ГОСТ Р ИСО/МЭК 10857—95

ГОСУДАРСТВЕННЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ

МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ

ИНТЕРФЕЙС ФЬЮЧЕБАС +. СПЕЦИФИКАЦИИ ЛОГИЧЕСКОГО УРОВНЯ

Издание официальное

БЗ 11-12—94/538

ГОССТАНДАРТ РОССИИ
Москва

Предисловие

1 РАЗРАБОТАН Научно-исследовательским институтом ядерной физики Московского государственного университета им. М. В. Ломоносова

ВНЕСЕН Управлением стандартизации и сертификации информационных технологий, продукции электротехники и приборостроения Госстандарта России

2 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Постановлением Госстандарта России от 21.12.95 № 622

Стандарт подготовлен методом прямого применения международного стандарта ИСО/МЭК 10857—94 «Информационная технология. Микропроцессорные системы. Интерфейс Фьючебас+. Спецификации логического уровня» и полностью ему соответствует

3 ВВЕДЕН ВПЕРВЫЕ

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ НАУЧНОГО РЕДАКТОРА РУССКОГО ТЕКСТА

В ведущих западных исследовательских центрах и фирмах, выпускающих вычислительную технику и аппаратуру систем автоматизации, разворачиваются работы на основе международного сотрудничества по выработке принципов построения и структурных решений, созданию технических и программных средств следующего поколения многопроцессорных, магистрально-модульных высокопроизводительных систем сбора и обработки данных на основе стандарта Futurebus+.

Одной из причин появления нового магистрально-модульного интерфейса были постоянные требования повышения производительности процессоров в инженерных расчетах, моделировании и анализе данных измерений. При существующей технологии производства микросхем памяти неэкономично повышение тактовой частоты процессоров св. 50 МГц, поэтому повышение производительности процессоров возможно за счет:

а) повышения их разрядности св. 32 разрядов (однако следует учитывать, что далеко не во всех задачах данные представлены с точностью, требующей более 32 разрядов);

б) более широкого внедрения параллельных методов обработки, что требует достижения предельной пропускной способности средств связи между процессорами.

В качестве второй причины можно назвать разобщенность и отсутствие унификации на рынке 32-разрядных магистрально-модульных систем в предыдущем десятилетии, когда каждая из ведущих западных фирм имела свой внутренний стандарт: Motorola-VME, Intel-Multibus-2, DEC-VAX-BI, Texas Instruments-Nu Bus, что порождало проблемы несовместимости, особенно для крупных потребителей средств вычислительной техники и автоматизации.

Особенности и возможности интерфейса Фьючебас+ для создания перспективных радиоэлектронных средств

Интерфейс Futurebus+ разрабатывался с 1979 по 1990 гг. как проект P896 микропроцессорного комитета IEEE на магистрально-модульные системы будущего. Основными целями его разработки были:

– создание стандарта, который обеспечит существенный шаг вперед по возможностям и характеристикам будущих мультипроцессорных систем;

— обеспечение стабильной платформой производителей для создания нескольких генераций компьютерных систем.

Для этого было необходимо:

1) обеспечить эффективными протоколами поддержку работы современных высокопроизводительных микропроцессоров в многопроцессорных системах реального времени;

2) достичь предельной производительности магистрали;

3) иметь эффективные средства для построения сложных многомагистральных систем различной конфигурации, а также более простые версии протоколов для менее сложных систем (свойство масштабируемости);

4) иметь полностью асинхронный протокол с полностью распределенным управлением;

5) обеспечить высокий уровень надежности и диагностируемости операций, а также возможность динамической реконфигурации систем;

6) добиться технологической, микропроцессорной и архитектурной независимости интерфейса, иметь наиболее открытую систему для максимального круга пользователей.

Futurebus+ содержит все современные протоколы:

а) поиска распределенных данных и обработки запросов обслуживания, включая протокол передачи сообщений, а также

б) приоритетный и «справедливый» варианты арбитража запросов доступа к магистрали с множественными приоритетными уровнями;

в) асинхронной (с подтверждением) и синхронной (пакетной) передачи данных одному или группе исполнителей, в том числе с расщеплением цикла передачи;

г) обеспечивает когерентность кешей в системах с разделением памяти.

В нем стандартизован не только протокол магистрали, но и архитектура модулей, а также, частично, протоколы системных взаимодействий.

Futurebus+ позволяет достичь предельной скорости передачи данных по магистрали 150—1000 Мбайт/с в режиме с подтверждением и 300—2000 Мбайт/с — в пакетном режиме. Эти величины весьма близки к физическому пределу, определяемому возможными скоростями распространения электрических сигналов в природе. Для достижения предельных показателей для Futurebus+ разработаны специальные приемопередатчики, работающие в логических уровнях плюс 1 В — плюс 2 В (BTL).

Масштабируемость Futurebus+ заключается:

а) в изменяемости разрядности магистрали с 32 до 256 разрядов (4 градации);

б) в изменяемости размеров печатных плат от 6 до 18 SU (3 градации);

в) в существовании различных профилей для конкретных областей применений;

г) в расширяемости протоколов от простых к более сложным, что обеспечивает понижение стоимости в более простых применениях.

Операции управления магистралью в Futurebus+ выполняются единообразно каждым модулем без подчиненности другим, процедура управления полностью распределена и не требует наличия централизованных источников управления.

Все шины передачи данных и управления на магистрали Futurebus+ имеют проверку по четности. Конструкция магистрали обеспечивает малый уровень перекрестных наводок и достаточное отношение сигнал/шум для обеспечения надежных передач на больших скоростях. Введены специальные линии магистрали и статусные регистры для диагностирования качества проводимого цикла.

Futurebus+ является открытым стандартом, это означает, что он:

а) не оптимизирован под конкретные типы микропроцессоров (что типично для упоминавшихся фирменных стандартов) и технологию;

б) развит по согласию большого числа возможных производителей;

в) не требует патентов и лицензий и не содержит ограничений для пользования;

г) допускает развитие по мере улучшения полупроводниковой технологии и обеспечивает совместимость старых и «медленных» модулей с новыми разработками.

В процессе работы над стандартом было осуществлено еще одно важное нововведение: произведен переход на полностью метрическую систему типоразмеров механических конструктивов (основной шаг для печатных плат и каркасов 1SU=25 мм).

Отмеченные особенности предопределяют следующие области применения Futurebus+.

1) суперкомпьютеры общего назначения;

2) системы для научных исследований, моделирования, распознавания образов;

3) системы сбора данных и управления сложными технологическими и производственными процессами;

4) видео (графические) системы высокого разрешения, высокопроизводительные рабочие станции;

5) высокоскоростные системы цифровой связи;

6) системы специального применения.

Особенно следует подчеркнуть, что после 10-летнего периода применения множества фирменных стандартов появился интерфейс, имеющий реальные шансы стать широко признанным международным стандартом на высокопроизводительные микропроцессорные системы.

Освоение Фьючебас+ ведущими западными фирмами

О своей заинтересованности в применении этого стандарта заявили крупнейшие ведущие производители микропроцессорного оборудования: Apple Computer, AT & T, Bell Labs, Boeing Aerospace, CDC, Data General, DEC, Ferranti, Force Computers, General Dynamics, Harris Semiconductor, Hewlett Packard, Intel, ITT, Motorola, Phylips, Siemens, Sun Micro, Tektronix, Texas Instruments и др. Ассоциация VITA объявила Futurebus+ преемником широкораспространенного стандарта VME; ожидаемое время массового перехода к новому интерфейсу — 1992—1996 гг.

За рубежом усилия по развитию Futurebus+ поддерживаются ведущими фирмами—производителями элементной базы и аппаратуры автоматизации, чем обеспечивается использование новейших перспективных технологий в создании СБИС и высокопроизводительных средств связи для этих интерфейсов.

Фирмами BICC-Vero, Mupac и Schroff освоены и выпускаются механические конструктивы (крейты, магистрали, модули), фирмами AnTel, Cable & Computer выпускаются процессорные модули, модули памяти, модули ввода-вывода, анализаторы (состояний) магистрали и мосты VME-Futurebus+.

Фирма Texas Instruments объявила о предстоящем выпуске комплекта интерфейсных СБИС для реализации протокола Фьючебас+, в который входит: Арбитр магистрали, контроллер, приемопередатчики адресов и данных.

В нашей стране до настоящего времени работы по этому направлению практически не проводились, вследствии этого отсутствует необходимый технический и технологический опыт, особенно в функциональной и схемной реализации узлов интерфейса, в освоении алгоритмов и программ сбора и обработки данных по протоколам Futurebus+.

Основные документы по Futurebus+

Протоколами Futurebus+ стандартизуются.

1) механические конструктивы и разъемы;
2) напряжения и мощности источников питания;
3) требования к охлаждению аппаратуры;
4) требования к величинам электрических сигналов на магистрали;
5) логика обмена сигналами по магистрали;
6) архитектура модулей и взаимодействия в системе.

Стандарт Futurebus+ представляет собой семейство протоколов:

P896.1 — ФБ+. Спецификации логического уровня

P896.2 — ФБ+. Спецификации физического уровня.

Кроме того, на различных стадиях разработки находятся стандарты, определяющие конфигурацию систем, спецификацию тестов, спецификации для телекоммуникационных систем, кеш-когерентную кабельную магистраль, требования к реализации магистрали, интерфейсных схем и т. д.

Существующие профили Futurebus+ перечислены в табл. 1.

Таблица 1

Профиль	Область применения	Кто заинтересован
A	Основные применения	VITA/VME сообщество
B	Ввод/вывод	Производители мини и суперЭВМ
C	Кабельные связи	Межсточные соединения
D	Настольная техника	Производители компьютеров
F	Рабочие станции	Производители рабочих станций
M	Военная техника	U. S. DOD
T	Телекоммуникации	США, Европы, Япония

Ниже приводится перевод основного документа по Futurebus+: P896.1/Проект 8.5, май 16, 1991 — Спецификации логического уровня (международный стандарт ИСО/МЭК 10857—94).

Перевод документа на русский язык выполнен коллективом сотрудников НИИ ядерной физики Московского Государственного университета им. М. В. Ломоносова под руководством профессора С. Г. Басиладзе.

ГОСУДАРСТВЕННЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Информационная технология

МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ

Интерфейс Фьючебас+. Спецификации логического уровня

Information technology. Microprocessor systems.
Futurebus+ Logical protocol specification

Дата введения 1997—01—01

1 НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

Настоящий стандарт регламентирует логический (относящийся к временным соотношениям и протоколу поведения) уровень, определяющий набор сигнальных линий, который образует архитектуру магистрали из множества сегментов, и интерфейс модулей, подключенных к сегменту магистрали. Стандарт следует применять как составную часть профиля (набор соответствующих спецификаций, которые должны быть использованы совместно при разработке продукта для выполнения требований функционального стандарта) при построении систем с высоким уровнем совместимости.

Фьючебас+ обеспечивает средства для передачи двоичной информации между платами через одну или несколько логических магистралей. Платы могут содержать любую комбинацию из одного или более процессоров и логических ресурсов, таких как кеш, память, периферийные или коммуникационные контроллеры и т. д. На рис. 1—1 показана структурная схема типового применения Фьючебас+.

Протоколы специфицированы для распределения времени доступа модулей к магистрали, которым необходимо вступить в связь с другими модулями через эту магистраль. Однако стандарт не устанавливает приоритетных правил для модулей, которые соревнуются за использование магистрали. Считается, что это является привилегией и ответственностью разработчика системы. Стандарт определяет полный набор правил для сигналов, передаваемых всеми модулями как при распределенном, так и централизованном способе доступа к магистрали (гл. 4 и 5). Стандарт также дает полный набор правил для сигналов, выдаваемых всеми модулями, участвующими в передачах на магистрали (гл. 6).

Большинство протоколов передачи в этом стандарте являются предопределенными, т. е. они регулируются причиной и эффектом взаимоотношений. Протоколы этого стандарта не зависят от их технологической реализации. Предопределенность сигналов обеспечивает разработчику логическую простоту при реализации протоколов. В результате обеспечивается максимальная совместимость продуктов, разработанных в соответствии с этим стандартом в течение срока его действия.

Для любой магистрали имеется дилемма: в какой степени должны быть стандартизованы протоколы магистрали. Необходимо гарантировать, чтобы все платы, спроектированные различными производителями, могли работать совместно, если со стороны пользователей отсутствуют какие-либо специальные требования к проектируемой системе. Так как область действия этого документа ограничена, регламентация многих системных требований к магистральным компьютерным системам отсутствует; при необходимости можно обратиться к соответствующим стандартам.

Набор протоколов спроектирован по возможности технологически независимым, что обеспечивает очень высокий уровень эффективности и производительности. Для сигналов на магистрали

Издание официальное

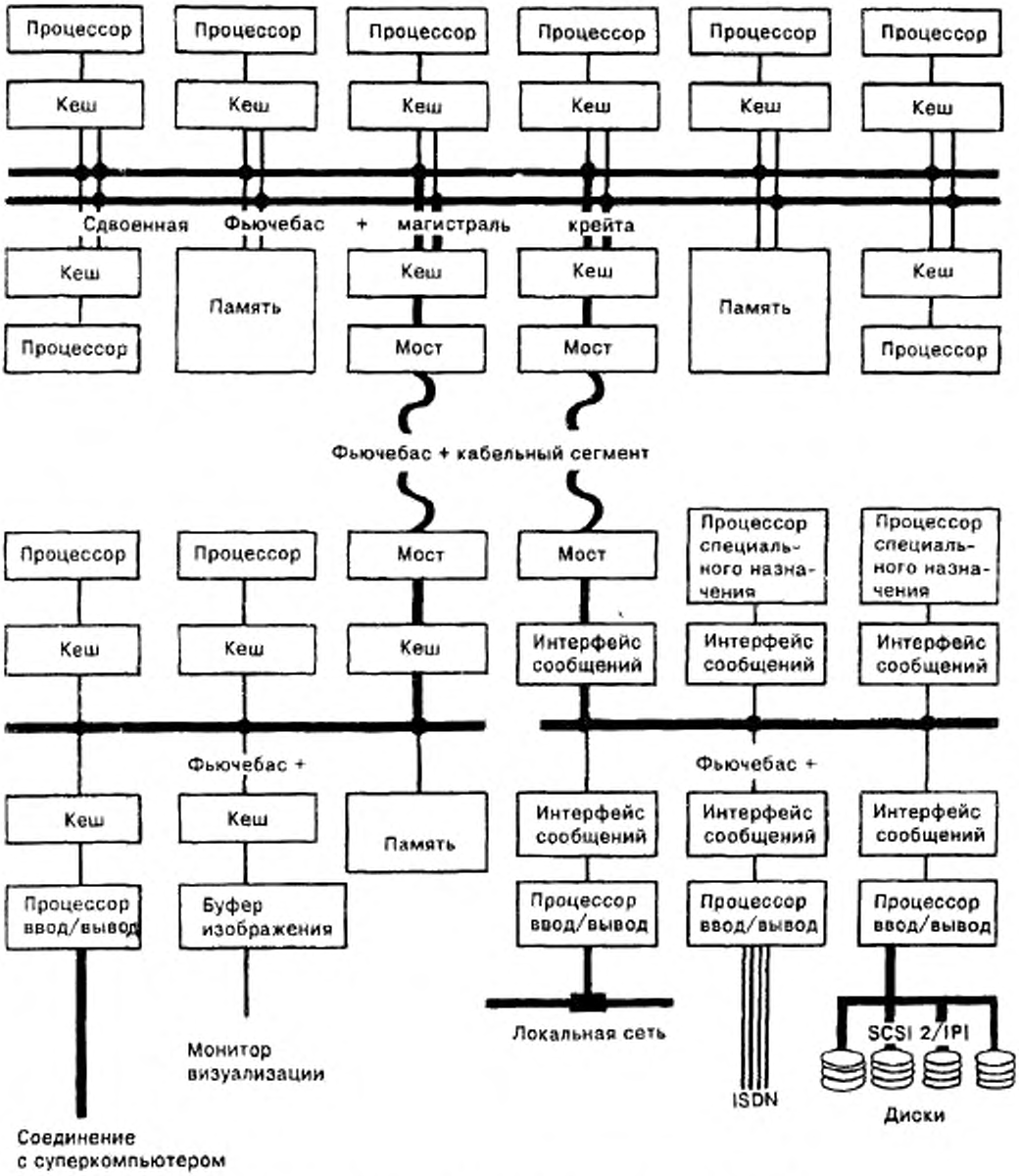

Рисунок 1—1 — Интерфейсы в семействе типовых систем Фьючебас+

может использоваться любая логика: (ТТЛ, BTL-приемопередатчики, ЭСЛ, КМОП, арсенид галлия и т. д.), поэтому приводятся условия для Фьючебас+ -сигналов (в отношении переключения сигналов на линиях передач с ограничениями на перекосы, перекрестных наводок и надежности передачи). Однако предполагается, что для максимальной совместимости между продуктами их реализация должна быть осуществлена в соответствии с одним или более Фьючебас+ -профилями, которые определяют физический уровень и набор сообщений, необходимых для частных случаев применения.

2 ОПРЕДЕЛЕНИЯ И СТРУКТУРА

2.1 **Определения действий**

МОЖЕТ — MAY

Обозначает гибкость выбора, без предопределенности.

ДОЛЖЕН — SHALL

Означает требование стандарта. Разработчики обязаны выполнять все такие требования для обеспечения совместимости.

ЖЕЛАТЕЛЬНО — SHOULD

Отмечает гибкость выбора при строгой предопределенности. Соответствует рекомендуемой практике.

2.2 **Определения линий магистрали и сигналов**

АКТИВИРОВАТЬ — ACTIVATE

Действие, состоящее в выставлении сигналов на группу линий магистрали. Аналогично, термин активированный используется для описания состояния группы линий, когда они несут сигналы.

★

Суффикс «★», добавленный к имени сигнала, индицирует, что этим сигналом состояние логической единицы заменяется состоянием логического нуля от любого другого модуля на этой линии.

ВЫСТАВЛЕН — ASSERT

Действие по выдаче состояния логической единицы на линию магистрали. Аналогично, термин выставленный используется для описания состояния линии магистрали, когда на ней присутствует логическая единица.

ЛИНИЯ МАГИСТРАЛИ — BUS LINE

Носитель для передачи сигналов. Поскольку ФБ+ требует шинных формирователей, способных работать по «проводному ИЛИ», на линию магистрали могут выставлять сигналы несколько модулей одновременно. Однако сигнал, находящийся на линии магистрали, есть комбинация сигналов от каждого модуля.

СНЯТ — RELEASE

Действие по выдаче состояния логического нуля на линию магистрали. Аналогично, термин снятый используется для описания состояния линии магистрали, когда на ней присутствует логический нуль.

НАИМЕНОВАНИЕ СИГНАЛОВ — SIGNAL NAMES

Когда группа линий магистрали представлена одинаковыми знаками, линии внутри группы нумеруются: AD0★, AD1★, AD2★ и т. д. Для того, чтобы представить группу линий или сигналов в более удобной форме, используются обозначения AD[63 . . . 0]★. Обозначение AD[]★ используется по отношению ко всем линиям внутри группы

Пример соглашения о сигналах показан на рис. 2—1 для схемотехники с «открытым коллектором» (хотя эта спецификация не зависит от типа логики). Сигнал определенного модуля, прикладываемый ко входу его шинного формирователя, обозначается строчными буквами, т. е. ai. Сигнал в модуле, выводимый на магистраль, обозначается строчными буквами со звездочкой, т. е. ai★. Сигнал, который появляется на линии магистрали как результат объединения сигналов от всех модулей, обозначается прописными буквами со звездочкой, т. е. AI★.

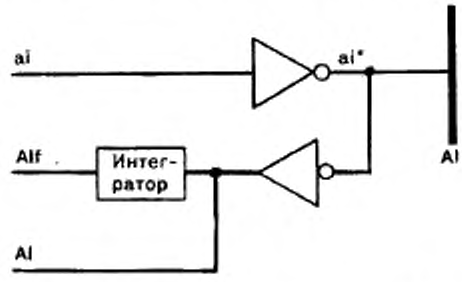

Рисунок 2.1 – Соглашения о сигналах

Суффикс «f» (фильтрованный), добавленный к имени сигнала, относится к магистральному сигналу после его прохождения через приемник и фильтр «шпилек» с «проводным-ИЛИ» (интегратор). Для примера, AIf относится к сигналу на линии AI, который после его прохождения через инвертирующий приемник становится AI, а после фильтра — AIf.

В обозначении WR★ знак «0» означает логический нуль (отпущен), а «1» означает логическую единицу (выставлен).

2.3 **Определения**

ПЕРЕДАЧА ТОЛЬКО АДРЕСА — ADDRESS ONLY TRANSACTION

Передача на магистрали, которая не включает фазы данных. Информация передается только во время фазы соединения и иногда в фазе рассоединения.

АРБИТРАЖ — ARBITRATION

Процесс выбора следующего задатчика.

СООБЩЕНИЕ АРБИТРАЖА — ARBITRATED MESSAGE

Широковещательное обращение по линиям арбитража магистрали ко всем модулям магистрали.

ОБМЕН — BEAT

Событие, которое начинается сигналом синхронизации задатчика, следующим за отпусканием линии подтверждения одним или несколькими исполнителями. Управляющая информация и данные могут быть переданы от мастера к одному или нескольким исполнителям в первой части обмена. Во второй части обмена исполнители могут передать информацию о способности, статусе и данные обратно задатчику.

КОПИРОВАНИЕ БЛОКА — BLOCK COPY

Операция копирования блока, характеризуемая длиной серии передач чтения или записи в последовательные ячейки памяти.

МАГИСТРАЛЬНЫЙ МОСТ — BUS BRIDGE

Межсоединение между двумя или более магистралями, которое обеспечивает трансляцию сигналов и протокола с одной шины на другую. Магистрали могут принадлежать к различным стандартам по механике, электрическим параметрам и логическому протоколу.

ВЛАДЕНИЕ МАГИСТРАЛЬЮ — BUS TENURE

Интервал управления магистралью задатчиком, т. е. время, в течение которого модуль имеет право инициировать и выполнять передачи по магистрали.

ПЕРЕДАЧА ПО МАГИСТРАЛИ — BUS TRANSACTION

Событие, начинаемое фазой соединения и завершаемое фазой рассоединения. Данные могут передаваться, а могут и не передаваться.

ЗАНЯТО — BUSY

Если исполнитель не способен воспринять передачу от задатчика, он может выставить статус занятости задатчику. Задатчик должен освободить магистраль и может после соответствующего временного интервала снова получить к ней доступ и повторить передачу.

БАЙТ — BYTE

Набор из восьми соседних двоичных цифр.

БАЙТОВАЯ ШИНА — BYTE LANE

Путь данных, образованный восемью линиями данных и одной линией четности, используемый для переноса одного байта между системными модулями.

КЕШ-КОГЕРЕНТНОСТЬ — CACHE COHERENCE

Система кешей является когерентной по отношению к кеш-строке, если каждый кеш и главная память в когерентном домене отмечают все модификации в своей кеш-строке. Модификации отмечаются кешем, когда любое последующее чтение возвратит новую записываемую величину.

КЕШ ПАМЯТЬ — CACHE MEMORY

Буферная память, помещаемая между одним или несколькими процессорами и магистралью, содержащая текущую активную копию блоков из главной памяти. Кеш памяти пользуется пространственной локальностью при попадании в кеш. Временная локальность используется стратегией, применяемой для определения того, что убрано из кеша.

КОГЕРЕНТНЫЙ ДОМЕН — COHERENCE DOMAIN

Область в мультикешевой системе, внутри которой мера целостности поддерживается принудительно. В системах, содержащих мосты, когерентность домена может выходить, а может и не выходить за локальную магистраль через мост на удаленную магистраль.

КОГЕРЕНТНАЯ СТРОКА — COHERENCE LINE

Блок данных, для которого поддерживаются атрибуты целостности кеша.

ПРОТОКОЛ ПРИНУДИТЕЛЬНОЙ ПЕРЕДАЧИ ДАННЫХ — COMPELLED DATA TRANSFER PROTOCOL

Механизм независимой передачи, относящийся к принудительному, поскольку исполнитель принуждается к выдаче ответа перед тем, как задатчик начнет следующую передачу.

СОРЕВНУЮЩИЙСЯ — COMPETITOR

Модуль, активно участвующий в текущем цикле занятия магистрали процесса арбитража.

СВЯЗНАЯ ПЕРЕДАЧА — CONNECTED TRANSACTION

Передача, в которой и запрос и ответ выполняются внутри одного цикла передачи.

ФАЗА СВЯЗИ — CONNECTION PHASE

Событие, которое начинается выставлением синхронизации адреса вслед за снятием подтверждения адреса. Используется для широковещания адресной и управляющей информации. Модули определяют, хотят ли они принять участие в передаче на основе этой информации.

ЗАХВАТ УПРАВЛЕНИЯ — CONTROL ACQUISITION

Активность всей магистрали, связанная с приобретением исключительного права управления магистралью.

ВОЗВРАТНЫЙ КЕШ — COPYBACK CACHE

Схема организации кеш-памяти с атрибутами данных, которые обычно записываются из процессора в кеш быстрее, чем в основную память. Модифицированные данные в кеше записываются в основную память, когда кеш-цепочка заполняется или замещается принудительно для записи в основную память для избежания потери данных.

РУС — CSR

Управляющий и статусный регистр.

АРУС — CSRA

Архитектура управляющего и статусного регистра (см. IEEE P1212).

ФАЗА ДАННЫХ — DATA PHASE

Интервал внутри цикла передачи, используемый для передачи данных.

ТУПИК (ТУПИКОВАЯ СИТУАЦИЯ) — DEADLOCK

Тупик — состояние, когда одни модули ожидают действий, которые могут быть выполнены только другими ожидающими модулями, а другие ожидающие не могут выполнить этих действий (в т. ч. из-за ожидания первых).

ФАЗА РАССОЕДИНЕНИЯ — DISCONNECTION PHASE

Интервал внутри цикла передачи, используемый для возвращения сигналов магистрали к состоянию покоя. Кроме того, эта фаза может быть использована для передачи дополнительной информации, требуемой для выполнения или исключения запрошенной операции.

ДУБЛЕТ — DOUBLET

Набор из двух соседних байтов.

ВХОДЯЩИЙ — ENTRANT

Вставленный модуль, находящийся в процессе соотнесения себя с протоколом арбитража.

Очень быстрый, но технологически зависимый непринудительный механизм передачи, который использует принудительный протокол над целостным пакетом для обеспечения контроля потока.

ПАРАЛЛЕЛЬНЫЙ АРБИТРАЖ — PARALLEL CONTENTION ARBITRATION

Процесс, в котором модули выставляют их уникальный код арбитража на параллельную магистраль и снимают сигналы в соответствии с алгоритмом так, что по истечении времени выигравший код появляется на магистрали.

УЧАСТВУЮЩИЙ ИСПОЛНИТЕЛЬ — PARTICIPATING SLAVE

Исполнитель, вовлеченный в передачу как селектированный, внедренный или широковещательный исполнитель, либо исполнитель, вовлеченный в множественный пакетный режим.

ОТКАЗЫВАНИЕ (ОТ МАГИСТРАЛИ) — PREEMPTION

Отказывание происходит, когда действующий задатчик освобождает магистраль потому, что другой модуль запрашивает ее. В некоторых системах любой модуль может вызывать отказывание, а в других — только модуль с высоким приоритетом запроса.

КВАДЛЕТ — QUADLET

Набор из четырех соседних байтов.

ХРАНИТЕЛЬ ПОСЛЕДНЕГО ПОЛОЖЕНИЯ REPOSITORY OF LAST RESORT

В иерархической памяти (на базе кеша) хранитель последнего положения разделяемых данных есть накопительная ячейка, которая имеет только последнюю оставшуюся копию разделяемых данных. Она может быть уникальным источником последнего приемника или просто хранителем данных, которые могут быть недействительны, если не предприняты действия по предохранению копии данных на некотором более высоком уровне иерархии памяти (или кеша). Только в кешевых ФБ− системах (т. е. таких, где даже главная ОЗУ спроектирована как аппаратный кеш) хранитель последнего положения задействуется, когда устанавливается связь адреса физической ячейки в одном из кешей при создании данных или при инициализации копии с какого-либо высшего уровня иерархии памяти, или по их появлении с какого-либо устройства ввода-вывода. Эти данные могут мигрировать по системе и быть во владении различных кешей в различное время, обеспечивая не менее одной копии данных, сохраняемой где-либо: это и есть хранитель последнего положения. Хранитель последнего положения может прекратить свое существование при ясной инструкции «разрушить» данные путем миграции на верхний уровень памяти (или иерархического кеша) или путем передачи владения через какое-либо устройство ввода-вывода к другой системе, накопительному прибору или дисплею.

ЗАПРОС REQUEST

Запрос есть команда, генерируемая запросчиком для инициирования действия ответчика. Для передачи чтения процессором памяти, например, запрос передает адрес памяти и команду от процессора к памяти. В случе расщепленного цикла запрос может быть разделенной передачей. В случае связной передачи запрос есть фаза связи цикла передачи.

ЗАПРОСЧИК REQUESTER

Модуль, который инициирует передачу посылкой запроса (содержащего адрес, команду и иногда данные) по отношению к ответчику.

ОТВЕТЧИК RESPONDER

Модуль, который завершает передачу посылкой ответа (содержащего завершающий статус и иногда данные), относящегося к ответчику.

СВОБОДНЫЙ ДЛЯ ВЫПОЛНЕНИЯ FORWARD PROGRESS

Модуль, который не заблокирован от решения задач, необходимых для достижения цели, называется свободным для выполнения. Свобода выполнения гарантирует только отсутствие тупиков и зависаний.

ГЕОГРАФИЧЕСКИЙ АДРЕС GEOGRAPHICAL ADDRESS

Уникальный идентификатор, присваиваемый каждому физическому месту на магистрали и принимаемый модулем, подключенным на это место.

ГЛОБАЛЬНАЯ ИДЕНТИФИКАЦИЯ GLOBAL IDENTIFICATION

Уникальный идентификатор, присваиваемый каждому физическому месту для модуля в системе. Этот идентификатор может обычно включать как идентификатор магистрали, так и идентификатор места. Р1212 определяет формат для такого глобального идентификатора.

ВНЕДРЯЮЩИЙСЯ ИСПОЛНИТЕЛЬ INTERVENING SLAVE

Участвующий исполнитель, который хотя и не является хранителем последнего местоположения запрашиваемых данных, находит необходимым внедриться к хранителю последнего местоположения ради обеспечения запрашиваемых данных. Сделав так, внедренный исполнитель поставляет данные вместо хранителя.

ЗАЦИКЛИВАНИЕ LIVELOCK

Зацикливание есть метастабильная ситуация, в которой некоторые модули приобретают и отдают ресурсы таким образом, что в их действиях нет продвижения вперед.

БЛОКИРОВАНИЕ (ДОСТУПА) LOCKING

Возможность, когда модули получают гарантированный исключительный доступ к адресованным данным, блокируя другие модули от доступа к ним. Это позволяет производить неразделимые операции с адресованными ресурсами.

ЗАДАТЧИК MASTER

Модуль, который приобрел управление магистралью посредством процедуры приобретения.

ВЫИГРАВШИЙ ЗАДАТЧИК MASTER ELECT

Модуль, который выиграл последнее арбитражное соревнование.

МОДУЛЬ — MODULE

Схемное устройство, спроектированное для выполнения специфических функций, которые включают интерфейс к ФБ+.

ГЛАВНЫЙ ПРОЦЕССОР — MONARCH PROCESSOR

Главный процессор или просто главный — есть процессор, выбранный для выполнения конфигурации и инициализации всех модулей на одной логической магистрали. Системный процессор — это главный процессор, предназначенный для управления конфигурацией и инициализацией всей системы, состоящей из множества взаимосвязанных логических магистралей.

ОКТЛЕТ — OCTLET

Набор из восьми соседних байтов.

ПРОТОКОЛ ПАКЕТНОЙ ПЕРЕДАЧИ ДАННЫХ — PACKET DATA TRANSFER PROTOCOL

ОТВЕТ — RESPONSE

Ответ генерируется ответчиком при завершении передачи, инициированной запросчиком. Для передач чтения процессор—память, например, ответ возвращает данные и статус из памяти в процессор. В случае расщепленной передачи ответ будет отдельной передачей по магистрали. В случае связной передачи ответ состоит из фаз данных и рассоединения в цикле магистрали.

КРУГОВОЙ (АРБИТРАЖ) — ROUND-ROBIN

Правило занятия магистрали, по которому модуль, владевший магистралью, не допускается к магистрали снова до тех пор, пока все остальные модули с текущими запросами того же уровня приоритета не получат доступа к магистрали.

ВЫБРАННЫЙ ИСПОЛНИТЕЛЬ — SELECTED SLAVE

Исполнитель считается выбранным задатчиком, когда он распознает свой адрес на линиях магистрали в течение фазы соединения.

РАЗДЕЛЯЕМАЯ ПАМЯТЬ — SHARED MEMORY

Адресное пространство, системно доступное всем кешированным модулям.

ПЕРЕКОС — SKEW

Различие между задержками распространения двух или более сигналов на любых линиях магистрали.

ИСПОЛНИТЕЛЬ — SLAVE

Модуль, который может быть адресован и способен участвовать в передаче по магистрали.

ЛОВЯЩИЙ — SNARF

Модуль считается ловящим передачу, если он берет копию данных, проходящих по магистрали, хотя он не запрашивал их.

НАБЛЮДАЮЩИЙ — SNOOP

Модуль считается наблюдающим передачу, если он не инициировавший ее задатчик или хранитель последнего положения данных, но отслеживает передачу. Кеш-памяти наблюдают передачи для поддержания когерентности.

ПРОСТРАНСТВЕННАЯ ЛОКАЛЬНОСТЬ — SPATIAL LOCALITY

Свойство программ обращаться к тесно связанным кластерным адресам памяти в короткие временные интервалы.

РАСЩЕПЛЕННАЯ ПЕРЕДАЧА — SPLIT TRANSACTION

Системная передача, в которой запрос передается в одном цикле, а ответ — в отдельном последующем цикле магистрали.

ЗАВИСАНИЕ — STARVATION

Системное условие, которое встречается, когда один или более модулей не выполняют полезных действий в течение неопределенного промежутка времени из-за отсутствия доступа к магистрали или к другим системным ресурсам.

СИЛЬНО СВЯЗАННАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ — STRONG SEQUENTAL CONSISTENCY

Система представляет сильно связанную последовательность, если каждый участвующий кеш в системе наблюдает за всеми модификациями линий внутри себя таким же образом, как все участвующие кеши в системе. См. также слабо связанную последовательность.

ФБ+ позволяет модулям осуществлять передачи, которые динамически меняются между следованием слабосвязанной модели поведения (которая подразумевает большую согласованность и, соответственно, более высокие характеристики) и сильносвязанной модели поведения (которая может быть необходима для убежденности в правильном действии алгоритма записи программиста, не познаваемой из-за согласованности, возможной в различных частях системы).

СИСТЕМНАЯ ПЕРЕДАЧА SYSTEM TRANSACTION

Законченная операция, такая как чтение или запись памяти, как представление с инициируемого устройства. Системная передача может быть транслирована в одну или более передач по магистрали при помощи ФБ+ интерфейса завершения операции.

ВРЕМЕННАЯ ЛОКАЛЬНОСТЬ TEMPORAL LOCALITY

Свойство программ обращаться к тем же самым ячейкам памяти в коротких временных интервалах.

ПЕРЕДАЧА TRANSACTION

Событие, начинающееся с фазы соединения и оканчивающееся фазой рассоединения. Данные могут передаваться, а могут и не передаваться в течение передачи. «Передача» часто используется вместо более точной фразы «передача по магистрали» ради краткости. См. «системную передачу».

НЕВЫБРАННЫЙ ИСПОЛНИТЕЛЬ UNSELECTED SLAVE

Исполнитель, который не распознал своего адреса на магистрали в течение фазы соединения цикла передачи.

СЛАБО СВЯЗАННАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ WEAK SEQUENTAL CONSISTENCY

Система представляет слабо связанную последовательность, если отсылки к глобальным синхронизирующим переменным представляют сильно связанную последовательность и если не используются отсылки к синхронизирующим переменным, выданным любым процессором до тех пор, пока все предыдущие модификации глобальных данных не будут исследованы всеми кешами и если нет отсылок к глобальным данным, выданным любым процессором, до тех пор, пока все предыдущие модификации синхронизирующих переменных не исследованы всеми кешами. См. также «сильно связанную последовательность».

2.4 **Структура документа**

Каждая из гл. 3—9 разделена на секцию описания и секцию спецификаций. Номера в секциях описаний начинаются с номера главы и имеют индекс «.1». Номера в секциях спецификаций начинаются с номера главы и имеют индекс «.2».

Секции описаний призваны помочь разработчикам понять действие магистрали и не содержат требований к оборудованию. Секция спецификаций содержит все требования к оборудованию согласно стандарту.

Секции спецификаций написаны с использованием атрибутов (включая разряды РУС), состояний и уравнений. Этот формат, хотя он иногда труден для чтения, позволяет спецификациям быть точными. Разработчики могут познакомиться с секциями описаний перед чтением секций спецификаций.

Атрибуты обозначены прописными буквами, знак подчеркивания разделяет слова. Например: ВЫНУЖДЕННЫЙ_СТАТУС, ЗАДАТЧИК, ДЕЙСТВУЮЩИЙ_ЗАДАТЧИК являются атрибутами. Следующие символы используются в уравнениях:

- Отрицание
& Логическое И
¦ Логическое ИЛИ

Например, спецификация «СОРЕВНУЮЩИЙСЯ должен выставить ab6★, когда cn6 & СОРЕВНОВАНИЕ & (cn7 ¦ -AB7★)» будет означать: модуль с установленным атрибутом СОРЕВНУЮЩИЙСЯ должен выставить сигнал ab6★, когда установлен сигнал cn6 и установлен атрибут СОРЕВНОВАНИЕ и либо cn7 установлен или AB7★ снят.

Атрибуты есть логические величины, которые не обязательно соответствуют схемотехнике модуля. Изменения внутренних состояний устройств могут отвечать или не отвечать атрибутам, описанным в этой спецификации. Внешнее поведение любого устройства эквивалентно отвечает определенному в спецификации устройству.

Хотя эта спецификация индицирует соответствующим образом те места, где разработчик может встретиться с метастабильностями (ошибочным поведением цифровой логики в границах между двумя синхронизуемыми областями), нет гарантии, что такие предупреждения покрывают все возможные случаи метастабильности; избежать их есть ответственность разработчика, т. к. различная применяемая архитектура требует различных границ синхронизации.

2.5 **ФБ+ символ**

Символ 896, показанный на рис. 2—2, является защищенным компьютерным обществом IEEE; он может быть использован только на продуктах, полностью удовлетворяющих всем требованиям одного или более профилей, определяемых в P896.2.

Рисунок 2.2 — P896 символ

2.6 **Нормативные ссылки**

P896.1 ФБ+. Спецификации логического уровня

P896.2 ФБ+. Спецификации физического уровня

2.7 **Описание линий магистрали**

В табл. 2.1 приведен список линий магистрали; параграфы 2.7.1.1—2.7.1.5 дают краткое описание их функций. Сигналы на этих линиях описываются более полно в дальнейшем.

Таблица 2-1 · **Перечень линий магистрали**

		Разрядность, бит			
		32	64	128	256
	Информационные линии				
AD[63... 0]★	Адрес — Данные	32	64	64	64
D[255... 64]★	Данные	–	–	64	192
BP[3]... 0]★	Четность магистрали	4	8	16	32
TG[7 0]★	Теги	8	8	8	8
TP★	Четность тегов	1	1	1	1
CM[7.. 0]★	Команда	8	8	8	8
CP★	Четность команды	1	1	1	1
ST[7... 0]★	Статус	8	8	8	8
CA[2... 0]★	Способность	3	3	3	3
	Линии синхронизации				
AS★, AR★, AK★	Адресная синхронизация	3	3	3	3
DS★, DK★, DI★	Синхронизация данных	3	3	3	3
ET★	Конец владения	1	1	1	1
	Линии арбитражных сообщении				
AB[7... 0]★	Шина арбитражных сообщений	8	8	8	8
ABP★	Четность шины арбитража	1	1	1	1
AP★, AQ★, AR★	Синхронизация шины арбитража	3	3	3	3
AC[1... 0]★	Условия сообщения арбитража	2	2	2	2
	Линии инициализации-сброса				
RE★	Сброс	1	1	1	1
	Централизованный арбитраж				
RQ[1.. 0]★	Запрос	2	2	2	2
GR★	Предоставление	1	1	1	1
PE★	Отказ	1	1	1	1
	Географический адрес				
GA[4 . 0]★	Географический адрес	5	5	5	5
	Всего	96	132	204	348

О количестве линий ОВ, линий последовательной магистрали и другой информации, относящейся к профилям, обращайтесь к IEEE P896.2.

2.7.1 **Информационные линии**

Следующие линии магистрали используются для выполнения передач по магистрали.

2.7.1.1 **AD[63 . . . 0]★**

Адрес/Данные/Деселекция шины/Адрес запроса/возврата

Эти мультиплексированные двунаправленные линии адреса/данных содержат адрес во время передачи адреса. Адрес всегда выставляется задатчиком и принимается одним или несколькими исполнителями. В течение передачи данных на этих линиях находятся данные. Данные выставляются задатчиком в течение цикла записи и выбранным или внедренным исполнителем в течение цикла чтения. В течение первого обмена данных частной передачи AD[31 . . . 0]★ несут сигналы деселекции шины. В течение кешированной расщепленной передачи AD[31 . . 0]★ несут адрес запрашивающих модулей и приоритет, и статус отвечающих модулей. В течение некешированной расщепленной передачи AD[31 . . . 0]★ несут глобальные идентификаторы запрашивающих модулей и идентификатор передачи.

2.7.1.2 **D[255 . . . 64]★ Линии данных**

Эти двунаправленные линии данных содержат данные в течение передачи данных. Данные выставляются задатчиком в течение цикла записи и в ответной части расщепленного цикла чтения. Данные выставляются выбранным или внедренным исполнителем в течение цикла чтения.

2.7.1.3 **CM[7 . . . 0]★ Команда**

Набор линий, несущих управляющую информацию от задатчика к одному или нескольким исполнителям текущего цикла.

2.7.1.4 **CP★ Четность команды**

Линия содержит четность сигналов на шине управления.

2.7.1.5 **ST[7 . . . 0]★ Статус**

Набор линий, которые активируются исполнителями, отвечающими на команду задатчика. ST[7 . . . 0]★ обеспечивают информацию как о статусе самих исполнителей, так и об их отношении к текущей передаче.

2.7.1.6 **CA[2 . . . 0]★ Способность**

Набор линий, которые активируются модулями, декларирующими их способность воспринять основной режим передачи по магистрали. CA[2 . . . 0]★ обеспечивают базовый механизм, позволяющий модулям с различными способностями к передаче сосуществовать в системе; т. е. модулю, который может отвечать на пакетную передачу, и модулю, который может выполнять только принудительную передачу данных.

2.7.1.7 **BP[31 . . . 0]★ Четность магистрали**

Эти линии содержат признак четности для шин адреса/данных. Имеется одна линия четности на 8 линий адреса/данных.

2.7.1.8 **TG[7 . . . 0]★ Теги**

Эти двунаправленные линии несут дополнительную информацию, относящуюся к линиям адрес/данные. Данный стандарт не предписывает использование TG[7 . . 0]★, разработчик системы может использовать их соответственно специфике своей задачи. Количество активных линий тегов определено в регистре управления-статуса.

2.7.1.9 **TP★ Четность тегов**

Линия содержит признак четности для TG[7 . . . 0]★.

2.7.2 **Линии синхронизации**

Линии синхронизации координируют обмен адресом, командой, способностью, статусом и данными в течение передачи.

2.7.2.1 **AS★ Синхронизация адреса**

Сигнал синхронизации, который выставляется в течение фазы соединения задатчиком для того, чтобы информировать исполнителей, что адрес и управляющая информация действительны. AS★ снимается в течение фазы рассоединения для индикации того, что рассоединение данных и управления действительно.

2.7.2.2 **AK★ Подтверждение адреса**

Сигнал подтверждения снимается всеми модулями в течение фазы рассоединения для указания задатчику, что их статусная информация действительна.

2.7.2.3 **AI★ Подтверждение адреса инверсное**

Сигнал подтверждения, снимаемый всеми модулями в течение фазы соединения для указания задатчику, что их статусная и способная информация действительна.

2.7.2.4 **DS★ Синхронизация данных**

Синхросигнал, выставляемый или снимаемый задатчиком в течение фазы передачи для указания участвующим исполнителям, что управляющая информация и данные записи действительны или задатчик готов получить считываемые данные.

2.7.2.5 **DK★ Подтверждение данных**

Сигнал подтверждения, снимаемый участвующими исполнителями в течение четного обмена или четного пакета фазы данных для указания задатчику, что статусная информация и данные чтения действительны или что получены данные записи.

2.7.2.6 **DI★ Подтверждение данных инверсное**

Сигнал подтверждения, снимаемый участвующими исполнителями в течение нечетного обмена или нечетного пакета фазы данных для указания задатчику, что статусная информация и данные чтения действительны или что получены данные записи.

2.7.2.7 **ET★ Конец владения**

Этот сигнал отмечает конец владения текущим задатчиком. Снятие этого сигнала сообщает выигравшему задатчику, что он стал действующим задатчиком.

2.7.3 **Линии распределенного арбитража и сообщений арбитража**

2.7.3.1 **AB[7 . . . 0]★ Шина арбитража**

Набор линий, содержащий код старшинства соревнующихся в течение процесса арбитража сообщений и процесса распределенного арбитража.

2.7.3.2 **ABP★ Четность арбитража**

ABP★ содержит признак четности шины AB[7 . . . 0]★.

2.7.3.3 **AP★, AQ★, AR★ Синхронизация управления доступом**

Линии синхронизации с подтверждением, которые выполняют циклическую последовательность подтверждений, управляющую последовательностью процесса арбитражных сообщений и процесса распределенного арбитража.

2.7.3.4 **AC[1 . . . 0]★ Условия арбитража**

Линии, которые управляют процессом арбитража сообщений и процессом распределенного арбитража.

2.7.4 **RE★ Инициализация/сброс магистрали**

Выставляется модулем для инициализации интерфейсной логики всех модулей в предопределенное состояние и вырабатывает сигнал сброса для содержимого плат. Длительность выставленного состояния на RE★ используется для указания, соответствует ли сигнал вставлению модуля при включенном питании, инициализации магистрали или системному сбросу.

2.7.5 **Центральный арбитр**

В системах с центральным арбитром этот сигнал используется для выбора задатчика магистрали, как описано ниже.

2.7.5.1 **PE★ Отказывание**

Центральный арбитр выставляет этот сигнал для указания текущему задатчику, что имеется необходимость его отказа от магистрали.

2.7.5.2 **GR★ Предоставление**

Сигнал на этой линии выставляется центральным арбитром при предоставлении права быть задатчиком. Это есть отдельная линия от центрального арбитра к каждому модулю на магистрали.

2.7.5.3 **RQ[1 . . . 0]★ Запрос**

Сигнал на одной или обоих линиях выставляется модулями, запрашивающими владение магистралью. Имеются две отдельные линии от каждого модуля магистрали к центральному арбитру

2.7.6 **GA[4 . . .0]★ Географический адрес**

Набор статических сигналов на магистрали с кодом позиции места модуля на магистрали.

2.8 **Перечень атрибутов**

Каждый атрибут сопровождается списком ссылок на него. Подчеркнутый номер содержит описание атрибута.

СПОСОБНОСТЬ_32_РАЗРЯДНОГО_АДРЕСА — 32_BIT_ADRESS_CAPABLE
7.2.4.1.1

32_РАЗРЯДНАЯ_СПОСОБНОСТЬ — 32_BIT_CAPABLE
6.2.1.3

ПЕРЕДАЧА_32	TRANSFER_32
6.2.1.6, 6.2.1.8, 7.2.4.1.1.	
ПЕРЕДАЧА_64	TRANSFER_64
6.2.1.6, 6.2.1.8, 7.2.4.1.1.	
НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ	TRANSFER_COUNT_ZERO
6.2.1.1, 6.2.1.8, 6.2.2.1.1, 6.2.2.1.4, 6.2.3.1, 6.2.3.3, 6.2.3.5, 6.2.3.7, 6.2.3.9, 6.2.3.11, 6.2.3.15, 6.2.3.17.	
НЕОГРАНИЧЕННАЯ_ПЕРЕДАЧА	TRANSFER_UNRESTRICTED
6.2.1.6.	
ДВА_ПРОХОДА	TWO_PASS
5.2.3, 5.2.4, 5.2.8.2.	
УНИКАЛЬНОСТЬ_СОЗДАННАЯ	UNIQUENESS_CREATED
5.2.3, 5.2.8.1, 5.2.8.4.	
ОЖИДАНИЕ	WAIT
6.2.1.7.2, 6.2.2.4.8.	
ОЖИДАНИЕ_КЕШИРОВАННОЕ	WAIT_CACHED
6.2.1.7.2, 8.2.2.	
СТАТУС_ОЖИДАНИЯ	WAIT_STATUS
5.2.3, 6.2.1.7.1, 6.2.1.8, 8.2.3, 8.2.4.	
ВЫИГРАВШИЙ	WINNER
5.2.1—5.2.4, 5.2.5, 5.2.7.1.2, 5.2.7.1.3, 5.2.8.1, 5.2.8.3, 5.2.8.5, 5.2.8.6, 5.2.9.1, 5.2.9.3, 5.2.9.6.	
CMD_ЗАПИСЬ	WRITE_CMD
6.2.1.4, 6.2.1.6, 6.2.1.7.2, 6.2.2.2.5, 6.2.2.5.1, 6.2.2.5.2, 6.2.2.5.3, 6.2.3.3, 6.2.3.5, 6.2.3.7, 6.2.3.9, 6.2.3.11, 6.2.3.13, 6.2.3.15, 6.2.3.17, 6.2.3.22, 6.2.3.23.	
НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ	WRITE_INVALID
6.2.1.4, 6.2.2.5.1, 6.2.3.1, 8.2.2—8.2.5.	
БЛОКИРОВАННАЯ_ЗАПИСЬ	WRITE_LOCKED
6.2.1.4, 6.2.1.5, 6.2.2.5.1, 6.2.3.1.	
ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ	WRITE_NO_ACKNOWLEDGE
6.2.1.4, 6.2.5.2.1, 6.2.3.1, 9.2.1, 9.2.2.1—9.2.2.14.	
ЧАСТНАЯ_ЗАПИСЬ	WRITE_PARTIAL
6.2.1.4, 6.2.2.5.1, 6.2.3.1.	
БЛОКИРОВАННАЯ_ЧАСТНАЯ_ЗАПИСЬ	WRITE_PARTIAL_LOCKED
6.2.1.4, 6.2.1.5, 6.2.2.5.1, 6.2.3.1.	
ОТВЕТ_ЗАПИСИ	WRITE_RESPONSE
6.2.1.3, 6.2.1.4, 6.2.1.8, 6.2.2.5.1.	
СТАТУС_ЗАПИСИ	WRITE_STATUS
6.2.1.4, 6.2.1.7.2, 6.2.3.4, 6.2.3.6, 6.2.3.8, 6.2.3.10, 6.2.3.12, 6.2.3.14, 6.2.3.16, 6.2.3.18.	
НЕБЛОКИРОВАННАЯ_ЗАПИСЬ	WRITE_UNLOCKED
6.2.1.4, 6.2.2.2.2, 6.2.2.5.1, 6.2.3.1, 9.2.1, 9.2.2.1—9.2.2.14.	

