тактов. В противном случае, если временное внутреннее состояние (например, регенерация DRAM, полная очередь отложенных запросов на запись и т. д.) увеличивает задержку запроса более, чем на 16 тактов, то немедленно осуществляется повтор.
2. Многоуровневое целевое устройство должно повторять запрос, который находится в коллизии с занятым ресурсом. Например, если осуществляется запрос, подчиненный EISA - шине, которая захвачена другим мастером, то следует немедленно повторить этот запрос (в любом случае, был бы необходим цикл ожидания для исключения тупиковой ситуации, поэтому ожидание и происходит). Аналогично, при запросе регенерируемого буфера изображения DRAM следует повторить данный запрос.
3. «Многоуровневые» целевые устройства (особенно, шинные интерфейсы) должны соблюдать большую осторожность в стратегии записи буфера. Буферы записи затрудняют определение задержки, так как обычно предъявляемые строгие требования гласят, что перед разрешением какого-то запроса, должна быть обслужена вся очередь записей. Рекомендуется, в частности, чтобы основной интерфейс ЦП - PCI мог откладывать записи в резидентный PCI - буфер, при этом подавляя отложенные записи на подчиненную шину ISA/EISA/MC.

Обычно, чтобы облегчить задачу создания систем, продавцы компонентов должны прилагать описание «поведения» для времени ожидания (обычно запрещено давать описание поведения адаптеров стандартных шин в реальном времени). Для мастер - устройства необходимо указывать как время ожидания, так и последствия его нарушения. Целевые устройства должны определить наихудший случай ответа, а также все события, которые могут вызывать повторение или разъединение. Если время ожидания адресата обуславливается внешними факторами (например, как долго адаптер ISA производит цикл), то это должно быть ясно установлено.

3.5. Монопольный доступ

PCI обеспечивает исключительный механизм доступа, который разрешает завершить неисключительный доступ до начала исключительного доступа. Это называется блокировкой ресурса. Это позволяет будущим процессорам задержать аппаратную блокировку параллельно нескольким неисключительным доступам, передачу анимационных данных в реальном масштабе времени, например, видео. Механизм работает только при блокировке PCI- ресурса, для которого оригиналу использовался блокированный доступ. Этот механизм полностью совместим с существующим программным использованием исключения.

LOCK# требуется на любом устройстве, обеспечивающем системную память. Определенно, если устройство является выполнимой памятью, оно должно также выполнить LOCK# и гарантировать полное исключение доступа в память (то есть, если имеется хозяин, использующий эту память, он должен также выполнить блокировку). Ведущие интерфейсы, которые имеют системную память ниже них, должны также выполнить LOCK#.

Сигнал LOCK# указывает, что исключительный доступ начал работать. Утверждение GNT# не гарантируют управление сигналом LOCK#. Управление LOCK# получено под собственный протокол вместе с GNT#. При использовании блокировки ресурса, агенты выполняющие неисключительный доступ могут продолжать свою работу, даже в то время, когда другой мастер использует сигнал LOCK#. Однако, из требований совместимости, арбитр может преобразовывать блокировку ресурса в блокировку "шины", предоставляя агенту, который выставил LOCK#, исключительный доступ к шине до выставления LOCK#. За более подробной информацией относительно полной блокировки шины обращайтесь к разделу 3.5.6.

Блокировка ресурса, гарантируется адресатом доступа, не исключая всех других агентов, обратившихся к шине. Степень детализации блокировки должна быть 16 выровненных байтов. Исключительный доступ к любому байту в блоке 16 байтов будет весь 16 байтовый блок. Мастер не может использовать любые адреса вне 16 байт, чтобы быть блокирован. Адресат может блокировать минимум 16 (выровненных) байтов и максимумом весь ресурс. Исходя из этого, следующие параграфы описывают поведение хозяина и адресата.

Правила LOCK# будут установлены и для хозяина и адресата. После правил, следует описание того, как запустить, продолжать и завершать исключительную операцию, и будет их обсуждение. Затем обсуждение того, как адресат ведет себя, когда он поддерживает и блокировку ресурса и кэшированную память с обратной записью. В заключении раздела обсуждается, как выполнить полную блокировку шины на PCI.

Адресат, который поддерживает LOCK# на PCI, должен твердо придержаться следующих правил:

1. Адресат доступа блокирует себя самостоятельно, когда LOCK# является деактивированным в течение адресной фазы.
2. Если только блокировка установлена, адресат остается блокированным, пока не произведена выборка FRAME# и LOCK# одновременно или поступило сообщение об аварийном прекращении работы.
3. Исключительно гарантировать владельцу сигнала LOCK# (если только блокировка установлена) минимум l6 байтов (выровненных) ресурса[7] . Это включает доступ, которого нет на PCI для многопортовых устройств.

Все PCI адресаты, которые поддерживают монопольные доступы, должны произвести выборку LOCK# с адресом. Если адресат доступа выполняет декодировку средне или медленно, нужно удерживать сигнал LOCK# в течение фазы адреса, чтобы определить, блокирован ли доступ, пока завершается декодирование. Адресат маркера транзакции блокирует сам себя, если LOCK# выключен в течение фазы адреса. Если адресат ожидает сигнала LOCK#, пока активирован DEVSEL#, он не может различить, блокирован ли текущий доступ или тот, который происходит одновременно с блокированным доступом. Исполнитель может сохранять "состояние", чтобы определить, блокирован ли доступ, но это требует защелки LOCK# на последовательных тактах и сравнения для определения блокирован ли доступ. Более простой способ для адресата - пометить себя блокированным при любом доступе. Это требуется там, где LOCK# является деактивированным в течение фазы адреса. Блокированный адресат остается в блокированном состоянии, пока FRAME# и LOCK# являются деактивированными. Для разрешения другого доступа на многопортовое устройство, адресат может производить выборку LOCK# на такте, следующим после фазы адреса, чтобы определить, блокировано ли устройство действительно. Когда LOCK# является выключенным в течение фазы адреса и включенным (такт после фазы адреса), многопортовое устройство блокировано и должно гарантировать доступ только PCI мастеру. Когда LOCK# является выключенным в течение фазы адреса и такта после фазы адреса, адресат свободен, чтобы ответить на другие запросы и не блокирован. Блокированный адресат может только принимать запросы, когда LOCK# выключен в течение фазы адреса. Блокированный адресат ответит, выставив STOP# с выключенным TRDY# всем транзакциям, когда LOCK# включена в течение фазы адреса.

Итак, адресат доступа самостоятельно блокирует себя при любом доступе, это требуется, когда LOCK# выключен в течение фазы адреса. Адресат отпирает себя, когда FRAME# и LOCK# являются оба выключенными. Это небольшая путаница для адресата, который блокирует себя на транзакцию, которая не блокирована. Однако с точки зрения реализации, это простой механизм, который использует комбинаторную логику и всегда работает. Устройство отопрет себя непосредственно в конце транзакции, когда FRAME# и LOCK# выключены. Адресат может также помнить состояние (которое является пригодным для многопортового устройства), чтобы определить, правильно блокировано ли он или нет. Адресат правильно блокирован, когда LOCK# выключен в течение фазы адреса и включен на следующем такте.)

Обратите внимание: арбитр должен быть в алгоритме "равнодоступности", когда LOCK# включен; иначе может происходить Livelock (длительная блокировка).

Существующее программное обеспечение, которое не поддерживает правила использования блокировки PCI, неправильно работать. Для PCI резидентной памяти (прежде всего системной памяти), которая поддерживает LOCK#, требуется быть обратно совместимой к существующему программному обеспечению, рекомендуется выполнить полную блокировку ресурса. Обратитесь к разделу 3.5.5. для получения подробной информации по избежанию зависания.

[7] Максимум - полностью весь ресурс.

Мастер, который использует LOCK# на PCI, должен твердо придержаться следующих правил:

- Мастер может обращаться только к одиночному ресурсу в течение операции блокировки.
- Блокировка не может колебаться между границей устройства.
- Шестнадцать байтов (выровненные) - максимальный размер ресурса, который доступен только мастеру в течение операции блокировки. Доступ к любой части из этих 16 байтов блокирует все 16 байтов.
- Первая транзакция операции блокировки должна быть транзакцией чтения.
- LOCK# должен быть выставлен на такте, следующем после фазы адреса и оставаться таким до поддержки управления.
- LOCK# должен быть снят, если повторение сообщается прежде, чем фаза данных завершилась и блокировка не была установлена[8] .
- LOCK# должен быть всегда выключен, когда доступ завершен аварийным прекращением работы как со стороны ведомого, так и со стороны ведущего.
- LOCK# должен быть выключен минимум для одного цикла ожидания (IDLE) между последовательными операциями блокировки.

3.5.2. Начало монопольного доступа

Когда устройство должно сделать исключительную операцию, оно внутренне проверяет состояние LOCK# перед выставлением REQ#*. Мастер отмечает LOCK# занятым всегда, когда LOCK# выставлен, и не занятым, когда FRAME# и LOCK# являются выключенным. Если LOCK# занята, агент должен удержать сигнал REQ#, пока LOCK# не является доступной.

Пока ожидается разрешение, мастер продолжает контролировать LOCK#. Если LOCK# является занятой, мастер деактивирует REQ#, потому что другой агент получил управление LOCK#.

Когда мастер получил доступ к шине и LOCK# не занята, использование LOCK# произошло. Мастер свободен, чтобы выполнить исключительную операцию, когда текущая транзакция завершается и только агент на шине может управлять LOCK#. Все другие агенты не должны управлять LOCK# даже когда они в данное время являются хозяином.

Рисунок 3-10 иллюстрирует старт монопольного доступа. LOCK# является выключенной в течение фазы адреса для запроса операции блокировки, которая должна быть инициализирована с командой чтения. LOCK# должен быть выставлен на такте после фазы адреса, которая находится на такте 3. перехода адресата в блокированное состояние, которое позволяет текущему хозяину сохранять монопольное использование LOCK# до конца текущей транзакции.

[8] Мастер продолжает монопольно владеть LOCK#, если завершение осуществляется с повтором или без него.

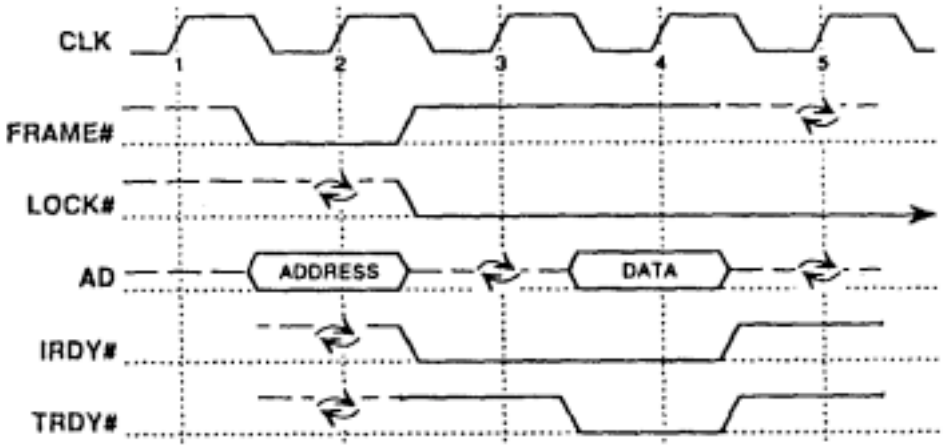


Рисунок 3-10: Начало монопольного доступа

Блокированная операция не установлена на шине до завершения первой фазы данных первой транзакции (IRDY# и TRDY# установлены). Если адресат повторяет первую транзакцию без завершения фазы данных, мастер не только должен запретить транзакцию, но и также должен также выставить LOCK#. Если только первая фаза данных завершается, исключительная операция установлена, и мастер сохраняет LOCK# выставленной, пока либо операция блокировки не завершается, либо по ошибке (аварийное прекращение работы по инициативе мастера или адресата). Аварийное прекращение работы по инициативе адресата разъединяет нормальное соединение даже когда операция блокировки установлена. Когда работа мастера завершена адресатом с разъединением или повторением после того, как блокировка была установлена, адресат указывает, что в настоящее время шина занята и невозможно завершить запрошенную фазу данных. Адресат получит доступ, когда он не занят и продолжает удерживать блокировку, исключая все другие доступы. Мастер продолжает управлять сигналом LOCK#. Неисключительный доступ к неблокированным адресатам на PCI возможен в то время, когда LOCK# установлен. Когда монопольный доступ выполнен, LOCK# является выключенным, и другие владельцы могут соперничать за монопольный доступ.

3.5.2. Продолжение монопольного доступа

Рисунок 3-11 показывает главное продолжение монопольного доступа. Однако этот доступ может или не может завершить исключительную операцию. Когда хозяину предоставлен доступ к шине, он начинает другой монопольный доступ к адресату, которого он предварительно блокировал. LOCK# является выключенным в течение фазы адреса, чтобы восстановить блокировку. Блокированное устройство принимает и отвечает на запрос. LOCK# выставлена на такте 3, чтобы сохранить адресата в блокированном состоянии и позволить текущему мастеру сохранять монопольное использование LOCK# до конца текущей транзакции.

Когда мастер продолжает операцию блокировки, он продолжает удерживать LOCK#. Когда мастер завершает операцию блокировки, он снимает LOCK# после последней фазы данных, которая происходит на такте 5 (обратитесь к разделу 3.5.4. за более подробной информации относительно завершения монопольного доступа).

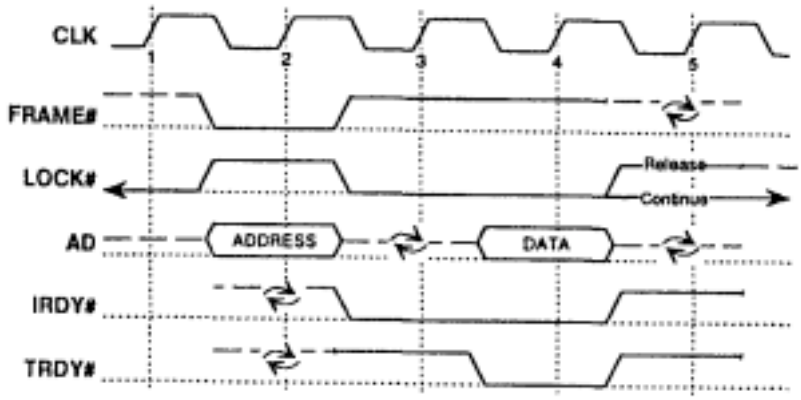


Рисунок 3-11: Продолжение монопольного доступа

3.5.3. Осуществление доступа к блокированному агенту

Рисунок 3-12 показывает попытку мастера получить неисключительный доступ к блокированному агенту. Когда LOCK# выставлен в течение фазы адреса, и если адресат блокирован, он сообщает о том, что никакие данные не перемещены. Разблокированный адресат игнорирует LOCK# при решении, что он должен ответить. Также, пока LOCK# и FRAME# выставлены в течение фазы адреса, разблокированный адресат не входит в блокированное состояние.

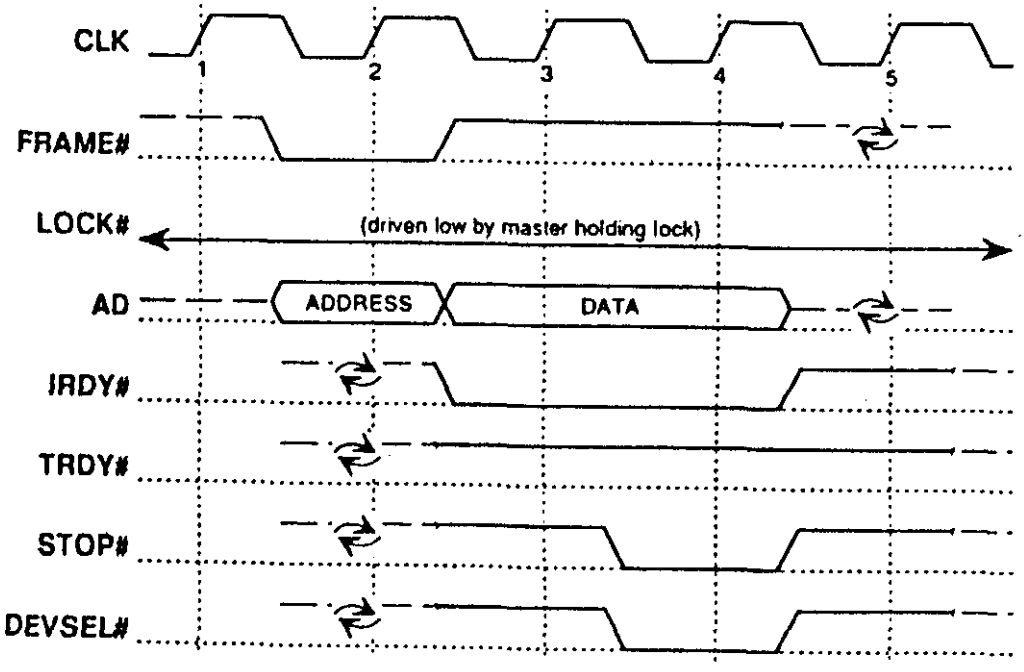


Рисунок 3-12: Доступ к блокированному агенту

3.5.4. Завершение монопольного доступа

В течение заключительной передачи исключительной операции, LOCK# является невыставленным, так что адресат примет запрос, а затем подтверждение, пока монопольный доступ не завершится успешно. Мастер может снять LOCK# в любое время, когда исключительная операция завершена. Однако рекомендуется (но не требуется) чтобы LOCK# была выключена с выключением IRDY# после завершения последней фазы данных блокированной операции. Выставление LOCK# в любое другое время может приводить к последующей транзакции, завершаемой с лишним повторением. Блокированный агент разблокирует себя всякий раз, когда LOCK# и FRAME# выключены.

Если хозяин хочет выполнять две независимых исключительных операции на шине, он должен гарантировать минимум один такт между операциями, где и FRAME# и LOCK# являются выключенными. (Например, случай, описанный на рисунке 3-8 (такт 3) был бы запрещен). Это гарантирует, что любой адресат, блокированный первой операцией, будет разблокирован до старта второй операции. Агент должен разблокировать себя, когда FRAME# и LOCK# оба выключены на одном фронте синхроимпульса.)

3.5.5. Поддержка сигнала LOCK# и согласование кэша обратной записи

Блокировка ресурса, как описано ранее, может зависнуть при использовании связанного кэша. Зависание может произойти, если программное обеспечение позволяет блокировкам пересекать границу строки кэша, когда обеспечивается кэширование, и используется полная блокировка ресурса. Пример этого зависания - строки кэша диапазонов блокировки n и n + 1. Строка кэша n + 1 была изменена самим кэшем. Мастер устанавливает блокировку, читая строку кэша n. Операция блокировки продолжается чтением строки кэша n + 1. Результат (n + 1) приводит к HITM, который указывает, что изменяемая строка была обнаружена. Обратная запись изменяемой строки неудачна, потому что адресат только получил доступ от владельца LOCK#. Это приводит к зависанию, потому что чтение не может происходить, пока изменяемая строка не записана, и запись не может происходить до выставления LOCK# мастером, имеющим монопольное использование.

Этого зависания избегают, требуя адресатов, которые поддерживают writeback кэшированную память (и полную блокировку ресурса), чтобы позволить обратную запись, даже когда он блокирован.

Адресат может различать обратную запись и другие транзакции записи с помощью SDONE и SBO# в течение фазы адреса (или такта после выставления SDONE, когда адреса поставлены в очередь). Когда CLEAN обозначен в течение фазы адреса, текущая транзакция является или CLEAN или writeback. Переход из HITM к CLEAN в течение фазы адреса указывает замену строки. В то время как переход из HITM к CLEAN в течение фазы адреса указывает обратную запись. Адресат обратной записи, вызванной HITM к CLEAN в течение фазы адреса должен принять ее, даже когда он блокирован. Адресат может принимать замены строки (STANDBY к CLEAN в течение фазы адреса), но это не требуется когда он блокирован. Все другие транзакции завершены адресатом с повторением, когда он блокирован. (Обратите внимание, что адресат обратной записи не может завершать транзакцию, пока строка кэша не была перемещена).

3.5.6. Полная блокировка шины

Блокировка PCI-ресурса может быть преобразована в полную блокировку шины, если арбитр предоставляет шину любому другому агенту пока LOCK# включена. Когда первый доступ повторен, мастер должен выключить и REQ#, и LOCK#. Когда первый доступ завершается нормально, полная блокировка шины тоже, арбитр не будет предоставлять шину любому другому агенту. Если арбитр предоставил шину другому агенту, когда полная блокировка шины устанавливалась, арбитр должен удалить другой доступ, чтобы гарантировать, что установлена полная блокировка шины. Полная блокировка шины может нанести значительный удар по эффективности системы, и особенно видеосистемы. Все неисключительные доступы не будут продолжены, пока происходит операция блокировки.
Полная блокировка ресурса и кэшируемой памяти с обратной записью необходимы для полной блокировки шины. Арбитр, который поддерживает полную блокировку шины, должен предоставлять шине доступ к кэшу, чтобы выполнить обратную запись в изменяемой строке, когда происходит блокировка. (Адресат может принять обратную запись, когда он блокирован, потому что не может сообщить, используется полная блокировка шины или блокировка ресурса).

3.6. Другие операции с шиной

3.6.1. Выбор устройства

DEVSEL# управляется адресатом текущей транзакции, как показано на Рисунке 3-13. DEVSEL# может управляться одним, двумя или тремя тактами после фазы адреса и выставляется в регистре состояния, описанном в Разделе 6.2.3. DEVSEL# должен быть выставлен до фронта, на котором адресат выставляет IRDY#, STOP#, или данные. Другими словами, адресат должен выставить DEVSEL# прежде, чем выдает любой другой ответ. Во всех случаях, за исключением случая, когда адресат выставил DEVSEL#, он должен не выставлять DEVSEL#, пока не сброшен FRAME# (IRDY# выставлен) и последняя фаза данных завершена. При нормальном завершении мастером, снятие DEVSEL# должно совпадать со снятием IRDY#. Исключение - аварийное прекращение работы.

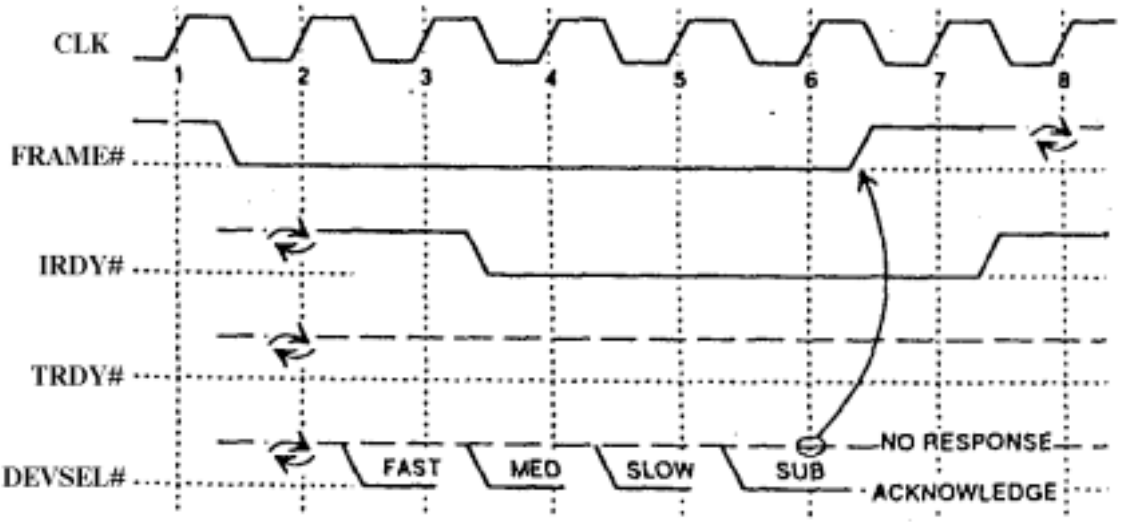


Рисунок 3-13: Установка в активное состояние сигнала DEVSEL#

Если никакой агент не выставил DEVSEL# внутри трех тактов FRAME#, агент, делающей декодирование, может требовать и выставление DEVSEL#. Если система не имеет такого агента, мастер никогда не увидит выставленный DEVSEL# и завершает транзакцию с механизмом прекращения работы мастером (смотрите раздел 3.3.3.1.).
Адресат должен делать полную декодировку перед запуском / выставлением DEVSEL# или любым другим выходным сигналом. Запрещено управлять DEVSEL# до полной декодировки, а затем позволять декодирующие действия на шине. (Это могло бы вызывать конкуренцию.) Адресат должен определить строки AD с FRAME# прежде, чем может быть выставлен DEVSEL# на других командах. Адресат должен определить IDSEL с FRAME# и AD[1:: 0] прежде, чем DEVSEL# может быть выставлен в команде конфигурации.
Ожидается, что большинство (возможно, все) выходные устройства будут способны декодировать и выставлять DEVSEL# внутри одного или двух тактов FRAME# (быстро и средне на рисунке).
Соответственно, декодирующий агент, может обеспечивать устройство зависимым регистром конфигурации, который может программироваться одним или двумя фронтами такта, в котором устройство производит выборку DEVSEL#, позволяя более быстрый доступ к шине расширения. Использование такой опции ограничено самым медленным декодирующим агентом на шине.
Однажды выставленный DEVSEL# не может быть снят, пока последняя фаза данных не завершена, за исключением сообщения об аварийном прекращении работы.
Если первый доступ отображается в адресный интервал адресата, он выставляет DEVSEL#, чтобы требовать доступа. Но если мастер пытается продолжать доступ, пересекая границы ресурса, адресат требует сигнала разъединения.
Когда адресат требует доступа для операций ввода/ вывода, и есть возможность указать один или большее байтов для доступа вне адресного интервала адресата, он должен сообщить об аварийном прекращении работы. При наличии проблем с этим типом ввода/вывода, декодирующее устройство (мост шины расширения) может поступать следующим образам:

- Делать положительное декодирование (включением карты байта) на адресах, которые различные устройства совместно используют общие DWORDs, дополнительно используя байт. Это дает возможность обнаружить эту проблему и выдать сигнал аварийного прекращения работы.
- Передать полный доступ к шине расширения, где у части доступа, которая не может быть обслужена спокойно, понизится приоритет. (Это возможно только когда адресат постоянно находится на шине расширения, а другой находится на PCI.)

3.6.2. Специальный цикл (Special Cycle)

Команда Special Cycle обеспечивает простой механизм передачи сообщения на PCI. В дополнение к (поддерживающему связь) состоянию процессора (который выполнен на шинах процессора Intel) она может также использоваться для логической передачи сигналов между PCI агентами, когда такая передача сигналов не требует точной синхронизации или синхронизации физических сигналов.

Хорошей парадигмой команды Special Cycle является "логический провод", который только сообщает об одиночных тактовых импульсах; то есть может использоваться, чтобы устанавливать и сбрасывать сигналы в реальном времени, подразумевая, что доставка его гарантируется. Это позволяет проектировщику определять необходимую связь без того, чтобы требовать дополнительные контакты. Как с передачей сигналов вообще, реализация поддержки команды Special Cycle необязательная.

Команда Special Cycle не содержит никакого явного целевого адреса, но передается всем агентам. Каждый агент, получивший сообщение, должен определить, является ли оно применимым к нему. PCI агенты никогда не выставляют DEVSEL# в ответ на команду Special Cycle.

Это означает, что не имеется никакого целевого подтверждения связи любого вида на этих транзакциях, и вычитающие декодирующие мосты не должны передать эту операцию шины на их вторичную шину. Команды Special Cycle не будут распространяться поперек мостов.

Команда Special Cycle может содержать сообщение зависимых данных, которые не интерпретированы PCI секвенсором непосредственно, но переданы по мере необходимости к аппаратным средствам, связанным с PCI секвенсором. В большинстве случаев явно адресованные сообщения должны быть обработаны в одном из трех физических адресных пространств на PCI.

Команда Special Cycle подобна любой другой команде шины, где имеется фаза адреса и фаза данных. Фаза адреса начинается подобно всем другим командам с выставления FRAME# и завершается подобно всем другим командам, когда FRAME# и IRDY# снимаются. Уникальность этой команды по сравнению с другими - это то, что никакой агент не отвечает выставлением DEVSEL#. Эта команда, в основном, передает всем агентам, принимает команду агента и обрабатывает запрос.

Фаза адреса не содержит никакую важную информацию по сравнению с полем команды. Там нет никакого явного адреса. В течение фазы данных AD[15::00] содержится тип сообщения и необязательное поле данных. Сообщение закодировано на младшие 16 бит, а именно AD[15::00]. Необязательное поле данных закодировано на старшие 16 бит, а именно AD[31::16] и не используется во всех сообщениях. Мастер команды Special Cycle может вставлять состояния ожидания подобно любой другой команде, в то время как адресат не может. Сообщение и связанные данные имеют значение только на первых тактах IRDY#. Информация содержится и в синхронизации последующих фаз данных.

В течение фазы адреса C/BE [3::0]# = 0001 (Special Cycle) и AD[31::00] содержат случайные значения и должны игнорироваться. В течение фазы данных C/BE [3::0]# выставлен и AD[31::00] содержит:

AD[15::00]	Закодированное сообщение
AD[31::16]	Сообщение зависимых полей данных

Секвенсор PCI шины начинает эту команду подобно всем другим и завершает ее с аварийным прекращением работы. Аппаратное приложение обеспечивает всю информацию подобно любой другой команде и запускает секвенсор шины. Когда секвенсор сообщает, что доступ завершен аварийным прекращением работы, аппаратное приложение считает доступ завершенным. В этом случае бит Received Master Abort в конфигурации регистра состояния (Раздел 6.2.3.) должен быть не установлен. Самая быстрая команда Special Cycle может завершиться за 5 тактов. Один дополнительный такт требуются для оборотного цикла перед следующим доступом. Следовательно, всего 6 PCI тактов требуется от начала Special Cycle до начала другого доступа.
Общий объем сообщений составляет 64КБ. Коды сообщений определены и описаны в Приложении А.

3.6.3. Пошаговая передача адреса / данных

Способность агента к установке сигналов больше, чем за несколько тактов, называется «продвижением» или «пошаговой передачей». Это понятие позволяет агенту со "слабыми" буферами вывода выставлять набор сигналов важного состояния более, чем за несколько тактов (непрерывное продвижение), таким образом, уменьшается текущая загрузка, сгенерированная каждым буфером. Альтернативный подход позволяет агенту с "сильными" буферами вывода управлять подмножеством их на каждом из нескольких тактов, пока они не все управляются (дискретное продвижение), таким образом, уменьшив количество сигналов, которые должны быть включены одновременно.

Любой подход позволяет агенту изменять соотношение эффективность/стоимость. При использовании непрерывного продвижения нужно избегать взаимосвязи между управляющими сигналами, которые должны быть на каждом фронте такта, и сигналами продвижения, которые изменяться на срезе такта. Эффективные периферийные устройства должны разрешить этот вопрос в пользу низкой стоимости.

Продвижение разрешается только на AD[31:: 00], AD[63:: 32], PAR, PAR64# (для 64-разрядных передач данных, но не для команды DAC), и выводов IDSEL, потому что они всегда ограничиваются сигналами управления: то есть эти сигналы имеют силу только по фронту такта, на котором они используются. AD ограничен сигналом FRAME# в фазах адреса, и IRDY# или TRDY# - в фазах данных (в зависимости от направления перемещения данных). PAR неявно ограничивается на каждом фронте такта, после которого были использованы AD. IDSEL ограничен комбинацией сигнала FRAME# и дешифрированной команды конфигурации.

Рисунок 3-14 иллюстрирует задержку выставления FRAME# , до управления всеми адресными линиями AD. Мастеру разрешается (и требуется) управлять AD и C/BE# и шиной IDLE. Но может потребоваться много тактов, чтобы управлять имеющим значение адресом перед выставлением FRAME#. Однако, при задержке выставления FRAME# мастер может потерять свое обращение к шине. Как и в случае работы с любым мастером, сигнал GNT# должен быть выставлен на фронте прежде, чем будет выставлен сигнал FRAME#. Если GNT# был деактивирован, на фронте отмечено "A", мастер требует немедленно выставить сигналы в третье состояние, потому что арбитр предоставил шину другому агенту (новый мастер имел бы более высокий приоритет). Если GNT# был деактивирован (на фронте такта отмечено "B" или "C"), то FRAME# будет уже установлен, и транзакция продолжится.

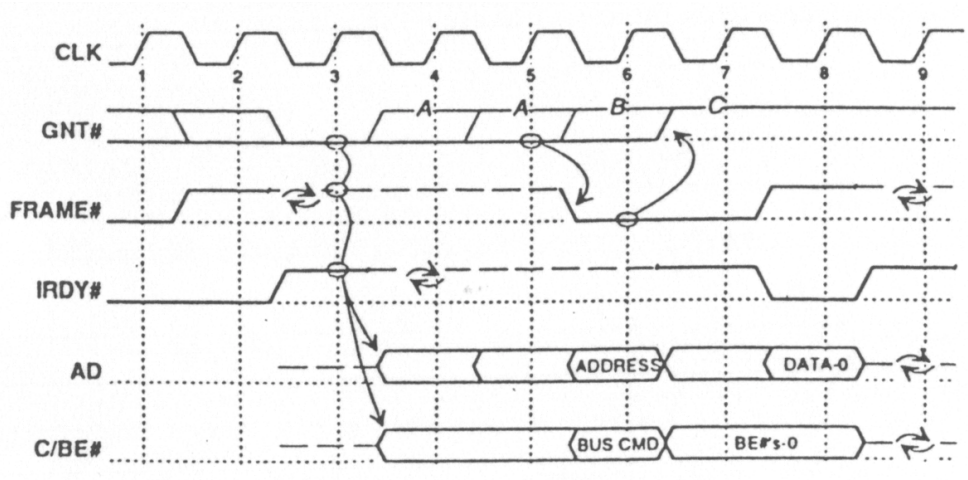


Рисунок 3-14: Продвижение адреса

3.6.4. Цикл конфигурации

PCI обеспечивает полную программно управляемую инициализацию и конфигурацию через отдельное адресное пространство конфигурации. PCI устройства должны обеспечить 256 байтов регистров конфигурации для этой цели. Описания регистра можно найти в главе 6. Этот раздел описывает команды PCI-шины для доступа к пространству конфигурации. Как обсуждалось ранее, каждое устройство дешифрирует собственные адреса для нормального доступа. Однако доступ в адресном пространстве конфигурации требует выбирающего декодирования, которое должно быть выполнено извне, и сообщено на PCI устройство через IDSEL выход, который функционирует подобно классическому сигналу "выбор микросхемы". PCI устройство - является адресатом команды конфигурации (чтения или записи) только в случае, если выставлен IDSEL и AD[1:0] - "00" в течение фазы адреса команда. Внутренняя адресация 64-DWORD регистрового пространства выполняется по AD[7:: 2]. Команда конфигурации подобно другим командам позволяет доступ байта, слова, DWORD, операции пакета. Остальная часть транзакции осуществляется также, как другие команды, включая всю семантику завершения. Если никакой агент не отвечает, запрос завершается аварийным прекращением работы (Раздел 3.3.3.1.). Интерфейс шины расширения не должен передавать этот запрос через шину расширения.

Сигналом IDSEL управляется интерфейс главный компьютер / память или проектировщик системы. Однако этот сигнал выбора был разработан, чтобы позволить подключение с одним из старших 2l линии адреса, которые не используются иначе в доступе конфигурации. Не существует никакого известного или стандартного пути определения IDSEL из старших 21 битов адреса: следовательно, сигнал IDSEL должен быть поддержан. Устройства не должны делать внутреннее подключение между линией и внутренним сигналом IDSEL, чтобы сохранить сигнал. Единственная исключительная ситуация - первичный мост шины, так как это определяет выставлен ли IDSEL. AD[3l::00] линии должны быть активно управляемы в течение фазы адреса. Подключая различные линии адреса на каждое устройство и выставляя один из AD[31:11] линии в это время, 21 различные устройства могут быть однозначно выбраны для доступа конфигурации.

При этом подходе (соединять одну из старших 21 линий с IDSEL) может возникать такая проблема, как дополнительная загрузка на линии. Это может быть исправлено сопротивлением, соединяющим IDSEL и соответствующую линию. Это, однако, приводит к очень медленной скорости на IDSEL, и поэтому он может быть в недопустимом логическом состоянии большую часть времени, как показано на Рисунке 3-15 метками "XXXX". Однако так как он используется только на фазе адреса цикла конфигурации, шина адреса может быть настроена несколькими тактами перед FRAME#[9] , таким образом гарантируя, что IDSEL будет в рабочем состоянии, когда он понадобится. Для всех других циклов IDSEL не определен и может быть в неопределенном состоянии в течение фазы адреса. Настройка шины адреса подобна адресному продвижению, как было описано в предыдущем разделе.

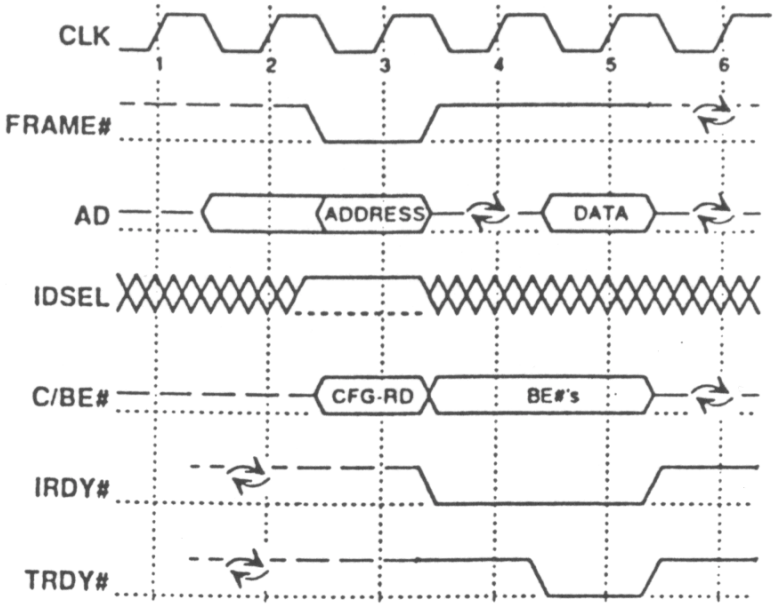


Рисунок 3-15: Чтение конфигурации

Для поддержки иерархии PCI шины используются два типа доступа конфигурации. Они имеют следующие форматы, которые показывают состояние AD линий в течение фазы адреса доступа конфигурации:

31 — 11	10 — 8	7 — 2	1 — 0	
Reserved	Function Number	Register Number	0	0

Type 0

31 — 24	23 — 16	15 — 11	10 — 8	7 — 2	1 — 0	
Reserved	Bus Number	Device Number	Function Number	Register Number	0	1

Type 1

[9] Число тактов, за которое шина адреса должна быть настроена, определяется из RC константы времени на IDSEL.

Тип 1 и тип 0 доступа конфигурации различаются значениями на AD[1::0]. Тип 0 цикла конфигурации (когда AD[1::0] = "00") используется, чтобы выбрать устройство на PCI шине, где цикл выполняется. Тип 1 цикла конфигурации (когда AD[1::0] = "01") используется, чтобы передать запрос конфигурации на другую PCI шину.

Поля Register Number и Function Number имеют то же самое значение для обоих типов конфигурации, тогда как Device Number и Bus Number используется только в 1-ом типе доступа. Зарезервированные поля должны игнорироваться адресатами.

Register number	закодированное значение, используемое, чтобы индексировать DWORD в пространстве конфигурации адресата.
Function Number	закодированное значение, используемое, чтобы выбрать одну из восьми возможных функций на устройстве.
Device Number	закодированное значение, используемое, чтобы выбрать 1 из 32 устройств на данной шине. (Есть только 21 устройства, которые могут быть выбраны, с помощью *IDSEL* AD[3l::11] линий).
Bus Number	закодированное значение, используемое, чтобы выбрать 1 из 256 шин в системе.

Интерфейсы (и главный компьютер и PCI-to-PCI), которые генерируют Тип 0 цикла конфигурации, используют Device Number , чтобы выбрать выставление IDSEL. Function Number обеспечивается на AD[10::08]. Register Number обеспечивается на AD[7::2]. AD[1::0] должен быть "00" для Типа 0 доступа конфигурации.
Тип 0 доступа конфигурации не работает вне локальной PCI шины и должен требоваться локальным устройством или завершаться аварийным прекращением работы.
Если адресат доступа конфигурации находится на другой шине (не локальной PCI шине), должен использоваться тип 1 доступ конфигурации. 1 тип доступа игнорируется всеми адресатами за исключением мостов PCI-to-PCI. Эти устройства декодируют поле Bus Number , чтобы определить, находится ли адресат доступа конфигурации позади моста. Если Bus Number не для шины позади моста, доступ игнорируется. Мост требует доступа, если адресат на шине позади моста. Если Bus Number не на вторичной шине моста, доступ просто передается неизменяемым. Если Bus Number соответствует вторичному номеру шины, мост преобразовывает доступ в Тип 0 доступа конфигурации. Мост изменяет AD[1::0] на "00" и передает AD[10:: 02] неизмененным. Device Number декодируется, чтобы выбрать 1 из 32 устройств на локальной шине. Мост выставляет правильный IDSEL и инициализирует доступ конфигурации.

Устройства, которые отвечают на 0 тип тактов конфигурации, разделяются на два класса. Класс первый (одиночное функциональное устройство) определен для обратной совместимости, и использует только IDSEL и AD[1::0] ("00") чтобы определить, отвечать или нет. Второй класс устройств (многофункциональное устройство) использует поля Function Number, IDSEL, AD[1::0] ("00"), а также закодированное значение AD[10:: 08] чтобы определить, отвечать или нет. Два класса отличаются кодированием в верхней части пространства конфигурации.

Многофункциональные устройства требуют делать полную декодировку AD[10::08] и отвечают, только если они выполнили регистры пространства конфигурации для выбранной функции. Они также требуют всегда выполнять функцию 0 в устройстве. Выполнение других функций необязательно и может быть назначено в любом порядке (т.е. функцию двухфункциональное устройство может ответить на функцию 0, но может выбирать любую из других возможных функциональных номеров (1-7) для второй функции).

Код конфигурации исследует шину в порядке Device Number (то есть Function Number будет нулевой). Если одиночное функциональное устройство обнаружено, то Device Number больше не будет проверяться. Если обнаружено многофункциональное устройство, то все Device Number далее будут проверены.

3.6.4.1. Генерация циклов конфигурации

Системы должны обеспечить механизм, который позволяет PCI циклам конфигурации быть сгенерированным программным обеспечением. Этот механизм обычно размещен в главном мосте. Для PC-AT совместимых систем, механизм генерации циклов определен ниже. Для другой архитектуры спецификации не существует.
Для PC-AT совместимых машин имеются два механизма, позволяющие программному обеспечению генерировать циклы конфигурации. Они упоминаются как Механизм Конфигурации #1 и Механизм Конфигурации #2. Механизм Конфигурации #1 - предпочитаемая реализация, и должен обеспечиваться всеми будущими ведущими мостами (существующие мосты должны преобразоваться тоже, если это возможно). Механизм Конфигурации #2 определен для обратной совместимости и не должен использоваться в соответствии в новых проектах[10] . Ведущие мосты, которые нужно использовать в PC-AT совместимых системах должны выполнять по крайней мере, один из этих механизмов.

3.6.4.1.1. Механизм конфигурации #1

Два DWORD расположения ввода/вывода используются в этом механизме. Первое DWORD по адресу (CF8H) ссылается на регистр чтения/записи, который называется CONFIG_ADDRESS. Второй DWORD адрес (CFCh) ссылается на регистр, называемый CONFIG_DATA. Общий механизм для доступа к пространству конфигурации должен писать значение в CONFIG_ADDRESS, который определяет PCI шину, устройство на этой шине, и регистр конфигурации в устройстве, к которому обращаются. Читают или пишут CONFIG_DATA, затем заставляют мост транслировать этот CONFIG_ADDRESS к запрошенному циклу конфигурации на PCI шине.

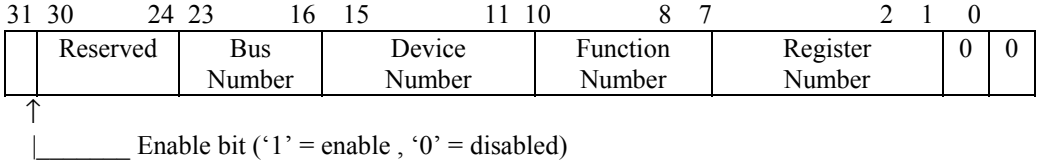


Рисунок 3-16: Размещение Регистра CONFIG_ADDRESS

CONFIG_ADDRESS - 32-битный регистр, показанный на Рисунке 3-16. Бит 31 - флажок для определения, когда доступ к CONFIG_DATA должен транслирован циклам конфигурации на PCI шине. Биты от 30 до 24 зарезервированы, они только для чтения, и должны возвратить нули при чтении. Биты от 23 до 16 выбирают PCI шину в системе.

[10] Этот механизм значительно снижает эффективность, когда используется в мультипроцессорной системе. Операционная система и драйверы должны сотрудничать, чтобы гарантировать взаимоисключающий доступ к адресному интервалу ввода/вывода C000h-CFFFh, и пространству конфигурации и устройству доступов ввода/вывода. Подходящий механизм синхронизации трудно добавить в существующие мультипроцессорные операционные системы /драйверы, где в настоящее время драйверы управляются прямым доступом к их пространству ввода-вывода.

Биты с 15 до 11 выбирают специфическое устройство на шине. Биты с 10 до 8 выбирают специфическую функцию в устройстве (если устройство поддерживает многократные функции). Биты с 7 до 2 выбирают DWORD в пространстве конфигурации устройства. Биты 1 и 0 только для чтения и должны возвратить 0 при чтении.

Всегда, когда ведущий мост видит, что полное DWORD ввод/вывод пишется в CONFIG_ADDRESS, мост должен задержать данные в регистре CONFIG_ADDRESS. На полном DWORD ввода/вывода читается в CONFIG_ADDRESS, мост должен возвратить данные в CONFIG_ADDRESS. Любые другие типы доступа к этому адресу (не DWORD) должны обработаться подобно нормальному доступу ввода/вывода, и не должно выполняться никакое специальное действие. Следовательно, только пространство ввода/вывода использованное этим регистром - DWORD в выданном адресе. Устройства ввода/вывода, использующие регистры BYTE или WORD, не действуют, потому что они будут передаваться неизмененными.

Когда мост видит доступ ввода/вывода, который попадает внутрь DWORD, начиная с адреса CONFIG_DATA, он проверяет допускающийся бит и BUS NUMBER в регистре CONFIG_ADDRESS. Если транзакция цикла конфигурации допускается и BUS NUMBER соответствует номеру шины мостов или любому номеру шины позади моста, тогда трансляция цикла конфигурации должна быть выполнена.

Имеются два типа трансляции, которые могут происходить. Первый, 0 тип - трансляция, когда адресуемое устройство находится на PCI шине, связанной с мостом. Второй, тип 1, происходит, когда устройство находится на другой шине где-нибудь позади этого моста.

Для трансляций типа 0 (см. Рисунок 3-17), мост декодирует поле DEVICE NUMBER, чтобы выставить соответствующий IDSEL[11] и выполняет цикл конфигурации на PCI шине, где AD[1::0] = "00". Биты 10 - 8 из CONFIG_ADDRESS скопированы в AD[10::8] на PCI шину как закодированное значение, которое может использоваться компонентами, которые содержат многократные функции. AD[7::2] также скопирован из регистра CONFIG_ADDRESS. Рисунок 3-17 показывает трансляцию из регистра CONFIG_ADDRESS на AD линии на PCI шине.

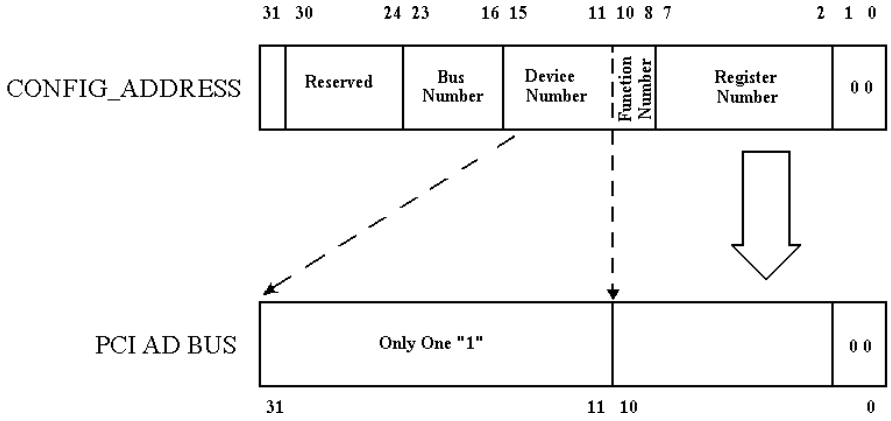


Рисунок 3-17: Трансляция типа 0 через интерфейс циклов конфигурации

Для трансляции Типа 1, интерфейс непосредственно копирует содержание регистра CONFIG_ADDRESS на AD линии в течение фазы адреса цикла конфигурации, устанавливающим, что AD[1:: 0] - "01".

[11] Поле DEVICE NUMBER выбирает линию IDSEL, которую мост не выполняет, мост должен дать доступ процессору, пропустить запись данных и повторить чтение. Только при выполнении с учетом доступа к конфигурации Типа 0 IDSEL не выставляется. При этом ситуация заканчивается прекращением работы мастера, который теряет записываемые данные и возвращает данные, предназначенные для чтения.

В обоих случаях разрешающий байт для передач данных должен быть непосредственно скопирован с шины процессора.

Для систем с равноправными мостами на шине процессора, один мост обычно всегда предназначен для доступа к регистру CONFIG_ADDRESS. Другие мосты берут данные, записанные в этот регистр. Доступ к регистру CONFIG_DATA обычно имеет мост, делающий трансляцию конфигурации.

Ведущие мосты и мосты PCI-to-PCI обычно требуют двух регистров конфигурации, и их содержание используется для того, чтобы определить, когда мост делает трансляцию цикла конфигурации. Один регистр (BUS NUMBER) определяет номер PCI шины непосредственно позади моста, другой регистр (SUBODINATE BUS NUMBER) определяет номер последней иерархической шины позади моста[12] . код POST отвечает за инициализацию этих регистраторов.

3.6.4.1.2. Производство специальных циклов с механизмом конфигурации #1

Этот раздел описывает, как ведущие мосты, которые выполняют Механизм Конфигурации #1 для доступа к пространству конфигурации, должны позволить программному обеспечению генерировать Специальные Циклы. Ведущие мосты не должны обеспечивать механизм для разрешения программному обеспечению генерировать Специальные Циклы.