2.9 **Используемая мнемоника**

Сопроводительная документация для изделий на базе настоящего стандарта должна использовать следующую мнемонику для индикации возможностей:

А32	A32	32-разрядный адрес применен.
А64	A64	64-разрядный адрес применен.
КЕШ	CACHE	Протокол кеш-когерентности применен.
ЦАРБ	CARB	Центральный арбитр применен.
РАРБ	DARB	Распределенный арбитр применен.
Д32	D32	32-разрядные данные.
Д64	D64	64-разрядные данные.
Д128	D128	128-разрядные данные.
Д256	D256	256-разрядные данные.
ПКТn	PKTn	Способность к пакетной передаче на тактовой частоте *n* МГц. Если используется более чем одна частота, то должен быть список ПКТn.

ППС	MSG	Протокол передачи сообщений применен.
ЗДЧК	MSTR	Способность становиться задатчиком.
МАОБЗ	MSLOCK	Способность обслуживать команды запрещения маски и обмена.
СРОБЗ	CSLOCK	Способность обслуживать команды запрещения сравнения и обмена.
ВЫСУС	FABLOCK	Способность обслуживать команды выборки и суммирования со старшего.
ВЫСУМ	FALLOCK	Способность обслуживать команды выборки и суммирования с младшего.
РСЩП	SPLT	Способность обслуживать команды расщепленной передачи.
РСЩПИ	SPLTING	Способность обслуживать и инициировать команды расщепленной передачи.
ТЕГn	TAGn	Применены линии тегов. Количество линий тегов есть *n*.

Модуль, способный стать задатчиком магистрали, с 64-разрядными данными, 32-разрядным адресом, механизмом когерентного кеша и способный обслуживать расщепленные передачи, будет иметь мнемонику: А32/КЕШ/Д64/ЗДЧК/РСЩП.

3 СПЕЦИФИКАЦИЯ СИГНАЛОВ МАГИСТРАЛИ

3.1 **Описание**

Эта глава описывает и определяет спецификацию сигналов, с которыми должны работать средства ФБ+. Эти спецификации независимы от используемого типа логики: ФБ+ может быть использован с любой логикой (ТТЛ, BTL-логикой магистрали ФБ+, ЭСЛ, КМОП или GaAs), обеспечивающей условие переключения логических уровней и определенные ограничения на встречающиеся перекосы сигналов. Для обеспечения совместимости между изделиями внутри набора, описываемого разновидностью ФБ+, конкретный тип логики может быть привязан к определенной разновидности IEEE P896.2.

3.1.1 **Переключение логических уровней**

Требования к переключению логических уровней имеются для всех ФБ+ систем. Требования заключаются в следующем: значение тока, напряжения и характеристического импеданса для всех случаев нагрузки модуля должны обеспечивать выдачу перепада напряжения шинным формирователем в передающую линию, превышающего порог переключения приемника для обеспечения правильного приема как при положительном, так и при отрицательном перепадах напряжения, с тем, чтобы отражения не превышали порога срабатывания приемника. Обычно это определяется отношением токовой способности шинного формирователя к емкости линии магистрали, когда формирователь выключен.

3.1.2 **Перекосы**

Перекосы — это расхождение по времени двух или более сигналов, которые могут идти разными путями, но имеют общий источник и пункт назначения. Никогда нельзя гарантировать, что прохождение двух сигналов будет совершенно одинаковым. Поэтому протокол шины должен всегда допускать наличие перекосов (гонок) между сигналами.

Критический перекос в системе ФБ+ измеряется между информационными сигналами и синхронизирующим сигналом, определяющим их действительность на магистрали. Разработчик передающего модуля должен гарантировать, что передаваемые информационные сигналы прибудут в место соединения с магистралью раньше, чем соответствующий синхронизирующий сигнал. Эти требования к перекосам должны учитывать все перекосы, вносимые как собственно шинными формирователями, так и средой распространения сигналов.

Кроме того, разработчик среды распространения сигналов должен гарантировать, что перекосы, вносимые емкостной нагрузкой, не задерживают информационные сигналы больше, чем стробирующие. Модуль, который совершенно пассивен логически, тем не менее активен электрически (с точки зрения внесения перекосов на магистрали). Длина печатных подводов, переходные отверстия и собственно шинные формирователи — все это вносит дополнительную нагрузку.

Разработчик приемного модуля должен также задержать синхронизирующий сигнал с учетом любых перекосов в приемниках при получении информации. Это будет гарантировать то, что причинная связь и действующие соотношения между данными сохранятся, так как сигналы имеют необходимое время предустановки внутри приемного модуля.

3.2 **Спецификация**

3.2.1 **Перекосы**

Магистраль должна вносить перекосы так, чтобы задержки распространения сигнальных линий левого оператора отношений в соотношениях 3.1—3.8 были больше или равны задержке распространения сигнальных линий правого оператора отношений в этих соотношениях.

(AS★, DS★)	> = (AD[]★, D[]★, BP[]★, TG[]★, CM[]★, CP★)	(3.1)
AK★	> = ST[]★	(3.2)
AI★	> = CA[]★, ST[]★	(3.3)
(DK★, DI★)	> = (AD[]★, D[]★, BP[]★, TG[]★, TP★, ST[]★)	(3.4)
(AP★, AQ★, AR★)	> = (AB[]★, AC[]★, ABP★)	(3.5)
DS★	> = DK★	(3.6)
AS★	> = (AK★, AI★)	(3.7)
GR★	> = PE★	(3.8)

Все модули должны ограничивать перекос, вносимый емкостной нагрузкой так, чтобы относительная нагрузка сигнальных линий модулем отвечала неравенствам 3.1—3.8.

3.2.2 **Фильтры «шпилек»**

Должны быть обеспечены фильтры «шпилек», если необходимо отфильтровать основной дефект линий передачи, называемый «шпильками проводного ИЛИ». Предопределено, что они потребуются для AK★, AI★, DK★, DI★, AP★, AQ★, AR★ и RE★ Эта спецификация написана в предположении, что они необходимы. Документ, определяющий электрические характеристики, будет устанавливать, понадобятся ли они для частных вариантов. Например 896.2 предписывает использование фильтров «шпилек» в системах, использующих BTL-логику магистрали ФБ+.

4 ЦЕНТРАЛИЗОВАННЫЙ АРБИТРАЖ

4.1 **Описание**

Когда модулю необходимо послать данные или получить данные от другого модуля, он должен сначала овладеть параллельной магистралью. Поскольку два или более модуля могут пытаться получить доступ к магистрали в одно и то же время, используется процедура арбитража для ограничения доступа к магистрали одним модулем в одно время.

Схема центрального арбитра включает в себя основной набор сигналов запрос/предоставление от каждого запрашивающего модуля к средствам центрального арбитра, который разбирается с одновременными запросами и упорядочивает предоставления запрашивающим модулям.

Эта глава описывает простой протокол централизованного арбитража, который может быть использован модулями для получения доступа к магистрали. Это обеспечивает высокую скорость выполнения команд и малое время ожидания операций запрос/предоставление.

4.1.1 **Линии магистрали для централизованного арбитража**

4.1.1.1 **RQ0★ Запрос 0**

Модуль выставляет rq0★ для запроса на владение магистралью в одном из двух его приоритетов. Модули удерживают rq0★ выставленным до последней передачи при их владении магистралью.

Модули могут выставлять rq0★ даже если ими выставлен rq1★. Эта возможность позволяет увеличить приоритет их запроса. Модуль удерживает оба сигнала rq0★ и rq1★, выставленными до тех пор, пока он не станет задатчиком на магистрали и не перейдет к своей последней передаче по параллельной магистрали или пока им не будет получен сигнал GR★, а внутренний запрос снят. Модули отпускают rq0★ и rq1★ одновременно.

4.1.1.2 **RQ1★ Запрос 1**

Модуль выставляет rq1★ для запроса на владение магистралью в одном из двух приоритетов. Модули удерживают rq1★ выставленным до последней передачи при их владении магистралью.

Модули могут выставлять rq1★, даже если ими выставлен rq0★. Эта возможность позволяет увеличить приоритет их запроса. Модуль удерживает оба сигнала rq0★ и rq1★, выставленными до тех пор, пока он не станет задатчиком на магистрали и не перейдет к своей последней передаче по параллельной магистрали или пока им не будет получен сигнал GR★, а внутренний запрос снят. Модули отпускают rq0★ и rq1★ одновременно.

Модули не должны выполнять передачи с низким приоритетом во время владения магистралью по запросу с высоким приоритетом. Модули должны отпустить магистраль и запросить ее снова с низким приоритетом, для того чтобы, избежать инверсии приоритета.

4.1.1.3 **GR★ Предоставление**

Центральный арбитр выставляет gr★ в ответ на установку RQ0★ или RQ1★ для того, чтобы указать модулю, что он выбран задатчиком магистрали. Модуль может начинать параллельную передачу, когда он обнаружит, что сигнал ET★ снят. Центральный арбитр удерживает выставленным gr★ до тех пор, пока оба RQ0★ и RQ1★ не будут сняты задатчиком.

4.1.1.4 **PE★ Отказывание**

Центральный арбитр выставляет pe★, чтобы указать задатчику, что существуют условия отказывания от магистрали. Центральный арбитр снимает pe★ после того, как он снимет gr★. Центральный арбитр решает любые возникающие у него проблемы метастабильности, если PE★ удерживается в течение времени близкого к тому, за которое задатчик снимает свой запрос.

Преимущественные условия могут быть определены при проектировании центрального арбитра. Центральный арбитр может выставить PE★, если имеются другие запрашивающие модули равного или более высокого приоритета. В другом варианте центральный арбитр мог бы выставить сигнал PE★, если любой другой модуль запросил магистраль.

Центральный арбитр удерживает сигнал pe★ во время включения питания и системного сброса для сигнализации главному модулю, что в системе есть центральный арбитр. Модули, не обладающие способностями главного, переводится в режим обращения к центральному арбитру через разряд ЦЕНТРАЛЬНЫЙ_АРБИТР в Общем Логическом Регистре Управления (7.1.2.2.1).

4.1.2 **Операции централизованного арбитража**

Показанное на рис. 4—1 соревнование при централизованном арбитраже происходит следующим образом.

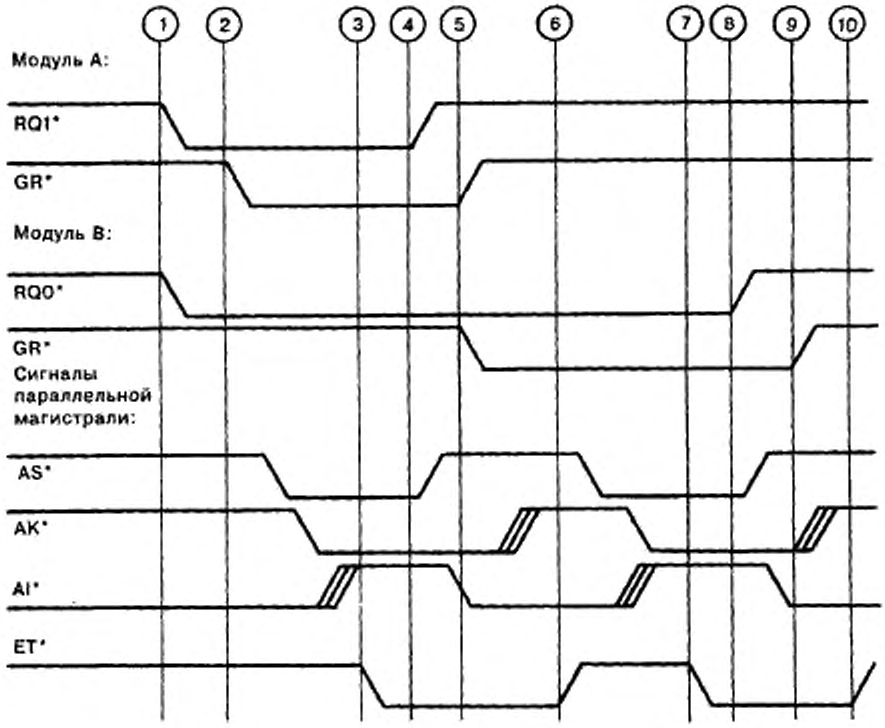

Рисунок 4—1 — Централизованный арбитраж при двух соревнующихся

1) Модули A и B выставляют каждый свой запрос на арбитраж. В данном примере запрос модуля A имеет более высокий приоритет, чем запрос модуля B.

2) Центральный арбитр выставляет gr★ модулю A.

3) Текущий задатчик — модуль А выставляет сигнал et★ после начала передачи.

4) Во время последней передачи заканчивается необходимость владения магистралью для модуля А и он снимает свой запрос.

5) После обнаружения того, что модуль А снял свой запрос, центральный арбитр снимает разрешение для модуля А и выставляет разрешение модулю В. В этот момент модуль В становится выбранным задатчиком.

6) Задатчик — модуль А после снятия сигналов с параллельной магистрали снимает et★ для сигнализации выбранному задатчику, что он — новый задатчик.

7) После начала своей передачи текущий задатчик — модуль В выставляет et★.

8) Во время последней передачи заканчивается необходимость владения магистралью для модуля В и он снимает свой запрос.

9) После обнаружения того, что модуль В снял свой запрос, центральный арбитр снимает разрешение для модуля В. До тех пор, пока нет запросов, центральный арбитр не выставляет разрешения.

10) Когда текущий задатчик — модуль В снял сигналы с параллельной магистрали, он снимает et★.

4.1.3 **Описание центрального арбитра**

Центральный арбитр должен отвечать нескольким обязательным требованиям. Так как возможны различные типы арбитров, которые будут способны к взаимодействиям, наиболее существенными требованиями являются согласованность и определение привилегий в системе в целом.

Любое число вариантов распределения магистрали может быть применено. Некоторые возможности таковы:

— простой арбитраж с единственным приоритетом — первый пришедший первым обслуживается;

— двухприоритетный арбитраж со справедливым распределением внутри каждого уровня приоритетов;

— более сложные схемы, где каждому запросу присваивается один из 256 приоритетов. Приоритеты могут быть изменены модулями, посылающими сообщения центральному арбитру, использующими возможности арбитражного сообщения.

4.2 **Спецификация**

4.2.1 **Атрибуты магистрального арбитража**

ЦЕНТРАЛИЗОВАННЫЙ ЗАДАТЧИК

Модуль должен установить ЦЕНТРАЛИЗОВАННЫЙ_ЗАДАТЧИК, когда есть—ИНИЦ & (rq0★ ¦ rq1★) & GR★ & —ET★, и поддерживать его установленным, пока есть ИНИЦ ¦ —rq0★ &— rq1★ & —et★.

4.2.2 **Сигналы магистрального арбитража**

4.2.2.1 **RQ0★**

Модуль может выставить rq0★, если есть —ИНИЦ & ЗАДАТЧИК_РАЗРЕШЕН & ЦЕНТРАЛЬНЫЙ_АРБИТР & —GR★ для запроса на обладание статусом задатчика магистрали и должен удерживать его выставленным до тех пор, пока не будет ИНИЦ ¦ GR★ & et★ (в течение последней передачи задатчиком для этого владения магистралью) ¦ GR★ и задатчик не выбирает для выполнения любую передачу в течение этого владения магистралью.

4.2.2.2 **RQ1★**

Модуль может выставить rq1★, если есть —ИНИЦ & ЗАДАТЧИК_РАЗРЕШЕН & ЦЕНТРАЛЬНЫЙ_АРБИТР & —GR★ для запроса на обладание статусом задатчика магистрали и должен удерживать его выставленным до тех пор, пока не будет ИНИЦ ¦ GR★ & et★ (в течение последней передачи задатчиком для этого владения магистралью) ¦ GR★ и задатчик не выбирает для выполнения любую передачу в течение этого владения магистралью.

4.2.2.3 **GR★**

Центральный арбитр выставляет gr★ для указания модулю, что он выбран задатчиком. Центральный арбитр должен выставлять gr★ только одному модулю в одно время. Центральный арбитр должен снять gr★ в режиме ЦЕНТРАЛИЗОВАННЫЙ_ЗАДАТЧИК, когда он обнаруживает наличие —RQ1★ & —RQ0★.

4.2.2.4 PE★

Центральный арбитр должен выставить pe★, если он обнаруживает условие для отказывания и должен удерживать его выставленным до тех пор, пока не будет снят gr★. Центральный арбитр должен снять pe★ перед выставлением gr★ для других модулей. Если центральному арбитру необходимо выставить pe★ для поочередного владения магистралью, он может проигнорировать предыдущий запрос и снять pe★, выставленный от выставления одного gr★ к следующему.

Центральный арбитр должен выставить pe★, если есть ПИТАНИЕ_ВКЛ ¦ СИСТ_СБРОС и должен удерживать его как минимум в течение 1 мкс после того, как обнаружит, что сигнал REf в нуле.

Обнаружение PE★, выставленного центральным арбитром, есть основание для метастабильности. Длительность импульса PE★ может быть короче, чем это требуется для модуля. Модули должны удовлетворять этим условиям и работать корректно.

5 РАСПРЕДЕЛЕННЫЙ АРБИТРАЖ И СООБЩЕНИЯ АРБИТРАЖА

5.1 **Описание**

Когда модулю необходимо послать данные в другой модуль или получить данные из другого модуля, он должен сначала занять параллельную магистраль. Так как два или более модуля могут одновременно претендовать на магистраль, арбитраж используется для ограничения занятия магистрали только одним модулем в одно время. ФБ+ обеспечивает пользователю на выбор два метода определения следующего действующего задатчика.

Протокол центрального арбитража описан в гл. 4. В данной главе описан протокол распределенного арбитража. Другими словами, распределенный протокол используется для широковещательных арбитражных сообщений одному или нескольким модулям, или центральному арбитру.

Так как арбитраж происходит параллельно с передачей данных и независимо от него, арбитражное соревнование за право владения магистралью или передача сообщения обычно происходят одновременно с передачами данных по магистрали.

5.1.1 **Арбитражные сообщения — центральный арбитр**

Когда используется центральный арбитр, процесс арбитража нужен исключительно для передачи арбитражных сообщений. Существуют два типа сообщений: общие сообщения арбитража и сообщения центрального арбитра. Общие сообщения обычно направлены от других модулей на магистраль, в то время как центральные сообщения используются для управления приоритетами запросов модулей в центральном арбитре.

Некоторые примеры использования таких сообщений: посылка отказа питания и других асинхронных широковещательных прерываний, асинхронные направленные прерывания, программные барьеры синхронизации, повторные события и предложения поддерживающих услуг.

5.1.1.1 **Общие арбитражные сообщения**

Модули используют номер сообщений для разрешения арбитражного соревнования. Когда два или более модуля спорят за право посылки сообщения, победителем будет тот, чей номер больше. Проигравший модуль снова будет инициировать новое арбитражное соревнование. Используемый механизм, называемый параллельным разрешением конфликта, — процесс, при котором модули выставляют свои номера сообщений на шину арбитража и снимают сигналы в соответствии с алгоритмом, который дает гарантию того, что по прошествии некоторого времени на арбитражной шине остается только номер выигравшего модуля.

Таблица 5—1 — **Поля общего арбитражного сообщения**

Общие Арбитражные Сообщения (Центральный Арбитр)								
Бит	cn7	cn6	cn5	cn4	cn3	cn2	cn1	cn0
Проход 1	1	AM6	AM5	AM4	AM3	AM2	AM1	AM0

Общие арбитражные сообщения занимают только 7 разрядов, так что единственный номер может быть выбран за одно соревнование. Каждый цикл содержит соревнование с использованием 8-разрядного соревновательного номера cn[7. . 0]. Модуль, желающий послать общее арбитражное сообщение, использует соревновательный номер, показанный в таблице 5—1. Старший разряд есть «1», он гарантирует, что общие арбитражные сообщения имеют более высокий приоритет, чем центральные сообщения.

5.1.1.2 **Центральные арбитражные сообщения**

Центральные арбитражные сообщения занимают 14 разрядов и в первую очередь предназначены для посылки информации о приоритете центральному арбитру. Центральные арбитражные сообщения требуют двух циклов соревнования для того, чтобы дать возможность использования до 256 уровней приоритета в системе. Новые запросы сообщений не могут быть инициированы между первым и вторым проходами при двухпроходных передачах сообщений. Хотя возможны несколько победителей после первого прохода, второй проход выберет единственное сообщение.

Таблица 5—2 — **Поля центрального арбитражного сообщения**

	Центральное Арбитражное Сообщение (Центральный Арбитр)							
Бит	сп7	сп6	сп5	сп4	сп3	сп2	сп1	сп0
Проход 1	0	PR7	PR6	PR5	PR4	PR3	PR2	PR1
Проход 2	1	PR0	RQ	GA4	GA3	GA2	GA1	GA0

Центральное арбитражное сообщение состоит из трех полей, как показано в табл. 5.2. Поле приоритета PR[7. . . 0] выбирает уровень приоритета линии запросов модулей. Этот уровень приоритета соотносится с линией запроса в соответствии с полем выбора запроса RQ. Если RQ равен нулю, выбирается линия RQ0★. Если RQ равен единице, выбирается RQ1★. Географическое поле адресов GA[4. . . 0] идентифицирует модуль, запрашивающий модифицированный уровень запросов. В основном модуль управляет своим собственным уровнем приоритета, но, возможно, чтобы один модуль стал ответственным за приоритеты других. Старший разряд соревнующегося кода определяет, принадлежит ли сообщение первому или последнему проходу и подтверждает, что сообщения центрального арбитража имеют меньший приоритет, чем сообщения общего арбитража.

5.1.2 **Арбитраж и сообщения — распределенный арбитр**

Системы, где нет центрального арбитра, используют протокол распределенного арбитража. В этом случае процесс арбитража используется как для запросов магистрали, так и для арбитражных сообщений. Запросы распределенного арбитража позволяют модулям овладеть магистралью. Сообщения распределенного арбитража схожи с общими арбитражными сообщениями, направленными к другим модулям на магистрали.

5.1.2.1 **Сообщения распределенного арбитража**

Таблица 5—3 — **Поля сообщений распределенного арбитража**

	Распределенное Арбитражное Сообщение (Распределенный Арбитр)							
Бит	сп7	сп6	сп5	сп4	сп3	сп2	сп1	сп0
Проход 1	1	1	1	1	1	1	1	1
Проход 2	1	AM6	AM5	AM4	AM3	AM2	AM1	AM0

Сообщения распределенного арбитража занимают 7 разрядов, включая AM[6. . . 0]. Однако они требуют двухпроходного арбитража, так как при первом проходе необходимы все логические единицы для того, чтобы отличить их от запроса распределенного арбитража и убедиться, что эти запросы не интерферируют с передачей аварийной информации. Модуль, которому необходимо выставить сообщение распределенного арбитража, использует соревновательный номер (код), приведенный в табл. 5—3. В течение второго прохода старший разряд устанавливается в единичное состояние в соответствии с форматом общего арбитражного сообщения.

5.1.2.2 **Запросы распределенного арбитража**

Величина номеров запроса распределенного арбитража используется модулями в состязании для определения числа проходов. Некоторые коды требуют двух проходов в цикле занятия управления, в то время как другие — только одного прохода. Каждый проход цикла занятия управления состоит из арбитражного соревнования, использующего 8-разрядный код состязания сп[7. . . 0]. Несколько модулей могут иметь наивысший номер после первого прохода, но после второго

прохода будет только один победитель. Это гарантировано, так как код, используемый во втором проходе, уникален. Коды арбитража, необходимые для единственного цикла занятия управления, могут свободно и динамично смешиваться с теми, которые требуют двух циклов занятия управления.

Таблица 5—4 — **Поля запросов распределенного арбитража**

Распределенные Запросы Арбитража (Распределенный Арбитр)								
Бит	cn7	cn6	cn5	cn4	cn3	cn2	cn1	cn0
Один проход								
Проход 1	1	PR0	RR	GA4	GA3	GA2	GA1	GA0
Два прохода								
Проход 1	0	PR7	PR6	PR5	PR4	PR3	PR2	PR1
Проход 2	1	PR0	RR	GA4	GA3	GA2	GA1	GA0

Код арбитража содержит три поля, как показано в таблице 5—4: поле приоритетов, поле кругового арбитража и поле географических адресов. Арбитражный код модуля отображается на один или два соревнующихся кода. Это отображение есть точно выполненное расширение и не подразумевает, что существуют два особых класса кодов арбитража. Разряд cn7 используется для индикации, запрашивает ли модуль второй проход арбитража, так что код однопроходного арбитража побеждает код двухпроходного арбитража, если оба они встретились в одном и том же цикле занятия управления.

5.1.2.2.1 **Поле приоритетов**

8-разрядное поле приоритетов, PR[7. . . 0], выделяет один из 256 уровней приоритетов для запроса. Класс приоритета модуля может быть установлен системой или модулем. Два наивысших уровня приоритета, где все разряды есть «1», за исключением PR0, используют однопроходный цикл соревнования. Это возможно, потому что, когда все PR[7. . . 1] в состоянии «1», источник запросов будет всегда выигрывать первый проход, таким образом в этом случае он может быть пропущен. Поэтому, приоритеты 254 и 255 имеют почти удвоенное соотношение к остальным приоритетам, таким образом они должны быть использованы в системах, где есть только один или два уровня приоритета.

5.1.2.2.2 **Поле кругового арбитража**

Одноразрядное поле кругового арбитража RR в сочетании с полем географических адресов определяет позицию модуля в очереди круговых запросов. Протокол кругового арбитража гарантирует справедливое и беспристрастное распределение владением магистралью для соревнующихся модулей посредством доступа всех модулей по кругу к владению магистралью в последовательности, определяемой величиной уникального поля, которое присваивается модулю.

Модули, использующие протокол кругового арбитража, мониторируют арбитражные соревнования. Значение разряда кругового арбитража модуля устанавливается каждый раз во время передачи права другому в его классе приоритетов, независимо от того, соревнуется он или нет в этом арбитражном соревновании. Разряд кругового арбитража устанавливается всякий раз, когда право на магистраль передается модулю в том же классе приоритетов, но с большим размером уникального поля. Всякий раз, когда право на магистраль передается модулю в своем классе приоритетов с меньшим размером уникального поля, разряд кругового приоритета сбрасывается.

Когда модуль овладевает магистралью и сбрасывает свой разряд кругового арбитража, это приводит к прекращению любой последовательности арбитража всех модулей с любыми географическими адресами, включая те, которые ниже. Когда все выставленные запросы от модулей с меньшими величинами географическими адресами оказываются удовлетворенными, он имеет возможность снова занять магистраль.

Если модуль, использующий протокол кругового арбитража, отказывается от магистрали в пользу модуля с высшим классом приоритета до того, как закончится его собственная передача, то он может установить разряд кругового приоритета в «1», подтверждая свое право овладения магистралью, как только это станет возможным для его приоритетного класса.

5.1.2.2.3 **Поле географических адресов**

Поле географических адресов является 5-разрядным GA[4. . . 0], что гарантирует уникальность номера арбитража. Если модуль претендует на магистраль, номер должен соответствовать географическому адресу модуля. Величина «11111» не допускается, чтобы не вызвать конфликта с широковещательной адресной схемой РУС.

5.1.2.3 **Устранение выигравшего задатчика**

При распределенном арбитраже модуль с высшим арбитражным номером в конце соревнования должен ждать до тех пор, пока задатчик не завершит своих действий на магистрали для того, чтобы он сам смог стать задатчиком. Во время такого ожидания статус модуля определен как выигравший задатчик.

Могут быть случаи, когда какой-либо модуль нуждается в срочном занятии магистрали после того, как определен выигравший задатчик, но еще до того, как последний стал действующим задатчиком. Если такой модуль имеет больший приоритет, чем выигравший задатчик, то он может инициировать новое соревнование для определения нового выигравшего задатчика. Этот процесс называется устранением выигравшего задатчика.

Устранение выигравшего задатчика позволяет модулю с более высоким приоритетом достичь владения магистралью с минимальными потерями (хотя с некоторыми жертвами с точки зрения общих системных характеристик, так как вызывается новое арбитражное соревнование).

5.1.2.4 **Отказ от передачи**

Там, где действует приоритетная схема, модуль, являющийся инициатором длинной блочной передачи или серии взаимосвязанных передач, которые могли бы осуществиться в пределах одного владения магистралью, в случае появления выигравшего задатчика из класса с более высоким приоритетом может подавить свою текущую передачу. Это предполагает наличие средств отказывания от магистрали. Задатчик получает необходимую информацию о новом выигравшем задатчике, отслеживая идентификационный номер и приоритет победителя арбитражного соревнования одновременно с передачей своей длинной посылки или серии посылок по параллельной шине.

Если действующий задатчик обнаруживает, что модуль с более высоким приоритетом стал выигравшим задатчиком, желательно, чтобы он освободил магистраль для модуля с более высоким приоритетом как можно скорее.

5.1.2.5 **Остановка**

Если нет запросов на магистраль, задатчик магистрали сохраняет управление шиной данных даже после завершения своих передач. Это состояние известно как остановка. Если у задатчика есть новый запрос в то время, пока он владеет шиной данных, он может начать новую передачу по шине данных немедленно, без инициирования цикла арбитража. Это позволяет уменьшить время простоя магистрали в некоторых типах систем.

5.1.3 **Линии магистрали**

5.1.3.1 **AP★, AQ★, AR★ Синхронизация получения управления**

По этим линиям осуществляется синхронизация модулей в течение всего процесса занятия магистрали. Эти линии функционируют во всех шести последовательных стадиях, начиная с фазы 0 и кончая фазой 5.

5.1.3.2 **AC[1. . . 0]★ Условие арбитража**

В интервале с 3 по 5 фазу модули выставляют AC0★ для сообщения об ошибке арбитража.

При распределенном арбитраже в интервале с 3 по 5 фазу модули выставляют AC1★ для запрещения передачи владения магистралью в цикле получения управления. При центральном арбитраже модули выставляют AC1★ для запрещения передачи частных центральных арбитражных сообщений, которые могут возникать в течение первого прохода при двухпроходном состязании за право владения магистралью. AC1★ всегда выставляется вместе с AC0★, если возникает ошибка. Если AC1★ выставляется один в интервале с 3 по 5 фазу, передача права владения магистралью запрещается по причинам, не связанным с обнаружением ошибки.

5.1.3.3 **AB[7. . . 0]★ Шина арбитража**

Соревнующиеся модули выставляют свои коды приоритетов cn[] на ab[]★ в течение первой фазы. Однако, если какой-либо из сигналов cn_x есть логический нуль, а сигнал на соответствующей линии магистрали AB_x★ есть логическая единица, то, пока эти условия сохраняются, все младшие разряды ab[x—1. . .0]★ обнуляются. Это производится в течение третьей фазы.

5.1.3.4 **ABP★ Четность арбитра**

Если код приоритета одного из соревнующихся четный, то он должен выставить abp★ в первой фазе. Однако, если каждый из разрядов ab[]★ сброшен, как описано в 5.1.8.3, abp★ должен быть также сброшен. Это производится в течение третьей фазы.

5.1.4 **Логика соревнования арбитража**

К.ч. показано на примерах схем (рис. 5—1а, б), если код приоритета модуля в каком-либо из разрядов нулевой, а другой модуль имеет единицу в этом разряде, то модуль, у которого присутствует логический нуль, отпускает все младшие линии ab[]★. В конце концов на линиях AB[]★ установится код модуля с наивысшим приоритетом.

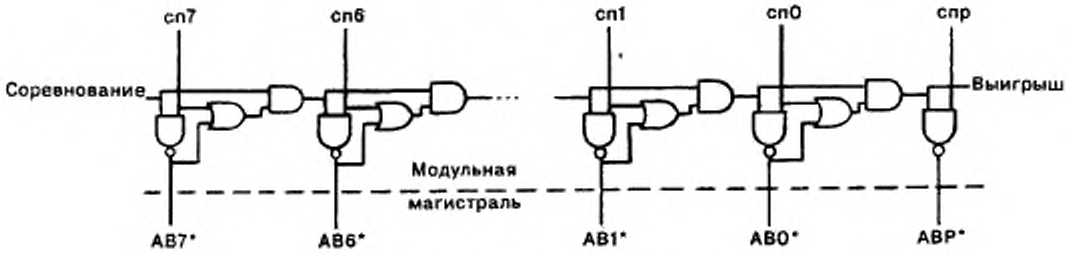

Рисунок 5—1а — Последовательная логика

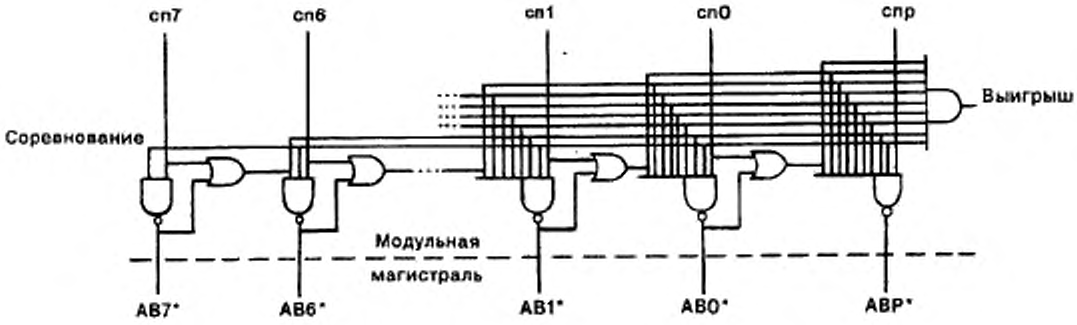

Рисунок 5—1б — Параллельная логика

В некоторых случаях модули сбрасывают сигналы на линиях ab[]★, чтобы выставить их впоследствии. На рис. 5—2 приведен пример арбитража, где три модуля одновременно начинают состязание, используя коды «00001010», «00001001» и «00000100» соответственно. AB[7. . .4]★ не показаны на рисунке, т. к. они остаются сброшенными в этом цикле арбитража.

Арбитражное соревнование на рис. 5—2 происходит следующим образом.

1) Три модуля выставляют свои коды приоритета на шину арбитража.

2) После задержки, обусловленной прохождением сигналов по магистрали, уровни на линиях магистрали AB[3. .0]★ выставляются как «проводное ИЛИ» сигналов, выставленных каждым модулем в отдельности.

3) Первый модуль, определив выставленный AB2★, снимает ab1★. Второй модуль, определив выставленные AB2★ и AB1★, снимает ab0★. Третий модуль, определив выставленный AB3★, снимает ab2★.

4) Магистраль отражает результат отпускания этих линий.

5) Первый модуль, определив отпускание AB2★, выставляет ab1★. Второй модуль, определив отпускание AB2★ и AB1★, выставляет ab0★.

6) Магистраль отражает результат выставления этих сигналов.

7) Второй модуль, определив выставленный AB1★, снимает AB0★.

8) Магистраль отражает результат отпускания этой линии.

Как показано на рис. 5—2, победивший модуль (модуль 1) отпускает ab1★, а после снова выставляет его. Модуль 1 также должен ждать, пока модуль 2 не отпустит ab0★, чтобы узнать, что он победитель. Время, требуемое для установления победившего арбитражного кода, называется временем арбитражного соревнования.

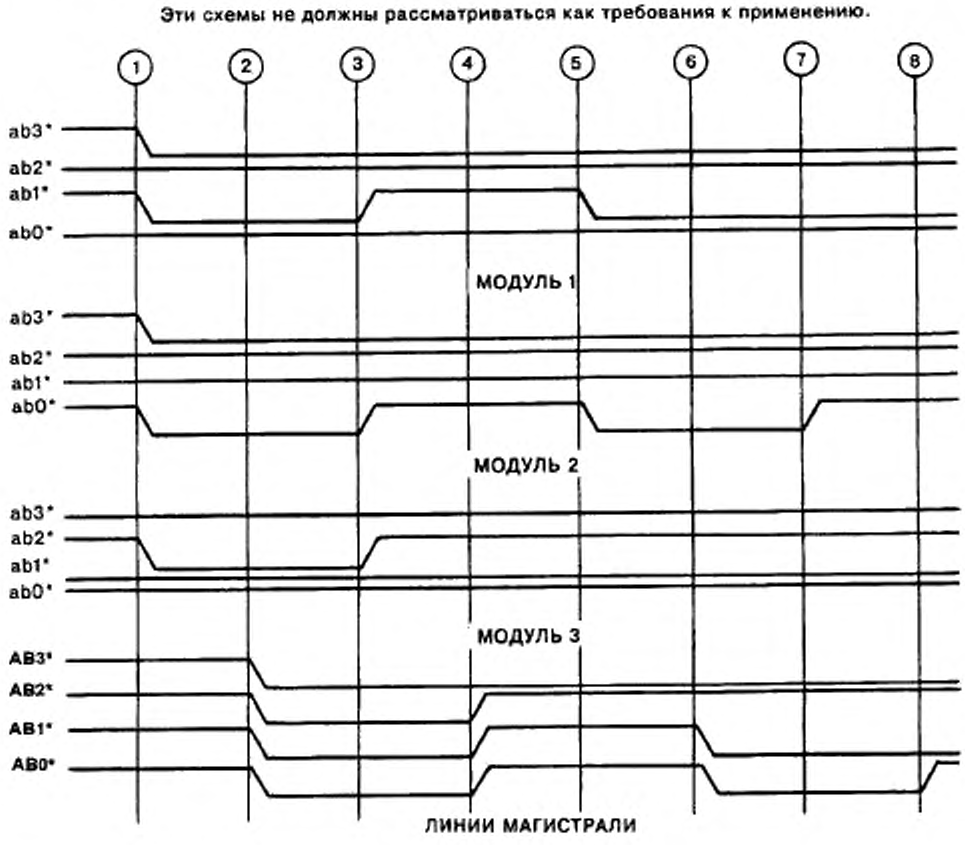

Рисунок 5—2 — Пример арбитражного соревнования

5.1.5 **Время арбитражного соревнования**

Время арбитражного соревнования является функцией физической длины магистрали, кода приоритета модуля, задержек приемопередатчиков и логики — при самых неблагоприятных условиях как внутренней логики арбитража самого модуля, так и соревнующихся с ним. Теоретическое вычисление наихудшего времени арбитражного соревнования может быть чрезвычайно сложным и не входит в рамки данного стандарта. Наихудшее время для любого кода приоритета, однако, может быть описано следующим уравнением

$$t\_a = 10\star t\_pd' + 5\star t\_ext + 4\star t\_int + t\_win.$$

Задержка в наихудшем случае может быть значительно ниже для большинства кодов приоритетов; так, если важным является быстродействие распределенного арбитра, то будет неверной попытка оптимизировать данное уравнение для запросов и сообщений каждого модуля.

Символ t_pd' используется для представления максимального времени распространения сигналов вдоль линий, которые образуют логическую магистраль. Величина t_pd' хранится в регистре задержки распространения линий передач, описанного в разделе 7.1, и является функцией максимальной длины линии передачи в применяемой системе.

Символ t_int используется для представления задержки модуля между изменением на любой арбитражной линии AB_x★ на его входе и соответствующим изменением на соседних арбитражных выходах ab[x—1]★ или abp★, где коды приоритетов выставляются или снимаются с этих разрядов.

Символ t_ext используется для представления задержки между изменением на любой линии арбитража AB_x★ на входе любого другого соревнующегося модуля и соответствующими изменениями на тех арбитражных выходах модуля ab[x—1]★ или abp★, который выставил этот код приоритета или снял эти разряды.

Символ t_win используется для представления максимальной задержки между временем, когда код на магистрали AB[7 . . .0]★ становится стабильным и эквивалетным коду cn[7. . .0] на входе модуля, давая понять модулю, что он победитель.

5.1.6 **Арбитражные состояния**

Потенциальные состояния, в которых может находиться модуль, вовлеченный в процесс арбитража, показаны на рис. 5—3 совместно со случаями, которые могут изменить статус в каждом из состояний. Рисунок представляет примеры допустимых переходов между состояниями, которые могут возникнуть как следствие цикла получения управления.

Переходы от одного состояния к другому происходят, только когда выполняются определенные условия.

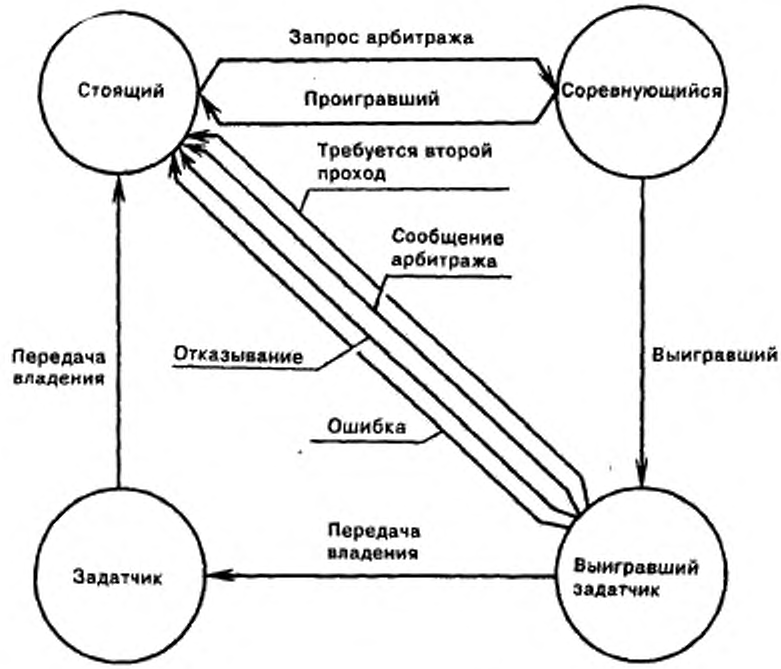

Рисунок 5—3 — Состояния занятия управления

Отметим, что этот рисунок предназначен лишь для облегчения понимания и не точно отображает диаграмму процесса получения управления.

5.1.7 **Фазы арбитража**

Шесть фаз трех синхросигналов, AP★, AQ★ и AR★, используются в течение цикла занятия управления. Обычно каждая фаза завершается так быстро, как позволяют задержка распространения и время разрешения метастабильностей. Если требуются два прохода арбитражного соревнования, то повторяется весь цикл занятия управления.

Сигналы AP★, AQ★ и AR★ являются синхронизирующими и представляют проводное ИЛИ модульных сигналов ap★, aq★ и ar★ соответственно. Любой из них может быть снят с магистрали, только если каждый модуль отпустит этот сигнал. Снятие сигнала с линии — это общая синхронизирующая точка для каждого модуля. Модуль не может перейти к следующей фазе синхронизированного арбитражного цикла, пока все модули не отпустят соответствующую арбитражную линию. Аналогичным образом, выставление какого-либо из этих сигналов показывает, что началась новая фаза, и все модули, даже если они не участвуют в данном арбитражном соревновании, должны оставаться синхронизированными, выставляя свои собственные сигналы.

Рис. 5—4 показывает последовательность изменений фаз и состояния линии.

Цикл приобретения управления состоит из шести последовательных фаз, начиная с фазы 0 и заканчивая фазой 5. Начало и конец каждой фазы определены перепадом одного из синхронизирующих арбитражных сигналов: AP★, AQ★ и AR★. Эти сигналы гарантируют, что все модули, участвуют они в соревновании или нет, остаются синхронизированными в течение всего цикла приобретения управления.

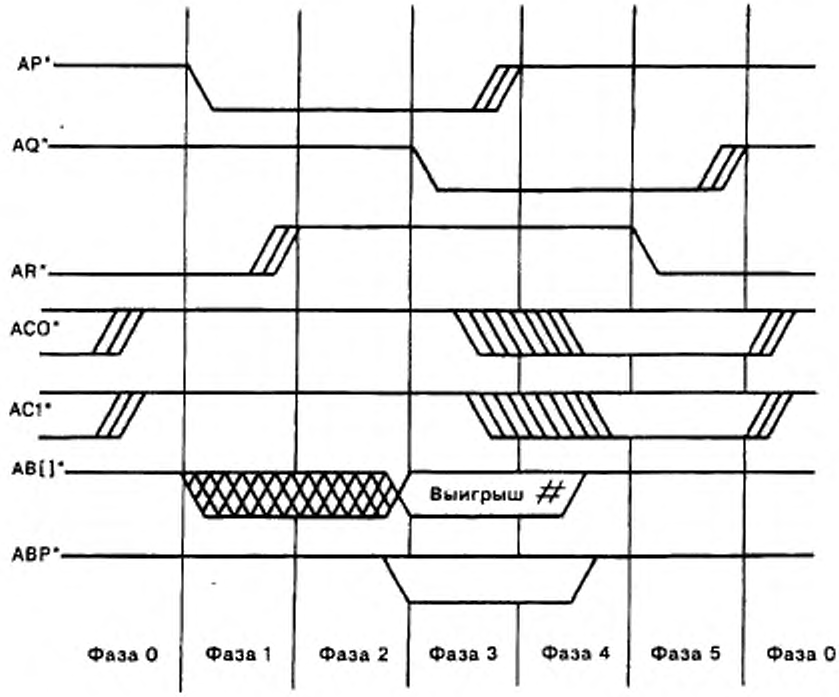

Рисунок 5—4 — Последовательность приобретения управления

5.1.7.1 **Фаза 0: Фаза ожидания**

Фаза 0 или фаза ожидания определена, когда AP★ отпущен, AQf в логическом нуле и AR★ выставлен. Это то состояние линий синхронизации арбитража, когда не происходит цикла соревнования. Это может быть и промежуток между двумя циклами соревнования в двухфазном запросе или сообщении. Модули, которые желают послать сообщение или запрос на магистраль, выставляют ap★. Это первое указание другим модулям о том, что соревнование началось.

5.1.7.2 **Фаза решения**

Фаза 1 или фаза решения определена, когда AP★ выставлен, AQ★ отпущен и AR★ выставлен. Каждый модуль отвечает на выставление AP★ выставлением ap★. Так как другие модули (не инициатор соревнования) могут принять решение посоревноваться в это же самое время, решение о соревновании — предмет метастабильности в связи с асинхронностью между опознаванием выставления AP★ и внутреннего сообщения модуля или сигнала запроса магистрали для арбитражной логики. Модули должны разрешать любые метастабильные ситуации до прекращения фазы решения.

После того, как каждый модуль разрешит любую метастабильность и сделает свой выбор — участвовать или не участвовать в соревновании, он отпускает ar★. Несоревнующиеся модули синхронизованы с арбитражным циклом и не могут инициировать новое соревнование, пока последовательность не будет окончена возвращением в фазу 0.

Ни один модуль не может снять ar★ до тех пор, пока не будут сняты сигналы ac0★ и ac1★, и, если модуль соревнующийся, это позволяет ему установить свой арбитражный номер cn[] и код четности cnp, соответственно, на ab[]★ и abp★ шины арбитража.

5 1.7.3 **Фаза 2: Фаза состязания**

Фаза 2, т. е. фаза соревнования, определяется выставлением AP★, снятием AQ★ и установлением ARf в нулевое состояние. После обнаружения начала фазы 2 соревнующиеся начинают синхронизировать время установления арбитражного соревнования.

В начале фазы 2 все модули запускают на 1--2 мкс таймер для обнаружения ошибок. Этот таймер уведомляет систему об ошибке арбитража, если это происходит до выставления AQ★. Это обеспечивает механизм исправления некоторых типов ошибок, которые в противном случае могли бы вызвать зависание магистрали.

Каждый соревнующийся модуль проверяет свой внутренний сигнал «победитель» с логики арбитражного соревнования. Поскольку соревнование является итерационным процессом, состояние этого сигнала может изменяться множество раз, пока не установится его окончательное значение. Если время арбитражного соревнования модуля заканчивается выставлением его сигнала «победитель», он изменяет свое состояние как победитель (или выбранный задатчик) и выставляет aq★.

Если же интервал обнаружения ошибки (1 мкс) заканчивается до выставления AQ★, это означает ошибку арбитража. Все модули, обнаружившие ошибки, выставляют aq★ и подтверждают, что арбитражное соревнование заканчивается без передачи владения магистралью посредством выставления и ac0★ и ac1★.

5.1.7.4 **Фаза 3: Проверка ошибки**

Фаза 3, т. е. фаза проверки ошибки, она идентифицируется выставлением сигналов AP★, AQ★ и снятием AR★. Все модули после обнаружения AQ★ выставляют aq★.

В течение фазы 3 все модули фиксируют номер (код) победителя в соревновании, полученный с магистрали, и проверяют его четность. Проигравшие модули могут также убедиться в том, что код победителя больше, чем его собственный. Любой модуль, обнаруживший ошибку, выставляет ac0★ и ac1★.

В системах с распределенным арбитражом, если победитель делает запрос магистрали, он становится выбранным задатчиком. Фаза 3 продолжается до тех пор, пока продолжается передача действующего задатчика на параллельной магистрали. Модули могут делать запросы с целью устранения выбранного задатчика выставлением ac1★ в течение фазы 3, вызывая арбитражное соревнование без передачи владения магистралью. При двухпроходном соревновании возможны предприятия попыток устранения при первом проходе. Заметим, что решение устранить выбранного задатчика является предметом для метастабильности.

В системах с распределенным арбитражом каждый модуль проверяет AS★ с целью обнаружения его снятия действующим задатчиком. Эти сигналы, которыми заканчивается передача по магистрали задатчиком, после чего становится возможной передача права владением магистралью. Когда AS★ снимается, каждый модуль снимает ap★. (Заметим, что если AS★ выставлен во время выполнения фазы 3, некоторые модули могут обнаруживать эти условия, выставляя ap★, а другие не могут. Модули, которые сняли ap★, будут приостановлены, т. е. не будут в состоянии устранить выбранного задатчика магистрали вплоть до следующего снятия AS★ или выставления AC1★).

В системах с распределенным арбитром модули, посылающие арбитражные сообщения, уведомляют, что отсутствует передача права владения магистралью, выставляя ac1★ в течение фазы 3.

В системах с центральным арбитром модули снимают ap★, как только становятся действительными данные на ac0★ и ac1★.

В том и другом случаях модуль, соревнующийся с двухпроходным номером и определивший себя как промежуточного победителя, выставляет ac1★, чтобы предотвратить преждевременную передачу сообщения или права владения магистралью. Только после завершения обоих проходов появляется единственный победитель и производится посылка сообщения или передача права владения магистралью. Все модули также проверяют AC1★ и снимают ap★, если обнаруживают выставление AC1★. Это означает, что цикл был отменен или не закончился.

5.1.7.5 **Фаза 4: Освобождение задатчиком**

Фаза 4, т. е. фаза освобождения задатчиком, идентифицируется установкой APf в логический нуль выставлением сигнала AQ★ и снятием AR★. В начале фазы 4 все модули сбрасывают ab[]★.

В системах с распределенным арбитражом все модули проверяют AC0★ и AC1★ с целью обнаружения сообщения: об ошибке, необходимости устранения, о блокировке передачи права владения магистралью.

В системах с распределенным арбитражом, если действующий задатчик хочет произвести дополнительные обмены, то он может исключить передачу права владения магистралью в фазе 4, выставив ac1★. Если он готов передать право на владение магистралью и ни один из сигналов AC0★ и AC1★ не выставлен, а также AKf находится в логическом нуле, то задатчик проверяет, что все информационные линии на параллельной магистрали освобождены. После этого он выставляет ar★, сообщая о переходе к фазе 5. Если не намечается передача права на владение магистралью, задатчик может сразу перейти к фазе 5 выставлением ar★.

В системах с распределенным арбитражом каждый модуль, кроме действующего задатчика, включает таймер на 1 мкс после обнаружения APf в состоянии логического нуля. Затем он следит за AR★, ожидая его установки задатчиком. После обнаружения установки AR★ модуль должен остановить таймер и выставить ar★. Если 1 мкс интервал закончился по причине какой-либо неисправности задатчика, модуль должен выставить ar★ и закончить фазу 4.

В системах с центральным арбитром все модули сразу выставляют ar★ для выхода из фазы 4

5.1.7.6 **Фаза 5: Передача права на владение магистралью или передача сообщения**

Фаза 5, т. е. фаза передачи сообщения или права на владение магистралью, идентифицируется снятием AP★ и выставлением сигналов AQ★ и AR★. Во время фазы 5 все модули запоминают состояние сигналов AC0★ и AC1★, чтобы определить, какое событие имело место: передача сообщения, передача права на владение магистралью или передача не была осуществлена вследствие отмены или обнаружения ошибки арбитража. После запоминания статуса модули сбрасывают aq★.

В системах с распределенным арбитражом выигравший задатчик при обнаружении установки AR★ и снятия обоих сигналов AC0★ и AC1★ принимает право на владение магистралью. Если один из последних выставлен, то текущий задатчик сохраняет свое право на владение магистралью.

Если в начале фазы 5 был выставлен AC1★, то передача сообщения или права на владение шиной блокируется. Это может произойти по нижеследующим причинам.

1) Возникла ошибка арбитража.

2) Необходим второй проход соревнования до завершения цикла занятия.

3) В системе с центральным арбитром второй проход соревнования необходимо провести до завершения сообщения центрального арбитра.

4) В системе с распределенным арбитражом арбитражное сообщение должно быть посланным.

5) В системе с распределенным арбитражом действующий задатчик решил сохранить свое право на владение магистралью.

6) В системе с распределенным арбитражом новый выигравший задатчик устраняется модулем с более высоким приоритетом.

В случае двухпроходного соревнования после завершения первого прохода модуль, который не участвовал в первом цикле занятия, должен быть заблокирован от участия во втором проходе цикла занятия управления. Любой участник, который был промежуточным победителем, должен участвовать во втором проходе, используя коды приоритета, предназначенные для второго прохода двухпроходного арбитража.

После второго прохода цикла занятия управления любой модуль, который был заблокирован в начале этого цикла, может быть разблокирован и ему разрешается участвовать в следующем арбитраже.

5.1.8 **Примеры арбитража**

5.1.8.1 **Однопроходное соревнование (Централизованное или распределенное)**

Рис. 5—5 иллюстрирует однопроходное арбитражное соревнование, результатом которого является передача управления магистралью (в распределенных системах) или передача общего сообщения (в центральных системах). Заметим, что состояние сигнала AS★ для центральной арбитражной системы не играет никакой роли. Процесс происходит следующим образом.

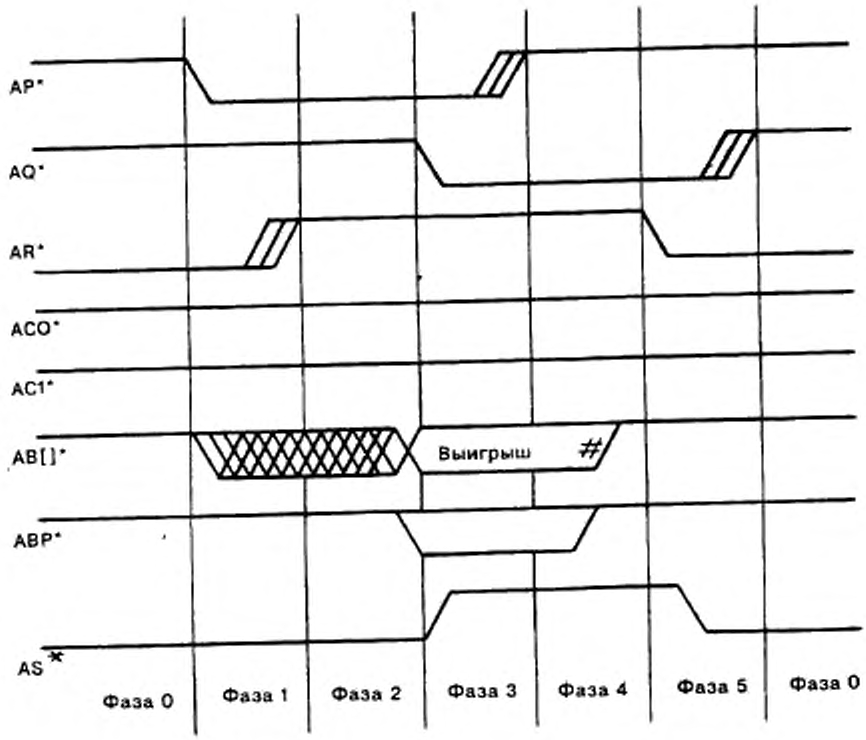

Рисунок 5—5 — Однопроходное соревнование

1) Когда сделан запрос на магистраль (распределенный или общее арбитражное сообщение) одним или несколькими модулями, эти модули выставляют ap★, начиная фазу 1.

2) Остальные модули, определив выставление AP★, принимают решение о своем участии или неучастии в соревновании и выставляют ap★.

3) Соревнующиеся модули активируют свои арбитражные номера на ab[]★, abp★, и отпускают ar★.

4) После того, как соревнующиеся модули определят наличие логического нуля на ARf, индицирующее начало фазы 2, они начинают отсчет времени установления арбитражного соревнования.

5) После установления арбитражных линий один из модулей определяет, что он является победителем, и выставляет aq★, тем самым начиная фазу 3.

6) Модули запоминают идентификатор (номер) победителя с линий AB[]★, проводят проверку отсутствия ошибок и, в случае распределенного запроса, ждут отпускания AS★.

7) В случае распределенного запроса после того, как будет отпущен AS★, все модули отпускают ap★. В случае общего сообщения запроса ap★ отпускается немедленно.

8) Когда модули определяют, что APf в логическом нуле, начинается фаза 4.

9) В случае распределенного запроса действующий задатчик, закончив последнюю передачу, выставляет ar★ и изменяет свой статус на статус задатчика, не нуждающегося в магистрали. В случае общего сообщения запроса все модули выставляют ar★ немедленно.

10) В случае распределенного запроса определение выбранным задатчиком выставленного сигнала AR★, индицирующее начало фазы 5, изменяет его статус на статус действующего задатчика и он может начинать передачи по параллельной магистрали.

11) Модули, определившие передачу управления или прохождение сообщения, отпускают aq★.

5.1.8.2 **Однопроходное соревнование с устранением (Распределенное)**

Рис. 5—6 иллюстрирует однопроходное арбитражное соревнование с устранением, отменяющее передачу права на владение магистралью (в распределенных системах). Процесс происходит следующим образом.

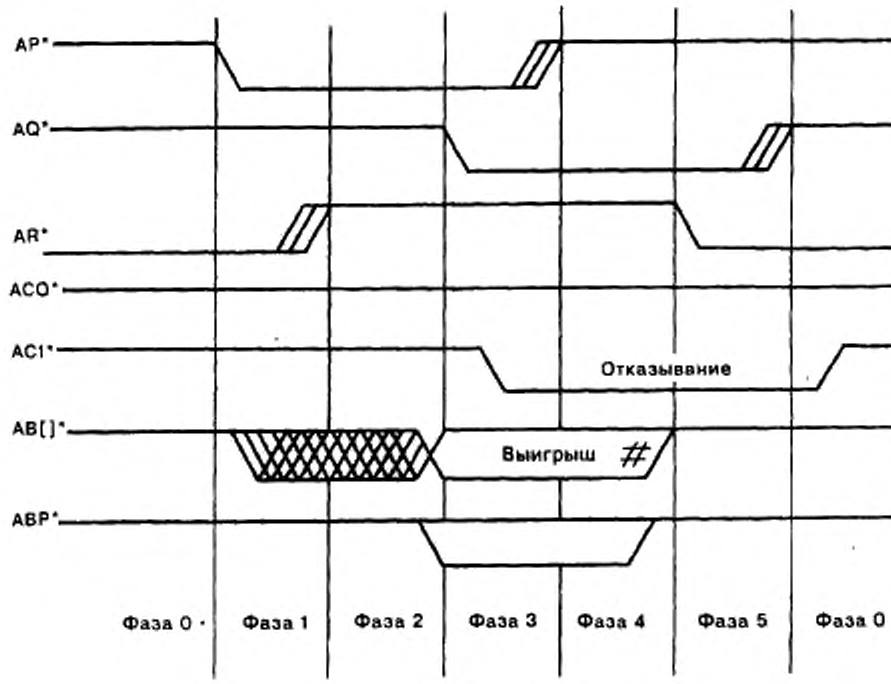

Рисунок 5—6 — Однопроходное соревнование с устранением

1) Когда сделан запрос на магистраль одним или несколькими модулями, эти модули выставляют ap★, начиная фазу 1.

2) Остальные модули, определив выставление AP★, принимают решение о своем участии или неучастии в соревновании и выставляют ap★.

3) Соревнующиеся модули активируют свои арбитражные номера на ab[]★, abp★ и отпускают ar★.

4) После того, как соревнующиеся модули определят наличие логического нуля на ARf, индицирующее начало фазы 2, они начинают отсчет времени установления арбитражного соревнования.

5) После установления арбитражных линий один из модулей определяет, что он является победителем, и выставляет aq★, тем самым начиная фазу 3.

6) Модули запоминают идентификатор (номер) победителя с линий AB[]★, проводят проверку отсутствия ошибок и ждут отпускания AS★.

7) Не участвующий в арбитраже модуль, получая (внутреннее) требование запросить магистраль, определяет, что его арбитражный номер выиграл бы предыдущее соревнование. Поэтому этот модуль выставляет ac1★ для устранения выигравшего задатчика и отпускает ap★.

8) Когда другие модули определяют, что AC1★ установлен, они отпускают ap★.

9) Когда модули определили, что AC1★ установлен и APf в логическом нуле, они выставляют ar★, чтобы начать фазу 5.

10) Затем модули определяют, что передача права на владение магистралью не состоялась, и отпускают aq★.

11) После обнаружения AQf в логическом нуле модули начинают новое арбитражное соревнование.

5.1.8.3 **Посылка сообщения распределенного арбитража**

Рис. 5—7 иллюстрирует двухпроходное арбитражное соревнование, результатом которого является посылка арбитражного сообщения. Процесс происходит следующим образом.

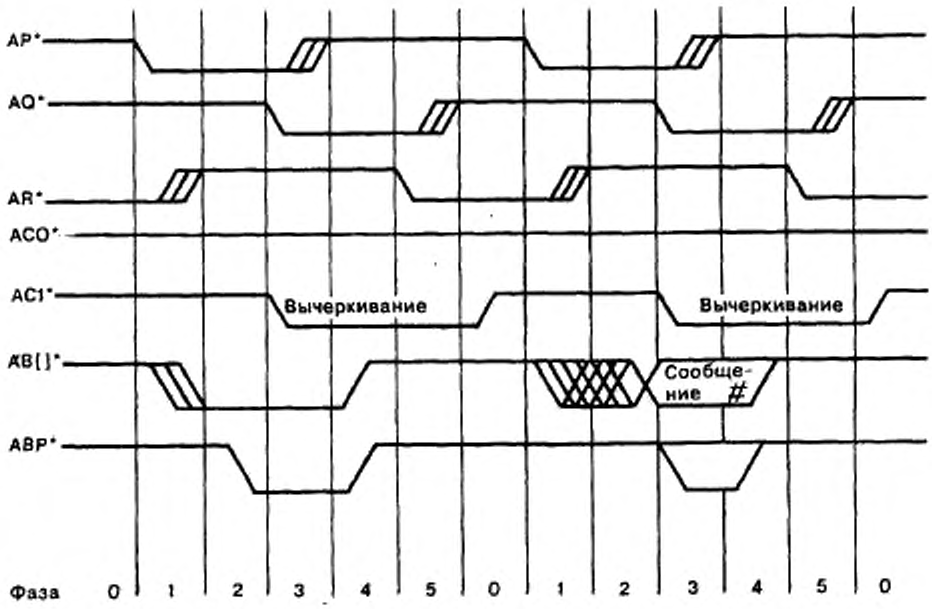

Рисунок 5—7 — Арбитражное сообщение

1) Когда сделан запрос на магистраль одним или несколькими модулями, эти модули выставляют ap★, начиная фазу 1.

2) Остальные модули, определив выставление AP★, принимают решение о своем участии или неучастии в соревновании и выставляют ap★.

3) Соревнующиеся модули активируют свои арбитражные номера на ab[]★, abp★ и отпускают ar★.

4) После того, как соревнующиеся модули определят наличие логического нуля на ARf, индицирующее начало фазы 2, они начинают отсчет времени установления арбитражного соревнования.

5) Модули, посылающие арбитражные сообщения, быстро определяют, что они победители, и выставляют aq★, тем самым начиная фазу 3.

6) Модули, посылающие арбитражное сообщение, выставляют ac1★, чтобы вызвать проведение второго прохода соревнования, и отпускают ap★.

7) Когда другие модули определяют, что AC1★ установлен, они отпускают ap★.

8) Когда модули определили, что AC1★ установлен и APf в логическом нуле, они выставляют ar★ для начала фазы 5.

9) Затем модули, определив, что передача права на владение магистралью не состоялась, и требуется провести второй проход, отпускают aq★

10) Когда модули, которые не были заблокированы проигрышем в первом проходе, определяют, что AQf в логическом нуле, они начинают новое арбитражное соревнование.

11) Соревнование затем продолжается, и выигрывает модуль с самым высоким номером арбитражного сообщения. Соревнование опять отменено, чтобы не допустить передачи права на пользование магистралью.

5.1.8.4 **Двухпроходное соревнование (Центральное или распределенное).**

Рис. 5—8 иллюстрирует двухпроходное арбитражное соревнование, результатом которого является передача права на владение магистралью (в случае распределенной системы) или передачей центрального сообщения (в случае централизованной системы). Первый проход состоит из однопроходного соревнования, отмененного победителями первого прохода. Участники, проигравшие первый проход, заблокированы от участия во втором проходе. Второй проход производится аналогично однопроходному соревнованию с передачей права владения магистралью, описанному в 5.1.8.1, за исключением того, что соревнуются только выигравшие в первом проходе. Заметим, что состояние AS★ опять несущественно в центральной арбитражной системе.

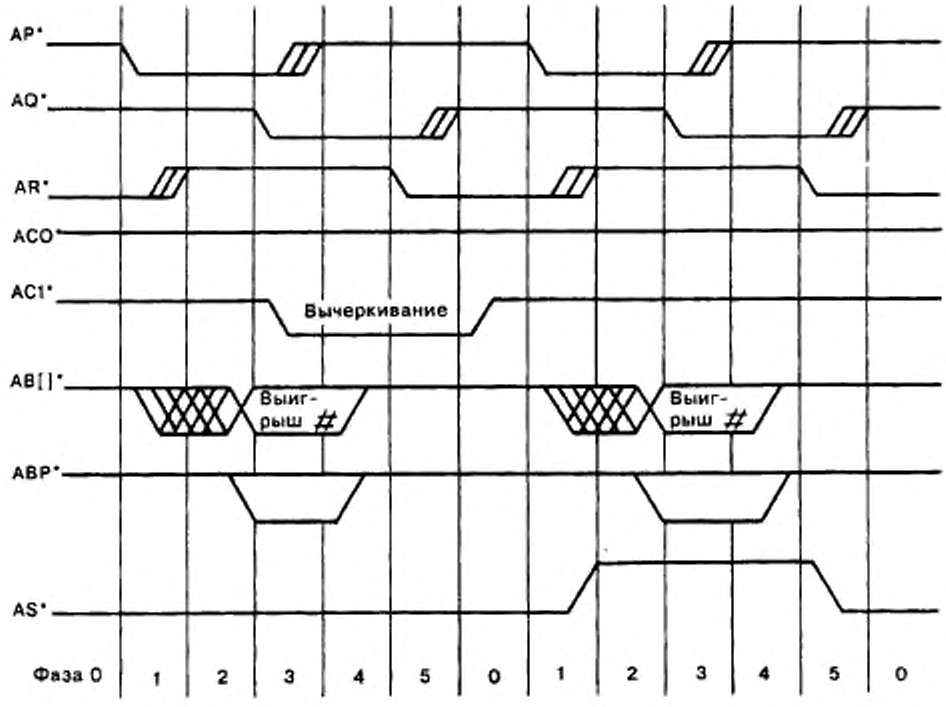

Рисунок 5—8 — Двухпроходное соревнование

5.1.8.5 **Однопроходное с ошибкой**

Рис. 5—9 иллюстрирует однопроходное арбитражное соревнование, в котором обнаруживается ошибка, что отменяет либо передачу права на владение магистралью, либо передачу сообщения. Операция выполняется нормально до тех пор, пока в фазе 3 не обнаруживается ошибка, вызываю-

щая выставление одним или несколькими модулями обоих сигналов ac0★ и ac1★. После этого выполнение операции производится как и в случае любого отмененного цикла, за исключением того, что модули приняли во внимание ошибку в фазе 5 и выполняют соответствующее действие по восстановлению после ошибки.

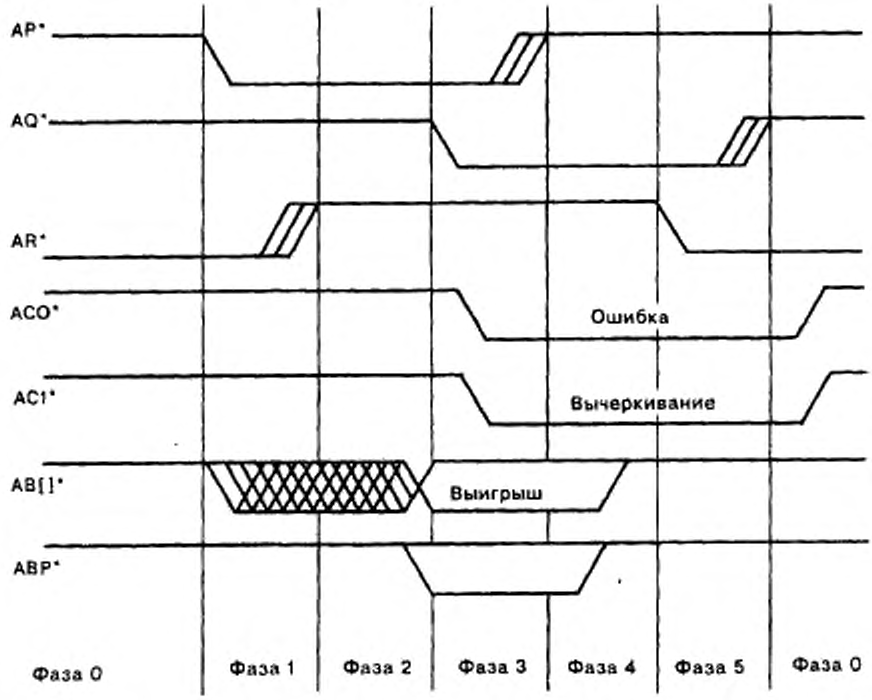

Рисунок 5—9 — Однопроходное с ошибкой

5.2 **Спецификация**

5.2.1 **Атрибуты арбитражного сообщения — центральный арбитр**

ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ

Модули могут установить ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ при условии: ЦЕНТРАЛЬНЫЙ_АРБИТР & —ИНИЦ & РАСПРЕДЕЛЕННОЕ_СООБЩЕНИЕ_РАЗРЕШЕНО & —УДЕРЖАНИЕ_МАГИСТРАЛИ & —ПОСЫЛКА_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & —ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & —ВТОРОЙ_ПРОХОД & ФАЗА_0 & —ар★, чтобы послать арбитражное сообщение. Модули должны сбросить ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ при условии: —ЦЕНТРАЛЬНЫЙ_АРБИТР | (ИНИЦ | —УДЕРЖАНИЕ_МАГИСТРАЛИ | —ПОСЫЛКА_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ | РАСПРЕДЕЛЕННОЕ_СООБЩЕНИЕ_РАЗРЕШЕНО) & —ВТОРОЙ_ПРОХОД & ФАЗА_0 & —ар★.

Решение об установке ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ является предметом метастабильности. Модули не должны снимать аг★ в течение времени, достаточного для разрешения этого условия.

ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ

Модули могут установить ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ при условии: ЦЕНТРАЛЬНЫЙ_АРБИТР & —ИНИЦ & РАСПРЕДЕЛЕННОЕ_СООБЩЕНИЕ_РАЗРЕШЕНО & —УДЕРЖАНИЕ_МАГИСТРАЛИ & —ПОСЫЛКА_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & —ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ & —ВТОРОЙ_ПРОХОД & ФАЗА_0 & —ар★, чтобы послать арбитражное сообщение. Модули должны сбросить ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБ-

ЩЕНИЯ при условии: —ЦЕНТРАЛЬНЫЙ АРБИТР | (ИНИЦ | –УДЕРЖАНИЕ_МАГИСТРАЛИ | —ПОСЫЛКА_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ | РАСПРЕДЕЛЕННОЕ_СООБЩЕНИЕ_РАЗРЕШЕНО) & —ВТОРОЙ_ПРОХОД & ФАЗА_0 & —ар★.

Решение об установке ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ является предметом метастабильности. Модули не должны снимать аг★ в течение времени, достаточного для разрешения этого условия.

ПОСЫЛКА_АРБИТРАЖНОГО_СООБЩЕНИЯ

Модуль должен установить ПОСЫЛКА_АРБИТРАЖНОГО_СООБЩЕНИЯ при условии: ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ & ВЫИГРАВШИЙ & —АС0★ & ФАЗА_5. Модуль должен сбросить ПОСЫЛКА_АРБИТРАЖНОГО_СООБЩЕНИЯ при условии: —ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ.

ПОСЫЛКА_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ

Модуль должен установить ПОСЫЛКА_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ при условии: ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & ВЫИГРАВШИЙ & —ТРЕБУЕТСЯ_ВТОРОЙ_ПРОХОД & —АС0★ & ФАЗА_5. Модуль должен сбросить ПОСЫЛКА_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ при условии: —ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ.

PR[7 . . 0]

Модуль может загрузить 8-разрядный код приоритета в PR[7 . . . 0] для смены уровней приоритета центрального арбитра при условии: —ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ.

RQ

Модули могут установить RQ для смены уровня приоритета линии запроса центрального арбитра RQ1★. Модули могут сбросить RQ для смены уровня приоритета линии запроса центрального арбитра RQ0★. Любые изменения RQ могут производиться только при условии : —ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ.

AM[13 . . 0]

Модуль должен загрузить 7_разрядный код арбитражного сообщения в AM[6 . . . 0] при условии: (ФАЗА_0 | ФАЗА_1) & —ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ & аг★. Модуль должен загрузить 14-разрядный код арбитражного сообщения в AM[13 . . 0] при условии: - ВТОРОЙ_ПРОХОД & (ФАЗА_0 | ФАЗА_1) & –ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & аг★; при этом код AM[13 . . . 6] должен равняться коду PR[7 . . 0], AM5 должен быть равен RQ и код AM[4 . . 0] должен равняться коду GA[4 . . 0].

5.2.2 **Атрибуты арбитражного сообщения — распределенный арбитр**

ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ

Модули могут установить ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ при условии: —ЦЕНТРАЛЬНЫЙ_АРБИТР & —ИНИЦ & РАСПРЕДЕЛЕННОЕ_СООБЩЕНИЕ_РАЗРЕШЕНО & —УДЕРЖАНИЕ_МАГИСТРАЛИ & —ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ & —ПОСЫЛКА_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ & —ВТОРОЙ_ПРОХОД & ФАЗА_0 & —ар★, чтобы послать арбитражное сообщение. Модули должны сбросить ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ при условии: ЦЕНТРАЛЬНЫЙ_АРБИТР | (ИНИЦ | – РАСПРЕДЕЛЕННОЕ_СООБЩЕНИЕ_РАЗРЕЩЕНО | УДЕРЖАНИЕ_МАГИСТРАЛИ . ПОСЫЛКА_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ) & —ВТОРОЙ_ПРОХОД & ФАЗА_0 & —ар★.

Решение об установке ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ является предметом метастабильности. Модули не должны снимать аг★ в течение времени, достаточного для разрешения этого условия.

ПОСЫЛКА_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ

Модуль должен установить ПОСЫЛКА_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ при условии: ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ & ВЫИГРАВШИЙ & —ЗАПРОС_ВТОРОГО_ПРОХОДА & —АС0★ & ФАЗА_5. Модуль должен сбросить ПОСЫЛКА_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ при условии: —ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ.

РАСПРЕДЕЛЕННОЕ_СООБЩЕНИЕ

Модуль должен установить РАСПРЕДЕЛЕННОЕ_СООБЩЕНИЕ при условии: ФАЗА_3 & —ВТОРОЙ_ПРОХОД & АВ7★& АВ6★& АВ5★ & АВ4★& АВ3★& АВ2★& АВ1★& АВ0★. Модуль должен удерживать РАСПРЕДЕЛЕННОЕ_СООБЩЕНИЕ до тех пор, пока: ИНИЦ | ФАЗА_0 | —ВТОРОЙ_ПРОХОД.

AM[6 . . 0]

Модуль должен загрузить 7-разрядный код арбитражного сообщения на AM[6 . . 0] при условии: (ФАЗА_0 , ФАЗА_1) & ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ & аг★.

5.2.3 **Атрибуты арбитража — распределенный арбитр**

ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРАЖА

Для получения права на владение магистралью модули могут установить ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРАЖА при условии: —ЦЕНТРАЛЬНЫЙ_АРБИТР & —ИНИЦ & РАЗРЕШЕНИЕ_ЗАДАТЧИКА & —УДЕРЖАНИЕ_МАГИСТРАЛИ & —ДЕЙСТВУЮЩИЙ_ЗАДАТЧИК & - ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ & -ВТОРОЙ_ПРОХОД & ФАЗА_0 & —ар★. Модули должны сбросить ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРАЖА при условии: ЦЕНТРАЛЬНЫЙ АРБИТР (ИНИЦ ¦ РАЗРЕШЕНИЕ_ЗАДАТЧИКА ¦ УДЕРЖАНИЕ_МАГИСТРАЛИ ¦ РАСПРЕДЕЛЕННЫЙ_ЗАДАТЧИК) & —ВТОРОЙ_ПРОХОД & ФАЗА_0 & —ар★.

Решение об установке ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРАЖА является предметом метастабильности. Модули не должны снимать аг★ в течение времени, достаточного для разрешения этого условия.

УСТРАНЕНИЕ

Если модулю необходимо послать арбитражное сообщение или он хочет получить право владения магистралью при помощи использования более высокого кода приоритета (PR[7 . . . 0], RR, GA[4 . . . 0]) по сравнению с текущим выигравшим, он может установить УСТРАНЕНИЕ при условии: —ЦЕНТРАЛЬНЫЙ_АРБИТР & —ИНИЦ & РАЗРЕШЕНИЕ_ЗАДАТЧИКА & —УДЕРЖАНИЕ_МАГИСТРАЛИ & ФАЗА_3 & ар★. Модули должны удерживать УСТРАНЕНИЕ до ФАЗА_0

Решение об установке УСТРАНЕНИЕ является предметом метастабильности. Модули не должны снимать ар★ в течение времени, достаточного для разрешения этого условия.

ОТСУТСТВИЕ_ВЛАДЕЛЬЦА_АРБИТРАЖА

Модули должны установить ОТСУТСТВИЕ_ВЛАДЕЛЬЦА_АРБИТРАЖА при ИНИЦ и удерживать его до ФАЗА_5 & —АС1.

ВЛАДЕЛЕЦ_АРБИТРАЖА

Модуль должен установить ВЛАДЕЛЕЦ_АРБИТРАЖА при условии: —ИНИЦ & (ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРАЖА & ВЫИГРАВШИЙ & —ТРЕБУЕТСЯ_ВТОРОЙ_ПРОХОД & —АС1★) & ФАЗА_5 и удерживать его установленным до: ИНИЦ ¦ (—ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРАЖА & --ТРЕБУЕТСЯ_ВТОРОЙ_ПРОХОД & —АС1★ ¦ ¦ ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРАЖА & -ВЫИГРАВШИЙ & —ЗАПРОС_ВТОРОГО_ПРОХОДА & —АС1★) & ФАЗА_5.

РАСПРЕДЕЛЕННЫЙ ЗАДАТЧИК

Модуль должен установить РАСПРЕДЕЛЕННЫЙ_ЗАДАТЧИК при условии: ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРАЖА & (ВЫИГРАВШИЙ & -ЗАПРОС_ВТОРОГО_ПРОХОДА & —АС1★ & ФАЗА_5 ¦ ВЛАДЕЛЕЦ_АРБИТРАЖА). Модуль должен сбросить РАСПРЕДЕЛЕННЫЙ_ЗАДАТЧИК при условии: —ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРАЖА & —ст★.

ДВА_ПРОХОДА

Модуль должен установить ДВА_ПРОХОДА при условии: —(PR7★ & PR6★ & PR5★ & PR4★ & PR3★ & PR2★ & PR1★).

PR[7 . . . 0]

Модуль должен загрузить 8-разрядный код приоритета в PR[7 . . . 0] при условии. -ВТОРОЙ_ПРОХОД & (ФАЗА_0 ¦ ФАЗА_1) & ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА & аг★.

КРУГОВОЙ_АРБИТРАЖ

Модули должны либо удерживать атрибут КРУГОВОЙ_АРБИТРАЖ для каждого класса приоритетов, либо, если классы со множественными приоритетами объединены единственным атрибутом КРУГОВОЙ_АРБИТРАЖ, должны сбрасывать КРУГОВОЙ_АРБИТРАЖ каждый раз, когда производится смена класса приоритета, связанного с атрибутом КРУГОВОЙ_АРБИТРАЖ.

Модуль должен установить КРУГОВОЙ_АРБИТРАЖ для класса приоритета при условии: —ИНИЦ & СОЗДАТЕЛЬ_УНИКАЛЬНОСТИ & ФАЗА_5 & -АС1★ и должен удерживать его до: ИНИЦ ¦ —СОЗДАТЕЛЬ_УНИКАЛЬНОСТИ & ФАЗА_5 & —АС1★. Модуль может удерживать КРУГОВОЙ АРБИТРАЖ установленным, если получено: СТАТУС_ЗАНЯТОСТИ ¦ СТАТУС_ОЖИДАНИЯ.

GA[4 . . . 0]

5-разрядный код GA[4 . . . 0] должен выставляться на GA[4 . . . 0]★.

УНИКАЛЬНОСТЬ_БОЛЬШЕ

Модули должны устанавливать УНИКАЛЬНОСТЬ_БОЛЬШЕ при условии: ФАЗА_3 & —РАСПРЕДЕЛЕННОЕ_СООБЩЕНИЕ & AB7★ и значении кода на AB[4 . . . 0]★ большем, чем GA[4 . . . 0] и уровне приоритета таком же, как и у текущего выигравшего. Модули должны удерживать УНИКАЛЬНОСТЬ_БОЛЬШЕ до: ФАЗА_0.

5.2.4 **Атрибуты общего арбитража и сообщения**

СОРЕВНУЮЩИЙСЯ

Модуль должен установить СОРЕВНУЮЩИЙСЯ при условии: —ИНИЦ & —ВТОРОЙ_ПРОХОД & (ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ | ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ | ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА | ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ) & (ФАЗА_0 | ФАЗА_1) & —ar★, модуль должен удерживать его выставленным до: ИНИЦ | (AC0★ | —ТРЕБУЕТСЯ_ВТОРОЙ_ПРОХОД | —ВЫИГРАВШИЙ) & ФАЗА_5.

ВЫЧЕРКНУТЫЙ

Модули должны установить ВЫЧЕРКНУТЫЙ при условии: ФАЗА_3 & (УСТРАНЕНИЕ | ЗАПРОС_ВТОРОГО_ПРОХОДА & СОРЕВНУЮЩИЙСЯ | ОШИБКА_АРБИТРАЖА | РАСПРЕДЕЛЕННОЕ_СООБЩЕНИЕ) | ФАЗА_4 & ВЛАДЕЛЕЦ_АРБИТРАЖА & НАЧАЛО_ПЕРЕДАЧИ. Модули должны сбрасывать ВЫЧЕРКНУТЫЙ при: ФАЗА_0.

CN7

Модуль должен установить cn7 при условии: ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ | ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & ВТОРОЙ_ПРОХОД | ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА & (—ДВА_ПРОХОДА | ВТОРОЙ_ПРОХОД) | ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ.

CN6

Модуль должен установить cn6 при условии: ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ & AM6 | ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & (—ВТОРОЙ_ПРОХОД & AM13 | ВТОРОЙ_ПРОХОД & AM6) | ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА & ((—ДВА_ПРОХОДА | ВТОРОЙ_ПРОХОД) & PR0 | ДВА_ПРОХОДА & ВТОРОЙ_ПРОХОД & PR7) | ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ & (—ВТОРОЙ_ПРОХОД | ВТОРОЙ_ПРОХОД & AM6).

CN5

Модуль должен установить cn5 при условии: ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ & AM5 | ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & (—ВТОРОЙ_ПРОХОД & AM12 | ВТОРОЙ_ПРОХОД & AM5) | ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА & ((—ДВА_ПРОХОДА | ВТОРОЙ_ПРОХОД) & КРУГОВОЙ_АРБИТРАЖ | ДВА_ПРОХОДА & ВТОРОЙ_ПРОХОД & PR6) | ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ & (—ВТОРОЙ_ПРОХОД | ВТОРОЙ_ПРОХОД & AM5).

CN4

Модуль должен установить cn4 при условии: ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ & AM4 | ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & (—ВТОРОЙ_ПРОХОД & AM11 | ВТОРОЙ_ПРОХОД & AM4) | ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА & ((—ДВА_ПРОХОДА | ВТОРОЙ_ПРОХОД) & GA4 | ДВА_ПРОХОДА & —ВТОРОЙ_ПРОХОД & PR5) | ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ & (—ВТОРОЙ_ПРОХОД | ВТОРОЙ_ПРОХОД & AM4).

CN3

Модуль должен установить cn3 при условии: ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ & AM3 | ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & (—ВТОРОЙ_ПРОХОД & AM10 | ВТОРОЙ_ПРОХОД & AM3) | ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА & ((—ДВА_ПРОХОДА | ВТОРОЙ_ПРОХОД) & GA3 | ДВА_ПРОХОДА & —ВТОРОЙ_ПРОХОД & PR4) | ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ & (—ВТОРОЙ_ПРОХОД | ВТОРОЙ_ПРОХОД & AM3).

CN2

Модуль должен установить cn2 при условии: ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ & AM2 | ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & (—ВТОРОЙ_ПРОХОД & AM9 | ВТОРОЙ_ПРОХОД & AM2) | ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА & ((—ДВА_ПРОХОДА | ВТОРОЙ_ПРОХОД) & GA2 | ДВА_ПРОХОДА & —ВТОРОЙ_ПРОХОД & PR3) | ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ & (—ВТОРОЙ_ПРОХОД | ВТОРОЙ_ПРОХОД & AM2).

CN1

Модуль должен установить cn1 при условии: ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ & AM1 ¦ ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & (—ВТОРОЙ_ПРОХОД & AM8 ¦ ВТОРОЙ_ПРОХОД & AM1) ¦ ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА & ((—ДВА_ПРОХОДА ¦ ВТОРОЙ_ПРОХОД) & GA1 ¦ ДВА_ПРОХОДА & —ВТОРОЙ_ПРОХОД & PR2) ¦ ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ & (—ВТОРОЙ_ПРОХОД ¦ ВТОРОЙ_ПРОХОД & AM1).

CN0

Модуль должен установить cn0 при условии: ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ & AM0 ¦ ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & (—ВТОРОЙ_ПРОХОД & AM7 ¦ ВТОРОЙ_ПРОХОД & AM0) ¦ ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА & ((—ДВА_ПРОХОДА ¦ ВТОРОЙ_ПРОХОД) & GA0 ¦ ДВА_ПРОХОДА & —ВТОРОЙ_ПРОХОД & PR1) ¦ ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ & (—ВТОРОЙ_ПРОХОД ¦ ВТОРОЙ_ПРОХОД & AM0).

CNP

Модуль должен устанавливать cnp, если номер у cn[7 . . . 0] атрибутов установлен и является четным.

ЗАПРОС_ВТОРОГО_ПРОХОДА

Модули должны устанавливать ЗАПРОС_ВТОРОГО_ПРОХОДА при условии: —ВТОРОЙ_ПРОХОД & ФАЗА_3 & (—AB7★ ¦ РАСПРЕДЕЛЕННОЕ_СООБЩЕНИЕ) и удерживать его установленным до: ФАЗА_0.

ВТОРОЙ_ПРОХОД

Модули должны устанавливать ВТОРОЙ_ПРОХОД при условии: —ИНИЦ & —ЗАПРОС_ВТОРОГО_ПРОХОДА & —AC0★ & ФАЗА_5 и удерживать его установленным до: ИНИЦ ¦ (—ЗАПРОС_ВТОРОГО_ПРОХОДА ¦ AC0★) & ФАЗА_5.

5.2.5 **Временные атрибуты арбитража**

ФАЗА_0

Модули должны устанавливать ФАЗА_0 и очищать ФАЗА_5, если все действия в ФАЗА_5 закончены и: ФАЗА_5 & (—AQf ¦ AP★). Модули должны устанавливать ФАЗА_0 при условии: ИНИЦ.

ФАЗА_1

Модули должны устанавливать ФАЗА_1 и очищать ФАЗА_0, если все действия в ФАЗА_0 закончены и: ФАЗА_0 & (ap★ ¦ AP★). Модули должны очищать ФАЗА_1 при условии: ИНИЦ.

ФАЗА_2

Модули должны устанавливать ФАЗА_2 и очищать ФАЗА_1, если все действия в ФАЗА_1 закончены и: ФАЗА_1 & (—ARf ¦ AQ★). Модули должны очищать ФАЗА_2 при условии. ИНИЦ.

ФАЗА_3

Модули должны устанавливать ФАЗА_3 и очищать ФАЗА_2, если все действия в ФАЗА_2 закончены и: ФАЗА_2 & (aq★ ¦ AQ★). Модули должны очищать ФАЗА_3 при условии: ИНИЦ.

ФАЗА_4

Модули должны устанавливать ФАЗА_4 и очищать ФАЗА_3, если все действия в ФАЗА_3 закончены и: ФАЗА_3 & (—APf★ ¦ AR★). Модули должны очищать ФАЗА_4 при условии: ИНИЦ.

ФАЗА_5

Модули должны устанавливать ФАЗА_5 и очищать ФАЗА_4, если все действия в ФАЗА_4 закончены и: ФАЗА_4 & (ar★ ¦ AR★). Модули должны очищать ФАЗА_5 при условии: ИНИЦ.

СОРЕВНОВАНИЕ

Модуль должен устанавливать СОРЕВНОВАНИЕ при условии: СОРЕВНУЮЩИЙСЯ & ФАЗА_1 и удерживать его до: ФАЗА_4.

УСТАНОВЛЕННЫЙ

Модули должны очищать УСТАНОВЛЕННЫЙ при условии: ФАЗА_0.

СОРЕВНУЮЩИЙСЯ должен установить УСТАНОВЛЕННЫЙ при условии, что прошел достаточный промежуток времени t_a с начала ФАЗА_2, гарантирующий СОРЕВНУЮЩЕМУСЯ победу в соревновании ¦ ФАЗА_3.

ПОБЕДИТЕЛЬ

Модуль должен выставить ПОБЕДИТЕЛЬ при условии: УСТАНОВЛЕННЫЙ & СОРЕВНОВАНИЕ & (cn7 ¦ —AB7★) & (cn6 ¦ —AB6★) & (cn5 ¦ —AB5★) & (cn4 ¦ —AB4★) & (cn3 ¦ —AB3★) & (cn2 ¦ —AB2★) & (cn1 ¦ —AB1★) & (cn0 ¦ —AB0★) и удерживать ПОБЕДИТЕЛЬ до: ФАЗА_0.

5.2.6 **Атрибуты ошибок арбитража**

ОШИБКА_АРБИТРАЖА

Модуль должен устанавливать ОШИБКА_АРБИТРАЖА при условии: ТАЙМ-АУТ_АРБИТРАЖА . ОШИБКА_ЧЕТНОСТИ_АРБИТРАЖА & РАЗРЕШЕНИЕ_СООБЩЕНИЯ_ЧЕТНОСТИ ¦ ОШИБКА_СРАВНЕНИЯ_АРБИТРАЖА. Модули должны очищать ОШИБКА_АРБИТРАЖА при условии: ФАЗА_0.

ОШИБКА_ЧЕТНОСТИ_АРБИТРАЖА

Модули должны устанавливать ОШИБКА_ЧЕТНОСТИ_АРБИТРАЖА в ФАЗА_3 при условии, если код, выставленный на AB[]★, является четным и очищен ABP★ или, если код, выставленный на AB[]★, является нечетным и установлен ABP★. Модули должны удерживать ОШИБКА_ЧЕТНОСТИ_АРБИТРАЖА до: ФАЗА_0.

ОШИБКА_СРАВНЕНИЯ_АРБИТРАЖА

Модули должны устанавливать ОШИБКА_СРАВНЕНИЯ_АРБИТРАЖА при условии: ФАЗА_3 & СОРЕВНУЮЩИЙСЯ & (AB[]★ <cn[]).

ТАЙМ_АУТ_АРБИТРАЖА

Модули должны устанавливать ТАЙМ-АУТ_АРБИТРАЖА при условии: (ФАЗА_2 длится более 1—2 мкс) . (ФАЗА_4 длится более 1—2 мкс) и должны удерживать его выставленным до: ФАЗА_0.

5.2.7 **Определение сигналов**

5.2.7.1 **Синхронизация арбитража**

5.2.7.1.1 **AP★**

Модули должны выставлять ap★ при условии: ФАЗА_0 (ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ ¦ ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ ¦ ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА ¦ ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ) ¦ ФАЗА_1 и удерживать ap★ до появления условия: ИНИЦ ¦ ФАЗА_3 & (ЦЕНТРАЛЬНЫЙ_АРБИТР ¦ —ЦЕНТРАЛЬНЫЙ_АРБИТР & (ac1★ ¦ AC1★ ¦ —AS★)) и модуль запомнил номер, выигравший арбитраж, и проверил его на ошибки.

5.2.7.1.2 **AQ★**

Модули должны устанавливать aq★ при условии: ФАЗА_2 & (ПОБЕДИТЕЛЬ ¦ ТАЙМ-АУТ_АРБИТРАЖА) ¦ ФАЗА_3 и удерживать установленным aq★ до появления условия: ИНИЦ ¦ ФАЗА_5 и оба AC0★ и AC1★ были выбраны.

5.2.7.1.3 **AR★**

Модули должны устанавливать ar★ при условии: —(ВКЛЮЧЕНИЕ_ПИТАНИЯ & —НЕЗАНЯТОСТЬ_МАГИСТРАЛИ_1US) & (ИНИЦ ¦ ФАЗА_4 & (ЦЕНТРАЛЬНЫЙ_АРБИТР & —ЦЕНТРАЛЬНЫЙ_АРБИТР & (ВЛАДЕЛЕЦ_АРБИТРАЖА & ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ ¦ ОТСУТСТВИЕ_ВЛАДЕЛЬЦА_АРБИТРАЖА & ПОБЕДИТЕЛЬ ¦ ТАЙМ-АУТ_АРБИТРАЖА ¦ AC1★ ¦ ac1★)) ¦ ФАЗА_5) и удерживать установленным ar★ до появления условия: ap★ ФАЗА_1.

5.2.7.2. **Условия арбитража**

5.2.7.2.1 **AC0★**

Модули должны устанавливать ac0★ при условии: ФАЗА_3 & ОШИБКА_АРБИТРАЖА. Модули должны удерживать установленным ac0★ до: ФАЗА_0.

5.2.7.2.2 **AC1★**

Модуль должен устанавливать ac1★ при условии: ВЫЧЕРКНУТЫЙ и должен удерживать установленным ac1★ до: ФАЗА_0.

5.2.7.3 **Шина арбитража**

5.2.7.3.1 **AB7★**

СОРЕВНУЮЩИЙСЯ должен устанавливать ab7★ при условии: cn7 & СОРЕВНОВАНИЕ.

5.2.7.3.2 **AB6★**

СОРЕВНУЮЩИЙСЯ должен устанавливать ab6★ при условии: cn6 & СОРЕВНОВАНИЕ & (cn7 ¦ AB7★).

5.2.7.3.3 **AB5★**

СОРЕВНУЮЩИЙСЯ должен устанавливать ab5★ при условии: cn5 & СОРЕВНОВАНИЕ & (cn7 ¦ —AB7★) & (cn6 ¦ —AB6★).

5.2.7.3.4 **AB4★**

СОРЕВНУЮЩИЙСЯ должен устанавливать ab4★ при условии: cn4 & СОРЕВНОВАНИЕ & (cn7 ¦ —AB7★) & (cn6 ¦ —AB6★) & (cn5 ¦ —AB5★).

5.2.7.3.5 **AB3★**

СОРЕВНУЮЩИЙСЯ должен устанавливать ab3★ при условии: cn3 & СОРЕВНОВАНИЕ & (cn7 ¦ —AB7★) & (cn6 ¦ —AB6★) & (cn5 ¦ —AB5★) & (cn4 ¦ —AB4★).

5.2.7.3.6 **AB2★**

СОРЕВНУЮЩИЙСЯ должен устанавливать ab2★ при условии: cn2 & СОРЕВНОВАНИЕ & (cn7 ¦ —AB7★) & (cn6 ¦ —AB6★) & (cn5 ¦ —AB5★) & (cn4 ¦ —AB4★) & (cn3 ¦ —AB3★).

5.2.7.3.7 **AB1★**

СОРЕВНУЮЩИЙСЯ должен устанавливать ab1★ при условии: cn1 & СОРЕВНОВАНИЕ & (cn7 ¦ —AB7★) & (cn6 ¦ —AB6★) & (cn5 ¦ —AB5★) & (cn4 ¦ —AB4★) & (cn3 ¦ —AB3★) & (cn2 ¦ —AB2★).

5.2.7.3.8 **AB0★**

СОРЕВНУЮЩИЙСЯ должен устанавливать ab0★ при условии: cn0 & СОРЕВНОВАНИЕ & (cn7 ¦ —AB7★) & (cn6 ¦ —AB6★) & (cn5 ¦ —AB5★) & (cn4 ¦ —AB4★) & (cn3 ¦ —AB3★) & (cn2 ¦ —AB2★) & (cn1 ¦ —AB1★).