Когда в регистр CONFIG_ADDRESS записано такое значение, что BUS NUMBER соответствует номеру шины мостов, DEVICE NUMBER - все 1, FUNCTION NUMBER - все 1, REGISTER NUMBER имеет значение 0, тогда мост выполняет Специальный Цикл после того, как регистр CONFIG_DATA записан. Когда регистр CONFIG_DATA записан, мост генерирует кодирование Специального Цикла на выходе C/BE [3:: 0]# в течение цикла адреса и управляет данными из ввода/вывода на AD[3l:00] в течение первого цикла данных. Чтение CONFIG_DATA, после того, как CONFIG_ADDRESS был установлен описанным выше способом, имеет неопределенные результаты. Мост может обрабатывать это как нормальную операцию цикла конфигурации (то есть генерировать Тип 0 цикла конфигурации на PCI шине). Это аварийное прекращение работы, и процессору будет возвращены все 1.
Если поле BUS NUMBER CONFIG_ADDRESS не соответствует номеру шины моста, то мост пишет в CONFIG_DATA как Тип 1 цикла конфигурации, точно так же, когда не соответствует номер шины.

3.6.4.1.3. Механизм конфигурации #2

Этот механизм для доступа к пространству конфигурации обеспечивает режим, который отображает PCI-пространство конфигурации в 4Кб пространства ввода/вывода центрального процессора. Когда режим установлен, любой доступ центрального процессора внутри адресного интервала ввода/вывода C000h- CFFFh будет транслироваться к PCI циклу конфигурации. Когда режим установлен для отключения PCI-пространства конфигурации, все доступы ввода/вывода центрального процессора в этом диапазоне будут направлены к соответствующему порту ввода/вывода в системе. Этот механизм не поддерживает равноправные мосты на шине процессора

[12] Ведущие интерфейсы не позволяют равноправным интерфейсам не использовать любой из этих регистров, так как шина позади этого интерфейса, по определению, шина 0 и все другие PCI шины привязаны к шине 0.

В данном механизме используются два регистра. Эти регистры описаны ниже.

Регистр для доступа к пространству конфигурации (CSE регистр):

Пространство конфигурации отображено в пространстве ввода/вывода с помощью записи в CSE регистр, размещенный в расположении ввода/вывода CF8H. Поля в CSE регистре показаны на рисунке 3-18.

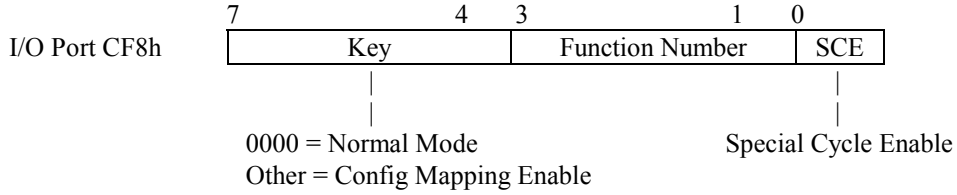


Рисунок 3-18: Размещение регистра разрешения пространства конфигурации

Этот регистр чтения/записи порта ввода/вывода, который логически находится в ведущем мосте. Бит 0 - разрешающий бит для генерации Специальных Циклов. Этот бит должен быть установлен в 0, чтобы генерировать циклы конфигурации и установлен в 1, чтобы генерировать Специальные Циклы. Раздел 3.6.4.1.4. описывает Специальные Циклы с Механизмом Конфигурации * 2. Биты от 1 до 3 обеспечивают функциональный номер для цикла конфигурации. Эти три бита перемещены в AD[l0:: 08], когда цикл конфигурации сгенерирован. Поле ключа используется для того, чтобы разрешить функцию отображения. Ведущий мост отвечает на чтение CSE регистра, возвращая последние данные, записанные в этот регистр. Все доступы к CSE регистру должны быть одиночные байтовые операции.

После сброса регистр CSE очищен, и ведущий мост устанавливается в заданное по умолчанию состояние и обрабатывает все доступы ввода/вывода обычно.

Forward - регистр:

Forward регистр, размещенный на адресе ввода/вывода CFAh используется, чтобы определить то, к чему на PCI шине обращаются. Это регистр чтения/записи, инициализирован в 0 при сбросе, и возвращает последнее записанное значение при чтении. Когда Forward регистр установлен в 00, значит обращаются к шине сразу позади моста, и тип 0 доступа конфигурации сгенерирован. Когда Forward регистр отличен от нуля, сгенерирован тип 1 доступа конфигурации, содержание Forward регистра отображено в AD[23:: l6] в течение фазы адреса цикла конфигурации.

Отображение пространства конфигурации:

Когда мосту разрешают делать отображение пробела конфигурации (то есть поле ключа CSE регистра отлично от нуля) мост должен преобразовать все доступы ввода/вывода в диапазоне C000h- CFFFh в PCI циклы конфигурации. Шестнадцать PCI устройств можно адресовать с помощью битов 11:: 8 из адреса ввода/вывода. Биты 7::2 адреса Ввода - вывода выбирают DWORD внутри пространства конфигурации устройства.

Рисунок 3-19 показывает трансляцию, сделанную, когда Forward регистр нулевой. Это указывает, что устройство, к которому обращаются, находится на PCI шине 0 непосредственно позади моста. Трансляция вызывает тип 0 цикла конфигурации.

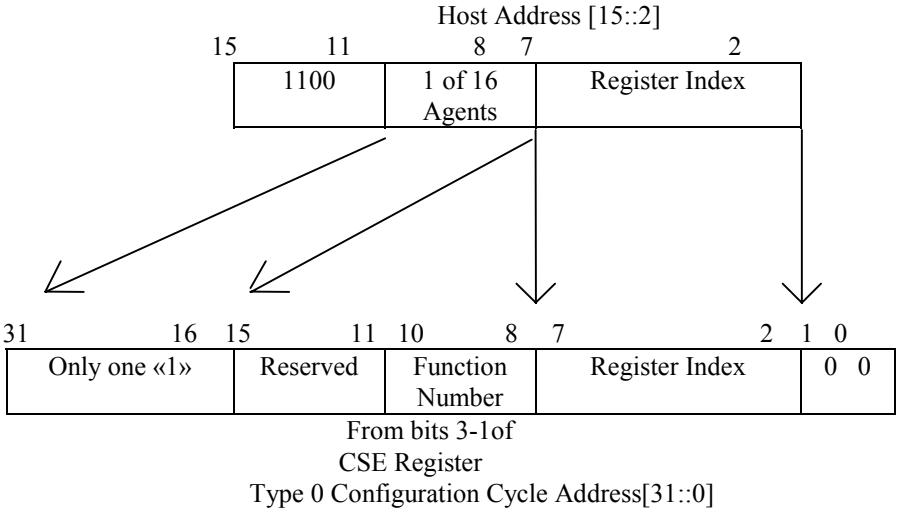


Рисунок 3-19: Трансляция типа 0 цикла конфигурации

Рисунок 3-20 показывает трансляцию, сделанную, когда Forward регистр отличен от нуля. Это указывает, что устройство, к которому обращаются, находится на PCI шине в другом месте, чем непосредственно позади моста. Мост должен генерировать тип 1 конфигурации цикла и отображать Forward регистр в AD [23:: 16] PCI Шины.

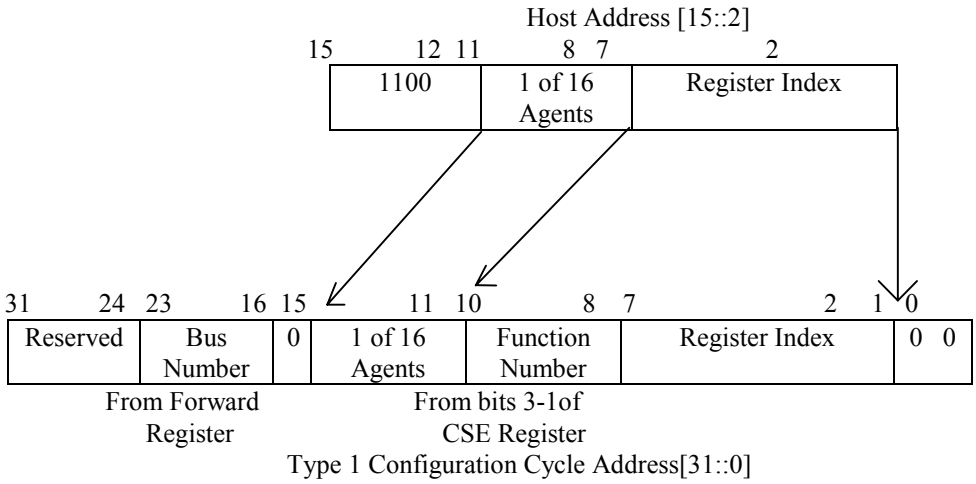


Рисунок 3-20: Трансляция типа 1 цикла конфигурации

3.6.4.1.4. Производство специальных циклов с механизмом конфигурации #2

Этот раздел определяет, как ведущие мосты, которые выполняют Механизм Конфигурации #2 для доступа к пространству конфигурации должны позволить программному обеспечению генерировать Специальные Циклы. Ведущие мосты не должны обеспечивать механизм для разрешения программному обеспечению генерировать Специальные Циклы.

Когда CSE регистр установлен так, чтобы бит 0 был "1", поле Function Number все 1, и поле ключа отлично от нуля, тогда мосту разрешено делать Специальный Цикл или Тип 1 цикла конфигурации на PCI шине.
Когда Специальный Цикл разрешен, и центральный процессор адресует для записи ввода/вывода адресует CF00H, мост сравнивает содержание Forward регистра с 00. Если содержание Forward регистра 00, то ведущий мост генерирует Специальный Цикл на PCI шине. В течение фазы адреса мост генерирует кодирование Специального Цикла на C/BE [3:: 0]# и управляет записью данных из ввода/вывода (CF00H) на AD[31:00] в течение первой фазы данных Специального Цикла. Если содержание Forward регистра не равно "0", то Ведущий Мост генерирует Тип 1 цикла конфигурации на PCI шине с полями Device Number и Function Number равными 1 (AD[15::08] будут все 1), и Register Number будет 00h (AD[7::2] будут все 0) в течение фазы адреса PCI конфигурации записывает цикл.
Доступ для чтения к вводу/выводу адреса CXXXh, пока CSE регистр, разрешенный для Специальных Циклов, будет иметь неопределенные результаты. Доступы для записи к вводу/ выводу адресуют CXXXh (если бы не CF00H) в то время как CSE регистр, разрешенный для Специальных Циклов, будет иметь неопределенные результаты. В CSE регистре, всякий раз, когда бит Разрешения Специального Цикла (SCE) установлен, и поле Function Number не все 1, доступ ввода/вывода в диапазоне CXXXh будет иметь неопределенные результаты.

3.6.5. Подтверждение прерывания

PCI шина поддерживает цикл подтверждения прерывания как показано на 3-21 Рисунке. Этот рисунок иллюстрирует цикл прерывания x86 на PCI, и приведен только как пример. В течение фазы адреса, AD[31:: 00] не содержит имеющий силу адрес, но должен управляться с устойчивыми данными, PAR имеет силу, и четность может быть проверена. Только одиночный агент может отвечать на подтверждение прерывания. Устройство, которое делает так, должно выставить DEVSEL#. Вектор должен быть возвращен, когда выставлен TRDY#. Цикл подтверждение прерывания подобен любому другому циклу, в котором циклы могут быть вставлены, и запрос может быть завершен, как обсуждено в Разделе 3.3.3.2.

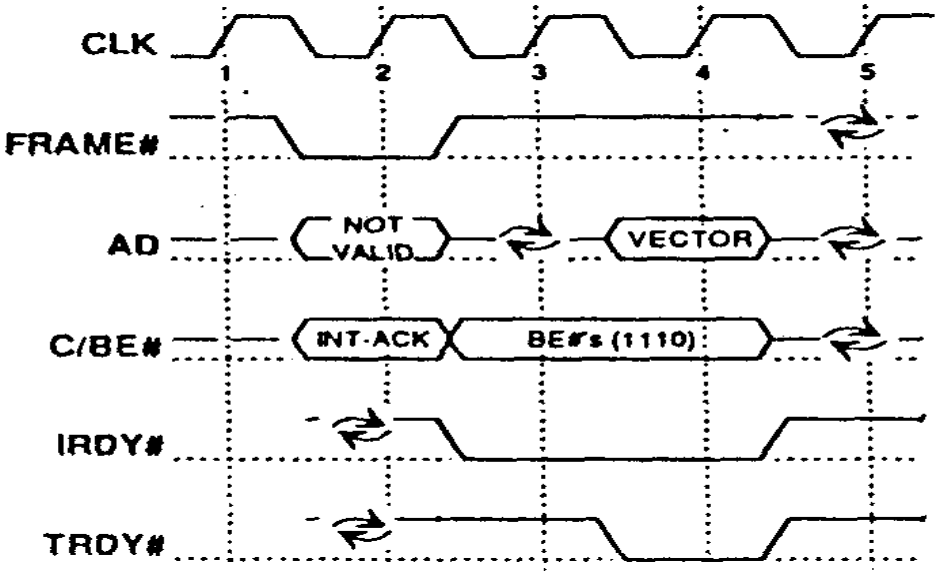


Рисунок 3-21: Подтверждение прерывания

В отличие от традиционного 8259 двойного цикла подтверждения, PCI выполняет одиночное подтверждение. Преобразование из формата процессорных двух циклов в PCI формат цикла легко выполняется в мосте, отбрасывая первое подтвержденное прерывание из процессора.

3.7. Функции ошибки

PCI обеспечивает обнаружение ошибок четности и других ошибок системы. Зона действия PCI ошибки может поступать из устройств, для которых не имеют никакого интереса обнаруженные ошибки (особенно ошибки четности) к агентам, которые обнаруживают, сообщают и исправляют ошибки. Это позволяет агентам, которые исправляют ошибки четности, избежать воздействия на операцию агентов, которые этого не делают. Чтобы позволить эту гибкость, генерация четности требуется на всех транзакциях всеми агентами. Обнаружение и сообщение ошибок требуется, с некоторыми исключениями, для некоторых классов PCI агентов, как перечислено ниже.

Обсуждение ошибок разделено в следующие два раздела, покрывающие генерацию четностей, обнаружение и сообщение об ошибке. Каждый раздел объясняет, что является необязательными, а что требуется для каждой функции.

3.7.1. Контроль по четности

Контроль по четности на PCI обеспечивает механизм для определения транзакцию транзакцией, если мастер достиг успеха в достижении желательного адресата и если данные, перемещаемые между ними, проходят правильно. Чтобы гарантировать, что выполняется правильная операция шины, четыре командных строки включены в вычисление четности. Чтобы гарантировать, что правильные данные перемещены, допускается также включение 4 байтов в вычисление четности. Агент, который является ответственным за запуск AD[3l::00] на любой данной фазе шины, также ответственен за запуск проверки на четность на PAR.

В течение фазы адреса и фазы данных, четность покрывает AD[31::00] и C/BE [3::0]# линии независимо от того, несут или нет все линии значимую информацию. В течение конфигурации, Специального Цикла, или подтверждения прерывания, команды некоторых (или всех) линий адреса не определены, но требуется довести их до устойчивых значений и включить в вычисление четности.

Четность сгенерирована согласно следующим правилам:

- Четность вычислена одинакова на всех PCI транзакциях независимо от типа или формы.
- Число «1» на AD[31:: 00], C/BE [3:: 0]#, и PAR равняется четному числу.
- Четность - генерируется обязательно; это должно быть выполнено всеми PCI совместимыми устройствами.

На любой данной фазе шины, PAR управляется агентом, который управляет AD[31::00] и отстает от соответствующего адреса или данных на один такт. Рисунок 3-22 иллюстрирует чтение и запись транзакции с четностью. Мастер управляет PAR для фаз адреса на тактах 3 и 7. Адресат управляет PAR на фазе данных на транзакции чтения (такт 5), в то время как мастер управляет PAR на фазе данных на транзакции записи (такт 8). Обратите внимание, что один такт отстает от другого. PAR ведет себя подобно AD[3l::00], включая состояния ожидания и оборотные циклы.

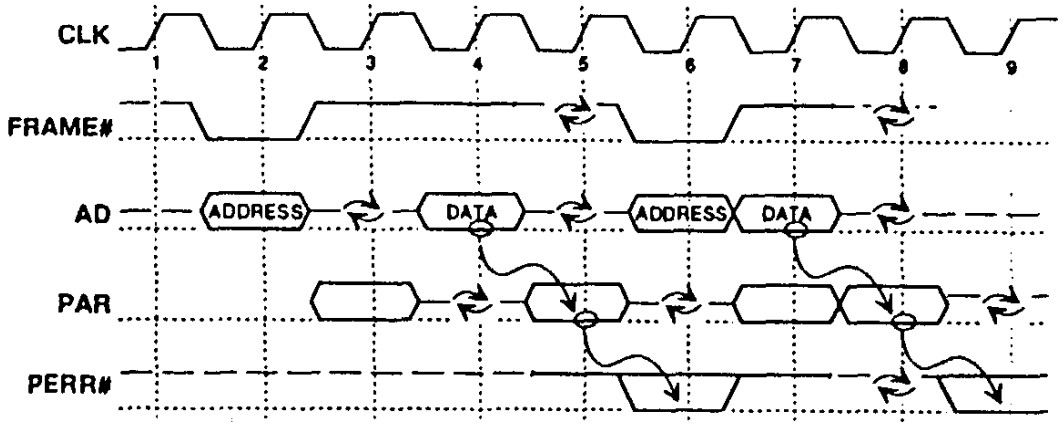


Рисунок 3-22: Операция четности

Четность должна быть проверена, чтобы определить, успешно ли адресовал мастер желательному адресату и переместились ли данные правильно. Проверка четности на PCI требуется за исключением двух классов устройств, перечисленных в Разделе 3.7.2. Агенты, которые поддерживают проверку четности, должны всегда устанавливать Обнаруженный бит Ошибки Четности в пространстве конфигурации регистр Состояния (обратитесь к Разделу 6.2.3.) когда ошибка четности обнаружена. Любое дополнительное действие вне установки этого бита обусловлено установкой бита Ответа Ошибки Четности в пространстве конфигурации регистра команд и выдается в раздел ошибок.

Любой агент может проверять и сообщать об ошибке четности адреса на SERR#. Только мастер может сообщать ошибку четности данных чтения и только выбранный адресат может сообщать об ошибке четности данных записи.

3.7.2. Сообщения об ошибках

Как упомянуто ранее, PCI обеспечивает обнаружение и передачу сигналов четности и других ошибок системы. Предназначается, что ошибки четности будут сообщены через доступ и цепочку драйверов устройств всякий раз, когда это возможно. Эта цепочка сообщения об ошибке от адресата к мастеру шины, к драйверу устройства, к администратору устройства, на операционную систему предназначена, чтобы позволить параметрам восстановления при ошибках быть выполненными на любом уровне. Так как не возможно сопоставить ошибки системы со специфической цепочкой доступа, они сообщаются непосредственно на уровень системы.

Два сигнала используются на PCI схеме сообщения об ошибке. PERR# используется исключительно для сообщения об ошибках четности данных на всех транзакциях, за исключением команд Специального Цикла. Протокол шины гарантирует, что PERR# одновременно не будет никогда управляться многократными агентами шины.

SERR# используется для другой передачи сигналов ошибки, включая четность адреса и четность данных на команде Специального Цикла, и может необязательно использоваться на любой другой ошибке системы. Это открытый сигнал утечки, который является соединением со всеми другими PCI агентами и, следовательно, может одновременно управляться многократными агентами. Так как открытая передача сигналов утечки не может гарантировать устойчивые сигналы на каждом фронте такта, как только выставляется SERR#, его логическое значение должно быть принято неопределенным, пока сигнал не установится в выключенное состояние по крайней мере, на два такта.

И PERR# и SERR# являются требуемыми сигналами, так как требуется передача сигналов ошибки четности на PCI. Однако это требование отклонено для двух классов устройств.

1. Устройства, которые разработаны исключительно для использования на плоских системных платах; например, наборы чипов. Продавцы систем имеют контроль над использованием этих устройств, пока они не будут появляться на сетевых платах.
2. Устройства, которые никогда не имеют дело с любыми данными, которые представляют постоянную или остаточную системы состояния прикладной программы, например, человеческого интерфейса и видео / аудио устройств. Эти устройства только касаются данных, которые являются временным представлением (например, пиксели) постоянных или остаточных систем или состояний прикладной программы, и следовательно, не склонны создавать проблемы целостности системы в случае необнаруженного сбоя.

Обратите внимание, что все агенты требуют генерировать четность (нет никаких исключений для этого требования). Использование SERR# для сообщения об ошибках нечетности необязательно. Однако должно быть принято, что передача сигналов на SERR# генерирует NMI и, следовательно, неисправима. Следовательно, нужно осторожно использовать SERR#.

Следующие разделы описывают ответственность каждого агента шины относительно передачи сигналов на PERR# и SERR#.

3.7.2.1. Ответ ошибки четности и сообщений на PERR#

Этот раздел описывает соответствующий запрос, сообщение, ошибки четности данных во всех операциях шины за исключением команд Специальных Циклов. Все ошибки четности адреса, также как ошибки четности данных команд Специального Цикла сообщаются на SERR#, и описаны в следующем разделе. Все ссылки к ошибкам четности в этом разделе ограничены строго четностью данных (за исключением команд Специального Цикла).

PCI использует сигнал PERR# , чтобы сообщить об ошибке четности данных между связанными устройствами на PCI (за исключением команд Специального Цикла). Только мастеру разрушенной передачи данных позволяют сообщить ошибки четности программному обеспечению, используя не PERR#, а другие механизмы. Адресаты всегда сообщают об ошибке четности данных обратно мастеру на PERR#. Это дает создателю доступа на каждом аппаратном или программном уровне возможность восстановления.

Исключая установку бита обнаружения Ошибки Четности, вся передача сигналов ошибки четности и ответ управляется битом Ответа Ошибки Четности. Этот бит требуется в предварительно перечисленных устройствах. Если бит очищен, агент игнорирует все ошибки четности и завершает транзакцию, как если бы четность была правильна. Если бит установлен, агент должен выставить PERR#. Во всех случаях, Обнаруженный бит Ошибки Четности должен быть установлен.

Агент должен всегда выставлять PERR# на два PCI такта после передачи данных, в котором ошибка произошла, как показано на Рисунке 3-22. Агент, получающий данные свободен выставить PERR#, когда ошибка четности обнаружена (которая может происходить прежде, чем данные перемещены)[13] . Когда PERR# выключен, он должен остаться таким два такта после фактической передачи. Мастер знает, что ошибка четности данных произошла в любое время, когда PERR# выставлен, но также знает, что передача была свободна от ошибок 2 такта после передачи.

В случае многократной передачи данных без вмешательства, PERR# будет квалифицирован на многократных последовательных тактах соответственно, и может быть выставлен на любом или на всех.

[13] На транзакции записи это может происходить, когда IRDY# выставлен, и адресат вставляет состояние ожидания. На транзакции чтения это происходит, когда TRDY# выставлен, и мастер вставляет состояние ожидания.

С этих пор PERR# является поддержанным сигналом tri-состояния, нужно довести его до правильного значения на каждом срезе такта. Для возвращения его в номинальное состояние в конце каждая операции шины, он должен активно управляться за один такт, спустя два такта после цикла AD шины (например, такт 7 на Рисунке 3-22). PERR# цикл происходит на один такт позже (такт 8 на Рисунке 3-22). PERR# может никогда не быть выставленным в текущем цикле до истечения трех тактов после фазы адреса.

Когда мастер обнаруживает ошибку четности данных и выставляет PERR# (на транзакции чтения) или выставляет PERR# (на транзакции записи), он должен установить бит сообщения четности данных, бит 6 регистра состояния и может или продолжать транзакцию или завершать ее. Адресат транзакции, который обнаруживает ошибку четности, может или продолжать операцию или завершать ее. Адресаты никогда не устанавливают бит сообщения четности данных. Когда PERR# выставлен, рекомендуется, чтобы и мастер, и адресат завершили транзакции. PERR# является только сигналом вывода для адресатов, в то время как мастер - устройства используют PERR# как для ввода, так и для вывода.

Когда мастер доступа узнает, что ошибка четности произошла на транзакции, он сообщает это системе. Рекомендуется, чтобы мастер сообщил это драйверу устройства ошибки, генерируя прерывание (или изменяя состояние регистров, или флажок, чтобы упомянуть несколько параметров). Ни один из этих параметров не является доступным устройству. Обратите внимание, что проектировщик системы может выбирать сообщать все ошибки четности операционной системе, преобразуя весь сигнал ошибки PERR# в сигнал ошибки центрального ресурса SERR#.

3.7.2.2. Реакция на ошибку и сообщений о ней по SERR#

SERR# используется, чтобы сообщить обо всех ошибках четности адреса, ошибки четности данных на командах специального цикла и все другие ошибки. Любой агент может проверять и сообщать об ошибках четности адреса на SERR#. SERR# может только быть установлен, когда бит разрешения SERR# в регистре команд установлен в единицу независимо от типа ошибки. Когда агент выставляет SERR#, требуется установить бит сообщения Ошибки Системы в пространстве конфигурации регистра состояния, независимо от типа ошибки. Кроме того, если ошибка типа четность (например, четность адреса), бит обнаружения ошибки четности должен быть установлен во всех случаях, поместить сообщение на SERR# обусловлено битом ответа ошибки четности в регистре команд.
Выбранный агент, который обнаруживает ошибку контроля четности адреса, должен выполнить одно из следующих действий: завершать цикл, как если бы адрес был правилен, завершить цикл с прекращением работы целевого устройства, либо не прекращать цикл и позволить аварийное завершение работы мастер - устройства. Целевому устройству не разрешается прекращать свою работу с повтором или разрывом связи, так как была обнаружена ошибка контроля по четности адреса.
Сигнал SERR# не имеет никакой связи синхронизации с любой PCI - транзакцией[14] . Однако, об ошибки необходимо сообщать так скоро, насколько это возможно; предпочтительно -внутри двух тактов детектирования ошибки. Единственный агент заинтересовал SERR# (поскольку вход) - центральный ресурс, который преобразовывает низкий импульс в сигнал на процессор. Так как центральный ресурс системно - зависим от сигналов процессора, то можно было бы генерировать немаскируемые прерывания NMI, высокоприоритетное прерывание, установить состояние ожидания или флажок. Однако, агент, который устанавливает в активное состояние сигнал SERR#, должен будет для центрального ресурса сгенерировать NMI; иначе об ошибке надо сообщит с помощью другого механизма (например, прерыванием, через статусный регистр, или флаг).

[14] Отсутствует сигнал CLK

Когда бит ответа ошибки четности установлен и бит разрешения SERR# установлен тоже, агент может выставлять SERR# при следующих условиях:

- Обнаружена ошибка четности адреса или ошибка четности данных на Специальных Циклах
- Обнаружена ошибка четности, которая не сообщена некоторым другим механизмом (только текущим мастером шины).

Когда бит разрешения SERR# установлен, агент может выставлять SERR# при следующих условиях:

- Мастер (который не имеет драйвера) включался в транзакции, которые неправильно завершены.
- Катастрофическая ошибка, которая сделала агента невозможным функционировать правильно.

Обратите внимание, что аварийное прекращение работы мастером - не аварийное условие для мостов для команд конфигурации и
Специального цикла. SERR# не должен использоваться для этих условий или для обычно восстанавливаемых случаев. Выставление SERR# должно быть выполнено обдуманно и осторожно, начиная, когда результата может быть NMI. Аварийное прекращение работы адресатом - может быть сообщено (только мастером) как ошибка передачей сигналов SERR#, когда мастер не может сообщать ошибку через драйвер устройства.

Ведущий интерфейс должен выполнять такой счетчик, что когда счет истекает, ведущий мост не повторяет доступ. Счетчик увеличивается (уменьшается), когда доступ завершен с повторением. Счетчик сброшен всякий раз, когда мастер передает данные. Это не требование, но рекомендуется гарантировать, что доступ не будет продолжен завершением с повторением, таким образом предотвращая ситуацию, когда процессор обрабатывает прерывание, которое могло бы указывать на условия ошибки.

3.8. Поддержка кэша

В уровне входа или подвижных системах, часть или вся память системы может быть на PCI. Это может включать доступные только для чтения модули программы, такие как DRAM, которые должны быть кэшированы. Опция поддержки кэша PCI обеспечивает стандартный интерфейс между PCI агентом (-ами) памяти и мостом, который позволяет использование механизм согласованного отслеживания кэша. Эта опция кэширования поддерживает простое адресное пространство (то есть, одиночный адрес имеет уникального адресата независимо от происхождения доступа) и одиночную топологию уровня моста. Обратите внимание, что эта поддержка оптимизирована для простого уровня входа системы, скорее, чем максимальное соотношение процессор/кэш/память.

Кэширование поддержки для общедоступной памяти выполнено двумя необязательными контактами, названными SDONE и SBO#. Они передают информацию состояния кэша между мостом /кэшем и адресатом запроса памяти. Интерфейс / кэш отслеживает доступ памяти на PCI, и определяет то, что отклик требуется целевым устройством, чтобы гарантировать согласованность с памятью системы.

Для избежания «отслеживающей перегрузки» интерфейс можно запрограммировать так, чтобы сигнализировать о «чистом отслеживании» (Clean Snoop) немедленно, по адресным интервалам, по которым наиболее часто происходят обращения, и которые конфигурированы как кэшируемые (например, буфер изображения). Любое целевое устройство PCI, поддерживающее кэшируемую память, должно контролировать поддержку выводов кэша PCI, и соответственно реагировать. Целевые устройства, конфигурированные так, чтобы не кэшироваться, могут игнорировать сигналы SDONE и SBO#, так как это может сохранять небольшой время задержки запроса, в зависимости от конфигурации. Так как PCI позволяет блоки безграничной длины, целевое устройство кэшируемого запроса должно прервать запрос, если встретятся диапазоны памяти, которые выходят за границу строки кэша.

Если пакету позволяют пересечь границу кэша, связь с кэшем может прерваться. (Интерфейс/ кэш может контролировать транзакцию и генерировать следующий адрес кэша.)

Для более эффективного использования PCI шины, требуется контроллер кэшированной памяти и кэш / мост, чтобы проследить операцию шины. (Когда одиночный адрес заперт, условие может выполняться, когда кэшированная транзакция будет отсрочена. Это происходит, когда не кэшированная транзакция инициализирована, когда кэш может иметь доступ к адресу. Так как транзакция не кэшированная, она завершится независимо от состояния SDONE. Когда следующая транзакция инициализирована, пока первый доступ все еще задержан, требуется, чтобы кэшированная транзакция была повторена, иначе к адресу не будет доступа. Если не кэшированная и кэшированная транзакции чередуются, кэшированная транзакция может никогда не выполниться.) Чтобы минимизировать повторения кэшированных транзакций, требуется, чтобы агенты, включаемые в кэшированные транзакции, принимали два адреса. Это означает, что в то время как первый адрес доступен, следующий адрес, находящийся на шине будет запираться. Когда первый доступ завершается, доступ ко второму адресу начнется немедленно. Требуемое максимальное число адресов, которые могут запираться - два. Третий адрес никогда не может появляться на шине, либо без завершения доступа к списку адресов, либо завершения второй транзакции.

Когда доступ или вторая транзакция завершается, контроллер кэша и памяти готов принять новый адрес. Следовательно, требуется только запираемых два адреса.

Если вторая транзакция - кэшированная, требуется контроллер памяти, чтобы вставить ожидание состояния, пока первый доступ завершается. Когда доступ первой транзакции завершается, контроллер памяти продолжает работу с транзакцией. Если вторая транзакция к не кэшированному адресу, адресат может завершать транзакцию пока SDONE и SBO# не проверены. Если адресат второй транзакции выставляет TRDY# (или STOP#) до или с выставлением SDONE, это подразумевает не кэшированную транзакцию и кэш не будет иметь доступ к адресу, когда TRDY# выставлен. Следовательно, максимальное число адресов, которые могут быть используемы в любое время - два.

3.8.1. Определение состояний кэша

PCI спецификация определяет SDONE и SBO#, чтобы обеспечить информацию между агентами, которые участвуют в протоколе кэша. Когда SDONE выставлен, это указывает что доступ завершился. Когда SBO# выставлен, это указывает на изменяемую линию.
Когда SBO# является выключенным и SDONE выставлен, это указывает "CLEAN" результат доступа.

Имеется три состояния кэша на PCI. Значение каждого состояния, когда управляется кэшем/ мостом (которые будут описаны после кэша) и как контроллер кэшированной памяти должен интерпретировать их, будет обсуждено далее. Сигналы кэширования PCI SDONE и SBO# имеют одно из трех следующих состояний:

STANDBY	0 x
CLEAN	1 0
HITM	1 1

3.8.1.1. Контроллер кэша / кэшированной памяти

Когда кэш управляет тремя состояниями на шине, подразумевается следующее:

STANDBY - указывает, что кэш находится в одном из трех условий. Первое условие, когда кэш в настоящее время не имеет доступа к адресу, но готов делать его. Второе условие - адрес заперт, и кэш - в настоящее время имеет доступ к адресу и готов принять второй адрес, если он есть на шине. Последнее условие - когда кэш - в настоящее время имеет доступ и запер второй адрес. Кэш будет запускать доступ второго адреса, когда доступ завершается. (Примечание: Это состояние сообщено, когда SDONE - выключен.) Контроллер памяти должен проследить за управляющими сигналами, чтобы знать, какие условия благоприятны для кэша. Контроллер памяти отвечает на запрос, поскольку он выбирает, когда адрес недоступен. Если доступ запущен и контроллер памяти - адресат второй транзакции, он должен вставить состояния ожидания, пока первый доступ завершится. Контроллер памяти продолжает вторую транзакцию, когда доступ к первому адресу завершается. Если контроллер памяти - не адресат, он должен контролировать шину для определения, второй адрес - достигнут или отброшен.

CLEAN - указывает, что нет конфликта кэшей, и доступ к памяти может завершаться обычно. Он подразумевает пропадание к кэшу, или на неизменяемую линию в течение транзакции записи или попадание на изменяемую линию в течение команды Memory Write и Invalidate. Обратная запись, вызванная командой Memory Write и Invalidate или Memory Write к неизменяемой линии кэша не требуется. (Это допустимо, пока мастер транзакции гарантирует, что каждый байт будет изменяться, и адресат не будет завершать транзакцию до всего перемещения.) Кэш выставит CLEAN в течение фазы адреса, когда он текущий мастер шины и записывает обратно изменяемую линию.

Кэш может выставить CLEAN на двух последовательных тактах, когда два адреса заперты. Первый такт, когда SDONE выставлен, то это показывает, что первый доступ завершен. Если первый доступ CLEAN, и вторая транзакция была инициализирована кэшем, он может продолжать утверждать SDONE, что доступ к этому (второму) адресу выполнялся CLEAN. (Второй последовательный CLEAN имеет то же самое значение, как в течение фазы адреса - операция writeback или CLEAN доступ.) Иначе, сигналы кэша STANDBY после CLEAN указывают, что происходит доступ. В этом случае STANDBY (SDONE выключен) появился бы на шине на следующем такте после выставления SDONE.

HITM - указывает, что доступ к изменяемой линии, и требуется writeback адрес доступа на следующей операции. Кэш останется в этом состоянии до обратной записи. Все другие кэшированные транзакции будут завершены с повторением контроллером памяти, в то время как HITM выставлен на шине. (Если любая другая кэшированная транзакция требует завершения перед обратной записью изменяемой линии, произойдет livelock.) В течение обратной записи изменяемой линии, кэш будет делать транзакцию от HITM до CLEAN в течение фазы адреса.

Контроллер памяти "обычно" завершает транзакцию с повторением, позволяя происходить обратной записи, и затем заново запрашивает агента, который был завершен с повторением. Контроллер памяти завершит все последующие кэшированные транзакции с повторением, пока выставлен HITM.

3.8.2. Поддерживаемые состояния и переходы

[1] STANDBY --> CLEAN --> [CLEAN] --> STANDBY
[2] STANDBY --> HITM --> CLEAN --> [CLEAN] --> STANDBY

Последовательность [1] - нормальный случай, где кэш находится в STANBY до завершения доступа и затем выставляет CLEAN, чтобы указать, что транзакция должна завершиться обычно. Кэша переходит к STANDBY, если второй адрес не задержан, когда доступ первой транзакции завершается. Если второй адрес заперт и кэш не мастер, он переходит к STANDBY, указывающему доступ. Если кэш - мастер второй транзакции, он может продолжать удерживать CLEAN, когда транзакция - обратная запись кэша, или знает, что доступ CLEAN; иначе он будет переходить к STANDBY.

Последовательность [2] - когда изменяемая линия обнаружена в течение доступа. Однажды выставленный кэшем HITM, будет находиться в этом состоянии, пока изменяемая линия не написана обратно. Кэш будет переходить к CLEAN, выполняя обратную запись. После CLEAN кэш выставит STANDBY, указывающий, что кэш готов к доступу по новому адресу. Если кэш - мастер второй транзакции, он может продолжать удерживать CLEAN, когда транзакция - обратная запись кэша или знает, что доступ CLEAN; иначе, он будет переходить к STANDBY.

3.8.3. Временные диаграммы

В следующих рисунках принимается, что шина стартует при условии выставления IDLE в такте 1.

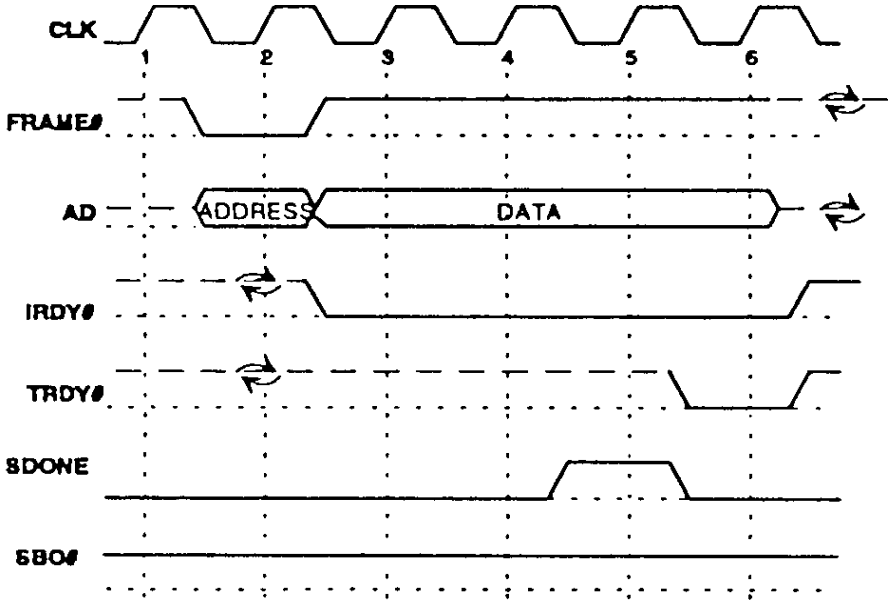


Рисунок 3-23: Состояния ожидания завершения доступа

Транзакция на Рисунке 3-23 начинается, когда адрес заперт на такте 2. Адресат сохраняет TRDY# выключенным (вставляя состояние ожидания), пока доступ не завершается. Доступ завершается на такте 5, когда SDONE выставлен. SBO# не был выставлен, пока результат доступа показывает CLEAN.

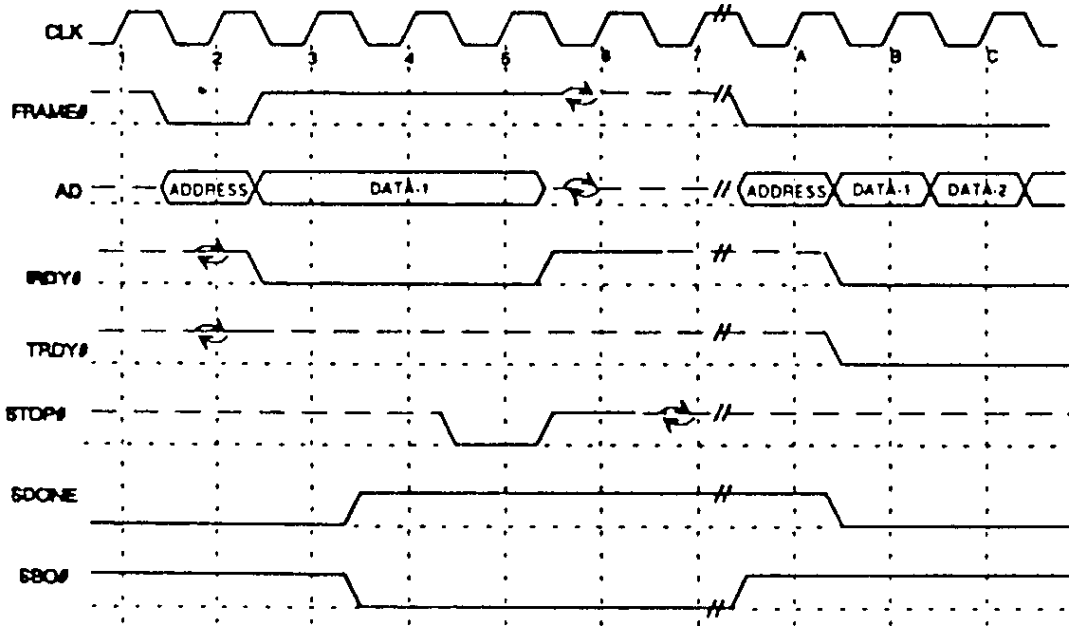


Рисунок 3-24: Доступ на изменяемую линию, следующую за обратной записью.

На рисунке 3-24 инициализация транзакции начинается на такте 2 с адресом, который заперт. Адресат этой транзакции вставляет состояния ожидания, пока выставлен SDONE. В этом примере кэш показывает, доступ к изменяемой линии на такте 4, выставляя SBO# и SDONE. (Однажды выставленный SBO# должен остаться таким, пока выставлен SDONE). Так как адресат транзакции кэшированный, он выставляет STOP#, чтобы завершить транзакцию на такте 5. Это позволяет кэшу, который выставил HITM, писать изменяемую линию обратно в память. Для транзакций чтения, контроллер памяти должен устанавливать в третье состояние AD линии, когда выставлен HTIM. Все транзакции кэшированным адресатам завершены с повторением, в то время как HITM выставлен на шине.

Рассмотрение линии указывает, что прошло некоторое количество времени, с тех пор, как доступ был выставлен на первой транзакции. В это время не кэшированные транзакции могут завершаться, и кэшированные транзакции могут начинаться, но требуется, чтобы они были завершены с повторением, пока выставлен HITM. Транзакция обратной записи начинается на такте A. Обратите внимание на переходы кэша из HITM к CLEAN в течение фазы адреса. Это указывает контроллеру памяти, что доступ обратной записи начинается, и требуется принять всю линию. (Если контроллер памяти не способен завершить транзакцию, он должен вставить состояния ожидания, пока он не способен завершить ее. Это условие должно произойти, когда кэшированный адресат имеет внутренний конфликт, например, операция регенерации массива.) (Если адресат блокирован, он принимает обратные записи, вызванные доступами на изменяемые линии; иначе происходит зависание.) И кэш, и контроллер памяти могут вставлять состояния ожидания в течение обратной записи. Контроллер памяти требуется, чтобы принять всю линию в одиночной транзакции, и кэш будет обеспечивать всю линию. Обратите внимание, что кэш переходит от CLEAN к STANDBY на такте B. Кэш теперь готов принять другой адрес доступа. Как только обратная запись завершается, шина возвращается к нормальной операции, где кэшированные транзакции будут выполняться. Порядок обратной записи не зависит от транзакции, которая вызвала обратную запись. На рисунке, DATA - 1 только указывает первую передачу данных, а не DWORD номер.

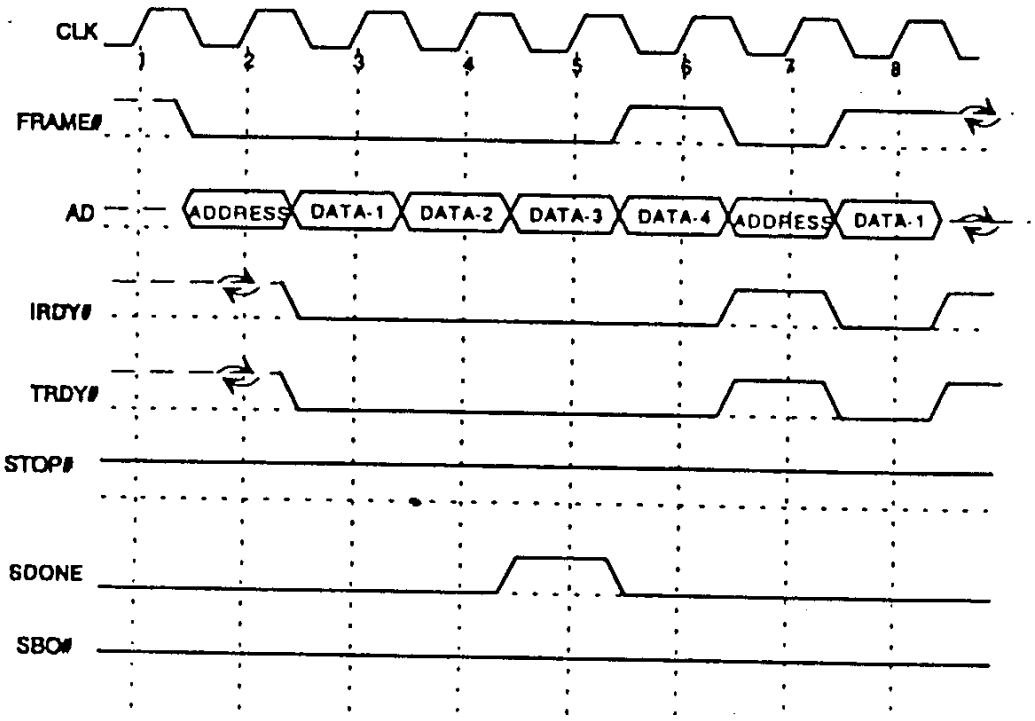


Рисунок 3-25: Запись памяти и лишение силы команду

Рисунок 3-25 - пример команды Memory Write и Invalidate. Кэш имеет несколько параметров того, как обработать эту команду. Так как мастер гарантирует, что каждый байт на линии кэша будет изменяться, кэш мог бы просто выставить CLEAN, даже если линия попадает на изменяемую линию. В этом примере, кэш выставляет CLEAN на такте 5. Как только кэш выставляет CLEAN, он готово к доступу к следующему адресу, обеспеченному на шине. Следовательно, кэш должен ждать, пока он будет готов, перед выставлением SDONE.

Если SBO# был выставлен на такте 5, доступ закончился попаданием на изменяемую линию и будет записан обратно. Кэш может обрабатывать команды Memory Write и Invalidate подобно любой другой команде и, следуя условию HITM, появиться на шине. (Обратная запись вызывает транзакцию дополнительного пространства на шине, которая не требуется.) Кэш мог бы ждать фиксированное число тактов для выставления TRDY# перед индикацией результата доступа. Если TRDY# выставлен перед результатом, рекомендуется, чтобы кэш отбросил линию и выставил CLEAN. Если TRDY# не выставлен, кэш продолжает обеспечивать результат доступа. Однако, время ожидания TRDY# должно быть фиксировано, потому что контроллер памяти может всегда ждать выставления SDONE перед продолжением транзакции.

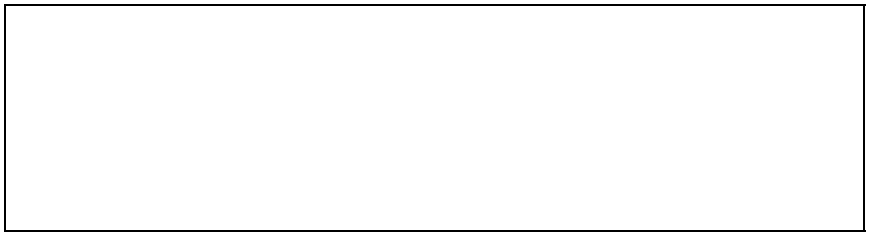


Рисунок 3-26: Передача данных в измененную строку после обратной записи

В рисунке 3-26 начальная транзакция начинается в такте 2 и завершается в такте 3. В то время как отслеживание первой транзакции находится в прогрессе, начинается следующая транзакция в такте 5. Вторая транзакция также короткая и завершается в такте 6. Вторая транзакция кэшируема, и она завершается, пока еще идет первая транзакция. В такте 7 завершается первая транзакция. Если был активен сигнал FRAME#, и идет процесс, то состояние SDONE и SBO# имеет значение только для первого адреса, пока SDONE не станет активным. Если SDONE был активным, то в следующий раз, когда он вновь станет активным, это будет означать вторую транзакцию. Если в вышеупомянутой диаграмме SDONE активен в такте 5 вместо такта 7, то результат такого отслеживания не имеет никакого эффекта на вторую транзакцию, даже если это происходит во время второй транзакции.

3.8.4. Поддержка кэша сквозной записи

Поддержка кэша сквозной записи - такая же, как и у кэша обратной записи, за исключением сигнала SBO#, который в этом случае не используется. Контроллер памяти контролирует шину, а также «смотрит», сколько на ней имеется ожидающих обработки адресов. Максимум может ожидаться два адреса. Каждый раз когда активен SDONE, контроллер памяти может разрешить завершение другой кэшируемой транзакции.

В режиме сквозной записи поддерживаются только переходы с промежуточными состояниями:

STANDBY->CLEAN->[CLEAN]->STANDBY

Если SBO# не используется в режиме сквозной записи, он может быть установлен в высокий уровень проектировщиком системы. Следовательно, между состояниями STANDBY и CLEAN могут быть только переходы, имеющие промежуточные состояниями. Если кэш является мастером второй транзакции, то он это может продолжать сигнализировать CLEAN (это показано как необязательное состояние), когда идет транзакция строки кэша методом обратной записи или известно состояние CLEAN; в противном случае, будет переход к состоянию STANDBY. Рекомендуется, чтобы кэшируемые целевые устройства предусматривали использование как SDONE, так и SBO#.

Для каждого сигнала FRAME#, который установлен на шине, кэш установит SDONE при появлении адреса. Если два установления сигнала FRAME# происходят без установки SDONE, то вторая кэшируемая транзакция не может завершиться. Если второй запрос кэшируется, то контроллер памяти должен вставить состояния ожидания, пока предыдущий опрос не завершается (активный сигнал SDONE). Если второй запрос некэшируемый, запрос завершается, и кэш не будет обращаться по адресу. В этом смысле, обработки ожидает только один адрес.

3.8.5. Замечания по арбитражу

Арбитр требуется, чтобы осуществить некоторый вид алгоритма «равнодоступности», когда сигнал HITM проходит на шине, в противном случае может происходить длительная блокировка. Длительная блокировка происходит, когда кэш, который имеет изменяемую строку, неспособен выполнить обратную запись, потому что два высокоприоритетных агента обращаются к памяти кэшируемого. Когда HTTM появляется на шине, все транзакции кэшируемого завершаются с повтором.

Рекомендуется, чтобы арбитр, когда кэш присутствует в системе, мог выбирать, чтобы соединить REQ# с фиксированным входом, так что приоритетный уровень может быть поднят, когда HITM появляется на шине. Это обеспечивает, что в то время, как была отложена обратная запись, число транзакций кэшируемого, которые завершены с повтором, сохранилось минимальным, а время ожидания также было небольшим.

Когда в системе используется кэш (в частности, кэш обратной записи), время ожидания целевого устройства должен увеличиться, чтобы подсчитать время, которое требуется для обратной записи измененной строки. Эта величина зависит от алгоритма арбитража и того, когда на шине может появиться запрос на обратную запись.