5.2.7.3.9 **ABP★**

СОРЕВНУЮЩИЙСЯ должен устанавливать abp★ при условии: cnp & СОРЕВНОВАНИЕ & (cn7 ¦ —AB7★) & (cn6 ¦ —AB6★) & (cn5 ¦ —AB5★) & (cn4 ¦ —AB4★) & (cn3 ¦ —AB3★) & (cn2 ¦ —AB2★) & (cn1 ¦ —AB1★) & (cn0 ¦ —AB0★).

5.2.8 **Определение протокола — распределенный арбитраж и сообщения**

5.2.8.1 **ФАЗА_0 — ОЖИДАНИЕ**

Модуль должен очищать свои РАЗМЕЩЕНИЕ, УНИКАЛЬНОСТЬ_БОЛЬШЕ, ВЫЧЕРКНУТЫЙ, УСТАНОВЛЕННЫЙ, ЗАПРОС_ВТОРОГО_ПРОХОДА, ПОБЕДИТЕЛЬ, ОШИБКА_АРБИТРАЖА, ОШИБКА_ЧЕТНОСТИ_АРБИТРАЖА и ТАЙМ-АУТ_АРБИТРАЖА при условии если они установлены, а также снять свои сигналы ac0★ и ac1★, если они установлены.

Модуль должен очистить РАСПРЕДЕЛЕННОЕ_СООБЩЕНИЕ при условии —ВТОРОЙ_ПРОХОД.

ВЛАДЕЛЕЦ_АРБИТРАЖА может выставлять РАСПРЕДЕЛЕННЫЙ_МАСТЕР при условии: ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА & —ВТОРОЙ_ПРОХОД.

При условии: ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА & —ВЛАДЕЛЕЦ_АРБИТРАЖА ¦ ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ & —ПОСЫЛКА_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ модуль должен устанавливать ap★.

5.2.8.2 **ФАЗА_1 — РЕШЕНИЕ**

При условии: (ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА & —ВЛАДЕЛЕЦ_АРБИТРАЖА ¦ ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ & —ПОСЫЛКА_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ) & —ВТОРОЙ_ПРОХОД модуль должен:

1) убедиться, что закончились все переходные процессы;

2) выставить СОРЕВНУЮЩИЙСЯ;

3) перезагрузить PR[7 . . 0], КРУГОВОЙ_АРБИТРАЖ, GA[4 . . . 0], ДВА_ПРОХОДА и AM[7 . 0].

При условии: ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА & —ВЛАДЕЛЕЦ_АРБИТРАЖА ¦ ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ & —ПОСЫЛКА_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ модуль должен:

1) перезагрузить cn[7 . . . 0] и cnp;

2) выставить СОРЕВНОВАНИЕ;

3) убедиться, что прошло достаточно времени для того, чтобы старшие разряды cn[7 . . . 0] были помещены на соответствующие линии AB[7 . . . 0]★.

Все модули должны выставить ap★, если он еще не установлен.

После этого все модули должны затем снять ag★.

5.2.8.3 **ФАЗА_2 — СОРЕВНОВАНИЕ**

Если модуль остается в ФАЗА_2 больше, чем 1—2 мкс, он должен установить ТАЙМ-АУТ_АРБИТРАЖА и ОШИБКА_АРБИТРАЖА и выставить aq★.

СОРЕВНУЮЩИЙСЯ должен:

1) начать отсчет времени t_a, как описано в 5.2.5;

2) при условии УСТАНОВЛЕННЫЙ и ПОБЕДИТЕЛЬ убедиться, что abp★ действителен и затем выставить aq★.

5.2.8.4 **ФАЗА_3 — Проверка ошибок**

Все модули должны выставить aq★, если он еще не установлен.

Все модули должны:

1) запомнить коды на линиях AB[7 . . 0]★ и ABP★;

2) выставить ОШИБКА_ЧЕТНОСТИ_АРБИТРАЖА и ОШИБКА_АРБИТРАЖА при неправильной четности кодов на AB[7 . . 0] и ABP★;

3) выставить РАСПРЕДЕЛЕННОЕ_СООБЩЕНИЕ при условии: —ВТОРОЙ_ПРОХОД & AB7★ & AB6★ & AB5★ & AB4★& AB3★ & AB2★ & AB1★ & AB0★;

4) выставить УНИКАЛЬНОСТЬ_БОЛЬШЕ при условии, описанном в 5.2.3;

5) установить ЗАПРОС_ВТОРОГО_ПРОХОДА при условии: —AB7★; РАСПРЕДЕЛЕННОЕ_СООБЩЕНИЕ;

6) установить ВЫЧЕРКНУТЫЙ при условии: ОШИБКА_АРБИТРАЖА ¦ РАСПРЕДЕЛЕННОЕ_СООБЩЕНИЕ ¦ ЗАПРОС_ВТОРОГО_ПРОХОДА & СОРЕВНУЮЩИЙСЯ.

При условии РАЗМЕЩЕНИЕ модуль должен:

1) убедиться, что закончились все метастабильные переходные процессы;

2) установить ВЫЧЕРКНУТЫЙ.

Все модули должны:

1) выставить ac0★ при условии: ОШИБКА_АРБИТРАЖА;

2) выставить ac1★ при условии: ВЫЧЕРКНУТЫЙ.

Все модули должны освобождать ap★ при условии: AC1★. —AS★

5.2.8.5 **ФАЗА_4 — Освобождение от владения**

Если модуль остается в ФАЗА_4 больше, чем 1—2 мкс, он должен установить ТАЙМ-АУТ_АРБИТРАЖА и ОШИБКА_АРБИТРАЖА и выставить ar★.

СОРЕВНУЮЩИЙСЯ должен:

1) очистить СОРЕВНОВАНИЕ;

2) выставить ar★ при условии: ПОБЕДИТЕЛЬ & ОТСУТСТВИЕ_ВЛАДЕЛЬЦА_АРБИТРАЖА.

ВЛАДЕЛЕЦ_АРБИТРАЖА должен:

1) при условии: НАЧАЛО_ПЕРЕДАЧИ установить ВЫЧЕРКНУТЫЙ, выставить ac1★ и затем выставить ar★;

2) при условии: ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ выставить ar★.

Все модули должны выставить ar★ при условии: AC1★.

5.2.8.6 **ФАЗА_5 — Передача владения**

Все модули должны выставить ar★ при условии, что он не установлен.

Модуль должен снять свои сигналы с ab[]★ и abp★, если они выставлены.

Модуль должен:

1) определить код на линиях AC0★ и AC1★;

2) перезагрузить КРУГОВОЙ_АРБИТРАЖ, как описано в 5.2.1.4;

3) выставить ВТОРОЙ_ПРОХОД при условии: ЗАПРОС_ВТОРОГО_ПРОХОДА & —AC0★;

4) очистить ОТСУТСТВИЕ_ВЛАДЕЛЬЦА_АРБИТРАЖА при условии —AC1★.

СОРЕВНУЮЩИЙСЯ должен:

1) выставить ВЛАДЕЛЕЦ_АРБИТРАЖА и РАСПРЕДЕЛЕННЫЙ_МАСТЕР при условии: ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА & ПОБЕДИТЕЛЬ & —ЗАПРОС_ВТОРОГО_ПРОХОДА & —AC1★;

2) выставить ПОСЫЛКА_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ при условии: ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ & ПОБЕДИТЕЛЬ & —ЗАПРОС_ВТОРОГО_ПРОХОДА & AC0★;

3) очистить СОРЕВНУЮЩИЙСЯ при условии: —ПОБЕДИТЕЛЬ ¦ —ЗАПРОС_ВТОРОГО_ПРОХОДА ¦ AC0★.

ВЛАДЕЛЕЦ_АРБИТРАЖА должен очистить ВЛАДЕЛЕЦ_АРБИТРАЖА при условии: —ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА & —ЗАПРОС_ВТОРОГО_ПРОХОДА & –АС1★¦ ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРА & —ПОБЕДИТЕЛЬ & —ЗАПРОС_ВТОРОГО_ПРОХОДА & - -АС1★.

Все модули после этого должны освободить aq★.

5.2.9 **Определение протокола — сообщения центрального арбитра**

5.2.9.1 **ФАЗА_0 — ОЖИДАНИЕ**

Модуль должен очищать свои: ВЫЧЕРКНУТЫЙ, УСТАНОВЛЕННЫЙ, ЗАПРОС_ВТОРОГО_ПРОХОДА, ПОБЕДИТЕЛЬ, ОШИБКА_АРБИТРАЖА, ОШИБКА_ЧЕТНОСТИ_АРБИТРАЖА и ТАЙМ-АУТ_АРБИТРАЖА при условии, если они установлены, а также снять свои сигналы с ас0★ и ас1★, если они выставлены.

При условии: ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ & —ПОСЫЛКА_АРБИТРАЖНОГО_СООБЩЕНИЯ ¦ ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & —ПОСЫЛКА_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ модуль должен выставить ар★

5.2.9.2 **ФАЗА_1 — РЕШЕНИЕ**

При условии: (ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ & —ПОСЫЛКА_АРБИТРАЖНОГО_СООБЩЕНИЯ ¦ ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & —ПОСЫЛКА_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ) & —ВТОРОЙ_ПРОХОД модуль должен:

1) убедиться, что закончились все метастабильные переходные процессы;

2) выставить СОРЕВНУЮЩИЙСЯ;

3) перезагрузить PR[7 . . . 0], RQ и AM[13 . . .0].

При условии: ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ & —ПОСЫЛКА_АРБИТРАЖНОГО_СООБЩЕНИЯ ¦ ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & —ПОСЫЛКА_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ модуль должен:

3) изменить сп[7 . . . 0] и спр;

4) выставить СОРЕВНОВАНИЕ;

5) убедиться, что прошло достаточно времени для того, чтобы старшие разряды сп[7 . . . 0] были помещены на соответствующие линии AB[7 . . . 0]★.

Все модули должны выставить ар★, если он еще не установлен.

После этого все модули должны снять аг★.

5.2.9.3 **ФАЗА_2 — СОРЕВНОВАНИЕ**

Если модуль остается в ФАЗА_2 больше чем 1—2 мкс, он должен установить ТАЙМ-АУТ_АРБИТРАЖА, ОШИБКА_АРБИТРАЖА и выставить aq★.

СОРЕВНУЮЩИЙСЯ должен:

1) начать отсчет времени t_a, как описано в 5.2.5;

2) при условии: УСТАНОВЛЕННЫЙ и ПОБЕДИТЕЛЬ убедиться, что abp★ действителен и затем выставить aq★.

5.2.9.4 **ФАЗА_3 — Проверка ошибок**

Все модули должны выставить aq★, если он еще не выставлен.

Все модули должны:

1) запомнить коды на линиях AB[7 . . . 0]★ и ABP★;

2) установить ОШИБКА_ЧЕТНОСТИ_АРБИТРАЖА и ОШИБКА_АРБИТРАЖА при неправильной четности кодов на AB[7 . . 0] и ABP★;

3) установить ЗАПРОС_ВТОРОГО_ПРОХОДА при условии: —AB7★;

4) установить ВЫЧЕРКНУТЫЙ при условии: ОШИБКА_АРБИТРАЖА ¦ ЗАПРОШЕННЫЙ_ВТОРОЙ_ПРОХОД & СОРЕВНУЮЩИЙСЯ.

Все модули должны:

1) выставить ас0★ при условии ОШИБКА_АРБИТРАЖА;

2) выставить ас1★ при условии ВЫЧЕРКНУТЫЙ

Все модули должны освобождать ар★.

5.2.9.5 **ФАЗА_4 — Освобождение мастерства**

Если модуль остается в ФАЗА_4 больше, чем 1—2 мкс, он должен установить ТАЙМ-АУТ_АРБИТРАЖА, ОШИБКА_АРБИТРАЖА и выставить аг★.

СОРЕВНУЮЩИЙСЯ должен очистить СОРЕВНОВАНИЕ.

Все модули должны выставить аг★.

5.2.9.6 **ФАЗА_5 — Передача владения**

Все модули должны выставить ar★ при условии, что он не установлен.

Модуль должен снять свои сигналы с ab[]★ и abp★, если они выставлены.

Модуль должен:

1) определить код на линии AC0★;

2) выставить ВТОРОЙ_ПРОХОД при условии: ЗАПРОС_ВТОРОГО_ПРОХОДА & —AC0★.

СОРЕВНУЮЩИЙСЯ должен:

1) выставить ПОСЫЛКА_АРБИТРАЖНОГО_СООБЩЕНИЯ при условии ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ & ПОБЕДИТЕЛЬ & —AC0★;

2) выставить ПОСЫЛКА_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ при условии ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ & ПОБЕДИТЕЛЬ & —AC0★;

3) очистить СОРЕВНУЮЩИЙСЯ при условии —ПОБЕДИТЕЛЬ | —ЗАПРОШЕННЫЙ_ВТОРОЙ_ПРОХОД | AC0★.

Все модули после этого должны освободить aq★.

6 ПАРАЛЛЕЛЬНЫЙ ПРОТОКОЛ

6.1 **Описание**

Данный раздел с описанием параллельного протокола описывает и специфицирует, каким образом передаются данные в стандарте Futurebus+.

6.1.1 **Владение магистралью**

Выиграв право на управление магистралью в процессе арбитража, задатчик может использовать магистраль передачи данных для проведения передачи по магистрали между им самим и одним или более исполнителями. Управление задатчика известно как владение магистралью. Следует отметить, что в течение времени, когда задатчик владеет магистралью, он также выполняет все функции исполнителя.

Не существует ограничений на количество или величину длительности передач, которые задатчик может осуществлять вплоть до отказа от магистрали. Если задатчик выиграл право на управление магистралью, он не может быть устранен при обычных обстоятельствах. Ограничение права управления магистралью находится в ведении системного интегратора, который ограничивает право владения магистралью задатчиком с целью оптимизации полного построения системы. Профили или другие высшие стандарты будут в основном определять максимальное время владения магистралью.

Некоторые типы передач являются частью передач более высокого системного уровня. Эти системные передачи, которые включают множественные магистральные передачи, называются расщепленными передачами. Расщепленные передачи могут включать любое число магистральных передач от множества магистральных задатчиков.

6.1.2 **Передачи**

Модуль использует шину передачи данных для проведения системных передач. Системная передача состоит из запроса с последующим ответом. Существует два пути проведения системных передач.

1) Связная передача используется для проведения запросов и ответов модулей при однократной магистральной передаче.

2) Расщепленная передача используется для расщепления фаз запросов и ответов модулей в раздельные интервалы владения магистралью. Модуль, обслуживающий запрос, становится задатчиком, адресуется к источнику запроса и передает ответ.

6.1.3 **Фазы магистральной передачи**

Задатчик может проводить магистральные передачи в течение времени владения магистралью. Каждая магистральная передача состоит из трех фаз:

1) фаза подключения, при которой задатчик выбирает и устанавливает соединение с желаемыми исполнителями;

2) фаза передачи данных (по выбору), при которой данные передаются между задатчиком и подсоединенными исполнителями;

3) фаза рассоединения, при которой задатчик завершает передачу и рассоединяется с исполнителями.

Необходимо, чтобы все магистральные передачи имели фазы соединения и рассоединения. Фаза передачи данных не включается в некоторые типы передач. Передачи без данных обычно называются передачами только адреса. Эти типы передач используются различным образом при выполнении расщепленных передач. Только адресные передачи могут быть использованы в сообщениях о системных событиях, которые, например, могут устанавливать простые переменные, такие как направленные прерывания в процессорном модуле. Приемник и действие команды указываются в кодах адреса и команды.

Передача начинается со связи задатчика с одним или несколькими исполнителями в течение фазы соединения. Фаза передачи данных начинается, когда связь установлена и заканчивается с окончанием связи. Задатчик может передавать данные исполнителю (исполнителям) во время этой фазы. Связь разрывается во время фазы рассоединения.

Во время фазы передачи данных задатчик может проводить передачу данных, используя либо режим принудительной передачи данных, либо режим пакетной передачи данных.

Задатчик может проводить одну из следующих передач по магистрали.

1) Передача только адреса, состоящая из передачи адреса, за которой следует фаза рассоединения. Здесь нет фазы передачи данных и, следовательно, нет передаваемых данных, за исключением информации, содержащейся внутри фаз соединения и рассоединения.

2) Принудительная передача данных, состоящая из передачи адреса и следующим за ней блока из одной или более передачи данных по одному или нескольким смежным адресам, начинается с адреса, переданного в адресной передаче. Каждая передача данных происходит в соответствии с полным протоколом принудительной передачи данных с подтверждением между задатчиком и исполнителем(лями)

3) Пакетная передача данных состоит из передачи адреса и следующего за ней блока данных фиксированной длины из набора смежных адресов, начиная с адреса, переданного в адресной передаче. Передача данных внутри пакета происходит по протоколу без принуждения. Каждая законченная пакетная передача происходит в соответствии с полным протоколом принудительной передачи данных с подтверждением между задатчиком и исполнителем(лями).

6.1.4 **Протоколы передачи данных**

Принудительный протокол — это механизм, не зависящий от технологической реализации аппаратуры. Этот тип протокола с подтверждением называется принудительным, потому что исполнитель вынуждается к выдаче отклика до того, как задатчик продолжит свои действия. Второй тип протокола — пакетный протокол — является зависимым от технологической реализации (физического быстродействия) механизмом передачи без принуждения на уровне индивидуальной передачи и протоколом с принуждением на уровне пакета. Пакетный протокол оптимизирован для высокоскоростных блочных передач и не включает в себя много особенностей, присущих протоколам с принуждением.

Какой из протоколов будет использован, определяется во время фазы соединения передачи. Любой участник может заставить вести передачу по принудительному протоколу.

Все участники пакетной передачи должны действовать в одном темпе передачи. Во все модули, способные действовать в пакетном режиме, во время системной инициализации загружаются две стандартных скорости передачи для данной системы. Каждый из этих модулей обеспечен регистром, из которого любой другой модуль может считать поддерживаемые этим модулем скорости передачи (см. подробнее в гл. 7). Главный процессор считывает поддерживаемые скорости передачи всех других модулей и устанавливает в качестве меньшей скорости передачи наименьшую из скоростей пакетных модулей. Наивысшая скорость устанавливается равной наибольшей скорости, на которой может работать выбранное подмножество модулей. Определение, на которой из двух скоростей будет происходить конкретная передача, происходит во время фазы соединения. Любой участник может заставить вести передачу на наименьшей скорости.

Так как множественные передачи данных в пакетном режиме происходят после одного подтверждения, модули-участники заранее должны знать, сколько передач будет осуществляться. Каждый модуль, способный работать в пакетном режиме, обеспечивается регистром, из которого другие модули могут считать поддерживаемую длину пакета (см. подробнее в гл. 7). Разрешены только следующие длины: 2, 4, 8, 16, 32 и 64. Задатчик указывает одну из этих стандартных длин пакета во время фазы данных. В принудительной передаче передача данных может быть любой длины в пределах ограничений по тайм-ауту, когда используется неограниченная длина передаваемых данных, если она не ограничивается профилем.

6.1.5 **Широковещательная синхронизация с подтверждением**

Многие исполнители имеют уникальный адрес в системе и в передачах к этим исполнителям используется синхронизация для единственного исполнителя с подтверждением. Некоторые исполнители могут совместно использовать область пространства физических адресов и таким образом, когда задатчик обращается по адресу в одной из этих областей, все исполнители отвечают вместе. Такая передача известна как широковещательная или широкоозывная передача и модули, участвующие в ней, используют широковещательную синхронизацию с подтверждением. Участвующие в передаче исполнители могут вызывать использование широковещательной синхронизации с подтверждением. Широковещательная синхронизация с подтверждением также часто используется в системах с кеш-когерентной разделяемой памятью. Это происходит когда кеш определяет, что интересующие его данные передаются по магистрали. Кеш побуждает широковещательную синхронизацию с подтверждением и читает данные. В кешированном оборудовании эта широковещательная операция относится к довящему.

6.1.6 **Внедренность**

Другая разновидность действий, которые используются в кешированном оборудовании встречается, когда задатчик выделяет модуль памяти для чтения блока данных и кеш другого модуля видит, что он имеет более современную копию. Тогда кеш вмешивается в передачу отстраняя модуль памяти и обеспечивает данными задатчик и модуль памяти. Это то, что относится к внедренности.

Только один исполнитель вмешивается в передачу. Глава о кеш-когерентности объясняет, как атрибуты кешированной строки используются для гарантирования одного внздряющегося.

6.1.7 **Статус кешированной строки**

В кешированном оборудовании модули динамически сохраняют дорожку атрибутов кешированных строк. Они делают это посредством мониторирования сигналов магистрали в течение передачи. Более подробно это объясняется в главе о кеш-когерентности.

6.1.8 **Расщепленные передачи**

Большинство связных передач могут быть преобразованы в расщепленные передачи. Расщепленные передачи позволяют модулям с большой задержкой данных (т. е. временем доступа) использовать ресурсы магистрали более эффективно. Полностью расщепленная передача требует двух или более магистральных передач. Первая передача инициируется запрашивающим модулем и расщепляется другим модулем. Запрашивающий модуль затем завершает владение магистралью. Когда отвечающий модуль способен обеспечить запрашиваемые данные или ответ, он проходит арбитраж, чтобы стать задатчиком на магистрали. Становясь действующим задатчиком, он передает данные запрашивающему модулю. Таким образом, данные всегда передаются от задатчика к исполнителю при расщепленной передаче, т. е. когда используются только операции записи.

Расщепленные действия могут требовать дополнительной информации сверх той, что необходима для связной передачи. Для этих целей обеспечивается передача глобального идентификатора запрашивающего модуля, первоначального приоритета запрашивающего модуля и статуса отвечающего модуля в фазе рассоединения.

Если инициирована операция чтения и один из участвующих исполнителей требует, чтобы передача была расщепленной, связная передача будет преобразована в передачу по магистрали, которая будет запрашивать требуемые данные. После того, как задатчик (запросчик), который запросил данные освободит магистраль, модуль (ответчик) с запрашиваемыми данными овладевает магистралью и передает данные запросчику.

Расщепленная операция записи состоит из процедуры записи, следующей за передачей только адреса, инициированной ответчиком, который подтверждает завершение записи.

Модули не нуждаются в обеспечении возможности расщепленного ответа.

Управление приоритетами расщепленных передач определяется пользователем и не входит в данный стандарт.

6.1.9 **Заблокированные операции**

Поскольку некоторые модули могут содержать отдельные части, которые могут быть доступны более чем через один порт, не может быть гарантии того, чтобы передачи были неделимы. В качестве примера можно привести двухпортовое ОЗУ, в котором каждый порт подключен к различным магистралям ФБ+. Если модуль, подключенный к одному из портов, намерен увеличить содержимое конкретной ячейки памяти, необходимо первоначально прочитать содержимое этой ячейки, а затем увеличить эту величину и в заключении записать данные назад в память. Если модуль, под-

ключенный к памяти через другой порт, также намерен увеличить содержимое той же ячейки, возможна ситуация, когда оба модуля могут читать старое значение, прежде чем каждый запишет новое значение. Это привело бы к однократной инкрементации ячейки памяти вместо двукратного. Данная спецификация обеспечивает средства, гарантирующие, что передачи, подобные вышеприведенным, будут выполняться корректно. Это делается путем использования заблокированных операций.

Магистраль обеспечивает средства, которые задатчик использует для извещения модулей, чтобы одна или несколько передач были защищены. Обычно используются связные передачи там, где задатчик применяет команды соединения и рассоединения, чтобы поддерживать блокировку от передачи к передаче. Чтобы выполнить процессорную команду проверка-модификация, используя связный протокол, задатчику следует провести блокирующее чтение с блокирующим отключением, чтобы осуществить проверку. В течение того же владения магистралью, когда осуществляется чтение, задатчику следует сделать блокирующую запись с блокирующим отключением, чтобы осуществить модификацию. В дополнение к блокирующему рассоединению, все блокировки прекращаются по окончании владения задатчиком магистрали или при выполнении команды передачи, которая не поддерживает блокировку.

6.1.10 **Блокирующие команды**

Простая схема защиты, описанная в 6.1.9, не способна эффективно гарантировать неделимость передач в средах с расщепленными передачами. Как пример, для типичного генерируемого процессором семафора в цикле чтения с последующей записью, задатчик может просто сделать запрос чтения с последующим запросом записи совместно с данными. Ответчику следует вернуть запрашиваемые данные чтения при первом ответе и подтверждение записи при втором ответе. Кроме того, нет гарантии, что ответчик не получит запрос от другой стороны между двумя семафорными запросами. Если это случилось, семафор перестает быть «атомарным». Протоколы защиты могут вынудить магистраль препятствовать другим запросам, которые посланы, однако это может повредить намерению использовать расщепленные передачи. В связи с этим, обеспечивается возможность позволить модулям выполнять защищенные операции с единственным запросом при помощи блокирующих команд. Эти блокирующие команды могут быть использованы либо при расщепленных, либо при связных операциях записи. В общем эти блокирующие команды будут использоваться только в системах, где выполняются расщепленные передачи. К группе блокирующих команд относятся команды маскирование_и_обмен, выборка_и_сложение, сравнение_и_обмен. Разряды в регистре Логические Возможности Модуля указывают, поддерживает или нет модуль блокирующие команды. При операциях по обработке данных, включающих блокирующие команды маскирование_и_обмен и сравнение_и_обмен используется логика для каждого разряда в поле операнда, которая не зависит от всех других разрядов. Поэтому эти операции не зависят от старшинства байтов, и результаты будут идентичными при любом порядке расположения разрядов.

Команда выборка_и_сложение суммирует два операнда для получения результата. Логика суммирования должна иметь дело с переносами между разрядами. Разряды внутри байта всегда имеют постоянный порядок, и переносы всегда распространяются в одном направлении. Между байтами распространение переноса зависит от порядка байтов в операндах. Когда операнд имеет порядок с возрастающим старшинством байтов, переносы распространяются от AD[7 . . . 0]★ к AD[15 . . . 8]★, от AD[15 . . . 8]★ к AD[23 . . . 16]★ и т. д. Когда операнды имеют порядок с понижающимся старшинством байтов, для 32-разрядного слова переносы распространяются от AD[31 . . . 24]★ к AD[23 . . 16]★, от AD[23 . . . 16]★ к AD[15 . . . 8]★ и т. д. Для поддержки этих различных направлений распространения переноса есть две версии команды выборка_и_сложение:

1. «выбрать_и_сложить_со_старшего» для порядка с уменьшающимся старшинством байтов;
2. «выбрать_и_сложить_с_младшего» для порядка с увеличивающимся старшинством байтов.

Операции защищенного чтения обычно не используются при расщепленных передачах. Если защищенное чтение является расщепленным, то защита гарантирует только то, что блок данных, содержащийся в ответчике, был считан внутри при помощи неделимых операций.

6.1.10.1 **Блокирующая команда маскирования и обмена.**

Связная блокирующая команда маскирования и обмена, как показано на рис. 6—1, состоит из следующих частей.

1) Задатчик выполняет операцию типа защищенная запись, задавая команду маскирования и обмена в фазе данных для передачи операнда-маски исполнителю.

2) Используя такую же, как и перечисление 1) операцию, задатчик передает исполнителю операнд для обмена. Задатчик заканчивает операцию командой защищенного рассоединения. Исполнитель выполняет следующую логическую операцию: ОПЕРАНД_МАСКИРОВАНИЯ_И_ОБМЕНА= (ОПЕРАНД_МАСКИРОВАНИЯ & ОПЕРАНД_ОБМЕНА) | (—ОПЕРАНД_МАСКИРОВАНИЯ & ПЕРВОНАЧАЛЬНЫЕ_ДАННЫЕ) и сохраняет результат в буфере.

3) В пределах того же периода владения магистралью, где выполняются перечисления 1) и 2), и перед любой другой операцией задатчик затем совершает операцию защищенного чтения, чтобы вернуть данные, содержащиеся по данному адресу до операции записи из исполнителя.

4) Исполнитель помещает операнд маскирования и обмена вместо данных, находившихся в адресованной ячейке до операции записи.

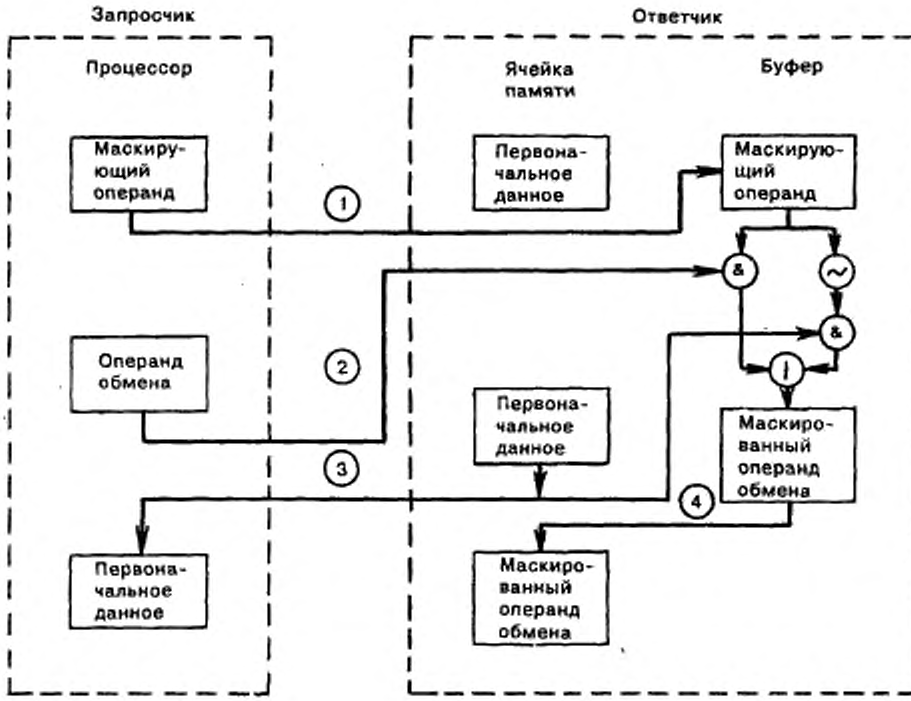

Рисунок 6—1 — Блокирующая команда маскирования и обмена

Если во время первоначальной операции записи исполнитель сообщает задатчику, что он желает, чтобы эта передача была расщепленной, операция записи продолжается, как и в связном случае, после чего запросчик заканчивает свое владение магистралью. Когда ответчик готов, он участвует в арбитраже и выполняет ответную операцию чтения, чтобы вернуть данные, находящиеся в адресованной ячейке до операции записи в запрашивающий модуль. Потом ответчик помещает результат операции маскирования и обмена вместо данных, находившихся в адресованной ячейке до операции записи.

6.1.10.2 **Блокирующие команды выборки и сложения**

В связной команде запрета выборки и сложения, как показано на рис. 6—2, производится следующее.

1) Прежде всего задатчик выполняет операцию типа защищенная запись, задавая команду выборки и сложения в фазе данных для передачи операнда-слагаемого исполнителю. Задатчик завершает передачу командой защищенного рассоединения.

2) Исполнитель складывает операнд-слагаемое с данным, находившимся в адресованной ячейке до операции записи.

3) Во время того же интервала владения магистралью, в котором выполняется перечисление 1), и перед любой другой операцией задатчик затем выполняет операцию защищенного чтения, чтобы вернуть данное, находившееся в адресованной ячейке до операции записи от исполнителя.

4) Исполнитель затем помещает результат в адресованную ячейку.

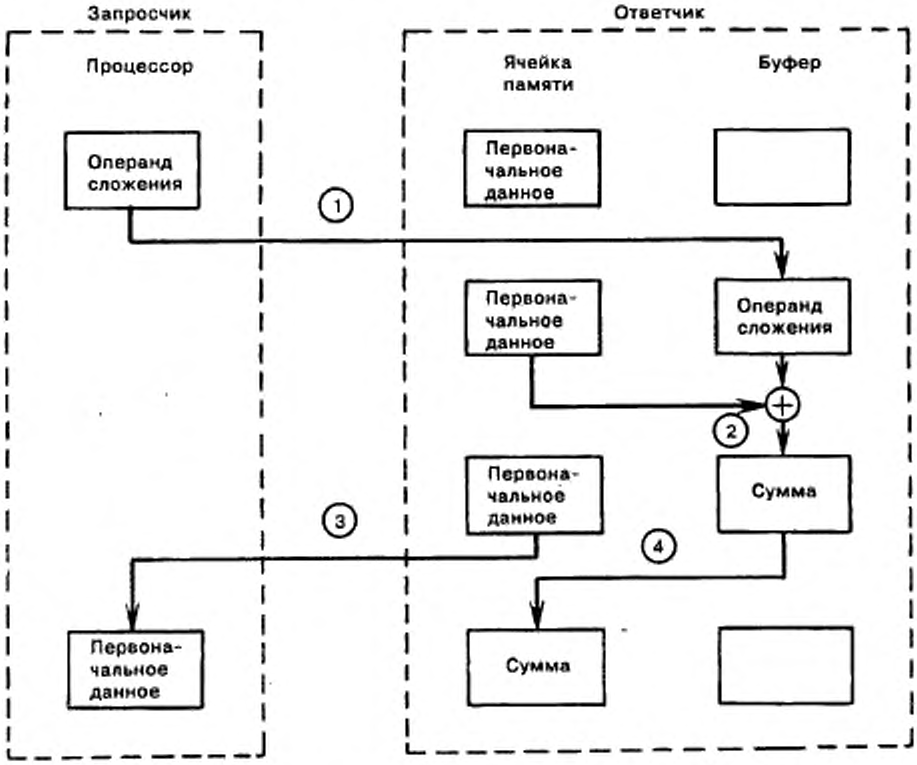

Рисунок 6—2 — Блокирующие команды выборки и сложения

Если во время первоначальной операции записи исполнитель сообщает задатчику, что он желает, чтобы эта передача была расщепленной, операция записи продолжается, как и в связном случае, после чего запросчик заканчивает свое владение магистралью. Когда ответчик готов, он участвует в арбитраже и выполняет операцию чтения ответа, чтобы вернуть данные, находившиеся в адресованной ячейке до операции записи в запросчик. Ответчик складывает операнд-слагаемое с данным, находившимся в адресованной ячейке до операции записи, и помещает результат вместо данного, находившегося в адресованной ячейке до операции записи.

6.1.10.3 **Блокирующая команда сравнения и обмена**

В связной команде запрета сравнения и обмена, как показано на рис. 6—3, выполняется следующее.

1) Задатчик начинает выполнять операцию типа защищенная запись, задавая команду сравнения и обмена в фазе данных, и затем передает операнд для сравнения исполнителю.

2) Используя такую же, как и в перечислении 1), операцию, задатчик передает исполнителю операнд для обмена. Задатчик заканчивает передачу командой защищенного рассоединения.

3) В пределах того же периода владения магистралью, в котором выполнялись перечисления 1) и 2), и перед любой другой операцией задатчик затем совершает операцию защищенного чтения, чтобы вернуть данное, содержавшееся по данному адресу до операции записи из исполнителя.

4) Исполнитель затем сравнивает операнд для сравнения с данным, находившимся в адресованной ячейке до операции записи, и если они равны, то помещает операнд для обмена вместо данного, находившегося в адресованной ячейке до операции записи.

РЕЗУЛЬТАТ=(ПЕРВОНАЧАЛЬНОЕ_ДАННОЕ=ОПЕРАНД_СРАВНЕНИЯ)?
ОПЕРАНД_ОБМЕНА : ПЕРВОНАЧАЛЬНОЕ_ДАННОЕ

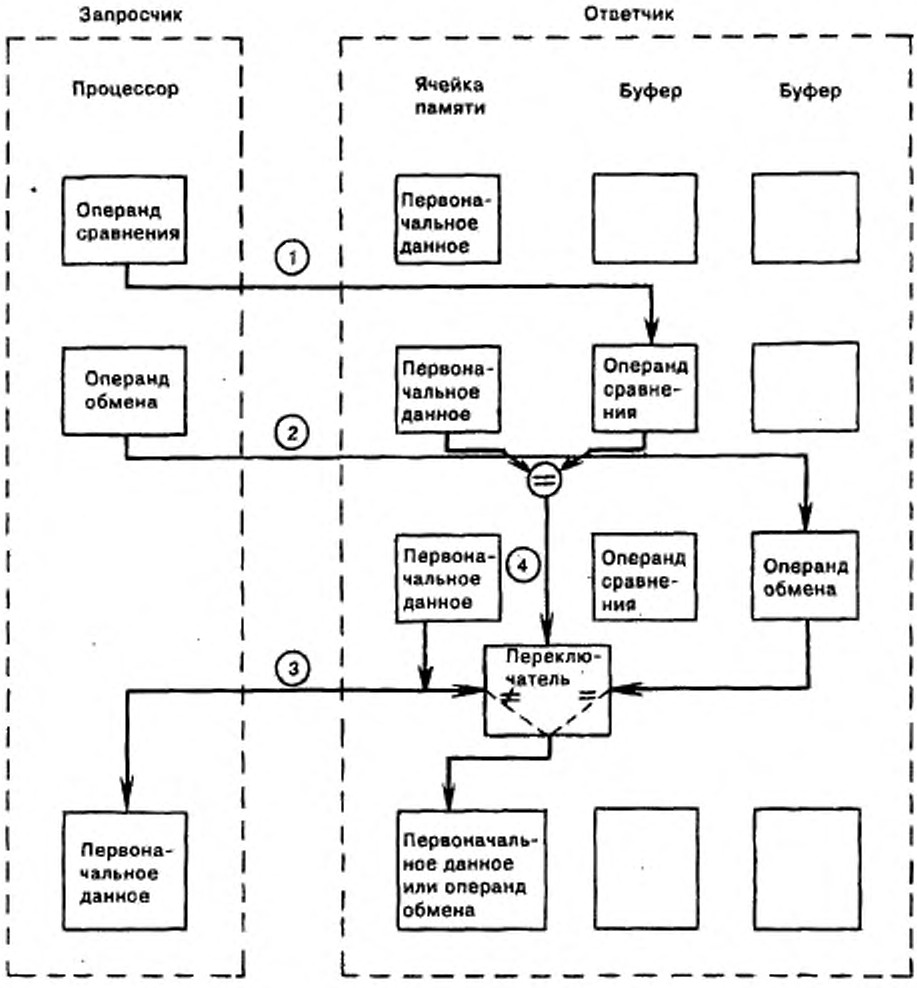

Рисунок 6—3 — Блокирующая команда сравнения и обмена

Если во время первоначальной операции записи исполнитель сообщает задатчику, что он желает, чтобы эта передача была расщепленной, то операция записи продолжается, как и в связном случае, после чего запросчик заканчивает свое владение магистралью. Когда ответчик готов, он участвует в арбитраже и выполняет операцию ответного чтения, чтобы вернуть данное, находившееся в адресованной ячейке до операции записи в запросчик. Ответчик затем сравнивает операнд для сравнения с данным, находившимся в адресованной ячейке до операции записи; если они равны, то ответчик помещает операнд для обмена вместо данного, находившегося в адресованной ячейке до операции записи.

6.1.11 **Занятость**

Если адресуемый модуль не в состоянии ответить обычным образом в тот момент, когда его запрашивают, он сообщает задатчику, что он занят. Это может случиться, если к его запрашиваемым ресурсам осуществляется доступ с другого порта. Если задатчик, выполняющий в данный момент защищенные передачи ко множеству исполнителей, получает статус занятости от любого из них, он должен разблокировать их всех и позже попытаться выполнить операцию снова. Это делается во избежание затыков, которые могут случаться, если два модуля с разных портов ожидают ресурсов, заблокированные другим модулем.

Передачи, которые заканчиваются занятостью, не изменяют состояния системы до тех пор, пока счетчик, определяющий время занятости, не переполнится.

Задатчику следует обеспечить механизм «продвижения вперед» такой, как временная задержка, до попытки очередного доступа по данному адресу. Рекомендуется, чтобы длительность временной задержки была переменной в течение каждой попытки с целью исключения проблем зацикливания. Спецификации высшего уровня могут определять эту задержку.

Механизм продвижения вперед гарантирует системе отсутствие затыков или зависаний. Зависание случается, когда один и более модулей не выполняют полезную работу в течение неопределенного периода времени. Тупик возникает, когда модули ожидают действий, которые могут быть выполнены только другими ожидающими. Зависания могут быть результатом зацикливаний, когда некоторые модули приобретают или отпускают ресурсы таким образом, что нет продвижения в их действиях.

Хотя возможно построение систем в стандарте Futurebus+, гарантированных от тупиков, разработки, использующие множество магистралей, могут быть уязвимы к зацикливаниям и должны быть спроектированы очень тщательно для того, чтобы уменьшить вероятность подобных событий. Большинство ситуаций, которые могут вызывать зацикливание, являются результатом использования операции занятости.

6.1.12 **Ожидание**

Если модуль не способен ответить соответствующим образом в то время, когда его запрашивают сделать это, и если модуль будет позднее генерировать различимое событие, прекращающее это состояние, то модуль уведомляет задатчика о состоянии ожидания. Различимое событие обычно специфицируется стандартом высшего уровня. Передачи, заканчивающиеся ожиданием, не изменяют состояния системы.

Например, протокол кеш-когерентности требует, чтобы самое большее один расщепленный запрос выставлялся по одному адресу на данном магистральном сегменте. Расщепленный магистральный мост использует операцию ожидания для задержки второго запроса в ту же ячейку, состояние ожидания очищается ответной передачей к источнику запроса.

Режим расщепление/ожидание формирует очередь, во главе которой находится модуль, чей запрос принят, а оставшиеся являются модулями, чьи запросы ожидают. Порядок обслуживания остающихся в очереди определяется в процессе магистрального арбитража, когда ожидания удовлетворяются. В системах реального времени или системах, основанных на приоритетах, исполнителю следует увеличить приоритет полученного запроса в большую сторону или приоритет ожидающего запроса. Это относится к приоритетному наследованию.

Задатчику следует обеспечить тайм-аут для исключения сигнала, если ответ не получен.

6.1.13 **Расширенная разрядность магистрали**

Модули могут поддерживать 4 разрядности данных. Это могут быть 32, 64, 128 и 256 разрядов. Задатчик задает разрядность магистрали в течение фазы соединения для данной передачи.

Если модуль не может поддерживать разрядность данных, определенную в фазе соединения, он выставляет признак ошибки. См. гл. 7 для выяснения деталей управления совместимостью через статус ошибки модуля.

6.1.14 **Расширенный адрес**

Модули могут распознавать как 32-, так и 64-разрядные адреса. Задатчик задает разрядность адреса в течение фазы соединения для данной передачи. Модули, которые не могут поддерживать 64-разрядные адреса, не отвечают на такие передачи. 32-разрядное адресное пространство является подмножеством 64-разрядного адресного пространства. Передачи в 32-разрядном адресном пространстве должны всегда использовать 32-разрядные адреса в целях обеспечения межвзаимодействий.

6.1.15 **Возможности модуля**

Задатчик и участвующий исполнитель разрешают все различия в своих возможностях в течение фазы соединения. Задатчик определяет тип передачи, которую он намерен выполнить. Участвующие модули, которые не могут выполнить указываемой передачи, либо выставляют статус ошибки, либо, если это возможно, преобразовывают передачу в одну из возможных для себя.

6.1.16 **Передачи**

ФБ+ передачи без кеширования следующие.

★ Незащищенное чтение
★ Незащищенная запись
★ Только адрес незащищенная
★ Защищенное чтение
★ Защищенная запись
★ Только адрес защищенная
★ Частное чтение
★ Частная запись
★ Защищенное частное чтение
★ Защищенная частная запись
★ Ответ записи
★ Ответ чтения
★ Запись без подтверждения

ФБ+ передачи с кешированием следующие.

★ Недействительное чтение
★ Недействительная запись
★ Разделенное чтение
★ Обратное копирование
★ Модифицированное чтение
★ Недействительность
★ Разделенный ответ
★ Модифицированный ответ

6.1.16.1 **Незащищенное чтение**

Передача типа незащищенное чтение используется для передачи данных от одного или более исполнителей к задатчику. Передачи данного типа могут быть выполнены с использованием либо принудительного, либо пакетного протоколов. Исполнители могут преобразовывать связную передачу в расщепленную, при которой информация для чтения передается в адрес только после запроса чтения. Исполнители могут вызывать широковещательную передачу и конвертировать передачу из односторонней в многостороннюю. Один из исполнителей может осуществлять внедрение в источник данных вместо выбранного исполнителя.

6.1.16.2 **Незащищенная запись**

Передача типа незащищенная запись используется для передачи данных к одному и более исполнителям от задатчика. Эти передачи могут быть выполнены с использованием либо принудительного, либо пакетного протоколов. Исполнители могут преобразовывать такую передачу в расщепленную. Исполнители могут вызывать широковещательную передачу и конвертировать передачу из односторонней в многостороннюю.

6.1.16.3 **Только адрес незащищенная**

Передача типа только адрес незащищенная используется для сообщений о системных событиях. Данные в ней не передаются. Эту передачу невозможно отличить от передачи «незащищенная запись» в течение фазы соединения. Модули определяют наличие передачи только адреса по отсутствию фазы данных.

6.1.16.4 **Защищенное чтение**

Передача защищенное чтение подобна передаче с незащищенным чтением, описанной в 6.1.16.1, за исключением того, что она вынуждает участвующие исполнители защитить любые защищаемые адресуемые ресурсы, и тем, что подпоследовательность команд рассоединения удерживает состояние защиты после окончания этой передачи. Защищенное состояние поддерживается до начала незащищенной передачи или до конца владения магистралью.

6.1.16.5 **Защищенная запись**

Передача защищенная запись подобна передаче с незащищенной записью, описанной в 6.1.16.2, за исключением того, что она вынуждает участвующие исполнители защитить любые защищаемые адресуемые ресурсы, и тем, что подпоследовательность команд рассоединения удерживает состояние защиты после окончания этой передачи. Защищенное состояние поддерживается до начала незащищенной передачи или до конца владения магистралью.

6.1.16.6 **Только адрес защищенная**

Передача только адрес защищенная подобна незащищенной передаче адреса, описанной в 6.1.16.3, за исключением того, что она вынуждает участвующие исполнители защитить любые защищаемые адресуемые ресурсы, и тем, что подпоследовательность команд рассоединения удерживает состояние защиты после окончания этой передачи. Защищенное состояние поддерживается до начала незащищенной передачи или до конца владения магистралью.

6.1.16.7 **Частное чтение**

Операция частного чтения используется для передачи данных от одного или более исполнителей к задатчику. Задатчик указывает, что будет воспринимать лишь некоторые байты из задаваемых разрядностью передаваемых данных. Исполнители могут преобразовывать передачу в расщепленную передачу. Исполнители могут вызывать широковещательную передачу и преобразовать одностороннюю передачу в многостороннюю.

6.1.16.8 **Частная запись**

Операция частной записи используется для передачи данных к одному или более исполнителям. Задатчик указывает исполнителю, какие из передаваемых байтов исполнитель должен воспринимать. Исполнители могут преобразовывать эту передачу в расщепленную передачу. Исполнители могут вызывать передачу сообщений и преобразовывать ее в многократную передачу.

6.1.16.9 **Защищенное частное чтение**

Передача защищенное частное чтение подобна операции частного чтения, описанной в 6.1.16.7, за исключением того, что она вынуждает участвующие исполнители защитить любые защищаемые адресуемые ресурсы, и тем, что подпоследовательность команд рассоединения удерживает состояние защиты до окончания этой передачи.

6.1.16.10 **Защищенная частная запись**

Передача защищенная частная запись подобна операции частной записи, описанной в 6.1.16.8, за исключением того, что она вынуждает участвующие исполнители защитить любые защищаемые адресуемые ресурсы, и тем, что подпоследовательность команд рассоединения удерживает состояние защиты до окончания этой передачи.

6.1.16.11 **Ответ записи**

Расщепленный ответ записи состоит из передачи только адреса. Отвечающий модуль передает глобальную идентификацию запроса в фазе соединения. Глобальная идентификация запроса передается в фазе соединения, а статус — в фазе рассоединения. Передача с ответом записи сообщает запрашивающему модулю, что предварительно запрашиваемая операция записи, которая была расщеплена и использовала незащищенную запись, защищенную запись, частную запись или частную защищенную запись, закончилась. Данные записи были переданы в первоначальной операции записи, которая была расщеплена.

Если какие-либо из модулей, которые получили запрос, не выполнили запрашиваемую запись, они сообщают источнику запросов выставлением sr★ в течение этой передачи, так что источник запросов знает, что ответы все еще продолжаются.

6.1.16.12 **Ответ чтения**

Передача типа ответ чтений является ответом на предварительный запрос, в котором модуль запрашивал данные, используя одну из следующих передач: незащищенное чтение, частное чтение, защищенное чтение, защищенное частное чтение, защищенная запись с блокирующей командой или защищенная частная запись с блокирующей командой. Передача содержит фазу данных, в которой данные записываются в запрашивающий модуль. В подобных передачах могут быть использованы либо принудительный, либо пакетный протокол. Глобальная идентификация запроса и передача ID (идентификатора) осуществляется в фазе соединения, а статус передается в фазе рассоединения. Существует только один ответ чтения для единственного запроса на расщепленное чтение.

6.1.16.13 **Запись без подтверждения**

Передача типа запись без подтверждения используется задатчиками для записи в один или более исполнителей. Исполнители не преобразуют передачу в расщепленную, так как задатчик определил, что он не нуждается в отклике. Исполнители могут осуществлять передачи другим узлам без поддержки связи с задатчиком.

6.1.16.14 **Недействительное чтение**

Операция недействительное чтение может быть использована задатчиком, таким как ПДП или ВВ (ввода/вывода) контроллером, для чтения кеш-строки из кеш-когерентной памяти. Не существует действительной копии в кеше задатчика ни до, ни после передачи.

Хранитель последнего положения (обычно основная память) перехватывает передачи недействительного чтения, если имеется сигнал о вмешательстве.

6.1.16.15 **Недействительная запись**

Передача недействительная запись может быть использована задатчиком, таким как ПДП или ВВ контроллером, для записи кеш-когерентной копии из кеш-строки. Не существует действительной копии в кеше задатчика ни до, ни после передачи. Модулям запрещено ловить эту передачу.

6.1.16.16 **Обратное копирование**

Передача обратное копирование используется для записи модифицированной строки кеша обратно в кеш-когерентную память для освобождения пространства. Задатчик должен иметь исключительно модифицированный атрибут перед началом передачи обратного копирования. Недействительные кеши, которые ожидают данных, могут ловить эту передачу во время ее прохождения в память. Модулям запрещено расщеплять эту передачу.

6.1.16.17 **Разделяемое чтение**

Передача разделяемое чтение используется кешированным задатчиком или магистральным мостом для обслуживания промаха чтения чтением строки кеша из кеш-когерентной памяти. Копия действительного чтения сохраняется только в кеше задатчика. Копия строки кеша помечается как разделенная, немодифицированная. Копия строки кеша может стать действительной исключительной немодифицированной, если не существует других наблюдаемых сообщений, что также имеется копия.

Хранитель последнего положения (обычно основная память) ловит разделенные передачи чтения, если имеется сигнал о вмешательстве.

6.1.16.18 **Модифицированное чтение**

Передача модифицированное чтение используется кешированным задатчиком или магистральным мостом для обслуживания промаха записи чтением строки кеша из кеш-когерентной памяти. Действительного чтения копия записи сохраняется только в кеше задатчика. Копия строки кеша помечается как исключительная модифицированная. Все другие кеши недействительны после окончания модифицированного чтения, как и любой связанный модифицированный ответ.

6.1.16.19 **Недействительность**

Операция недействительность используется кешированным задатчиком или магистральным мостом для обслуживания попадания записи в действительную разделяемую строку кеша из кеш-когерентной памяти путем перевода в недействительные всех остальных кешей и превращения его строки из разделенной в действительную исключительную модифицированную копию. Все другие кеши недействительны после окончания недействительности.

До тех пор, пока исходный задатчик ожидает расщепленного ответа, другой задатчик на другом магистральном сегменте может получить разрешение записи. В этом случае передача модифицированного ответа к исходному задатчику должна содержать обновленную копию строки кеша.

6.1.16.20 **Разделенный ответ**

Передача разделенный ответ используется для ответа на недействительное расщепленное чтение или разделенное чтение. Один разделенный ответ может удовлетворить любое число ожидающих или желающих разделенного либо недействительного чтения.

Хранитель последнего положения (обычно основная память) ловит передачи разделенного ответа.

6.1.16.21 **Модифицированный ответ**

Передача модифицированный ответ используется как ответ на модифицированное расщепленное чтение, недействительность или недействительную запись.

Магистральный мост, который является задатчиком при модифицированном ответе, гарантирует, что не существует других удаленных действительных копий строк кеша на другой его стороне.

Модифицированный ответ также используется модулями памяти для разрешения состязаний чтения и записи между соседними магистралями.

6.1.17 **Описание сигналов магистрали**

6.1.17.1 **AS★ Синхронизация адреса**

Задатчик выставляет as★, чтобы показать, что активированы сигналы AD[]★, BP[]★, TG[]★, TP★, CM[]★, и CP★ с действительной адресной и командной информацией и что передача инициирована.

Задатчик снимает as★, чтобы показать, что передача закончилась. Если передача только адресная, имеет принудительную фазу данных или использует расщепленный ответ, тогда снятие сигнала as★ показывает, что активированы сигналы CM[]★ и CP★ с действительной командой рассоединения. Если передача использует расщепленный ответ, тогда снятие as★ также показывает, что активированы сигналы AD[31 . . . 0]★ BP[3 . . . 0]★ с действительными глобальным идентификатором запросчика, приоритетом и статусом.

6.1.17.2 **AK★ Подтверждение адреса**

Задатчик и исполнитель(и) выставляют ak★, когда они обнаруживают AS★ выставленным. Модули снимают сигнал ak★ после того, как они обнаруживают снятие AS★, получают любую действительную информацию, передаваемую задатчиком, активируют te★ при ошибочной действительной информации и выставляют ai★.

6.1.17.3 **AI★ Инверсное подтверждение адреса**

Задатчик и исполнитель(и) выставляют ai★, когда они обнаруживают снятие AS★.

Модули снимают ak★, когда обнаруживают AS★ выставленным, получают любую действительную информацию, передаваемую другими модулями, активируют st[7 . . . 1]★, ca[2 . . . 0]★ и di★ в ответ на адрес и команду, передаваемую задатчиком, и выставляют ak★.

6.1.17.4 **DS★ Синхронизация данных**

Задатчик выставляет или снимает сигнал в течение времени, когда as★ выставлен, чтобы показать, что сигналы CM[]★ и CP★ активированы с действительной командной информацией и что задатчик запрашивает передачу другого слова или пакета. В течение передачи записи перепад ds★ также показывает, что были активированы AD[]★, D[]★, BP[]★, TG[]★ и TP★.

Если задатчик закончил передачу данных в данной пересылке и выставлен ds★, то первым будет снят as★, далее он будет ожидать снятия DK★ перед снятием ds★.

6.1.17.5 **DK★ Подтверждение данных**

Во время широковещательных или передач с внедрением, в которых исполнитель является источником данных, исполнитель выставляет dk★ после того, как были активированы ad[]★ и d[]★ в ответ на выставление DS★. В других передачах участвующие исполнители выставляют dk★, когда они обнаруживают выставленный DS★.

Исполнители снимают dk★ после того, как удовлетворены все нижеследующие условия.

1) Они обнаруживают снятие DS★.

2) Они активировали st[7 . . . 1]★ с действительной статусной информацией.

3) Они получили информацию, переданную задатчиком по AD[]★, D[]★, BP[]★, TG[]★ и TP★, если передача является записью.

4) Они активировали ad[]★, d[]★, bp[]★, tg[]★ и tp★, если передача является связным чтением и

5) Они выставили di★.

6.1.17.6 **DI★ Инверсное подтверждение данных**

Участвующие исполнители выставляют di★, когда они обнаруживают выставленный AS★. Во время широковещательных или передач с внедрением, в которых исполнители являются источниками данных, они выставляют di★ после того, как они активировали данные на ad[]★, d[]★ в ответ на снятие DS★. В других передачах участвующие исполнители выставляют di★, когда они обнаруживают DS★ снятым.

Исполнители снимают di★ после того, как удовлетворены все нижеследующие условия.

1) Они обнаруживают выставление DS★.

2) Они активировали st[7 . . . 1]★ с действительной статусной информацией.

3) Они получили информацию, переданную задатчиком по AD[]★, D[]★, BP[]★, TG[]★ и TP★, если передача является записью.

4) Они активировали ad[]★, d[]★, bp[]★, tg[]★ и tp★, если передача является связным чтением и

5) Они выставили dk★.

6.1.17.7 **CM[7 . . . 0]★ Командное поле**

Задатчик активирует командное поле с информацией, которую исполнители используют для определения типа текущей передачи, или информацию о передаче. Эти линии однозначно определяются в каждой фазе передачи. Эти сигналы выдаются только задатчиком. Интервал, в течение которого данные сигналы не определены, резервируется для других стандартов, и поэтому следует быть снятым. Табл. 6.1 показывает определения командных сигналов для различных фаз передачи. Только в режиме множественного пакета выполняется изменение командных сигналов в течение фазы данных передачи 1.

Таблица 6.1 — **Форматы командных полей**

Фаза	CM7★	CM6★	CM5★	CM4★	CM3★	CM2★	CM1★	CM0★
Соединения	AW★	DW1★	DW0★	WR★	TC3★	TC2★	TC1★	TC0★
Принудительная незащищенная фаза данных	DL2★	DL1★	DL0★	SW★	—	-	—	-
Принудительная защищенная фаза данных	DL2★	DL1★	DL0★	SW★	—	LC2★	LC1★	LC0★
Пакетная фаза данных	DL2★	DL1★	DL0★	SW★	–	—	LT★	PR★
Фаза рассоединения	DL2★	DL1★	DL0★	SW★	–	—	MR★	LK★

6.1.17.7.1 **CM7★ Команда 7**

В течение фазы соединения линия CM7★ определяется как Разрядность адреса (AW★). Задатчик выставляет aw★, чтобы показать, что AD[63 . . . 32]★ активированы с действительной адресной информацией и что исполнитель(и) должен(ы) декодировать 64-разрядный адрес. Исполнители, не способные декодировать 64-разрядные адреса, не отвечают на передачи, в которых выставлен AW★. Задатчик снимает aw★, чтобы показать, что исполнитель(и) должен (ы) декодировать 32-разрядный адрес, активированный на AD[31 . . . 0]★.

В течение фаз данных и рассоединения линия CM7★ определяется как длина передаваемых данных — второй разряд (DL2★). Текущий задатчик активирует DL[2 . . . 0]★ для выбора одной из 7 длин передаваемых данных или неограниченную длину передачи, как показано в таблице 6.2.

Таблица 6.2 — **Длина посылки данных**

DL2★	DL1★	DL0★	Длина посылки данных
0	0	0	Неограниченная
0	0	1	1
0	1	0	2
0	1	1	4
1	0	0	8
1	0	1	16
1	1	0	32
1	1	1	64

Длина передаваемых данных указывает число запрашиваемых передач задатчика, а не число байтов. Неограниченная длина данных не разрешена для пакетных передач и для передач с расщепленным чтением. Неограниченная длина данных может быть использована для расщепленных передач записи. Если задатчик определяет количественно длину посылки со стартовым адресом, который не является четным по отношению к адресу, вычисленному по комбинации разрядности и длины данных, тогда задатчик будет заканчивать передачу до того, как он достигнет адреса, являющегося четным. Например, передача 32-разрядных данных длиной 8 слов, которая начинается с шестнадцатиричного адреса 8, заканчивается шестнадцатиричным адресом 1С. Передача 64-разрядных данных длиной 8 слов, которая начинается с шестнадцатиричного адреса 10, заканчивается шестнадцатиричным адресом 38.

В течение принудительных передач задатчик управляет количеством пересылок посредством числа передач DS★. В течение принудительных передач исполнители могут заканчивать передачу выставлением ED★ в независимости от длины передачи. Когда задатчик определяет длину посылки другую, чем ограничена в течение принудительной передачи, он никогда не превысит число этих посылок. Обычно модулям, которые не способны на расщепленные передачи или пакетный режим, следует, но необязательно, активировать DL[2 . . . 0]★.

6.1.17.7.2 **СМ6★ Команда 6**

В течение фазы соединения эта линия определяет Разрядность данных 1 (DW1★). Задатчик выставляет dw1★, чтобы показать, что разрядность магистрали составляет либо 128, либо 256 бит. Задатчик снимает dw1★, чтобы показать, что посылка составляет либо 32, либо 64 бит.

В течение фазы рассоединения линия СМ6★ определяется как длина передаваемых данных — первый разряд (DL1★).

6.1.17.7.3 **СМ5★ Команда 5**

В течение фазы соединения эта линия определяет Разрядность данных 0 (DW0★). Задатчик выставляет dw0★, чтобы показать, что разрядность магистрали составляет либо 64, либо 256 бит, как показано в таблице 6.3. Задатчик снимает dw0★, чтобы показать, что посылка составляет либо 32, либо 128 бит.

В течение фазы рассоединения линия СМ5★ определяется как длина передаваемых данных — нулевой разряд (DL0★).

Таблица 6.3 — **Разрядность переливаемых данных**

DW1★	DW0★	Разрядность передаваемых данных
0	0	32
0	1	64
1	0	128
1	1	256

6.1.17.7.4 **СМ4★ Команда 4**

В течение фазы соединения эта линия определяется как запись (WR★). Задатчик выставляет wr★, чтобы показать, что данные должны быть переданы от задатчика к исполнителю(ям) в течение текущей передачи. Задатчик снимает wr★, чтобы показать, что данные должны быть переданы от исполнителя(ей) к задатчику.

В течение фазы рассоединения эта линия определяется как Запись исполнителя (SW★). Задатчик выставляет sw★, когда он обнаруживает статус IW★ линии выставленным в конце фазы соединения или после каждого пакетного цикла запрос—ответ исполнителя в пакетном режиме. Выставление SW★ показывает выбранному исполнителю (разделяемой памяти), что он должен получить данные в течение текущей передачи или пакет вместо исходных данных.

6.1.17.7.5 **СМ3★ Команда 3**

В течение фазы соединения эта линия определяется как Команда передачи 3 (TC3★). Задатчик активирует tc[3 . . . 0]★ и wr★ для выбора команды передачи из таблицы 6.4.

В течение фазы рассоединения эта линия не определена, и сигнал должен быть снят.

Таблица 6.4 — **Кодировка команд перелач**

Передачи	TC3★	TC2★	TC1★	TC0★	WR★
Незащищенное чтение	0	0	0	0	0
Незащищенная запись	0	0	0	0	1
Незащищенный только адрес	0	0	0	0	1
Защищенное чтение	0	0	0	1	0
Защищенная запись	0	0	0	1	1
Защищенный только адрес	0	0	0	1	1
Частное чтение	0	0	1	0	0
Частная запись	0	0	1	0	1

Окончание таблицы 6.4

Передача	TC3★	TC2★	TC1★	TC0★	WR★
Защищенное частое чтение	0	0	1	1	0
Защищенная частная запись	0	0	1	1	1
зарезервировано	0	1	0	0	0
Ответ записи	0	1	0	0	1
зарезервировано	0	1	0	1	0
Ответ чтения	0	1	0	1	1
зарезервировано	0	1	1	0	0
Запись без подтверждения	0	1	1	0	1
зарезервировано	0	1	1	1	0
зарезервировано	0	1	1	1	1
Недействительное чтение	1	0	0	0	0
Недействительная запись	1	0	0	0	1
Разделенное чтение	1	0	0	1	0
Обратное копирование	1	0	0	1	1
зарезервировано	1	0	1	0	0
зарезервировано	1	0	1	0	1
Модифицированное чтение	1	0	1	1	0
Недействительность	1	0	1	1	1
зарезервировано	1	1	0	0	0
Разделенный ответ	1	1	0	0	1
зарезервировано	1	1	0	1	0
зарезервировано	1	1	0	1	1
зарезервировано	1	1	1	0	0
зарезервировано	1	1	1	0	1
зарезервировано	1	1	1	1	0
Модифицированный ответ	1	1	1	1	1

6.1.17.7.6 **CM2★ Команда 2**

В течение фазы соединения эта линия определяется как Команда передачи 2 (TC2★).

В течение фазы данных защищенной передачи эта линия определяется как Блокирующая команда 2 (LC2★). Задатчик активирует lc[2 . . . 0]★, чтобы выбрать одну из семи возможных блокирующих команд, как показано в таблице 6.5.

В течение фазы рассоединения этот сигнал неопределен и должен быть снят.

Таблица 6.5 — **Кодировка блокирующих команд**

LC2★	LC1★	LC0★	Блокирующая команда
0	0	0	Нет команды
0	0	1	Маскирование & Взаимный обмен
0	1	0	Сравнение & Взаимный обмен
0	1	1	Выборка & Сложение со старшего
1	0	0	Выборка & Сложение с младшего
1	0	1	резервная команда
1	1	0	резервная команда
1	1	1	резервная команда

6.1.17.7.7 **CM1★ Команда 1**

В течение фазы соединения эта линия определяется как Команда передачи 1 (TC1★).

В течение фазы данных защищенной передачи эта линия определяется как Блокирующая команда 1 (LC1★).

В течение фазы данных пакетной посылки эта линия определяется как Флаг последней передачи (LT★). Задатчик выставляет lt★, когда он обнаруживает статус линии TF★ выставленным в конце фазы соединения или после каждого пакетного цикла запрос—ответ исполнителя.

В течение фазы рассоединения эта линия определяется как Больше (MR★). Задатчик выставляет Больше, чтобы показать, что операция блочного копирования находится в действии и разрешены предвыборка, переупорядоченная запись и запись с буферизацией. См. P896.3 для получения более полной информации об этом.

6.1.17.7.8 **CM0★ Команда 0**

В течение фазы соединения эта линия определяется как Команда передачи 0 (TC★).

В течение фазы данных во время защищенной передачи она определяется как Блокирующая команда 0 (LC0★).

В течение фазы рассоединения защищенной передачи эта линия определяется как Запрет (LK★). Задатчик выставляет lk★, чтобы показать, что все модули, которые имеют защищаемые ресурсы, должны выставить и сохранять защиту после конца текущей передачи.

В течение фазы данных, когда выбран пакетный режим, эта линия определяется как Пакетный запрос (PR★). Задатчик выставляет запрос на пакетную передачу, чтобы показать исполнителю(ям), что задатчик запрашивает дополнительную пакетную посылку. В течение фазы данных для незащищенных передач, когда пакетный режим не определен, эта линия должна быть освобождена.

6.1.17.8 **CP★ Команда четности**

Линия CP★, несущая бит четности для линий CM[7 . . . 0]★, выставляется, если число выставленных сигналов четно.

6.1.17.9 **ST[7 . . . 0]★ Статус**

Одна или более статусных линий могут быть выставлены одним или более участвующими модулями, чтобы показать статус исполнителя и интерпретацию информации, полученной задатчиком. Следующая таблица показывает определение статусных сигналов для различных фаз передачи.

Таблица 6.6 — **Кодировка статуса**

Фаза	ST7★	ST6★	ST5★	ST4★	ST3★	ST2★	ST1★	ST0★
Соединение	WT★	BE★	IV★	TF★	BC★	SL★	BS★	TE★
Данные	WT★	BE★	IV★	TF★	BC★	SL★	ED★	TE★
Рассоединение	WT★	BE★	IV★	TF★	BC★	SL★	BS★/ED★	TE★

6.1.17.9.1 **ST7★ Статус 7**

Эта линия определяется как Ожидание (WT★).

Выбранный или участвующий исполнитель выставляет wt★ в течение фазы соединения, чтобы закончить передачу без фазы данных. Задатчик тогда ожидает системно определенного события до повторной передачи. Никаких изменений в атрибутах кеша не происходит, если в течение передачи выставлен WT★. Только задатчик способен обнаруживать состояние WT★ в течение фазы соединения. Исполнитель продолжает удерживать сигнал WT★ в течение фазы рассоединения. Все модули в состоянии обнаруживать состояние WT★ в течение фазы рассоединения.

6.1.17.9.2 **ST6★ Статус 6**

ST6★ определяется как Ошибка обмена (BE★).

В течение фазы соединения любой модуль выставляет be★, если он обнаруживает ошибку четности на линиях CM[]★, AD[]★ или TG[]★. Участвующие модули выставляют be★ в фазе соединения, если они обнаруживают нелегальную операцию. Модули, воспринимающие 64-разрядные адреса, также проверяют четность AD[64 . . . 32]★, если выставлен AW★. Модули также выставляют be★, если задатчик требует от них действия, которого они не в состоянии выполнить. Если задатчик обнаруживает BE★ выставленным в течение фазы соединения, он приступает к фазе рассоединения.

В течение принудительной фазы данных любой из участвующих исполнителей выставляет be★, если он обнаруживает ошибку четности на линиях CM[]★, AD[]★, D[]★ или TG[]★. Модули также проверяют четность на AD[64 . . . 32]★ и D[]★, если соответствующая разрядность данных (DW[]★) была выставлена. В режиме без подтверждения выставление BE★ не гарантируется при обмене данными, когда была обнаружена ошибка.

Задатчик может продолжать режим запрос—ответ при наличии выставленного BE★, однако данные при этом игнорируются как задатчиком, так и исполнителями.

В течение фазы рассоединения модули, которые выставили be★ в фазах соединения или данных, продолжают удерживать be★.

6.1.17.9.3 **ST5★ Статус 5**

Эта линия определяется как Внедрение (IV★).

В течение принудительных передач модуль выставляет iv★, если он внедряется в передачу. Модуль сохраняет выставленный сигнал iv★ до конца фазы рассоединения. Когда задатчик обнаруживает выставленный сигнал IV★, он открывает свои проводного-ИЛИ интеграторы на приемниках DK★ или DI★ и оставляет их открытыми в течение передачи.

В течение пакетной передачи модуль выставляет iv★, если он внедряется в пакет, который запрашивается. Модуль сохраняет сигнал iv★ до следующей переключения DS★ или до конца фазы рассоединения.

6.1.17.9.4 **ST4★ Статус 4**

Эта линия определяется как Флаг передачи (TF★).

Протоколы высшего уровня, такие как кеш-когерентность и посылка сообщений, определяют информацию, которую несет эта линия. Определение сигнала в передаче зависит от базиса передачи.

6.1.17.9.5 **ST3★ Статус 3**

Эта линия определяется как Широковещательная передача (BC★).

Модули выставляют bc★ в течение фазы соединения для преобразования передачи с одним исполнителем в передачу со множеством исполнителей. Когда задатчик обнаруживает BC★ выставленным, он открывает свои проводного-ИЛИ интеграторы на приемниках DK★, DI★ и оставляет их открытыми в течение передачи. Модули сохраняют bc★ до конца фазы рассоединения.

6.1.17.9.6 **ST2★ Статус 2**

Эта линия определяется как Выбранный (SL★).

Модули выставляют sl★ в течение фазы соединения, если распознали адрес, на который они должны ответить, и не обнаружили ошибки четности. Если задатчик не обнаруживает SL★ в конце фазы соединения, он приступает прямо к фазе рассоединения, так как не было получено ответа ни от одного из модулей.

6.1.17.9.7 **ST1★ Статус 1**

В течение фазы соединения ST1★ определяется как Занятость (BS★).

Если сигнал на линии был выставлен в течение фазы соединения, он также определяется как BS★ в течение фазы рассоединения. Выбранный или участвующий исполнитель выставляют bs★, чтобы закончить передачу без фазы данных. Задатчик тогда ожидает в течение системно определенного времени до очередной попытки передачи. Никаких изменений в атрибутах кеша не происходит, если BS★ выставлен в течение передачи. Только задатчик способен обнаруживать BS★ в течение фазы соединения. Все модули способны обнаруживать сигнал BS★ в течение фазы рассоединения.

Если сигнал BS★ не был выставлен в течение фазы соединения, ST1★ определяется как Конец данных (ED★) в течение фаз данных и рассоединения. Участвующий модуль выставляет ed★ в течение принудительной передачи, чтобы показать, что текущая передача должна закончиться без дальнейшей посылки данных и что задатчику следует приступить к фазе рассоединения. Участвующий модуль выставляет ed★ в режиме множественной пакетной передачи, чтобы показать, что текущая передача должна закончиться после последнего подтвержденного пакета без сигнала ED★. В большинстве случаев задатчик будет инициировать новую передачу после получения выставленного сигнала ED★, если не все данные еще переданы. Если задатчик выполнял передачу ответа чтения и обнаружил выставленный ED★, тогда он будет игнорировать все оставшиеся данные и не будет инициировать новую передачу. В течение фазы рассоединения ED★ показывает статус конца данных для последней принудительной посылки данных или последнего запрашиваемого пакета.

Задатчик может продолжать циклы запрос—ответ, сопровождающиеся ED★, однако данные в этом случае будут игнорироваться как задатчиком, так и исполнителями.

6.1.17.9.8 **ST0★ Статус 0**

Эта линия определяется как Ошибка передачи (TE★).

TE★ недействителен в течение фазы соединения. В течение фаз данных или рассоединения модули выставляют te★, чтобы сообщить об ошибках четности, переполнения или недобора данных при пакетных передачах, запросе задатчиком пакетов вне статуса данных и потере четности. Модули сохраняют te★ выставленным до конца фазы соединения следующей передачи вне зависимости от изменения владения магистралью. Модули, которые выставляют ошибку обмена (be★) в течение фаз соединения или данных, также выставляют te★ в течение фазы рассоединения. Хотя TE★ может быть выставлен в течение фазы данных, его выборки не может быть до конца фазы рассоединения, когда снимается AFf.

6.1.17.10 **CA[2 . . . 0]★ Способность**

Одна или более линий способности выставляются одним или более модулями, чтобы показать способности участвующих модулей. Таблица 6.7 показывает определения сигналов способности для различных фаз передач.

Таблица 6.7 — **Кодировка способности**

Фаза	CA2★	CA1★	CA0★
Соединение	SR★	CO★	TS★
Данные	SR★	CO★	TS★
Рассоединение	SR★	CO★	TS★

6.1.17.10.1 **CA2★ Способность 2**

Эта линия определяется как Расщепленный ответ (SR★). Модули выставляют sr★, требуемый для того, чтобы текущая передача была расщеплена на множественные передачи или чтобы показать в течение ответа, что существуют все еще ответы, не обслуженные для запрошенных данных.

6.1.17.10.2 **CA1★ Способность 1**

Эта линия определяется как Принудительная (передача) (CO★). Модули выставляют co★, требуемый для того, чтобы текущая передача использовала принудительный протокол вместо пакетного протокола.

6.1.17.10.3 **CA0★ Способность 0**

Эта линия определяется как Скорость передачи (TS★). Модули выставляют сигнал ts★, запрашиваемый для того, чтобы была использована низшая из двух программно-управляемых скоростей передачи пакетов в течение текущей передачи. Если выставлен co★, тогда ts★ должен быть снят.

6.1.17.11 **AD[31 . . . 0]★ Адрес/Данные/Отмена байтовой шины/Адрес возврата**

В течение фазы соединения для большинства передач на линиях AD[31 . . . 5]★ находится адрес текущей передачи, при этом AD[1 . . . 0]★ определяются пользователем, AD[4 . . . 2]★ содержат либо адрес, либо информацию, определяемую пользователем, зависящую от выбранной разрядности данных (DW[]★).

Единственное исключение допускается в течение фазы соединения при передачах типа ответ чтения или ответ записи. В этом случае:

1) AD[31 . . . 28]★ устанавливаются в единицу;

2) AD[27 . . . 12]★ содержат глобальный идентификатор запросчика;

3) AD[11 . . . 6]★ являются резервными;

4) AD[5 . . . 0]★ содержат идентификатор передачи, который определяется пользователем.

В течение фазы данных передачи записи линии (AD[31 . . . 0]★) содержат данные записи, выставленные задатчиком. В течение фазы данных передачи чтения эти линии активированы одним или более исполнителями данными чтения.

Данные в 32 разряда всегда передаются по линиям AD[31 . . . 0]★ вне зависимости от разрядности модуля.

Байты передаются по своим естественным позициям без выравнивания. Порядок передачи байтов определяется в соответствующих стандартах.

В течение первого нечетного обмена частной передачи AD[31 . . . 0]★ определяются как сигналы отмены шины (L[]★) для передачи, которая происходит в течение последующих обменов. Частная посылка состоит из одного обмена отмены шины и одной или более обменов данными. В течение передач частной записи, если сигнал отмены шины выставлен, соответствующая байтовая шина не модифицируется принимающим исполнителем. В течение передач частного чтения, если сигнал отмены шины выставлен задатчиком, соответствующая байтовая шина не воспринимается задатчиком. Выставленный сигнал L[]★ запрещает проверку четности на соответствующей байтовой шине. Сигнал отмены шины, соответствующие не разрешенным шинам, могут быть отменены командой разрядности данных. Проверка байтовой четности на полных полях AD[]★ и D[]★ разрешается командой разрядности данных в течение обмена отмены шины.

В течение фазы рассоединения связных передач чтения AD[31 . . . 0]★ могут все еще содержать данные последнего обмена. Поэтому информация рассоединения не может быть помещена на этих линиях, и четность при этом не проверяется.

В течение фазы рассоединения передач типа ответ чтения и ответ записи производятся следующие операции.

1) AD[31 . . . 16]★ резервные.

2) AD[15 . . . 8]★ содержат приоритет первоначального запросчика передачи. Приоритет арбитража не должен быть использован, поскольку он может быть унаследован.

3) AD[7 . . . 3]★ резервные.

4) AD2★ содержит избранный статус SL★. Этот разряд выставляется модулем на запрос конечного получателя, как было сигнализировано в адресе запроса.

5) AD1★ содержит статус конфликта. Этот разряд устанавливается модулем на запрос конечного получателя, когда этот модуль обнаруживает потенциальное условие затыка, которое требует, чтобы запрошенная передача была возвращена назад по всему пути к источнику запроса. Это приводит к занятости от края до края, и запросчик не должен ожидать дальнейших ответов.

6) AD0★ содержит статус ошибки передачи TE★. Этот разряд выставляется, если условия ошибки обнаруживаются в любом месте вдоль пути запроса на передачу. Запросчик не должен ожидать дальнейших ответов.

В течение фазы рассоединения передач разделенного и модифицированного ответа производится следующее.

1) AD[31 . . . 28]★ устанавливаются в единицу.

2) AD[27 . . 12]★ содержат глобальный идентификатор запросчика.

3) AD[11 . . . 4]★ содержат приоритет первоначального запросчика передачи. Арбитраж приоритета не должен использоваться, поскольку он может быть унаследован.

4) AD3★ резервная.

5) AD2★ содержит избранный статус SL★. Этот разряд выставляется модулем на запрос конечного получателя, как было сигнализировано в адресе запроса.

6) AD1★ содержит статус конфликта. Этот разряд устанавливается модулем на запрос конечного получателя, когда этот модуль обнаруживает потенциальное условие тупика, которое требует, чтобы запрошенная передача была возвращена назад по всему пути к источнику запроса. Это приводит к занятости от края до края, и запросчик не должен ожидать дальнейших ответов.

7) AD0★ содержит статус ошибки передачи TE★. Этот разряд выставляется, если условия ошибки обнаруживаются в любом месте вдоль пути запроса на передачу. Запросчик не должен ожидать дальнейших ответов.

В течение фазы рассоединения других передач, не описанных выше, производится следующее.

1) AD[31 . . . 16]★ содержат глобальный идентификатор запросчика.

2) AD[15 . . . 8]★ содержат приоритет первоначального запросчика передачи.

3) AD[7 . . . 6]★ резервные.

4) AD[5 . . . 0]★ содержат идентификатор передачи.

6.1.17.12 **AD[63 . . . 32]★ Адрес/Данные**

В течение фазы соединения в передачах с выставленным AW★ на линиях AD[63 . . . 32]★ выставляется 32 старших разряда адреса текущей передачи. В течение фазы данных передачи записи при выставленных либо DW1★, либо DW0★ на этих линиях находятся данные записи задатчика. В течение фазы данных передачи чтения с выставленными сигналами либо DW1★, либо DW0★ на этиях линиях находятся данные чтения одного или более исполнителей.

В течение фазы рассоединения эти линии не определены. Дополнительные возможности этих линий в течение фазы рассоединения могут быть определены другими стандартами.

6.1.17.13 **D[255 . . . 64]★ Данные**

В течение фазы подключения D[255 . . . 64]★ не определены.

В течение фазы данных передачи записи при выставленном DW1★ на D[255 . . . 64]★ находятся данные записи задатчика. В течение фазы данных передачи записи с выставленными DW1★ и DW0★ на D[255 . . . 64]★ находятся данные записи задатчика. В течение фазы данных передачи чтения при выставленном DW1★ на D[127 . . . 64]★ находятся данные чтения исполнителя. В течение фазы данных передачи чтения при выставленных DW1★ и DW0★ на D[255 . . . 128]★ находятся данные чтения одного или более исполнителей.

В течение фазы рассоединения эти линии не определены. Дополнительные возможности этих линий в течение фаз подключения и рассоединения могут быть определены другими стандартами.

6.1.17.14 **BP [31 . . . 0]★ Четность байта**

Каждая линия байта четности несет бит четности для одной байтовой шины и, если число сигналов в каждой байтовой шине четное, то выставляется сигнал четности байта.

BP0★ соответствует четности AD[7 . . . 0]★, BP1★ соответствует четности AD[15 . . . 8]★, и т. д. до BP31★ соответствующей четности D[255 . . . 248]★.

6.1.17.15 **TG[7 . . . 0]★ Тег**

TG[7 . . . 0]★ линии несут дополнительную – по выбору информацию, относящуюся к линиям AD[63 . . . 0]★ и D[255 . . . 64]★. Информация, представленная сигналами на этих линиях, не регламентируется этим стандартом. Рекомендации, как они могут быть использованы, могут быть предписаны в будущем стандарте. Число активных линий TG[]★ согласовывается при инициализации и содержится в регистре логических возможностей модуля, как описано в 7.1.

6.1.17.16 **TP★ Четность тегов**

Линия четности тегов несет бит четности для TG[7 . . . 0]★ и, если число выставленных сигналов четно, то выставляется TP★.

Если линии тегов не активны (определяется регистром управления и статуса), сигналы на этих линиях не выставляются.

6.1.17.17 **ET★ Окончание владения**

Текущий задатчик снимает et★, чтобы показать, что его текущее владение магистралью закончилось. Снятие ET★ является индикацией того, что текущий задатчик освободил все линии параллельной магистрали и что выбранный задатчик может начать магистральную передачу.

Задатчик выставляет et★ в течение его первой передачи, когда он обнаруживает AIf в состоянии логического нуля, и перед снятием as★.

Снятие ET★ также делает заметным изменение владения магистралью для всех модулей, побуждая их к снятию любых блокировок ресурсов.

6.1.18 **Обмены в магистрали**

Обмен состоит из посылки по линии синхронизации, выполняемой задатчиком, следовавшим за снятием линии подтверждения одного или более исполнителей. Команда и данные могут быть посланы от задатчика к исполнителю(лям) в течение первой половины обмена, а статус, способности и данные могут быть возвращены задатчику от исполнителей в течение второй половины. Этот режим запрос—ответ относится к принудительному, так как исполнитель обязывается обеспечить ответ до того, как задатчик приступит к следующему обмену. Передачи на магистрали состоят из набора этих обменов, проводимых задатчиком и одним или более исполнителями.

6.1.18.1 **Магистральный обмен с одним исполнителем**

На рис. 6—4 показан общий обмен, в котором информация передается между задатчиком и единственным исполнителем. Детальное описание операции следующее.

1) Задатчик сначала активирует информационные линии с соответствующими информационными сигналами.

2) Задатчик затем ожидает окончания возможных перекосов между его передатчиками и распространением сигналов по магистрали, чтобы убедиться, что все информационные сигналы действительны перед выставлением сигналов синхронизации на магистрали.

3) Задатчик выставляет сигнал синхронизации.

4) Исполнитель обнаруживает сигнал синхронизации задатчика.

5) Исполнитель затем ожидает окончания возможных перекосов между его приемниками.

6) Далее исполнитель получает информацию от задатчика. (Эта фиксация включает дополнительное время, которое может быть необходимым для предустановки и удержания в цепях приема данных.

Этот метод синхронизации обеспечивает технологическую независимость каждого модуля, поскольку каждый учитывает только собственные перекосы и временные характеристики.

7) Исполнитель выставляет на информационные линии соответствующие информационные сигналы ответа.

8) Исполнитель ожидает окончания возможных перекосов между его передатчиками и распространением сигналов по магистрали, чтобы убедиться, что все информационные сигналы действительны перед выставлением сигнала синхронизации на магистрали.

9) Исполнитель затем снимает сигнал синхронизации.

10) Задатчик обнаруживает снятие сигнала синхронизации.

11) Задатчик может далее активировать информационные линии новой информацией. Затем задатчик ожидает окончания возможного перекоса в своих приемниках.

12) Задатчик фиксирует ответную информацию от исполнителя. (Фиксация включает любое дополнительное время, которое может быть необходимо, для предустановки и удержания в цепях фиксации данных).

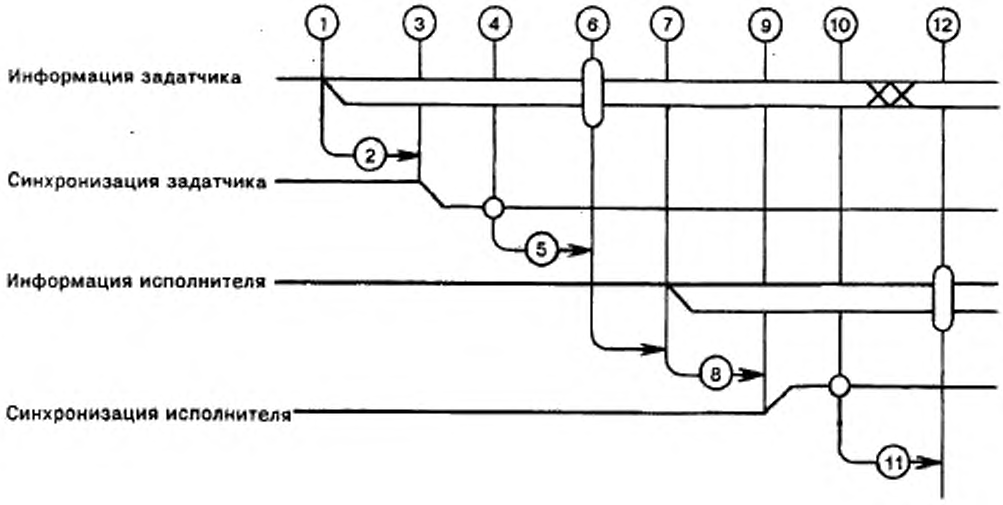

Рисунок 6—4 — Магистральный обмен с одним исполнителем

6.1.18.2 **Магистральный обмен со множественными исполнителями**

Рис. 6—5 показывает общий обмен, при котором информация передается между задатчиком и множественными исполнителями. Передачи, в которые вовлечены множество исполнителей, известны как широковещательные передачи. Различие между передачами со множественными и единичным исполнителями заключается в использовании сигналов синхронизации исполнителей. Первоначально все исполнители, вовлеченные в обмен, должны выставить сигнал синхронизации. Шаги 1—6 идентичны 6.1.18.1.

7) Исполнители помещают свою информацию на магистрали.

8) Исполнители ожидают окончания возможных перекосов.

9) Каждый исполнитель снимает свой сигнал синхронизации.

10) Поскольку сигнал синхронизации исполнителя, который появляется на магистрали, является сигналом проводного ИЛИ всех исполнителей, выставивших этот сигнал, все исполнители должны снять этот сигнал до того, как он снимется с магистрали. Это позволяет самому медленному из исполнителей управлять временем снятия. Побочный эффект на линии передачи, называемый «шпилькой» проводного ИЛИ, вызывается этой ситуацией. Получатель сигнала должен использоваться таким образом, чтобы воспринимать сигнал только после того, как тот пройдет интегратор проводного ИЛИ. Таким образом, обмены со множеством исполнителей производятся в темпе передачи с единственным самым медленным исполнителем.

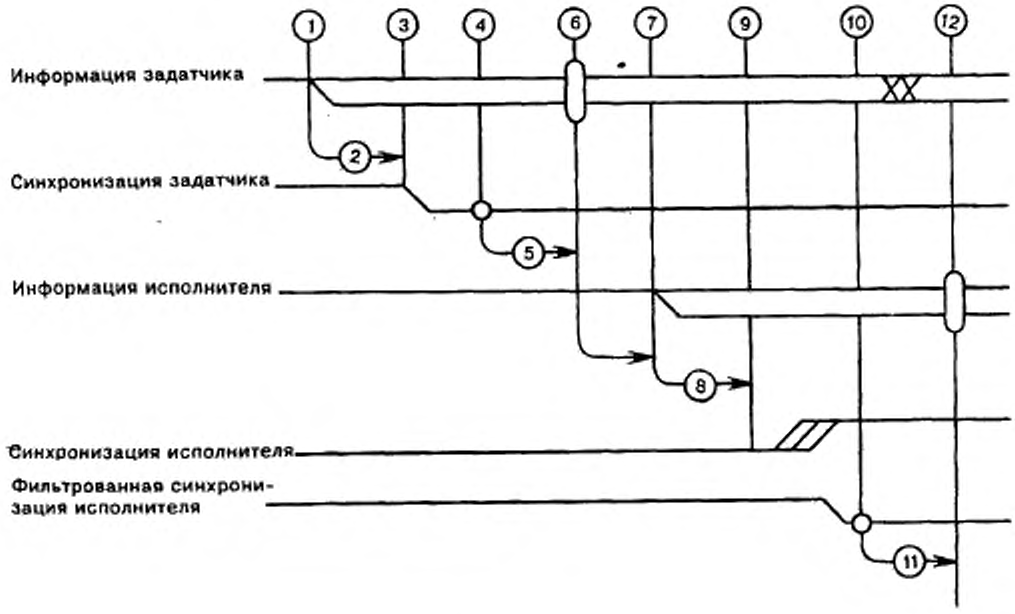

Рисунок 6—5 — Магистральный обмен со множественными исполнителями

6.1.19 **Примеры передач**

6.1.19.1 **Передача только адреса**

Передача только адреса состоит из двух последовательных обменов. Первый обмен заключается в установлении связи между задатчиком и одним или более исполнителями и поэтому относится к фазе соединения передачи. Вторым обменом выполняется окончание связи между задатчиком и подключенными исполнителями, он относится к фазе рассоединения. Рис. 6—6 показывает передачу только адреса.

Обмены на магистрали, при которых задатчик выставляет сигналы синхронизации, относятся к нечетным событиям. Соответственно, обмены на магистрали, при которых задатчик снимает сигналы синхронизации, относятся к четным событиям. Протоколы, в которых используются оба перепада сигнала синхронизации, называются запрос-ответ на каждый перепад.

Эта передача является основным элементом, предоставляющим задатчику два временных интервала, в которые он может передать информацию к одному или более исполнителям. Два временных интервала также обеспечивают задатчику получение информации от исполнителя(ей) в ответ.

Задатчик сначала активирует линии AD[]★ адресом, линии CA[]★ — информацией способности, CM[]★ линии — соответствующими битами команды соединения, TG[]★ — теговой информацией и соответствующей линией четности; далее он выставляет as★, чтобы показать их действительность. Когда исполнители обнаруживают выставленным AS★, они выставляют ak★ и проверяют информацию об адресе и команде. Если адрес является одним из тех, на который они должны ответить в качестве выбранного или участвующего исполнителя, они выставляют di★. Выставление ak★ и di★ в фазе соединения является просто подготовкой к последующему использованию. Выбранные исполнители выставляют sl★, как показано на рис. 6—6, чтобы сообщить задатчику, что они отвечают на передачу. Исполнители также выставляют все линии с информацией о статусе и способности, которая им присуща. Далее исполнители снимают al★. Так как не все исполнители снимают al★, на выходе проводного ИЛИ могут проскакивать шпильки. Поэтому задатчик отслеживает AL★ через интегратор проводного ИЛИ. Когда задатчик обнаруживает AIf в состоянии логического нуля, он проверяет линии статуса и способности.

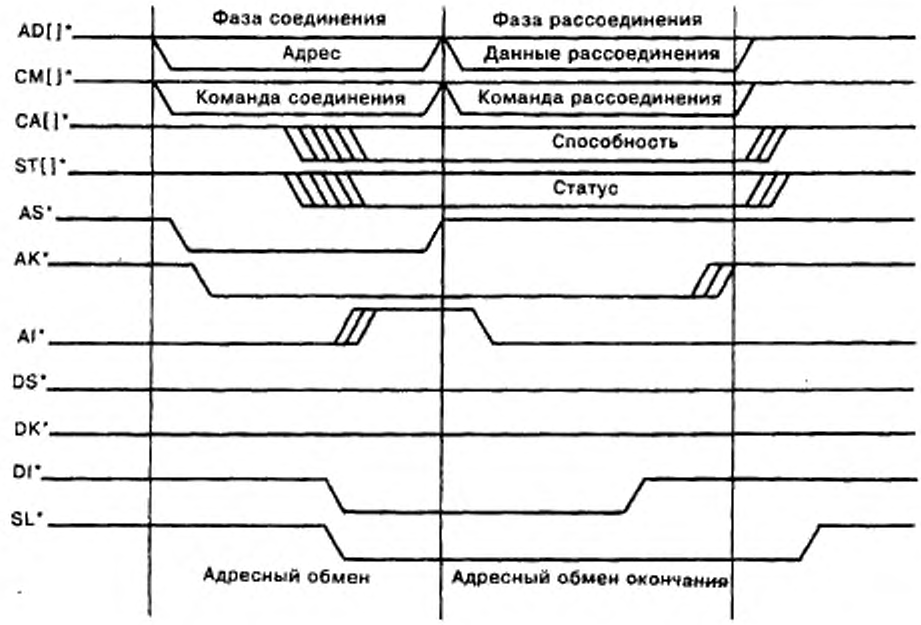

Рисунок 6—6 — Передача только адреса

Задатчик может заранее выбрать передачу только адреса или, если это необходимо, преобразовать в передачу только адреса, основываясь на информации о статусе и способности. Например, если задатчик обнаружил SR★ выставленным, он может преобразовать передачу чтения в запрос на чтение.

В случае расщепленной передачи ответа задатчик активирует AD[]★ и BP[]★ данными рассоединения, CM[]★ и CP★ — командой рассоединения и TG[]★ и TP[]★ — соответствующей теговой информацией. Затем задатчик снимает as★, чтобы указать на действительность AD[]★, BP[]★, TG[]★, TP★, CM[]★, CP★ и показать участвующим исполнителям, что передача заканчивается. Исполнители после обнаружения снятия AS★ проверяют информацию на AD[]★, BP[]★, TG[]★, TP★, CM[]★, CP★ и ST[]★, затем выставляют ai★ и снимают ak★, di★. Когда задатчик и исполнители обнаруживает AKf в состоянии логического нуля, они снимают AD[]★, BP[]★, TG[]★, TP★, CM[]★, CP★, CA[]★, ST[]★ и передача заканчивается.

6.1.19.2 **Принудительное чтение**

Передача принудительное чтение (см. рис. 6—7) начинается с фазы соединения, как в передаче только адреса. После того, как задатчик обнаруживает AIf в состоянии логического нуля и проверяет CA[]★ и ST[]★, он активирует CM[]★ соответствующей командой фазы данных и снимает AD[]★ и TG[]★. Далее задатчик выставляет ds★, чтобы запросить данные от исполнителя. Когда выбранный исполнитель обнаруживает выставленный DS★, он проверяет CM[]★ и выставляет dk★. Когда запрашиваемые данные доступны, исполнитель активирует AD[]★ и TG[]★. Затем исполнитель снимает di★, чтобы показать задатчику, что данные действительны. Когда задатчик обнаруживает снятие DI★, он фиксирует данные и снимает ds★, чтобы запросить данные со следующих адресов. Когда исполнитель обнаруживает снятие DS★, он выставляет di★. Когда запрашиваемые данные становятся доступными, исполнитель активирует AD[]★, TG[]★ и, например, выставляет ed★, чтобы показать, что он не может поддерживать дальнейшую передачу данных до конца текущей передачи. Затем исполнитель снимает dk★. Когда задатчик обнаруживает снятие DK★, он фиксирует данные на AD[]★ и приступает к фазе рассоединения, как при передаче только адреса.

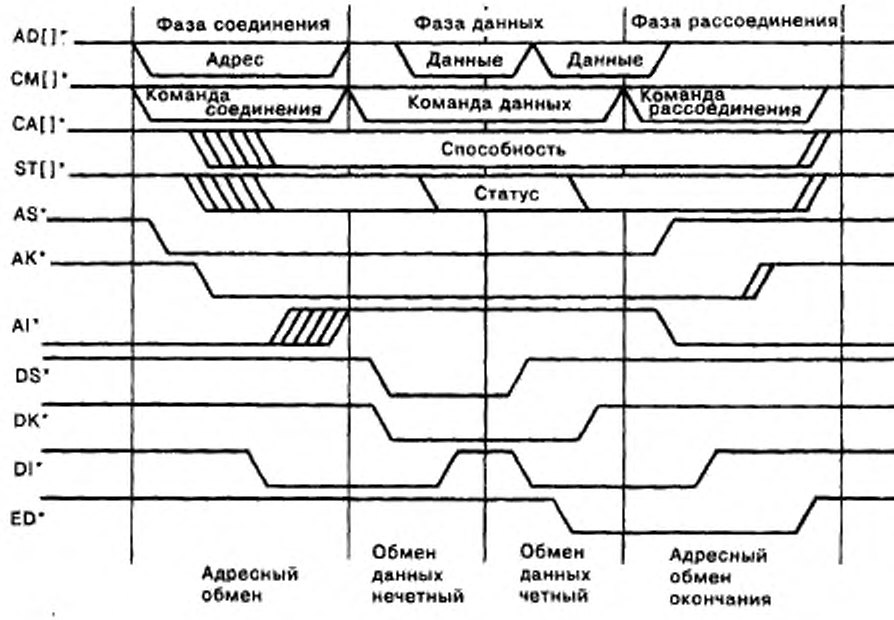

Рисунок 6—7 — Передача принудительное чтение

6.1.19.3 **Принудительное широковещательное/широкозапросное чтение**

Принудительное широковещательное чтение отличается от предыдущей передачи тем, что один или более исполнителей выставляют bs★ во время фазы соединения. Это показывает задатчику, что он должен проверить DI★ и DK★ после того, как они пройдут через интеграторы проводного ИЛИ, т. к. несколько исполнителей снимают эти сигналы. В этой передаче задатчик запрашивает единичную передачу. Это оставляет как DS★, так и DK★ выставленными до начала фазы рассоединения. В этом случае исполнители снимают dk★, когда они обнаруживают снятие AS★. Задатчик снимает DS★, когда он определит, что DKf стал логическим нулем. Это возвращает DS★ в снятое состояние.

Один и тот же протокол используется для двух возможных типов передачи. широкозапросной и широковещательной.

Широкозапросная (см. рис. 6—8) передача производится, когда несколько исполнителей посылают данные, а задатчик принимает данные по проводному ИЛИ от всех исполнителей. Например, когда каждый модуль использует только одну линию магистрали, статус может быть принят от всех модулей одновременно.