3.9. Расширение шины до 64 разрядов

PCI поддерживает 64-разрядные линии данных, чтобы обеспечить дополнительную ширину пропускания для агентов, которым это необходимо. 64-разрядные устройства нуждаются в дополнительных 39 выводах: REQ64#, ACK64#[63::32], C/BE[7::4], и PAR64. Эти сигналы определены в разделе 2.2.8. В конце сброса, REQ64# сообщает 64 -разрядному устройству, что оно либо соединено с 64 - разрядными линиями данных, либо нет. Когда REQ64# - неактивный, в конце сброса, то устройство соединено с 64-разрядными линиями данных. Когда REQ64# активен, в конце сброса, то устройство соединено с 64-разрядными линиями данных. Обращайтесь к разделу 4.3.2 за информацией относительно того, как устройство ведет себя после сброса. 64 - битные транзакции динамически «заключаются» (один раз в фазе адреса) между мастером и целевым устройством. Это происходит, когда мастер устанавливает REQ64#, а целевое устройство отвечает на это установкой сигнала ACK64#. REQ64# и ACK64# устанавливаются извне, гарантируя соответствующее поведение, когда имеются смешанные 32- и 64-битные агенты. Как только встречается 64-разрядная транзакция, то она держится до завершения транзакции. Работа 32-разрядных агентов не нарушается 64-разрядными агентами. В 32-разрядном режиме 64-разрядные агенты должны переключиться в состояние по умолчанию, пока они не будут затребованы. Следовательно, 64-разрядные транзакции полностью прозрачны для 32-разрядных устройств.

В течение 64-разрядной транзакции, весь протокол PCI и синхронизация остаются прежними. К 64-разрядным пересылкам данных чувствительными являются только команды по работе с памятью. Команды Interrupt Acknowledge и Special Cycle[16] - это обычные 32-разрядные команды и не должны использоваться с REQ64#. Требования команд конфигурации к ширине диапазона ввода - вывода не могут сгладить излишнюю сложность, а, следовательно, 64-разрядные пересылки данных поддерживают только команды работы с памятью.

[16] Так как никакой агент не запрашивает доступ путем установки DEVSEL#, то, следовательно, он не может отвечать установкой ACK64#.

Все команды памяти и передачи шины те же самые, перемещены ли данные 32 или 64 бита одновременно. 64-разрядные агенты могут передать от одного до восьми байтов на фазе данных, и все комбинации разрешающего байта допустимы. Как в 32-разрядном режиме, разрешающий байт может изменяться на каждой фазе данных. Мастер, инициализирующий 64-разрядная транзакцию данных должен использовать двойной (DWORD - 4 слова или 8 байтов) адрес (AD2 должен быть "0" в течение фазы адреса).

64-разрядная четность работает, также как 32-разрядная четность, за исключением одного дополнительного сигнала четности. PAR64 покрывает AD[63::32] и C/BE[7::4] и имеет ту же самую синхронизацию и функцию как PAR. PAR64 должен иметь силу один такт после фазы адреса на любой транзакции, в которой выставлен REQ64#. Число "1" на AD[63::32]. C/BE [7:: 4]#, и PAR64 равняется четному числу.

Однако PAR64 должен быть дополнительно квалифицирован с REQ64# и ACK64# для фаз данных. PAR64 требуется для 64-разрядных фаз данных; это опция не для 64-разрядного агента.

В следующих двух рисунках 64-разрядный мастер запрашивает 64-разрядную транзакцию. Первая - чтение, где адресат отвечает с ACK64#. Вторая - запись, где адресат не отвечает и по умолчанию транзакция 32-разрядная. Эти два рисунка идентичны рисункам 3-1 и 3-2 за исключением того, что были добавлены 64-разрядные сигналы, и чтение передает вдвое больше данных. Те же самые транзакции используются для иллюстрации, тот же самый протокол работает для 32 и 64-разрядных транзакций.

AD[63::32] зарезервированы в течение фазы адреса, но содержат данные в течение 64-разрядных фаз данных. C/BE [7::4]# зарезервированы в течение фазы адреса, но содержат разрешающий байт для старших четырех байтов в течение фаз данных.

Рисунок 3-27 иллюстрирует запрос 64-разрядной транзакции чтения мастером, выставляя REQ64# (который точно отражает FRAME#). Адресат подтверждает запрос выставлением ACK64# (который отражает DEVSEL#). 64-разрядные сигналы требуют тех же самых оборотных циклов, требуемых их 32-разрядными дубликатами.

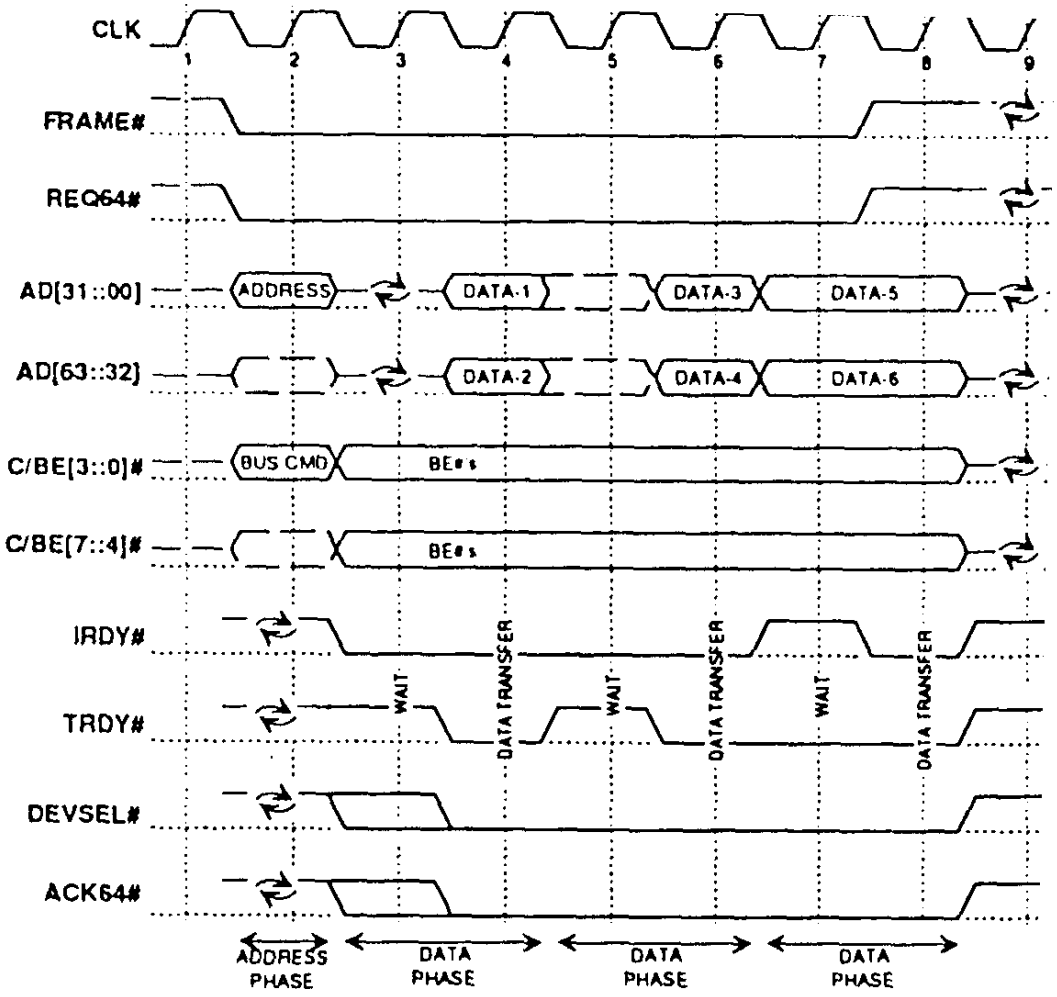


Рисунок 3-27: 64-разрядный запрос - 64-разрядная передача

Рисунок 3-28 иллюстрирует запрос 64-разрядной передачи мастером. Адресат не трогает REQ64# и ACK64#, сохраняемые в выключенном состоянии. Можно понять, что это - 32-разрядная передача. Мастер преобразовывает транзакцию от 64 до 32 битов. Так как мастер преобразовывает 64-разрядные передачи данных в 32-разрядные передачи данных, там может быть или не быть выставленным любой разрешающий байт в течение любой фазы транзакции. Следовательно, все 32-разрядные адресаты должны быть способны обработать фазы данных без разрешающего байта. Адресат не должен использовать разъединение или повторение, потому что сталкивается с фазой данных, где не имеется никакого разрешающего байта, но надо выставить TRDY# и завершить фазу данных. Мастер снова посылает данные, которые появились на AD[63:: 32] в течение первой фазы данных и на AD[31:00] в течение второй. Последующие фазы данных появляются подобно 32-разрядной передаче. (Если удалить 64-разрядные сигналы, то рисунки 3-28 и 3-2 будут похожи).

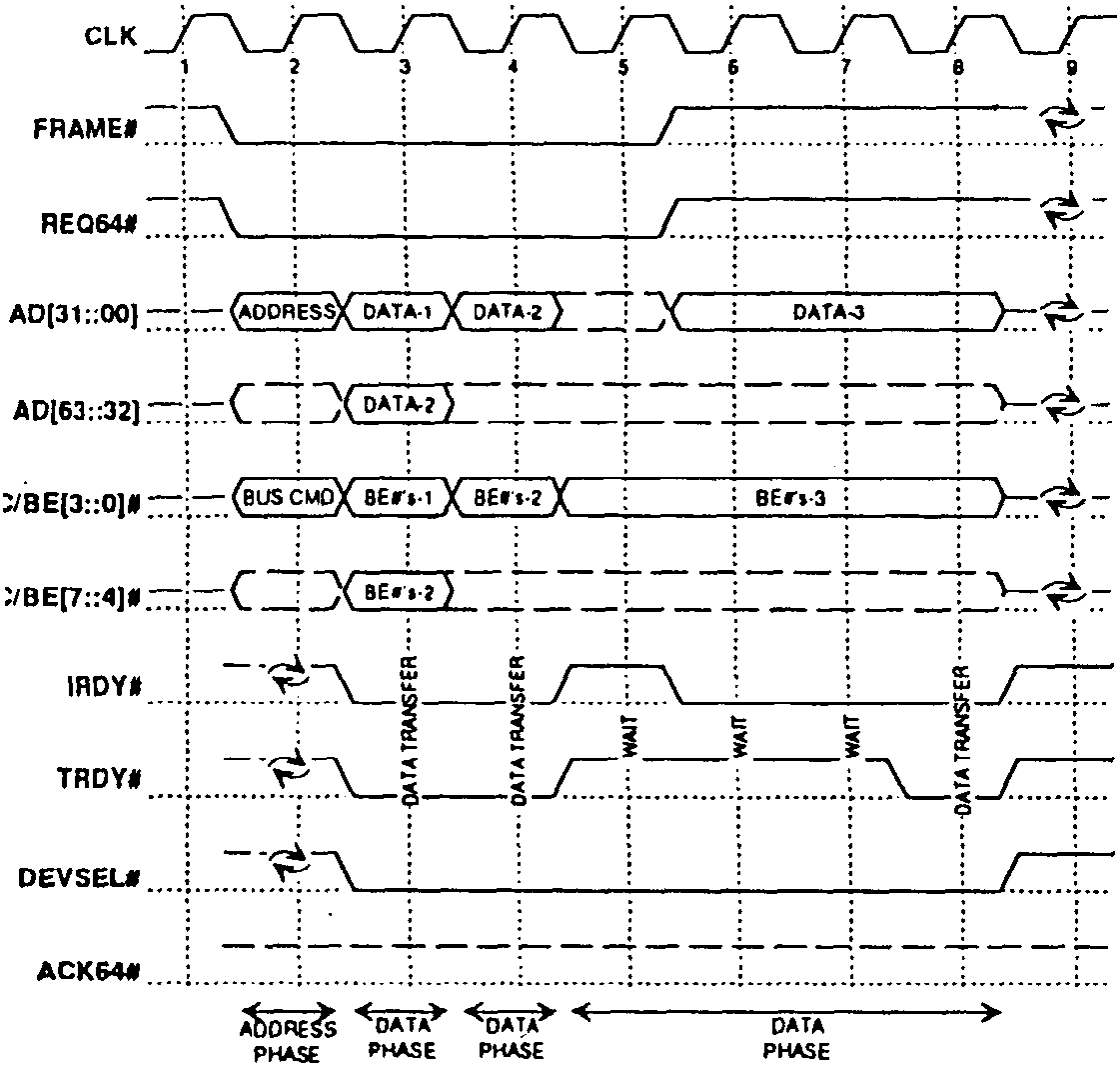


Рисунок 3-28: 64-разрядный запрос - 32-разрядная передача

Использование одиночной фазы данных с 64-разрядными передачами не может быть очень эффективно. Так как мастер не знает, как цикл будет решен до возвращения DEVSEL#, он не знает цикл, на котором надо выключать FRAME#. IRDY# должен остаться выключенным до решения сигнала FRAME#, что означает две фазы 32-разрядной передачи - по крайней мере, с такой скоростью как одна фаза 64-разрядной передачи. Если 64-разрядный мастер имеет явное знание относительно адресного интервала, который ответит как 64-разрядный адресат (например, через регистр конфигурации), требуется ждать DEVSEL#, чтобы решить одиночную передачу данных (FRAME# является выключенным).

3.9.1. 64-разрядная адресация на PCI

PCI поддерживает адресацию более 4 GB, определяя механизм для передачи 64-разрядного адреса от мастера транзакции адресату. Мастер может генерировать 64-разрядный адрес независимо от того, поддерживает он или нет 32 или 64-разрядных линии данных. Не требуется никаких дополнительных сигналов для поддержки 64-разрядной адресации. Если и мастер и адресат поддерживают 64-разрядные линии данных, то весь 64-разрядный адрес можно обеспечивать в одном такте. Адресаты, которые поддерживают только 32-разрядные адреса, будут работать с мастерами, которые могут генерировать 64-разрядные адреса, а те отображаются в младшие четыре гигабайта адресного пространства.

Стандартная транзакция PCI поддерживает Цикл Одиночного Адреса (SAC), где адрес имеет силу для одиночных тактов, когда FRAME# является первым выставленным сигналом. Чтобы поддерживать передачу 64-разрядного адреса без того, чтобы требовать 64-разрядный путь данных, используется Цикл Двойного Адреса (DAC). DAC использует два такта, чтобы передать весь 64-разрядный адрес. Мастера, которые используют продвижение адреса, не могут выполнять 64-разрядную адресацию, пока нет механизма для задержки или продления второй фазы адреса.

Рисунок 3-29 иллюстрирует DAC. В базисной транзакции чтения, оборотный цикл следует за фазой адреса. В DAC дополнительная фаза адреса вставлена между стандартной фазой адреса и оборотным циклом. В диаграмме первые и вторые фазы адреса происходят на тактах 2 и 3 соответственно. Оборотный цикл между адресом и фазами данных отсрочен до такта 4. Обратите внимание, что FRAME# должен быть выставлен в течение обеих фаз адреса. Чтобы твердо придерживаться связи FRAME# - IRDY#, FRAME# не может быть выключен, пока не выставлен IRDY#. IRDY# не может быть выставлен, пока мастер не обеспечивает данные для транзакции записи или готов принять данные транзакции чтения. DAC декодируется потенциальным адресатом, когда "1101" присутствует на C/BE [3:: 0]# в течение первой фазы адреса. Если адресат поддерживает 64-разрядную адресацию, он сохраняет адрес, который был перемещен на AD[31:: 00] и защелкивает остальную часть адреса на следующих тактах. Фактическая команда, используемая для транзакции, перемещается в течение второй фазы адреса на C/BE [3:: 0]#. Как только весь адрес перемещен, и команду заперта, адресат определяет, должен ли DEVSEL# быть выставлен. Адресат может делать быстро, средне или медленно декодировку, которая отстает на один такт от декодирования SAC. Не имеется никакой проблемы с этим, так как мост, выполняющий декодировку, игнорирует всю транзакцию, если он не поддерживает 64-разрядную адресацию. Если мост поддерживает 64-разрядную адресацию, он задержит выставленным DEVSEL# на один такт. Мастер (DAC) также задержит завершение транзакции аварийным прекращением работы на один дополнительный такт.

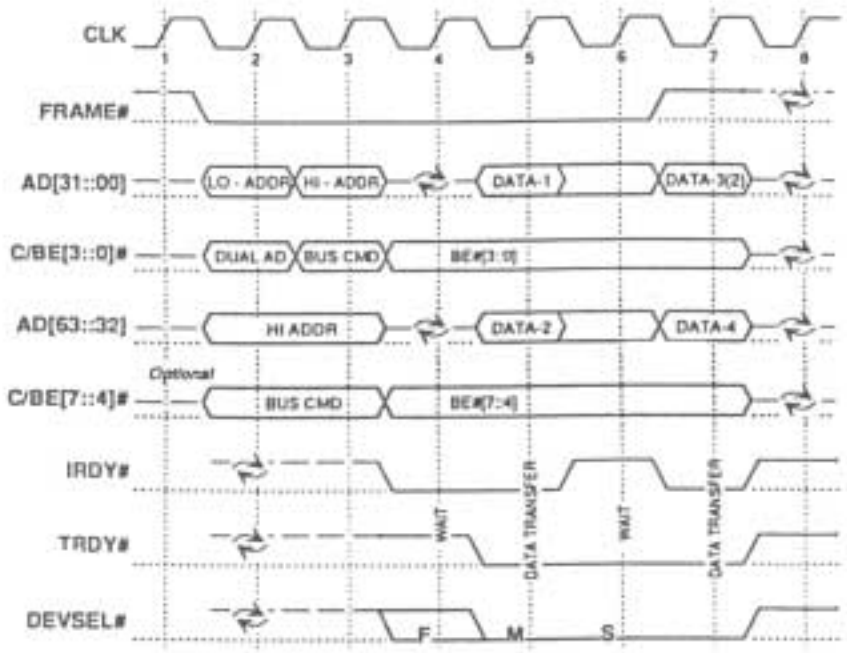


Рисунок 3-29: 64-разрядный цикл двойного адреса

Теневая область в рисунке используется только, когда мастер доступа поддерживает 64-разрядный путь данных. Мастер управляет всем адресом (младшая часть адреса на AD[3l::00] и старшая часть адреса на AD[63::32]) и обеими командами (DAC "110l" на C/BE[3::0]# и действующей командой на C/BE[7::4]#) в течение начальной фазы адреса. На второй фазе адреса мастер управляет старшим адресом на AD[3l::00] (и AD[63::32]) в то время как команда управляется на C/BE [3::0]# (и C/BE[7::4]#). Мастер не может определять, поддерживает ли адресат 64-разрядный путь данных, пока весь адрес не был перемещен и, следовательно, должен принять 32-разрядного адресата при обеспечении адреса.

Если и мастер и адресат имеют 64-разрядные пути данных, то 64-разрядная адресация не использует никакое время ожидания при определении DEVSEL#. Если адресат вставляет состояние ожидания из-за того, что доступ задержан, то дополнительная фаза адреса не имеет никакого значения. Если или мастер, или адресат не поддерживают 64-разрядный путь данных, придется сталкиваться с одним дополнительным тактом задержки.

Мастер, который поддерживает 64-разрядную адресацию, должен генерировать SAC, вместо DAC, когда старшие 32 бита адреса нулевые. Это позволяет мастерам, которые могут генерировать 64-разрядные адреса для связи с 32-разрядными адресатами через SAC.

Тип адресации (SAC или DAC) определяется размером адреса (больше, чем 4 GB), а не возможностями адресата.

Мастер, который поддерживает только 32-разрядную адресацию, может связываться с 64-разрядным адресатом двумя способами: 64-разрядный адресат может действовать подобно 32-разрядному адресату, или мастер может поддерживать 64-разрядную адресацию через DAC (другой вариант - мастер поддерживает только 32-разрядную адресацию, и драйвер устройства перемещает данные от 32-разрядного адресного пространства до 64-разрядного адресного пространства).

3.10. Соображения по специальному проектированию

Этот раздел описывает другие интересные разделы, касающиеся PCI шины, но не являющихся частью базисной операции шины.

- **Третий участник DMA**

Третий участник DMA не поддержан на PCI, так как сигналы не поддерживаются на разъеме. PCI должна группировать вместе DMA - функции в устройствах, которые для этого нуждаются в мастере, и, следовательно, третий участник DMA не поддерживается.

- **Доступ к PCI транзакциям**

Любая транзакция, сгенерированная агентом на PCI может быть доступна любому другому агенту. В общем, агент не может управлять любым PCI сигналом, но должен быть способен функционировать независимо от поведения текущего мастера или адресата

.

Глава 4

Электрическая спецификация

4.1. Краткий обзор

Эта глава определяет все электрические характеристики и ограничения PCI - компонентов. Глава также описывает систему и платы расширения, включая назначение контактов на плате расширения. PCI обеспечивает возможность передачи как сигналов амплитудой 5В, так и сигналов амплитудой 3.3В. Не путайте это с 5-вольтовой и 3.3-вольтовой технологиями изготовления компонентов. 5-вольтовый компонент может быть разработан, чтобы работать в среде передачи сигналов 3.3В и наоборот; технологии компонентов могут использовать различные по напряжению величины сигналов для передачи. Среды передачи сигналов не могут быть смешаны; все компоненты на данной PCI - шине должны использовать одно и то же соглашение передачи сигналов 5В или 3.3В.

4.1.1. Схема перехода от питания 5В к питанию 3.3В.

Основная цель электрической спецификации PCI состоит в том, чтобы обеспечить быстрый и простой переход от 5-вольтового компонента до компонента рассчитанного на напряжение 3.3В. Чтобы облегчить этот переход, PCI определяет два контакта платы расширения - один для 5-вольтовой среды передачи сигналов и один для среды передачи сигналов 3.3В и три типа плат, как показано на рисунке 4-l. Разъём имеет ключ, чтобы избежать установление платы в несоответствующий слот.

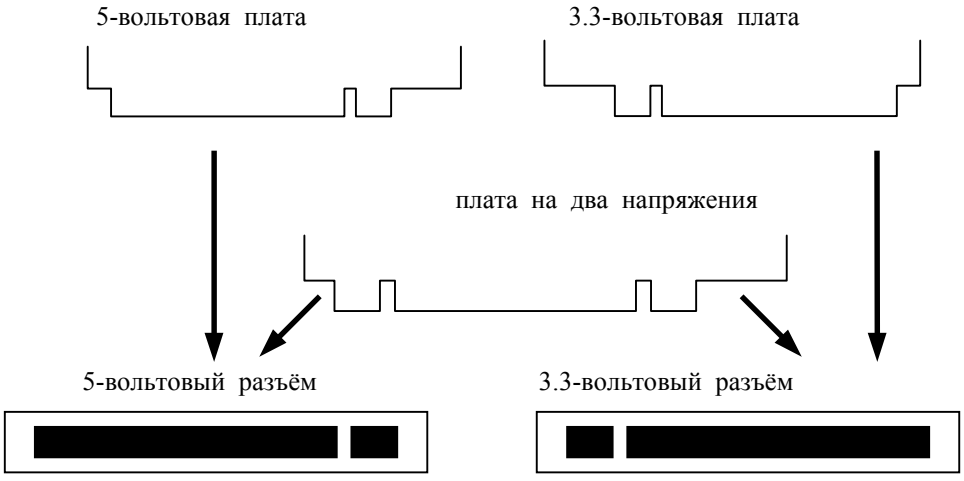


Рисунок 4-1: Разъёмы PCI-платы

Материнская плата (включая контакты) определяет среду передачи сигналов для шины, 5В или 3.3В. 5-вольтовая плата разработана, чтобы работать только в 5-вольтовой среде передачи сигналов и, следовательно, может быть подключена только в 5-вольтовый разъём. Аналогично плата на 3.3В разработана, чтобы работать только в среде передачи сигналов 3.3В. Однако, универсальная плата способна обнаруживать среды передачи сигналов в использовании, и адаптироваться к этой среде. Она может, следовательно, быть подключенной в любой тип разъёма. Все три типа плат определяют соединения с 5-и и 3-хвольтовыми источниками питания и могут содержать любой 5-и или 3-хвольтовый компонент. Различие между типами плат - протокол передачи сигналов который они используют и шины питания, которые они используют для соединения со своими компонентами.

PCI - компоненты на универсальной плате должны использовать буферы ввода-вывода, которые могут применяться в любой среде передачи сигналов. Так как имеются различные реализации буферов, которые могут совмещать среды сигналов, предполагается что они способны работать при любых сбоях питания. Они должны управляться посредством контакта назначения среды напряжения на разъёме PCI, состояние которого зависит от изменения используемой среды передачи сигналов. Это означает, что в 5-вольтовой среде передачи сигналов, эти буферы включены на 5-вольтовой шине. Когда та же плата включена в разъём 3.3В, эти буферы включены на шине 3.3В. Это дает возможность универсальной плате быть совместимой с любой средой передачи сигналов.

Цель этого подхода перехода состоит в том, чтобы совместить 5-вольтовую технологию с 3.3-вольтовой средой передачи сигналов, вместо принуждения 3-вольтовой технологии компонентов для эксплуатации в 5-вольтовой среде передачи сигналов. Хотя последний подход и может быть выполнен, это более трудно и более дорого, особенно в модульной среде шины. Привилегированный вариант - обеспечивающий совместимость 5-вольтового компонента с 3-вольтовой средой передачи сигналов. Он может быть выполнен без увеличения стоимости, и имеет, кроме того, некоторые достоинства эффективности передачи сигналов.

Таким образом первый PCI - компонент будет иметь только 5-вольтовые буферы ввода - вывода. 5-вольтовая плата необходима изначально. Однако, все дополнительные компоненты 5-вольтовой технологии должны использовать двойные буферы напряжений и должны быть встроены в универсальную плату. Это позволяет расширениям, основанным на 5-вольтовой технологии использоваться как в 5-вольтовых так и в 3-вольтовых системах , таким образом допуская совместимость с системами 3.3В. Там где наиболее новые PCI-системы используют среду передачи сигналов 3.3В, компоненты расширения разрабатываются по 3-вольтовой технологии для "зелёных" машин или исходя из причин функциональной плотности, при этом будет сэкономлена стоимость и решена проблема 5-вольтовой совместимости.

Как показано на рисунке 4-2, результатом является "короткое позиционирование" - использование только 5-вольтовой платы и 5-вольтового соединителя и "длинное позиционирование" основанное на использовании 3-вольтового соединителя с 5-вольтовыми компонентами на универсальной плате (использующими двойные буферы напряжений) и 3-вольтового компонента на 3-вольтовой плате. Переход между этими реализациями - прежде всего универсальная плата имеющая как 5-вольтовые, так и 3-вольтовые разъёмы. Важный шаг этого перехода - возможность быстро перейти от 5-вольтовой плате к универсальной плате. Если допустимая масса 5-вольтовой технологии способна к охвату как систем 5В так и систем 3.3В,то это не будет препятствовать переходу к 3.3В

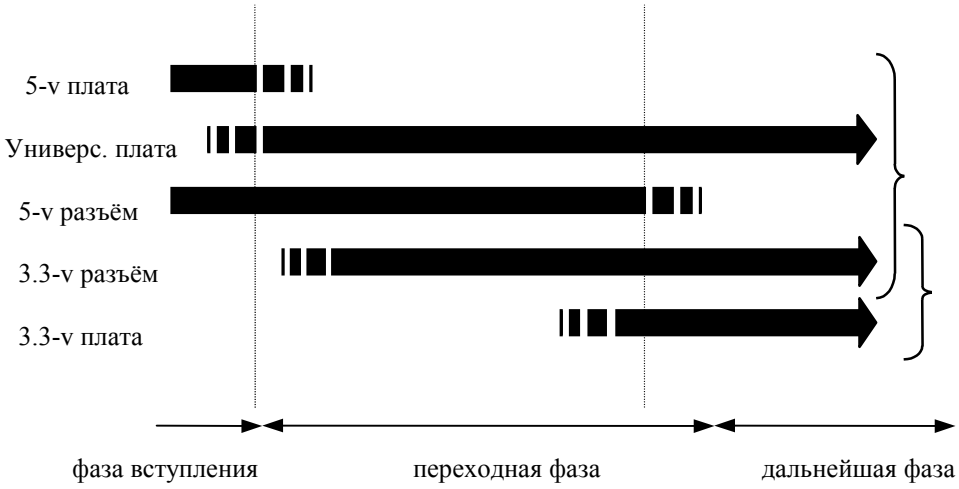


Рисунок 4-2 : Фазы 5-вольтовых и 3- вольтовых технологий

4.1.2 Спецификации динамических и статических характеристик

PCI - Шина имеет две электрические характеристики, которые мотивируют различный подход к определению характеристик буфера ввода - вывода. Первая: PCI - шина - это CMOS-шина, что означает, что устойчивые токи состояния (после переходных процессов) очень минимальны. Фактически, большая часть тока тратится на нагрузочных резисторах. Вторая: PCI основана на отраженной волне вместо случайной волновой передачи сигналов. Это означает, что драйверы шины обеспечивают включение шины наполовину требуемого высокого или низкого уровня. Электрическая волна распространяется по шине, отражается от незавершенного конца и идёт обратно к точке её происхождения, в связи с этим достигается удвоение начального напряжения, чтобы достигнуть требуемого уровня напряжения. Драйвер шины фактически находится в середине диапазона переключения в течение этого времени распространения, которое продолжается до 10 нс (одна треть цикла шины при 33 МГц).
Драйверы PCI- шины тратят относительно большую часть времени при быстром переключении, и постоянный ток минимален, так что типичный подход определения буферов, основанных на их возможности быть источником постоянного тока, бесполезен. Драйверы шины определяются скорее в терминах их характеристик переключения при переменном токе, чем при постоянном. Вольтамперная характеристика драйвера проходящая через активный диапазон переключения и есть спецификация. Эти вольтамперные характеристики нацелены на достижение приемлемого переключения в типичных конфигурациях шести нагрузок на материнской плате и двух слотах расширения, или двух нагрузках на материнской плате и четырех слотах расширения. Однако, возможно достигнуть различные целевые конфигурации в зависимости от фактической специфики оборудования, его размещения, установленного полного сопротивления материнской платы и т.д.

4.2. Спецификация компонентов

Этот раздел определяет электрические и временные параметры синхронизации для компонентов PCI. Описываются как сигналы 5-вольтовой среды передачи сигналов , так и 3-вольтовой. 5-вольтовая среда основана на абсолютных напряжениях переключения, чтобы быть совместимой с переключающими уровнями ТТЛ. Среда 3.3В, с другой стороны, основана на напряжениях переключения относительно Vcc , и это достигается с помощью КМОП. Цель электрической спецификации - соединение компонентов вместе непосредственно на планарной плате или плате расширения, без каких-либо внешних буферов или чего-либо подобного.

Назначение такой спецификации - функционирование компонентов внутри "коммерческого" диапазона параметров среды. Однако, это не препятствует наличию других действующих сред.

Буферы вывода PCI определяются в условиях их вольт-амперных характеристик. Ограничения приемлемых вольт-амперных характеристик обеспечивают максимальное полное выходное сопротивление, которое может достигать приемлемого первого порогового напряжения в типичных конфигурациях, также они обеспечивают минимальное полное выходное сопротивление, которое обеспечивает параметры отраженной волны внутри приемлемых пределов. Входы и выходы буфера имеют различные ВАХ, которые обеспечиваются параметрической спецификацией. Эффективная буферная производительность прежде всего определяется постоянной составляющей тока, которая определяет приемлемое первое пороговое напряжение, как сверху, так и снизу вместе с требуемыми токами, чтобы достигнуть этого напряжения в типичных конфигурациях. Переменная составляющая определяет устойчивые условия состояния, которые должны удерживаться, но в среде КМОП они являются минимальными, и не индицируют реальную эффективность вывода. Теневые области на ВАХ, показанных в Рисунке 4-3 определяют допустимый диапазон для характеристик вывода.

Параметры постоянного тока должны быть поддерживаемы в устойчивом состоянии. Параметры переменного тока нужно гарантировать при условиях переключения, которые могут представлять до 33 % цикла. Знак у всех текущих параметров (здесь: направление тока) оценивается относительно земли внутри компонента; то есть положительные токи текут в компонент, в то время как отрицательные токи текут из компонента. Поведение при сбросе (RESET) описано в разделе 4.3.2. (спецификация системы).

4.2.1. 5В - режим передачи сигналов

4.2.1.1. Спецификации постоянного тока

Таблица 4-1 описывает спецификации постоянного тока для 5-вольтовой среды передачи сигналов.

Таблица 4-1

Обозначение	Параметр	Условие	Min	Max	Ед. изм	Примечание
V_{cc}	Напряжение функционирования		4.75	5.25	V	
V_{ih}	Входное напряжение высокого уровня		2.0	Vcc+0.5	V	
V_{il}	Входное напряжение низкого уровня		-0.5	0.8	V	
I_{ih}	Входной ток утечки высокого уровня	V_{in} = 2.7		70	microA	1
I_{il}	Входной ток утечки низкого уровня	V_{in}=0.5		-70	microA	1
V_{oh}	Выходное напряжение высокого уровня	I_{out}=-2 mA	2.4		V	
V_{ol}	Выходное напряжение низкого уровня	I_{out}=3mA,6mA		0.55	V	2
C_{in}	Входная ёмкость			10	pF	3
C_{clk}	Ёмкость входа синхронизации		5	12	pF	
C_{idsel}	Ёмкость входа IDSEL			8	pF	4
L_{pin}	Индуктивность контакта			20	nH	5

ПРИМЕЧАНИЯ:

1. Входные токи утечки включают выходную утечку для всех двунаправленных буферов с тристабильными выходами.
2. Сигналы без нагрузочных резисторов должны иметь выходной ток низкого уровня = 3 mA. Сигналы с нагрузочными резисторами должны иметь этот ток = 6 mA. К числу последних относятся: FRAME#, TRDY#, IRDY#, DEVSEL#, STOP#, SERR#, PERR# и когда используются: LOCK#, AD[63::32], C/BE [7::4]#, PAR64, REQ64# и ACK64#.
3. Абсолютная максимальная емкость контакта для входа PCI равна 10pF (кроме CLK) за исключением устройств на материнской плате, у которых этот параметр может быть до 16 pF, в порядке установленном PGA-упаковкой. Это означает, что компонент для плат расширения должен будет использовать вариант для керамической PGA-упаковки (то есть PQFP, SGA, и т.д.).
4. Пониженная ёмкость на этом входном выводе позволяет устанавливать нерезистивную связь с AD[xx].
5. Это рекомендация, а не абсолютное требование. Фактическое значение нужно обеспечить в соответствии с инструкцией к компоненту.

Контакты, используемые для расширенного доступа к данным, AD[63::32], C/BE[7::4]# и PAR64, требуют или нагрузочных резисторов или входных буферов, потому что они не используются в транзакциях с 32-битными устройствами, и могут, следовательно, переместиться к пороговому уровню, вызывая колебания или высокую потерю мощности через входной буфер. Функция этих резисторов или буферов должна быть частью центрального ресурса материнской платы (не платы расширения) чтобы гарантировать непротиворечивое взаимодействие и избежать перегрузки нагрузочных резисторов. Когда 64-разрядная шина данных присутствует в устройстве, но не присоединена (как в случае 64-разрядной платы, подключенной в 32-разрядный PCI-слот), то PCI-компонент несёт ответственность за обеспечение отсутствия колебаний на входах и за отсутствие значительной потери мощности через входной буфер. Это может быть выполнено различными способами, например, смещением входных буферов, активно ведущих выводы непрерывно (так как они не соединены с чем-либо). Внешние резисторы на плате расширения, или любом решении, которое нарушает спецификацию входной утечки, запрещены. Сигнал REQ64# используется во время сброса, чтобы различить устройства, которые соединены с 64-разрядной шиной данных, и которые не соединены (см. раздел 4.3.2.).

4.2.1.2. Спецификации переменного тока

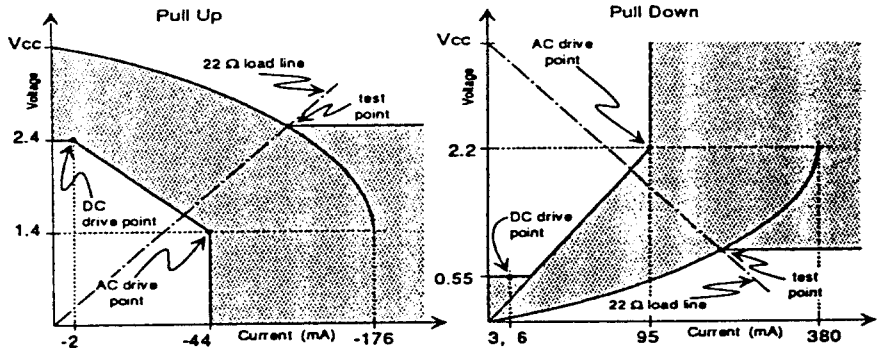
Таблица 4-2 описывает спецификации переменного тока для 5-вольтовой среды передачи сигналов.

Таблица 4-2

Обозначение	Параметр	Условие	Min	Max	Ед. изм.	Примечание
I_{oh} (перем.ток)	Переключающий ток высокого уровня	$0<V_{out}<=1.4$	-44		mA	1
		$1.4<V_{out}<2.4$	$-44+(V_{out}-1.4)/0.024$	Eqt'n A	mA	1,2,3
	(Точка теста)	$V_{out}=3.1$		-142	mA	3
I_{ol} (перем.ток)	Переключающий ток низкого уровня	$V_{out}>=2.2$	95		mA	1
		$2.2>V_{out}>0.55$	$V_{out}/0.023$	Eqt'n B	mA	1,3
	(Точка теста)	$V_{out}=0.71$		206	mA	3
	Наименьший ток замыкания	$-5<V_{in}<=-1$	$-25+(V_{in}+1)/0.015$		mA	
	Время нарастания выходного напряжения	от 0.4V до 2.4V	1	5	V/ns	4
	Время спада выходного напряжения	от 2.4V до 0.4V	1	5	V/ns	4

ПРИМЕЧАНИЯ:

1. Обратите внимание на BAX на рисунке 4-3. Характеристике переключающего тока для REQ# и GNT# разрешается быть одной половиной из той, что определена здесь; то есть, половина драйверов вывода может использоваться для этих сигналов. Эта спецификация не относится к CLK и RST#, которые являются системными выводами. Спецификация "переключающего тока высокого уровня" не относится к SERR#, который является открытым выходом.
2. Обратите внимание, что сегмент минимальной кривой тока выведен из точки AC(переменный ток) непосредственно к точке DC(постоянный ток) а не к шине напряжения (как это выполнено в спускающейся кривой).
3. Максимальные требования к току должны быть соблюдены, поскольку драйверы обеспечивают напряжение выше первого порогового напряжения (точка AC). Уравнения, определяющие эти максимумы (для A и B) определяются соответствующими диаграммами на Рисунке 4-3. Максимумы должны обеспечиваться в соответствии с проектом и этими уравнениями. Чтобы облегчить тестирование компонента, максимальная тестовая точка тока определяется для каждой стороны драйвера вывода.
4. Минимальная скорость нарастания/спада (самый длинный край сигнала) должна обеспечиваться всеми PCI-устройствами. Максимальная скорость нарастания/спада (самый короткий край сигнала) - это то к чему нужно стремиться. Поставщики компонента должны иметь в виду, что чем короче фронт/спад сигнала, тем более вероятно, что это повлечёт помехи, которые могут вызывать ошибки передачи сигналов в системе. Проектировщики материнской платы должны иметь в виду, что времена нарастания/спада должны быть меньше чем максимальное требование к ним, и должны гарантировать, что моделирование целостности сигнала обеспечивает это.



Уравнение A:

*Ioh = 11.9 * (Vout - 5.25) * (Vout + 2.45)*

Для Vcc > Vout > 3.1v

Уравнение B:

*Iol = 78.5 *Vout* (4.4 - Vout)*

Для 0v < Vout < 0.71v

Рисунок 4-3: ВАХ для 5-вольтовой сигнальной среды.

Требуется, чтобы входы были "посажены" на землю. Контакты на 5-вольтовой шине необязательны, но может быть необходимо защитить 3-вольтовые устройства ввода данных(см. "Максимальные значения переменного тока" ниже). Контакты на шине 3.3V никогда не должны использоваться в 5-вольтовой среде передачи сигналов. Когда используются сдвоенные шины питания, могут быть установлены диодные связи между линиями. Эти диоды могут стать проводящими, если одна из шин питания выходит на мгновение из допустимых параметров. Диоды присоединяются к шине питания также как устройства нагрузки, они должны быть способны противостоять короткому замыканию пока драйверы могут находиться в 3-м состоянии. См. раздел 4.3.2. для подробной информации.

4.2.1.3. Максимальные значения переменного тока и защита устройств

Спецификация теста переменного тока на максимальность включена здесь как рекомендация тестирования, потому что PCI-среда содержит много реактивных элементов и, вообще, должна обрабатываться как незавершенная среда линий электропередачи. Базисная предпосылка среды требует, чтобы сигнал отразился в конце линии и возвратился драйверу прежде, чем сигнал рассматривается активным.

Как следствие этой среды, при некоторых особенностях драйверов, топологий устройств, полного сопротивления платы, и т.д., напряжение на контактах PCI- устройств повысится от "земли" до некоторого уровня вплоть до Vcc, оно может достигать значительных величин. Технология, используемая для выполнения PCI может изменяться от поставщика к поставщику, так что не гарантируется, что технология может противостоять этим эффектам. Эта спецификация теста обеспечивает фактически наихудший случай среды переменного тока, против которой надежность устройства может быть оценена.

Все входы: двунаправленные, и выводы с тремя состояниями, используемые на каждом PCI-устройстве должны быть способны к внезапному испытанию следующим тестом. Тест проводится с эквивалентом источника напряжения с нулевым полным сопротивлением присоединяя резистор последовательно непосредственно в каждый вход или три-стабильный контакт выхода PCI- устройства. Форма сигнала, обеспечиваемая источником напряжения (или открытое напряжение схемы, включающей резистор) и значение резистора показывается на рисунке 4-4.

Этот тест покрывает только эксплуатационные режимы переменного тока; условия постоянного тока определены в другом месте.

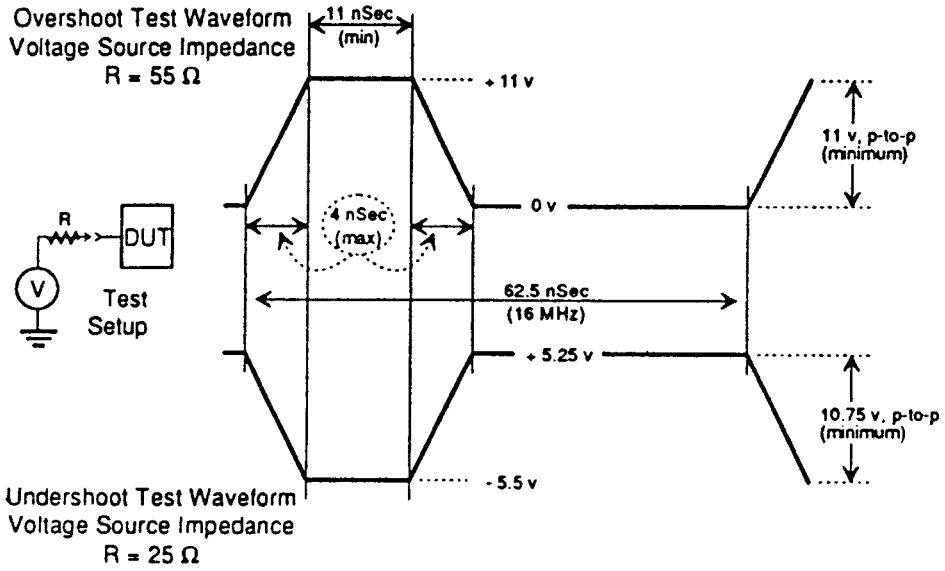


Рисунок 4-4: Форма тестового сигнала для 5-вольтовой среды передачи сигналов

4.2.2. 3.3В - режим передачи сигналов

4.2.2.1. Спецификации постоянного тока

Таблица 4-3 подводит итог спецификаций постоянного тока для передачи сигналов 3.3В.

Таблица 4-3

Обозначение	Параметр	Условие	Min	Max	Ед. изм.	Примечание
Vcc	напряжение функционирования		3.0	3.6	V	
Vih	входное напряжение высокого уровня		0.475Vcc	Vcc+0.5	V	
Vil	входное напряжение низкого уровня		-0.5	0.325Vcc	V	
Vipu	входное падение напряжения		0.7Vcc		V	1
Iil	входной ток утечки	0<Vin<Vcc		+-10	microA	2
Voh	выходное напряжение высокого уровня	Iout=-500microA	0.9Vcc		V	
Vol	выходное напряжение низкого уровня	Iout=1500microA		0.1Vcc	V	
Cin	ёмкость входного контакта			10	pF	3
Cclk	ёмкость входа CLK		5	12	pF	4
Cidsel	ёмкость входа IDSEL			8	pF	4
Lpin	Индуктивность входа			20	nH	5

ПРИМЕЧАНИЯ:

1. Эту спецификацию нужно гарантировать в соответствии с проектом. Это - минимальное напряжение, для которого рассчитываются нагрузочные резисторы по "плавающей" сетке. Приложения, чувствительные к статическому напряжению должны гарантировать, что входной буфер проводит минимальный ток при этом входном напряжении
2. Входные токи утечки включают выходные для всех двунаправленных буферов с выводами с тремя состояниями.
3. Абсолютная максимальная емкость входного контакта PCI=10 pF (кроме CLK) за исключением устройств на материнским плате, для которых она может быть до 16 pF, согласно стандартам PGA-упаковки. Фактически это означает, что компонент для плат расширения должен использовать варианты для керамической PGA- упаковки: то есть, PQFP, SGA и т.д.
4. Более низкая емкость только на этом входе позволяет нерезистивную связь с AD[xx].
5. Это - рекомендация, не абсолютные требования. Фактическое значение нужно обеспечить в соответствии с инструкцией компонента.

Контакты, используемые для расширенной шины данных, AD[63::32], C / BE [7::4]# и PAR64, требуют или нагрузочных резисторов или входных буферов, потому что они не используются в транзакциях с 32-разрядными устройствами, и могут следовательно "плавать" к пороговому уровню, вызывая колебания или высокое падение напряжения через входной буфер. Функция этого нагрузочного резистора или буфера должна быть частью центрального ресурса материнской платы (не платы расширения) чтобы гарантировать непротиворечивое решение и избегать перегрузки нагрузочного резистора.

Когда 64-разрядная шина данных присутствует на устройстве, но не соединена (как в 64-разрядной плате, подключенной в 32-разрядный PCI-слот), то PCI-компонент ответственен за то чтобы обеспечить отсутствие колебаний на входе и большое падение напряжения на входном буфере. Это может быть выполнено различными способами, как например,

смещение входного буфера непосредственно управляющего выходом (так как они не соединены с чем-либо). Внешние резисторы на плате расширения, или любом решении, которое нарушает спецификацию входных токов, запрещены. Сигнал REQ64# используется в течение сброса, чтобы различить устройства соединенные с 64-разрядной шиной данных и не соединённые с ней (см. раздел 4.3.2.).

4.2.2.2. Спецификации переменного тока

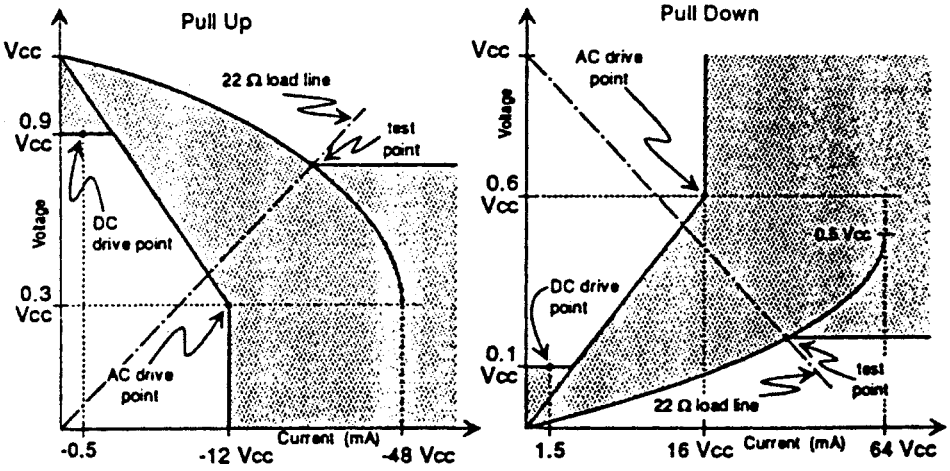
Таблица 4-4 подводит итог спецификаций переменного тока для передачи сигналов 3.3V.

Таблица 4-4

Обозначение	Параметр	Условие	Min	Max	Ед. изм.	Примечание
I_{oh} (перем. ток)	переключающий ток высокого уровня	$0<V_{out}<=0.3V_{cc}$	$-12V_{cc}$		mA	1
		$0.3V_{cc}<V_{out}<0.9V_{cc}$	$-17.1(V_{cc}-V_{out})$	Eqt'n C	mA	1,2
	(Точка теста)	$V_{out}=0.7Vcc$		$-32V_{cc}$	mA	2
I_{ol} (перем. ток)	переключающий ток низкого уровня	$V_{cc}>V_{out}>=0.6V_{cc}$	$16V_{cc}$		mA	1
		$0.6V_{cc}>V_{out}>0.1V_{cc}$	$26.7V_{out}$	Eqt'n D	mA	1,2
	(Точка теста)	$V_{out}=0.18V_{cc}$		$38V_{cc}$	mA	2
I_{cl}	замыкающий ток низкого уровня	$-3<V_{in}<=-1$	$-25+(V_{in}+1):0.015$		mA	
I_{ch}	замыкающий ток высокого уровня	$V_{cc}+4>V_{in}>=V_{cc}+1$	$25+(V_{in}-V_{cc}-1):0.015$		mA	
T_r	время нарастания выходного сигнала	$0.2V_{cc}$-$0.6V_{cc}$	1	4	V:ns	3
T_f	время спада выходного сигнала	$0.6V_{cc}$-$0.2V_{cc}$	1	4	V:ns	3

ПРИМЕЧАНИЯ:

1. Обратите внимание на BAX на Рисунке 4-5. Характеристике переключающего тока для REQ# и GNT# разрешается быть одной половиной из той, что определена здесь; то есть, половина драйверов вывода может использоваться для этих сигналов. Эта спецификация не относится к CLK и RST#, которые являются системными выводами. Спецификация "переключающего тока высокого уровня" не относится к SERR#, который является открытым выходом.
2. Максимальные требования к току должны быть соблюдены, поскольку драйверы обеспечивают напряжение выше первого порогового напряжения (точка AC). Уравнения, определяющие эти максимумы (для C и D) определяются соответствующими диаграммами на Рисунке 4-5. Максимумы должны обеспечиваться в соответствии с проектом и этими уравнениями. Чтобы облегчить тестирование компонента, максимальная тестовая точка тока определяется для каждой стороны драйвера вывода.
3. Минимальная скорость нарастания/спада (самый длинный край сигнала) должна обеспечиваться всеми PCI-устройствами. Максимальная скорость нарастания/спада (самый короткий край сигнала) - это то к чему нужно стремиться. Поставщики компонента должны иметь в виду, что чем короче фронт/спад сигнала, тем более вероятно что это повлечёт помехи, которые могут вызывать ошибки передачи сигналов в системе. Проектировщики материнской платы должны иметь в виду, что времена нарастания/спада должны быть меньше чем максимальное требование к ним, и должны гарантировать, что моделирование целостности сигнала обеспечивает это.