Широковещательная передача производится, когда один из исполнителей выставляет данные, а один или более исполнителей принимают данные вместе с задатчиком. При широковещательном чтении только модуль, выставляющий данные на шину, устанавливает dk★, показывая, что данные действительны. Другие модули ждут, пока они не обнаружат DK★, выставляемый перед выставлением dk★.

6.1.19.4 **Принудительное чтение с внедренностью**

В принудительном чтении с внедренностью (см. рис. 6—9) исполнитель выставляет iv★ во время фазы соединения, чтобы показать, что он имеет более современную копию данных, запрошенных задатчиком, чем выбранный исполнитель. Задатчик по обнаружении IV★ отвечает выставлением sw★ во время фазы данных. Выбранный исполнитель, обнаружив SW★, выставляет подготовленные для чтения данные вместо их источника. Поскольку в чтение вовлечено более одного исполнителя, задатчик должен следить за DK★ и DI★ после того, как они пройдут через интеграторы проводного ИЛИ.

При чтении с внедренностью только модуль—источник данных первым выставляет dk★. Другие модули ждут, пока они не обнаружат DK★ перед выставлением своего dk★. Это же справедливо и в четном обмене данными за исключением того, что модуль—источник данных первым выставляет di★. Другие модули ждут, пока они не обнаружат DI★ перед выставлением своего di★.

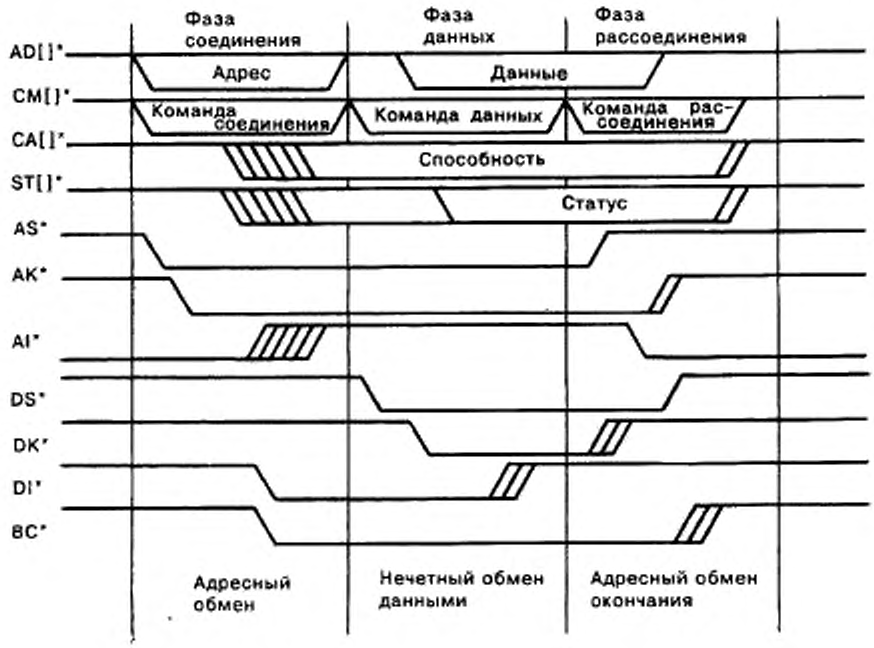

Рисунок 6—8 — Принудительное широкозапросное чтение

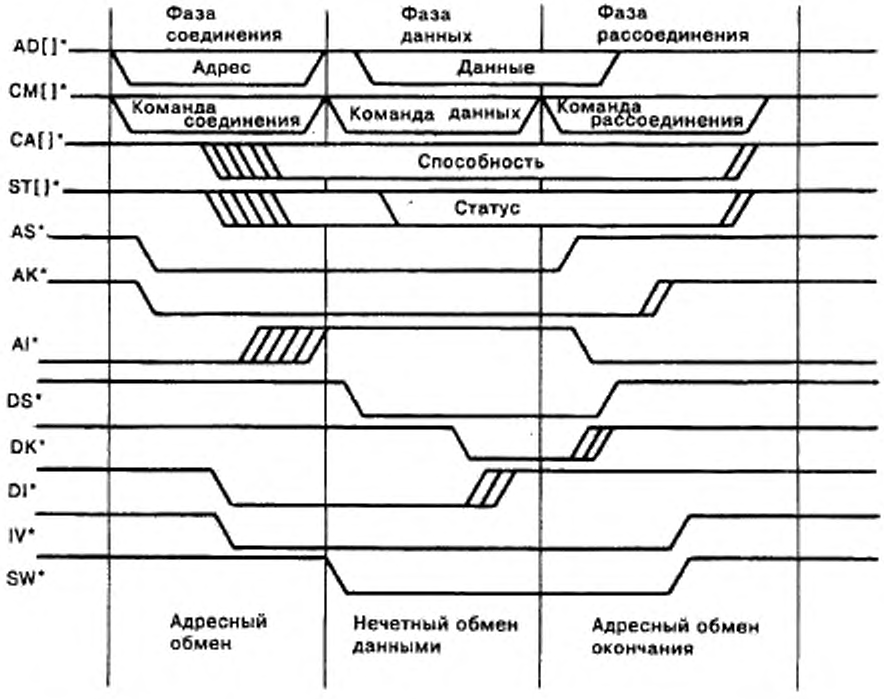

Рисунок 6—9 — Принудительное чтение с внедренностью

6.1.19.5 **Частное принудительное чтение**

Частное принудительное чтение (см. рис. 6—10) используется для передачи только отдельных байтов поля данных. В первом нечетном обмене задатчик передает информацию отмены шины на AD[31 . . 0]★. В четном и всех последующих обменах данных исполнитель(и) выставляют запрошенные данные. Задатчик при получении данных игнорирует байтовые шины, если установлен соответствующий сигнал отмены шины (см. 6.1.17.11).

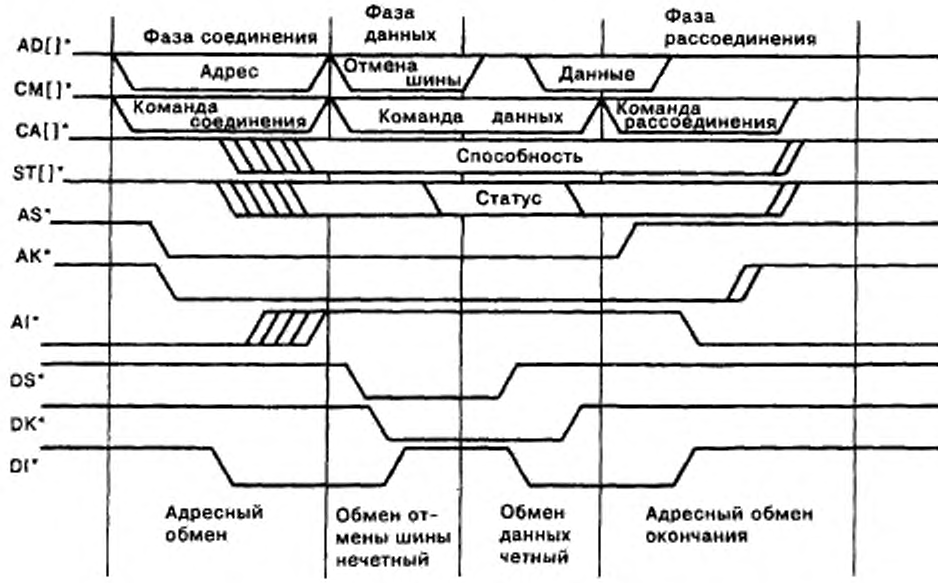

Рисунок 6—10 — Частное принудительное чтение

6.1.19.6 **Принудительная запись**

Принудительная запись (рис. 6—11) начинается с фазы соединения, как и при передаче только адреса. После того, как задатчик обнаруживает, что AIf стал логическим нулем, и проверит CA[]★ и ST[]★, он активирует CM[]★ соответствующей командой фазы данных AD[]★, D[]★ — данными записи и TG[]★ — теговой информацией. Затем задатчик выставляет ds★, чтобы показать исполнителю, что CM[]★, AD[]★ и TG[]★ действительны. Когда выбранный исполнитель обнаружит DS★, он выставляет dk★ и фиксирует данные на AD[]★ и TG[]★. Затем исполнитель снимает di★, чтобы показать задатчику, что он зафиксировал данные и что ST[]★ действительны. Когда задатчик обнаруживает, что DI★ снят, он фиксирует сигналы статуса исполнителя, выставляет на AD[]★ следующие данные записи, на TG[]★ — следующую соответствующую теговую информацию и снимает ds★. Когда выбранный исполнитель определяет, что DS★ снят, он выставляет di★ и фиксирует данные на AD[]★ и TG[]★. Затем исполнитель снимает dk★, чтобы показать задатчику, что он зафиксировал данные и что ST[]★ действительны. Когда задатчик обнаруживает, что DK★ снят, он фиксирует статус исполнителя и переходит к фазе рассоединения, если ED★ был выставлен исполнителем или все данные уже переданы.

6.1.19.7 **Принудительная широковещательная запись**

Принудительная широковещательная запись (см. рис. 6– 12) идентична предыдущей передачи за исключением того, что один или более исполнителей выставляют bc★ во время фазы соединения. Это показывает задатчику, что он должен наблюдать за состоянием DI★ и DK★ после того, как они пройдут через интеграторы проводного ИЛИ, т. к. несколько исполнителей снимают эти сигналы.

Все участвующие исполнители принимают данные от задатчика в этой передаче.

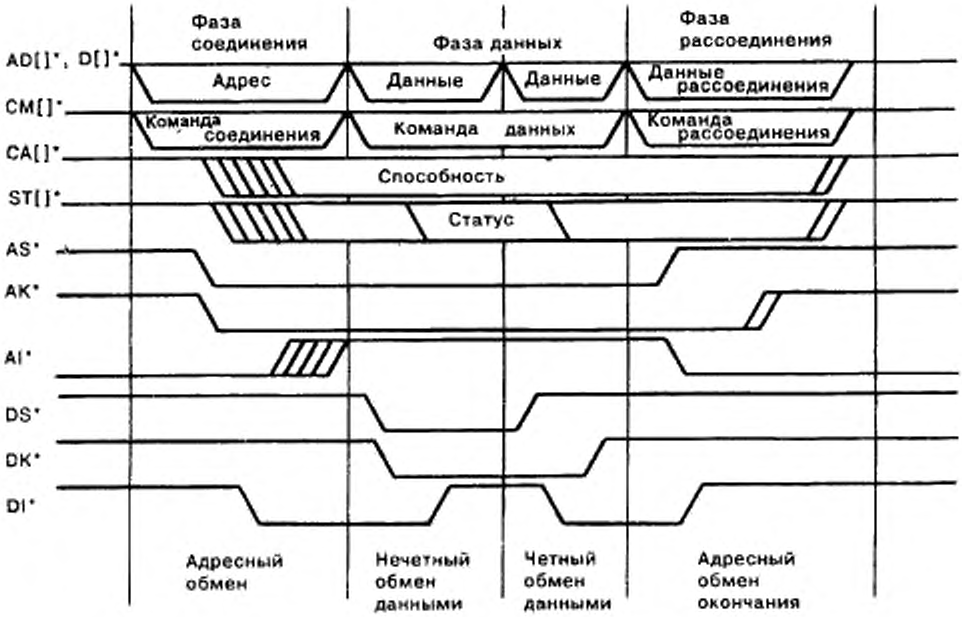

Рисунок 6—11 — Принудительная запись

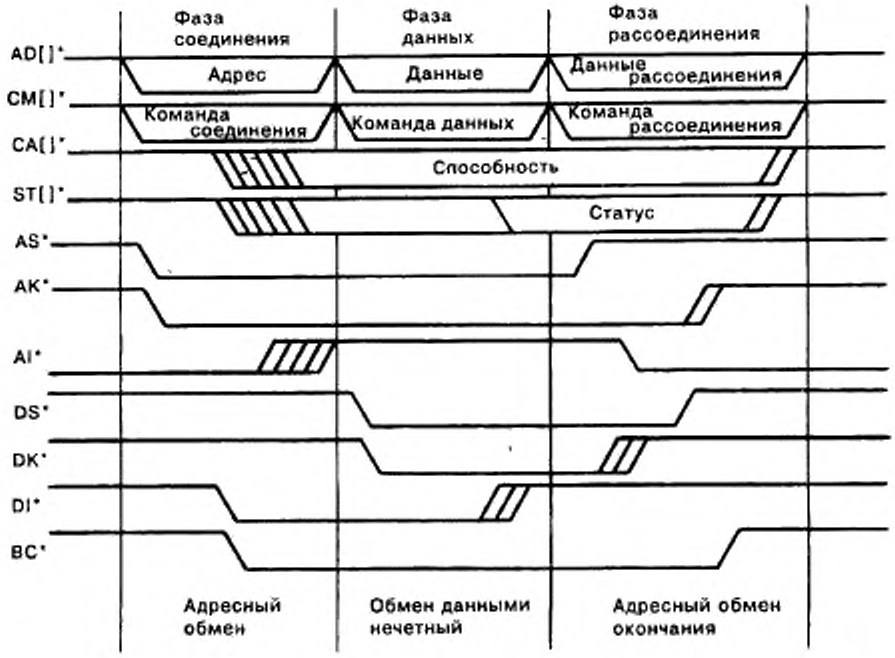

Рисунок 6—12 — Принудительная широковещательная запись

6.1.19.8 **Частная принудительная запись**

Частная принудительная запись (см. рис. 6—13) используется, чтобы передать только отдельные байты поля данных. В первом нечетном обмене данных задатчик передает информацию отмены байтовой шины на AD[31 . . . 0]★. В первом четном и последующих обменах задатчик передает данные записи. Получая данные, исполнитель(и) игнорируют байтовые шины, если установлен соответствующий сигнал отмены шины (см. 6.1.17.11)

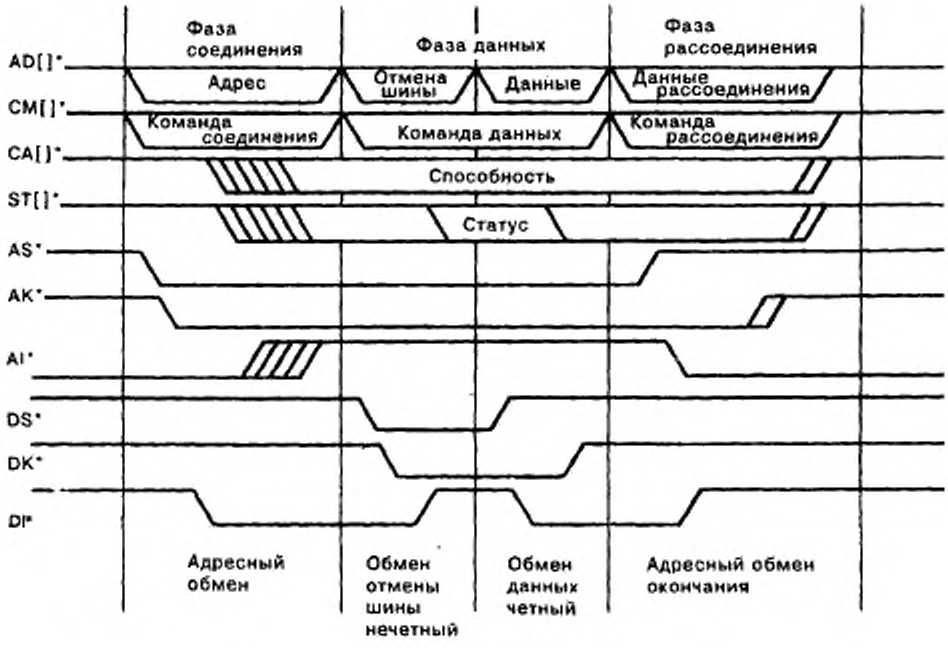

Рисунок 6—13 — Принудительная частная запись

6.1.19.9 **Ошибка передачи**

В предыдущих примерах передачи были введены различные статусные линии. TE★ отличается от всех других статусных линий тем, что он действителен между передачами, как показано на рис. 6—14. Модуль выставляет te★ во время фазы рассоединения или фазы данных перед снятием ak★. Модуль удерживает te★ выставленным, пока не обнаружит, что AIf стал логическим нулем во время фазы соединения в следующей передаче. Это позволяет всем модулям сообщить об ошибках независимо от того, в каком месте передачи ошибки обнаружены. Все модули на шине могут определять состояние TE★ с того времени, как они обнаружили, что AKf стал логическим нулем или AS★ выставлен, пока они не сняли ai★

6.1.19.10 **Пакетная фаза данных**

Одиночный нечетный пакет пакетной передачи показан на рис. 6—15. Фазы соединения и рассоединения пакетной передачи идентичны фазам, описанным для принудительной передачи. Рисунок иллюстрирует передачу восьмисловного пакета (например, передачу 64 байтов через 64-разрядную систему P896.1) В восьмисловной передаче существуют десять битовых периодов. Каждый период равен периоду локального таймера передающего модуля. Передача данных начинается с выставления всех линий адрес/данные передающим модулем. Этот переход определяет начало первого битового периода. Приемный модуль синхронизируется каждым индивидуальным битом адрес/данные, используя этот первоначальный переход. Передающий модуль использует переход в битовом периоде, чтобы указать «единицу». Битовый период без перехода является «нулем». Первый битовый период используется как бит синхронизации. Последний бит используется для про-

верки на четность. Если число предыдущих битовых периодов нечетно, то в бите четности будет «единица». Если не обнаружено ошибок четности, то сигналы со всех линий будут сняты при окончании передачи.

Чтобы декодировать эту информацию, принимающий модуль синхронизуется каждым битом независимо, а затем использует внутреннюю схему для таймирования последовательных битовых периодов.

В пакетной моде сигнал ошибки вырабатывается только на уровне передачи, используя TE★ так, что скорость сохраняется.

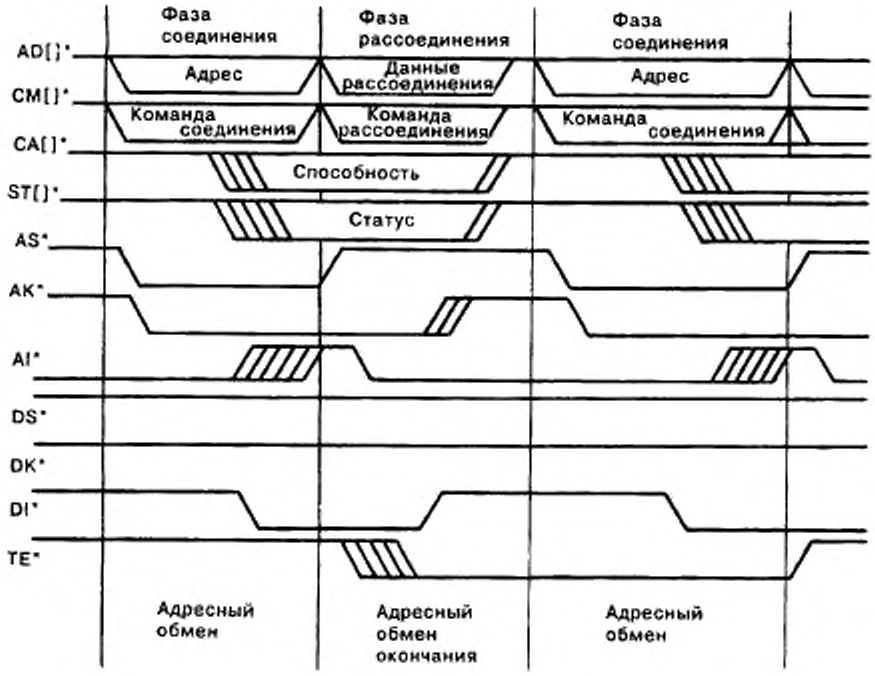

Рисунок 6—14 — Диаграмма ошибки передачи

6.1.19.11 **Одиночное пакетное чтение**

При одиночном пакетном чтении, показанном на рис. 6—16, исполнитель является источником пакета данных, пока он не обнаружит DS★, выставленный задатчиком. В приведенном примере исполнитель задерживает снятие di★ до тех пор, пока пакет не будет передан полностью, хотя это не является обязательным. Задатчик всегда ждет, когда пакеты будут полностью переданы, потом снимает as★. В отличие от принудительного протокола, задатчик может снять ds★, как только он обнаружит, что DI★ снят, без предварительного снятия as★, в то время как он не устанавливает pr★ в команде данных.

6.1.19.12 **Одиночное пакетное широковещательное чтение**

Рис. 6—17 показывает передачу одиночного пакетного чтения с одним или более исполнителями, выставляющими bs★ статусную линию. В этом случае один или более исполнителей принимают пакет вместе с задатчиком.

6.1.19.13 **Одиночное пакетное чтение с внедренностью**

Передача одиночного пакетного чтения с внедренностью, показанная на рис. 6—18, идентична передаче в принудительной версии, за исключением того, что передается пакет.

6.1.19.14 **Одиночная пакетная запись**

Передача одиночной пакетной записи, показанная на рис. 6—19, идентична передаче в принудительной версии, за исключением того, что передается пакет. Задатчик задерживает посылку пакета на время, по крайней мере равное одному битовому периоду после выставления ds★. Исполнитель должен подготовить свою внутреннюю схему для принятия пакета за этот период времени.

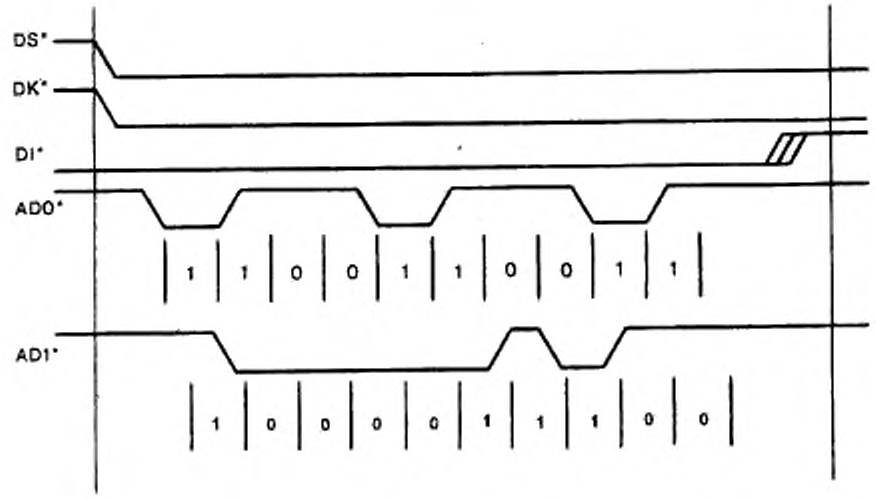

Рисунок 6—15 — Диаграмма пакетной фазы данных

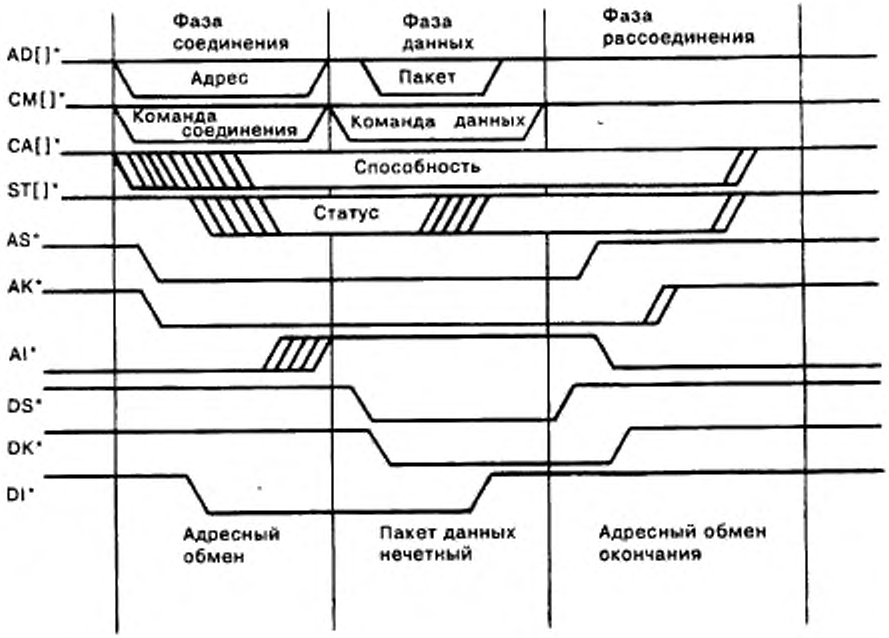

Рисунок 6—16 — Диаграмма одиночного пакетного чтения

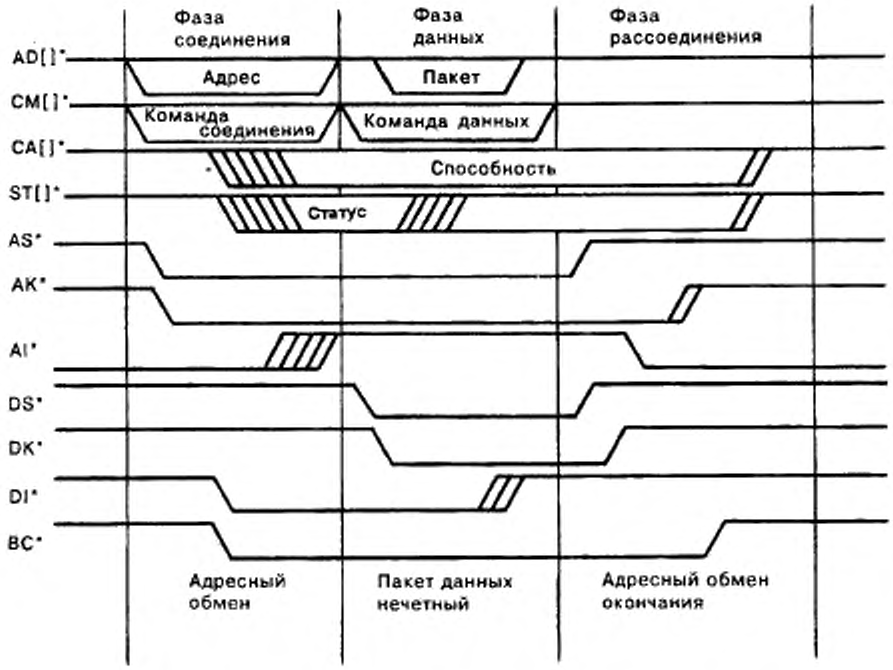

Рисунок 6—17 — Широковещательное одиночное пакетное чтение

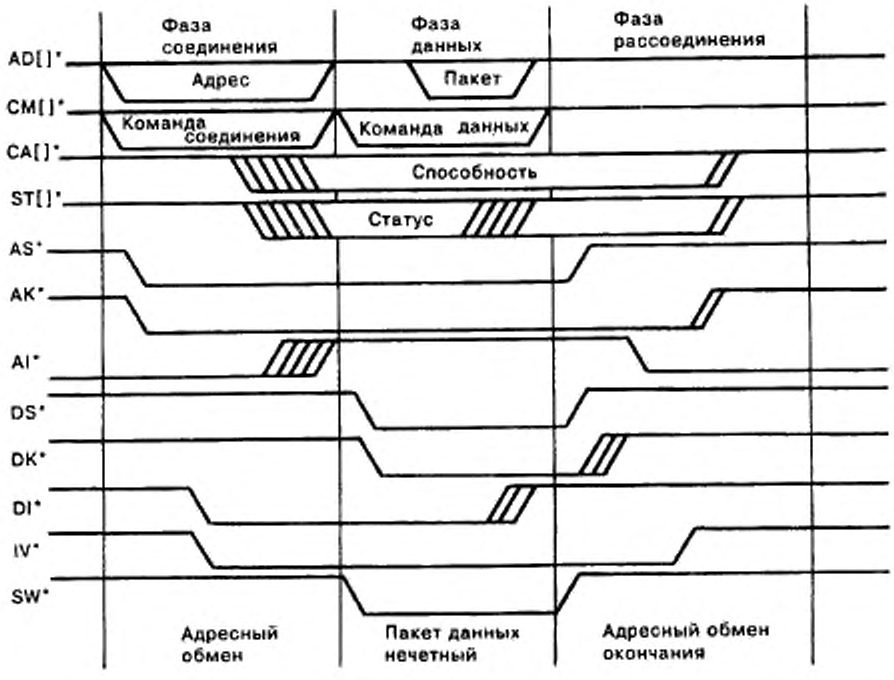

Рисунок 6—18 — Одиночное пакетное чтение с внедренностью

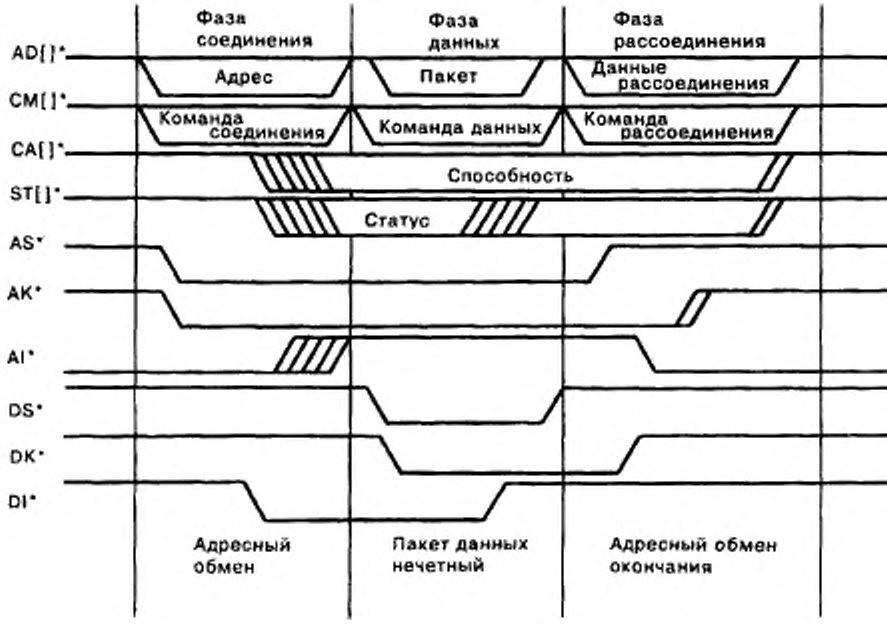

Рисунок 6–19 — Одиночная пакетная запись

6.1.19.15 **Многократное пакетное чтение с внедренностью**

Во время инициализации системы главный процессор определяет, все ли модули, способные к пакетной передаче, могут работать в многократной пакетной моде. Если все способные к пакетной передаче модули поддерживают многократную пакетную передачу, тогда может быть достигнута очень высокая скорость блочной передачи по отношению к физическим адресам. Если только часть способных к пакетной передаче модулей поддерживают многократную пакетную передачу, то многократная пакетная мода может быть использована, только когда выбранные модули имеют точную информацию о системных границах физических адресов. В этом случае модули—участники многократной пакетной моды должны гарантировать, что передача не потребует пакета, который может вовлечь неспособный к многократной пакетной передаче модуль.

Внутри фазы данных многократной пакетной моды каждый новый пакет запрашивается задатчиком, и каждый новый запрос подтверждается статусной информацией с использованием принудительного обмена всеми участниками. В любое время модуль может отказать запросу выставлением сигнала статуса конца данных. Задатчик использует rr★ для запроса добавочных пакетов. Если исполнитель(и) неспособны поддержать или принять запрошенные данные, они выставляют ed★.

Рисунок 6—20 показывает многократное пакетное чтение с внедренностью. IV★ и SW★ активируются на пакетный базис, а не базис передачи. Это позволяет использовать передачу с кешированным оборудованием, которая автоматически собирает и делает недействительными кешевые строки, расположенные среди кешей и системной памяти. Каждый цикл запрос/ответ становится наблюдающим следующей кешевой строки. Кеши возвращают свое кешевое разделение и статус внедрения, как они делали бы это во время фазы соединения.

6.1.19.16 **Многократное пакетное чтение с внедрением и очередностью**

Протокол запрос/ответ при многократной пакетной передаче не обязательно должен находиться в фазе с реальной передачей данных. С условной очередностью аппаратные циклы запрос/ответ могут проходить перед непрерывно передаваемым пакетом.

Рис. 6—21 показывает передачу многократного пакетного чтения, в которой пакеты передаются значительно позже, чем циклы запрос/ответ. Все участвующие модули должны следить за линиями AD[]★, чтобы определить, когда они должны выставить данные.

Если кеш-когерентность не используется, то можно использовать сверхскоростную блочную передачу с возможностью пересекать границы модулей, пока цикл запрос/ответ проводит конвейр с передачей данных.

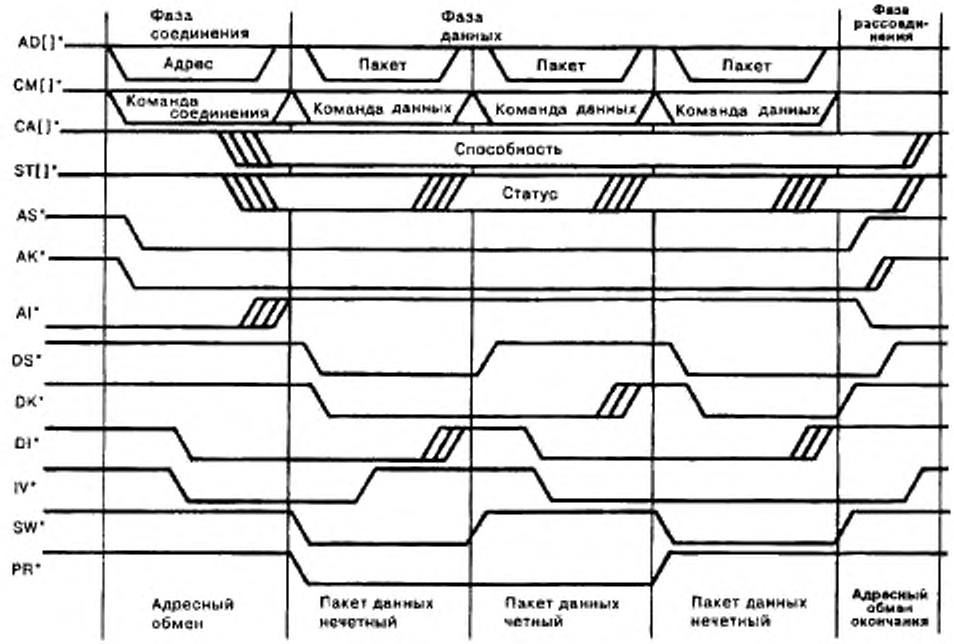

Рисунок 6—20 – Многократное пакетное чтение с внедрением

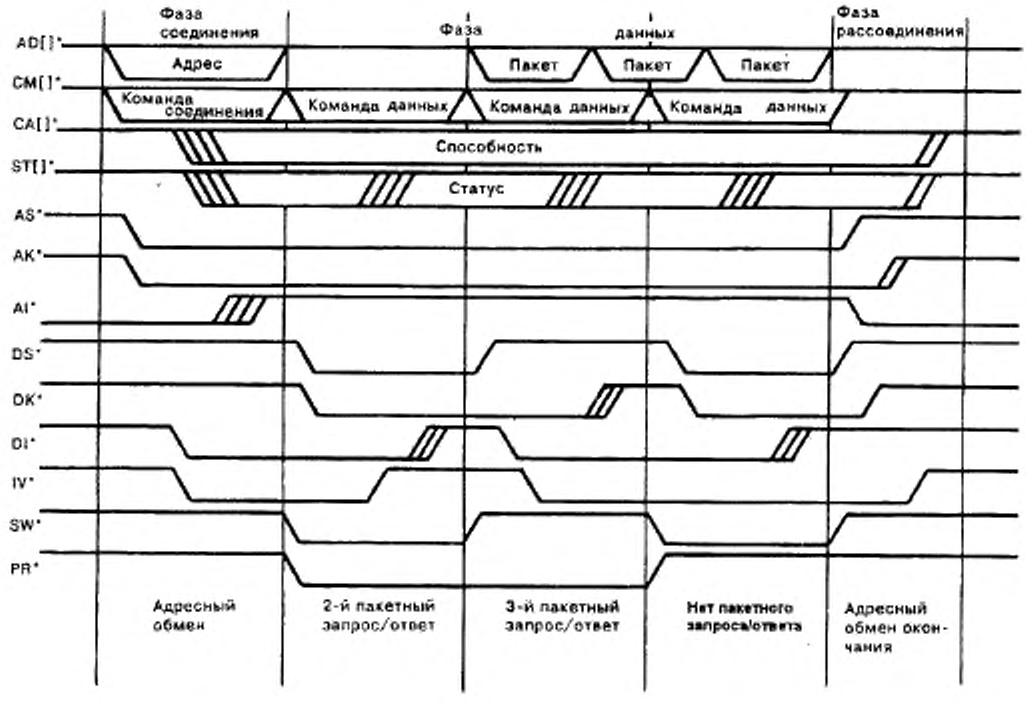

Рисунок 6—21 — Многократное пакетное чтение с внедрением и очередностью

6.2 **Спецификация**

6.2.1 **Основные определения**

6.2.1.1 ***Определения протокола передачи***

ЗАДАТЧИК MASTER

Модуль должен установить ЗАДАТЧИК, если ЦЕНТРАЛИЗОВАННЫЙ_ЗАДАТЧИК | РАСПРЕДЕЛЕННЫЙ_ЗАДАТЧИК

РАЗРЕШЕННОСТЬ_ПАКЕТА PACKET_ENABLED

Модуль должен установить РАЗРЕШЕННОСТЬ_ПАКЕТА, если ПАКЕТНАЯ_СПОСОБНОСТЬ & (РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_2 | РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_4 | РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_8 | РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_16 | РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_32 | РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_64)

ПРИНУДИТЕЛЬНОСТЬ COMPELLED

Модуль может установить ПРИНУДИТЕЛЬНОСТЬ, если -ИНИЦ & (ЗАДАТЧИК & ФАЗА_СОЕДИНЕНИЯ | УЧАСТИЕ) Модуль должен установить ПРИНУДИТЕЛЬНОСТЬ, если-ИНИТ & (-РАЗРЕШЕННОСТЬ_ПАКЕТА & (ЗАДАТЧИК & ФАЗА_СОЕДИНЕНИЯ | УЧАСТИЕ) | ФАЗА_СОЕДИНЕНИЯ & ЗАПРЕЩЕННЫЙ). Модуль должен удерживать ПРИНУДИТЕЛЬНОСТЬ, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

СТАТУС_ПРИНУЖДЕНИЯ COMPELLED_STATUS

Модуль должен установить СТАТУС_ПРИНУЖДЕНИЯ, если-ИНИТ & CA1★ & СТАТУС_СОЕДИНЕНИЯ. Модули должны удерживать СТАТУС_ПРИНУЖДЕНИЯ, пока ИНИТ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

НИЗКАЯ_СКОРОСТЬ LOW_SPEED

Модуль должен установить НИЗКУЮ_СКОРОСТЬ, если -ИНИТ & —РАЗРЕШЕНИЕ ВЫСОКОЙ_СКОРОСТИ & (ЗАДАТЧИК & ФАЗА_СОЕДИНЕНИЯ & -ПРИНУДИТЕЛЬНОСТЬ | УЧАСТИЕ & -ПРИНУДИТЕЛЬНОСТЬ). Модули должны удерживать НИЗКУЮ_СКОРОСТЬ, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

СТАТУС_НИЗКОЙ_СКОРОСТИ LOW_SPEED_STATUS

Модули должны установить СТАТУС_НИЗКОЙ_СКОРОСТИ, если -ИНИЦ & CA0★ & СТАТУС_СОЕДИНЕНИЯ. Модули должны удерживать СТАТУС_НИЗКОЙ_СКОРОСТИ, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ MULTIPLE_PACKET_MODE

Модули должны установить РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ, если ФАЗА_ДАННЫХ & -СТАТУС_ПРИНУЖДЕНИЯ & РАЗРЕШЕНИЕ_РЕЖИМА_МНОЖЕСТВЕННЫХ_ПАКЕТОВ. Модули должны удерживать РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ЗАПРОС_ПАКЕТА PACKET_REQUEST

ЗАДАТЧИК должен установить ЗАПРОС_ПАКЕТА, если -ИНИЦ & РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ перед выставлением ds★, чтобы начать ФАЗУ_ДАННЫХ, если в текущей передаче запрошено более одного пакета. ЗАДАТЧИК должен удерживать ЗАПРОС_ПАКЕТА, пока ИНИЦ | НУЛЬ СЧЕТЧИКА ПЕРЕДАЧ & ЗАНЯТИЕ ДИСКА_ИНФО | ЗАВЕРШЕНИЕ ПЕРЕДАЧИ.

Модули должны удерживать ЗАПРОС_ПАКЕТА, если СМ0★ &УЧАСТИЕ & РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ & ПРИЗНАК_ДАННЫХ_ФИКС. Модули должны удерживать ЗАПРОС_ПАКЕТА, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ | -СМ0★ & ПРИЗНАК_ДАННЫХ ФИКС.

6.2.1.2 **Определения расщепленной передачи**

РАСЩЕПЛЕНИЕ SPLIT

Модули могут вызывать расщепленную передачу, устанавливая РАСЩЕПЛЕНИЕ, если -ИНИТ & РАСЩЕПЛЕННЫЙ_ИНИЦИАТОР & РАСЩЕПЛЕНИЕ_РАЗРЕШЕНО & (ЗАДАТЧИК & ФАЗА_СОЕДИНЕНИЯ | УЧАСТИЕ) & ЗАПРЕТ_УДЕРЖАНИЯ | РАСЩЕПЛЕННЫЙ_КЕШИРОВАННЫЙ). Модули должны удерживать РАСЩЕПЛЕНИЕ, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

РАСЩЕПЛЕНИЯ_СТАТУС SPLIT_STATUS

Модули должны устанавливать РАСЩЕПЛЕНИЯ_СТАТУС, если -ИНИЦ & CA2★ & СТАТУС_СОЕДИНЕНИЯ. Модули должны удерживать РАСЩЕПЛЕНИЯ_СТАТУС, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

6.2.1.3 **Определения разрядности Адреса/Данных**

32_РАЗРЯДНАЯ_СПОСОБНОСТЬ 32_BIT_CAPABLE

Все модули должны быть способны к передаче 32-разрядных адресов и данных.

РАЗРЯДНОСТЬ_АДРЕСА ADDR_WIDTH

Задатчик должен выбирать передачу с 64-разрядным адресом, устанавливая РАЗРЯДНОСТЬ_АДРЕСА, если -ИНИЦ & ФАЗА_СОЕДИНЕНИЯ & СПОСОБНОСТЬ_64_РАЗРЯДНОГО_АДРЕСА & -(ОТВЕТ_ЧТЕНИЯ | ОТВЕТ_ЗАПИСИ). ЗАДАТЧИК должен удерживать ШИРИНУ_АДРЕСА, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

64_РАЗРЯДНЫЙ_АДРЕС 64_BIT_ADDR

Модули должны устанавливать 64_РАЗРЯДНЫЙ_АДРЕС если СМ7★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ | РАЗРЯДНОСТЬ_АДРЕСА. Модули должны удерживать 64_РАЗРЯДНЫЙ_АДРЕС, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

РАЗРЯДНОСТЬ_ДАННЫХ_1 DATA_WIDTH_1

Чтобы выбрать 128- или 256-разрядную разрядность данных на шине, ЗАДАТЧИК должен установить РАЗРЯДНОСТЬ_ДАННЫХ_1, если -ИНИЦ & ФАЗА_СОЕДИНЕНИЯ. ЗАДАТЧИК должен удерживать РАЗРЯДНОСТЬ_ДАННЫХ_1, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

РАЗРЯДНОСТЬ_ДАННЫХ_0 DATA_WIDTH_0

Чтобы выбрать 64- или 256-разрядную разрядность данных на шине, ЗАДАТЧИК должен установить РАЗРЯДНОСТЬ_ДАННЫХ_0, если -ИНИЦ & ФАЗА_СОЕДИНЕНИЯ. ЗАДАТЧИК должен удерживать РАЗРЯДНОСТЬ_ДАННЫХ_0, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

64_РАЗРЯДНЫЕ_ДАННЫЕ 64_BIT_DATA

Модули должны установить 64_РАЗРЯДНЫЕ_ДАННЫЕ, если -СМ6★ & СМ5★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ | -РАЗРЯДНОСТЬ_ДАННЫХ_1 & РАЗРЯДНОСТЬ_ДАННЫХ_0, и должны удерживать это, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

128_РАЗРЯДНЫЕ_ДАННЫЕ 128_BIT_DATA

Модули должны установить 128_РАЗРЯДНЫЕ_ДАННЫЕ, если СМ6★ & -СМ5★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ , ШИРИНА_ДАННЫХ_1 & -ШИРИНА_ДАННЫХ_0, и должны удерживать это, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

256_РАЗРЯДНЫЕ_ДАННЫЕ 256_BIT_DATA

Модули должны установить 256_РАЗРЯДНЫЕ_ДАННЫЕ, если СМ6★ & СМ5★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ | РАЗРЯДНОСТЬ_ДАННЫХ_1 & РАЗРЯДНОСТЬ_ДАННЫХ_0, и должны удерживать это, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

6.2.1.4 **Определения передачи**

НЕБЛОКИРОВАНИЕ_ЧТЕНИЕ READ_UNLOCKED

ЗАДАТЧИК должен установить НЕБЛОКИРОВАНИЕ_ЧТЕНИЕ перед началом ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу чтения без блокирования. Модули должны установить НЕБЛОКИРОВАНИЕ_ЧТЕНИЕ, если -СМ4★ & -СМ3★ & -СМ2★ & -СМ1★ & -СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать НЕБЛОКИРОВАНИЕ_ЧТЕНИЕ, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

НЕБЛОКИРОВАНИЕ_ЗАПИСЬ WRITE_UNLOCKED

ЗАДАТЧИК должен установить НЕБЛОКИРОВАНИЕ_ЗАПИСЬ перед началом ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу записи без блокирования. Модули должны установить НЕБЛОКИРОВАНИЕ_ЗАПИСЬ, если -СМ4★ & -СМ3★ & -СМ2★ & -СМ1★& -СМ0 & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать НЕБЛОКИРОВАНИЕ_ЗАПИСЬ, пока ИНИЦ , ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ | ФАЗА_РАССОЕДИНЕНИЯ & НЕБЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС.

НЕБЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС ADDRESS_ONLY_UNLOCKED

ЗАДАТЧИК должен установить НЕБЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС перед началом ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу только адреса без блокировки. Модули должны установить НЕБЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС, если СМ4★ & -СМ3★ & -СМ2★ & -СМ1★ & -СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать НЕБЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ ' ФАЗА_ДАННЫХ.

БЛОКИРОВАНИЕ_ЧТЕНИЕ READ_LOCKED

ЗАДАТЧИК должен установить БЛОКИРОВАНИЕ_ЧТЕНИЕ до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу чтения с блокировкой. Модули должны установить БЛОКИРОВАНИЕ_ЧТЕНИЕ, если -СМ4★& -СМ3★& -СМ2★& -СМ1★& -СМ0★& УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать БЛОКИРОВАНИЕ_ЧТЕНИЕ, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

БЛОКИРОВАНИЕ_ЗАПИСЬ WRITE_LOCKED

ЗАДАТЧИК должен установить БЛОКИРОВАНИЕ_ЗАПИСЬ до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу записи с блокировкой. Модули должны установить БЛОКИРОВАНИЕ_ЗАПИСЬ, если СМ4★& -СМ3★ & -СМ2★ & -СМ1★& -СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать БЛОКИРОВАНИЕ_ЗАПИСЬ, пока ИНИЦ_ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ | ФАЗА_РАССОЕДИНЕНИЯ & НЕБЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС.

БЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС ADDRESS_ONLY_LOCKED

ЗАДАТЧИК должен установить БЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу только адреса с блокировкой. Модули должны установить БЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС, если СМ4★& -СМ3★ & -СМ2★ & -СМ1★& -СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать БЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ | ФАЗА_ДАННЫХ.