Уравнение C:

*Ioh = (98.0/Vcc) * (Vout - Vcc) * (Vout + 0.4Vcc)*

Для Vcc > Vout > 0.7Vcc

Уравнение D:

*Iol = (256/Vcc)*Vout*(Vcc -Vout)*

Для 0v < Vout < 0.18Vcc

Рисунок 4-5: ВАХ для 3.3-вольтовой среды передачи сигналов

Требуется, чтобы входы были "посажены" на "землю". Когда используются сдвоенные шины питания, могут быть установлены диодные связи между линиями. Эти диоды могут стать проводящими, если одна из шин питания выходит на мгновение из допустимых параметров. Диоды присоединяются к шине питания также как устройства нагрузки, они должны быть способны противостоять короткому замыканию пока драйверы могут находиться в 3-м состоянии. См. раздел 4.3.2. для подробной информации.

4.2.2.3. Максимальные значения переменного тока и защита устройств

См. раздел "Максимальные значения переменного тока " в 5-вольтовой среде передачи сигналов. Все входы, двунаправленные выводы, и выводы с тремя состояниями, используемые на каждом PCI-устройстве должны быть способны к внезапному испытанию следующим тестом. Тест проводится с эквивалентом источника напряжения с нулевым полным сопротивлением и присоединением нагрузочного резистора непосредственно на каждый вход или три-стабильный выход PCI-устройства. Форма сигнала, обеспечиваемая источником напряжения (или напряжение схемы, включающей резистор) и номинал резистора показывается на рисунке 4-6. Этот тест охватывает эксплуатационные режимы только переменного тока, условия постоянного тока определены в другом месте.

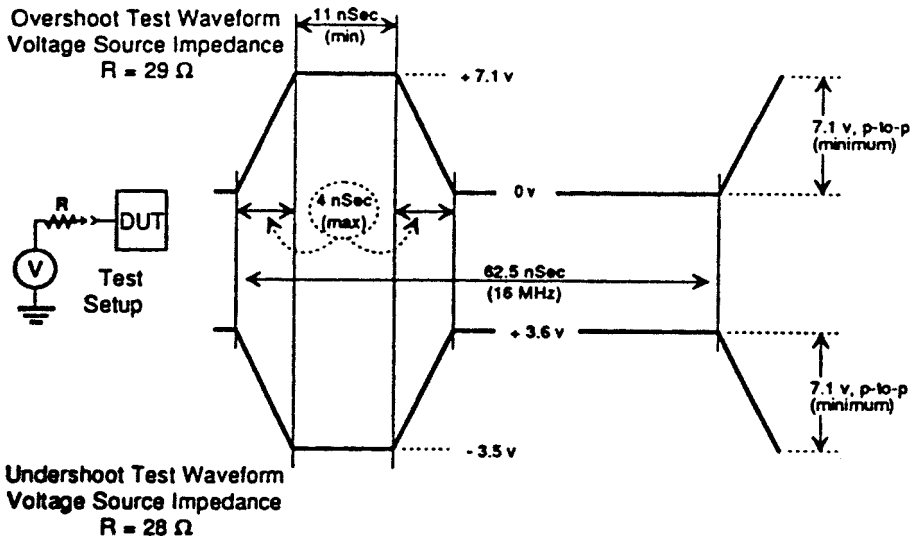


Рисунок 4-6: Форма тестового сигнала для 3.3-вольтовой среды передачи сигналов

4.2.3. Спецификация синхронизации

4.2.3.1. Спецификация CLK

Форма сигнала CLK должна быть определена для каждого PCI-компонента в системе. В случае плат расширения, соглашение о спецификации CLK измеряется в компоненте платы расширения, а не в слоте соединителя. Рисунок 4-7 показывает форму сигнала CLK и требуемые точки измерения как для 5-вольтовой среды передачи сигналов, так и для 3.3-вольтовой. Таблица 4-5 подводит итог спецификации CLK.

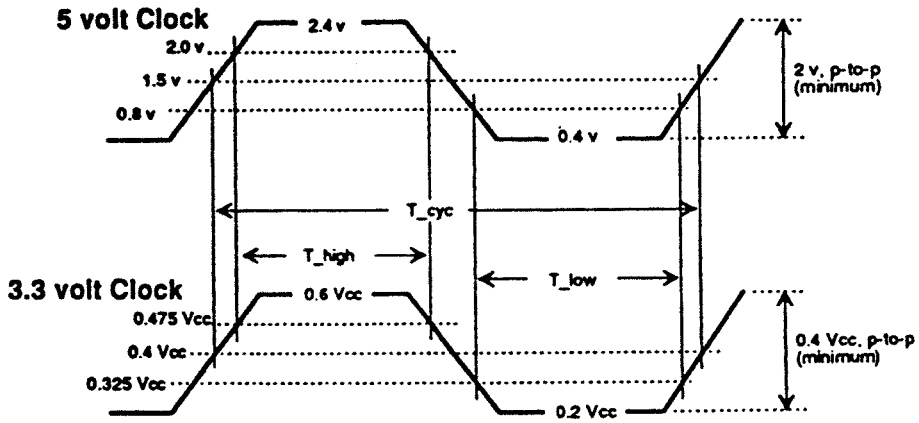


Рисунок 4-7: Форма сигнала CLK

Таблица 4-5: Спецификация CLK

Обозначение	Параметр	Min	Max	Ед. Изм.	Примечание
Tcyc	время цикла CLK	30	∞	ns	1
Thigh	время нижнего уровня CLK	12		ns	
Tlow	время верхнего уровня CLK	12		ns	
-	время нарастания/спада CLK	1	4	V/ns	2

ПРИМЕЧАНИЯ:

1. Вообще, все PCI-компоненты должны работать при любой тактовой частоте между постоянным током и 33 MHz. Параметры устройства при частотах до 16 MHz можно гарантировать в соответствии с проектом даже не проверяя. Тактовая частота может быть изменена в любое время в течение работы системы, поскольку фронт синхроимпульса остается "чистым"(без колебаний) а минимальный цикл и времена высокого и низкого уровней не нарушаются. CLK может быть остановлен только в состоянии нижнего уровня. Разница на этот счёт позволяется для компонентов, которые могут функционировать при любой фиксированной частоте до 33 MHz, и могут предписывать стратегию неизменной частоты.
2. Времена нарастания/спада определены в терминах параметров фронта/спада, измеряемого в V/ns. Этот параметр должен быть обеспечен поперек минимальной части двойной амплитуды формы сигнала CLK как показано на рисунке 4-7.

4.2.3.2. Параметры синхронизации

Таблица 4-6 определяет параметры синхронизации как для 5-вольтовой, так и для 3-вольтовой среды передачи сигналов.

Таблица 4-6 Параметры синхронизации для 5В и 3.3В

Обозначение	Параметр	Min	Max	Ед.Изм.	Примечание
Tval	задержка между CLK и установкой сигналов на шине	2	11	ns	1,2,3
Tval(ptp)	задержка между CLK и установкой сигнала (точка-точка)	2	12	ns	1,2,3
Ton	задержка до активизации	2		ns	1
Toff	задержка до деактивации		28	ns	1
Tsu	задержка между установкой сигналов на входе и CLK на шине	7		ns	3,4
Tsu(ptp)	задержка между установкой сигналов на входе и CLK (точка-точка)	10,12		ns	3,4
Th	время удержания от фронта CLK	0		ns	4
Trst	время действия сброса после стабилизации питания	1		ms	5
Trst-clk	время действия сброса после стабилизации CLK	100		microS	5
Trst-off	задержка между активизацией сброса до его дестабилизации		40	ns	5,6
Trrsu	время между установкой REQ64# и RST#	10Tcyc		ns	
Trrh	время удержания между RST# и REQ64#	0	50	ns	

ПРИМЕЧАНИЯ:

1. См. условия измерения синхронизации на рисунке 4-8.
2. Минимальные времена измеряются с эквивалентной нагрузкой в 0 pF; максимальные времена измеряются с 50-пикофарадной эквивалентной нагрузкой. Фактическая тестовая емкость может изменяться, но результаты должны быть соотнесены с этими спецификациями.
3. REQ# и GNT# - двухточечные сигналы, они имеют различную задержку до установки сигнала на выходе и различные времена установки, чем у шинных сигналов. GNT# имеет установку 10; REQ# имеет установку 12. Все другие сигналы шинные.
4. См. условия измерения синхронизации на рисунке 4-9.
5. RST# устанавливается и деактивизируется асинхронно относительно CLK. См. раздел 4.3.2 для подробной информации.
6. Все драйверы вывода должны быть в неопределённом состоянии, когда RST# активен.

4.2.3.3. Измерение и условия теста

Рисунки 4-8 и 4-9 определяют условия, при которых установлены параметры синхронизации. Тест компонента гарантирует, что параметры синхронизации удовлетворяют требованию минимального времени нарастания/спада(самый длинный край сигнала) и колебаниям напряжения. Проект должен гарантировать, что при минимальной тактовой частоте обеспечиваются характеристики самого короткого края сигнала и колебания напряжения. Кроме того, проект должен гарантировать правильность входных операций по части колебаний входного напряжения и параметров нарастания/спада сигналов, которые не удовлетворяют определенным условиям теста.

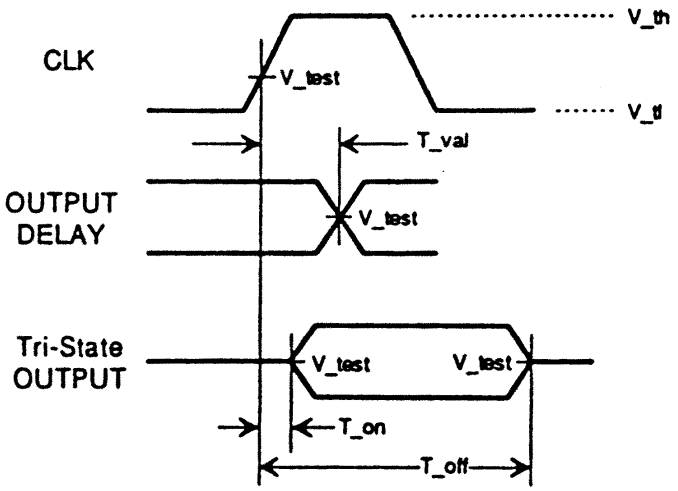


Рисунок 4-8: Условия измерения сигналов на выходах относительно синхронизации

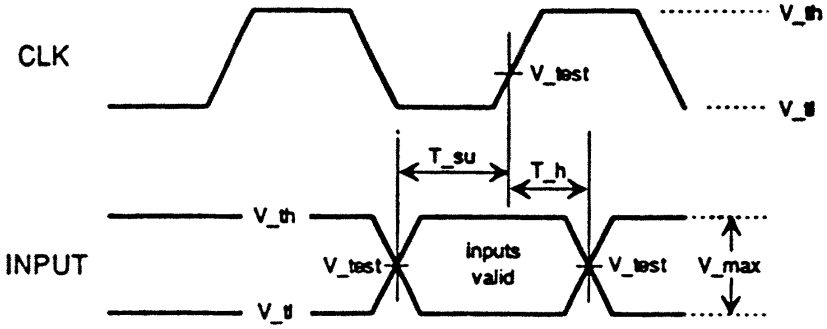


Рисунок 4-9: Условия измерения сигналов на входах относительно синхронизации

Таблица 4-7: Измерение и параметры тестовых условий

Обозначение	5-вольтовая среда	3.3-вольтовая среда	Ед.Изм.
Vth	2.4	0.6Vcc	V(примечание)
Vtl	0.4	0.2Vcc	V(примечание)
Vtest	1.5	0.4Vcc	V
Vmax	2.0	0.4Vcc	V(примечание)
параметр края входного сигнала	1V/нс		

ПРИМЕЧАНИЕ:

Входной тест для 5-вольтовой среды выполняется при 400-милливольтовой перегрузке (выше Vh и Vl); тест для среды 3.3V выполняется при 0.125Vcc-милливольтовой перегрузке. Параметры синхронизации должны быть выполнены без большей перегрузки чем эта. Vmax определяет максимальную амплитуду сигнала, позволенную для тестирования входной синхронизации.

4.2.4. Спецификация, обеспечиваемая поставщиками

Во временных рамках PCI, много поставщиков системы будут делать электрическое моделирование PCI-компонентов. Это гарантирует, что реализации системы доступны для производства, и эти компоненты будут использоваться правильно. Чтобы помочь облегчить эти усилия, также как обеспечить полную информацию о компоненте, поставщики компонентов должны делать следующую информацию доступной в их инструкциях:

1. Емкость для каждого из контактов.
2. Индуктивность для каждого из контактов
3. Выходные BAX при условиях переключения. Две BAX должны быть представлены для каждого используемого типа выхода: одна для описания высокого уровня, другая для описания низкого уровня. Обе должны показать, наилучшие, типичные и наихудшие характеристики. Также, при критическом напряжении питания, диапазон напряжения должен охватывать от -5 до l0V для 5-вольтовой среды передачи сигналов и от -3 до 7V для передачи сигналов 3.3V.
4. Входные BAX при условиях переключения. BAX входной структуры, когда вывод тристабилен также важна. Этот график должен также показать наихудшие, типичные и наилучшие BAX при диапазоне от 0 до Vcc.
5. Времена нарастания / спада для каждого типа выхода.
6. Все необходимые данные, включая температурный диапазон, максимальное значение постоянного тока, и т.д. В дополнение к информации для этого компонента, поставщики соединителей должны предлагать доступные точные имитационные модели PCI-соединителей.

4.2.5. Рекомендации по расположению контактов.

Этот раздел описывает рекомендуемый тип расположения контактов для PCI- компонента. Так как размеры платы расширения ограничены, то проблемы размещения значительно минимизируются если вывод компонента совпадает точно с контактом на плате. Компонент для использования только на материнских платах должен также следовать требованию наличия одного и того же сигнала на соответствующих контактах. Рисунок 4-10 показывает рекомендуемый тип расположения контактов PQFP PCI-компонента; обратите внимание, что имеется точное соответствие контактов на плате порядку следования сигналов (SDONE, и SBO# не показаны). Размещение и количество контактов питания зависит от типа устройства.

Дополнительные сигналы, необходимые в версиях с 64-битными шинами должны быть расположены на компоненте против часовой стрелки в том же самом порядке, в котором они расположены на 64-битном разъёме расширения.

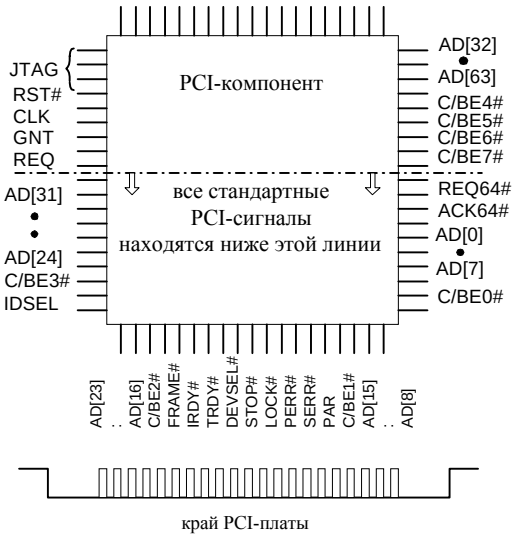


Рисунок 4-10: Расположение выводов для PQFP PCI-компонента

Размещение ввода IDSEL близко насколько возможно к AD[31::16] позволяет нерезистивную[2] связь IDSEL с соответствующей адресной линией с маленькой дополнительной нагрузкой. Обратите внимание, что этот контакт имеет меньшую ёмкость, чем та которая могла бы сдерживать её размещение в пакете.

4.3. Спецификация материнской платы

4.3.1. Перекос синхронизации

Максимальный допустимый перекос синхронизации - 2ns. Эта спецификация применяется не только в одиночной пороговой отметке, но и во всех точках на фронте синхроимпульса, которые попадают в диапазон переключения, определенном в Таблице 4-8 и рисунке 4-11. Максимальный перекос измеряется между любыми двумя компонентами[3], а не между контактами. Чтобы правильно оценить перекос синхронизации, проектировщик системы должен принять во внимание распределение синхронизации на плате расширения, которая определена в разделе 4.4.

[2] Нерезистивные связи IDSEL к одной из AD[xx]-линий создают техническое нарушение одиночной загрузки. PCI протокол обеспечивает предварительное получение адреса в циклах конфигурации и рекомендуется, чтобы это выполнялось, чтобы позволить резистивную связь IDSEL. В отсутствии этого, мощность сигнала нужно уменьшить налоги для дополнительной нагрузки IDSEL.

[3] Может быть дополнительный источник перекоса синхронизации, который проектировщик системы должен будет адресовать. Этот перекос синхронизации происходит между двумя компонентами, которые имеют точки отключения входа синхронизации на границы Vil -Vih диапазона. При некоторых обстоятельствах, это может способствовать перекосу синхронизации как описано здесь. Во всех случаях, суммарный перекос синхронизации должен быть ограничен.

Таблица 4-8: Параметры скоса синхронизации

Обозначение	5-вольтовая среда	3.3-вольтовая среда	Ед. Изм.
Vtest	1.5	0.4Vcc	V
Tskew	2(max)	2(max)	ns

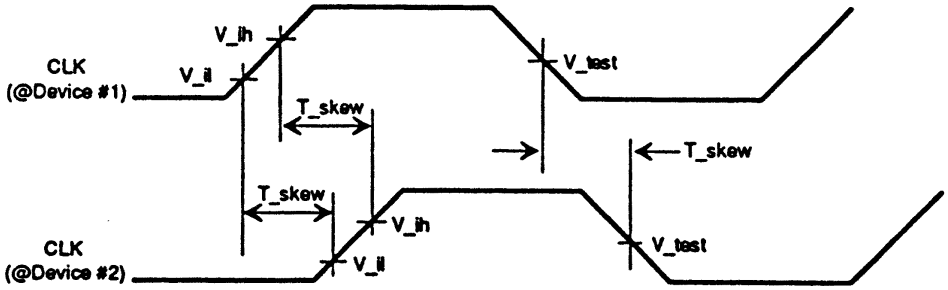


Рисунок 4-11: Диаграмма скоса синхронизации

4.3.2. Сброс (RESET)

Активизация и деактивизация сигнала сброса PCI (RST#) асинхронна относительно CLK. Фронт/спад сигнала RST# должны быть "чистыми". Спецификация PCI не препятствует реализации синхронного RST#, если это необходимо. Параметры синхронизации для сброса содержатся в Таблице 4-6, за исключением параметра Tfail. Этот параметр описывает реакцию системы на отказ одной или обеих шин питания. Если это происходит, диодные пути могли бы замкнуть накоротко активные входные буферы. Следовательно, RST# устанавливается при сбое питания чтобы привести буферы вывода в неопределённое состояние.

Минимальное значение T_{fail} :

- 500 нс (максимум) для любой шины питания, выходящей из строя (превышение определенных допусков больше, чем на 500mV).
- 100 нс (минимум) для 5V - шины, при падении напряжения ниже чем на шине 3.3V больше чем на 300mV.

Система должна выставить RST# в течение включения питания или в случае сбоя питания. После того как RST# выставлен, PCI-компонент не реагирует на сброс, пока T_{rst} и $T_{rst-clk}$ не будут выполнены. Рисунок 4-12 показывает синхронизацию сигнала RST#.

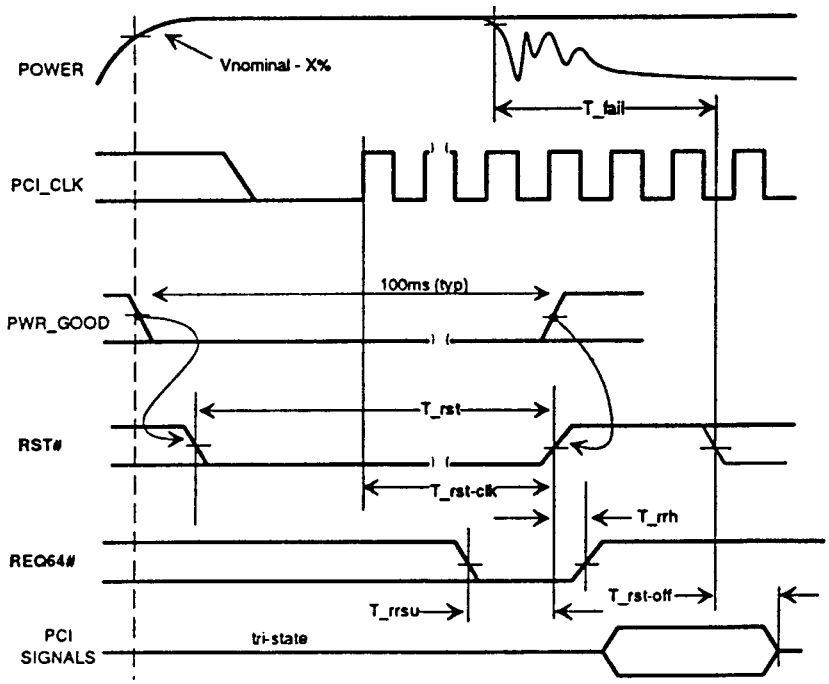


Рисунок 4-12: Синхронизация сброса

Сигнал REQ64# используется в течение сброса, чтобы различить компоненты, которые соединены с 64-разрядной шиной данных и которые не соединены. Сигнал REQ64# находится на шине устройств (включая PCI-слоты) которые поддерживают 64-разрядные данные. Этот сигнал имеет одиночный нагрузочный резистор на материнской плате. На PCI-слотах расширения, которые не поддерживают 64-разрядные данные, сигнал REQ64# не находится на шине или отсоединён, но имеет собственный, отдельный нагрузочный резистор. На REQ64# должен быть низкий уровень в течение времени, когда RST# выставлен. Устройства, которые видят REQ64# выставленным в течение сброса, соединены с 64-разрядной шиной данных, в то время как те, которые не видят REQ64# выставленным, не соединены. Эта информация может использоваться компонентом, чтобы стабилизировать плавающие входы в течение времени выполнения, как описано в разделах 4.2.1.1. и 4.2.2.1.

В течение сброса, REQ64# имеет установку и требования времени задержки относительно фронта[4] RST#, REQ64# асинхронный относительно CLK в течение сброса.

4.3.3. Нагрузка

Сигналы управления PCI всегда требуют, чтобы нагрузочные резисторы имели устойчивые номиналы, когда шина активно не используется. Это сигналы: FRAME#, TRDY#, IRDY#, DEVSEL#, STOP#, SERR#, PERR# и, когда используются: LOCK#, REQ64#, и ACK64#. Двухточечные сигналы и 32-разрядные разделённые сигналы не

[4] Это позволяет REQ64# быть распознанным на спаде RST#.

требуют этих резисторов: шина гарантирует их стабильность. Сигналы 64-разрядной шины данных AD[63::32], C/BE[7::4]# и PAR64 должны также быть нагружены, когда они соединены. Когда они оставлены несоединенными (как в 64-разрядной плате в 32-разрядном разъёме) компонент непосредственно должен взаимодействовать с плавающим входом как описано в разделе 4.2.

Формулы для минимума и максимума нагрузочных резисторов приводятся ниже. R_{min} прежде всего определяется I_{ol}, выходным постоянным током низкого уровня, в то время как число нагрузок имеет только вторичный эффект. С другой стороны, R_{max} прежде всего определяется числом представленных нагрузок. Спецификация предусматривает минимальное значение R, которое вычислено основанным на 16 нагрузках (самый плохой случай), и типичное значение R, которое вычисляется как максимальное значение R при 10 нагрузках. Максимальное значение R определяется только формулой и будет самым высоким в системе с самым маленьким числом нагрузок.

Rmin= [Vcc (максимум) -Vol] / [Iol + (16Iil)], где 16=max число загрузок

*Rmax= [Vcc (минимум) -Vx] / [num_loads * Iih],*

где Vx=2.7v для 5-вольтовой передачи сигналов, и Vx=0.7Vcc для передачи сигналов 3.3V.

Таблица 4-9 обеспечивает минимальные и типичные значения как для 5-вольтовой так и для 3-вольтовой среды передачи сигналов. Типичные значения уменьшены для 10%-х резисторов в номинальных значениях.

Таблица 4-9: Минимальные и типичные значения нагрузочных резисторов

Сигнальная среда	Rmin	Rtypical	Rmax
5V	963 Ohm	2.7 Kohm @ 10%	см. формулу
3.3V	2.42 KOhm	8.2 Kohm @ 10%	см. формулу

4.3.4. Питание

4.3.4.1. Требования к питанию

Все PCI-разъёмы требуют наличия четырех шин питания: +5V,+ 3.3V, + 12V, и -12V. Системы использующие среду передачи сигналов 3.3V всегда требуют, чтобы присутствовали все четыре шины в каждой системе, с токами, определенными в таблице 4-10. Системы, использующие 5-вольтовую среду передачи сигналов могут также использовать или 3.3V, или обеспечивать средства для добавления этого позже, чтобы поддерживать платы расширения, которые этого требуют, и должны обеспечить оставшиеся три шины питания для каждой системы.

Требования к току на контакте для двух 12-вольтовых шин приводятся в таблице ниже. Нет никаких специфических системных требований для тока на контактах 5-вольтовой и 3-вольтовой шине: это зависит от системы. Система обеспечивает общие требования питания для плат расширения, которые могут быть распределены между контактами произвольным способом. Контакты PRSNTn# на разъёме позволяют системе оценивать требования питания каждой платы, и определять, пригодна ли установленная конфигурация в общих требованиях питания. См. раздел 4.4.4.1. для дальнейших подробностей.

Таблица 4-10 определяет допуски шин питания.

Таблица 4-10

Шина питания	Платы расширения(короткие и длинные)
5V +- 5%	максимум 5А(зависит от системы)
3.3V +- 0.3V	максимум 7.6А(зависит от системы)
12V +- 5%	500mA
-12V +- 10%	100mA

4.3.4.2. Последовательность

Не имеется никакой определенной последовательности, в которой четыре шины питания активизируются или дезактивируются. Они могут включаться или выключаться в любом порядке. Система должна установить RST# при включении и всякий раз, когда 5V-шина или шина 3.3V выходит из строя (см. Раздел 4.3.2). В течение сброса, все сигналы PCI доводятся до "безопасного" состояния, как описано в Разделе 4.3.2.

4.3.4.3. Изоляция

Все шины питания должны быть изолированы от земли чтобы организовать таким способом управление токами переключения (d I/d t), которым подвергнута шина. Это зависит от платформы и не детализируется в спецификации.
Контакты 3.3В в PCI-разъёмах (даже если они не являются питающими) обеспечивают путь возврата переменного тока и должны быть «шинированы» вместе на материнской плате, предпочтительно при питании 3.3В, и изолированы от "земли" в соответствии с высокоскоростными методами передачи сигналов. Чтобы гарантировать адекватный возвратный путь переменного тока, рекомендуется, чтобы 12 быстродействующих конденсаторов ёмкостью 0.01microF были равномерно расположены на шине 3.3В.

4.3.5. Распределение синхронизации системы

При вычислении общего количества PCI- нагрузок, особое внимание должно быть уделено максимальной длине дорожки на плате и нагрузке плат расширения, как описано в разделе 4.4.3. Также, максимальная емкость контакта в l0 pF должна быть принята для плат расширения, в то время как фактическая емкость штырька может использоваться устройств на плате.
Общий период синхронизации может быть разделен в четыре части. Допустимая выходная задержка (Tval) и входное время установки (Tsu) определяется спецификацией компонента. Суммарный перекос синхронизации (Tskew) и время распространения по шине (Tprop) - системные параметры. Tprop определен как 10 ns, но может увеличиваться до 11, понижая перекос синхронизации; то есть сумма Tprop и Tskew вместе не может превышать 12 ns, однако, ни при каких обстоятельствах Tskew не может превышать 2 ns. Tprop измеряется как показано на рисунке 4-13. Он начинается во время перехода сигнала входного буфера через пороговую точку (Vtest в Рисунке 4-12). Заканчивается он, когда сигнал самого медленного входа пересекает Vih (высокий уровень) или Vil (низкий уровень) и никогда не возвращается обратно через этот уровень снова. Должна соблюдаться определённая предосторожность при точной оценке этой точки синхронизации. Обратите внимание, что входная буферная синхронизация тестируется с некоторой перегрузкой (Vih и Vil).Это может быть необходимо, чтобы гарантировать входную синхронизацию. Например, вход не может считаться правильным (и следовательно Tprop все еще продолжается) если он поднимается до Vih и не возвращается через Vih

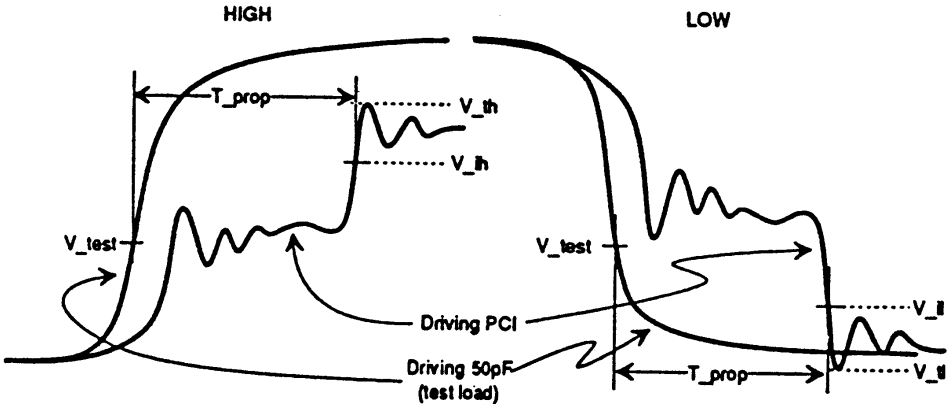


Рисунок 4-13: Измерение T_{prop} (см. таблицу 4-7 для значений параметра)

4.3.6. Конструктивные требования

4.3.6.1. Маршрутизация и размещение четырехслойных плат

Контакты питания размещены на разъёме, чтобы облегчить размещение на четырехслойных материнских платах. Питание может подводиться на изолированный участок разъёма 3.3В в 5-вольтовой плате, который объединяет все 3.3-вольтовые контакты PCI-разъёма, и может иметь разводку питания к разъёму питания. Хотя это стандартная методика, разводка высокоскоростных сигналов над этим слоем может вызывать проблемы целостности сигнала. Зазор в плоскости платы прерывает путь возврата переменного тока для сигнала, создавая неоднородность полного сопротивления.

Рекомендуемое решение состоит в том, чтобы упорядочивать сигнальные слои так, чтобы никакие высокоскоростные сигналы (например 33 МГц) не принадлежали к обоим слоям. Дорожки сигнала должны также оставаться полностью над плоскостью 3.3В или полностью над 5-вольтовой плоскостью. Сигналы, которые должны переходить из одной области в другую должны быть разведены на противоположных сторонах платы так, чтобы они находились в плоскости, которая не расчленяется. Если это не возможно и сигналы должны быть направлены над зазором, два слоя должны быть связаны вместе ёмкостью 0.01microF (5-вольтовая плоскость с 3.3-вольтовой) быстродействующими конденсаторами для каждых четырех сигналов, пересекающих зазор, и конденсатор располагается не далее 0.25 дюйма от точки пересечения сигналов с зазором.

Эта рекомендация не относится к медленным сигналам типа ISA-сигналов.

4.3.6.2. Полное сопротивление материнской платы

Не имеется никакой спецификации полного сопротивления для пустой материнской платы. Проектировщик системы имеет два первичных ограничения :

1. Длина и скорость сигнала должна обеспечить полное распространение его "туда и обратно" по шине внутри определенной задержки распространения равной 10 нс. (см. раздел 4.3.5.)
2. Полное сопротивление в любой точке сети должно быть таким, что PCI- устройство вывода (как определено его BAX) может удовлетворять спецификации устройства ввода данных с одиночным отражением сигнала. Это включает в себя нагрузку, обеспеченную платами расширения.

Частота функционирования может изменяться для изменения расстояния распространения "туда и обратно" по шине, чтобы формировать конфигурации, которые не могли бы выполнять два вышеуказанных ограничения. Но это всего лишь рекомендация.

4.3.7. Назначения контактов разъёма

PCI-разъём содержит все сигналы, определенные для PCI-компонента, плюс два контакта которые связаны только с разъёмом: это контакты: PRSNT1# и PRSNT2#, они описаны в Разделе 4.4.1. Материнские платы должна изолировать оба этих контакта индивидуально от "земли" с помощью высокоскоростных конденсаторов ёмкостью 0.01 microF, потому что один или оба из контактов также обеспечивают путь возврата переменного тока. Эти контакты не могут быть на шине или как-либо соединенными друг с другом на материнской плате. Далее использование этих контактов на материнской плате необязательно. Если проект материнской платы использует эти контакты, чтобы получить информацию о плате, то каждый контакт должен иметь соответствующий нагрузочный резистор (приблизительно 5 K) на материнской плате. Назначение контактов разъёма показаны в таблице 4-11.

Таблица 4-11: Контакты разъёма PCI

кон-	5-вольтовая среда		3.3-вольтовая среда		комментарий
такт	сторона B	сторона A	сторона B	сторона A	
1	-12V	TRST#	-12V	TRST#	начало 32-битового разъёма
2	TCK	+12V	TCK	+12V	
3	GROUND	TMS	GROUND	TMS	
4	TDO	TDI	TDO	TDI	
5	+5V	+5V	+5V	+5V	
6	+5V	INTA#	+5V	INTA#	
7	INTB#	INTC#	INTB#	INTC#	
8	INTD#	+5V	INTD#	+5V	
9	PRSNT1#	РЕЗЕРВ	PRSNT1#	РЕЗЕРВ	
10	РЕЗЕРВ	+5V(I:O)	РЕЗЕРВ	+3.3V(I:O)	
11	PRSNT2#	РЕЗЕРВ	PRSNT2#	РЕЗЕРВ	
12	GROUND	GROUND	КЛЮЧ РАЗЪЁМА		3.3-вольтовый ключ
13	GROUND	GROUND	КЛЮЧ РАЗЪЁМА		3.3-вольтовый ключ
14	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	
15	GROUND	RST#	GROUND	RST#	
16	CLK	+5V(I/O)	CLK	+3.3V(I/O)	
17	GROUND	GNT#	GROUND	GNT#	
18	REQ#	GROUND	REQ#	GROUND	
19	+5V(I/O)	РЕЗЕРВ	+3.3V(I/O)	РЕЗЕРВ	
20	AD[31]	AD[30]	AD[31]	AD[30]	
21	AD[29]	+3.3V	AD[29]	+3.3V	
22	GROUND	AD[28]	GROUND	AD[28]	
23	AD[27]	AD[26]	AD[27]	AD[26]	
24	AD[25]	GROUND	AD[25]	GROUND	
25	+3.3V	AD[24]	+3.3V	AD[24]	
26	C:BE[3]#	IDSEL	C:BE[3]#	IDSEL	
27	AD[23]	+3.3V	AD[23]	+3.3V	
28	GROUND	AD[22]	GROUND	AD[22]	
29	AD[21]	AD[20]	AD[21]	AD[20]	
30	AD[19]	GROUND	AD[19]	GROUND	
31	+3.3V	AD[18]	+3.3V	AD[18]	
32	AD[17]	AD[16]	AD[17]	AD[16]	
33	C:BE[2]#	+3.3V	C:BE[2]#	+3.3V	
34	GROUND	FRAME#	GROUND	FRAME#	
35	IRDY#	GROUND	IRDY#	GROUND	
36	+3.3V	TRDY#	+3.3V	TRDY#	
37	DEVSEL#	GROUND	DEVSEL#	GROUND	
38	GROUND	STOP#	GROUND	STOP#	
39	LOCK#	+3.3V	LOCK#	+3.3V	
40	PERR#	SDONE	PERR#	SDONE	
41	+3.3V	SBO#	+3.3V	SBO#	
42	SERR#	GROUND	SERR#	GROUND	

Таблица 4-11: Контакты разъёма PCI (продолжение)

кон-	5-вольтовая среда		3.3-вольтовая среда		комментарий
такт	сторона B	сторона A	сторона B	сторона A	
43	+3.3V	PAR	+3.3V	PAR	
44	C/BE[1]#	AD[15]	C/BE[1]#	AD[15]	
45	AD[14]	+3.3V	AD[14]	+3.3V	
46	GROUND	AD[13]	GROUND	AD[13]	
47	AD[12]	AD[11]	AD[12]	AD[11]	
48	AD[10]	GROUND	AD[10]	GROUND	
49	GROUND	AD[09]	GROUND	AD[09]	
50	КЛЮЧ РАЗЪЁМА		GROUND	GROUND	5-вольтовый ключ
51	КЛЮЧ РАЗЪЁМА		GROUND	GROUND	5-вольтовый ключ
52	AD[08]	C/BE[0]#	AD[08]	C/BE[0]#	
53	AD[07]	+3.3V	AD[07]	+3.3V	
54	+3.3V	AD[06]	+3.3V	AD[06]	
55	AD[05]	AD[04]	AD[05]	AD[04]	
56	AD[03]	GROUND	AD[03]	GROUND	
57	GROUND	AD[02]	GROUND	AD[02]	
58	AD[01]	AD[00]	AD[01]	AD[00]	
59	+5V(I/O)	+5V(I/O)	+3.3V(I/O)	+3.3V(I/O)	
60	ACK64#	REQ64#	ACK64#	REQ64#	
61	+5V	+5V	+5V	+5V	
62	+5V	+5V	+5V	+5V	конец 32-битового разъёма
	КЛЮЧ РАЗЪЁМА		КЛЮЧ РАЗЪЁМА		64-битовый разделитель
	КЛЮЧ РАЗЪЁМА		КЛЮЧ РАЗЪЁМА		64-битовый разделитель
63	РЕЗЕРВ	GROUND	РЕЗЕРВ	GROUND	начало 64-битового разъёма
64	GROUND	C/BE[7]#	GROUND	C/BE[7]#	
65	C/BE[6]#	C/BE[5]#	C/BE[6]#	C/BE[5]#	
66	C/BE[4]#	+5V(I/O)	C/BE[4]#	+3.3V(I/O)	
67	GROUND	PAR64	GROUND	PAR64	
68	AD[63]	AD[62]	AD[63]	AD[62]	
69	AD[61]	GROUND	AD[61]	GROUND	
70	+5V(I/O)	AD[60]	+5V(I/O)	AD[60]	
71	AD[59]	AD[58]	AD[59]	AD[58]	
72	AD[57]	GROUND	AD[57]	GROUND	
73	GROUND	AD[56]	GROUND	AD[56]	
74	AD[55]	AD[54]	AD[55]	AD[54]	
75	AD[53]	+5V(I/O)	AD[53]	+5V(I/O)	
76	GROUND	AD[52]	GROUND	AD[52]	
77	AD[51]	AD[50]	AD[51]	AD[50]	
78	AD[49]	GROUND	AD[49]	GROUND	
79	+5V(I/O)	AD[48]	+5V(I:O)	AD[48]	
80	AD[47]	AD[46]	AD[47]	AD[46]	
81	AD[45]	GROUND	AD[45]	GROUND	
82	GROUND	AD[44]	GROUND	AD[44]	

Таблица 4-11: Контакты разъёма PCI (продолжение)

кон-	5-вольтовая среда		3.3-вольтовая среда		комментарий
такт	сторона B	сторона A	сторона B	сторона A	
83	AD[43]	AD[42]	AD[43]	AD[42]	
84	AD[41]	+5V(I:O)	AD[41]	+3.3V(I:O)	
85	GROUND	AD[40]	GROUND	AD[40]	
86	AD[39]	AD[38]	AD[39]	AD[38]	
87	AD[37]	GROUND	AD[37]	GROUND	
88	+5V(I/O)	AD[36]	+3.3V(I/O)	AD[36]	
89	AD[35]	AD[34]	AD[35]	AD[34]	
90	AD[33]	GROUND	AD[33]	GROUND	
91	GROUND	AD[32]	GROUND	AD[32]	
92	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	
93	РЕЗЕРВ	GROUND	РЕЗЕРВ	GROUND	
94	GROUND	РЕЗЕРВ	GROUND	РЕЗЕРВ	конец 64-битового разъёма

Контакты, помеченные "+5V(I/O)" и "+ 3.3V(I/O)" - специальные контакты питания для управления и запуска PCI-шины передачи сигналов на универсальной плате. На материнской плате они соединены с "+5V" или "+3.3V".

Особое внимание должно быть уделено соединению REQ64# и ACK64# на материнской плате. Соответствующие нагрузочные резисторы располагаются на этой материнской плате (а не плате расширения) чтобы гарантировать, что имеется не более чем один соответствующий резистор, соединенный с любым из этих контактов. 64-разрядный разъём не всегда представлен на материнской плате, так что эти контакты размещаются на разделе 32-разрядного разъёма. Эти сигналы должны быть шинированы вместе (с одиночным нагрузочным резистором на каждом сигнале) на всех контактах, которые обеспечивают 64-разрядную шину данных. Для 32-разрядных контактов эти сигналы должны остаться открытыми. Они не должны быть соединены вместе, и каждый должен иметь собственный нагрузочный резистор так, чтобы 64-разрядные платы, подключенные в 32-разрядные слоты функционировали правильно. Для специальной информации REQ64# в течение сброса, обратитесь к разделу 4.3.2.

4.4. Спецификация платы расширения

4.4.1. Назначение контактов платы

PCI-разъём содержит все сигналы, определенные для PCI-компонента, плюс два контакта, которые связаны только с разъёмом. Это сигналы - PRSNT1# и PRSNT2#. Они используются для двух целей: индикация, что плата физически представлена в слоте и обеспечение информацией относительно общих требований питания платы. Таблица 4-12 определяет требуемую установку PRSNT# для плат расширения.

Таблица 4-12

PRSNT#	PRSNT#	Конфигурация расширения
открыт	открыт	нет плат расширения
ground	открыт	плата расширения есть,25W максимум
открыт	ground	плата расширения есть,15W максимум
ground	ground	плата расширения есть,7.5W максимум

При обеспечении индикации мощности, плата расширения должна индицировать максимальную потребляемую мощность для платы. Система должна принять, что плата расширения может выводить это питание из 5-вольтовой или из 3-вольтовой шины питания. Кроме того, если плата расширения с перестраиваемой конфигурацией (например, гнезда для расширения памяти, и т.д.) связывание контакта должно указать общую мощность, используемую полностью сконфигурированной платой, которая может быть больше чем та, что используется в пустой конфигурации.

Таблица 4-13: Контакты PCI-платы

кон-	5-вольтовая плата		универсальная плата		3.3-вольтовая плата		комментарий
такт	сторона B	сторона A	сторона B	сторона A	сторона B	сторона A	
1	-12V	TRST#	-12V	TRST#	-12V	TRST#	начало 32-битового разъёма
2	TCK	+12V	TCK	+12V	TCK	+12V	
3	GROUND	TMS	GROUND	TMS	GROUND	TMS	
4	TDO	TDI	TDO	TDI	TDO	TDI	
5	+5V	+5V	+5V	+5V	+5V	+5V	
6	+5V	INTA#	+5V	INTA#	+5V	INTA#	
7	INTB#	INTC#	INTB#	INTC#	INTB#	INTC#	
8	INTD#	+5V	INTD#	+5V	INTD#	+5V	
9	PRSNT1#	РЕЗЕРВ	PRSNT1#	РЕЗЕРВ	PRSNT1#	РЕЗЕРВ	
10	РЕЗЕРВ	+5V	РЕЗЕРВ	+Vio	РЕЗЕРВ	+3.3V	
11	PRSNT2#	РЕЗЕРВ	PRSNT2#	РЕЗЕРВ	PRSNT2#	РЕЗЕРВ	
12	GROUND	GROUND	КЛЮЧ РАЗЪЁМА		КЛЮЧ РАЗЪЁМА		3.3-вольтовый ключ
13	GROUND	GROUND	КЛЮЧ РАЗЪЁМА		КЛЮЧ РАЗЪЁМА		3.3-вольтовый ключ
14	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	
15	GROUND	RST#	GROUND	RST#	GROUND	RST#	
16	CLK	+5V	CLK	+Vio	CLK	+3.3V	
17	GROUND	GNT#	GROUND	GNT#	GROUND	GNT#	
18	REQ#	GROUND	REQ#	GROUND	REQ#	GROUND	
19	+5V	РЕЗЕРВ	+Vio	РЕЗЕРВ	+3.3V	РЕЗЕРВ	
20	AD[31]	AD[30]	AD[31]	AD[30]	AD[31]	AD[30]	
21	AD[29]	+3.3V	AD[29]	+3.3V	AD[29]	+3.3V	
22	GROUND	AD[28]	GROUND	AD[28]	GROUND	AD[28]	
23	AD[27]	AD[26]	AD[27]	AD[26]	AD[27]	AD[26]	
24	AD[25]	GROUND	AD[25]	GROUND	AD[25]	GROUND	
25	+3.3V	AD[24]	+3.3V	AD[24]	+3.3V	AD[24]	
26	C:BE[3]#	IDSEL	C:BE[3]#	IDSEL	C:BE[3]#	IDSEL	
27	AD[23]	+3.3V	AD[23]	+3.3V	AD[23]	+3.3V	
28	GROUND	AD[22]	GROUND	AD[22]	GROUND	AD[22]	
29	AD[21]	AD[20]	AD[21]	AD[20]	AD[21]	AD[20]	
30	AD[19]	GROUND	AD[19]	GROUND	AD[19]	GROUND	
31	+3.3V	AD[18]	+3.3V	AD[18]	+3.3V	AD[18]	
32	AD[17]	AD[16]	AD[17]	AD[16]	AD[17]	AD[16]	
33	C:BE[2]#	+3.3V	C:BE[2]#	+3.3V	C:BE[2]#	+3.3V	
34	GROUND	FRAME#	GROUND	FRAME#	GROUND	FRAME#	
35	IRDY#	GROUND	IRDY#	GROUND	IRDY#	GROUND	
36	+3.3V	TRDY#	+3.3V	TRDY#	+3.3V	TRDY#	
37	DEVSEL#	GROUND	DEVSEL#	GROUND	DEVSEL#	GROUND	
38	GROUND	STOP#	GROUND	STOP#	GROUND	STOP#	
39	LOCK#	+3.3V	LOCK#	+3.3V	LOCK#	+3.3V	
40	PERR#	SDONE	PERR#	SDONE	PERR#	SDONE	
41	+3.3V	SBO#	+3.3V	SBO#	+3.3V	SBO#	
42	SERR#	GROUND	SERR#	GROUND	SERR#	GROUND	

Таблица 4-13: Контакты PCI- платы (продолжение)

кон-	5-вольтовая среда		универсальная плата		3.3-вольтовая среда		коммен-
такт	сторона B	сторона A	сторона B	сторона A	сторона B	сторона A	тарий
43	+3.3V	PAR	+3.3V	PAR	+3.3V	PAR	
44	C:BE[1]#	AD[15]	C:BE[1]#	AD[15]	C:BE[1]#	AD[15]	
45	AD[14]	+3.3V	AD[14]	+3.3V	AD[14]	+3.3V	
46	GROUND	AD[13]	GROUND	AD[13]	GROUND	AD[13]	
47	AD[12]	AD[11]	AD[12]	AD[11]	AD[12]	AD[11]	
48	AD[10]	GROUND	AD[10]	GROUND	AD[10]	GROUND	
49	GROUND	AD[09]	GROUND	AD[09]	GROUND	AD[09]	
50	КЛЮЧ РАЗЪЁМА		КЛЮЧ РАЗЪЁМА		GROUND	GROUND	5-вольтовый ключ
51	КЛЮЧ РАЗЪЁМА		КЛЮЧ РАЗЪЁМА		GROUND	GROUND	5-вольтовый ключ
52	AD[08]	C:BE[0]#	AD[08]	C:BE[0]#	AD[08]	C:BE[0]#	
53	AD[07]	+3.3V	AD[07]	+3.3V	AD[07]	+3.3V	
54	+3.3V	AD[06]	+3.3V	AD[06]	+3.3V	AD[06]	
55	AD[05]	AD[04]	AD[05]	AD[04]	AD[05]	AD[04]	
56	AD[03]	GROUND	AD[03]	GROUND	AD[03]	GROUND	
57	GROUND	AD[02]	GROUND	AD[02]	GROUND	AD[02]	
58	AD[01]	AD[00]	AD[01]	AD[00]	AD[01]	AD[00]	
59	+5V	+5V	+Vio	+Vio	+3.3V	+3.3V	
60	ACK64#	REQ64#	ACK64#	REQ64#	ACK64#	REQ64#	
61	+5V	+5V	+5V	+5V	+5V	+5V	конец 32-битового разъёма
62	+5V	+5V	+5V	+5V	+5V	+5V	
	КЛЮЧ РАЗЪЁМА		КЛЮЧ РАЗЪЁМА		КЛЮЧ РАЗЪЁМА		64-битовый раздел.
	КЛЮЧ РАЗЪЁМА		КЛЮЧ РАЗЪЁМА		КЛЮЧ РАЗЪЁМА		64-битовый раздел.
63	РЕЗЕРВ	GROUND	РЕЗЕРВ	GROUND	РЕЗЕРВ	GROUND	начало 64-битового разъёма
64	GROUND	C:BE[7]#	GROUND	C:BE[7]#	GROUND	C:BE[7]#	
65	C:BE[6]#	C:BE[5]#	C:BE[6]#	C:BE[5]#	C:BE[6]#	C:BE[5]#	
66	C:BE[4]#	+5V	C:BE[4]#	+Vio	C:BE[4]#	+3.3V	
67	GROUND	PAR64	GROUND	PAR64	GROUND	PAR64	
68	AD[63]	AD[62]	AD[63]	AD[62]	AD[63]	AD[62]	
69	AD[61]	GROUND	AD[61]	GROUND	AD[61]	GROUND	
70	+5V	AD[60]	+Vio	AD[60]	+3.3V	AD[60]	
71	AD[59]	AD[58]	AD[59]	AD[58]	AD[59]	AD[58]	
72	AD[57]	GROUND	AD[57]	GROUND	AD[57]	GROUND	
73	GROUND	AD[56]	GROUND	AD[56]	GROUND	AD[56]	
74	AD[55]	AD[54]	AD[55]	AD[54]	AD[55]	AD[54]	
75	AD[53]	+5V	AD[53]	+Vio	AD[53]	+3.3V	
76	GROUND	AD[52]	GROUND	AD[52]	GROUND	AD[52]	
77	AD[51]	AD[50]	AD[51]	AD[50]	AD[51]	AD[50]	
78	AD[49]	GROUND	AD[49]	GROUND	AD[49]	GROUND	
79	+5V	AD[48]	+Vio	AD[48]	+3.3V	AD[48]	
80	AD[47]	AD[46]	AD[47]	AD[46]	AD[47]	AD[46]	
81	AD[45]	GROUND	AD[45]	GROUND	AD[45]	GROUND	
82	GROUND	AD[44]	GROUND	AD[44]	GROUND	AD[44]	

Таблица 4-13: Контакты PCI-платы (продолжение)

кон-	5-вольтовая среда		универсальная плата		3.3-вольтовая среда		коммен-
такт	сторона B	сторона A	сторона B	сторона A	сторона B	сторона A	тарий
83	AD[43]	AD[42]	AD[43]	AD[42]	AD[43]	AD[42]	
84	AD[41]	+5V	AD[41]	+Vio	AD[41]	+3.3V	
85	GROUND	AD[40]	GROUND	AD[40]	GROUND	AD[40]	
86	AD[39]	AD[38]	AD[39]	AD[38]	AD[39]	AD[38]	
87	AD[37]	GROUND	AD[37]	GROUND	AD[37]	GROUND	
88	+5V	AD[36]	+Vio	AD[36]	+3.3V	AD[36]	
89	AD[35]	AD[34]	AD[35]	AD[34]	AD[35]	AD[34]	
90	AD[33]	GROUND	AD[33]	GROUND	AD[33]	GROUND	
91	GROUND	AD[32]	GROUND	AD[32]	GROUND	AD[32]	
92	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	РЕЗЕРВ	
93	РЕЗЕРВ	GROUND	РЕЗЕРВ	GROUND	РЕЗЕРВ	GROUND	конец 64-битового разъёма
94	GROUND	РЕЗЕРВ	GROUND	РЕЗЕРВ	GROUND	РЕЗЕРВ	

Таблица 4-14 : контакты 32-разрядной платы

Тип контакта	5V плата	универсальная плата	3V плата
Ground	22	18	22
+5V	13	8	8
+3.3V	12	12	17
I/Opwr	0	5	0
резерв	6	6	6

Таблица 4-15: Дополнительные контакты на 64-разрядной плате

Тип контакта	5V плата	универсальная плата	3V плата
Ground	16	16	16
+5V	6	0	0
+3.3V	0	0	6
I/Opwr	0	6	0
резерв	5	5	5

Контакты помеченные “$+V_{io}$”- это специальные контакты питания для определения и управления сигналов PCI на универсальной плате. На этой плате, буферы ввода/вывода PCI-компонентов должны запитываться только от этих специальных контактов, а не от контактов +5V или +3.3V.