ЧАСТНОЕ_ЧТЕНИЕ READ_PARTIAL

ЗАДАТЧИК должен установить ЧАСТНОЕ_ЧТЕНИЕ до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить частное чтение. Модули должны установить ЧАСТНОЕ_ЧТЕНИЕ, если -СМ4★ & -СМ3★ & -СМ2★ & -СМ1★ & -СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать ЧАСТНОЕ_ЧТЕНИЕ, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ЧАСТНАЯ_ЗАПИСЬ WRITE_PARTIAL

ЗАДАТЧИК должен установить ЧАСТНУЮ_ЗАПИСЬ до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу частной записи. Модули должны установить ЧАСТНУЮ_ЗАПИСЬ, если СМ4★ & -СМ3★ & -СМ2★ & СМ1★ & -СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать ЧАСТНУЮ_ЗАПИСЬ, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

БЛОКИРОВАНИЕ_ЧАСТНОЕ_ЧТЕНИЕ READ_PARTIAL_LOCKED

ЗАДАТЧИК должен установить БЛОКИРОВАНИЕ_ЧАСТНОЕ_ЧТЕНИЕ до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу запрета частного чтения. Модули должны установить БЛОКИРОВАНИЕ_ЧАСТНОЕ_ЧТЕНИЕ, если -СМ4★ & -СМ3★ & -СМ2★ & СМ1★ & СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать БЛОКИРОВАНИЕ_ЧАСТНОЕ_ЧТЕНИЕ, пока ИНИТ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

БЛОКИРОВАНИЕ_ЧАСТНАЯ_ЗАПИСЬ WRITE_PARTIAL_LOCKED

ЗАДАТЧИК должен установить БЛОКИРОВАНИЕ_ЧАСТНАЯ_ЗАПИСЬ до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу частной записи с блокировкой. Модули должны установить БЛОКИРОВАНИЕ_ЧАСТНАЯ_ЗАПИСЬ, если СМ4★ & -СМ3★ & -СМ2★ & СМ1★ & СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать БЛОКИРОВАНИЕ_ЧАСТНАЯ_ЗАПИСЬ, пока ИНИТ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ОТВЕТ_ЗАПИСИ WRITE_RESPONSE

ЗАДАТЧИК должен установить ОТВЕТ_ЗАПИСИ до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу ответа записи. Модули должны установить ОТВЕТ_ЗАПИСИ, если СМ4★ & -СМ3★ & СМ2★ & -СМ1★ & -СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать ОТВЕТ_ЧТЕНИЯ, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ОТВЕТ_ЧТЕНИЯ READ_RESPONSE

ЗАДАТЧИК должен установить ОТВЕТ_ЧТЕНИЯ до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу ответа чтения. Модули должны установить ОТВЕТ_ЧТЕНИЯ, если СМ4★ & -СМ3★ & СМ2★ & -СМ1★ & СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать ОТВЕТ_ЧТЕНИЯ, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ WRITE_NO_ACKNOWLEDGE

ЗАДАТЧИК должен установить ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу записи без подтверждения. Модули должны установить ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ, если СМ4★ & -СМ3★ & СМ2★ & СМ1★ & -СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

НЕДЕЙСТВИТЕЛЬНОЕ_ЧТЕНИЕ READ_INVALID

ЗАДАТЧИК должен установить НЕДЕЙСТВИТЕЛЬНОЕ_ЧТЕНИЕ до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу недействительного чтения. Модули должны установить НЕДЕЙСТВИТЕЛЬНОЕ_ЧТЕНИЕ, если -СМ4★ & СМ3★ & -СМ2★ & -СМ1★ & -СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать НЕДЕЙСТВИТЕЛЬНОЕ_ЧТЕНИЕ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ WRITE_INVALID

ЗАДАТЧИК должен установить НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу недействительной записи. Модули должны установить НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ, если СМ4★ & СМ3★ & -СМ2★ & -СМ1★ & -СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

РАЗДЕЛЕННОЕ_ЧТЕНИЕ READ_SHARED

ЗАДАТЧИК должен установить РАЗДЕЛЕННОЕ_ЧТЕНИЕ до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу разделенного чтения. Модули должны установить РАЗДЕЛЕННОЕ_ЧТЕНИЕ, если -СМ4★ & СМ3★ & -СМ2★ & -СМ1★ & СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать РАЗДЕЛЕННОЕ_ЧТЕНИЕ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ

КОПИРОВАНИЕ_НАЗАД COPY_BACK

ЗАДАТЧИК должен установить КОПИРОВАНИЕ_НАЗАД до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу копирования назад. Модули должны установить КОПИРОВАНИЕ_НАЗАД, если СМ4★ & СМ3★ & -СМ2★ & -СМ1★ & СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать КОПИРОВАНИЕ_НАЗАД, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ READ_MODIFIED

ЗАДАТЧИК должен установить МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу модифицированного чтения. Модули должны установить МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ, если -СМ4★ & СМ3★ & -СМ2★ & СМ1★ & СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

НЕДЕЙСТВИТЕЛЬНОСТЬ INVALIDATE

ЗАДАТЧИК должен установить НЕДЕЙСТВИТЕЛЬНОСТЬ до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу недействительности. Модули должны установить НЕДЕЙСТВИТЕЛЬНОСТЬ, если СМ4★ & СМ3★ & -СМ2★ & СМ1★ & СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать НЕДЕЙСТВИТЕЛЬНОСТЬ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

РАЗДЕЛЕННЫЙ_ОТВЕТ SHARED_RESPONSE

ЗАДАТЧИК должен установить РАЗДЕЛЕННЫЙ_ОТВЕТ до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу разделенного ответа. Модули должны установить РАЗДЕЛЕННЫЙ_ОТВЕТ, если СМ4★ & СМ3★ & СМ2★ & -СМ1★ & -СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ. Модули должны удерживать РАЗДЕЛЕННЫЙ_ОТВЕТ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ

МОДИФИЦИРОВАННЫЙ_ОТВЕТ MODIFIED_RESPONSE

ЗАДАТЧИК должен установить МОДИФИЦИРОВАННЫЙ_ОТВЕТ до начала ФАЗЫ_СОЕДИНЕНИЯ, чтобы совершить передачу модифицированный ответ. Модули должны установить МОДИФИЦИРОВАННЫЙ_ОТВЕТ, если СМ4★ & СМ3★ & СМ2★ & СМ1★ & СМ0★ & УЧАСТИЕ & ФАЗА_СОЕДИНЕНИЯ Модули должны удерживать МОДИФИЦИРОВАННЫЙ_ОТВЕТ пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

КЕШ_СМД CACHE_CMD

Модули должны установить КЕШ_СМД, если НЕДЕЙСТВИТЕЛЬНОСТЬ_ЧТЕНИЯ НЕДЕЙСТВИТЕЛЬНОСТЬ_ЗАПИСИ | РАЗДЕЛЕННОЕ_ЧТЕНИЕ КОПИРОВАНИЕ_НАЗАД | МОДИФИКАЦИЯ_ЧТЕНИЯ | НЕДЕЙСТВИТЕЛЬНОСТЬ | РАЗДЕЛЕННЫЙ_ОТВЕТ | МОДИФИЦИРОВАННЫЙ ОТВЕТ.

СМД_ЗАПИСЬ WRITE_CMD

ЗАДАТЧИК должен установить СМД_ЗАПИСЬ, если НЕБЛОКИРОВАНИЕ_ЗАПИСЬ | НЕБЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС | БЛОКИРОВАНИЕ_ЗАПИСЬ | БЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС | ЧАСТНАЯ_ЗАПИСЬ БЛОКИРОВАНИЕ_ЧАСТНАЯ_ЗАПИСЬ | ОТВЕТ_ЗАПИСИ | ОТВЕТ_ЧТЕНИЯ | ЗАПИСЬ_БЕЗ ПОДТВЕРЖДЕНИЯ | НЕДЕЙСТВИТЕЛЬНОСТЬ_ЗАПИСИ | КОПИРОВАНИЕ_НАЗАД | НЕДЕЙСТВИТЕЛЬНОСТЬ | РАЗДЕЛЕННЫЙ_ОТВЕТ | МОДИФИЦИРОВАННЫЙ_ОТВЕТ.

ПЕРЕДАЧА СМД 3 TRANS_CMD_3

ЗАДАТЧИК должен установить ПЕРЕДАЧУ_СМД_3, если НЕДЕЙСТВИТЕЛЬНОСТЬ_ЧТЕНИЯ НЕДЕЙСТВИТЕЛЬНОСТЬ_ЗАПИСИ | РАЗДЕЛЕННОЕ_ЧТЕНИЕ | КОПИРОВАНИЕ_НАЗАД | МОДИФИКАЦИЯ_ЧТЕНИЯ | НЕДЕЙСТВИТЕЛЬНОСТЬ | РАЗДЕЛЕННЫЙ_ОТВЕТ | МОДИФИКАЦИЯ_ОТВЕТА.

ПЕРЕДАЧА_СМД_2 TRANS_CMD_2

ЗАДАТЧИК должен установить ПЕРЕДАЧУ_СМД_2, если ОТВЕТ_ЗАПИСИ | ОТВЕТ_ЧТЕНИЯ | ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ | РАЗДЕЛЕННЫЙ_ОТВЕТ | МОДИФИКАЦИЯ_ОТВЕТА

ПЕРЕДАЧА_СМД_1 TRANS_CMD_1

ЗАДАТЧИК должен установить ПЕРЕДАЧУ_СМД_1, если ЧАСТНОЕ_ЧТЕНИЕ | ЧАСТНАЯ_ЗАПИСЬ | БЛОКИРОВАНИЕ_ЧАСТНОЕ_ЧТЕНИЕ | БЛОКИРОВАНИЕ_ЧАСТНАЯ_ЗАПИСЬ | ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ | МОДИФИКАЦИЯ_ЧТЕНИЯ | НЕДЕЙСТВИТЕЛЬНОСТЬ | МОДИФИКАЦИЯ_ОТВЕТА.

ПЕРЕДАЧА_СМД_0 TRANS_CMD_0

ЗАДАТЧИК должен установить ПЕРЕДАЧУ_СМД_0, если БЛОКИРОВАНИЕ_ЧТЕНИЕ | БЛОКИРОВАНИЕ_ЗАПИСЬ | БЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС | БЛОКИРОВАНИЕ_ЧАСТНОЕ_ЧТЕНИЕ | БЛОКИРОВАНИЕ_ЧАСТНАЯ_ЗАПИСЬ | ОТВЕТ_ЧТЕНИЯ | РАЗДЕЛЕННОЕ_ЧТЕНИЕ | КОПИРОВАНИЕ_НАЗАД | МОДИФИКАЦИЯ_ЧТЕНИЯ | НЕДЕЙСТВИТЕЛЬНОСТЬ | МОДИФИЦИРОВАННЫЙ_ОТВЕТ.

ЧАСТНЫЙ PARTIAL

Модули должны установить ЧАСТНЫЙ, если ЧАСТНОЕ_ЧТЕНИЕ | ЧАСТНАЯ_ЗАПИСЬ | БЛОКИРОВАНИЕ_ЧАСТНОЕ_ЧТЕНИЕ | БЛОКИРОВАНИЕ_ЧАСТНАЯ_ЗАПИСЬ.

ТОЛЬКО_АДРЕС ADDRESS_ONLY

Модули должны установить ТОЛЬКО_АДРЕС, если -ИНИЦ & ФАЗА_СОЕДИНЕНИЯ, и должны удерживать его, пока ИНИЦ | ФАЗА_ДАННЫХ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ЗАПИСЬ_ИСПОЛНИТЕЛЮ SLAVE_WRITE

Модули должны установить ЗАПИСЬ_ИСПОЛНИТЕЛЮ, если ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН & СМ4★, и должны удерживать его, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ | РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ & -СМ4★ & ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН.

СТАТУС_ЗАПИСИ WRITE_STATUS

Модули должны установить СТАТУС_ЗАПИСИ, если -ИНИЦ & ФАЗА_СОЕДИНЕНИЯ & СМ4★, и должны удерживать его, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

БОЛЬШЕ MORE

ЗАДАТЧИК может установить БОЛЬШЕ в начале копирования блока. ЗАДАТЧИК должен сбросить БОЛЬШЕ в последней блочной передаче или до нее.

ФЛАГ_ПОСЛЕДНЕЙ_ПЕРЕДАЧИ LAST_TRANSACTION_FLAG

Модули должны установить ФЛАГ_ПОСЛЕДНЕЙ_ПЕРЕДАЧИ, если ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН & РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ & СМ1★, и должны удерживать его, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ | ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН & -СМ1★.

6.2.1.5 **Определения операций блокирования (доступа)**

БЛОКИРОВАННЫЙ LOCKED

Модули должны установить БЛОКИРОВАННЫЙ, если БЛОКИРОВАНИЕ_ЧТЕНИЕ ¦ БЛОКИРОВАНИЕ_ЗАПИСЬ ¦ БЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС ¦ БЛОКИРОВАНИЕ_ЧАСТНОЕ_ЧТЕНИЕ ¦ БЛОКИРОВАНИЕ_ЧАСТНАЯ_ЗАПИСЬ.

БЛОКИРОВАНИЕ РЕСУРСОВ RESOURCE_LOCKED

Модуль должен удерживать БЛОКИРОВАНИЕ_РЕСУРСОВ для каждого отдельного блокируемого ресурса, связанного с отдельными адресами, для которого модуль установит ВЫБРАННЫЙ. Модули должны установить БЛОКИРОВАНИЕ_РЕСУРСОВ, если БЛОКИРОВАННЫЙ & ВЫБРАННЫЙ, и должны удерживать БЛОКИРОВАНИЕ_РЕСУРСОВ, пока ИНИЦ ¦ -БЛОКИРОВАННЫЙ & УДЕРЖАНИЕ_БЛОКИРОВАНИЯ ¦ -ET★ & -AS★.

УДЕРЖАНИЕ_БЛОКИРОВАНИЯ LOCK_HOLD

Задатчик должен установить УДЕРЖАНИЕ_БЛОКИРОВАНИЯ, если БЛОКИРОВАНИЕ_РЕСУРСОВ появился до начала ФАЗЫ_РАССОЕДИНЕНИЯ, чтобы побудить модули, выставившие БЛОКИРОВАНИЕ_РЕСУРСОВ, удерживать его после окончания текущей передачи. Модули должны установить УДЕРЖАНИЕ_БЛОКИРОВАНИЯ, если ВЫБРАННЫЙ & CM0★ и они обнаружили снятие AS★. Модули должны удерживать УДЕРЖАНИЕ_БЛОКИРОВАНИЯ, пока ИНИЦ ¦ -ET★ & -AS★ ¦ -БЛОКИРОВАННЫЙ & AS★ & -ai★.

МАСКИРОВАННЫЙ_ОБМЕН MASK_SWAP

Чтобы выбрать блокирующую команду МАСКИРОВАННЫЙ_ОБМЕН, ЗАДАТЧИК должен установить МАСКИРОВАННЫЙ_ОБМЕН, если БЛОКИРОВКА_ЗАПИСЬ ¦ БЛОКИРОВКА_ЧАСТНАЯ_ЗАПИСЬ до начала ФАЗЫ_ДАННЫХ. Модули должны установить МАСКИРОВАННЫЙ_ОБМЕН, если -CM2★ & -CM1★ & CM0★ & УЧАСТИЕ & ФАЗА_ДАННЫХ & БЛОКИРОВАНИЕ_РЕСУРСОВ. Модули должны удерживать МАСКИРОВАННЫЙ_ОБМЕН, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

СРАВНЕНИЕ_ОБМЕН COMPARE_SWAP

Чтобы выбрать блокирующую команду СРАВНЕНИЕ_ОБМЕН, ЗАДАТЧИК должен установить СРАВНЕНИЕ_ОБМЕН, если БЛОКИРОВАНИЕ_ЗАПИСЬ ¦ БЛОКИРОВАНИЕ_ЧАСТНАЯ_ЗАПИСЬ до начала ФАЗЫ_ДАННЫХ. Модули должны установить СРАВНЕНИЕ_ОБМЕН, если -CM2★ & CM1★ & -CM0★ & УЧАСТИЕ & ФАЗА_ДАННЫХ & ЗАПРЕТ_РЕСУРСОВ. Модули должны удерживать МАСКИРОВАННЫЙ_ОБМЕН, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ВЫБОРКА_СЛОЖЕНИЕ_СО_СТАРШЕГО FETCH_ADD_BIG

Чтобы выбрать блокирующую команду ВЫБОРКА_СЛОЖЕНИЕ_СО_СТАРШЕГО, ЗАДАТЧИК должен установить ВЫБОРКА_СЛОЖЕНИЕ_СО_СТАРШЕГО, если БЛОКИРОВАНИЕ ЗАПИСЬ ¦ БЛОКИРОВАНИЕ_ЧАСТНАЯ_ЗАПИСЬ до начала ФАЗЫ_ДАННЫХ. Модули должны установить ВЫБОРКА_СЛОЖЕНИЕ_СО_СТАРШЕГО, если -CM2★ & CM1★ & CM0★ & УЧАСТИЕ & ФАЗА_ДАННЫХ & БЛОКИРОВАНИЕ_РЕСУРСОВ. Модули должны удерживать ВЫБОРКА_СЛОЖЕНИЕ_СО_СТАРШЕГО, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ВЫБОРКА_СЛОЖЕНИЕ_С_МЛАДШЕГО FETCH_ADD_LITTLE

Чтобы выбрать блокирующую команду ВЫБОРКА_СЛОЖЕНИЕ_С_МЛАДШЕГО, ЗАДАТЧИК должен установить ВЫБОРКА_СЛОЖЕНИЕ_С_МЛАДШЕГО, если БЛОКИРОВАНИЕ_ЗАПИСЬ ¦ БЛОКИРОВАНИЕ_ЧАСТНОЙ_ЗАПИСИ до начала ФАЗЫ_ДАННЫХ. Модули должны установить ВЫБОРКА_СЛОЖЕНИЕ_С_МЛАДШЕГО, если CM2★ & -CM1★ & -CM0★ & УЧАСТИЕ & ФАЗА_ДАННЫХ & БЛОКИРОВАНИЕ_РЕСУРСОВ. Модули должны удерживать ВЫБОРКА_СЛОЖЕНИЕ_С_МЛАДШЕГО, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

БЛОКИРОВАНИЕ_СМД_0 LOCK_CMD_0

ЗАДАТЧИК должен установить БЛОКИРОВАНИЕ_СМД_0, если МАСКИРОВАННЫЙ_ОБМЕН ¦ ВЫБОРКА_СЛОЖЕНИЕ_СО_СТАРШЕГО.

БЛОКИРОВАНИЕ_СМД_1 LOCK_CMD_1

ЗАДАТЧИК должен установить БЛОКИРОВАНИЕ_СМД_1, если СРАВНИТЕЛЬНЫЙ_ОБМЕН ¦ ВЫБОРКА_СЛОЖЕНИЕ_СО_СТАРШЕГО.

БЛОКИРОВАНИЕ_СМД_2 LOCK_CMD_2

ЗАДАТЧИК должен установить БЛОКИРОВАНИЕ_СМД_2, если ВЫБОРКА_СЛОЖЕНИЕ_С_МЛАДШЕГО.

6.2.1.6 **Длина передачи данных**

НЕОГРАНИЧЕННАЯ_ПЕРЕДАЧА TRANSFER_UNRESTRICTED

Чтобы определить неограниченную длину передачи, ЗАДАТЧИК должен установить НЕОГРАНИЧЕННАЯ_ПЕРЕДАЧА до ФАЗА_ДАННЫХ | ФАЗА_РАССОЕДИНЕНИЯ. ЗАДАТЧИК должен установить НЕОГРАНИЧЕННАЯ_ПЕРЕДАЧА, если БЛОКИРОВАНИЕ_СМД_0 . БЛОКИРОВАНИЕ_СМД_1 | БЛОКИРОВАНИЕ_СМД_2. Задатчик должен установить только НЕОГРАНИЧЕННАЯ_ПЕРЕДАЧА, если СТАТУС_ПРИНУЖДЕНИЯ | —(РАСЩЕПЛЕННЫЙ_СТАТУС & -ЗАПИСЬ_СМД). Задатчик не должен устанавливать НЕОГРАНИЧЕННАЯ_ПЕРЕДАЧА, а должен установить только одну из ПЕРЕДАЧА_1 | ПЕРЕДАЧА_2 | ПЕРЕДАЧА_4 | ПЕРЕДАЧА_8 | ПЕРЕДАЧА_16 | ПЕРЕДАЧА_32 | ПЕРЕДАЧА_64, если —СТАТУС_ПРИНУЖДЕНИЯ | РАСЩЕПЛЕННЫЙ_СТАТУС & -ЗАПИСЬ_СМД. Модули должны устанавливать НЕОГРАНИЧЕННАЯ_ПЕРЕДАЧА, если -ИНИЦ & УЧАСТИЕ & ФАЗА_ДАННЫХ & -СМ7★ & - СМ6★ & -СМ5★. Модули должны удерживать НЕОГРАНИЧЕННУЮ_ПЕРЕДАЧУ, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ПЕРЕДАЧА_1 TRANSFER_1

Чтобы определить длину передачи один, ЗАДАТЧИК должен установить ПЕРЕДАЧА_1 до ФАЗА_ДАННЫХ | ФАЗА_РАССОЕДИНЕНИЯ. Модули должны установить ПЕРЕДАЧА_1, если -ИНИЦ & УЧАСТИЕ & ФАЗА_ДАННЫХ & -СМ7★ & -СМ6★ & СМ5★. Модули должны удерживать ПЕРЕДАЧА_1, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ПЕРЕДАЧА_2 TRANSFER_2

Чтобы определить длину передачи два, ЗАДАТЧИК должен установить ПЕРЕДАЧА_2 до ФАЗА_ДАННЫХ | ФАЗА_РАССОЕДИНЕНИЯ. Модули должны установить ПЕРЕДАЧА_2, если -ИНИЦ & УЧАСТИЕ & ФАЗА_ДАННЫХ & -СМ7★ & СМ6★ & -СМ5★. Модули должны удерживать ПЕРЕДАЧА_2, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ПЕРЕДАЧА_4 TRANSFER_4

Чтобы определить длину передачи четыре, ЗАДАТЧИК должен установить ПЕРЕДАЧА_4 до ФАЗА_ДАННЫХ | ФАЗА_РАССОЕДИНЕНИЯ. Модули должны установить ПЕРЕДАЧА_4, если -ИНИЦ & УЧАСТИЕ & ФАЗА_ДАННЫХ & -СМ7★ & СМ6★ & СМ5★. Модули должны удерживать ПЕРЕДАЧА_4, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ПЕРЕДАЧА_8 TRANSFER_8

Чтобы определить длину передачи восемь, ЗАДАТЧИК должен установить ПЕРЕДАЧА_8 до ФАЗА_ДАННЫХ | ФАЗА_РАССОЕДИНЕНИЯ Модули должны установить ПЕРЕДАЧА_8, если -ИНИЦ & УЧАСТИЕ & ФАЗА_ДАННЫХ & СМ7★ & -СМ6★ & -СМ5★. Модули должны удерживать ПЕРЕДАЧА_8, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ПЕРЕДАЧА_16 TRANSFER_16

Чтобы определить длину передачи 16, ЗАДАТЧИК должен установить ПЕРЕДАЧА_16 до ФАЗА_ДАННЫХ | ФАЗА_РАССОЕДИНЕНИЯ. Модули должны установить ПЕРЕДАЧА_16, если -ИНИЦ & УЧАСТИЕ & ФАЗА_ДАННЫХ & СМ7★ & -СМ6★ & СМ5★. Модули должны удерживать ПЕРЕДАЧА_16, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ПЕРЕДАЧА_32 TRANSFER_32

Чтобы определить длину передачи 32, ЗАДАТЧИК должен установить ПЕРЕДАЧА_32 до ФАЗА_ДАННЫХ | ФАЗА_РАССОЕДИНЕНИЯ. Модули должны установить ПЕРЕДАЧА_32, если -ИНИЦ & УЧАСТИЕ & ФАЗА_ДАННЫХ & СМ7★ & СМ6★ & -СМ5★. Модули должны удерживать ПЕРЕДАЧА_32, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ПЕРЕДАЧА_64 TRANSFER_64

Чтобы определить длину передачи 64, ЗАДАТЧИК должен установить ПЕРЕДАЧА_64 до ФАЗА_ДАННЫХ | ФАЗА_РАССОЕДИНЕНИЯ. Модули должны установить ПЕРЕДАЧА_64, если -ИНИЦ & УЧАСТИЕ & ФАЗА_ДАННЫХ & СМ7★ & СМ6★ & СМ5★. Модули должны удерживать ПЕРЕДАЧА_64, пока ИНИЦ | ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ДЛИНА_ДАННЫХ_2 DATA_LENGTH_2

ЗАДАТЧИК должен установить ДЛИНА_ДАННЫХ_2, если ПЕРЕДАЧА_8 | ПЕРЕДАЧА_16 | ПЕРЕДАЧА_32 | ПЕРЕДАЧА_64.

ДЛИНА_ДАННЫХ_1 DATA_LENGTH_1

ЗАДАТЧИК должен установить ДЛИНА_ДАННЫХ_1, если ПЕРЕДАЧА_2 | ПЕРЕДАЧА_4 | ПЕРЕДАЧА_32 | ПЕРЕДАЧА_64.

ДЛИНА_ДАННЫХ_0 DATA_LENGTH_0

ЗАДАТЧИК должен установить ДЛИНА_ДАННЫХ_0, если ПЕРЕДАЧА_1 ¦ ПЕРЕДАЧА_4 ¦ ПЕРЕДАЧА_16 ¦ ПЕРЕДАЧА_64.

6.2.1.7 **Статус передачи**

6.2.1.7.1 **Статус передачи задатчика**

СТАТУС_ЗАНЯТОСТИ BUSY_STATUS

ЗАДАТЧИК должен установить СТАТУС_ЗАНЯТОСТИ, если ST1★ и он определил, что AIf перешел в логический нуль. ЗАДАТЧИК должен удерживать СТАТУС_ЗАНЯТОСТИ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ВЫБРАННЫЙ_СТАТУС SELECTED_STATUS

ЗАДАТЧИК должен установить ВЫБРАННЫЙ_СТАТУС, если ST2★ и он определил, что AIf перешел в логический нуль. ЗАДАТЧИК должен удерживать ВЫБРАННЫЙ_СТАТУС, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ ¦ РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ & -ST2★ и пока он не определил, что DKf или DIf перешли в логический нуль.

СТАТУС_ФЛАГА_ПЕРЕДАЧИ TRANSACTION_FLAG_STATUS

ЗАДАТЧИК должен установить СТАТУС_ФЛАГА_ПЕРЕДАЧИ, если ST4★ & (ПРИЗНАК_СОЕДИНЕНИЯ_ЗАФИКСИРОВАН ¦ ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН & РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ). ЗАДАТЧИК должен удерживать СТАТУС_ФЛАГА_ПЕРЕДАЧИ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ ¦ -ST4★ & РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ & ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН.

СТАТУС_ВНЕДРЕННОСТИ INTERVENTION_STATUS

ЗАДАТЧИК должен установить СТАТУС_ВНЕДРЕННОСТИ, если ST5★ & (ПРИЗНАК_СОЕДИНЕНИЯ_ЗАФИКСИРОВАН ¦ ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН & РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ). ЗАДАТЧИК должен удерживать СТАТУС_ВНЕДРЕННОСТИ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ ¦ -ST5★ & РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ & ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН.

СТАТУС_ОШИБКИ_ОБМЕНА BEAT_ERROR_STATUS

ЗАДАТЧИК должен установить СТАТУС_ОШИБКИ_ОБМЕНА, если ST6★ & (ПРИЗНАК_СОЕДИНЕНИЯ_ЗАФИКСИРОВАН ¦ ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН), и должен удерживать его, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

СТАТУС_ОЖИДАНИЯ WAIT_STATUS

ЗАДАТЧИК должен установить СТАТУС_ОЖИДАНИЯ, если ST7★ & (ПРИЗНАК_СОЕДИНЕНИЯ_ЗАФИКСИРОВАН ¦ ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН & РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ), и должен удерживать его, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

СТАТУС_КОНЦА_ДАННЫХ END_OF_DATA_STATUS

ЗАДАТЧИК должен установить СТАТУС_КОНЦА_ДАННЫХ, если ST1★ & ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН. ЗАДАТЧИК должен удерживать СТАТУС_КОНЦА_ДАННЫХ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

Во время принудительных передач модули должны игнорировать данные обменов, в которых выставлен КОНЕЦ_ДАННЫХ.

ОШИБКА_СПОСОБНОСТИ_ЗАДАТЧИКА MASTER_CAPABILITY_ERROR

Задатчик должен установить ОШИБКУ_СПОСОБНОСТИ_ЗАДАТЧИКА, если один или более исполнителей сигнализирует задатчику об использовании способности, которую задатчик не поддерживает. Задатчик должен удерживать ОШИБКУ_СПОСОБНОСТИ_ЗАДАТЧИКА, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

6.2.1.7.2 **Статус передачи модуля**

ОШИБКА_ЧЕТНОСТИ_АДРЕСА ADDRESS_PARITY_ERROR

Модули должны установить ОШИБКУ_ЧЕТНОСТИ_АДРЕСА, если во время фазы соединения на линиях AD[]★ обнаружена ошибка четности. Модули должны удерживать ОШИБКУ_ЧЕТНОСТИ_АДРЕСА, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ШИРОКОВЕЩАТЕЛЬНЫЙ_СТАТУС BROADCAST_STATUS

Модули должны установить ШИРОКОВЕЩАТЕЛЬНЫЙ_СТАТУС, если ЗАДАТЧИК & (ST3★ ¦ -CA1★ & РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ), и они обнаружили переход AIf★ в логический нуль или УЧАСТИЕ & ST3★ и обнаружили первое выставление DS★. Модули должны удерживать ШИРОКОВЕЩАТЕЛЬНЫЙ_СТАТУС, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ОШИБКА_ЧЕТНОСТИ_ДАННЫХ DATA_PARITY_ERROR

Модули должны установить ОШИБКУ_ЧЕТНОСТИ_ДАННЫХ, если во время фазы данных на линиях AD[]★ или D[]★ обнаружена ошибка четности. ОШИБКА_ЧЕТНОСТИ_ДАННЫХ также должна быть установлена, если во время пакетного режима обнаружена продольная ошибка четности. Модули должны удерживать ОШИБКУ_ЧЕТНОСТИ_ДАННЫХ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ОШИБКА_ЧЕТНОСТИ_ДАННЫХ_РАССОЕДИНЕНИЯ DISCONNECTION_DATA_PARITY_ERROR

Модули должны установить ОШИБКУ_ЧЕТНОСТИ_ДАННЫХ_РАССОЕДИНЕНИЯ, если во время фазы рассоединения на линиях AD[]31 . . . 0]★ обнаружена ошибка четности, когда ЗАПИСЬ_СМД. Модули должны удерживать ОШИБКУ_ЧЕТНОСТИ_ДАННЫХ_РАССОЕДИНЕНИЯ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ОШИБКА_ЧЕТНОСТИ_КОМАНДЫ_СОЕДИНЕНИЯ CONNECTION_COMMAND_PARITY_ERROR

Модули должны установить ОШИБКУ_ЧЕТНОСТИ_КОМАНДЫ_СОЕДИНЕНИЯ, если во время фазы соединения на линиях CM[]★ обнаружена ошибка четности. Модули должны удерживать ОШИБКУ_ЧЕТНОСТИ_КОМАНДЫ_СОЕДИНЕНИЯ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ОШИБКА_ЧЕТНОСТИ_КОМАНДЫ_ДАННЫХ DATA_COMMAND_PARITY_ERROR

Модули должны установить ОШИБКУ_ЧЕТНОСТИ_КОМАНДЫ_ДАННЫХ, если во время фазы данных на линиях CM[]★ обнаружена ошибка четности. Модули должны удерживать ОШИБКУ_ЧЕТНОСТИ_КОМАНДЫ_ДАННЫХ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ОШИБКА_ЧЕТНОСТИ_КОМАНДЫ_РАССОЕДИНЕНИЯ DISCONNECTION_COMMAND_PARITY_ERROR

Модули должны установить ОШИБКУ_ЧЕТНОСТИ_КОМАНДЫ_РАССОЕДИНЕНИЯ, если во время фазы рассоединения на линиях CM[]★ обнаружена ошибка четности. Модули должны удерживать ОШИБКУ_ЧЕТНОСТИ_КОМАНДЫ_РАССОЕДИНЕНИЯ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ОШИБКА_СПОСОБНОСТИ CAPABILITY_ERROR

Участвующий модуль должен установить ОШИБКУ_СПОСОБНОСТИ, если во время фазы соединения задатчик потребовал от модуля операцию, которую тот не может выполнить. Модули должны удерживать ОШИБКУ_СПОСОБНОСТИ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ОШИБКА_НЕЛЕГАЛЬНОЙ_ОПЕРАЦИИ ILLEGAL_OPERATION_ERROR

Участвующий модуль должен установить ОШИБКУ_НЕЛЕГАЛЬНОЙ_ОПЕРАЦИИ, если он обнаружил команду, статус и/или способность, приводящие к неправильной операции. Модуль должен удерживать ОШИБКУ_НЕЛЕГАЛЬНОЙ_ОПЕРАЦИИ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ОШИБКА_ЧЕТНОСТИ_ТЕГА_АДРЕСА ADDRESS_TAG_PARITY_ERROR

Модули должны установить ОШИБКУ_ЧЕТНОСТИ_ТЕГА_АДРЕСА, если во время фазы соединения на линиях TG[]★, являющихся активными, обнаружена ошибка четности. Модули должны удерживать ОШИБКУ_ЧЕТНОСТИ_ТЕГА_АДРЕСА, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ОШИБКА_ЧЕТНОСТИ_ТЕГА_ДАННЫХ DATA_TAG_PARITY_ERROR

Модули должны установить ОШИБКУ_ЧЕТНОСТИ_ТЕГА_ДАННЫХ, если во время фазы данных на линиях TG[]★, являющихся активными, обнаружена ошибка четности. Модули должны удерживать ОШИБКУ ЧЕТНОСТИ_ТЕГА_ДАННЫХ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ОШИБКА_ЧЕТНОСТИ_ТЕГА_РАССОЕДИНЕНИЯ DISCONNECTION_TAG_PARITY_ERROR

Модули должны установить ОШИБКУ_ЧЕТНОСТИ_ТЕГА_РАССОЕДИНЕНИЯ, если во время фазы рассоединения на линиях TG[]★, являющихся активными, обнаружена ошибка четности. Модули должны удерживать ОШИБКУ_ЧЕТНОСТИ_ТЕГА_РАССОЕДИНЕНИЯ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ОШИБКА_ОБМЕНА BEAT_ERROR

Модули должны установить ОШИБКУ_ОБМЕНА, если РАЗРЕШЕНИЕ_СООБЩЕНИЯ_ЧЕТНОСТИ & (ОШИБКА_ЧЕТНОСТИ_АДРЕСА ¦ ОШИБКА_ЧЕТНОСТИ_КОМАНДЫ_СОЕДИНЕНИЯ ¦ ОШИБКА_ЧЕТНОСТИ_ТЕГА_АДРЕСА) ¦ ОШИБКА_СПОСОБНОСТИ ¦ ОШИБКА_НЕЛЕГАЛЬНОЙ_ОПЕРАЦИИ. Модули должны использовать ОШИБКУ_ОБМЕНА только для точного сообщения об ошибке. Модули могут установить ОШИБКУ_ОБМЕНА, если РАЗРЕШЕНИЕ_СООБЩЕНИЯ_ЧЕТНОСТИ & (ОШИБКА_ЧЕТНОСТИ_ДАННЫХ ¦ ОШИБКА_ЧЕТНОСТИ_КОМАНДЫ_ДАННЫХ ¦ ОШИБКА_ЧЕТНОСТИ_ТЕГА_ДАННЫХ). Модули должны удерживать ОШИБКУ_ОБМЕНА, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ОШИБКА_ПЕРЕДАЧИ TRANSACTION_ERROR

Модули должны установить ОШИБКУ_ПЕРЕДАЧИ, если ОШИБКА_ОБМЕНА ¦ РАЗРЕШЕНИЕ_СООБЩЕНИЯ_ЧЕТНОСТИ & (ОШИБКА_ЧЕТНОСТИ_ДАННЫХ ¦ ОШИБКА_ЧЕТНОСТИ_ДАННЫХ_РАССОЕДИНЕНИЯ ¦ ОШИБКА_ЧЕТНОСТИ_КОМАНДЫ_ДАННЫХ ¦ ¦ОШИБКА_ЧЕТНОСТИ_КОМАНДЫ_РАССОЕДИНЕНИЯ ¦ ОШИБКА_ЧЕТНОСТИ_ТЕГА_ДАННЫХ ¦ ОШИБКА_ЧЕТНОСТИ_ТЕГА_РАССОЕДИНЕНИЯ).

СТАТУС_ОШИБКИ_ПЕРЕДАЧИ TRANSACTION_ERROR_STATUS

Модули должны установить СТАТУС_ОШИБКИ_ПЕРЕДАЧИ, если ST0★ & КОНЕЦ_РАССОЕДИНЕНИЯ. Модули должны сбросить СТАТУС_ОШИБКИ_ПЕРЕДАЧИ, если ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ЗАНЯТОСТЬ BUSY

Модуль должен установить ЗАНЯТОСТЬ, если ДЕКОДИРОВАННЫЙ_АДРЕС & ФАЗА_СОЕДИНЕНИЯ & УЧАСТИЕ & (СООБЩЕНИЕ_ЗАНЯТОСТИ или модуль не может осуществить передачу сейчас и хочет, чтобы задатчик повторил запрос позднее). Модули должны удерживать ЗАНЯТОСТЬ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ВЫБРАННЫЙ SELECTED

Модули должны установить ВЫБРАННЫЙ, если -(РАЗДЕЛЕННЫЙ_ОТВЕТ ¦ МОДИФИЦИРОВАННЫЙ_ОТВЕТ) & ДЕКОДИРОВАННЫЙ_АДРЕС и адрес, соответствующий следующему обмену данных, является одним из тех, для которых модуль оказывается последним источником и/или приемником данных или агентом, действующим от лица такого модуля, или если ЗАПРОСЧИК & (РАЗДЕЛЕННЫЙ_ОТВЕТ ¦ МОДИФИЦИРОВАННЫЙ_ОТВЕТ) & ДЕКОДИРОВАННЫЙ_АДРЕС и адрес, соответствующий следующему обмену данных, достигают адреса кешевой строки, связанного на этой основе с ЗАПРОСЧИКОМ, или если ВЫБРАННЫЙ_ПОЧТОВЫЙ_ЯЩИК.

Модули должны удерживать ВЫБРАННЫЙ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ ¦ ДЕКОДИРОВАННЫЙ_АДРЕС и адрес, соответствующий следующему обмену данных, являются одним из тех, для которых условия, изложенные в предыдущем параграфе, не выполнены.

ШИРОКОВЕЩАТЕЛЬНЫЙ BROADCAST

Модули могут установить ШИРОКОВЕЩАТЕЛЬНЫЙ, если -ИНИЦ & ДЕКОДИРОВАННЫЙ_АДРЕС & ai★. Модули должны удерживать ШИРОКОВЕЩАТЕЛЬНЫЙ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

ШИРОКОЗАПРОСНЫЙ_АДРЕС BROADCALL_ADDRESS

Модули должны устанавливать ШИРОКОЗАПРОСНЫЙ_АДРЕС, если -ИНИТ & ДЕКОДИРОВАННЫЙ_АДРЕС & -СТАТУС_ЗАПИСИ & ai★ и адрес, соответствующий следующему обмену данных, являются одним из относящихся к другим исполнителям в широкозапросном режиме. Четность линий адрес/данные не должна проверяться во время фазы данных, если установлен ШИРОКОЗАПРОСНЫЙ_АДРЕС.

ОЖИДАНИЕ WAIT

Модуль должен установить ОЖИДАНИЕ, если -ИНИЦ & ДЕКОДИРОВАННЫЙ_АДРЕС & (ОЖИДАНИЕ_КЕША или модуль не может выполнить передачу до следующего раза), и должен произвести системно определенное действие, чтобы поставить в известность запрашивающий модуль, что есть готовность произвести передачу. Модули должны удерживать ОЖИДАНИЕ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

КОНЕЦ_ДАННЫХ END_OF_DATA

Модули должны установить КОНЕЦ_ДАННЫХ, если СТАТУС_ПРИНУЖДЕНИЯ & ФАЗА_ДАННЫХ & ДЕКОДИРОВАННЫЙ_АДРЕС и адрес, соответствующий следующему обмену данных, являются одним из тех, для которых модуль неспособен послать или принять данные & -(ЧАСТИЧНЫЙ и первый обмен ФАЗЫ_ДАННЫХ). Модули должны удерживать КОНЕЦ_ДАННЫХ, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

УЧАСТИЕ PARTICIPATING

Модули должны установить УЧАСТИЕ, если ВЫБРАННЫЙ ¦ ВНЕДРЕННОСТЬ ¦ ШИРОКОВЕЩАНИЕ ¦ РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ & -СТАТУС_ПРИНУЖДЕНИЯ.

ПАКЕТ_NAK PACKET_NAK

Модули должны установить ПАКЕТ_NAK (без подтверждения), если РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ & ЗАПРОС_ПАКЕТА и модуль хочет предотвратить передачу требуемого пакета. Модули должны удерживать ПАКЕТ_NAK, пока ИНИЦ ¦ ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

6.2.1.8 **Определения таймирования передачи**

НАЧАЛО_ПЕРЕДАЧИ TRANSACTION_BEGIN

ЗАДАТЧИК должен установить НАЧАЛО_ПЕРЕДАЧИ, если он хочет осуществить передачу и должен удерживать его, пока ФАЗА_ДАННЫХ ¦ ФАЗА_РАССОЕДИНЕНИЯ во время последней передачи.

ТАЙМ-АУТ_ПЕРЕДАЧИ TRANSACTION_TIMEOUT

До или во время установки НАЧАЛА_ПЕРЕДАЧИ задатчик должен сбросить ТАЙМ-АУТ_ПЕРЕДАЧИ.

ТАЙМ-АУТ_ПЕРЕДАЧИ таков, что через 128 мкс после 30%-ного сброса он должен вновь установиться, если не установлен ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ. Продолжительность ТАЙМ-АУТА_ПЕРЕДАЧИ может быть изменена в соответствии с 7.2.4.

ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ TRANSACTION_COMPLETE

Модули должны установить ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ, если ИНИЦ ¦ КОНЕЦ_РАССОЕДИНЕНИЯ, и они зафиксировали ТЕ★ и обновили состояние всех внутренних переменных при окончании передачи. ЗАДАТЧИК должен сбросить ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ, если as★ & ak★. Модули должны сбросить ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ, если -ai★.

ДЕКОДИРОВАННЫЙ_АДРЕС ADDR_DECODED

Модули должны поддерживать возможность декодирования адреса следующего обмена данных (не с отмененной байтовой шиной). Этот адрес должен первоначально устанавливаться по адресу, принимаему по линиям AD[]★ и действительному после выставления AS★. Этот адрес должен непрерывно корректироваться каждым участвующим модулем после каждой установки или снятия DS★, для которого передаются данные. Способ, посредством которого адрес корректируется, зависит от системы и не определен, исключая случай, когда он должен соответствовать адресу данных, передаваемых в следующем обмене данных. Модуль должен установить ДЕКОДИРОВАННЫЙ_АДРЕС, когда он декодировал адрес, соответствующий следующему обмену данных, и определил свой ответ на него. Во время фазы соединения, если ОШИБКА_ЧЕТНОСТИ_АДРЕСА ¦ ОШИБКА_ЧЕТНОСТИ КОМАНДЫ_СОЕДИНЕНИЯ, модуль не должен устанавливать ДЕКОДИРОВАННЫЙ_АДРЕС на длительность передачи. Модули должны удерживать ДЕКОДИРОВАННЫЙ_АДРЕС, пока модуль не снимет ai★ ¦ dk★ ¦ di★.

ФАЗА_СОЕДИНЕНИЯ CONNECTION_PHASE

ЗАДАТЧИК должен установить ФАЗУ_СОЕДИНЕНИЯ, если -ИНИЦ & —УДЕРЖАНИЕ_МАГИСТРАЛИ & НАЧАЛО_ПЕРЕДАЧИ & -AKf & AI★. Модули должны установить ФАЗУ_СОЕДИНЕНИЯ, если AS★ & ai★. Модули должны удерживать ФАЗУ_СОЕДИНЕНИЯ, пока ИНИЦ ¦ ФАЗА_ДАННЫХ ¦ ФАЗА_РАССОЕДИНЕНИЯ.

ФАЗА_ДАННЫХ DATA_PHASE

ЗАДАТЧИК должен установить ФАЗУ_ДАННЫХ, когда -ИНИЦ & —НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ & as★ & ПРИЗНАК_СОЕДИНЕНИЯ_ЗАФИКСИРОВАН. Модули должны установить ФАЗУ_ДАННЫХ, когда AS★ & DS★ & ПРИЗНАК_СОЕДИНЕНИЯ_ЗАФИКСИРОВАН. Модули должны удерживать ФАЗУ_ДАННЫХ, пока ИНИЦ ¦ ФАЗА_РАССОЕДИНЕНИЯ.

ФАЗА_РАССОЕДИНЕНИЯ DISCONNECTION_PHASE

ЗАДАТЧИК должен установить ФАЗУ_РАССОЕДИНЕНИЯ, когда -ИНИЦ & НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ & (ПРИЗНАК_СОЕДИНЕНИЯ_ЗАФИКСИРОВАН ¦ ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН). Модули должны установить ФАЗУ_РАССОЕДИНЕНИЯ, когда они обнаружат снятие AS★. Модули должны удерживать ФАЗУ_РАССОЕДИНЕНИЯ до ИНИЦ ¦ КОНЕЦ_РАССОЕДИНЕНИЯ.

ПРИЗНАК_СОЕДИНЕНИЯ_ЗАФИКСИРОВАН CONNECTION_INFO_CAPTURED

Модули должны установить ПРИЗНАК_СОЕДИНЕНИЯ_ЗАФИКСИРОВАН, если СТАТУС_СОЕДИНЕНИЯ и они приняли информацию фазы соединения. Модули должны удерживать ПРИЗНАК_СОЕДИНЕНИЯ_ЗАФИКСИРОВАН до ИНИЦ ¦ КОНЕЦ_РАССОЕДИНЕНИЯ.

ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН DATA_INFO_CAPTURED

ЗАДАТЧИК должен установить ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН, если ФАЗА_ДАННЫХ и он приняли информацию исполнителя и

1) СТАТУС_ШИРОКОВЕЩАНИЯ & (ds★ & -DK★ ¦ -ds★ & -DK★),

2) СТАТУС_ШИРОКОВЕЩАНИЯ & (ds★ & -DIf ¦ -ds★ & -DKf).

ЗАДАТЧИК должен удерживать ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН, пока не произойдет следующий переход ds★ или до ФАЗЫ_РАССОЕДИНЕНИЯ.

Если (ЗАПИСЬ_ИСПОЛНИТЕЛЯ ¦ ШИРОКОВЕЩАНИЕ), то УЧАСТВУЮЩИЙ исполнитель должен установить ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН, когда он принял информацию ЗАДАТЧИКА, указанную переходом DS★.

Если ЗАПИСЬ_ИСПОЛНИТЕЛЯ ¦ ШИРОКОВЕЩАНИЕ, то УЧАСТВУЮЩИЙ исполнитель должен установить ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН, когда он принял информацию, указанную (DS★ & DK★ ¦ -DS★ & DI★). Исполнители должны гарантировать, что задержка их DS★ больше или равна задержке DK★ или DI★ в соответствии с требованиями этого параграфа.

УЧАСТВУЮЩИЕ исполнители должны удерживать ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН до тех пор, пока они не снимут dk★ или di★.

ПРИЗНАК_РАССОЕДИНЕНИЯ_ЗАФИКСИРОВАН DISC_INFO_CAPTURED

УЧАСТВУЮЩИЙ исполнитель должен установить ПРИЗНАК_РАССОЕДИНЕНИЯ_ЗАФИКСИРОВАН, когда он принял информацию ЗАДАТЧИКА, указанную снятием AS★, и должен удерживать ПРИЗНАК_РАССОЕДИНЕНИЯ_ЗАФИКСИРОВАН до того, как он снимет ak★.

НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ TRANSFER_COUNT_ZERO

ЗАДАТЧИК должен сбросить НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ до начала ФАЗЫ_СОЕДИНЕНИЯ, если он планирует передавать данные во время ФАЗЫ_ДАННЫХ.

Сразу же после того, как ЗАДАТЧИК вызовет переход DS★, требуемый в последнем обмене или пакете передачи, он должен установить НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ.

ЗАДАТЧИК должен установить НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ, если СТАТУС_ПРИНУЖДЕНИЯ & ПЕРЕДАЧА_1 & -ЧАСТНЫЙ и он выставил ds★.

ЗАДАТЧИК должен установить НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ, если СТАТУС_ПРИНУЖДЕНИЯ & ПЕРЕДАЧА_1 & ЧАСТНЫЙ и он снял ds★.

ЗАДАТЧИК должен установить НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ, если СТАТУС_ПРИНУЖДЕНИЯ и он инициировал обмен данных переходом ds★, чей порядок адресных битов, представленных A[]★, соответствует одному из следующих правил:

Если -ШИРИНА_ДАННЫХ_1 & -ШИРИНА_ДАННЫХ_0, то установить N=2
Если -ШИРИНА_ДАННЫХ_1 & ШИРИНА_ДАННЫХ_0, то установить N=3
Если ШИРИНА_ДАННЫХ_1 & -ШИРИНА_ДАННЫХ_0, то установить N=4
Если ШИРИНА_ДАННЫХ_1 & ШИРИНА_ДАННЫХ_0, то установить N=5

Если ПЕРЕДАЧА_2	и	A[N]★	равен 1,	то НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ
Если ПЕРЕДАЧА_4	и	A[N+1 . . . N]★	равны 1,	то НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ
Если ПЕРЕДАЧА_8	и	A[N+2 . . . N]★	равны 1,	то НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ
Если ПЕРЕДАЧА_16	и	A[N+3 . . . N]★	равны 1,	то НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ
Если ПЕРЕДАЧА_32	и	A[N+4 . . . N]★	равны 1,	то НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ
Если ПЕРЕДАЧА_64	и	A[N+5 . . . N]★	равны 1,	то НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ

ЗАДАТЧИК должен установить НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ, если (ST6★ ¦ -ST2★) & ПРИЗНАК_СОЕДИНЕНИЯ_ЗАФИКСИРОВАН ¦ ОШИБКА_ПЕРЕДАЧИ ¦ СТАТУС_ОЖИДАНИЯ ¦ СТАТУС_ЗАНЯТОСТИ ¦ НЕБЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС ¦ БЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС ¦ НЕДЕЙСТВИТЕЛЬНОСТЬ ¦ ОТВЕТ_ЗАПИСИ ¦ ОШИБКА_СПОСОБНОСТИ_ЗАДАТЧИКА.

ЗАДАТЧИК должен установить НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ, если СТАТУС_РАСЩЕПЛЕНИЯ &(БЛОКИРОВАНИЕ_ЧТЕНИЕ ¦ НЕБЛОКИРОВАНИЕ_ЧТЕНИЕ ¦ НЕДЕЙСТВИТЕЛЬНОСТЬ_ЧТЕНИЯ ¦ РАЗДЕЛЕННОЕ_ЧТЕНИЕ ¦ МОДИФИКАЦИЯ_ЧТЕНИЯ ¦ КОПИРОВАНИЕ_НАЗАД ¦ (ЧАСТНОЕ_ЧТЕНИЕ ¦ БЛОКИРОВАНИЕ_ЧАСТНОЕ_ЧТЕНИЕ) & он выставил ds★).