4.4.2. Требования питания

4.4.2.1. Изоляция

При типичных условиях, ёмкость между питанием и землёй будет обеспечивать хорошую изоляцию для контактов питания разъёма. Максимальная длина дорожки от разъёма до Vcc/GND слоя должна быть 0.25 дюймов.
Однако, на универсальной плате, вероятно шина питания буфера ввода - вывода не будет иметь адекватной ёмкости по отношению к земле, чтобы обеспечить необходимую изоляцию. Контакты, помеченные "+ Vio" должны быть изолированы от "земли" в среднем с ёмкостью 0.047 microF на контакт.
Дополнительно, все +3.3-вольтовые и любые неиспользуемые +5-вольтовые контакты должны быть изолированы от земли как описано ниже, чтобы гарантировать, что они продолжают функционировать как эффективные контрольные точки переменного тока:

1. Изолирование должно составить в среднем по крайней мере 0.01 microF на контакт питания.
2. Длина дорожки от контакта до конденсатора должна быть не больше 0.25 дюйма при использовании ширины дорожки по крайней мере 0.02 дюйма. Не имеется никакого ограничения относительно числа контактов, которые могут совместно использовать один и тот же конденсатор, если вышеуказанные 2 требования выполняются.

4.4.2.2. Потребляемая мощность

Максимальная мощность, позволенная для любой PCI-платы -- 25 Вт, и представляет общую мощность, выведенную от всех четырех шин питания, обеспечиваемых в разъёме. В самом плохом случае, все 25Вт могли бы быть выведены из + 5V- или из + 3V-шины.
Ожидается, что много систем не будут обеспечивать полные 25 ватт на разъём питания для каждой шины питания, потому что большинство плат обычно потребляет намного меньше чем это количество. По этой причине, PCI платы, которые потребляют больше чем l0 ватт должны включаться и сбрасываться к питанию, сохраняющему состояние, которое потребляет l0 Вт или меньше, если это возможно. Пока это состояние имеется, плата должна обеспечить полный доступ к PCI-пространству конфигурации, и должна выполнить требуемые функции начальной загрузки, типа базисного текстового режима на плате видеосигнала. Все другие функции платы могут быть приостановлены, если необходимо. Это питание, сохраняющее состояние может быть достигнуто различными способами. Например:

1. Тактовые частоты на плате могут быть уменьшены, что в принципе уменьшает эффективность, но не ограничивают функциональные возможности
2. Шины питания некритических частей могли бы быть закрыты от с FET, который мог бы ограничить функциональные возможности

После того, как драйвер для платы был инициализирован, плату можно довести до полностью включенного состояния, полное состояние эффективности, используя устройство - зависимый механизм выбора. В системах с продвинутым управлением питанием, драйвер устройства может требоваться, чтобы сообщить итоговую потребляемую мощность перед осуществлением полного доступа к плате, чтобы позволить системе определить, имеет ли она достаточный резерв мощности для всех плат в текущей конфигурации. Драйвер должен быть способен точно определить максимальные требования питания для платы, и из которых шин(ы) это питание будет подаваться.

4.4.3. Конструктивные требования

4.4.3.1. Ограничения длины дорожки

Длина дорожки от разъёма на краю платы до PCI-устройства следующая:
1. Максимальные длины дорожек для всех сигналов 32-разрядного интерфейса ограничены 1.5 дюймами для 64-битовых плат и 32-разрядных плат.
2. Длины дорожек дополнительных сигналов, используемых для 64-разрядного расширения ограничены до 2-х дюймов на всех 64-разрядных платах.
3. Длина дорожки для CLK сигнала равна 2.5 дюйма плюс/минус 0.1 дюйма для 64- разрядных плат и 32-разрядных плат и они должны быть направлены только к одной нагрузке.

4.4.3.2 Маршрутизация

Контакты питания размещаются на разъёме, чтобы облегчить размещение на четырехслойных платах. "зазор питания" может использоваться, как описано в Разделе 4.3.6.l, хотя это - стандартная методика, направление высокоскоростных сигналов непосредственно над этим зазором может вызывать проблемы целостности сигнала. Зазор в плоскости прерывает путь возврата переменного тока для сигнала, создавая неоднородность полного сопротивления.

Рекомендуемое решение состоит в том, чтобы упорядочивать размещение сигнальных уровней так, чтобы никакой высокоскоростной сигнал (например 33 MHz) не был привязан к обоим слоям. Сигнальные дорожки должны также остаться полностью над слоем 3.3V или полностью над слоем 5V. Сигнал, который должен перейти от одной области к другой должен быть разведён на противоположных сторонах платы так, чтобы они были отнесены к слою "земли" которая не расчленяется. Если это не возможно, и сигналы должны быть направлены над зазором, два слоя должны быть связаны вместе посредством ёмкости (5V слой непосредственно с 3.3-вольтовым слоем) в 0.01 microF быстродействующими конденсаторами для каждых четырех сигналов, пересекающих зазор, и конденсатор помещается не далее, чем на 0.15 дюймов от точки пересечения сигналами зазора.

4.4.3.3 Полное сопротивление (импеданс)

Характеристический импеданс (Z0) разделяемых дорожек сигнала на плате расширения должен контролироваться, чтобы быть в диапазоне от 60 до 100 Ом. Скорость прохождения сигнала по дорожке должна быть между 150 и 190 ps/inch (пикосекунд на дюйм).

4.4.3.4. Нагрузка сигнала

Разделяемые сигналы PCI должны быть ограничены одной нагрузкой на плате расширения. Нарушение длины дорожки на плате расширения или невыполнение ограничений ставит под угрозу целостность сигнала системы. Это специфично для плат расширения:

- Подключайте ROM(ПЗУ) расширения непосредственно (или через приемопередатчики шины) на любые PCI-контакты.
- Подключите два или более PCI-устройств на плату расширения, если они не поместились позади интерфейса PCI-PCI.
- Подключайте любую логику (другую, нежели одиночное PCI-устройство) - это позволит исследовать PCI-контакты.
- Используйте PCI-компонент, устанавливающий более одной нагрузка на каждом PCI-контакте; например, разделите компоненты шины адреса и шины данных.
- Используйте PCI-компонент, который имеет ёмкость большую чем 10 pF на контакт.
- Подключайте любые нагрузочные резисторы или другие дискретные устройства к сигналам PCI, если они не помещены позади моста PCI-PCI.

Глава 5
Конструктивная спецификация

5.1. Краткий обзор

PCI плата расширения основана на заготовке печатной платы (см. рисунки 5-1 и 5-2), которая реализована очень просто в существующих разработках многочисленных изготовителей. Разработанная плата адаптирована к ISA. EISA и MC - системам. PCI платы разложения имеют два базисных коэффициента формы: стандартной длины и короткой длины. Стандартная длина платы обеспечивает площадь 49 квадратных дюймов. Короткая длина была выбрана для панельной оптимизации, чтобы обеспечить самую низкую стоимость. Короткая плата также обеспечивает самую низкую стоимость при разработке системы с самым низким уровнем энергопотребления, и позволяет проектировать небольшие системы. Разъем для платы расширения предусматривает как 64-разрядные интерфейсы, так и 32 - разрядные.

PCI платы и разъемы имеют ключи для осуществления перехода 5В к 3.3В. Базисный 32-разрядный разъем содержит 120 контактов. Логическая нумерация контактов показывает 124 контактных опознавательных числа, но четыре контакта не представлены и заменены расположением самого ключа. При одной ориентации, ключ на разъем расположен так, чтобы можно было вставлять 5В - платы при передаче сигналов системы; если ключ повернут на 180°, то можно вставлять платы с сигнальным напряжением 3.3В. Универсальные вставляемые платы предусматривают работу с 5В и 3.3В и имеют два паза - ключа, поэтому их можно вставлять в любой разъем. При наличии расширения до 64 разрядов, на том же самом разъеме, увеличивает общее число контактов до 184. 32-разрядный разъем подразумевает соответствующее напряжение для передачи сигналов, 32-разрядные платы и 64-разрядные платы взаимодействуют внутри классов напряжений передачи сигналов в системе, определяемых положением ключа в 32-разрядном разъеме. 32-разрядная плата идентифицирует себя для 32-разрядной передачи на 64-разрядном разъеме. 64-разрядная плата в 32 - разрядном разъеме может быть сконфигурирована для 32 - разрядных пересылок.

Максимальное рассеяние мощности платы кодируется на контактах PRSNT1# и PRSNT2# платы расширения. Такое аппаратное кодирование может считываться программным обеспечением системы после инициализации. Программное обеспечение системы может затем определять, является ли адекватным охлаждение и ток питания в системе для осуществления надежной работы во время инициализации и запуска. Обеспечиваемые уровни питания и их кодирование описываются в главе 4, "Электрическая спецификация".

PCI плата расширения включает в себя кронштейн для установки платы и ее закрепления. Объединительная плата - интерфейс между платой и системой. Объединительная плата была разработана с возможностью установки в нее плат расширения в ISA/EISA и MC - системах.

См. рисунки 5-4 и 5-5 для ISA/EISA - трансляций и рисунки 5-6 и 5-7 - для MC трансляций. На плате PCI должны быть кронштейны для каждого типа, чтобы конечные пользователи могли сконфигурировать плату для своей системы. ISA/EISA комплект содержит PCI кронштейн , от 4 до 40 шурупов и расширитель платы. Общая длина PCI платы расширения - как такой же MC - платы. Расширитель закреплен на передней части платы PCI, чтобы обеспечить поддержку стандарта ISA для направляющей части платы. MC - комплект содержит кронштейн MC, от 4 до 40 шурупов и фигурный кронштейн.
Составляющая сторона PCI платы расширения противоположна платам ISA/EISA и MC. PCI плата является зеркальным изображением плат ISA/EISA и MC. Целью PCI является реализация такого исполнения плат в системах с ограниченном числом слотов для этих плат. В этих системах PCI разъем платы расширения может сосуществовать внутри одного слота с ISA/EISA или MC разъемом расширения. Эти слоты называются общедоступными. Общедоступные слоты позволяют конечному пользователю устанавливать PCI, ISA/EISA или MC - плату. Однако, в общедоступный разъем одновременно может быть установлена только одна плата расширения. Например, общедоступные слоты в PCI системах с ISA шиной расширения могут размещать ISA или PCI плату расширения: общедоступные пазы в PCI - системах с MC - шиной расширения могут размещать MC- или PCI - плату расширения и т.д.

5.2. Физические размеры и допуски платы расширения

Максимальная высота верхней стороны PCI платы расширения не должна превышать 0.570 дюймов (14.48 мм). Максимальная высота задней стороны платы не должна превышать 0.105 дюймов (2.67 мм). На рисунке используется положение A, чтобы разместить PCI - плату горизонтально к интерфейсам выводной рамки, обратно от рамки и направляющей платы. Положение A реализовано через ключ на ребре платы и ключ на соединителе.

См. на рисунках с 5-1 по 5-10 физические размеры PCI - платы расширения.

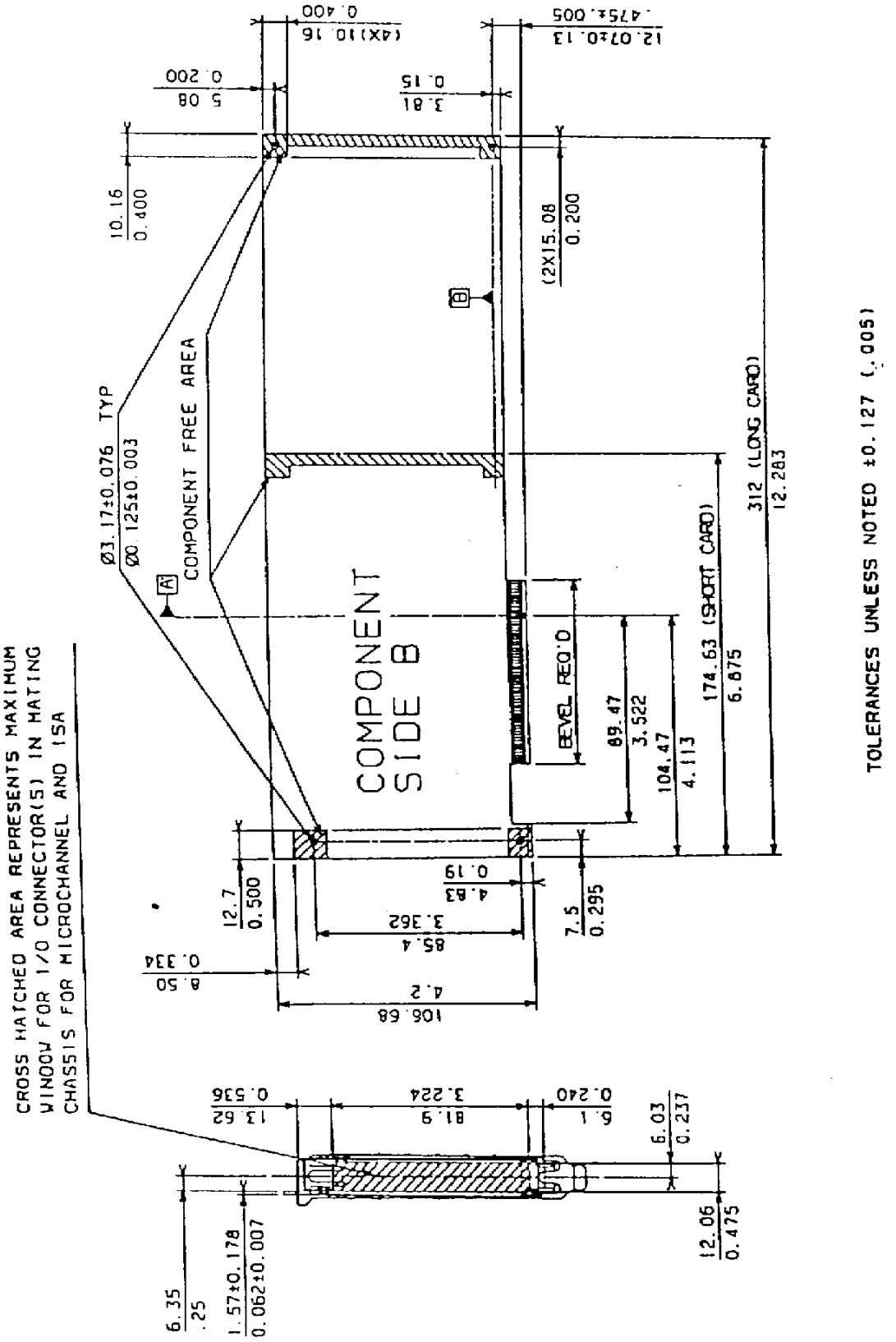


Рисунок 5-1: Заготовка PCI платы (5B)

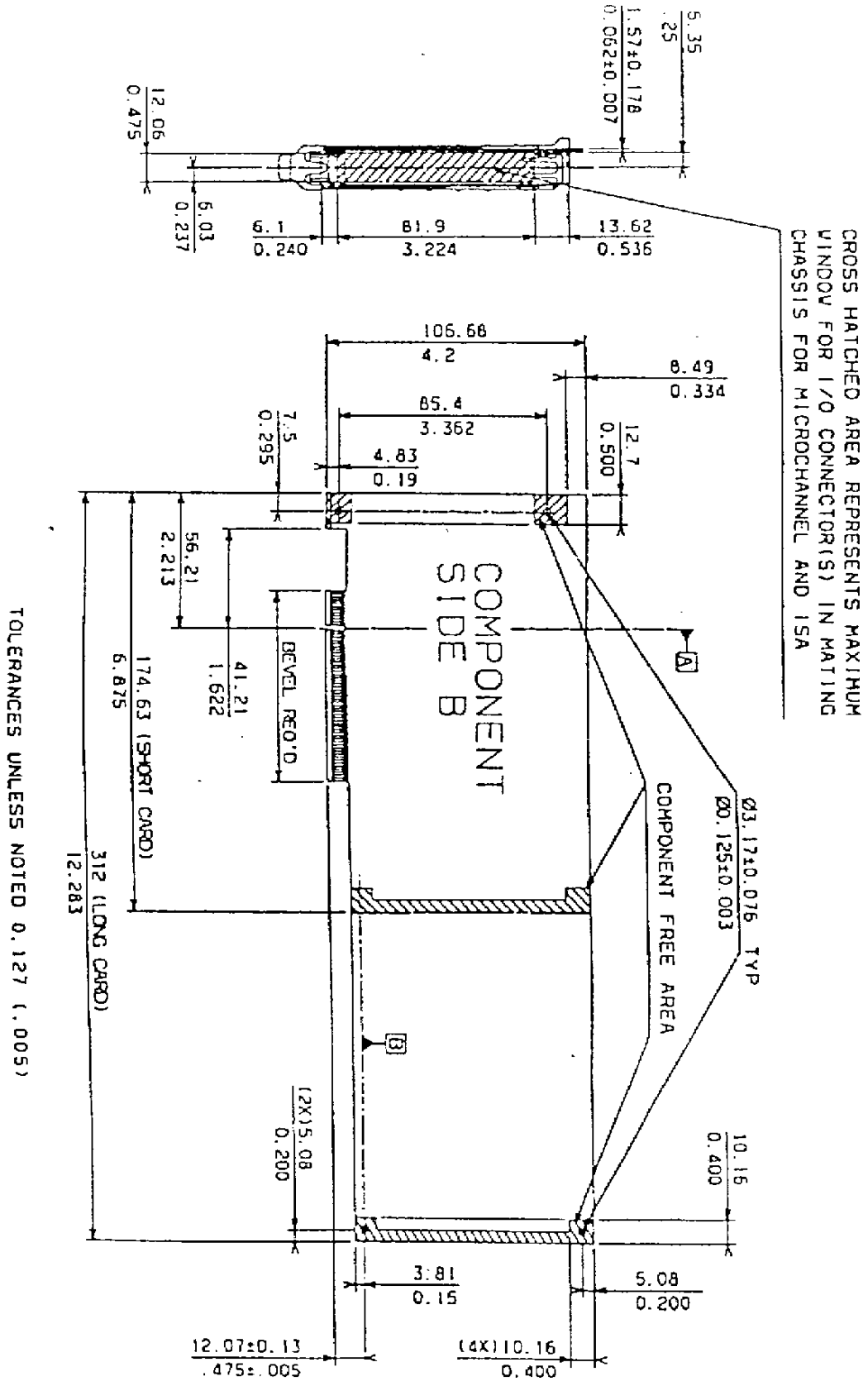


Рисунок 5-2: Заготовка PCI - платы (3.3В)

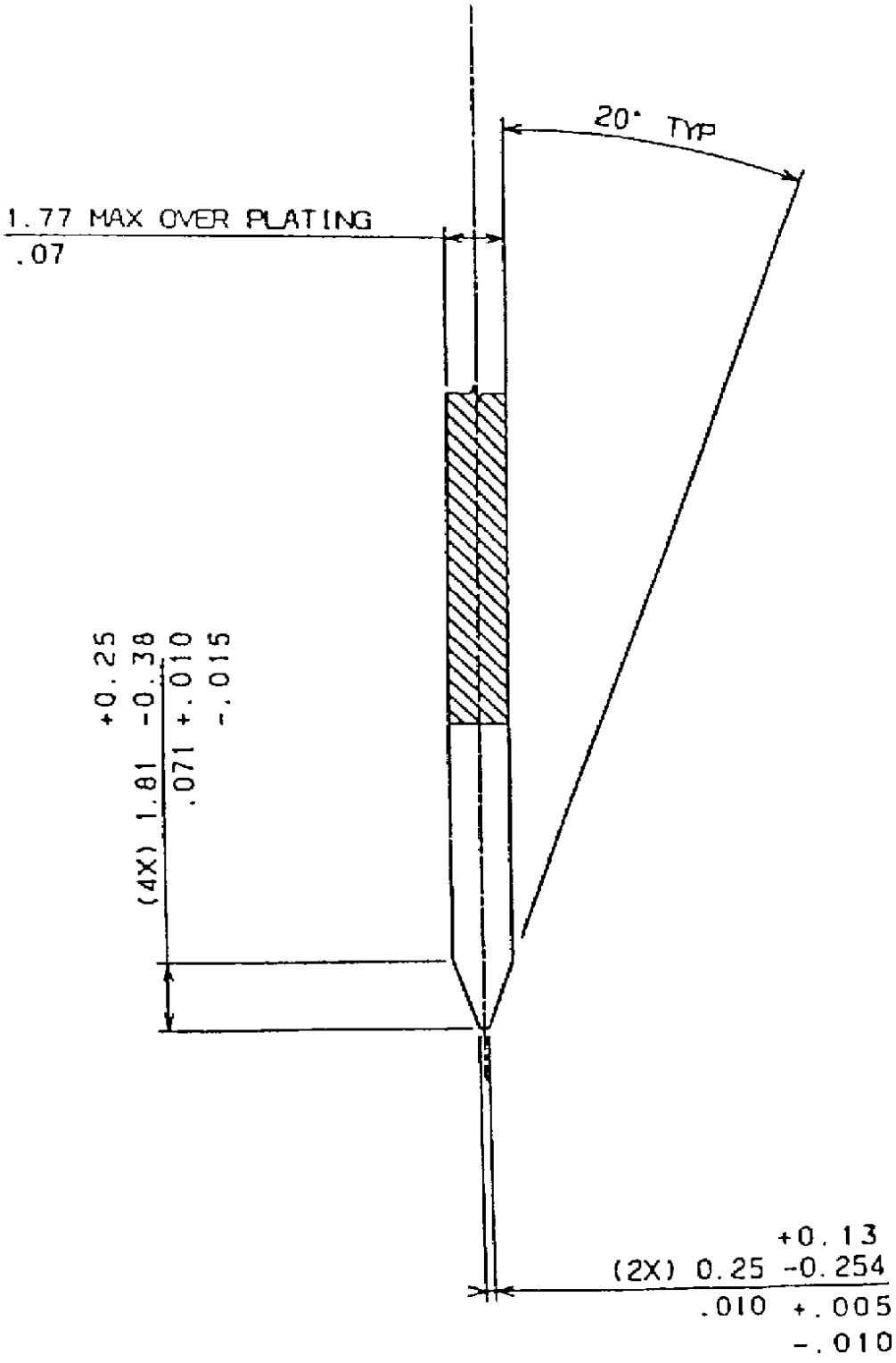


Рисунок 5-3: Ребро разъема на PCI - плате

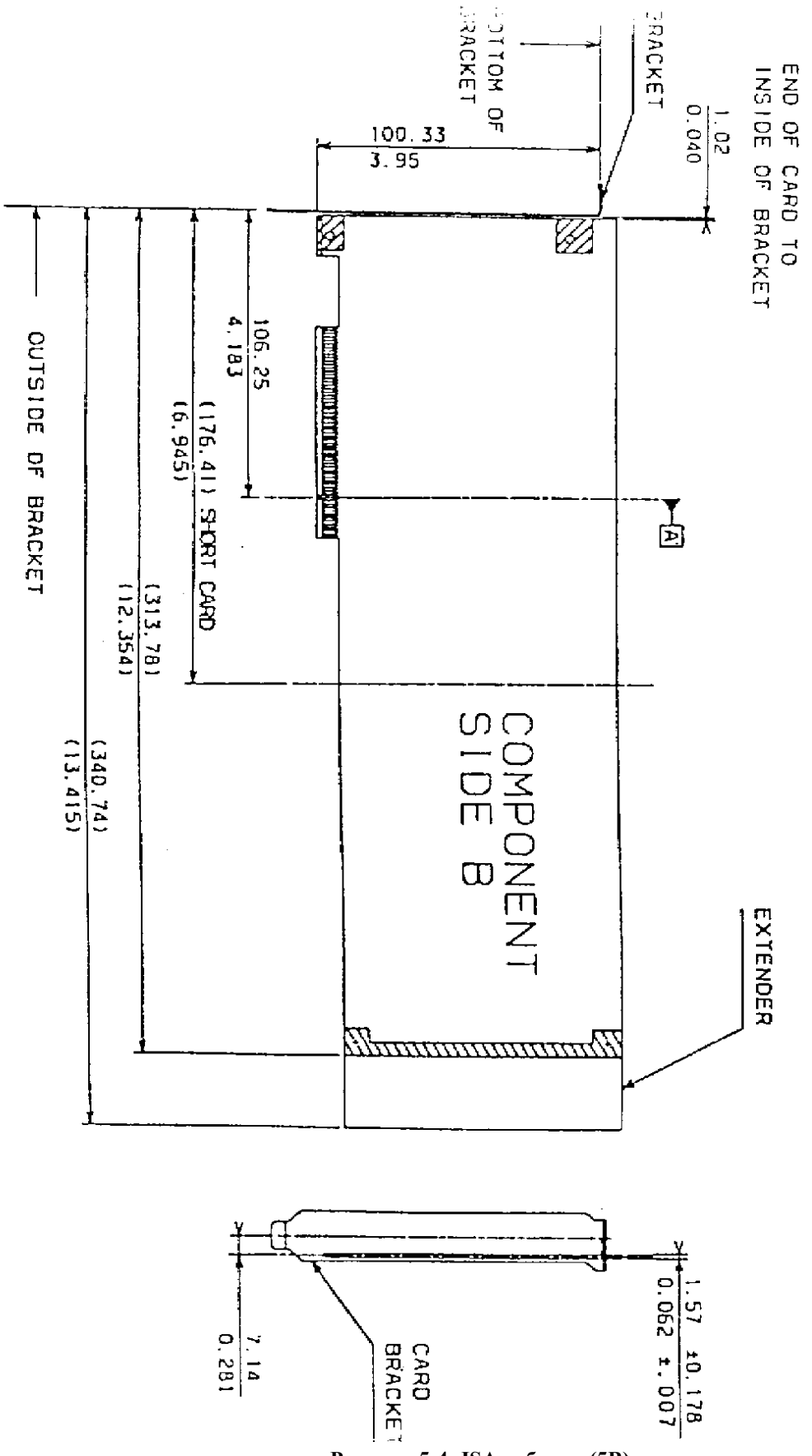


Рисунок 5-4: ISA - сборка (5B)

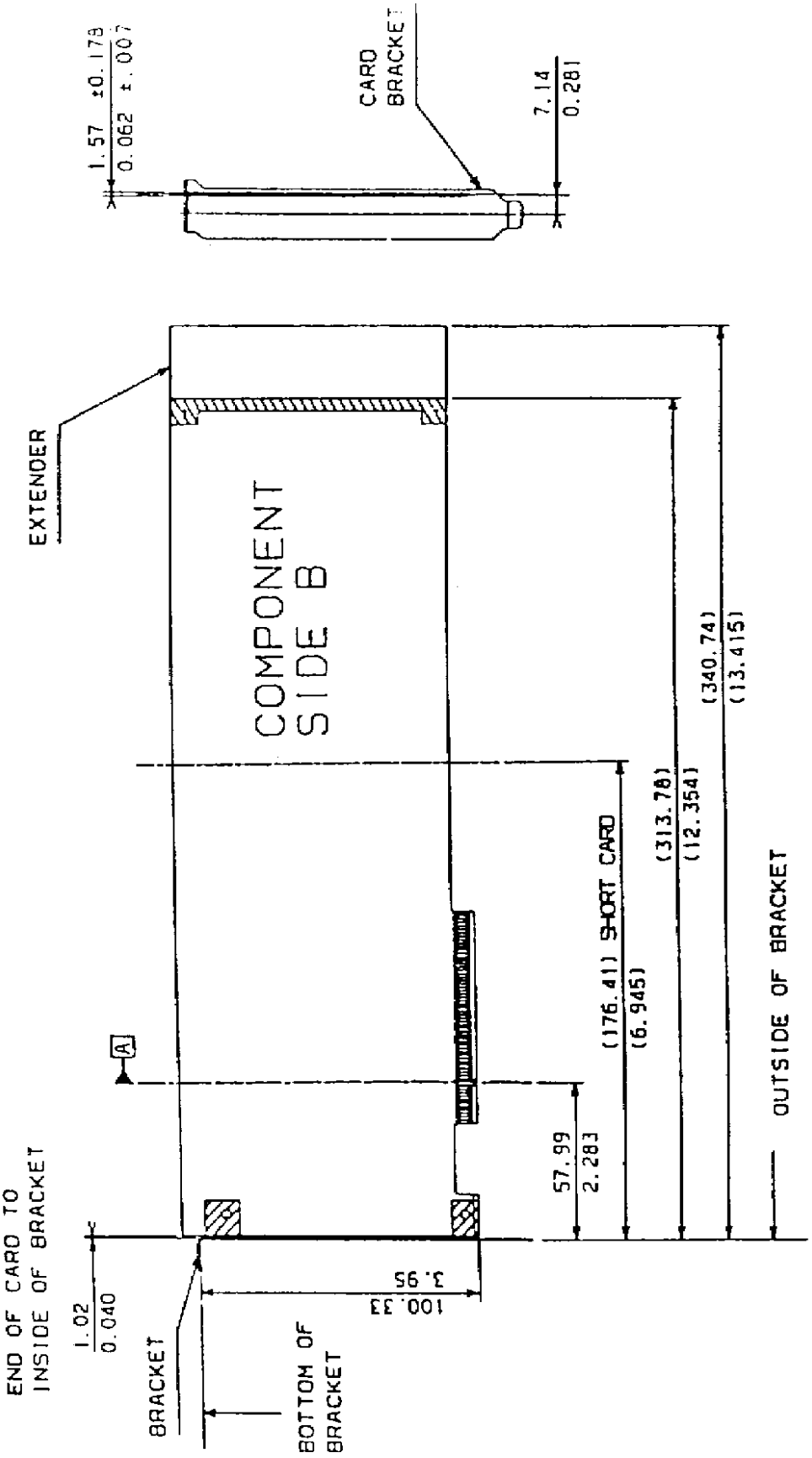


Рисунок 5-5: ISA - Сборка (5B)

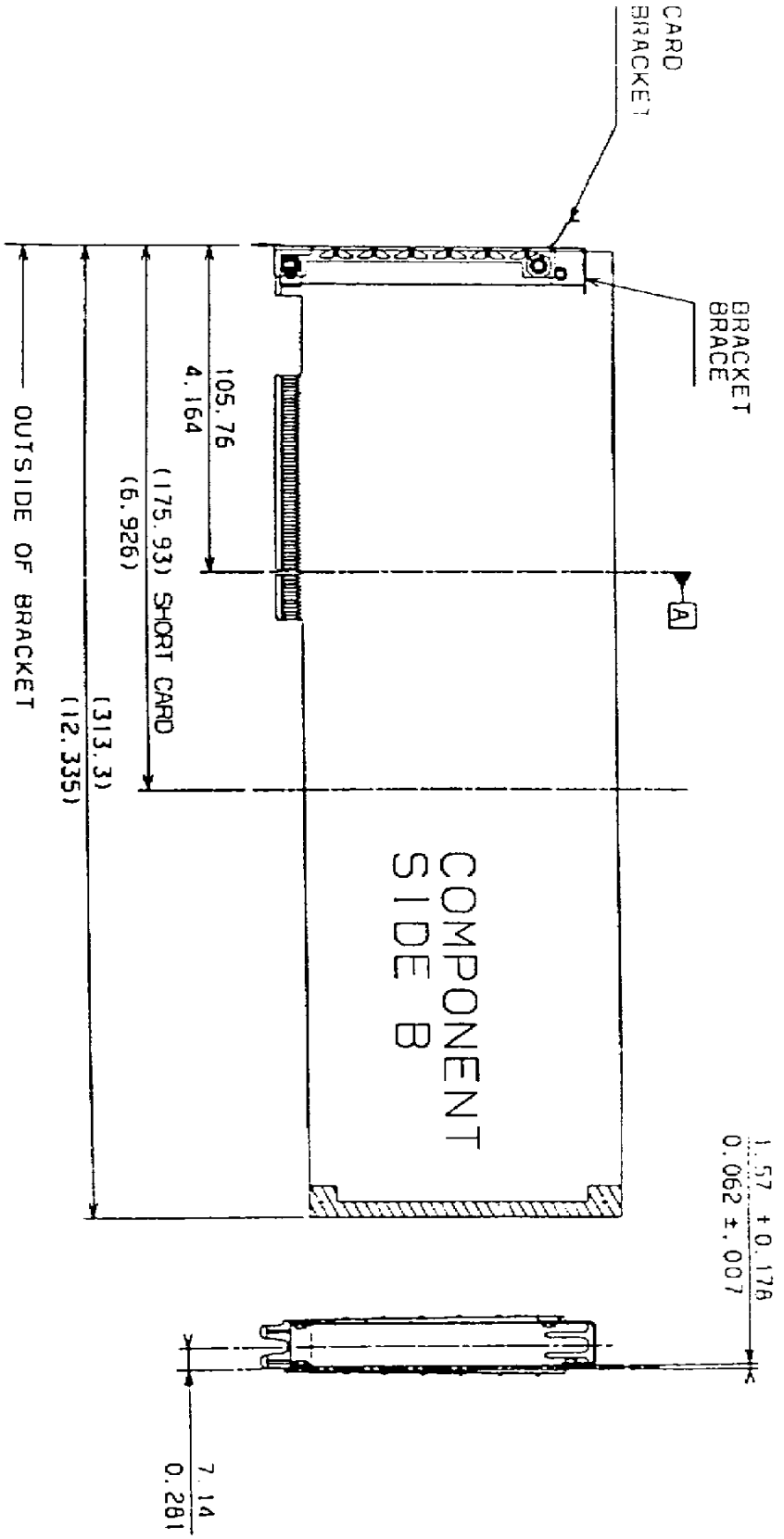


Рисунок 5-6: MC - сборка (5B)

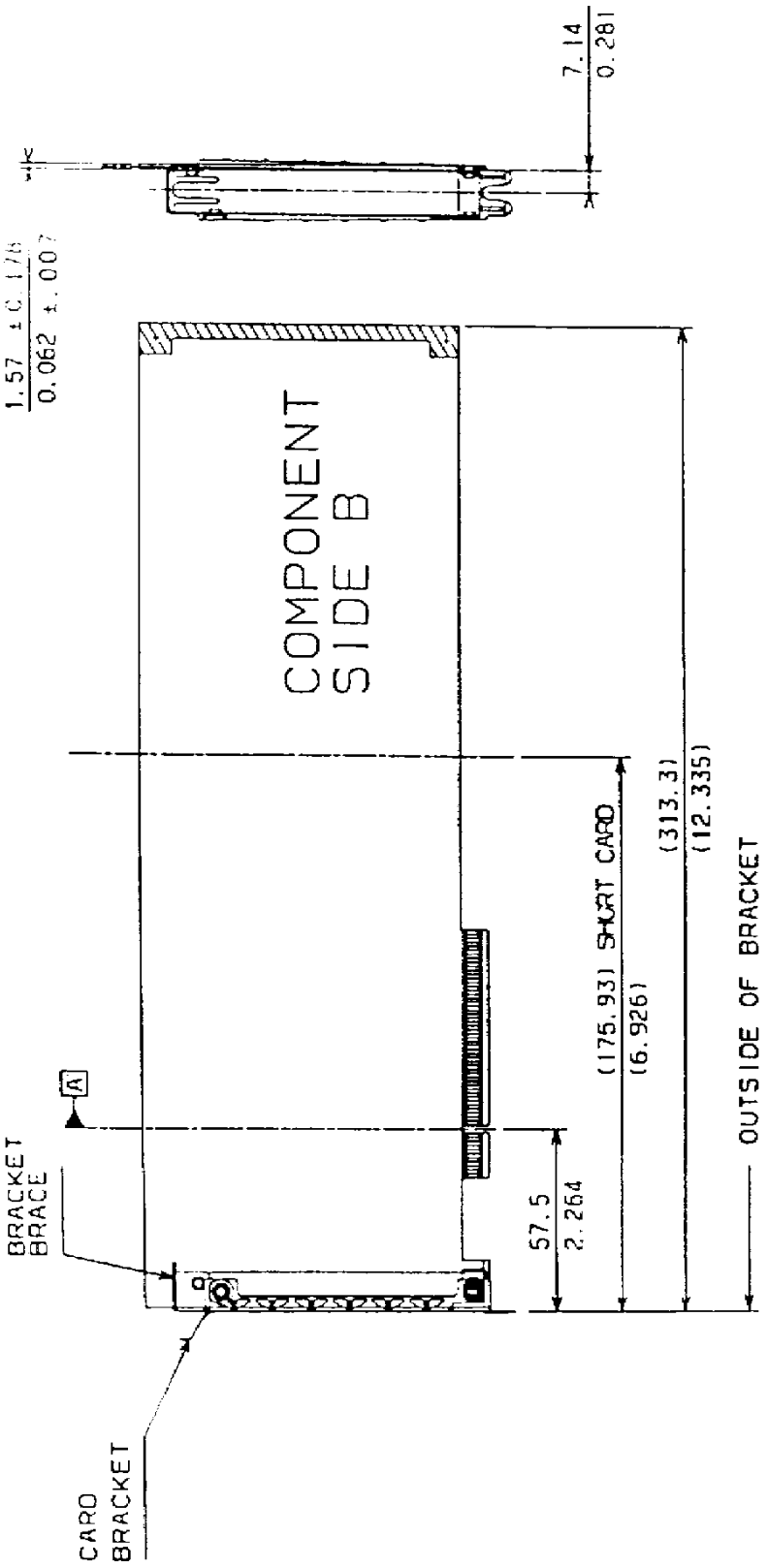


Рисунок 5-7: MC - сборка (3.3В)

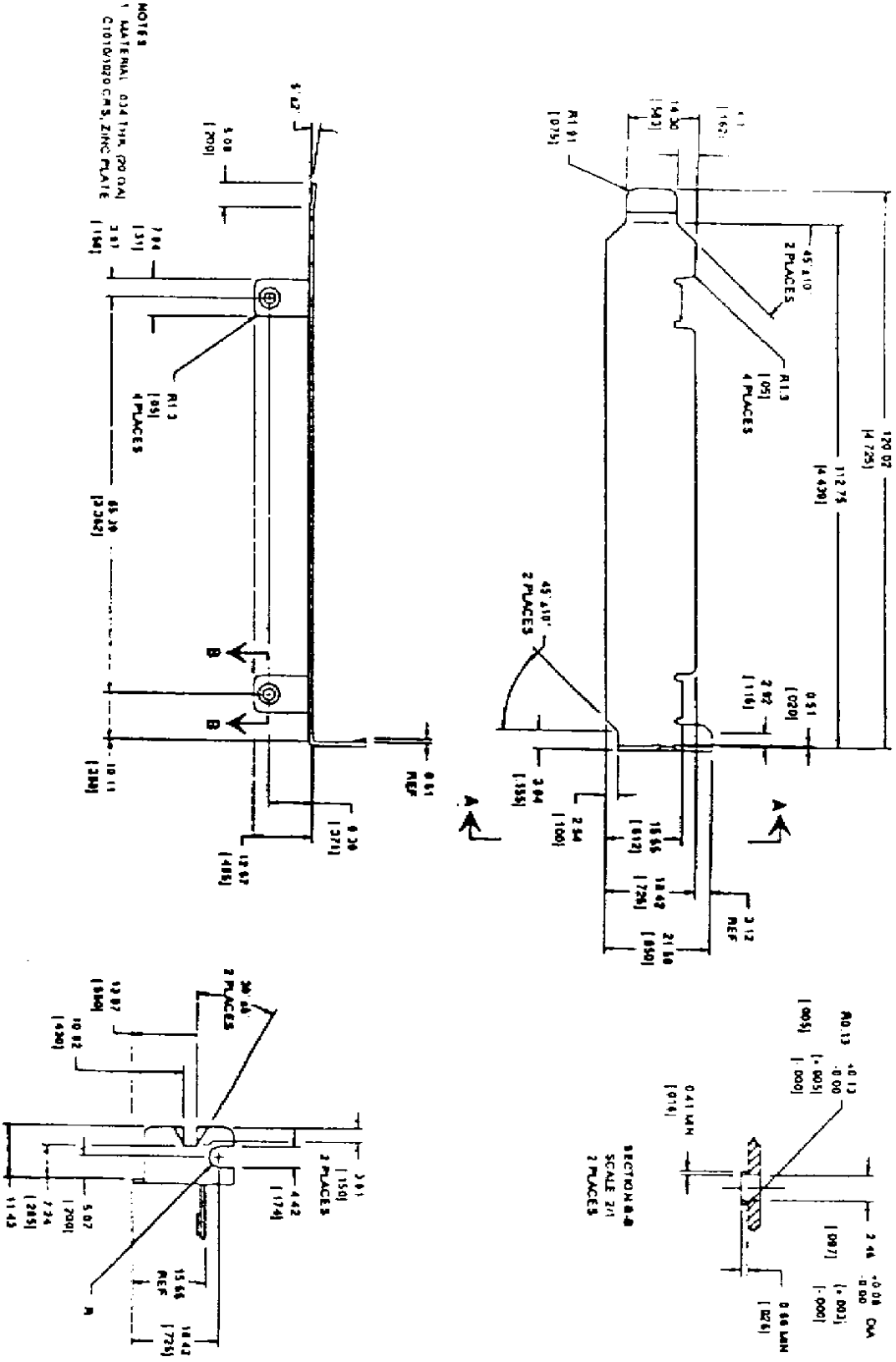


Рисунок 5-8: Кронштейн ISA

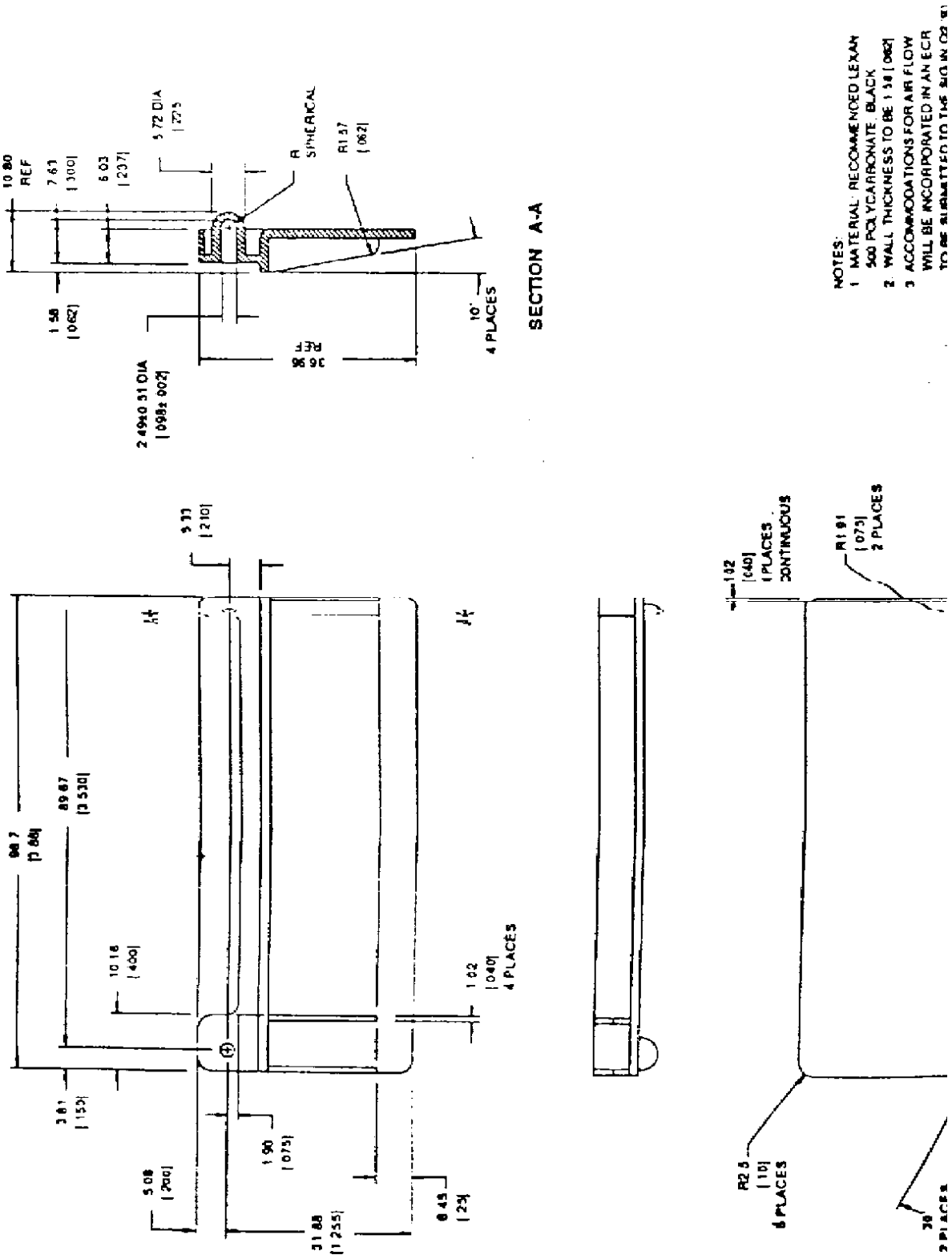


Рисунок 5-9: Крепежный элемент ISA

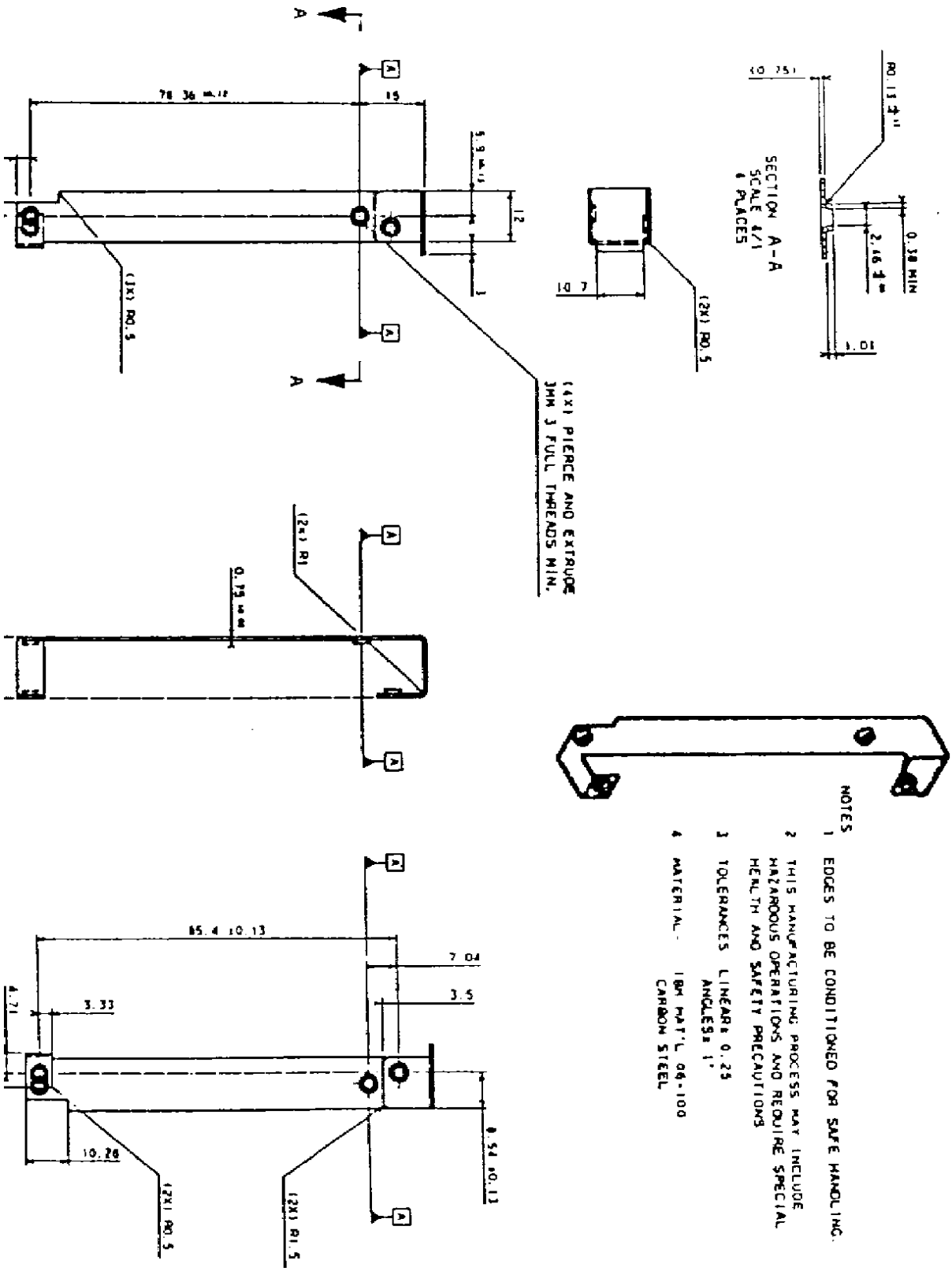


Рисунок 5-10: Распорка кронштейна MC

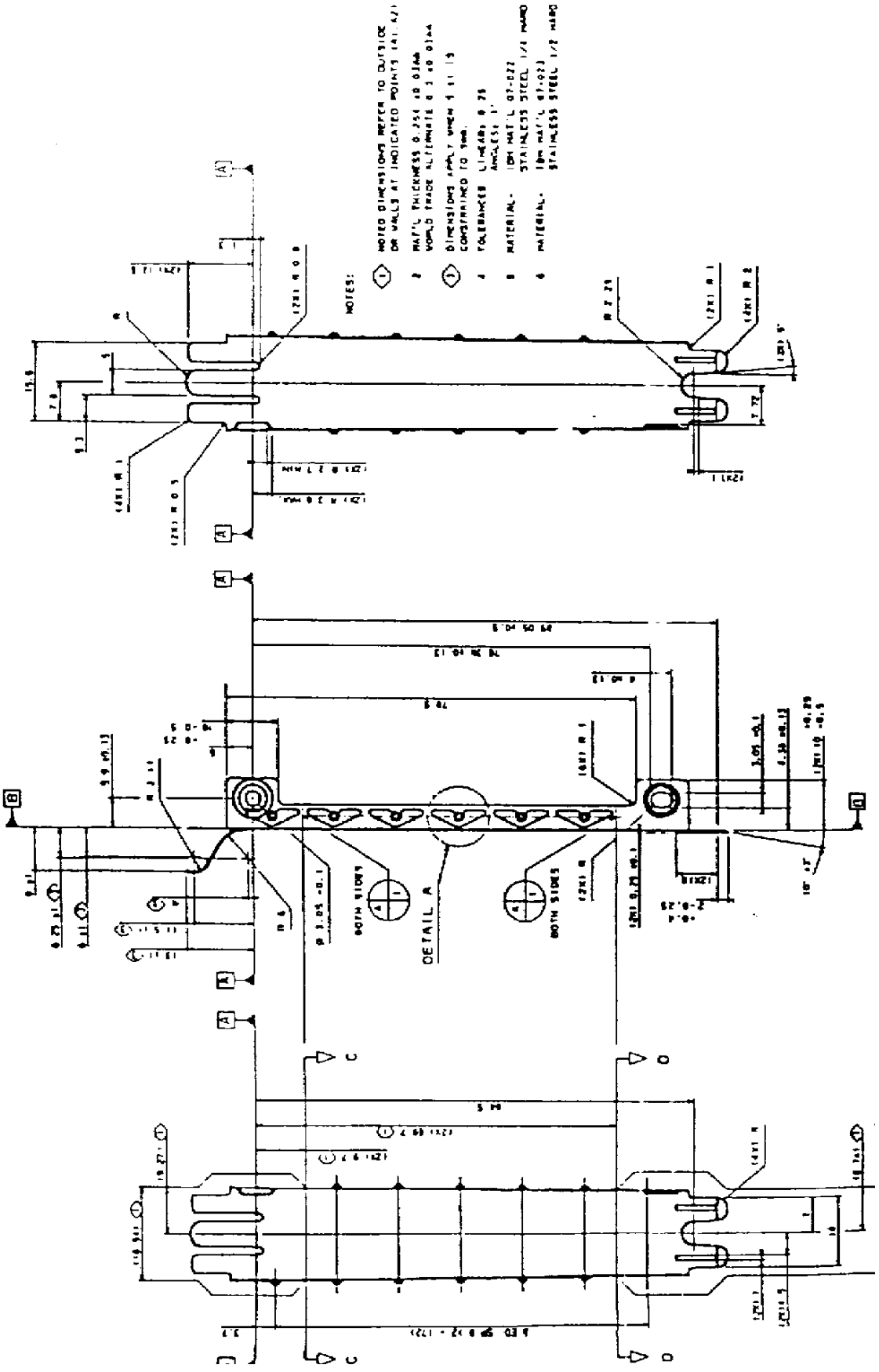


Рисунок 5-11: Кронштейн MC

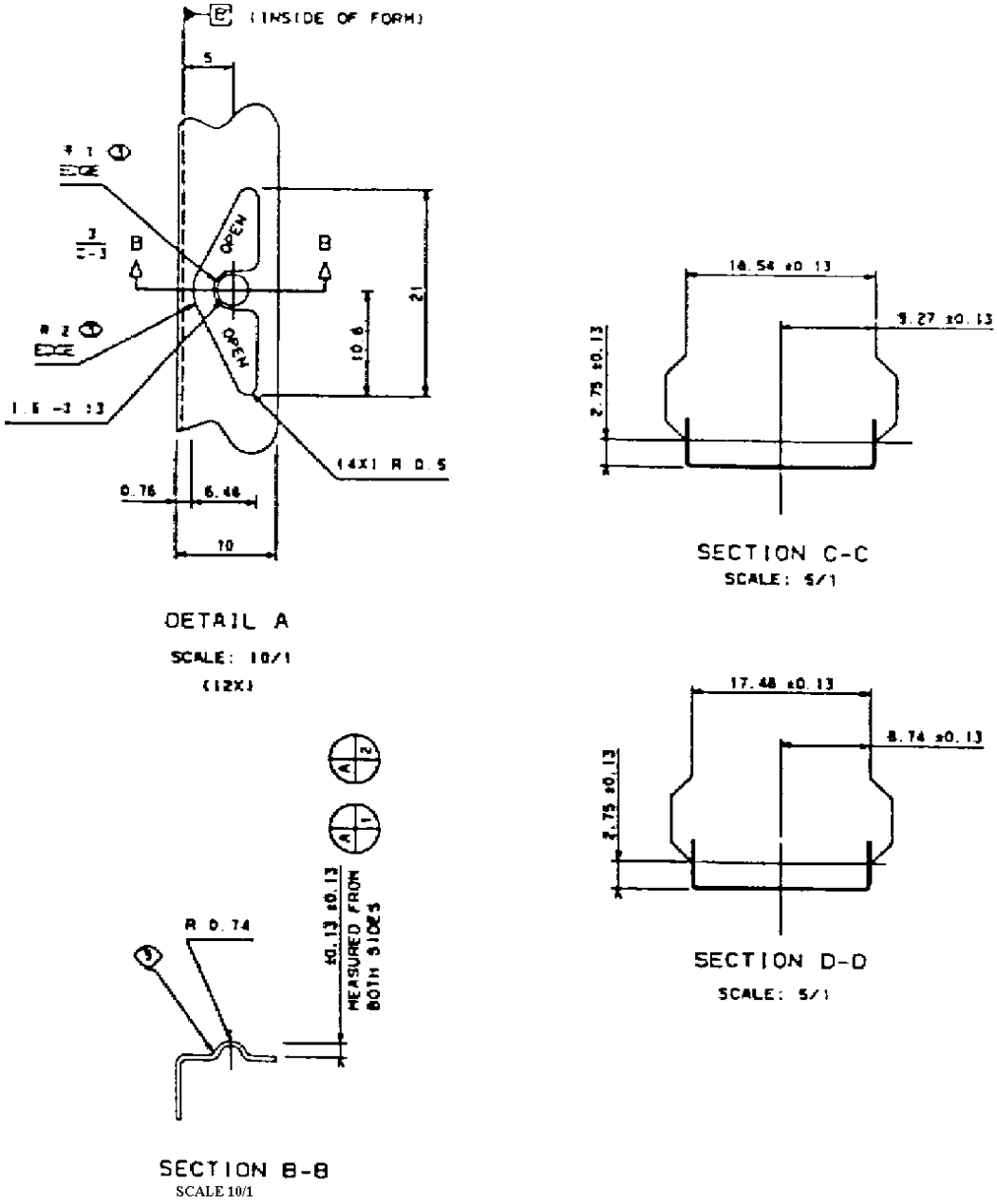


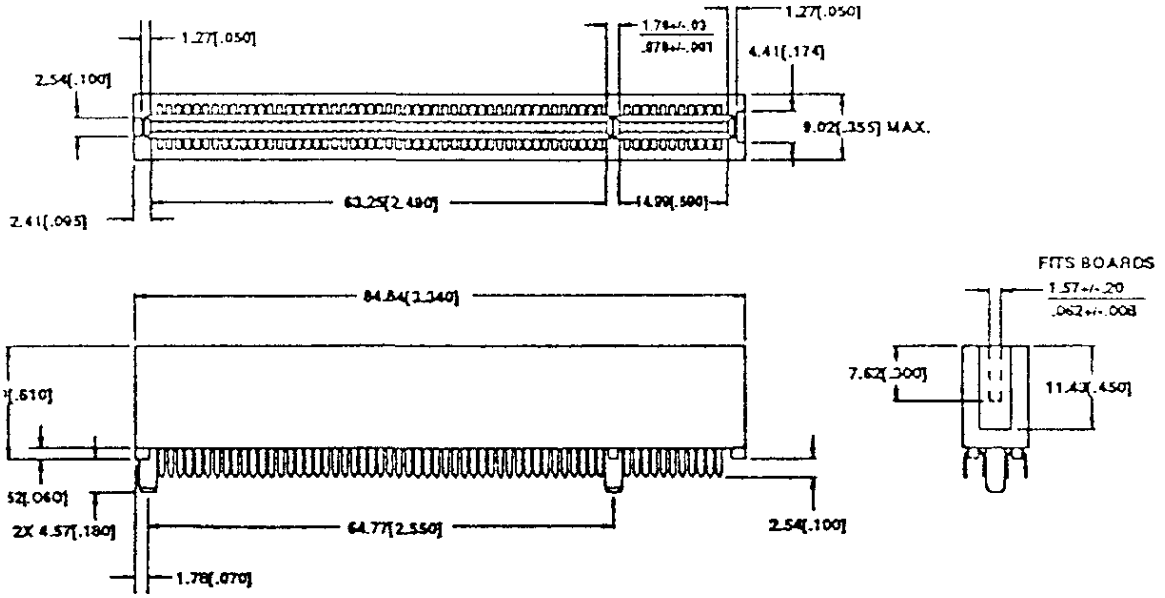
Рисунок 5-12: Детали кронштейна MC

5.2.1. Физическое описание разъема

Разъемы, поддерживающие PCI платы расширения, отведены от шины MC. MC разъемы подробно описаны и имеют проверенное качество и надежность. Имеются четыре разъема, которые могут использоваться в зависимости от PCI реализации. Различия между разъемами - 32 бит и 64 бит, и 5В и 3.3В среды передачи сигналов. Специальный код (ключ) дифференцирует показатели напряжения. Такой же физический разъем используется для 32-разрядных сигнальных оборудований. В одном положении ключ настроен на работу с 5В платами. При повороте на 180°, разъем допускает 3.3В платы. Вывод нумерует изменения разъема для различных сред передачи сигналов, чтобы поддержать то же самое относительное размещение сигналов на разъеме (см. Рисунки 5-14, 5-15, 5-17, и 5-l9 для детального изучения расположения платы).

На рисунках разъема, рекомендуемые детали схемы расположения даны как номинальные величины. Предельно допустимые отклонения в схеме расположения деталей должны соответствовать рекомендациям изготовителя разъемов и принятой инженерной практике.

См. рисунки с 5-13 по 5-19 для изучения размеров разъема и рекомендаций по схеме расположения. См. рисунки 5-20 до 5-26 для изучения размеров соединительного устройства для плат и допусков. Допуски для плат даны таким образом, чтобы можно было выпускать взаимозаменяемые платы .



Рèþунок 5-13 : 32-битовый разъем

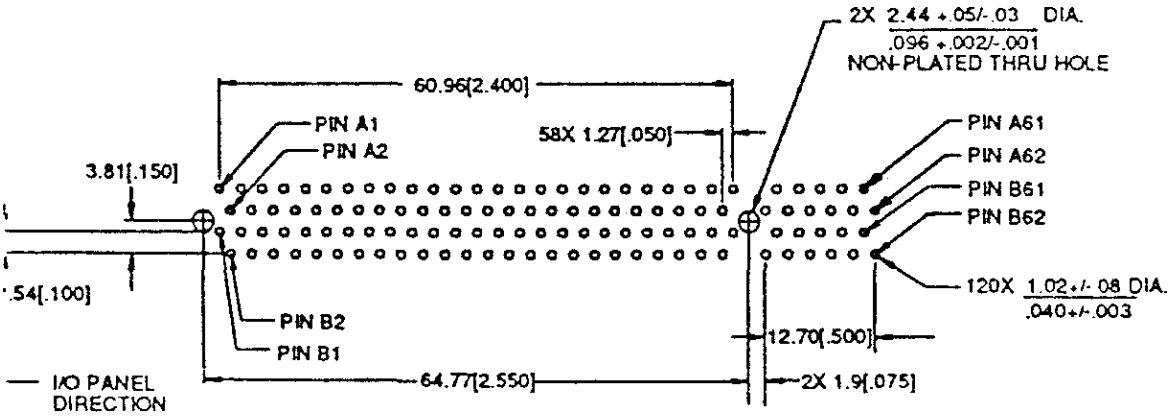


Рис. 5-14 : Рекомендации по схеме расположения 5В/32-битового разъема

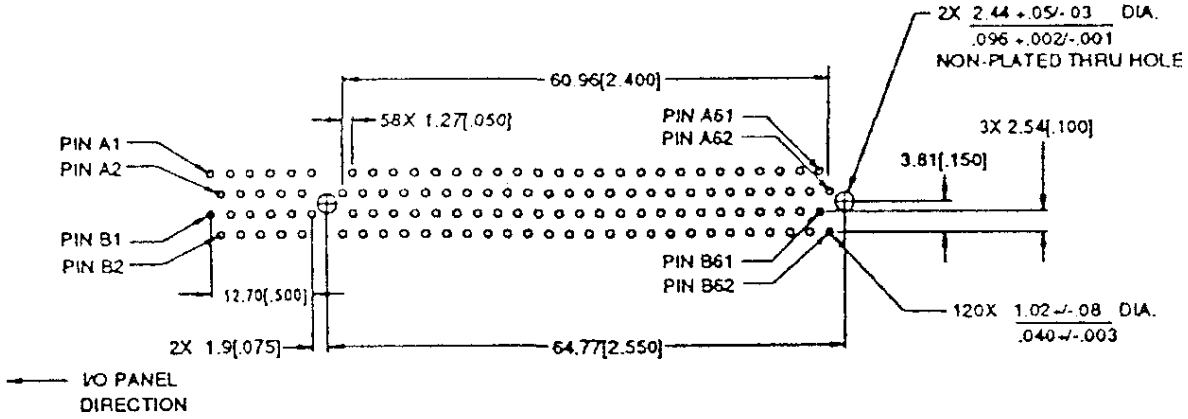


Рис. 5-15 : Рекомендации по схеме расположения 3.3V/32-битового разъема

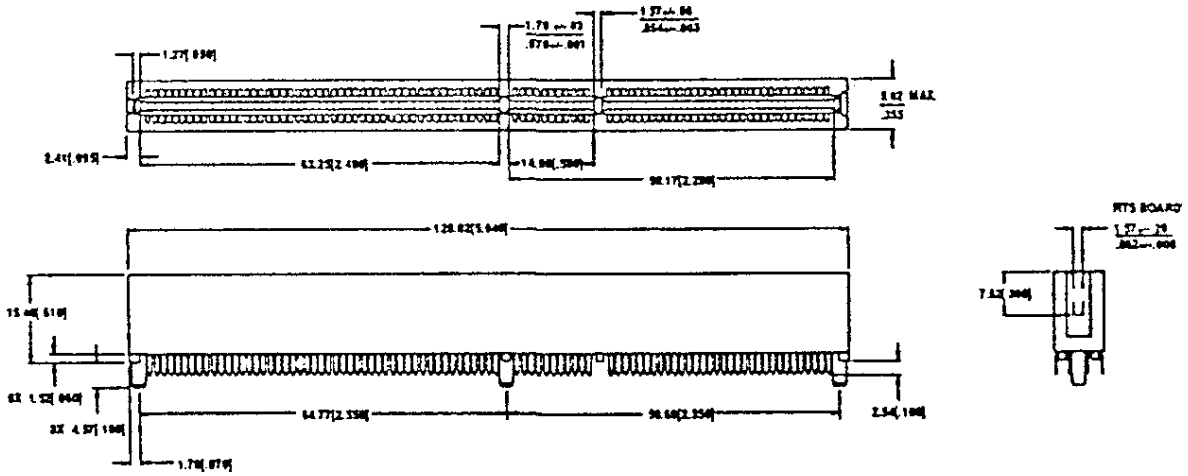


Рис. 5-16 : 5V/64-битовый разъем

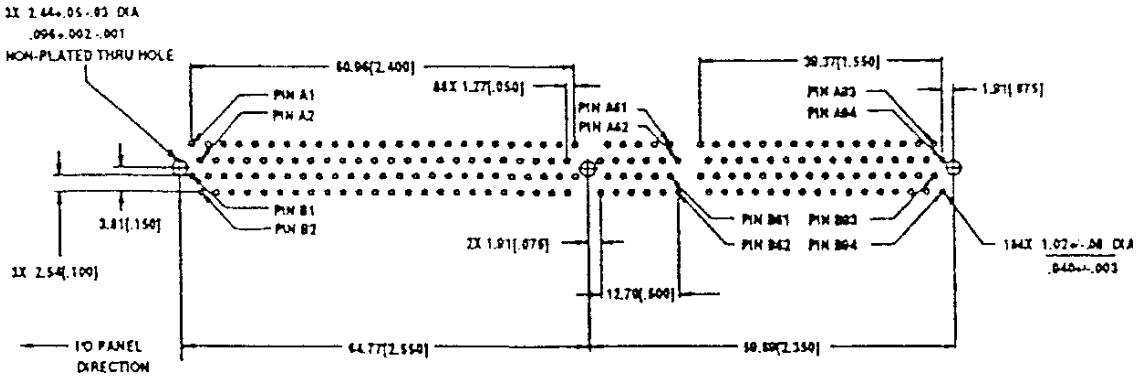


Рис. 5-17 : Рекомендации по схеме расположения 5V/64-битового разъема

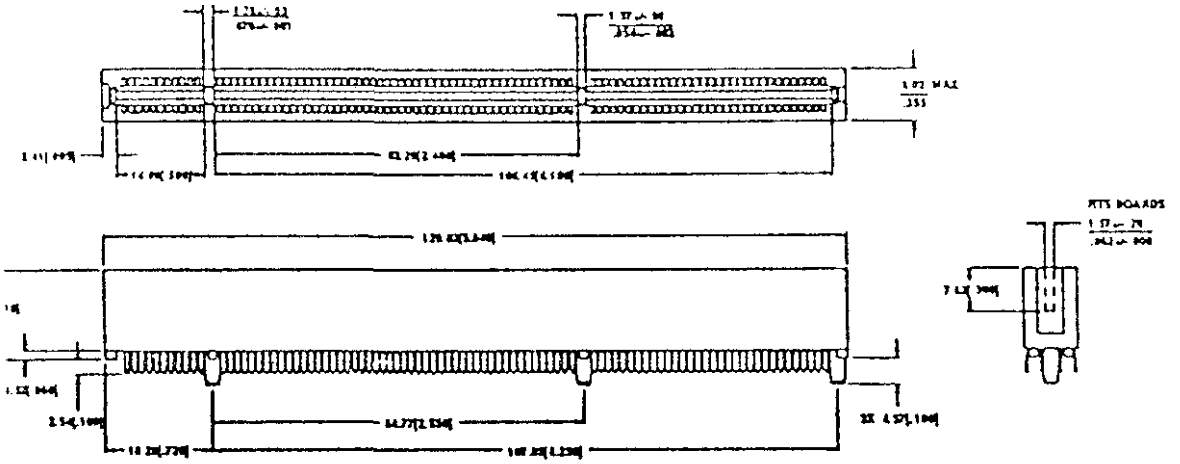


Рисунок 5-18 : 3.3V/64-битовый разъем

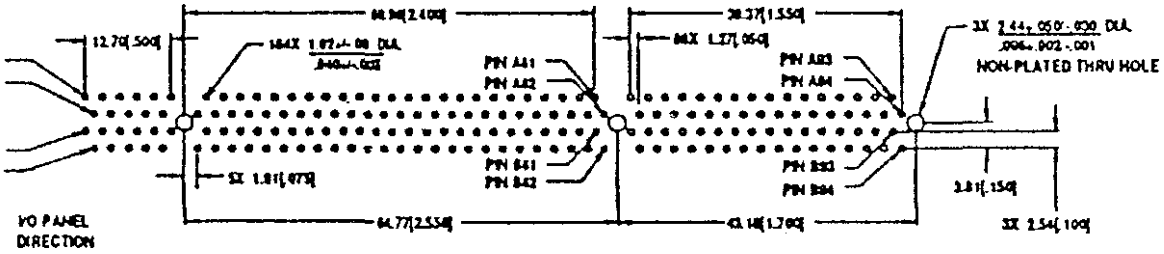


Рисунок 5-19 : Рекомендации по схеме расположения 3.3V/64-битового разъема

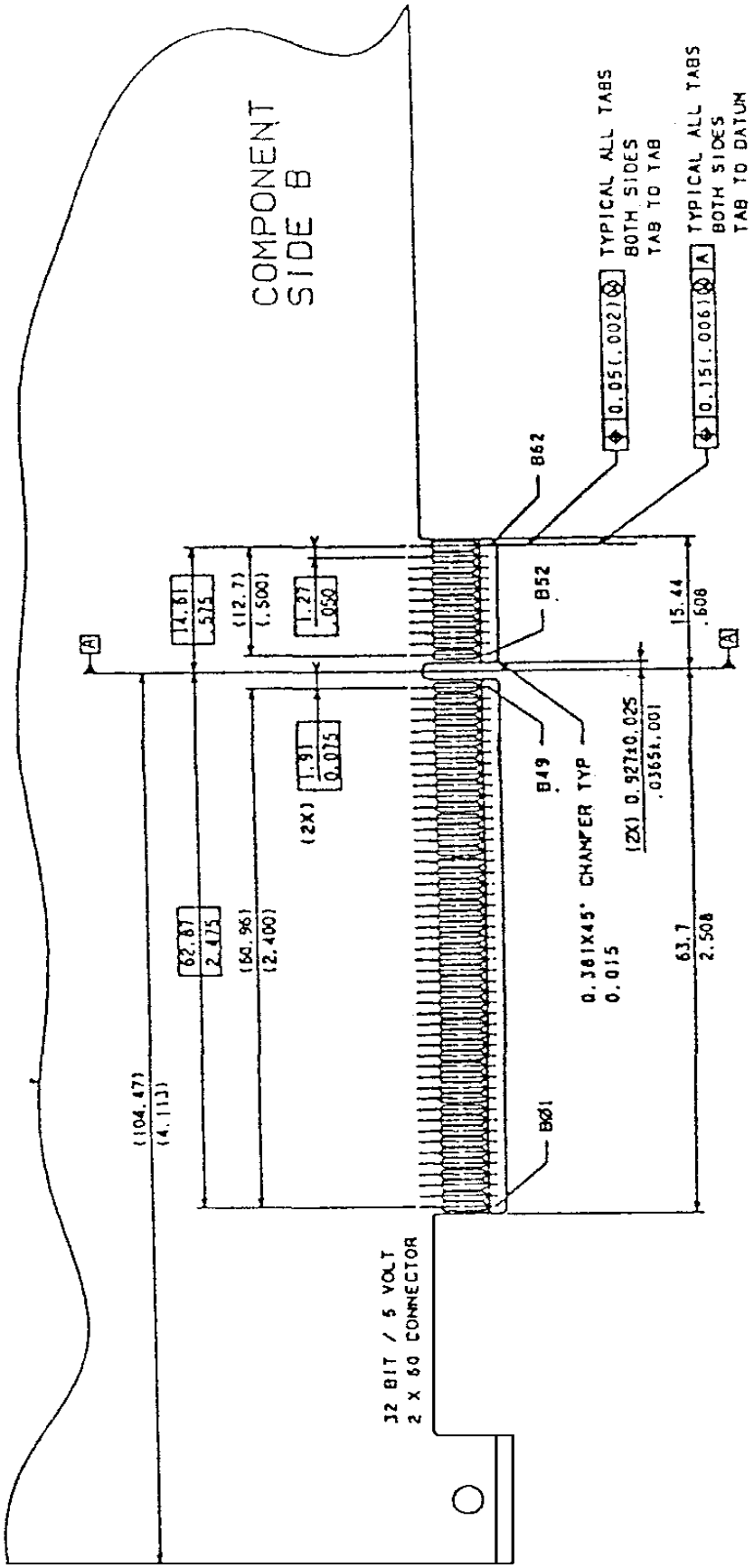


Рисунок 5-20 : Размеры 5V/32-битового соединительного устройства для плат и допуски

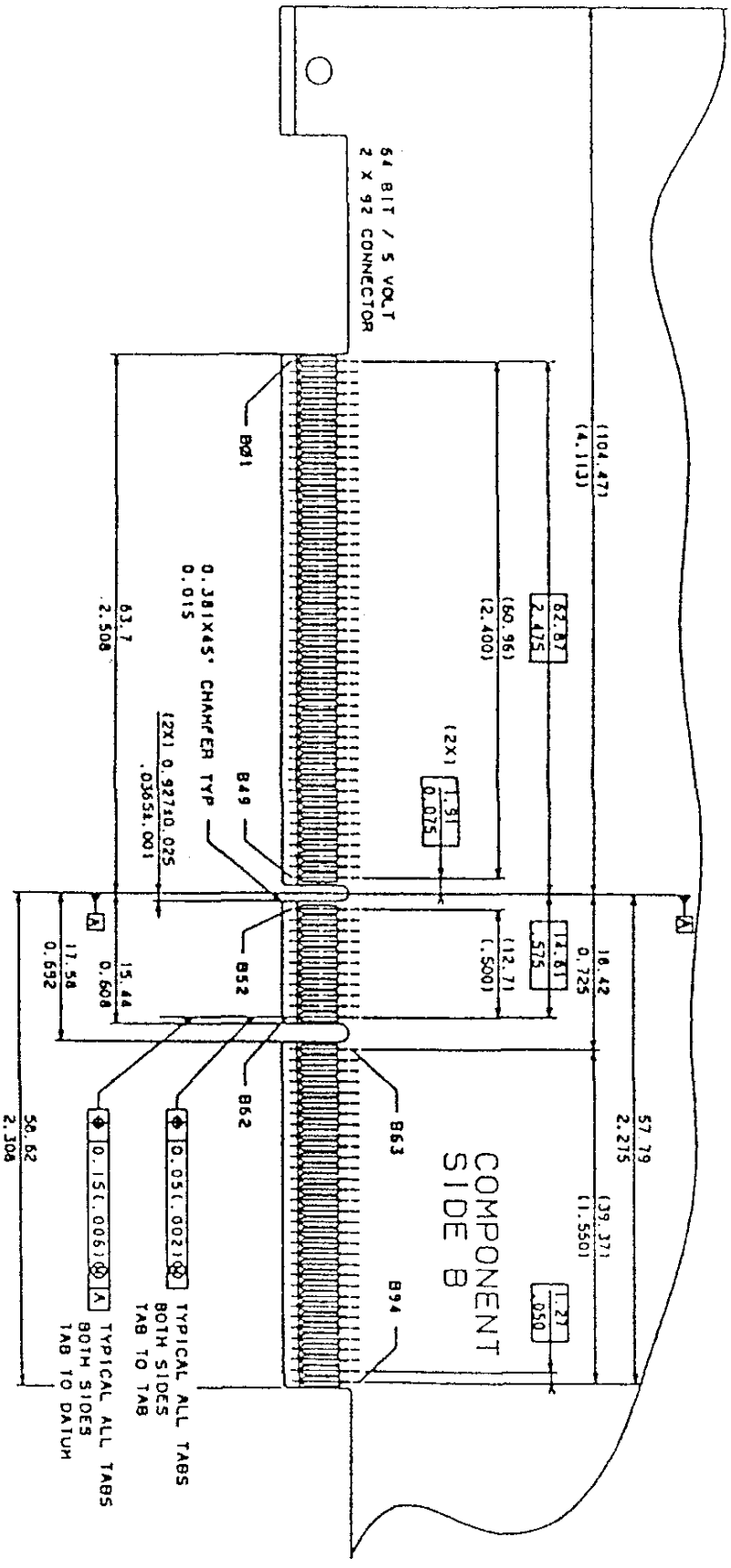


Рисунок 5-21: Размеры 5V/64-битового соединительного устройства для плат и допуски

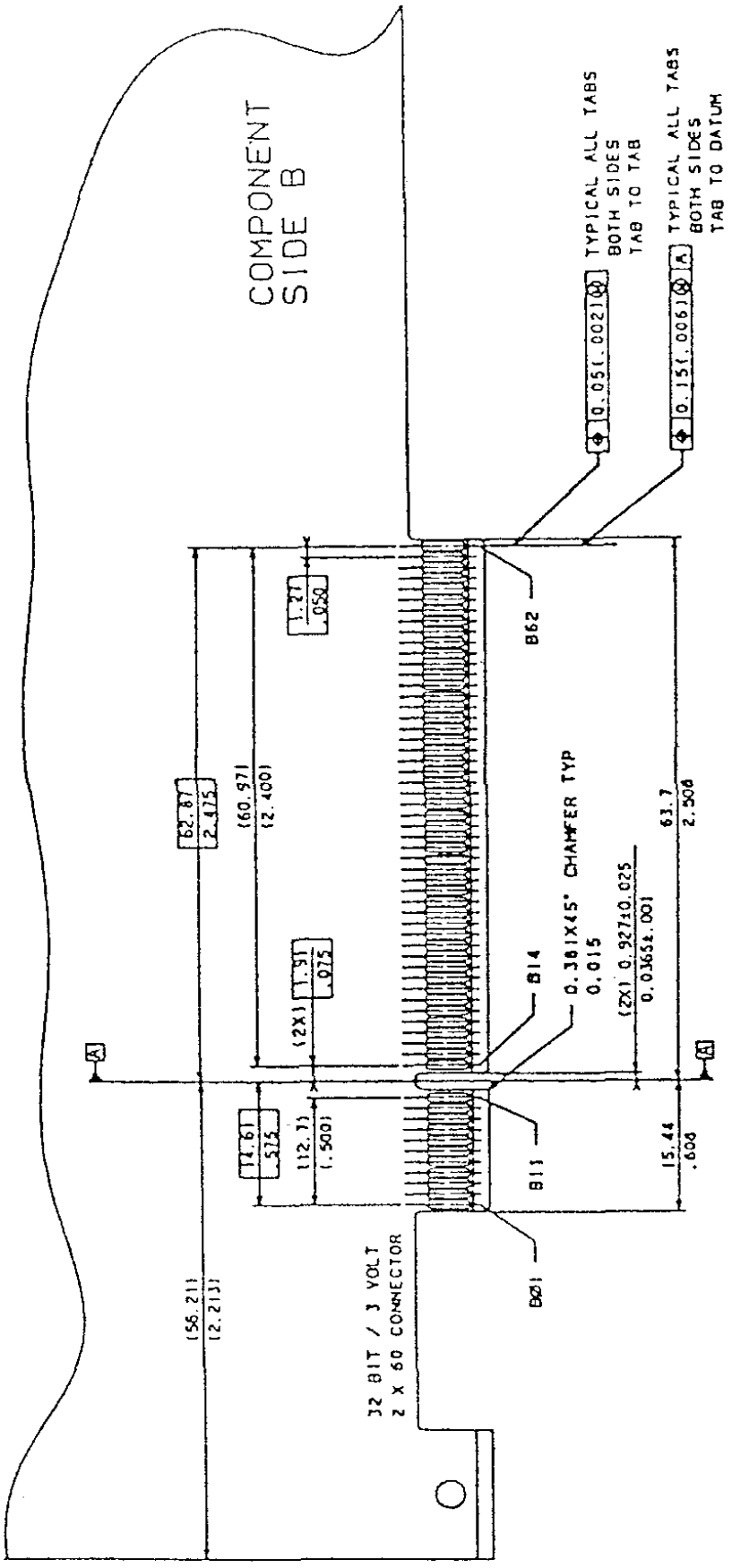


Рисунок 5-22: Размеры 3.3V/32-битового соединительного устройства для плат и допуски

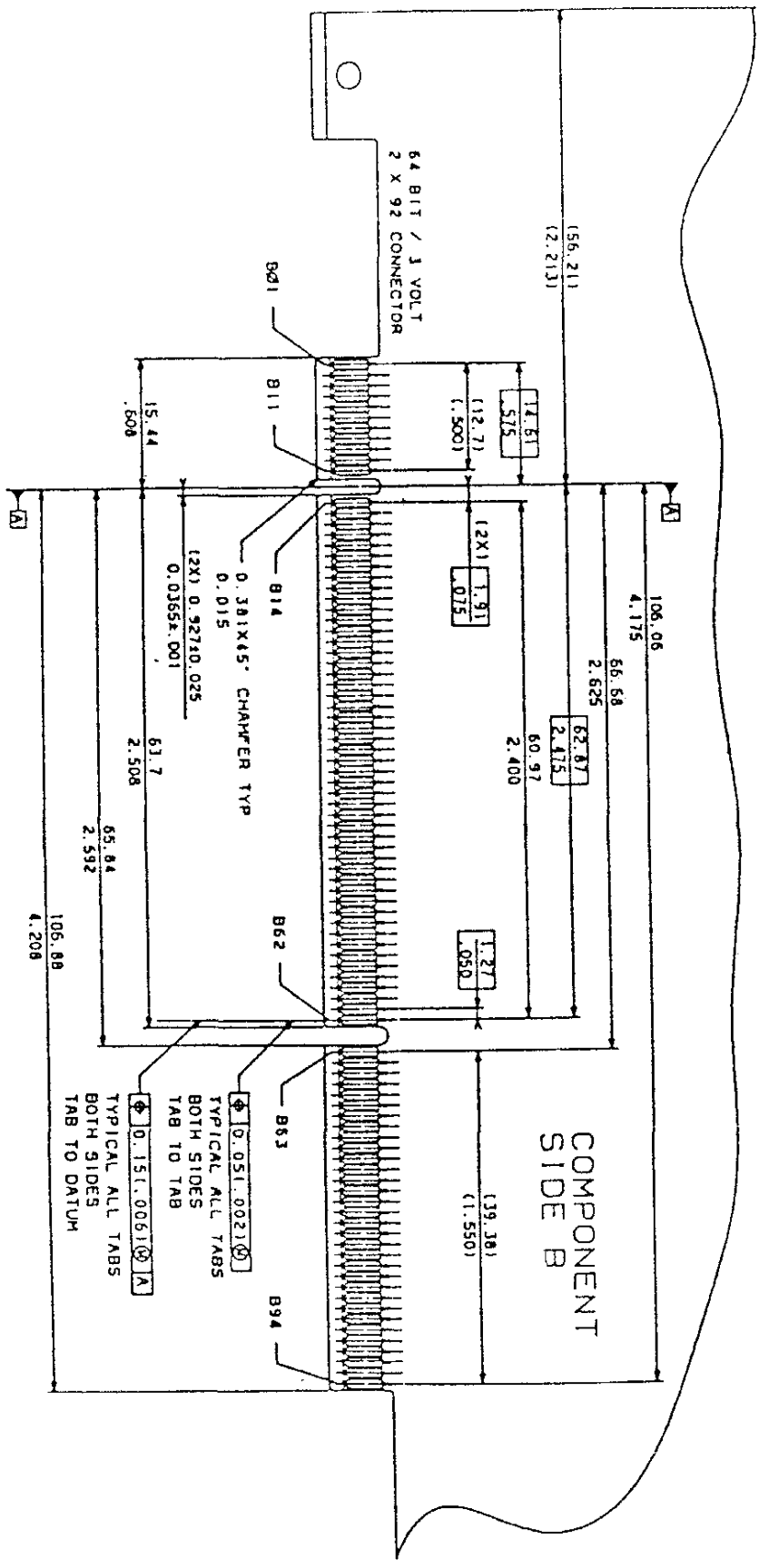


Рисунок 5-23: Размеры 3.3V/64-битового соединительного устройства для плат и допуски

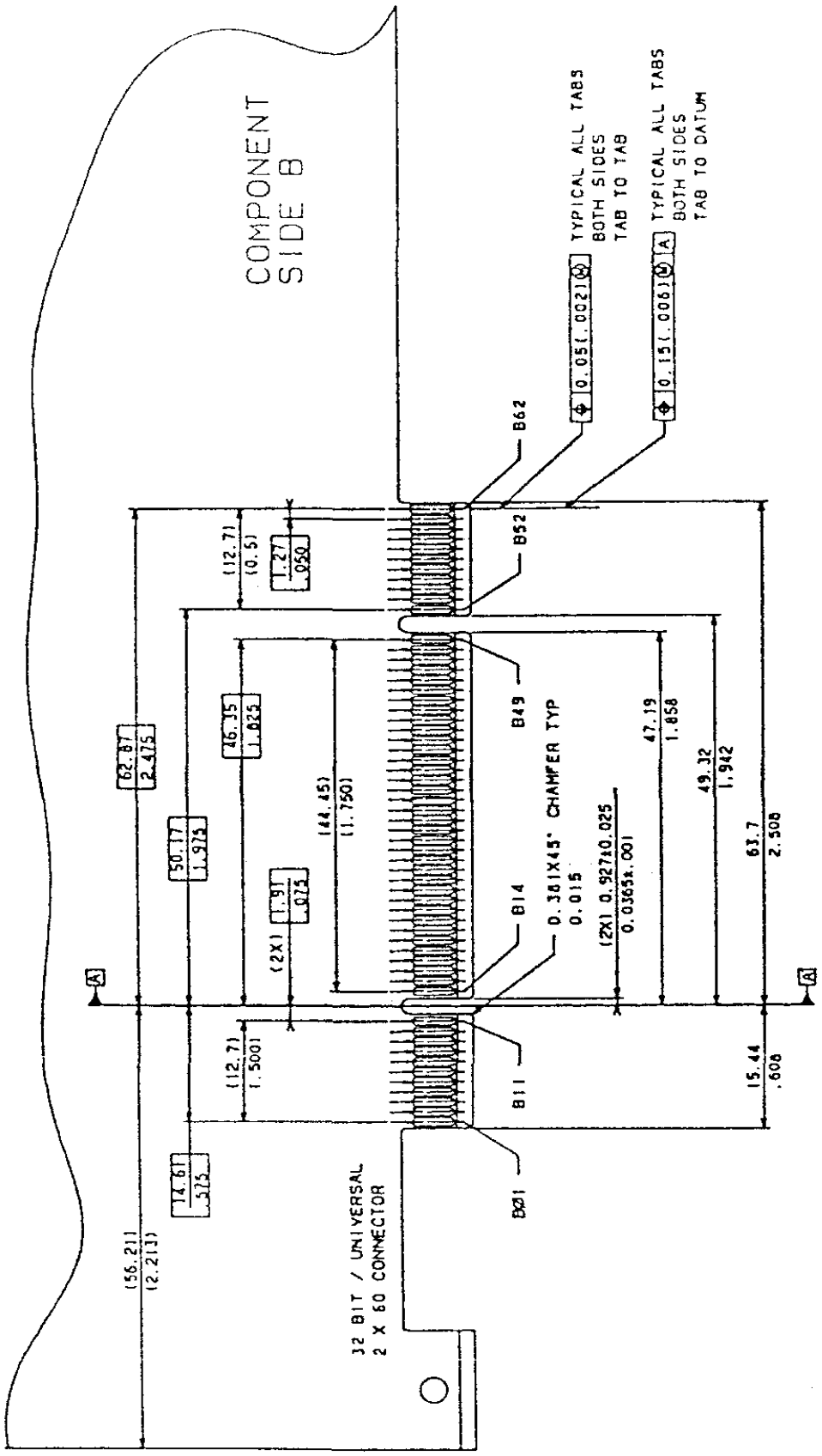


Рисунок 5-24 : Размеры универсального соединительного устройства для 32-битовых плат и допуски

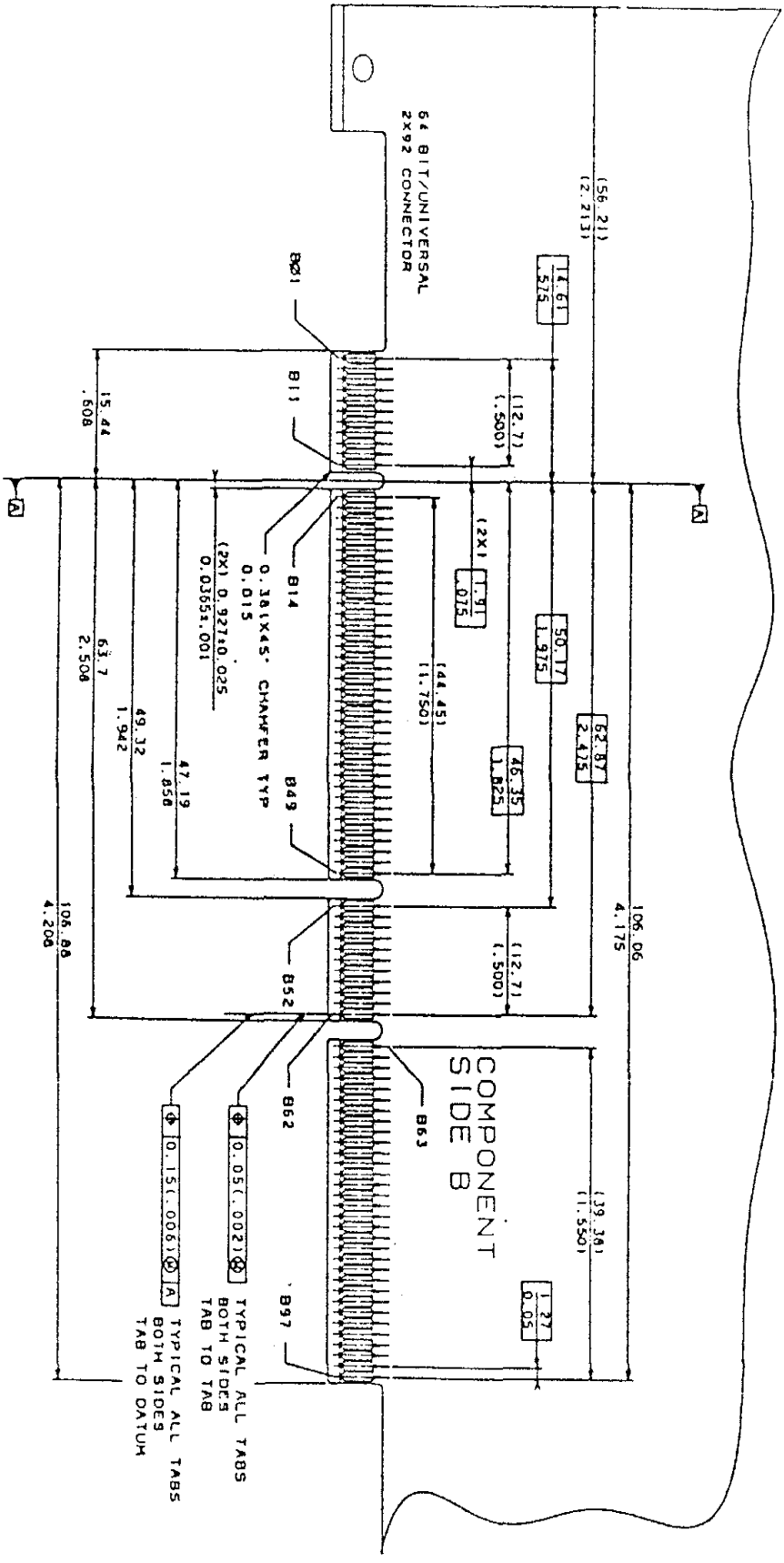


Рисунок 5-25: Размеры универсального соединительного устройства для 64-битовых плат и допуски

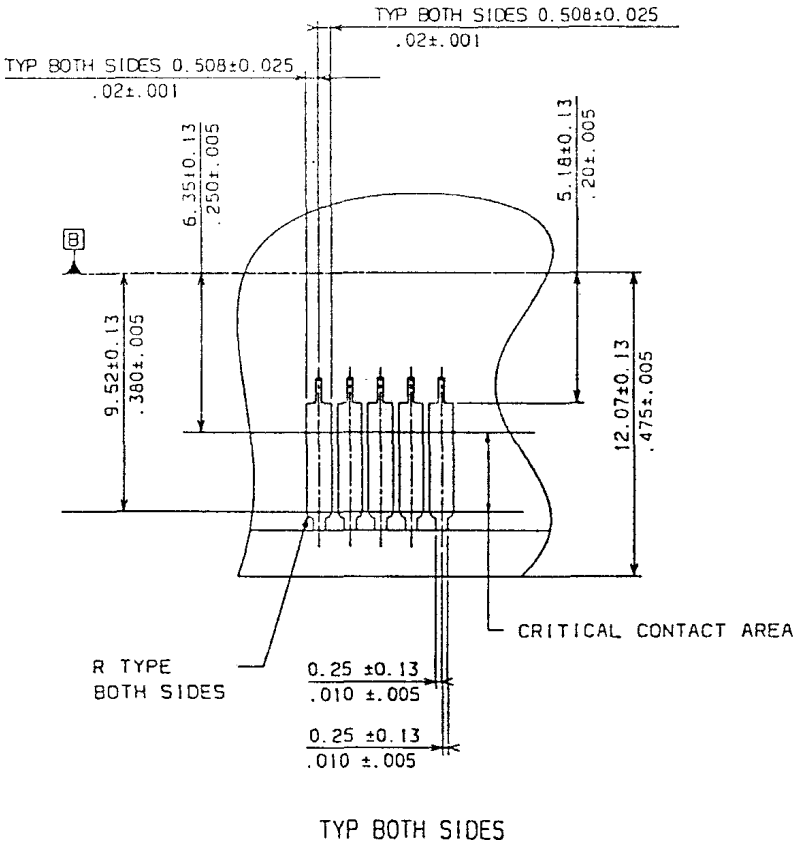


Рисунок 5-26: Контакты соединительного устройства для плат PCI

В таблице 5-1 перечислены разъемы PCI предлагаемые изготовителями.
Могут быть использованы и другие разъемы, если они соответствуют техническим характеристикам, изложенным в этом документе.

Таблица 5-1: разъемы PCI

Описание	AMP P/N	Burndy P/N	Foxconn P/N
32-bit, 5V	646255-1	CEE2X60S-V3Z14W	TBD
32-bit, 3.3V	646255-1	CEE2X60S-V3Z14W	TBD
64-bit, 5V	TBD	TBD	TBD
64-bit, 3.3V	TBD	TBD	TBD

Эта спецификация не содержит указаний относительно использования запасных частей того или иного производителя для конкретных условий функционирования.

5.2.1.1. Требования к физическим характеристикам разъема

Таблица 5-2 : Требования к физическим характеристикам разъема

Часть	Материалы
Корпус разъема	Полифенилина сульфид, UL класс легковоспламеняющихся 94V-0, цвет : белый.
Контакты	Фосфоризованная бронза.
Отделка контакта	Как минимум 0.000030-дюймовое золотое покрытие над не более чем 0.000050-дюймовым никелевым припоем в области контакта. В качестве альтернативы допустимо нанесение золотого блеска толщиной как минимум 0.000040 дюйма (1 микрон) из палладия или сплава палладий-никель над никелевым припоем в области контакта.

5.2.1.2. Стандартные технические характеристики

Таблица 5-3 : Требования к механическим характеристикам разъема

Параметр	Требования стандарта
Долговечность	100 сопряженных циклов без физических повреждений или превышения предельно-допустимых показателей контактного сопротивления при соединении с платами подобранными в соответствии с рекомендациями.
Сила сцепления	6 oz. (1.7N) максимум на противопоставленную контактную пару с использованием MIL-STD-1344, метод 2013.1 и датчик на MIL-C-21097 с профилем как показано в спецификации встроенной платы.
Нормальная сила контактного сцепления	75 грамм миíимóм.

Таблица 5-4 : Требования к электрическим характеристикам разъема

Параметр	Требования стандарта
Сопротивление контакта	(низкий уровень сигнала) 30 mOм max. при инициализации, максимальное увеличение при проведении тестирования 10 mOм. Испытания по определению контактного сопротивления проводятся с использованием MIL-STD-1344, метод 3002.1.
Сопротивление изоляции	1000 MΩ min. использован измерительный прибор MIL-STD-202, метод 302. Положение B.
Диэлектрическая прочность по напряжению	500V переменного тока. Использован измерительный прибор MIL-STD-1344, метод D3001.1. Положение 1.
Ёмкость	2 pF max. @1 MHz.
Текущая оценка	повышение на 1A при увеличении на 30 °C.
Оценка напряжения	125V.
Сертификат качества	UL Распознавание и требуемое подтверждение CSA

Таблица 5-5: Требования к характеристикам разъема, касающихся окружения

Параметр	Требования стандарта
Рабочая температура	от - 40 °C до +105 °C
Термический удар	от -55 °C до +85 °C, 5 циклов на MIL-STD-1344, метод 1003.1.
Испытания в потоке смешанного газа	Баттель, класс 2. Разъем сопряжен с платой и испытан по методу Баттеля.

5.2.2. Планарное исполнение

Возможны два типа планарного исполнения: разъемы монтируются на плоскости или на плате. Для иллюстрации здесь приводится подробное описание только разъемов, монтированных на плоскости. Эти основные принципы могут быть приложены к конструкции плат. Смотри рис.5-27, 5-28 и 5-29 для изучения деталей EISA, ISA и MC плат, соответственно. На рисунках показано относительное взаиморасположение базовых линий разъема PCI 5V и 3.3V к базовым линиям разъемов EISA, ISA и MC. Оба разъема на 5V и 3.3V показаны на плоскости с точным соблюдением пропорций. По нормальному, эта система должна включать в себя либо PCI разъем на 5V, либо на 3.3V , но не оба сразу. Стандартные зазоры между платами составляют 0.8 дюйма для ISA/EISA и 0.85 дюйма для MC, вследствие чего в системе имеется только один общий слот для всех плат. Система PCI допускает установку дополнительных слотов в том случае, если существующее пространство окажется недостаточным для размещения плат. Если смотреть на конструкцию сзади, то общий слот размещен таким образом, что слоты ISA, EISA или MC располагаются справа, а PCI слоты слева.

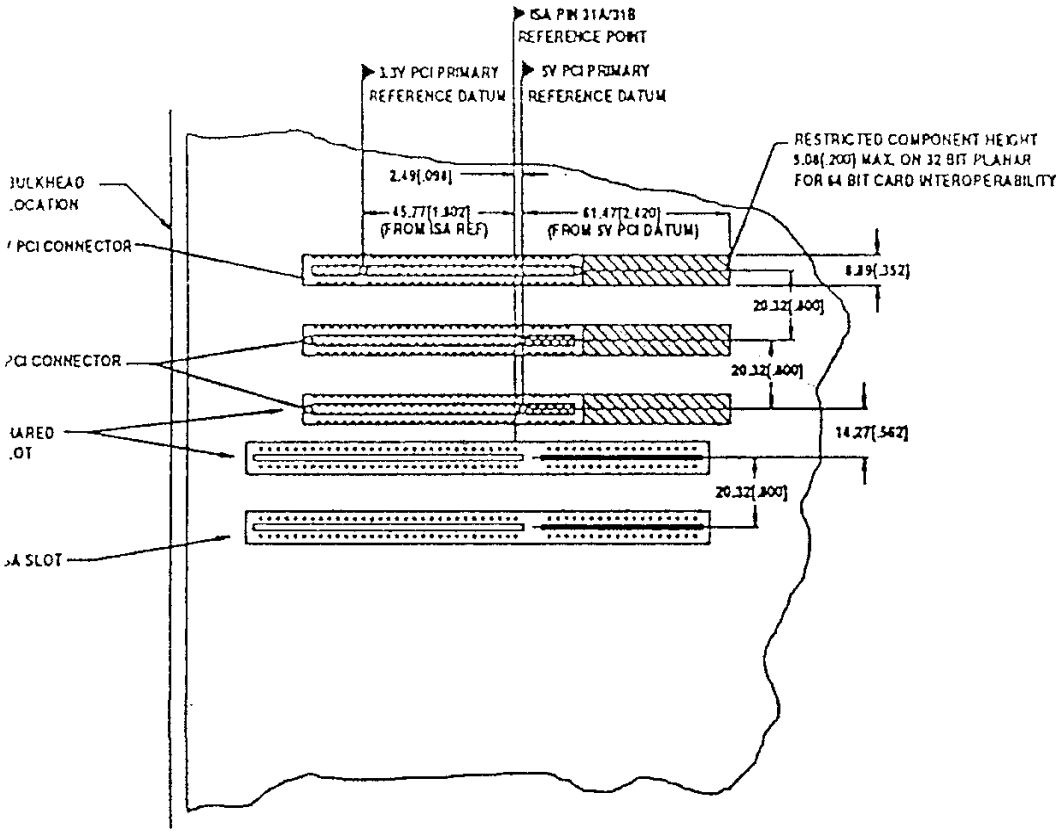


Рисунок 5-27 : Расположение PCI разъема на плоскости относительно базовой линии ISA разъема.

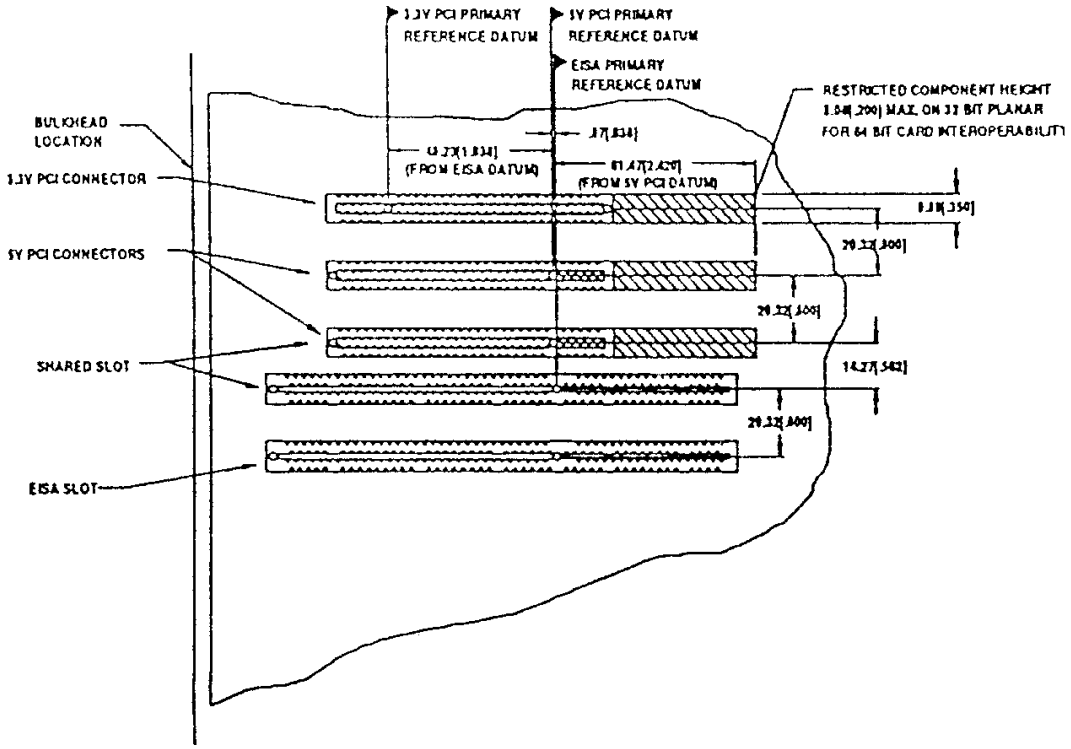


Рис. 5-28 : Расположение PCI разъема на плоскости относительно базовой линии EISA разъема

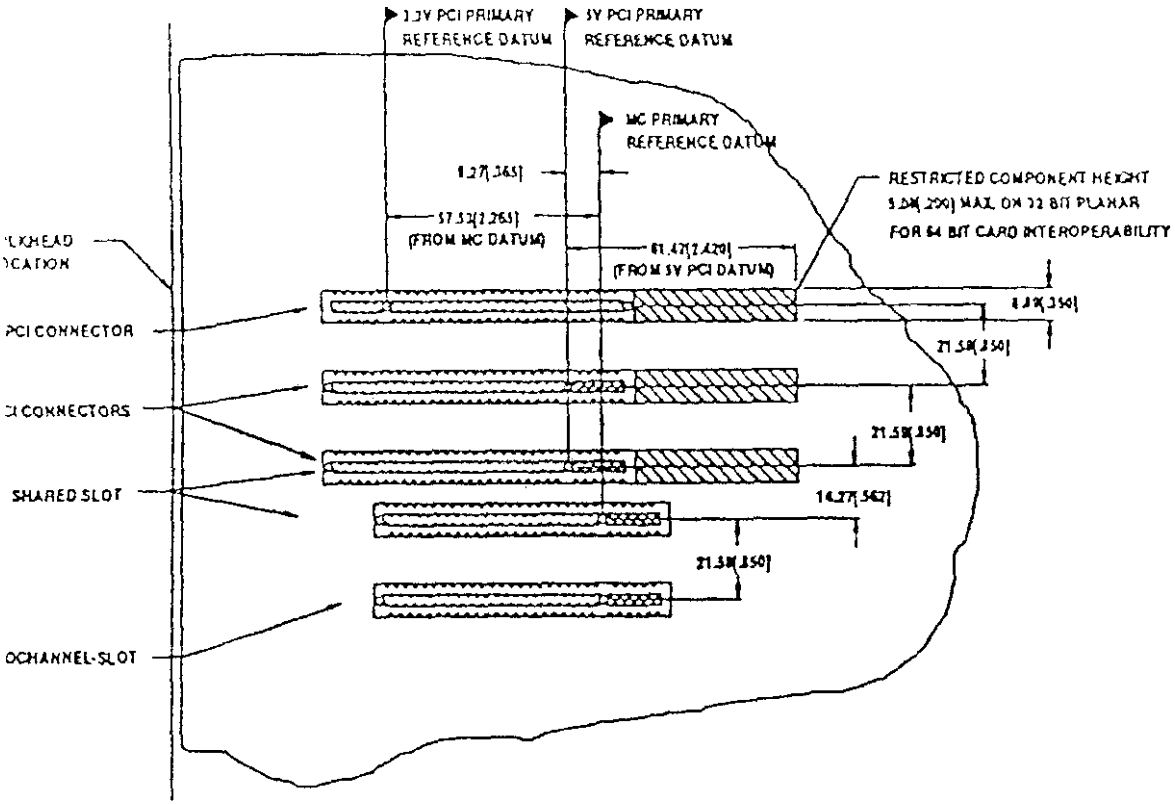


Рис. 5-29 : Расположение PCI разъема на плоскости относительно базовой линии MC разъема

Глава 6
Пространство конфигурации

В этой главе дается характеристика программной модели и правила использования для конфигурации регистрового пространства в соответствующих приборах PCI. Эта глава ограничивается определением составных частей PCI для множества типов систем. В этой главе не рассматривается система зависимых выходов для платформ особого типа. Система зависимых выходов характерна для PC - совместимых платформ, таких, как отображение адресного пространства PCI на адресное пространство процессора, входная последовательность команд, требования соединения host-to-PCI bus и т.д., эту информацию вы сможете найти в «Руководстве по проектированию PCI систем».

Пространство конфигурации PCI определяется с целью обеспечения необходимого набора ловушек (аппаратное средство отладки и диагностирования микропрограммы) конфигурации, отвечающих потребностям текущих и ожидаемых механизмов системы конфигурации, без точного определения этих механизмов или наложения каких-либо ограничений на их использование. Критерии оценки этих ловушек конфигурации перечислены ниже:

- наличие достаточной поддержки, позволяющей будущим механизмам конфигурации обеспечивать:
  - полное перераспределение оборудования, включая систему прерывания
  - монтаж, конфигурация и самозагрузка без вмешательства пользователя
  - составление адресного пространства системы с помощью оборудования независимого программного обеспечения
- эффективная поддержка существующих механизмов конфигурации (например, утилита конфигурации EISA)
- минимальная кремниевая поддержка создания требуемых функций
- система рычагов такая же, как и при шаблонном подходе к общим функциям, однако, это не мешает налагать особые требования на оборудование

6.1. Организация пространства конфигурации

Этот раздел посвящен организации регистров пространства конфигурации и характеристике специальной записывающей структуры или маски на 256-байтовом пространстве. Это пространство поделено на заранее определенный заголовок и область, зависящую от устройства. Устройства реализуют только необходимые и соответствующие регистры в каждой области. Пространство конфигурации устройства должно быть доступным в любое время, а не только при самозагрузке системы.
Вышеупомянутый заголовок имеет размер 64 байта и каждое устройство должно поддерживать формат этой области. Эта область состоит из полей, которые однозначно идентифицируют устройство и позволяет устройству быть контролируемым. Остальные 192 байта специфичны для устройства и не описываются в этом документе.

Программное обеспечение системы может нуждаться в сканировании канала PCI с целью определения, какие устройства реально присутствуют. Для этого программное обеспечение конфигурации должно считывать ID (идентификатор) поставщика в каждом возможном слоте PCI. Главная шина моста PCI должна однозначно определять попытки чтения ID поставщика несуществующего устройства. Поскольку 0FFFFh является неиспользуемым ID поставщика для главной шины моста PCI , то допустимо возвращение значения из всех единиц при попытке считывания регистров пространства конфигурации несуществующих устройств.

31 16 15 0

ID устройства		ID поставщика		00h
Status		Команда		04h
Код класса			ID изделия	08h
BIST	Тип заголовка	Таймер времени ожидания	Имеющийся размер строки	0Ch
Базовые адресные регистры				10h 14h 18h 1Ch 20h 24h
зарезервировано				28h
зарезервировано				2Ch
Базовый адрес расширения ПЗУ				30h
зарезервировано				34h
зарезервировано				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

Рис. 6-1 : Заголовок пространства конфигурации

Рисунок 6-1 описывает расположение 64-байтовой предопределенной части заголовка 256-байтового пространства конфигурации, которую должно поддерживать каждое PCI устройство. Устройства должны поместить все необходимые специфичные им регистры только в ячейки от 64 до 255. Все многобайтовые числовые поля следуют в little-endian порядке. То есть нижние адреса содержат наименее значительные части поля. Программное обеспечение должно соблюдать осторожность, чтобы правильно работать с полями закодированных битов, которые имеют биты, зарезервированные для будущего использования. При чтении, программное обеспечение должно использовать соответствующие маски(шаблоны), чтобы извлечь определенные биты, и не может полагаться на зарезервированные биты, являющиеся специфической величиной. При записи, программное обеспечение должно гарантировать, что значения зарезервированных битов сохраняются.

То есть значения зарезервированных битов должны сначала читаться, потом объединены с новыми значениями других битов и затем данные записываются обратно. Раздел 6.2. описывает регистры в вышеупомянутой части заголовка пространства конфигурации.
Вышеупомянутая часть заголовка пространства конфигурации разделена на две части. Первые 16 байтов одинаково определены для всех типов устройств. Оставшиеся 48 байтов могут иметь различное расположение в зависимости от основной функции, которую поддерживает устройство. Общее расположение определяется в зависимости от поля тип заголовка (размещенного на 0Eh). В настоящее время определен только один тип заголовка (00h), который имеет расположение, показанное в рисунке 6-1. В следующих изданиях будут определены другие типы заголовков имеющих особые функции (т. к., дополнительная память, PCI-to-PCI мосты, и т.д.).
Все PCI совместимые устройства должны соответствовать ID поставщика, ID устройства, командным и статусным полям в заголовке. Реализация других регистров не является обязательным (то есть, они могут обрабатываться как зарезервированные регистры) в зависимости от функциональных возможностей устройства. Если устройство выполняет функции необходимые для регистра , то он должен реализовывать их в определенном месте и с определенной функциональностью.

6.2. Функции пространства конфигурации

PCI имеет потенциал для значительного облегчения конфигурирования системы. В целях реализации этого потенциала все PCI приборы должны выполнять определенные функции, которые программное обеспечение конфигурации системы может использовать. В этом разделе также перечисляет функции, которые должны выполняться всеми PCI приборами через регистры, расположенные в вышеупомянутой части заголовка пространства конфигурации. Точный формат этих регистров (то есть, число задействованных битов) зависит от типа прибора. Однако, некоторые общие правила должны соблюдаться. Все регистры должны иметь считывающие устройства и полученные данные должны указывать реально использованные прибором значения.
Пространство Конфигурации предназначено для реализации таких функций как конфигурация, инициализация, и обработка катастрофических ошибок функций. Его использование должно быть ограничено для инициализации программного обеспечения и обработки ошибок программным обеспечением. Все функционирующее программное обеспечение должно использовать ввод/вывод и/или доступ к пространству памяти, чтобы управлять регистрами прибора.

6.2.l. Идентификация устройства

Пять полей в вышеупомянутой части заголовка имеют дело с идентификацией устройства. Все PCI приборы должны иметь эти поля. Общее программное обеспечение конфигурации должно легко определять какое устройство доступно на PCI шине системы. Все эти регистры только для чтения.

ID изготовителя	Это поле идентифицирует изготовителя прибора. Соответствующие опознавательные знаки изготовителя размещены на корпусах устройств с целью идентификации. 0FFFFh - недопустимое значение для ID изготовителя.
ID устройства	Это поле идентифицирует конкретный вид устройств. Этот идентификатор определяется изготовителем.
Изменение ID	Этот регистр определяет специальные изменения идентификатора. Значение определяется изготовителем. Ноль - приемлемая значение. Это поле должно рассматриваться как расширение, определенное изготовителем, к ID прибора.

Тип заголовка этот байт определяет расположение байтов от 10h до 3Fh в пространстве конфигурации, а также, наличие у прибора множественных функций. Бит 7 в этом регистре используется для идентификации многофункционального прибора. Если бит равен 0, то прибор имеет одну функцию. Если бит равен 1, то прибор многофункционален. Биты от 6 до 0 определяют расположение байтов от 10h до 3Fh. При первоначальной установке 00h определен и характеризует схему расположения, показанную на рисунке 6-1. Все другие кодировки зарезервированы.

Код Класса регистр Кода Класса только для чтения и используется для реализации общих функций прибора и (в некоторых случаях) для согласования программ на регистровом уровне. Регистр делится на три однобайтовых поля. Старший байт (на 0Bh) - код базового класса определяющий тип осуществляемых прибором функций. Средний байт (на 0Ah) - код подкласса, который идентифицирует более конкретно функции прибора. Младший байт (на 09h) принадлежит к программному интерфейсу регистрового уровня (если таковой имеется в наличии) таким образом, что прибор может взаимодействовать с отдельным от него оборудованием программного обеспечения . В таблице 6-1 приведены величины , заданные для базовых классов (то есть, старший байт в поле Кода Класса).

Таблица 6-1: кодирование Регистра Кода Класса

Базовый класс	Значение
00h	Прибор, построен перед окончательной установкой Кода Класса.
01h	Контроллер групповой памяти
02h	Контроллер сети
03h	Контроллер дисплея
04h	Мультимедиа
05h	Контроллер памяти
06h	Перемычки(мосты)
07h - FEh	Зарезервировано
FFh	Прибор не вписывается в рамки какого-либо определенного класса.

Кодирование подкласса и интерфейса программирования дано ниже для каждого определенного базового класса. Все неопределенные кодировки зарезервированы.

Базовый класс 00h

Этот базовый класс определяется для установления соответствий оборудования для приборов, которые были определены прежде, чем было определено поле Кода Класса. Никакие новые приборы не должны использовать это значение и существующие приборы должны быть переключаемыми , чтобы более соответствовать значению, если возможно.

Для кодов класса с этой величиной базового класса, для оставшихся полей имеются два определенных значения как показано в таблице ниже. Все другие величины зарезервированы.

Базовый класс	Подкласс	Программа	Значение
00h	00h	00h	Все изготавливаемые в настоящее время приборы за исключением VGA совместимых приборов.
00h	01h	00h	VGA совместимый прибор

Базовый класс 01h

Этот базовый класс определен для всех типов контроллеров накопителей информации. В этом классе выделяются несколько подклассов. Не определено никаких специальных программируемых интерфейсов на регистровом уровне.

Базовый класс	Подкласс	Программа	Значение
01h	00h	00h	контроллер шины SCSI
01h	01h	00h	IDE контроллер
01h	02h	00h	Контроллер гибкого диска
01h	03h	00h	контроллер шины IPI
01h	80h	00h	Контроллер других носителей информации

Базовый класс 02h

Этот базовый класс определен для всех типов контроллеров сети. В этом классе выделяются несколько подклассов. Не определено никаких специальных программируемых интерфейсов на регистровом уровне.

Базовый класс	Подкласс	Программа	Значение
02h	00h	00h	Ethernet контроллер
02h	01h	00h	Контроллер Toking Ring
02h	02h	00h	FDDI контроллер
02h	80h	00h	Контроллер другой сети

Базовый класс 03h

Этот базовый класс определен для всех типов контроллеров дисплея. В этом классе выделяются несколько специфичных подклассов, каждое из которых имеет известный программируемый интерфейс регистрового уровня.

Базовый класс	Подкласс	Программа	Значение
03h	00h	00h	VGA совместимый контроллер
03h	01h	00h	XGA контроллер
03	80h	00h	Другой контроллер дисплея

Базовый класс 04h

Этот базовый класс определен для всех типов устройств мультимедиа. В этом классе выделяются несколько подклассов. Не определено никаких специальных программируемых интерфейсов на регистровом уровне.

Базовый класс	Подкласс	Программа	Значение
04h	00h	00h	Видеосигнал
04h	01h	00h	Аудиосигнал
04h	80h	00h	Другое устройство мультимедиа

Базовый класс 05h

Этот базовый класс определен для всех типов контроллеров памяти (обратитесь к разделу 6.2.5.3.). В этом классе выделяются несколько подклассов. Не определено никаких специальных программируемых интерфейсов на регистровом уровне.

Базовый класс	Подкласс	Программа	Значение
05h	00h	00h	ОЗУ
05h	01h	00h	FLASH
05h	80h	00h	Другой контроллер памяти

Базовый класс 06h

Этот базовый класс определен для всех типов мостов. PCI мост - любой PCI прибор,
который отображает PCI ресурсы (память или ввод/вывод) от одной части устройства к другой. В этом классе выделяются несколько подклассов. Не определено никаких специальных программируемых интерфейсов на регистровом уровне.

Базовый класс	Подкласс	Программа	Значение
06h	00h	00h	Главный мост
06h	01h	00h	мост ISA
06h	02h	00h	мост EISA
06h	03h	00h	мост MC
06h	04h	00h	PCI-to-PCI мост
06h	05h	00h	мост PCMCIA
06h	80h	00h	Другое мостовое устройство

6.2.2. Управление устройством

Регистр Команды обеспечивает способность прибора генерировать циклы PCI и отвечать на них. Когда 0 записан в этот регистр, то прибор логически разъединен от PCI шины для всех доступов за исключением доступа конфигурации. Функциональная возможность всех приборов должна поддерживаться на этом базовом уровне. Отдельные биты в регистре Команды могут быть задействованы или нет в зависимости от функциональных возможностей приборов. Например, в том случае если в приборе не задействовано пространство ввода/вывода , то в 0 разряде регистра Команды не будет осуществляться регистрация данных. Как правило, используются устройства со всеми нулями в регистре Команды, но в разделе 6.4. объяснены некоторые исключения. Рисунок 6-2 показывает схему регистра, и Таблица 6-2 объясняет значения различных битов в регистре Команды.

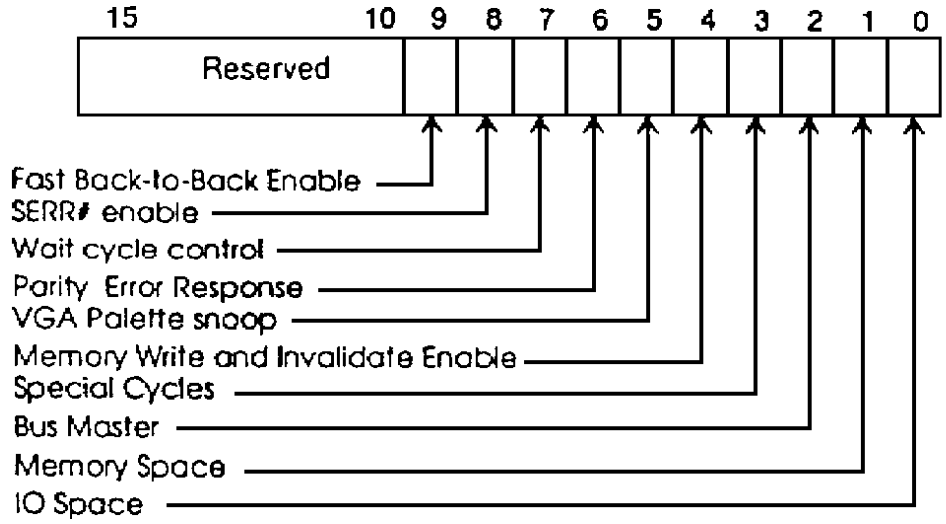


Рисунок 6-2: Схема регистра Команды

Таблица 6-2: Биты регистра Команды

Размещение битов	Описание
0	Управление откликом прибора на доступ к пространству ввода/вывода. Значение 0 запрещает отклик прибора. Значение1 разрешает прибору реагировать на доступ к пространству ввода/вывода.
1	Управление откликом прибора на доступ к пространству памяти. Значение 0 запрещает отклик прибора. Значение 1 разрешает прибору реагировать на доступ к пространству памяти.
2	Управление способностью прибора действовать как master на PCI шину. Значение 0 запрещает прибору доступ к PCI. Значение 1 позволяет прибору вести себя как master шины.
3	Управление функционированием прибора на Специальном Цикле. Значение 0 заставляет прибор игнорировать все Специальные Циклы. Значение 1 позволяет прибору контролировать Специальные Циклы.
4	Это вспомогательный бит для использования Write Memory и команды Invalidate. Когда этот бит равен 1, masters могут генерировать команду. Когда он равен 0, вместо этого должна использоваться Write Memory. Состояние после RST# - 0. Этот бит должен быть задействован master устройствами, которые могут генерировать Write Memory и команду Invalidate.
5	Этот бит контролирует как VGA совместимые приборы обрабатывают доступ к их регистрам палитры. Когда этот бит установлен, то допускается использование специальной палитры отслеживания поведения (то есть, прибор не должен отвечать). Когда бит не установлен, то прибор должен обработать доступы к палитре, как любые другие доступы. VGA совместимые приборы должны задействовать этот бит

(Продолжение следует)

Таблица 6-2: Биты регистра Команды (продолжение)

Размещение битов	Описание
6	Этот бит контролирует реакцию прибора на ошибки контроля по четности. Когда бит установлен, прибор возвращается к нормальному функционированию как только обнаружена ошибка контроля по четности. Когда бит не установлен, то прибор должен игнорировать любые ошибки контроля по четности, которые он обнаружил, и продолжить нормальную работу. Этот бит должен быть установлен в 0 после RST#. Приборы, которые проверяют контроль по четности, должны использовать этот бит. Приборы все еще нужны для генерирования контроля по четности даже если паритетный контроль запрещен.
7	Этот бит используется для контроля последовательности операций по обработке пошагового выполнения адреса / данных. Приборы, которые никогда не делают пошагового выполнения, должны аппаратно устанавливать этот бит в 0. Приборы, которые всегда делают пошаговое выполнение, должны аппаратно устанавливать этот бит в 1. Приборы, которые могут делать и то и другое, должны делать этот бит доступным по чтению / записи и инициализировать в 1 после RST#.
8	Это разрешающий бит для SERR# задающего устройства (драйвера). Значение 0 отключает SERR# драйвер. Значение 1, разрешает SERR# драйвер. После сброса системы бит устанавливается в 0. Все приборы, которые имеют SERR# выводы , должны использовать этот бит. Этот бит (и бит 6) должен сообщать адрес ошибки контроля по четности.
9	Это дополнительный бит контроля чтения/записи, который проверяет насколько быстро master может выполнять обратные транзакции различных приборов. Инициализация программного обеспечения устанавливает бит, если все объекты допускают возможность быстрой обратной транзакции. Значение 1 свидетельствует о том, что допустима быстрая обратная транзакция на различные объекты как описано в разделе 3.4.2. Значение 0 свидетельствует, что допустимы только быстрые обратные транзакции к прежнему объекту.
10-15	Зарезервированы

6.2.3. Состояние устройства

Регистр Состояния используется для протоколирования информации о состоянии PCI шины. Определение каждого из битов дано в Таблице 6-3, и схема регистра показана на Рисунке 6-3. Приборы могут не требовать задействования всех битов в зависимости от функциональных возможностей прибора. Например, прибор, который действует в качестве ведомого , но никогда не сообщает о своем аварийном прекращение работы, не использует бит 11.

Чтение этого регистра происходит нормально. Запись немного отличается, эти биты могут быть сброшены, но не установлены. Бит сбрасывается всякий раз, когда регистр записан, и данные в соответствующем бите равны 1. Например, чтобы очистить бит 14 и не воздействовать на все другие биты , запишите значение 0100_0000_0000_0000b в регистр.

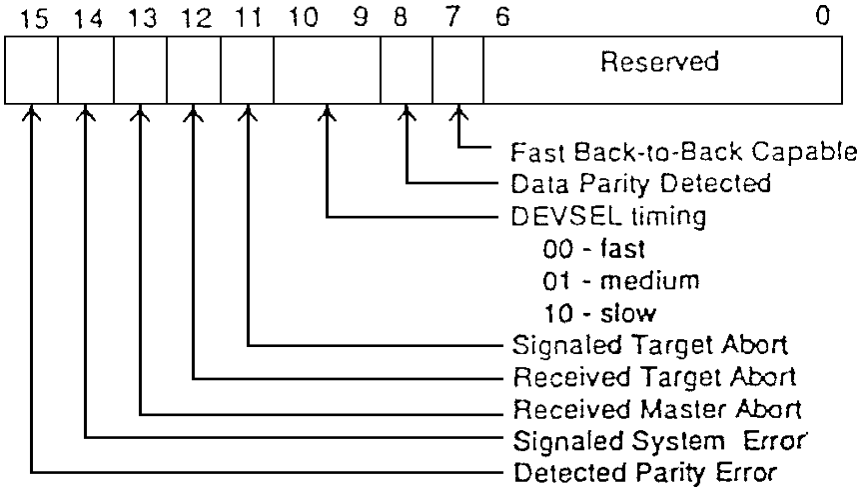


Рисунок 6-3: Схема регистра Состояния

Таблица 6-3: Биты регистра Состояния

Размещение битов	Описание
0-6	Зарезервированы
7	Вспомогательный бит, только для чтения ,указывает является или нет ведомый способным к принятию быстрой обратной транзакции, когда транзакции не для данного объекта. Этот бит может быть установлен в 1, если прибор может устанавливать эти транзакции, и должен быть установлен в 0 в противном случае. Смотрите раздел 3.4.2. , где приведено полное описание требований для установки этого бита.
8	Этот бит задействуется только мастерами шины. Он устанавливается, когда соблюдены три условия: l) агент шины заявляет о своих правах в рамках PERR# или следит за соблюдением этих прав; 2) агент, устанавливающий бит , действует как master шины при операциях , в которых произошла ошибка; 3) бит (регистр команды) установлен вследствие возникновения ошибки по четности
9-10	Эти биты кодируют синхронизацию DEVSEL#. Раздел 3.6.1 определяет три допустимых типа синхронизации для утверждения DEVSEL#. Быстрая закодирована как 00b, средняя - 01b, медленная - 10b (11b зарезервирован). Эти биты только для чтения и должны иметь индикацию минимального временного интервала, в течении которого прибор утверждает DEVSEL# для любой команды шины за исключением Записи Конфигурации и Чтения Конфигурации.
11	Этот бит должен быть установлен целевым прибором, когда он завершает транзакцию, с целевым аварийным прекращением работы. Приборы, которые никогда не сигнализируют целевое аварийное прекращение работы, не должны использовать этот бит.
12	Этот бит должен быть установлен master прибором, когда он завершает транзакцию, с целевым аварийным прекращением работы. Все master приборы должны использовать этот бит.
13	Этот бит должен быть установлен master прибором, когда транзакция (за исключением Специального Цикла) завершена с аварийным прекращением работы мастера. Все master приборы должны использовать этот бит.
14	Этот бит должен быть установлен прибором в том случае, когда им установлен SERR#. Приборы, которые никогда не устанавливают SERR#, не должны использовать этот бит.
15	Этот бит должен быть установлен прибором при обнаружении ошибки контроля по четности, даже если обработка ошибок контроля по четности запрещена (как управляется битом 6 в регистре Команды).

6.2.4. Смешанные функции

В этом разделе описаны регистры независимые от устройств и необходимые только для обеспечения перечисленных ниже функций.

Размер Кэша
Этот регистр определяет размер кэша системы в 32-разрядных словах. Этот регистр для чтения/записи должен использоваться мастер - устройствами, которые могут генерировать команды Write и Invalidate Памяти (Смотрите раздел 3.1.1). Приборы, участвующие в протоколе кэширования (Смотрите раздел 3.3) используют это поле, чтобы знать, когда повторить сигнал доступа в пределы кэша. Эти приборы могут игнорировать линии PCI поддерживающие кэш (SCONE и SBO#) когда этот регистр установлен в 0. Это поле должно быть установлено в 0 при сбросе.

Таймер Времени ожидания
Этот регистр определяет, в единицах часов шины PCI, значение Таймера задержки для этого мастера PCI шины (Смотрите раздел 3.4.4.1.). Этот регистр должен быть выполнен как перезаписываемый любым мастером в том случае, если возможна разбивка больше чем две фазы данных. Этот регистр может быть выполнен как только для чтения для устройств, которые разбивают данные на две или менее количество фаз, но фиксированное значение должно ограничиваться 16 или меньше. Типичной реализацией было бы формирование пяти старших битов (оставляя три младших только для чтения), приводящее к степени детализации таймера до восьми часов. При сбросе, регистр должен быть установлен в 0 (если программируемый).

Встроенный тест (BIST)
Этот вспомогательный регистр используется для управления и состояния BIST. Устройства, которые не поддерживают BIST, должны всегда возвращать значение 0 (т.е.. обрабатывать его как зарезервированный регистр). Устройство, которое вызывает BIST, не должно препятствовать нормальной работе PCI шины. Рисунок 6-4 показывает схему регистра, а Таблица 6-4 описывает биты в регистре.

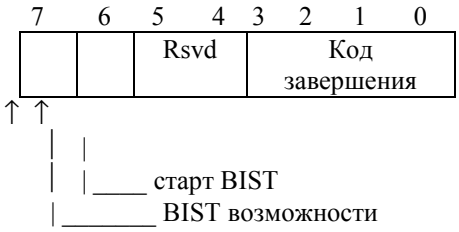


Рисунок 6-4: Схема регистра BIST

Таблица 6-4: биты регистра BIST

Размещение бит	Описание
7	Возвращает 1, если устройство поддерживает BIST. Возвращает 0, если устройство не поддерживает BIST.
6	Запишите 1, чтобы вызвать BIST. Устройство сбрасывает бит, когда BIST выполнен. Программное обеспечение должно отключать устройство, если BIST не завершен после 2 секунд.
5-4	Зарезервированный. Устройство возвращает 0.
3-0	Значение 0 свидетельствует о том, что устройство прошло тест. Ненулевые значения означают, что устройство потерпело неудачу. Коды отказа, зависящие от устройств, могут кодироваться в ненулевом значении в зависимости от типа устройства.

Линия Прерывания
Регистр линии Прерывания представляет собой 8-разрядный регистр, используемый для сообщения линии прерывания, направляющей информации. Этот регистр для чтения/записи и может использоваться любым устройством (или функцией устройства) который использует выводы прерывания. Программное обеспечение POST должно регистрировать информацию маршрутизации в этом регистре во время инициализации и конфигурирования системы.
Значение этого регистра сообщает, какой вход системы прерывания контроллера(-ов) соединен с выходом прерывания устройства. Задающие устройства и операционная система могут использовать эту информацию для определения приоритетной и векторной информации. Значение этого регистра зависит от архитектуры системы[1] .

Вывод Прерывания
Регистр вывода Прерывания сообщает, какие выводы прерывания использует устройство (или функция устройства). Значение 1 соответствует INTA#. Значение 2 соответствует INTB#. Значение 3 соответствует INTC#. Значение 4 соответствует INTD#. Устройства (или функции устройства) которые не используют вывод прерывания, должны поместить 0 в этот регистр. Этот регистр только для чтения.

MIN_GNT и MAX_LAT
Это байтовые регистры только для чтения и используются для определения устройств желающих установки значения Таймера времени ожидания. Для обоих регистров значение определяет период времени в единицах 1/4 микросекунды. Значение 0 указывают, что устройство не требует установки Таймеров времени ожидания.

MIN_GNT используется для определения длительности периода разбивки устройства, нуждающегося в принятии тактовой частоты 33 МГц. MAX_LAT используется устройством для определения частоты получения доступа к PCI шине.

Устройства должны определять значения, которые позволят им эффективно использовать PCI шину также как их внутренние ресурсы.

6.2.5. Базовые адреса

Одна из наиболее важных функций для установления наилучшей конфигурируемости и простоты использования - это способность перемещать PCI устройства в адресных пространствах. При включении питания системы устройство, независимое от программного обеспечения, должно быть способно определить, какое устройство присутствует, сформировать согласованную таблицу адресов, и определить имеет ли устройство расширенное ПЗУ . Каждая из этих областей рассматривается в следующих разделах.

6.2.5.1. Таблицы адресов

При включении питания программное обеспечение должно сформировать согласованную таблицу адресов перед подключением машины к рабочей системе. В связи с этим необходимо точно установить объем памяти системы, и сколько адресного пространства требуют в системе контроллеры ввода/вывода. После определения этой информации, программное обеспечение включения питания может точно разместить контроллеры ввода/вывода в правильных позициях и продолжать начальную загрузку системы. Основные регистры на схеме размещаются в заранее определенном заголовке пространства конфигурации.

[1] Для PC на основе x86 значения в этом регистре соответствуют числам IRQ (0-15) стандартной двойной 8259 конфигурации. Значение 255 определено как значение «неизвестного» или «несоединенного » контроллера прерывания. Значения от l5 до 255 зарезервированы.

Рисунок 6-5: Регистр базового адреса для памяти

Бит 0 во всех базовых регистрах только для чтения и использован, чтобы определить, отображает ли регистр в пространство ввода/вывода или память. Базовые регистры, которые отображают в пространство памяти, должны возвращать 0 в бите 0. Базовые регистры, которые отображают в пространство ввода/вывода, должны возвращать 1 в бите 0.

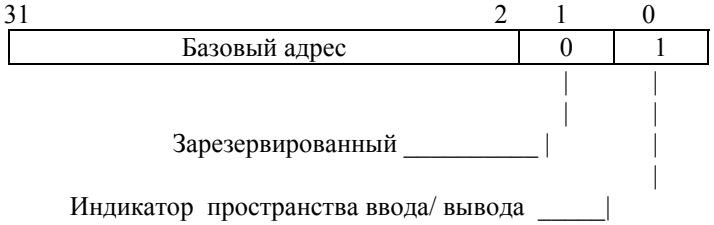


Рисунок 6-6: Регистр базового адреса для ввода - вывода

Базовые регистры, которые отображают в пространство ввода/вывода , всегда 32-битовые , где в бите 0 аппаратно определена 1, а бит 1 зарезервирован и должен возвращать 0 при чтении, другие биты используются, чтобы отобразить устройство в пространство ввода/вывода (см. Рисунок 6-6).

Базовые регистры, которые отображают в пространство памяти (Рисунок 6-5) могут быть 32 бита или 64 бита (для поддержки отображения в 64-разрядное адресное пространство) с битом 0, аппаратно установленным в 0. Для базовых регистров памяти, биты 2 и 1 имеют кодируемое значение как показано в Таблице 6-5. Бит 3 должен быть установлен в 1, если данные предопределены и сбрасываться в 0 в противном случае. Устройство может отмечать диапазон как предопределенный; если не имеется никаких побочных эффектов при чтении, устройство возвращается к считыванию данных независимо от байта разрешения и мост хоста может объединить записи процессора в этот блок[2] не вызывая помех. (см. раздел 3.2.3.) . Биты 0-3 - только для чтения.

Таблица 6-5: Кодирование битов 2/1

Биты 2/1	Значение
00	Базовый регистр 32-битовый и может отображаться в любое место 32-разрядного пространстве памяти.
01	Базовый регистр 32-битовый, но должен быть отображен в пространство памяти ниже 1М.
10	Базовый регистр 64-битовый и может отображаться в любое место 64-разрядного адресного пространства.
11	Зарезервированный

Число старших битов, которые устройство фактически устанавливает, зависит от того, сколько адресного пространства устройство может адресовать. Устройство, которое хочет адресное пространство памяти 1 МБ (использующее 32-разрядный регистр базового адреса) должно сформировать старшие 12 битов адресного регистра, аппаратно другие биты установлены в 0.

Программное обеспечение Включения питания может определять количество адресного пространства требуемое устройству для записи всех единиц в регистр и для чтения значений обратно. Устройство должно возвращать 0 во все незадействованные биты адреса , точно определяя требуемое адресное пространство.

Эта конструкция предполагает, что все задействованное адресное пространство питается от двух источников и располагается линейно. Устройства могут задействовать больше адресного пространства чем требуется, но декодирование для 4КБ пространства Памяти и для 256-байтового ввода/вывода предложено для устройств, которые нуждаются в меньшем их количестве. Устройства, которые потребляют больше адресного пространства чем они используют, не требуются, чтобы ответить на неиспользуемую часть того адресного пространства.

Шесть DWORD ячеек распределены под регистры Базового адреса стартующие со смещения 10h пространства конфигурации. Первый регистр Базового адреса всегда размещается со смещения 10h. Второй регистр может быть в смещении 14h или 18h в зависимости от размера первого. Смещения последующих регистров Базового адреса определены размером предыдущих регистров Базового адреса.

Типичное устройство требует одного блока памяти для функций управления. Некоторые графические устройства могут использовать два блока, один для функций управления и другой для буфера изображения. Устройство, которое хочет отображать функции управления, и в память и в пространство ввода/вывода одновременно, должно использовать два базовых регистра (один для Памяти, один для ввода/вывода). Драйвер для этого устройства может использовать только одно пространство и в этом случае другое пространство будет неиспользуемым. Устройства должны всегда отображать функции управления в пространство памяти.

[2] Все устройства, которые имеют блок памяти, работающий как обычная память, но не участвующий в кэшировании протокола PCI, должны помечать этот блок как желательный. Примером блока памяти, который должен быть помечен как желательный, может служить линейный буфер изображения в графическом устройстве.

6.2.5.2. Регистр базового адреса расширенного ROM

Некоторые PCI устройства, особенно, которые предназначены для использования в дополнительных модулях в PC архитектуре, требуют локальных EPROM для расширения ROM (смотрите раздел 6.3. для определения содержания ROM). 4-байтовый регистр в смещении 30h пространства конфигурации определен для хранения базового адреса и информации о размере этого расширенного ROM. На рисунке 6-7 показана организация этого слова. Этот регистр функционирует точно также как 32-разрядный регистр Базового адреса за исключением того, что кодирование (и использование) младших битов различно. Старшие 21 бит соответствуют старшим 21 битам базового адреса расширенного ROM. Число битов (из этих 21), которые устройство фактически использует, зависит от типа выравнивания, которое поддерживает устройство. Например, устройство, которое позволяет расширенному ROM быть отображенным в пределах любых 64КБ, должно использовать старшие 16 битов регистра, оставляя младшие 5 (из этих 21) аппаратно установленными в 0. Устройства, которые поддерживают расширенную ROM должны использовать этот регистр.
Программное обеспечение конфигурации , функционирующее независимо от устройства может определять какого типа выравнивание поддерживает устройство, путем занесения всех единиц в адресную часть регистра и последующего чтения значения обратно. Устройство возвращает 0 во всех незадействованных битах, придерживаясь заданного типа выравнивания.

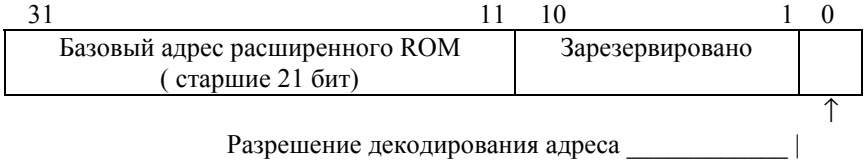


Рисунок 6-7: Схема расположения регистра базового адреса расширенного ROM

Бит 0 в регистре используется для проверки , имеет или нет устройство доступ к расширенному ROM. Когда этот бит сброшен, то для устройств адресное пространство расширенного ROM запрещено. Когда бит установлен, декодирование адреса допускается, используя параметры другой части базового регистра. Это позволяет использовать устройство с или без расширенного ROM в зависимости от конфигурации системы. Бит Пространства Памяти в регистре Команды имеет старшинство над битом разрешения расширенного ROM. Устройство должно ответить на доступ к расширенному ROM только в том случае , если и бит Пространства Памяти и бит Базового адреса расширенного ROM установлены в 1.
Для того, чтобы минимизировать число декодеров адреса, необходимых устройству, можно совместить использование декодера между регистром Базового адреса расширенного ROM и другими регистрами Базового адреса. Когда разрешено декодирование расширенного ROM, декодер используется для доступа к расширенному ROM и независимое от устройства программное обеспечение не должно обращаться к устройству через любые другие регистры базового адреса.

6.2.5.3. Дополнительная память

Механизм для обработки дополнительной памяти будет определен в следующем выпуске спецификации. Будет проведен сравнительный анализ значимости нового Типа Заголовка и регистров конфигурации дополнительной памяти. Эти описания позволят автоматически определять размеры и конфигурацию дополнительной памяти.

6.3. Расширенные PCI ROM

PCI спецификация обеспечивает механизм, при котором устройства могут обеспечивать код расширенного ROM, который может быть выполнен для устройств со специфической инициализацией и, возможно, функцией начальной загрузки системы (Смотрите раздел 6.2.5.2.). Этот механизм позволяет ROM содержать несколько различных образов для приспособления различных машин и архитектур процессора. Эта раздел содержит необходимую информацию и схему расположения кода образов в расширенном ROM. Обратите внимание, что PCI устройства поддерживающие расширенную ROM должны допускать, что ROM будет обращаться к любым разрешенным комбинациям байта. Это значит, что должен поддерживаться доступ DWORD к расширенному ROM.

Информация в ROM располагается так, чтобы быть совместимой с существующими Intel x86 заголовками расширенного ROM для ISA, EISA, и MC адаптеров, но она должна также поддерживать другие машинные архитектуры. Информация, доступная в заголовке была расширена так, чтобы более оптимально использовать функции, обеспеченные адаптером и свести к минимуму объем пространства памяти используемое во время выполнения кода расширенного ROM.

Информация заголовка расширенного PCI ROM поддерживает следующие функции:

- Длина кода обеспечивает идентификацию общего непрерывного адресного пространства, необходимого отображающему устройству PCI ROM при инициализации.
- Индикатор идентифицирует тип выполняемого или интерпретируемого кода , который существует в адресном пространстве ROM в каждом образе ROM.
- В ROM задана проверка уровня для кода и данных .
- Необходим указатель на Важнейшие Данные Изделия о устройстве.
- Включены в ROM также ID производителя и ID устройства поддерживающего PCI устройство.

Основное различие в модели использования между расширенным PCI ROM и стандартом ISA, EISA, и MC ROM в том , что код ROM никогда не выполняется просто так. Он всегда копируется из ROM в ОПЕРАТИВНУЮ ПАМЯТЬ и выполняется из ОПЕРАТИВНОЙ ПАМЯТИ. Это позволяет динамически устанавливать размеры кода (для инициализации и во время выполнения) и улучшает скоростные показатели при выполнении исполняемого кода.

6.3.1. Содержание расширенных PCI ROM

Расширенные PCI ROM могут содержать код (выполняемый или интерпретирующий) для различных архитектур процессора. Это может быть реализовано в одиночном физическом ROM, который может содержать столько образов кода, сколько нужно для различных систем и архитектур процессора (см. Рисунок 6-8). Каждый образ должен начинаться на 512-байтовой границе и должен содержать заголовок расширенного PCI ROM. Начальная точка каждого образа зависит от размера предыдущих образов. Последний образ в ROM имеет специальное кодирование в заголовке, чтобы идентифицировать его как последний образ.

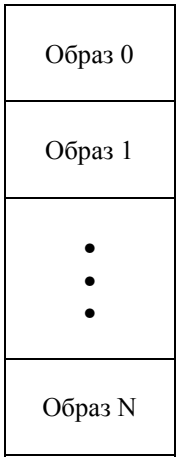


Рисунок 6-8: Структура расширенной платы ROM PCI

6.3.1.1. Формат заголовка расширенного ROM PCI

Информация, требуемая в каждом образе ROM делится на две различные области. Одна область, заголовок ROM, должна размещаться в начале образа ROM. Вторая область, Структура Данных PCI должна быть размещена в первых 64КБ образа. Формат для заголовка расширенного ROM PCI дан ниже. Смещение - шестнадцатеричный номер начала образа, а длина каждого поля дана в байтах.

Дополнения для заголовка расширенного ROM PCI и/или Структуры Данных PCI могут быть определены специфичностью архитектуры системы. Дополнения для PC-AT совместимых систем описаны в разделе 6.3.3.

Смещение	Длина	Значение	Описание
0h	1	55h	Сигнатура ROM, байт 1
1h	1	AAh	Сигнатура ROM, байт 2
2h-17h	16h	vv	Зарезервированы (уникальные данные архитектуры процессора)
18h-19h	2	vv	Указатель на Структуру Данных PCI

ROM Signature — Сигнатура ROM представляет собой двух-байтовое поле, содержащее 55h в первом байте, и AAh во втором байте. Эта сигнатура должна быть первыми двумя байтами адресного пространства ROM для каждого образа ROM.

Указатель на Структуру Данных PCI — указатель на Структуру Данных PCI представляет собой двух-байтовый указатель в формате little endian, который указывает на Структуру Данных PCI. Точкой отсчета для этого указателя является начало образа ROM.

6.3.1.2. Формат структуры данных PCI

Структура данных PCI должна быть размещена в первых 64КБ образа ROM и должна быть выровнена по DWORD. Структура Данных PCI содержит следующую информацию:

Смещение	Длина	Описание
0	4	Сигнатура, последовательность «PCIR»
4	2	Идентификация изготовителя
6	2	Идентификация устройства
8	2	Указатель на Важнейшие Данные Изделия
A	2	Длина Структуры Данных PCI
C	1	Обновление Структуры Данных PCI
D	3	Код Класса
10	2	Длина образа
12	2	Уровень обновления кода / данных
14	1	Тип кода
15	1	Индикатор
16	2	Зарезервирован

Сигнатура Эти четыре байта обеспечивает уникальную сигнатуру для Структуры Данных PCI. Последовательность "PCIR" - это сигнатура с нахождением "P" в смещении 0, "C" в смещении 1, и т.д.

Идентификация изготовителя Поле идентификации изготовителя - это16-разрядное поле с тем же описанием как и поле идентификации изготовителя в пространстве конфигурации для этого устройства.

Идентификация Устройства Поле идентификации устройства - это 16-разрядное поле с тем же описанием как и поле идентификации устройства в пространстве конфигурации для этого устройства.

Указатель на Важнейшие Данные Изделия Указатель на Важнейшие Данные Изделия (VPD) - это 16-разрядное поле со смещением от начала образа ROM и указывает на VPD. Этот поле в формате little-endian. VPD должен быть размещен с первых 64КБ образа ROM. Значение 0 не указывает ни на какие Важнейшие Данные Изделия находящиеся в образе ROM. Содержание структуры VPD будет описано в будущем издании спецификации.

Длина Структуры Данных PCI Длина Структуры Данных PCI - это 16-разрядное поле, которое определяет длину структуры данных от начала структуры данных (первый байт поля Сигнатуры). Это поле формата little- - endian и измеряется в байтах.

Обновление Структуры Данных PCI Поле обновления Структуры Данных PCI - это 8-разрядное поле идентифицирует уровень обновления структуры данных. Этот уровень обновления - 0.

Код Класса Поле кода класса - это 24-разрядное поле имеет тот же самый формат и поля , что и поле кода класса в пространстве конфигурации для этого устройства.

Длина образа — Поле длины образа - это двухбайтовое поле, которое содержит длину образа. Это поле в формате little-endian, и значение в блоках 512 байт.

Уровень обновления — Поле уровня обновления - это двух-байтовое поле, которое содержит уровень обновления кода в образе ROM.

Тип кода — Поле типа кода - это 1-байтовое поле , которое идентифицирует тип кода, содержащийся в этой области ROM. Код может быть двоичным исполняемым для специфичного процессора и архитектуры системы или интерпретирующим кодом. Определены следующие типы кода:

Тип	Описание
0	Intel x86, PC-AT совместимый
1	OPENBOOT стандарт для PCI[3]
2-FF	Зарезервированы

Индикатор — Бит 7 в этом поле сообщает, является или нет образ последним в ROM. Значение 1 указывает « последний образ » , значение 0 указывает, что следует другой образ. Биты 0-6 зарезервированы.

6.3.2. Листинг процедуры теста при включении питания (POST)

Главным образом, системный код POST обрабатывает дополнительные PCI устройства тождественно тем, которые впаяны на материнскую плату. Существует одно исключение - обработка расширенного ROM. Код POST обнаруживает выбор ROM в два шага. Сначала код определяет, установило ли устройство регистр Базового адреса расширенного ROM в пространстве конфигурации. Если регистр установлен, то POST должен отобразить и разрешить ROM в неиспользуемую область адресного пространства, и проверить первые два байта для сигнатуры AA55h. Если они найдены, то имеется представленный ROM, иначе никакой ROM не присоединен к устройству.

Если ROM присоединен, то POST должен искать образ ROM, который имеет соответствующий тип кода и согласованные поля ID изготовителя и ID устройства, соответствующие полям в устройстве.

После нахождения соответствующего образа, POST копирует соответствующее количество данных в оперативную память. Затем выполняется код инициализации устройства. Определение соответствующего количества копируемых данных , и как выполнять код инициализации устройства будет зависеть от типа кода для поля.

[3] OPENBOOT - это архитектура процессора и системы независимого стандарта для специализированного устройства, имеющего дело с кодом выбора ROM. Документация по OPENBOOT доступна в IEEE Draft Std 75iD7, за 4 января 1993.

6.3.3. PC - совместимые расширения ROM

Этот раздел описывает дальнейшие требования на образы ROM и по обработке образов ROM, которые используются в PC-совместимых системах. Это относится к любому образу характеризуемому Intel x86, совместимому с PC-AT в поле типа кода структуры данных PCI, и любой платформы, которая является совместимой с PC.

6.3.3.1. Заголовок расширений ROM

Стандартный заголовок для образов расширения ROM PCI немного расширен для PC - совместимости. Были добавлены два поля, одно в смещении 02h обеспечивает размер инициализации для образа. Смещение 03h - точка входа для функции INIT расширения ROM.

Смещение	Длина	Величина	Описание
0h	1	55h	Байт 1 сигнатуры ROM
1h	1	AAh	Байт 2 сигнатуры ROM
2h	1	vv	Размер инициализации - размер кода в 512-байтовых модулях.
3h	4	vv	Точка входа для функции INIT. POST делает FAR CALL к этой ячейке.
7h-17h	11h	vv	Зарезервированы (уникальные данные приложения)
18h-19h	2	vv	Указатель на Структуру Данных PCI

6.3.3.1.1. Обновления POST кода

POST код в этих системах копирует число байтов, определенных нициализацией поля размера в ОПЕРАТИВНУЮ ПАМßТЬ, и затем вызывает функцию INIT, где точкой входа является смещение 03h. POST код требуется для оставления области ОПЕРАТИВНОЙ ПАМßТИ , куда код расширения ROM был скопирован для возврата перезаписи до окончания функции INIT. Это позволяет коду INIT сохранять некоторые статические данные в области ОПЕРАТИВНОЙ ПАМßТИ и корректировать размер во время выполнения кода так, чтобы он потреблял меньшее количество пространства, во время работы системы.

PC -совместимый специфичный набор шагов для системного кода POST при обработке каждого расширения ROM:

1. Скопируйте из ROM в ОПЕРАТИВНУЮ ПАМЯТЬ количество байтов, определенных размером Инициализации.
2. Оставьте область ОПЕРАТИВНОЙ ПАМßТИ перезаписываемой и вызовите функцию INIT
3. Используйте байт в смещении 02h (который, возможно, изменялся) для определения количества памяти используемой во времени выполнения.

Перед начальной загрузкой системы, POST код должен подготовить область ОПЕРАТИВНОЙ ПАМЯТИ, содержащую код расширения ROM только для чтения.

POST код должен обрабатывать VGA устройства с расширением ROM специальным способом. Расширенная БАЗОВАЯ СИСТЕМА ВВОДА-ВЫВОДА VGA устройства должна быть скопирована с 0C0000h. VGA устройства могут быть идентифицированы при исследовании поля Кода Класса в пространстве конфигурации устройства

6.3.3.1.2. Функция INIT расширений

Совместимое с PC расширение ROM содержит функцию INIT, которая является ответственной за инициализацию устройства ввод/вывода и подготовку операций во время работы. Функции INIT в расширении ROM PCI имеют некоторые расширенные возможности , так как область ОПЕРАТИВНОЙ ПАМЯТИ, где размещен код, оставлена перезаписываемой, во время выполнения функции INIT.
Функция INIT может сохранять статические параметры в области ОПЕРАТИВНОЙ ПАМЯТИ в течение функции INIT. Эти данные могут затем использоваться БАЗОВОЙ СИСТЕМОЙ ВВОДА-ВЫВОДА во время работы или драйверами устройства. Эта область ОПЕРАТИВНОЙ ПАМЯТИ не будет перезаписываема во время работы.
Функция INIT может также корректировать количество ОПЕРАТИВНОЙ ПАМЯТИ, которую она потребляет во время работы. Это сделано для модификации размера байта в смещении 02h в образе. Это помогает сохранять
ограниченный ресурс памяти в расширенной области ROM (0C0000h-0DFFFFh).
Например, расширение ROM устройства может требовать 24 КБ для инициализации и кода во время работы, но имеется только 8КБ для кода во время работы. Образ в ROM должен показать размер из 24КБ, так чтобы код POST копировал всю вещь в ОПЕРАТИВНУЮ ПАМßТЬ. Затем, во время выполнения функции INIT, она может корректировать размер байта до 8КБ. При возвращении из функции INIT, POST видит, что размер во время выполнения - 8 КБ и может копировать следующее расширение БАЗОВОЙ СИСТЕМЫ ВВОДА-ВЫВОДА в нужное место.
Функция INIT гарантирует, что контрольная сумма размера образа правильна. Если функция INIT изменяет область ОПЕРАТИВНОЙ ПАМßТИ , то новая контрольная сумма должна быть вычислена и сохранена в образе.
При входе, функция INIT передает три параметра: номер шины, номер устройства и функциональный номер устройства, которое обеспечило расширение ROM. Эти параметры могут использоваться для обращения к инициализируемому устройству. Они переданы в x86 регистры, [AH] содержит номер шины, старшие пять бит [AL] содержат номер устройства и младшие три бита [AL] содержат функциональный номер.

6.3.3.1.3. Структура образа

PC - совместимый образ имеет три блока соответствующие этим: блок времени выполнения, блок инициализации и блок отображения. Блок отображения - результирующий блок отображения и он должен быть больше или равен блоку инициализации.
Блок инициализации определяет объем образа, который содержит, и инициализацию и код времени выполнения. Это объем данных, которые код POST копирует в ОПЕРАТИВНУЮ ПАМЯТЬ перед выполнением подпрограммы инициализации. Блок инициализации должен быть больше или равен блоку времени выполнения. Данные инициализации, скопированные в ОПЕРАТИВНУЮ ПАМЯТЬ, должны иметь контрольную сумму 0 (используя стандартный алгоритм).
Блок времени выполнения определяет объем образа, который содержит код выполнения. Это количество данных код POST оставит в ОПЕРАТИВНОЙ ПАМЯТИ во время работы системы . И снова, этот объем образа должен иметь контрольную сумму 0.

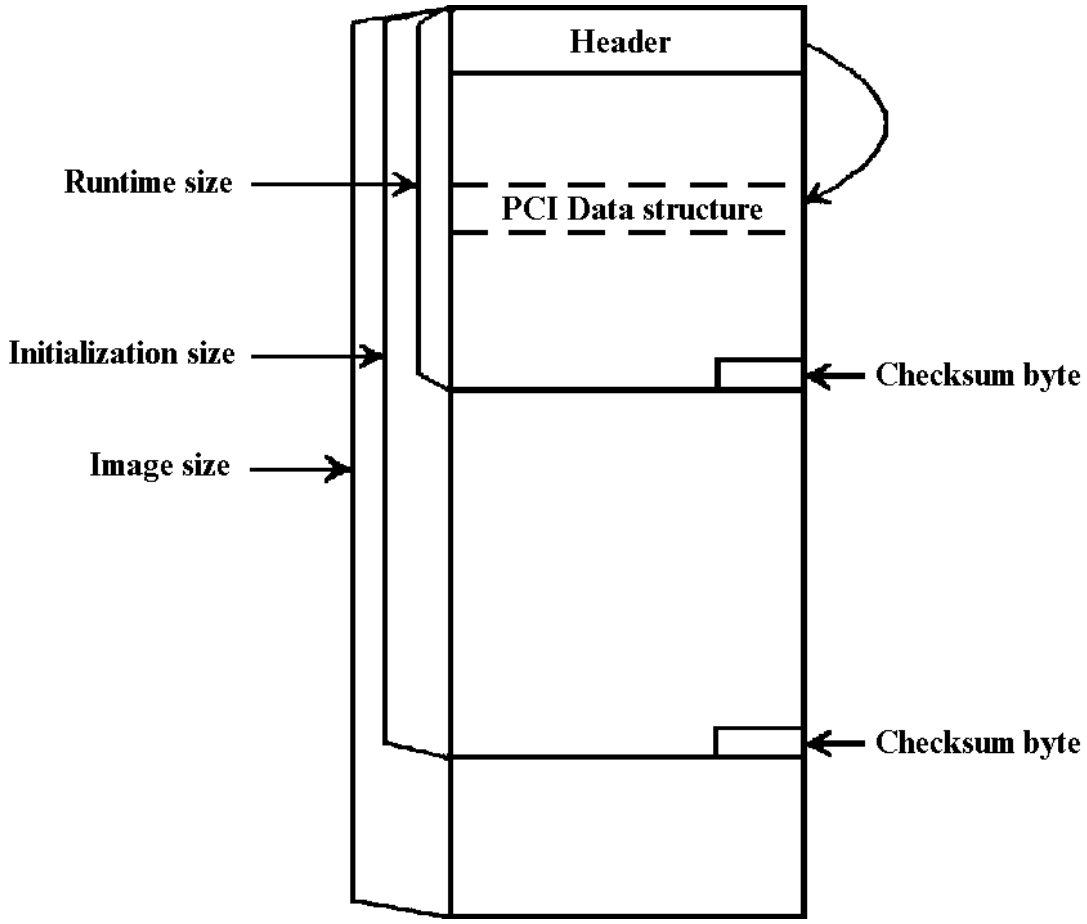


Рисунок 6-9: Типичная схема расположения образа

Эта структура данных PCI должна содержаться внутри блока времени выполнения образа (если таковой имеется), иначе она должна содержаться внутри блока инициализации. Рисунок 6-9 показывает типичную схему расположения образа в расширении ROM.

6.3.4. Драйверы устройств

Имеются две характеристики PCI устройств, которые отличают драйверы PCI устройства от стандартных существующих драйверов устройства. Первая характеристика - это то, что PCI устройства являются переместимыми (то есть, не аппаратными) в адресных пространствах. Драйверы PCI устройства (и другое программное обеспечение конфигурации) должны использовать информацию о расположении, сохраненную в регистре пространства конфигурации устройства, чтобы определить, где устройство было отображено. Это также применяется для определения используемой линии прерывания.
Вторая характеристика - это то, что прерывания PCI являются общедоступными. Драйверы PCI устройства требуются для поддержки общедоступных прерываний, так как система соединяет больше чем одно устройство с одной линией прерывания. Точный метод общедоступного прерывания специфичен для операционной системы и здесь не разбирается.
Некоторые системы не могут гарантировать, что данные поступили в оперативную память прежде, чем прерывания выставлены ЦЕНТРАЛЬНОМУ ПРОЦЕССОРУ. Если они не обработаны правильно, это может привести к проблемам согласованности данных (потери данных). Эта ситуация наиболее часто происходит при установке буферов регистрации в мостах между PCI шиной и другими шинами (см. обсуждение по мостам в Руководстве по Проектированию систем PCI).

Имеются три способа , при которых данные и прерывания согласованно могут быть гарантированы:

1. Аппаратные средства системы могут гарантировать, что регистрирующие буфера заполнятся прежде, чем прерывания будут выставлены процессору.
2. Устройство, передающее сигналы прерывания может выполнять чтение данных только что записанных перед передачей сигналов прерывания. Это заставляет регистрирующие буфера быть заполненными.
3. Драйвер устройства может выполнять чтение любого регистра в устройстве перед доступом к данным, записанным устройством. Это причины чтения регистрирующих буферов, которые нужно заполнить.

Драйвер устройства в конечном счете гарантирует согласованность прерываний и данных, обеспечивая по крайней мере один из трех методов, описанных выше, которые используются в системе. Это означает, что драйвер устройства должен делать метод 3, если он точно не знает, что метод 2 выполнен устройством, или он информирован (некоторыми внешними средствами) что метод 1 выполнен аппаратными средствами системы.

6.4. Сброс системы

После сброса системы, процессор(ы) должен иметь доступ к коду начальной загрузки и любым устройствам, необходимым для загрузки машины. В зависимости от архитектуры системы, мосты могут нуждаться в появлении сигнала разрешения для передачи этого доступа к удаленной шине[4] .

Точно так же устройства на PCI могут нуждаться в появлении сигнала разрешения, для распознания фиксированных адресов, чтобы поддержать последовательность начальной загрузки в архитектуре системы. Такие устройства требуются для поддержки функции запирания регистра Команды, описанного в разделе 6.2.2 .. Они должны также обеспечить механизм (вызываемый через пространство конфигурации), чтобы заново разрешить распознавание фиксированных адресов.

[4] Требования , специфичные для PC - совместимых архитектур, могут быть найдены в *Руководстве по Проектированию систем PCI.*

Приложение A

Сообщения Специального Цикла

В этом приложении определены кодировки сообщения Специального цикла. Текущий список определяет мало кодировок, но ожидается рост. Зарезервированные кодировки не должны использоваться.

Сообщения кодировок

[15::0] Тип сообщения

0C00h	ЗАКРЫТИЕ СИСТЕМЫ
0001h	ОСТАНОВ
0002h	x86 архитектура
0003h	Зарезервирован
.....	
FFFFh	Зарезервирован

ЗАКРЫТИЕ СИСТЕМЫ - широковещательное сообщение , указывающее что процессор вступает в режим закрытия системы.

ОСТАНОВ - широковещательное сообщение от процессора, указывающее что он выполнил команду останова.

Кодирование X86 архитектуры - обобщенное кодирование для использования x86 процессорами и chipsets. [31:: l 6] определяет специфичное значение сообщения специального цикла. Специфичные значения определены Корпорацией Intel и находятся в документации по изделию.

Использование специальных кодировок

Использование или генерирование кодировок специфичных для архитектуры не ограничено для инициатора запроса кодирования. Специфичные кодировки могут использоваться любым изготовителем в любой системе. Эти кодировки позволяют системе налаживать специфичную связь между PCI устройствами скооперированными для целей, которые не могут быть обработаны со стандартными типами цикла передачи данных.

Будущие кодировки

Члены компаний PCI SIG, которые требуют специального кодирования вне диапазона определенных в настоящее время кодировок должны послать письменный запрос в Комитет Управления PCI SIG. Комитет Управления распределит и определит кодировки специального цикла, основанные на информации при определении потребностей использования инициатором запроса и будущего изделия или направления приложения.

Приложение B

Конечные автоматы

Это приложение описывает ведущие и целевые конечные автоматы. Эти конечные автоматы только для иллюстративных целей и включены, чтобы помогать иллюстрировать PCI протокол. Фактические реализации не должны непосредственно использовать эти конечные автоматы. Машины, как полагают, являются правильными, однако, если конфликт существует между спецификацией и конечными автоматами, спецификация имеет старшинство.
Конечные автоматы используют три типа переменных; состояния, PCI сигналы и внутренние сигналы. Они отличаются от друг друга:

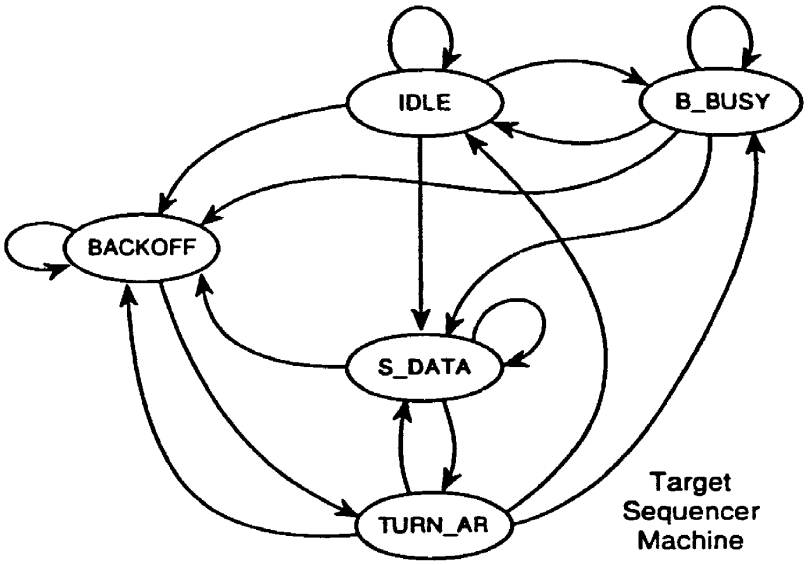
State in a state machine	= STATE
PCI signal	= SIGNAL
Internal signal	= Signal

Конечные автоматы не имеют задержек из состояния ввода, пока сигналы генерируются и доступны для использования в машине. Все сигналы PCI запираются на фронте CLK.
Конечные автоматы поддерживают некоторые опции (но не все) обсужденные в PCI спецификации. Обсуждение каждого состояния и иллюстрируемых опций будет следовать за определением каждого конечного автомата.
Интерфейс шины состоит из двух частей. Первая - bus sequencer, который выполняет фактическую операцию шины. Вторая часть - внутреннее или аппаратное приложение. В ведущем - внутреннее приложение генерирует транзакцию и обеспечивает адрес, данные, команду, byte enables и длину передачи. Оно также ответственно за адрес, когда транзакция повторяется. В адресате - внутреннее приложение определяет, когда транзакция прекращена. Программа упорядочения (sequencer) выполняет операцию шины как запрошено и гарантирует, что PCI протокол не нарушен. Обратите внимание, что адресат осуществляет блокировку ресурса.
Уравнения конечного автомата принимают логическую операцию, где "*" является функцией AND и имеет старшинство над "+", который является функцией OR. Круглые скобки имеют старшинство над обоими. Символ "!" используется для отрицания (NOT) переменных. В уравнениях конечного автомата, PCI SIGNAL's представляют фактическое состояние сигнала на PCI шине. Истинные сигналы низкого уровня будут истинны или установлены, когда они появляются как !SIGNAL# и ложны или сняты, когда они появляются как SIGNAL#. Истинные сигналы высокого уровня будут истинны или установлены, когда они появляются как SIGNAL и ложны или сняты, когда они появляются как !SIGNAL. Внутренние сигналы будут истинны,

когда они появляются как Signal и ложны, когда они появляются как !Signal. Несколько выходов допустимы в уравнении, когда используют символ "==", для ссылки на предыдущее состояние. Например,

OE[PAR] == (S_DATA * !TRDY# * (cmd=read)]

Это указывает, что буфер вывода для PAR разрешен, когда предыдущее состояние S_DATA и TRDY# установлен и когда транзакция чтения. Первый конечный автомат - для адресата, второй - для ведущего. Должно приниматься предосторожность, когда агент является и ведущим и адресатом. Каждый должен иметь собственный конечный автомат, который может функционировать независимо от другого, чтобы избежать тупиков. Это означает, что на целевой конечный автомат нельзя воздействовать ведущим конечным автоматом. Хотя они имеют подобные состояния, они не могут встраиваться в одиночную машину.



IDLE или TURN_AR условие неактивности или завершенная транзакция на шине

goto IDLE	if FRAME#
goto B_BUSY	if !FRAME# * !Hit
goto S_DATA	if !FRAME# * Hit * (!Term + Term * Ready) * (FREE + LOCKED * LOCK#)
goto BACKOFF	if !FRAME# * Hit * (Term * !Ready + LOCKED * !LOCK#)

B_BUSY - Не включается в текущую транзакцию.

goto B_BUSY	if (!FRAME# + !IRDY#) * !Hit
goto IDLE	if FRAME#
goto S_DATA	if (!FRAME# + !IRDY#) * Hit* (!Term + Term * Ready) * (FREE + LOCKED * L_lock#)
goto BACKOFF	if (!FRAME# + !IRDY#) * Hit * (Term * !Ready + LOCKED * L_lock#)

S_DATA - Агент принял запрос и ответит.

goto S_DATA	if !FRAME# * !STOP# * !TRDY# * IRDY# + !FRAME# * STOP# + FRAME# * TRDY# * STOP#
goto BACKOFF	if !FRAME# * !STOP# * (TRDY# +!IRDY#)
goto TURN_AR	if FRAME# * (!TRDY# + !STOP#)

BACKOFF - Агент занят и неспособен ответить в это время.

goto BACKOFF	if !FRAME#
goto TURN_AR	if FRAME#

Target LOCK Machine

FREE - Агент свободен, чтобы ответить на все транзакции.

goto LOCKED	if !FRAME# * LOCK# * Hit * (IDLE + TURN_AR) + L_lock# * Hit * B_BUSY)
goto FREE	if ELSE

LOCKED - Агент не будет отвечать, если LOCK# не снят в течение фазы адреса.

goto FREE	if FRAME# * LOCK#
goto LOCKED	if ELSE

Адресат транзакции ответственен за управление следующими сигналами[1]:

OE[AD[31::00]]	= S_DATA * Tar_dly * (cmd = read)
OE[TRDY#]	= BACKOFF + S_DATA + TURN_AR
OE[STOP#]	= BACKOFF + S_DATA + TURN_AR
OE[DEVSEL#]	= BACKOFF + S_DATA + TURN_AR
OE[PAR]	== S_DATA * !TRDY * (cmd = read)[2]
OE[PERR#]	= (cmd = write) * !IRDY# * (delayed by two clocks)
TRDY#	= !(Ready*!T_abort*S_DATA*(cmd=write+cmd=read*Tar_dly))
STOP#	= ![BACKOFF + S_DATA * (T_abort + Term) * (cmd = write + cmd = read * Tar_dly)]
DEVSEL#	= ![(BACKOFF + S_DATA) * !T_abort]
PAR	= even parity across AD[31::00] and C/BE[3::0] lines
PERR#	= R_perr

[1] Когда адресат поддерживает команды Special Cycle, дополнительный терм должен быть включен, чтобы не допустить в течение транзакции Special Cycle.

[2] OE[PAR] = OE[AD[31::00]] задержанное на один такт.

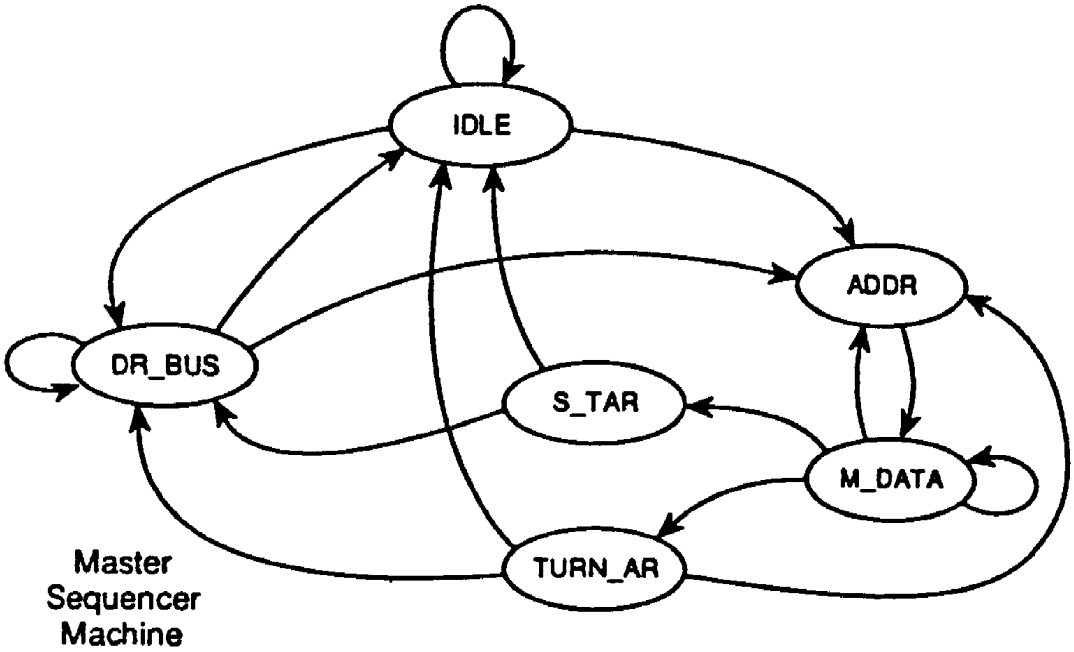
Определения

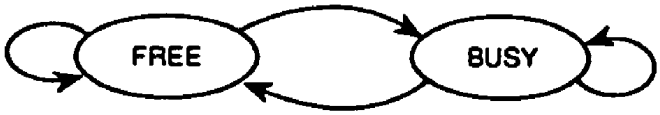
Внутренние входы конечного автомата

Эти сигналы между target bus sequencer и backend. Они указывают, как bus sequencer должен ответить на текущую операцию шины.

Hit	= Попасть на декодирование адреса.
Dt	= Devsel таймер истекает и DEVSEL# не установлен.
T-abort	= Адресат в ошибочном состоянии и требует, остановки текущей транзакции.
Term	= Завершить транзакцию. (Внутренний конфликт или >n состояний ожидания).
Ready	= Готов к передаче данных.
L_lock#	= Запирающая (в течение фазы адреса) версия LOCK#.
Tar_dly	= Turn around задержка, требуемая только для состояния с нулевым временем ожидания декодирования
R_perr	= сообщение об ошибке четности - импульс, продолжительностью в один такт PCI

Следующие абзацы обсуждают каждое состояние и описывают, какие уравнения могут быть удалены, если некоторые из PCI опций не реализованы.
IDLE и TURN-AR являются двумя отдельными состояниями конечного автомата, но объединены здесь, потому что переходы состояния те же самые для обоих состояний. Они выполнены как отдельные состояния, так как активные сигналы должны быть сняты перед установкой адресата в третье состояние.
Если адресат не может делать декодирование адреса за один цикл, путь из IDLE в S_DATA может быть удален. Адресат резонно требует путь от TURN_AR состояния в S_DATA и B_BUSY для back-to-back операций шины. Адресат должен быть способен декодировать back-to-back транзакции.
B_BUSY является состоянием, где агент ждет текущую транзакцию для завершения и возврата шины к простою. Это состояние полезно для устройства, которое медленно декодируют адрес или выполняет subtractive декодирование. Если адресат не делает ни одной из этих двух опций, путь к S_DATA и BACKOFF может быть удален. Терм "!Hit" также может быть удален из уравнения B_BUSY. Это сокращает состояние ожидания для завершения текущей транзакции шины.
S_DATA - состояние, целью передачи данных и нет необязательных уравнений.
BACKOFF - адресат идет после установки STOP# и ждет, когда ведущий снимет FRAME#.
FREE и LOCKED отнесятся к состоянию адресата относительно операции блокировки. Если адресат не реализует LOCK#, эти состояния не требуются. FREE указывает, когда агент может принимать любой запрос, когда он адресат. Если LOCKED, адресат повторит любой запрос, когда он не адресат, если LOCK# снят во время фазы адреса. Агент отмечает себя блокированным всякий раз, когда он адресат транзакции и LOCK# снят во время фазы адреса. Это небольшое перепутывание для адресата, чтобы блокировать себя на транзакции, которая не блокированная. Однако, с точки зрения реализации, это простой механизм, который использует комбинаторную логику и всегда работает. Устройство разблокирует себя в конце транзакции, когда обнаружит, что FRAME# и LOCK# сняты.
Второе уравнение в goto LOCKED в состоянии FREE может быть удалено, если выполнено быстрое декодирование. Первое уравнение может быть удалено, если выполнено среднее или медленное декодирование. L-lock# является запирающимся LOCK# в течение фазы адреса и используется, когда агент завершает декодирование.





Master LOCK Machine

Master Sequencer Machine

IDLE - Условие неактивности на шине.

goto ADDR	if Request * !GNT# * FRAME# * IRDY# * !Step
goto DR_BUS	if (!Request * !GNT# + Request * !GNT# * Step) * FRAME# * IRDY#
goto IDLE	if ELSE

ADDR - Ведущий начинает транзакцию.

goto M_DATA	На следующем фронте CLK

M_DATA - Ведущий передает данные.

```
goto M_DATA    if !FRAME# + FRAME# * TRDY# * STOP#
               * !Dev_to * !(cmd = 0001 * Comp)
goto ADDR      if FRAME# * !Step * !TRDY# * STOP# * !(cmd=0001)
               * (Sa * L_cycle * Request * !GNT#)
goto TURN_AR   if FRAME# * !TRDY# * STOP#
goto S_TAR     if FRAME# * !STOP# + FRAME# * Dev_to
```

TURN_AR - Транзакция завершила выполнение служебных действий.

```
goto ADDR      if Request * !GNT# * !Step
goto  DR_BUS   if !Request * !GNT# + Request * !GNT# * !Step
goto IDLE      if GNT#
```

S_TAR - Останов был установлен на обратном цикле.

```
goto DR_BUS    if !GNT#
goto IDLE      if GNT#
```

DR_BUS - Шина, закреплена для агента или агент использует продвижение адреса.

```
goto DR_BUS    if Request * !GNT# * Step + !Request * !GNT#
goto ADDR      if Request * !GNT# * !Step
goto IDLE      if GNT#
```

Master LOCK Machine

FREE - LOCK# не используется (не собственный).

```
goto FREE      if LOCK# + !LOCK# * Own_lock
goto BUSY      if !LOCK# * !Own_lock
```

BUSY - LOCK# используется в настоящее время (собственный).

```
goto FREE      if !LOCK# * FRAME#
goto BUSY      if !LOCK# * !FRAME#
```

Ведущий транзакции ответственен за управление следующими сигналами:

Enable the output buffers:

OE[FRAME#] = ADDR + M_DATA
OE[C/BE[3::0]#] = ADDR + M_DATA + DR_BUS
if ADDR drive command
if M_DATA drive byte enables
if DR_BUS if (Step * Request) drive command else drive lines to a valid state.

OE[AD[31::00]] = ADDR + M_DATA * (cmd = write) + DR_BUS
if ADDR drive address
if M_DATA drive data
if DR_BUS if (Step * Request) drive address else drive lines to a valid state.

OE[LOCK#] = Own_lock * M_DATA + OE[LOCK#] * (!FRAME# + !LOCK#)
OE[IRDY#] == [M_DATA + ADDR]
OE[PAR] == [ADDR + M_DATA * !IRDY# * (cmd = write)][3]
OE[PERR#] = (cmd = read) * !TRDY# * (delayed by two clocks)

Следующие сигналы генерируются из состояния и выбранных (не асинхронно) сигналов шины.

FRAME# = !(ADDR + M_DATA * !Dev_to * {[!Comp
* (!To + !GNT#) * STOP#] + !Ready})

IRDY# = ![M_DATA * (Ready + Dev_to)]
REQ# = ![(Request * !Lock_a + Request * Lock_a * FREE)
* (!S_TAR + Last State not S_TAR)]
LOCK# = Own_lock * ADDR + Target_abort
+ Master_abort + M_DATA * !STOP# * TRDY# * !Ldt
+ Own_lock * !Lock_a * Comp * M_DATA * FRAME# * !TRDY#

PAR = even parity across AD[31::00] and C/BE[3::0]# lines
PERR# = R_perr

Master_abort = (M_DATA * Dev_to)
Target abort = (!STOP# * DEVSEL# * M_DATA * FRAME# * !IRDY#)
Own_lock = LOCK# * FRAME# * IRDY# * Request *!GNT# * Lock_a
+ Own_lock * (!FRAME# + !LOCK#)

[3] OE[PAR] = OE[AD[31::00]] с задержкой на один такт.

Определения

Эти сигналы идут между bus sequencer и backend. Они обеспечивают информацию для sequencer, когда транзакция выполняется и обеспечивают информацию для backend о том, как транзакция продолжается. Если цикл повторяется, backend будет делать правильную модификацию используемых регистров и затем указывать sequencer выполнять другую транзакцию. Bus sequencer не забывает, что транзакция была повторена или прервана, но берет запросы из backend и выполняет PCI транзакцию.

Master-abort	Транзакция была прервана ведущим. (Нет DEVSEL#)
Target-abort	Транзакция была прервана адресатом.
Step	Агент, использующий продвижение адреса, (ждет в состоянии пока !Step).
Request	Задержка запроса.
Comp	Текущая транзакция в последней фазе данных.
L-cycle	Последний цикл был записью.
To	Тайм-аут ведущего истек.
Dev-to	Devsel таймер истек без установки DEVSEL#
Sa	Следующая транзакция тому же агенту, что и предыдущая.
Lock-a	Запрос блокированной операции.
Ready	Готов передать данные.
Sp-cyc	Команда Special Cycle.
Own-lock	Этот агент в настоящее время обладает LOCK#.
Ldt	Данные пересылались в течение операции LOCK.
R-perr	Сообщение об ошибке четности - импульс, продолжительностью в один такт PCI

Конечный автомат ведущего имеет много встроенных опций, который не интересны для некоторых реализаций. Каждое состояние будет обсуждено, показывая воздействие некоторых опций на уравнения.

IDLE - когда ведущий ждет запрос, чтобы сделать операцию шины. Единственая опция в этом состоянии - имеет термин "Step". Может быть удалено из уравнений, если продвижение адреса не поддерживается. Все пути должны быть реализованы. Путь к DR-BUS требуется для обеспечения, чтобы шина не оставлялась в третьем состоянии в течение длительных периодов Ведущий, чей GNT# установлен, должен управлять шиной, если Request не установлен.

ADDR не имеет опций и используется для управления адресом и командой на шине.
M_DATA - там, где данные перемещены. Если ведущий не поддерживает fast back-to-back транзакций , путь к состоянию ADDR не требуется.

Уравнения правильны с точки зрения протокола. Однако, компиляторы могут давать ошибки, когда они проверяют все возможные комбинации. Например, из-за протокола, Comp не может быть утвержден, когда FRAME# снят. Comp указывает, что ведущий находится в последней фазе данных, и FRAME# должен быть снят для того, чтобы быть истинным. Следовательно, первый терм в уравнении goto M_DATA вызывает проблемы с последним термом в уравнении goto TURN_AR. Чтобы устранения этого, добавьте FRAME# * TRDY# * STOP# к терму (cmd = 0001 * Comp). Если только FRAME# добавлен, то на другие уравнения воздействуют подобным способом. Последний терм в уравнении goto S_TAR требует добавления TRDY# по той же самой причине.

TURN_AR - это где ведущий снимает сигналы при подготовке к установке их в третье состояние. Путь к ADDR может быть удален, если ведущий не делает back-to-back транзакций.
S_TAR может быть реализован несколькими способами. Состояние было выбрано, чтобы разъяснить, что "состояние" должно быть вспомнено, когда адресат устанавливает STOP#.

DR_BUS используется, когда GNT# был установлен, и ведущий или не подготовлен, чтобы начать транзакцию (для продвижения адреса) или не имеет незавершеных транзакций. Если продвижение адреса не реализовано, то уравнение в goto DR_BUS, имеющее "Step" может быть удалено и уравнение goto ADDR может также удалить "Step".

Если LOCK# не поддерживается ведущим, то состояния FREE и BUSY могут быть удалены. Эти состояния для ведущего, чтобы знать состояние LOCK# когда он желает делать блокированную транзакцию. Конечный автомат просто проверяет, что LOCK# установлен. Если установлен, то остается BUSY пока FRAME# и LOCK# оба не сняты выражая, что LOCK# теперь свободен.

Приложение C

Правила эксплуатации

Когда сигналы устойчивы

1. Следующие сигналы гарантировано будут устойчивыми на всех фронтах CLK, если только сброс завершен: LOCK#, IRDY#, TRDY#, FRAME#, DEVSEL#, STOP#, REQ#, GNT#, REQ64#, ACK64#, SBO#, SDONE, SERR# (только на спаде) и PERR#.
2. Address/Data линии гарантированно будут устойчивыми определенном смене синхронизации следующим образом:
    a. Address - AD[31::00] устойчивы независимо от логического состояния на первом такте выборки при установленном FRAME#.
    b. Address - AD[63::32] устойчивы независимо от логического состояния на первом такте выборки при установленном REQ64#.
    c. Data - AD[31::00] устойчивы и действительны независимо, какой маршрут байта включен в транзакцию на чтение, когда установлен TRDY# и на запись, когда установлен IRDY#. В любое другое время они могут быть не определены. Линии AD не могут изменяться, пока текущая фаза данных не завершится, если IRDY# установлен на транзакции записи или TRDY# установлен на транзакции чтения.
    d. Data - AD[63::32] устойчивы и действительны независимо, какой маршрут байта включен в транзакцию, когда ACK64# установлен и TRDY# установлен на чтении и когда IRDY# установлен на записи. В любое другое время они могут быть не определены.
    e. Data - Special cycle command - AD [31::00] устойчивы и действительны независимо, какой маршрут байта включен в транзакцию, когда IRDY# установлен.
    f. Не пропускайте асинхронные данные непосредственно к PCI, пока IRDY# установлен на транзакции записи и пока TRDY# установлен на транзакции чтения.
3. Command/Byte enables гарантированно будет устойчивым на определенной смене синхронизации следующим образом:
    a. Command - C/BE[3::0]# и C/BE[7::4]# устойчивы и допустимы первый раз, когда FRAME# и REQ64# соответственно установлены и содержат коды команды.
    b. Byte Enables - C/BE[3::0]# и C/BE[7::4]# устойчивы и допустимы такт после фазы адреса и каждый такт в течение всей фазы данных независимо, если состояния ожидания вставлено. В течение пакета, ведущий обновляет byte enables в такте, заканчивающем каждую фазу данных (IRDY# и TRDY# установлены). Обновленное значение допустимо на следующий такт.

4. PAR устойчив и допустим один такт после времени действия AD[31::00]. PAR64 устойчив и допустим один такт после времени действия of AD[63::32].
5. IDSEL устойчив и допустим только в первый такт, когда FRAME# установлен, когда доступ - команда конфигурации. IDSEL не определен в любое другое время.
6. RST#, IRQA#, IRQB#, IRQC# и IRQD# не ограничены в использовании или синхронны.

Сигналы управления

7. Когда FRAME# и IRDY# установлены и установлен GNT#, агент может начинать доступ.
8. Транзакция начинается, когда FRAME# является установленным впервые.
9. Следующее определяет FRAME# и IRDY# во всех PCI транзакциях.
    a. FRAME# и соответствующий IRDY# определяют занятое/неактивное состояние шины, когда любой утвержден - шина занята; когда оба сняты, шина неактивна.
    b. Если FRAME# был снят, он не может быть переустановлен в течение той же самой транзакции.
    c. FRAME# не может быть снят, если IRDY# установлен. (IRDY# должен всегда устанавливаться на первой смене синхронизации, что FRAME# снят).
    d. Если ведущий установил IRDY#, он не может изменять IRDY# или FRAME#, пока текущая фаза данных не завершится.

10. Последняя фаза данных завершается, когда:
    a. FRAME# снят и TRDY# установлен (нормальное окончание) или
    b. FRAME# снят и STOP# установлен (target termination) или
    c. FRAME# снят и таймер выбора устройства истек (master abort)
    d. DEVSEL# снят и STOP# установлен (target abort).

11. Когда FRAME# и IRDY# сняты, транзакция закончилась.
12. Следующие правила определяют FRAME#, IRDY#, TRDY# и STOP# во всех PCI транзакциях.
    a. Всякий раз, когда STOP# установлен, FRAME# должен быть снят как можно скорее в соответствии с правилами снятия FRAME# (т.е., IRDY# должен быть установлен). Снятие FRAME# должен произойти как можно скорее после установки STOP#; предпочтительно в два или три цикла. Адресат не должен принять никакую связь синхронизации между установкой STOP# и снятием FRAME#, но должен хранить STOP# установленным пока снят FRAME#. Когда ведущий производит выборку установленного STOP#, он должен снять FRAME# на первом цикле с этого времени, в котором IRDY# установлен. Установка IRDY# (и следовательно снятие FRAME#) может происходить как следствие нормального IRDY# поведения ведущего (имел текущую транзакцию не завершенную адресатом) и быть отсрочено нуль или большее количество циклов в зависимости от того, когда ведущий готов для завершения передачи данных. В качестве альтернативы, ведущий может установить IRDY# немедленно (даже без подготовки для завершения передачи данных), если TRDY# снят, таким образом показывая отсутствие дальнейшей передачи данных.

b. Если STOP# установлен, то должен остаться установленным, пока FRAME# не снят; после чего, STOP# должен быть снят.
c. Если адресат установил TRDY# или STOP#, он не может изменять DEVSEL#, TRDY#, STOP#, пока текущая фаза данных не завершится.

13. Данные пересылаются между ведущим и адресатом на каждой смене синхронизации, для которой IRDY# и TRDY# установлены.
14. Источник данных требуется, чтобы прервать xRDY# сигнал безоговорочно, когда данные действительны (IRDY# на транзакции записи, TRDY# на транзакции чтения). Принимающий агент может установить xRDY# по выбору.
15. Ведущий должен снять REQ# сигнал, когда текущая транзакция завершена адресатом (STOP# установлен). Хозяин должен снять REQ# минимум на два такта PCI, один, когда шина идет к неактивному состоянию (в конце транзакции, где STOP# был установлен) и такт за или такт после неактивного состояния.
16. Агент утверждает, что был адресатом доступа, устанавливая DEVSEL#.
17. DEVSEL# должен быть установлен с или до смены, в которой адресат разрешает выходы (TRDY#, STOP# или (на чтении) линии AD).
18. Если DEVSEL# был установлен, он не может быть снят, пока не завершится последняя фаза данных, исключая сигнал target-abort.

Монопольный доступ

19. LOCK# находится в собственности, и управляется только одним агентом, и может сохраняться, когда шина освобождается.
20. Адресат, поддерживающий LOCK# на PCI должен твердо придерживаться следующих правил:
    a. Адресат на блокированных доступах самостоятелен, когда LOCK# снят в течение фазы адреса.
    b. Если только блокировка установлена, адресат остается блокированным, пока не производит выборку обоих FRAME# и LOCK# снятых вместе или это сигналы Target-abort.
    c. Гарантируйте монопольно владельцу LOCK# (если только блокировка установлена) минимум 16 байтов (выравненных) ресурса[1] . Это включает доступы, которые не происходят на PCI для многопортовых устройств.

21. Ведущий, поддерживающий LOCK# на PCI должен твердо придерживаться следующих правил:
    a. Ведущий может обращаться только к одиночному ресурсу в течение блокированной операции.
    b. Блокировка не может колебаться между границ устройства.
    c. Шестнадцать байтов (выровненных) - максимальный размер ресурса, на который ведущий может рассчитывать как на монопольный в течение блокированной операции. Доступ к любой части 16 байтов блокирует все 16 байтов.
    d. Первой транзакцией блокированной операции должна быть транзакция чтения.

[1] максимум - полный ресурс.

e. LOCK# должен быть установлен такт после фазы адреса и сохраняться установленным, чтобы сохранять управление.
f. LOCK# должен быть освобожден, если повторение сообщается прежде, чем фаза данных завершена и блокировка не установлена[2].
g. LOCK# должен быть освобожден всякий раз, когда доступ завершен с target-abort или master-abort.
h. LOCK# должен быть снят минимум на один неактивный цикл между последовательными операциями блокировки.

Арбитраж

22. Арбитр может снять GNT# агента на любом такте.
23. Если установлен, GNT# может быть снят согласно следующим правилам:
    a. Один GNT# может быть снят при совпадении с установкой другого GNT#, если шина не в неактивном состоянии. Иначе, задержка на один такт требуется между снятием GNT# и установкой следующего GNT# или еще может иметься состязание линий AD и PAR.

Пока FRAME# снят GNT# может быть снят в любое время, для обслуживания ведущего с более высоким приоритетом[3] (3) или в ответ на снятие связанного REQ#. Если GNT# снят и FRAME# установлен, транзакция шины допустима и продолжится.

24. Когда арбитр утверждает GNT# агента и шина неактивна, то агент должен разрешить AD[31::00], C/BE[3::0]# и(один такт спустя) PAR буферов вывода за восемь PCI тактов (требуемых), в то время как два-три такта рекомендуется.

Контроль по четности

25. Паритет генерируется согласно следующим правилам:
    a. Паритет вычисляется одинаково на всех PCI транзакциях независимо от типа или формы.
    b. Количество "1" на AD[31::00], C/BE[3::0]# и PAR равно четному числу.
    c. Количество "1" на AD[63::32], C/BE[7::4]# и PAR64 равно четному числу.
    d. Генерирование паритета обязательно; это должно выполняться всеми PCI-совместимыми устройствами.

[2] Если блокировка была установлена, ведущий сохраняет монопольное использование LOCK#, когда завершение с повторением или разъединение.
[3] Более высокий приоритет здесь не подразумевает фиксированный приоритетный арбитраж, но относится к агенту, который выиграл бы арбитраж в данный момент времени.

Глоссарий

64-bit extension	группа сигналов PCI, которая поддерживает 64-битовый канал данных.
address spaces	выражение для разделения на три области физических адресов PCI: память, ввод/вывод и конфигурация.
access latency	время между запросом доступа мастера к PCI и завершением первой фазы данных. Время ожидания доступа состоит из трёх частей: время ожидания арбитража, время ожидания захвата шины, и время ожидания целевого устройства.
agent	объект, который функционирует на шине компьютера.
arbitration latency	первая составляющая времени ожидания доступа - это время, которое ждет мастер после установления REQ#, пока его не получит GNT#.
backplate	металлическая пластина использованная для закрепления платы расширения к системному блоку.
BIST register	необязательный регистр в области заголовка, используемый для управления и определения состояния встроенных тестов.
bridge	логика, которая соединяет одну компьютерную шину с другой, разрешая агенту на одной шине обратиться к агенту на другой.
burst transfer	механизм передачи основной шины PCI. Импульс , включающий в себя фазу адреса и одну или больше фаз данных.
bus acquisition latency	вторая составляющая времени ожидания доступа. Количество времени, которое запрашивающее устройство ждет шину, освобождающуюся после установления GNT#.
bus commands	сигналы, используемые для указания целевому устройству типа транзакции , запрашиваемой мастером.

bus device	устройство шины может быть или мастером , или целевым: • master запускает фазу адреса, и границу транзакции (FRAME#). Мастер инициализирует транзакцию, запускает подтверждение установления связи данных (IRDY#) с целевым устройством • целевое устройство требует транзакции, устанавливая DEVSEL# и подтверждает установление связи транзакции (TRDY#) с инициатором.
central resources	функции поддержки шины, обеспеченные главной системой, обычно в PCI послушном мосте или стандартном chipset'e.
clean snoop	snoop, который не приводит к кэшу, обеспечивая изменяемые данные.
command	смотрите также команду шины.
configuration address space	набор из 64 регистров (DWORDS) используемый для конфигурации , инициализации и обработки катастрофических ошибок. Это адресное пространство состоит из двух областей: области заголовка и устройство зависимой области.
configuration cycle	циклы шины, используемые для инициализации системы и конфигурации через адресное пространство конфигурации.
DAC	двойной цикл адреса. PCI транзакция, где 64-битовая часть адреса перемещается через 32-битовый путь данных в двух временных циклах. См. также SAC.
device dependent region	последние 48 DWORDS пространства конфигурации PCI. Содержание этой области не описано в этом документе.
DWORD	32-битовый блок данных
EISA	Расширенный Промышленный Стандарт Архитектуры шины расширения, основанный на IBM AT шине, но расширенный до 32 бит адреса и данных.
expansion board	схема платы, которую добавили в материнскую плату и обеспечили дополнительную функциональность.
green machine	система, разработанная для минимального потребления мощности.
header region	первые 16 DWORDS пространства конфигурации PCI. Область заголовка состоит из полей, которые уникально идентифицируют PCI устройство и позволяет устройству быть общеуправляемым. См. также устройство-зависимая область.
hidden arbitration	арбитраж, который происходит в течение предыдущего доступа так, чтобы никакие циклы PCI шины не были использованы арбитражем, за исключением, когда шина не занята.
Host (bus) bridge	путь с наименьшим временем ожидания, через который процессор может непосредственно обратится к PCI устройствам, отображенным где-нибудь в память, ввод/вывод или адресные пространства конфигурации.

idle state — любой период времени, когда шина не занята (FRAME# и IRDY# не установлены).

ISA — Промышленный Стандарт Архитектуры шины расширения встроенный в компьютер IBM PC AT.

keepers — другое название для появляющихся резисторов, которые используются только для поддержания состояния сигнала.

latency — смотри access latency.

latency timer — механизм обеспечивающий, что мастер шины не увеличивает время ожидания доступа другого мастера более заданной величины.

livelock — состояние, в котором две или больше операций требуют завершения другой операции прежде, чем они могут завершиться.

master — агент, который инициализирует транзакцию шины.

master-abort — механизм сброса , который позволяет мастеру завершать транзакцию, когда никакое целевое устройство не отвечает.

MC — Микроканальная Архитектура шины расширения, определенная IBM для линии PS/2 персональных компьютеров

motherboard — схема платы, содержащая основные функции (например, ЦПУ, память, ввод/вывод, и разъемы расширения) компьютера.

NMI — немаскируемое прерывание.

operation — логическая последовательность транзакций, например, захват.

output driver — электрический задающий элемент (транзистор) для одиночного сигнала на PCI устройство.

PCI connector — разъем расширения, который соответствует электрическим и механическим требованиям стандарта локальной шины PCI.

PCI device — устройство (электрическая составляющая) соответствующее PCI спецификации для эксплуатации в среде PCI локальной шины.

PGA — контактная сетка ,составляющая пакет матрицы.

phase — один или больше временных периодов, в которых передается один блок информации, состоящий из:

- Фаза адреса (один адрес передается в одном такте для одиночного цикла адреса и в двух тактах для двойного цикла адреса)
- Фаза данных (одно состояние передачи плюс ноль или больше состояний ожидания)

positive decoding — метод декодирования адреса, в котором устройство отвечает на доступ только внутри назначенного адресного интервала. См. также subtractive decoding.

POST	самотестирование при включении. Ряд диагностических подпрограмм выполняющихся при включении системы.
pullups	Резисторы, используемые, чтобы обеспечить все сигналы поддерживающие устойчивые значения, когда никакой агент не активен на шине.
SAC	Одиночный цикл адреса. PCI транзакция, где 32-битовый адрес пересылается по 32-битовому каналу данных в одиночном временном цикле. См. также DAC.
shared slot	устройство на PCI материнской плате, которое позволяет PCI разъему совместно использовать слот системной шины самый близкий к PCI шине расположенный с разъемом ISA,EISA, или MC шины. В MC системе, например, общедоступный слот может размещать любую MC плату расширения или PCI плату расширения.
sideband signal	любой сигнал не являющийся частью PCI спецификации, который соединяет два или более PCI-совместимых агентов, и имеет значение только для этих агентов.
Special cycle	механизм широковещательного сообщения, используемый для связи Состояния процессора и/или (необязательно) логической передачи сигналов между PCI агентами.
Stale data	данные в кэш системе, которые больше не допустимы и, следовательно, должны быть отброшены.
stepping	способность агента к установлению длины подходящего сигнала более чем несколько временных интервалов.
subtractive decoding	метод декодирования адреса, в котором устройство принимает все доступы, не положительно декодируемые другими агентами. См. также положительное декодирование.
target	агент, который отвечает (с положительным подтверждением установления DEVSEL#), чтобы транзакция шины инициализировалась master'ом.
target abort	механизм окончания, который позволяет целевому устройству завершить транзакцию, в которой произошла фатальная ошибка, или которому целевое устройство никогда не будет больше отвечать.
target latency	третья составляющая времени ожидания доступа - количество времени, которое целевое устройство тратит на установление TRDY# для первой передачи данных.
termination	окончание транзакции приводящее шину транзакций к организованному и систематическому окончанию. Все транзакции заканчиваются когда FRAME# и IRDY# являются неустановленными (холостой цикл). Окончание может быть инициализировано master'ом или целевым устройством.
transaction	фаза адреса плюс одна или более количество фаз данных.
transfer state	любой временной интервал шины, в течение фазы данных, в котором данные перемещены.

turnaround cycle	цикл шины, используемый для предотвращения конкуренции, когда один агент останавливает запуск сигнала, а другой агент начинает его. Оборотный цикл должен длиться один временной интервал и требуется на все сигналы, которые могут управлять более чем одним агентом.
wait state	временной интервал шины, в котором не происходит никакая передача.