ЗАДАТЧИК должен установить НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ, если -ЧАСТНЫЙ & (ВЫБОРКА_СЛОЖЕНИЕ_СО_СТАРШЕГО ¦ ВЫБОРКА_СЛОЖЕНИЕ_С_МЛАДШЕГО) & выставил ds★ во время первого обмена данных ¦ (-ЧАСТНЫЙ & (МАСКИРОВАНИЕ_ОБМЕН ¦ СРАВНЕНИЕ_ОБМЕН) ¦ (ЧАСТНЫЙ & ВЫБОРКА_СЛОЖЕНИЕ_СО_СТАРШЕГО ¦ ВЫБОРКА_СЛОЖЕНИЕ_С_МЛАДШЕГО)) & снят ds★ во время второго обмена данных) ¦ ЧАСТНЫЙ & (МАСКИРОВАНИЕ_ОБМЕН ¦ СРАВНЕНИЕ_ОБМЕН) & выставлен ds★ во время третьего обмена данных.

Задатчик должен установить НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ, если РЕЖИМ МНОЖЕСТВЕННЫХ_ПАКЕТОВ & СТАТУС_КОНЦА_ДАННЫХ.

ЗАДАТЧИК может установить НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ, если СТАТУС_ПРИНУЖДЕНИЯ & (СТАТУС_КОНЦА_ДАННЫХ ¦ СТАТУС_ОШИБКИ_ОБМЕНА). ЗАДАТЧИК должен игнорировать данные, получаемые после того, как СТАТУС_КОНЦА_ДАННЫХ индицирован им.

Если ЗАДАТЧИК не знает о том, как исполнители устанавливают BE★ по отношению к обмену, в котором произошла ошибка (т. е. число стадий в потоке), он должен игнорировать все данные, пришедшие во время передачи.

Если ЗАДАТЧИК не знает о том, как исполнители устанавливают BE★, он может использовать передаваемые данные до того, как обнаружится ошибка.

СТАТУС_СОЕДИНЕНИЯ CONNECTION_STATUS

Модули должны установить СТАТУС_СОЕДИНЕНИЯ, когда ЗАДАТЧИК & -Alf ¦ УЧАСТИЕ & (AS★ & DS★ ¦ AS★).

КОНЕЦ_РАССОЕДИНЕНИЯ DISCONNECTION_END

Модули должны установить КОНЕЦ_РАССОЕДИНЕНИЯ, когда ЗАДАТЧИК & -AKf ¦ УЧАСТИЕ & (-AKf ¦ следующий AS★). Исполнители должны гарантировать, что задержка их AS★ больше или равна задержке AK★ в соответствии с требованиями этого параграфа.

6.2.2 **Определение сигналов**

6.2.2.1 **Сигналы синхронизации**

6.2.2.1.1 **AS★ (Синхронизация адреса)**

ЗАДАТЧИК должен выставить as★, когда -ИНИЦ & НАЧАЛО_ПЕРЕДАЧИ & ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ & -AKf & AI★ & -УДЕРЖАНИЕ_МАГИСТРАЛИ & ФАЗА_СОЕДИНЕНИЯ, и продолжать выставлять его до тех пор, пока ИНИЦ ¦ -ЗАДАТЧИК ¦ НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ & et★ & (ПРИЗНАК_СОЕДИНЕНИЯ_ЗАФИКСИРОВАН ¦ ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН).

6.2.2.1.2 **AK★ (Подтверждение адреса)**

Все модули должны выставить ak★, когда -ИНИЦ & AS★, и продолжать выставлять его до тех пор, пока ИНИЦ ¦ -AS★ ПРИЗНАК_РАССОЕДИНЕНИЯ_ЗАФИКСИРОВАН & -dk★ & di★ & -ds★.

6.2.2.1.3 AI★ **(Подтверждение адреса инверсное)**

Все модули должны выставить ai★ в то время, когда -(ВКЛЮЧЕНИЕ_ПИТАНИЯ & -НЕЗАНЯТОСТЬ_МАГИСТРАЛИ_IUS) & (ИНИЦ ¦ модуль обнаруживает снятие AS★). Модули должны продолжать выставлять ai★ до тех пор, пока AS★ & ДЕКОДИРОВАННЫЙ_АДРЕС & ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

6.2.2.1.4 **DS★ (Синхронизация данных)**

ЗАДАТЧИК должен выставить ds★, когда -ИНИЦ & ЗАДАТЧИК & (-НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ & (нарастающий фронт ФАЗЫ_ДАННЫХ ¦ нарастающий фронт ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН), и должен продолжать выставлять его до тех пор, пока ИНИЦ ¦ -ЗАДАТЧИК ¦ нарастающий фронт ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН & -НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ ¦ as★ & -СТАТУС_ШИРОКОВЕЩАНИЯ & -DK★ ¦ СТАТУС_ШИРОКОВЕЩАНИЯ & -DKf).

6.2.2.1.5 **DK★ (Подтверждение данных)**

Модули должны выставить dk★, когда -ИНИЦ & AS★ & DS★ & УЧАСТИЕ & -(ЗАПИСЬ_ИСПОЛНИТЕЛЯ ¦ СТАТУС_ШИРОКОВЕЩАНИЯ).

Если ЗАПИСЬ_ИСПОЛНИТЕЛЯ ¦ СТАТУС_ШИРОКОВЕЩАНИЯ:

1) Модуль(и), выставляющий(ие) данные, должен(ны) выставить dk★, когда -ИНИЦ & AS★ & DS★ & УЧАСТИЕ и данные действительны.

2) Модули, не выставляющие данные, должны выставить dk★, когда -ИНИЦ & AS★ & DS★ & DK★.

Модули должны выставлять dk★ до тех пор, пока ИНИЦ ¦ -DS★ & ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН & di★ ¦ -AS★.

6.2.2.1.6 **DI★ (Подтверждение данных инверсное)**

Модули должны выставить di★, когда -ИНИЦ & AS★ & -DS★ & УЧАСТИЕ & -(ЗАПИСЬ_ИСПОЛНИТЕЛЯ ¦ СТАТУС_ШИРОКОВЕЩАНИЯ).

Если ЗАПИСЬ_ИСПОЛНИТЕЛЯ ¦ СТАТУС_ШИРОКОВЕЩАНИЯ:

1) Модули, выставляющие данные, должны выставить di★, когда -ИНИЦ & AS★ & -DS★ & УЧАСТИЕ и данные действительны.

2) Модули, не выставляющие данные, должны выставить di★, когда -ИНИЦ & AS★ & -DS★ & DI★.

Модули должны выставлять di★ до тех пор, пока ИНИЦ DS★ & ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН & dk★ ¦ -AS★. В ФАЗЕ_СОЕДИНЕНИЯ, если УЧАСТИЕ, DI★ должен быть выставлен перед снятием ai★.

6.2.2.2 **CM[7. . .0]★ (Command)**

Сигналы cm[7. . .0]★ и cp★ должны быть стабильными при выходе на магистраль перед тем, как переход as★ или ds★ покажет их действительность. Если необходимо, разработчик модуля должен задержать переход as★ и ds★ больше, чем требуется в 6.2.2.1.1 и 6.2.2.1.4, чтобы выполнить условия этого параграфа.

6.2.2.2.1 **CM0★ (Команда 0)**

ЗАДАТЧИК должен выставить cm0★, когда -ИНИЦ & ЗАДАТЧИК & (ФАЗА_СОЕДИНЕНИЯ & ПЕРЕДАЧА_СМД_0 ¦ ФАЗА_ДАННЫХ & (БЛОКИРОВКА_СМД_0 ¦ -СТАТУС_ПРИНУЖДЕНИЯ & ЗАПРОС_ПАКЕТА) ¦ ФАЗА_РАССОЕДИНЕНИЯ & УДЕРЖАНИЕ_БЛОКИРОВКИ).

6.2.2.2.2 **CM1★ (Команда 1)**

ЗАДАТЧИК должен выставить cm1★, когда -ИНИЦ & ЗАДАТЧИК & (ФАЗА_СОЕДИНЕНИЯ & ПЕРЕДАЧА_СМД_1 ¦ ФАЗА_ДАННЫХ & (БЛОКИРОВКА_СМД_1 ¦ СТАТУС_ФЛАГА_ПЕРЕДАЧИ) ¦ ФАЗА_РАССОЕДИНЕНИЯ & БОЛЬШЕ & (СТАТУС_РАСЩЕПЛЕНИЯ & НЕБЛОКИРОВАНИЕ_ЧТЕНИЕ ¦ НЕБЛОКИРОВАНИЕ_ЗАПИСЬ)).

6.2.2.2.3 **CM2★ (Команда 2)**

ЗАДАТЧИК должен выставить cm2★, когда -ИНИЦ & ЗАДАТЧИК & (ФАЗА_СОЕДИНЕНИЯ & ПЕРЕДАЧА_СМД_2 ¦ ФАЗА_ДАННЫХ & БЛОКИРОВКА_СМД_1).

6.2.2.2.4 **CM3★ (Команда 3)**

ЗАДАТЧИК должен выставить cm3★, когда -ИНИЦ & ЗАДАТЧИК & (ФАЗА_СОЕДИНЕНИЯ & ПЕРЕДАЧА_СМД_3)

6.2.2.2.5 **CM4★ (Команда 4)**

ЗАДАТЧИК должен выставить cm4★, когда -ИНИЦ & ЗАДАТЧИК & (ФАЗА_СОЕДИНЕНИЯ & ЗАПИСЬ_СМД ¦ (ФАЗА_ДАННЫХ ¦ ФАЗА_РАССОЕДИНЕНИЯ) & СТАТУС_ВНЕДРЕННОСТИ).

6.2.2.2.6 **CM5★ (Команда 5)**

ЗАДАТЧИК должен выставить cm5★, когда -ИНИЦ & ЗАДАТЧИК & (ФАЗА_СОЕДИНЕНИЯ & ШИРИНА_ДАННЫХ_0 ¦ ФАЗА_ДАННЫХ & ДЛИНА_ДАННЫХ_0 ¦ ФАЗА_РАССОЕДИНЕНИЯ & ДЛИНА_ДАННЫХ_0).

6.2.2.2.7 **CM6★ (Команда 6)**

ЗАДАТЧИК должен выставить cm6★, когда -ИНИЦ & ЗАДАТЧИК & (ФАЗА_СОЕДИНЕНИЯ & РАЗРЯДНОСТЬ_ДАННЫХ_1 ¦ ФАЗА_ДАННЫХ & ДЛИНА_ДАННЫХ_1 ¦ ФАЗА_РАССОЕДИНЕНИЯ & ДЛИНА_ДАННЫХ_1).

6.2.2.2.8 **CM7★ (Команда 7)**

ЗАДАТЧИК должен выставить cm7★, когда -ИНИЦ & ЗАДАТЧИК & (ФАЗА_СОЕДИНЕНИЯ & РАЗРЯДНОСТЬ_АДРЕСА_2 ¦ ФАЗА_ДАННЫХ & ДЛИНА_ДАННЫХ_2 ¦ ФАЗА_РАССОЕДИНЕНИЯ & ДЛИНА_ДАННЫХ_2).

6.2.2.2.9 **(Четность команды)**

ЗАДАТЧИК должен выставить cp★, если он установил четное число cm[]★ сигналов

6.2.2.3 **CA[2. . .0]★ (Способность)**

Сигналы ca[2. . .0]★ должны быть стабильными при выводе на магистраль перед тем, как снятие ai★ покажет их действительность. Если необходимо, разработчик модуля должен задерживать снятие ai★ на большее время, чем требуется в 6.2.2.1.3, чтобы выполнить условие этого параграфа.

6.2.2.3.1 **CA0★ (TS★)**

Модули должны выставить ca0★, когда НИЗКАЯ_СКОРОСТЬ & ФАЗА_СОЕДИНЕНИЯ, и должны продолжать выставлять ca0★ до тех пор, пока ИНИЦ : КОНЕЦ_РАССОЕДИНЕНИЯ.

6.2.2.3.2 **CA1★ (C0★)**

Модули должны выставить ca1★, когда ПРИНУДИТЕЛЬНЫЙ, и должны продолжать выставлять ca1★ до тех пор, пока ИНИЦ ¦ КОНЕЦ_РАССОЕДИНЕНИЯ.

6.2.2.3.3 **CA2★ (SR★)**

Модули должны выставить ca2★, когда РАСЩЕПЛЕННЫЙ, и должны продолжать выставлять ca2★ до тех пор, пока ИНИЦ ¦ КОНЕЦ_РАССОЕДИНЕНИЯ.

6.2.2.4 **ST[7. . .0]★ (Статус)**

6.2.2.4.1 **ST0★ (TE★)**

Модули должны выставить st0★, когда -ИНИЦ & ОШИБКА_ПЕРЕДАЧИ & (AS★ & -AIf ¦ -AS★ & ak★), и должны продолжать выставлять st0★ до тех пор, пока ИНИЦ ¦ СТАТУС_СОЕДИНЕНИЯ.

Сигнал st0★ должен быть стабильным при выводе на магистраль перед тем, как снятие ak★ покажет его действительность. Если необходимо, разработчик модуля должен задерживать снятие ak★ на большее время, чем требуется в 6.2.2.1.2, чтобы выполнить это условие.

6.2.2.4.2 **ST1★ (BS★/ED★)**

Модули должны выставить st1★, когда ЗАНЯТОСТЬ, и должны продолжать выставлять st1★ до тех пор, пока ИНИЦ . КОНЕЦ_РАССОЕДИНЕНИЯ.

Модули должны выставить st1★, когда КОНЕЦ_ДАННЫХ ПАКЕТ NAK, и должны продолжать выставлять st1★ до тех пор, пока ИНИЦ ¦ КОНЕЦ_РАССОЕДИНЕНИЯ.

Сигнал st1★ должен быть стабильным при выводе на магистраль перед тем, как снятие ai★, dk★ или di★ покажет его действительность. Если необходимо, разработчик модуля должен задерживать снятие ai★, dk★ или di★ на большее время, чем требуется в 6.2.2.1.3, 6.2.2.1.5 или 6.2.2.1.6, чтобы выполнить это условие.

6.2.2.4.3 **ST2★ (SL★)**

Модули должны выставить st2★, когда ВЫБРАННЫЙ, и должны продолжать выставлять st2 ★ до тех пор, пока ИНИЦ ¦ -ВЫБРАННЫЙ ¦ КОНЕЦ_РАССОЕДИНЕНИЯ.

Сигнал st2★ должен быть стабильным при выводе на магистраль перед тем, как снятие ai★, dk★ или di★ покажет его действительность. Если необходимо, разработчик модуля должен задерживать снятие ai★, dk★ или di★ на большее время, чем требуется в 6.2.2.1.3, 6.2.2.1.5 или 6.2.2.1.6, чтобы выполнить это условие.

6.2.2.4.4 **ST3★ (BS★)**

Модули должны выставить st3★, когда ШИРОКОВЕЩАНИЕ ¦ ШИРОКОЗАПРОСНЫЙ_АДРЕС ¦ ЛОВЯЩИЙ_ДАННЫЕ, и должны продолжать выставлять st3★ до тех пор, пока ИНИЦ ¦ ¦ КОНЕЦ_РАССОЕДИНЕНИЯ.

Сигнал st3★ должен быть стабильным при выводе на магистраль перед тем, как снятие ai★ покажет его действительность. Если необходимо, разработчик модуля должен задерживать снятие ai★ на большее время, чем требуется в 6.2.2.1.3, чтобы выполнить это условие.

6.2.2.4.5 **ST4★ (TF★)**

Модули должны выставить st4★, когда КЕШ_СМД & ВЫСТАВЛЕНИЕ_CS ¦ ВЫБРАННЫЙ_ПОЧТОВЫЙ_ЯЩИК & СООБЩЕНИЕ_КОНФЛИКТА, и должны продолжать выставлять st4★ до тех пор, пока ИНИЦ ¦ -ВЫСТАВЛЕНИЕ_CS & КЕШ_СМД ¦ -СООБЩЕНИЕ_КОНФЛИКТА & ВЫБРАННЫЙ_ПОЧТОВЫЙ_ЯЩИК ¦ КОНЕЦ_РАССОЕДИНЕНИЯ.

Сигнал st4★ должен быть стабильным при выводе на магистраль перед тем, как снятие ai★, dk★ или di★ покажет его действительность. Если необходимо, разработчик модуля должен задерживать снятие ai★, dk★ или di★ на большее время, чем требуется в 6.2.2.1.3, 6.2.2.1.5 или 6.2.2.1.6, чтобы выполнить это условие

6.2.2.4.6 **ST5★ (IV★)**

Модули должны выставить st5★, когда ВНЕДРЕННОСТЬ, и должны продолжать выставлять st5★ до тех пор, пока ИНИЦ ¦ -ВНЕДРЕННОСТЬ ¦ КОНЕЦ_РАССОЕДИНЕНИЯ.

Сигнал st5★ должен быть стабильным при выводе на магистраль перед тем, как снятие ai★, dk★ или di★ покажет его действительность. Если необходимо, разработчик модуля должен задерживать снятие ai★, dk★ или di★ на большее время, чем требуется в 6.2.2.1.3, 6.2.2.1.5 или 7.2.2.1.6, чтобы выполнить это условие.

6.2.2.4.7 **ST6★ (BE★)**

Модули должны выставить st6★, когда ОШИБКА_СОБЫТИЯ, и должны продолжать выставлять st6★ до тех пор, пока ИНИЦ ¦ КОНЕЦ_РАССОЕДИНЕНИЯ.

Сигнал st6★ должен быть стабильным при выводе на магистраль перед тем, как снятие ai★, dk★ или di★ покажет его действительность. Если необходимо, разработчик модуля должен задерживать снятие ai★, dk★ или di★ на большее время, чем требуется в 6.2.2.1.3, 6.2.2.1.5 или 7.2.2.1.6, чтобы выполнить это условие.

6.2.2.4.8 **ST7★ (WT★)**

Модули должны выставить st7★, когда ОЖИДАНИЕ, и должны продолжать выставлять st7★ до тех пор, пока ИНИЦ ¦ КОНЕЦ_РАССОЕДИНЕНИЯ.

Сигнал st7★ должен быть стабильным при выводе на магистраль перед тем, как снятие ai★, dk★ или di★ покажет его действительность. Если необходимо, разработчик модуля должен задерживать снятие ai★, dk★ или di★ на большее время, чем требуется в 6.2.2.1.3, 6.2.2.1.5 или 6.2.2.1.6, чтобы выполнить это условие.

6.2.2.5 **Информационные поля**

6.2.2.5.1 **AD[63. . .0]★ (Адрес/Данные)**

Модули должны снимать ad[]★, если ИНИЦ.

Когда -ИНИЦ & ФАЗА_СОЕДИНЕНИЯ & -(ОТВЕТ_ЧТЕНИЯ ¦ ОТВЕТ_ЗАПИСИ) & -РАЗРЯДНОСТЬ_АДРЕСА & -РАЗРЯДНОСТЬ_ДАННЫХ_1 & -РАЗРЯДНОСТЬ_ДАННЫХ_0, ЗАДАТЧИК должен активировать ad[31. . .2]★ адресом, снять ad[63. . .32]★, и может активировать ad[1. . .0]★ определенными пользователем данными.

Когда -ИНИЦ & ФАЗА_СОЕДИНЕНИЯ & -(ОТВЕТ_ЧТЕНИЯ ¦ ОТВЕТ_ЗАПИСИ) & -РАЗРЯДНОСТЬ_АДРЕСА & -РАЗРЯДНОСТЬ_ДАННЫХ_1 & РАЗРЯДНОСТЬ_ДАННЫХ_0, ЗАДАТЧИК должен активировать ad[31. . .3]★ адресом, снять ad[63. . .32]★, и может активировать ad[2. . .0]★ определенными пользователем данными.

Когда -ИНИЦ & ФАЗА_СОЕДИНЕНИЯ & -(ОТВЕТ_ЧТЕНИЯ ¦ ОТВЕТ_ЗАПИСИ) & -РАЗРЯДНОСТЬ_АДРЕСА & РАЗРЯДНОСТЬ_ДАННЫХ_1 & -РАЗРЯДНОСТЬ_ДАННЫХ_0, ЗАДАТЧИК должен активировать ad[31. . .4]★ адресом, снять ad[63. . .32]★, и может активировать ad[3. . .0]★ определенными пользователем данными.

Когда -ИНИЦ & ФАЗА_СОЕДИНЕНИЯ & -(ОТВЕТ_ЧТЕНИЯ ¦ ОТВЕТ_ЗАПИСИ) & -РАЗРЯДНОСТЬ_АДРЕСА & РАЗРЯДНОСТЬ_ДАННЫХ_1 & РАЗРЯДНОСТЬ_ДАННЫХ_0, ЗАДАТЧИК должен активировать ad[31. . .5]★ адресом, снять ad[63. . .32]★, и может активировать ad[4. . .0]★ определенными пользователем данными.

Когда -ИНИЦ & ФАЗА_СОЕДИНЕНИЯ & -(ОТВЕТ_ЧТЕНИЯ ¦ ОТВЕТ_ЗАПИСИ) & -РАЗРЯДНОСТЬ_АДРЕСА & -РАЗРЯДНОСТЬ_ДАННЫХ_1 & -РАЗРЯДНОСТЬ_ДАННЫХ_0, ЗАДАТЧИК должен активировать ad[63. . .2]★ адресом, и может активировать ad[1. . .0]★ определенными пользователем данными.

Когда -ИНИЦ & ФАЗА_СОЕДИНЕНИЯ & -(ОТВЕТ_ЧТЕНИЯ ¦ ОТВЕТ_ЗАПИСИ) & РАЗРЯДНОСТЬ_АДРЕСА & -РАЗРЯДНОСТЬ_ДАННЫХ_1 & РАЗРЯДНОСТЬ_ДАННЫХ_0, ЗАДАТЧИК должен активировать ad[63. . .3]★ адресом, и может активировать ad[2. . .0]★ определенными пользователем данными

Когда -ИНИЦ & ФАЗА_СОЕДИНЕНИЯ & -(ОТВЕТ_ЧТЕНИЯ ¦ ОТВЕТ_ЗАПИСИ) & РАЗРЯДНОСТЬ_АДРЕСА & РАЗРЯДНОСТЬ_ДАННЫХ_1 & -РАЗРЯДНОСТЬ_ДАННЫХ_0, ЗАДАТЧИК должен активировать ad[63. . 4]★ адресом, и может активировать ad[3. . .0]★ определенными пользователем данными.

Когда -ИНИЦ & ФАЗА_СОЕДИНЕНИЯ & (ОТВЕТ_ЧТЕНИЯ ¦ ОТВЕТ_ЗАПИСИ) & РАЗРЯДНОСТЬ_АДРЕСА & РАЗРЯДНОСТЬ_ДАННЫХ_1 & РАЗРЯДНОСТЬ_ДАННЫХ_0, ЗАДАТЧИК должен активизировать ad[63. . .5]★ адресом, и может активизировать ad[4. . .0]★ определенными пользователем данными.

Когда -ИНИЦ & ФАЗА_СОЕДИНЕНИЯ & -(ОТВЕТ_ЧТЕНИЯ ¦ ОТВЕТ_ЗАПИСИ), ЗАДАТЧИК должен:

1) выставить ad[31. . .28]★.

2) активировать ad[27. . .12]★ глобальным идентификатором запросчика,

3) активировать ad[5. . .0]★ идентификатором передачи.

Когда -ИНИЦ & ФАЗА_РАССОЕДИНЕНИЯ & (ОТВЕТ_ЧТЕНИЯ ¦ ОТВЕТ_ЗАПИСИ), ЗАДАТЧИК должен.

1) активировать ad[15. . .8]★ первоначальным приоритетом передачи запросчика,

2) активировать ad2★ выбранным статусом, который был указан в передаче запроса,

3) активировать ad1★ статусом конфликта, который был указан в передаче запроса,

4) активировать ad0★ статусом ошибки передачи, который был указан в передаче запроса,

и снять ad[63. . .32]★, ad[32. . .16]★ и ad[7. . .3]★.

Когда -ИНИЦ & ФАЗА_РАССОЕДИНЕНИЯ & (РАЗДЕЛЕННЫЙ_ОТВЕТ ¦ МОДИФИЦИРОВАННЫЙ_ОТВЕТ), ЗАДАТЧИК должен:

1) выставить ad[31. . .28]★,

2) активировать ad[27. . .12]★ глобальным идентификатором запросчика,

3) активировать ad[11. . .4]★ первоначальным приоритетом передачи запросчика,

4) активировать ad2★ выбранным статусом, который был обозначен в передаче запроса,

5) активировать ad1★ статусом конфликта, который был обозначен в передаче запроса,

6) активировать ad0★ статусом ошибки передачи, который был обозначен в передаче запроса,

и снять ad[63. . .32]★ и ad3★.

Когда -ИНИЦ & ФАЗА_РАССОЕДИНЕНИЯ & (РАСЩЕПЛЕННЫЙ СТАТУС ¦ НЕБЛОКИРОВАНИЕ_ЗАПИСЬ ¦ НЕБЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС ¦ БЛОКИРОВАНИЕ_ЗАПИСЬ ¦ БЛОКИРОВАНИЕ_ТОЛЬКО_АДРЕС ¦ ЧАСТНАЯ_ЗАПИСЬ ¦ БЛОКИРОВАНИЕ_ЧАСТНАЯ_ЗАПИСЬ ¦ ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ ¦ НЕДЕЙСТВИТЕЛЬНОСТЬ_ЗАПИСИ ¦ КОПИРОВАНИЕ_НАЗАД ¦ НЕДЕЙСТВИТЕЛЬНОСТЬ), ЗАДАТЧИК должен:

1) активировать ad[31. . .16]★ глобальным идентификатором запросчика,

2) активировать ad[15. . .8]★ первоначальным приоритетом передачи запросчика,

3) активировать ad[5. . .0]★ идентификатором передачи

и снять ad[63. . .32]★ и ad[7. . .6]★.

Когда -ИНИЦ & ФАЗА_ДАННЫХ & ЗАПИСЬ_СМД, ЗАДАТЧИК должен активировать ad[31. . .0]★ данными, соответствующими текущему обмену или пакету.

Когда УЧАСТВУЮЩИЙ исполнитель отвечает за передачу данных ЗАДАТЧИКУ, он должен активировать ad[31. . .0]★ данными записи, связанными с текущим обменом или пакетом, в ответ на переходы DS★.

Когда УЧАСТВУЮЩИЙ исполнитель отвечает за передачу данных ЗАДАТЧИКУ и (64_РАЗРЯДНЫЕ_ДАННЫЕ ¦ 128_РАЗРЯДНЫЕ_ДАННЫЕ ¦ 256_РАЗРЯДНЫЕ_ДАННЫЕ), он должен активировать ad[63. . .32]★ данными записи, связанными с текущим обменом или пакетом, в ответ на переходы DS★.

Если выбранная передача использует пакетный протокол, сигналы на ad[]★ должны подчиняться пакетному протоколу в соответствии с 6.2.3.21. Пакеты всегда должны быть выравнены на размеры пакетных границ модуля.

Сигналы ad[]★ должны быть стабильными при выводе на магистраль перед тем, как переход as★ покажет их действительность. Если необходимо, разработчик модуля должен задерживать переход as★ на большее время, чем требуется в 6.2.2.1.1, чтобы выполнить это условие.

В принудительном режиме сигналы ad[]★ должны быть стабильными при выводе на магистраль перед тем, как переход ds★ покажет их действительность, если передает ЗАДАТЧИК. Если необходимо, разработчик модуля должен задерживать переход ds★ на большее время, чем требуется в 6.2.2.1.4, чтобы выполнить это условие. В принудительном режиме сигналы ad[]★ должны быть стабильными при выводе на магистраль перед тем, как переход dk★ или di★ покажет их действительность, если передают один или более исполнителей. Если необходимо, разработчик модуля должен задерживать переход dk★ di★ на большее время, чем требуется в 6.2.2.1.5 или 6.2.2.1.6, чтобы выполнить это условие.

В принудительных обменах, где ЗАПИСЬ_ИСПОЛНИТЕЛЯ ¦ ШИРОКОВЕЩАТЕЛЬНЫЙ_СТАТУС, сигналы ad[]★ должны быть стабильными при выводе на магистраль перед тем, как выставление dk★ или di★ покажет их действительность, если передают один или более исполнителей. Если необходимо, разработчик модуля должен задерживать установление dk★ или di★ на большее время, чем требуется в 6.2.2.1.5 или 6.2.2.1.6, чтобы выполнить это условие.

Определение невыбранной байтовой шины должно быть следующим:

ad[31. . .0]★	Байтовая шина	Невыбранная шина
ad31★	D[255. . .248]★	L31★
ad30★	D[247. . .240]★	L30★
ad29★	D[239. . .232]★	L29★
ad28★	D[231. . .224]★	L28★
ad27★	D[223. . .216]★	L27★
ad26★	D[215. . .208]★	L26★
ad25★	D[207. . .200]★	L25★
ad24★	D[199. . .192]★	L24★
ad23★	D[191. . .184]★	L23★
ad22★	D[183. . .176]★	L22★
ad21★	D[175. . .168]★	L21★
ad20★	D[167. . .160]★	L20★
ad19★	D[159. . .152]★	L19★
ad18★	D[151. . .144]★	L18★
ad17★	D[143. . .136]★	L17★
ad16★	D[135. . .128]★	L16★
ad15★	D[127. . .120]★	L15★
ad14★	D[119. . .112]★	L14★
ad13★	D[111. . .104]★	L13★
ad12★	D[103. . .96]★	L12★
ad11★	D[95. . .88]★	L11★
ad10★	D[87. . .80]★	L10★
ad9★	D[79. . .72]★	L9★
ad8★	D[71. . .64]★	L8★
ad7★	AD[63. . .56]★	L7★
ad6★	AD[55. . .48]★	L6★
ad5★	AD[47. . .40]★	L5★
ad4★	AD[39. . .32]★	L4★
ad3★	AD[31. . .24]★	L3★
ad2★	AD[23. . .16]★	L2★
ad1★	AD[15. . .8]★	L1★
ad0★	AD[7. . .0]★	L0★

Во время первого обмена передачи, когда -ИНИЦ & ФАЗА_ДАННЫХ & ЧАСТНЫЙ, ЗАДАТЧИК должен активировать ad[31 . . . 0]★ информацией о невыбранной шине в соответствии с этой таблицей и должен снять ad[63 . . . 32]★. Модули должны снимать сигналы с невыбранной шины, если шины, которыми они управляют, не были выбраны DW[]★ командой в фазе соединения

6.2.2.5.2 **D[255. . . 64]★ (Данные)**

Когда -ИНИЦ & ФАЗА_ДАННЫХ & ЗАПИСЬ_СМД & ШИРИНА_ДАННЫХ_1, ЗАДАТЧИК должен активировать d[127. . . 64]★ данными записи, относящимися к текущему обмену или пакету.

Когда -ИНИЦ & ФАЗА_ДАННЫХ & ЗАПИСЬ_СМД & (ШИРИНА_ДАННЫХ_1 & ШИРИНА_ДАННЫХ_0), ЗАДАТЧИК должен активировать d[255 . . . 128]★ данными записи, относящимися к текущему обмену или пакету.

Когда УЧАСТВУЮЩИЙ исполнитель отвечает за передачу данных ЗАДАТЧИКУ и (128_РАЗРЯДНЫЕ_ДАННЫЕ | 256_РАЗРЯДНЫЕ-ДАННЫЕ), он должен активировать d[127 . . . 64]★ данными записи, относящимися к текущему обмену или пакету

Когда УЧАСТВУЮЩИЙ исполнитель отвечает за передачу данных ЗАДАТЧИКУ и 256_РАЗРЯДНЫЕ_ДАННЫЕ, он должен активировать d[255 . . . 128]★ данными записи, относящимися к текущему ОБМЕНУ или пакету.

Во время первого обмена передачи, когда -ИНИЦ & ФАЗА_ДАННЫХ & ЧАСТНЫЙ, ЗАДАТЧИК должен снять d[255 . . . 64]★.

Если выбрана передача с использованием пакетного протокола, то сигналы на d[]★ должны подчиняться пакетному протоколу в соответствии с 6.2.3.21.

В принудительном режиме сигналы d[]★ должны быть стабильными при выводе на магистраль перед тем, как переход ds★ покажет их действительность, если передают один или более исполнителей. Если необходимо, разработчик модуля должен задерживать переход ds★ на большее время, чем требуется в 6.2.2.1.4, чтобы выполнить это условие.

В принудительном режиме сигналы d[]★ должны быть стабильными при выводе на магистраль перед тем, как переход dk★ или di★ покажет их действительность, если передают один или более исполнителей. Если необходимо, разработчик модуля должен задерживать переход ds★ или di★ на большее время, чем требуется в 6.2.2.1.5 или 6.2.2.1.6, чтобы выполнить это условие.

В принудительном режиме, где ЗАПИСЬ_ИСПОЛНИТЕЛЯ | ШИРОКОВЕЩАТЕЛЬНЫЙ_СТАТУС, сигналы d[]★ должны быть стабильными при выводе на магистраль перед тем, как выставление dk★ или di★ покажет их действительность, если передают один или более исполнителей. Если необходимо, разработчик модуля должен задерживать установление dk★ или di★ на большее время, чем требуется в 6.2.2.1.5 или 6.2.2.1.6, чтобы выполнить это условие.

6.2.2.5.3 **BP[31 . . . 0]★ (Четность магистрали)**

Модули должны снимать bp[]★, если ИНИЦ.

Определение разрядов четности должно быть следующим:

БАЙТОВАЯ ШИНА	СИГНАЛ ЧЕТНОСТИ
D[255 . . . 248]★	BP31★
D[247 . . 240]★	BP30★
D[239 . . . 232]★	BP29★
D[231 . . . 224]★	BP28★
D[223 . . . 216]★	BP27★
D[215 . . . 208]★	BP26★
D[207 . . . 200]★	BP25★
D[199 . . . 192]★	BP24★
D[191 . . . 184]★	BP23★
D[183 . . 176]★	BP22★
D[175 . . . 168]★	BP21★
D[167 . . . 160]★	BP20★
D[159 . . . 152]★	BP19★
D[151 . . . 144]★	BP18★
D[143 . . . 136]★	BP17★
D[135 . . . 128]★	BP16★
D[127 . . . 120]★	BP15★
D[119 . . . 112]★	BP14★
D[111 . . . 104]★	BP13★
D[103 . . . 96]★	BP12★
D[95 . . . 88]★	BP11★
D[87 . . . 80]★	BP10★
D[79 . . . 72]★	BP9★
D[71 . . . 64]★	BP8★
AD[63 . . . 56]★	BP7★
AD[55 . . . 48]★	BP6★
AD[47 . . . 40]★	BP5★
AD[39 . . . 32]★	BP4★
AD[31 . . 24]★	BP3★
AD[23 . . . 16]★	BP2★
AD[15 . . . 8]★	BP1★
AD[7 . . . 0]★	BP0★

Если модуль выставляет четное число разрядов в байтовой шине, определенной в левой колонке таблицы, он должен установить разряд четности, определенный в правой колонке.

Байтовые шины, не активные при AW★ или DW[]★ командах, не должны активировать соответствующий им сигнал четности.

Во время первого обмена передачи, когда -ИНИЦ & ФАЗА_ДАННЫХ & ЧАСТНЫЙ, модули должны использовать DW[]★ команду, чтобы выбрать байты с активированной четностью.

Во время фазы рассоединения, когда ЗАПИСЬ_СМД, ЗАДАТЧИК должен активировать правильную четность для AD[31 . . 0]★.

Если выбрана передача с использованием пакетного протокола, сигналы на bp[]★ должны подчиняться пакетному протоколу в соответствии с 6.2.3.21.

Сигналы bp[]★ должны быть стабильными при выводе на магистраль до того, как переход as★ покажет их действительность. Если необходимо, разработчик модуля должен задержать переход as★ на большее время, чем требуется в 6.2.2.1.1, чтобы выполнить это условие.

В принудительном режиме сигналы bp[]★ должны быть стабильными при выводе на магистраль до того, как переход ds★ покажет их действительность, если передает ЗАДАТЧИК. Если необходимо, разработчик модуля должен задержать переход ds★ на большее время, чем требуется в 6.2.2.1.4, чтобы выполнить это условие.

В принудительном режиме сигналы bp[]★ должны быть стабильными при выводе на магистраль до того, как переход dk★ или di★ покажет их действительность, если передают один или несколько исполнителей. Если необходимо, разработчик модуля должен задержать переход dk★ или di★ на большее время, чем требуется в 6.2.2.1.5 или 6.2.2.1.6, чтобы выполнить это условие.

В принудительных событиях с данными, где ЗАПИСЬ_ИСПОЛНИТЕЛЯ ¦ ШИРОКОВЕЩАТЕЛЬНЫЙ_СТАТУС, сигналы dp[]★ должны быть стабильными при выводе на магистраль до того, как выставление dk★ или di★ покажет их действительность, если передают один или несколько исполнителей. Если необходимо, разработчик модуля должен задержать выставление dk★ или di★ на большее время, чем требуется в 6.2.2.1.5 или 6.2.2.1.6, чтобы выполнить это условие.

6.2.2.5.4 **TG[7 . . . 0]★ (Ter)**

Сигналы tg[7 . . .0]★ должны следовать той же временной диаграмме и тем же особенностям, что и ad[31 0]★, за исключением того, что на усмотрение разработчика системы оставляется выбор, когда они должны нести информацию.

Число активных сигналов tg[]★ будет описано в 7.2.4.

6.2.2.5.5 **TP★ (Четность тега)**

Если установлено РАЗРЕШЕНИЕ_ТЕГА и модуль выставляет четное число tg[]★, он должен выставить tp★.

6.2.2.5.6 **ET★**

Модули должны снимать et★, если ИНИЦ.

ЗАДАТЧИК должен выставить et★, когда он обнаружит, что Alf стал логическим нулем и перед снятием as★ во время первой передачи его текущего владения. ЗАДАТЧИК должен удерживать et★ выставленным до тех пор, пока -rq1★ & -rq0★ & -AKf, и он не снял ad[]★, d[]★, bp[]★, tg[]★, tp★, cm[]★ и cp★.

6.2.3 **Определения протокола**

6.2.3.1 **Фаза соединения задатчика**

1) ЗАДАТЧИК должен установить атрибуты, определенные в 6.2.1.1, 6.2.1.2 и 6.2.1.4.

2) ЗАДАТЧИК должен сбросить ТАЙМ-АУТ_ПЕРЕДАЧИ.

3) ЗАДАТЧИК должен ждать, пока -AKf & AI★ ¦ ТАЙМ-АУТ_ПЕРЕДАЧИ.

4) Если установлен ТАЙМ-АУТ_ПЕРЕДАЧИ, ЗАДАТЧИК должен перейти к восстановлению тайм-аута передачи в соответствии с 6.2.3.24.

5) ЗАДАТЧИК должен активировать ad[]★, bp[]★, tg[]★, tp★, cm[]★, cp[]★ и ca[]★ соответствующей информацией.

6) ЗАДАТЧИК должен установить НАЧАЛО_ПЕРЕДАЧИ.

7) ЗАДАТЧИК должен выставить as★ и ak★.

8) ЗАДАТЧИК должен снять ai★.

9) ЗАДАТЧИК должен ждать, пока -Alf ¦ ТАЙМ-АУТ_ПЕРЕДАЧИ.

10) Если установлен ТАЙМ-АУТ_ПЕРЕДАЧИ, ЗАДАТЧИК должен перейти к восстановлению тайм-аута передачи в соответствии с 6.2.3.24.

11) ЗАДАТЧИК должен снять tc★ и выставить et★.

12) ЗАДАТЧИК должен установить атрибуты, определенные в 6.2.1.7.1.

13) Если НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ, ЗАДАТЧИК должен перейти к фазе рассоединения, как определено в 6.2.3.22.

14) Если СТАТУС_ПРИНУЖДЕНИЯ & (НЕБЛОКИРОВАНИЕ_ЧТЕНИЕ | НЕБЛОКИРОВАНИЕ_ЗАПИСЬ | БЛОКИРОВАНИЕ_ЧТЕНИЕ | БЛОКИРОВАНИЕ_ЗАПИСЬ | ЧАСТНОЕ_ЧТЕНИЕ | ЧАСТНАЯ_ЗАПИСЬ | БЛОКИРОВАНИЕ_ЧАСТНОЕ_ЧТЕНИЕ | БЛОКИРОВАНИЕ_ЧАСТНАЯ_ЗАПИСЬ | ОТВЕТ_ЧТЕНИЯ | РАЗДЕЛЕННЫЙ_ОТВЕТ | МОДИФИЦИРОВАННЫЙ_ОТВЕТ | ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ | НЕДЕЙСТВИТЕЛЬНОСТЬ_ЧТЕНИЯ | НЕДЕЙСТВИТЕЛЬНОСТЬ_ЗАПИСИ | РАЗДЕЛЕННОЕ_ЧТЕНИЕ | КОПИРОВАНИЕ_НАЗАД | МОДИФИКАЦИЯ_ЧТЕНИЯ), ЗАДАТЧИК может перейти к нечетному обмену принудительной фазы данных, как определено в 6.2.3.3.

15) Если -МНОЖЕСТВЕННЫЙ_ПАКЕТНЫЙ_РЕЖИМ & (НЕБЛОКИРОВАНИЕ_ЧТЕНИЕ | НЕБЛОКИРОВАНИЕ_ЗАПИСЬ | ОТВЕТ_ЧТЕНИЯ | ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ | НЕДЕЙСТВИТЕЛЬНОСТЬ_ЧТЕНИЯ | НЕДЕЙСТВИТЕЛЬНОСТЬ_ЗАПИСИ | РАЗДЕЛЕННОЕ_ЧТЕНИЕ | КОПИРОВАНИЕ_НАЗАД | РАЗДЕЛЕННЫЙ_ОТВЕТ | МОДИФИЦИРОВАННЫЙ_ОТВЕТ | МОДИФИКАЦИЯ_ЧТЕНИЯ), ЗАДАТЧИК может перейти к одиночной пакетной фазе данных, как определено в 6.2.3.13.

16) Если МНОЖЕСТВЕННЫЙ_ПАКЕТНЫЙ_РЕЖИМ & (НЕБЛОКИРОВАНИЕ_ЧТЕНИЕ | НЕБЛОКИРОВАНИЕ_ЗАПИСЬ | ОТВЕТ_ЧТЕНИЯ | ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ | НЕДЕЙСТВИТЕЛЬНОСТЬ_ЧТЕНИЯ | НЕДЕЙСТВИТЕЛЬНОСТЬ_ЗАПИСИ | РАЗДЕЛЕННОЕ_ЧТЕНИЕ | КОПИРОВАНИЕ_НАЗАД | РАЗДЕЛЕННЫЙ_ОТВЕТ | МОДИФИЦИРОВАННЫЙ_ОТВЕТ | МОДИФИКАЦИЯ_ЧТЕНИЯ), ЗАДАТЧИК может перейти к множественной пакетной фазе данных, как определено в 6.2.3.13.

17) ЗАДАТЧИК может перейти к фазе рассоединения, как определено в 6.2.3.22.

6.2.3.2 **Фаза соединения исполнителя**

Исполнитель должен ждать до тех пор, пока он обнаружит выставление AS★, а затем должен:

1) выставить ak★, если -ГОТОВ_ДЛЯ_ВЫНИМАНИЯ,

2) проанализировать AD[]★, BP[]★, TG[]★, TP★, CM[]★ и CP★,

3) ожидать, пока ДЕКОДИРОВАННЫЙ_АДРЕС,

4) активировать st[7 . . . 1]★ и ca[2 . . . 0]★ соответствующей информацией,

5) если БЛОКИРОВАНИЕ, установить БЛОКИРОВАНИЕ_РЕСУРСОВ в соответствии с адресом,

6) выставить di★, если УЧАСТИЕ,

7) затем снять ai★,

8) затем ожидать до СТАТУС_СОЕДИНЕНИЯ,

9) затем снять te★,

10) проанализировать информацию на CA[]★,

11) если -УЧАСТИЕ, ожидать до ФАЗА_РАССОЕДИНЕНИЯ и перейти к фазе рассоединения в соответствии с 6.2.3.23,

12) ожидать, пока ФАЗА_ДАННЫХ | ФАЗА_РАССОЕДИНЕНИЯ,

13) если ФАЗА_ДАННЫХ & СТАТУС_ПРИНУЖДЕНИЯ, перейти к нечетному событию, как описано в 6.2.3.4,

14) если ФАЗА_ДАННЫХ & -СТАТУС_ПРИНУЖДЕНИЯ & РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ, перейти к нечетному пакету, как описано в 6.2.3.14,

15) если ФАЗА_ДАННЫХ & РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ, перейти к нечетному пакету, как описано в 6.2.3.16,

16) перейти к фазе рассоединения, как описано в 6.2.3.23.

6.2.3.3 **Принудительная фаза данных задатчика — первый нечетный обмен**

1) ЗАДАТЧИК должен активировать cm[]★ и cp★ соответствующей информацией.

2) Если ЧАСТНЫЙ, ЗАДАТЧИК должен активировать ad[f]★ соответствующей информацией невыбранной шины и затем перейти к 7).

3) Если МАСКИРОВАНИЕ_ОБМЕН, ЗАДАТЧИК должен активировать ad[]★ и d[]★ соответствующим операндом маски и затем перейти к 7).

4) Если СРАВНЕНИЕ_ОБМЕН, ЗАДАТЧИК должен активировать ad[]★ и d[]★ соответствующим операндом сравнения и затем перейти к 7).

5) Если (ВЫБОРКА_СЛОЖЕНИЕ_СО_СТАРШЕГО ¦ ВЫБОРКА_СЛОЖЕНИЕ_С_МЛАДШЕГО), ЗАДАТЧИК должен активировать ad[]★ и d[]★ соответствующим операндом сложения и затем перейти к 7).

6) Если ЗАПИСЬ_СМД, ЗАДАТЧИК должен активировать ad[]★ и d[]★ соответствующими данными и затем перейти к 7). Если -ЗАПИСЬ_СМД, проследовать к 8).

7) ЗАДАТЧИК должен активировать bp[]★, tg[]★ и tp★ соответствующей информацией и затем перейти к 9).

8) Если -ЗАПИСЬ_СМД, ЗАДАТЧИК должен снять ad[]★, d[]★, bp[]★, tg[]★ и tp★.

9) Затем ЗАДАТЧИК должен установить ds★.

10) ЗАДАТЧИК должен ждать, пока ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН ¦ ТАЙМ-АУТ_ПЕРЕДАЧИ.

11) Если установлен ТАЙМ-АУТ_ПЕРЕДАЧИ, ЗАДАТЧИК должен перейти к восстановлению тайм-аута передачи в соответствии с 6.2.3.24.

12) Если -ЧАСТНЫЙ & -ЗАПИСЬ_СМД, ЗАДАТЧИК должен проанализировать информацию на AD[]★, D[]★, BP[]★, TG[]★ и TP★.

13) Если -ЧАСТНЫЙ & (ВЫБОРКА_СЛОЖЕНИЕ_СО_СТАРШЕГО ¦ ВЫБОРКА_СЛОЖЕНИЕ_С_МЛАДШЕГО) ¦ РАСЩЕПЛЕННЫЙ_СТАТУС & ЧАСТНЫЙ & -ЗАПИСЬ_СМД, ЗАДАТЧИК должен перейти к фазе рассоединения, как описано в 6.2.3.22.

14) ЗАДАТЧИК должен перейти к четному обмену, как описано в 6 2.3.5, если -НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ, или к фазе рассоединения, как описано в 6.2.3.22, если НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ.

6.2.3.4 **Принудительная фаза данных исполнителя — первый нечетный обмен**

Исполнитель должен.

1) если -СТАТУС_ЗАПИСИ & (ВЫБРАННЫЙ & ЗАПИСЬ_ИСПОЛНИТЕЛЯ ' ШИРОКОВЕЩАНИЕ ¦ ЛОВЯЩИЙ_ДАННЫЕ), ждать, пока выставится DK★, выставить dk★, проанализировать данные на AD[]★, D[]★ и перейти к 10);

2) если -СТАТУС_ЗАПИСИ & (ВНЕДРЕННОСТЬ ¦ ВЫБРАННЫЙ & ШИРОКОВЕЩАТЕЛЬНЫЙ_СТАТУС), активировать ad[]★, d[]★, bp[]★, tg[]★ и tp★ соответствующей информацией, выставить dk★ и перейти к 11);

3) выставить dk★,

4) если -СТАТУС ЗАПИСИ & ВЫБРАННЫЙ, активировать ad[]★, d[]★, bp[]★, tg[]★ и tp★ соответствующей информацией и перейти к 11),

5) если ЧАСТНЫЙ & ВЫБРАННЫЙ, проанализировать информацию невыбора шины на AD[31 . . . 0]★ и перейти к 10),

6) если МАСКИРОВАНИЕ_ОБМЕН & ВЫБРАННЫЙ, проанализировать операнд маски на AD[]★, D[]★ и перейти к 10),

7) если СРАВНЕНИЕ_ОБМЕН & ВЫБРАННЫЙ, проанализировать операнд сравнения на AD[]★, D[]★ и перейти к 10),

8) если (ВЫБОРКА_СЛОЖЕНИЕ_СО_СТАРШЕГО ¦ ВЫБОРКА_СЛОЖЕНИЕ_С_МЛАДШЕГО) & ВЫБРАННЫЙ, проанализировать операнд сложения на AD[]★, D[]★ и перейти к 10),

9) если СТАТУС_ЗАПИСИ & (ВЫБРАННЫЙ ¦ ШИРОКОВЕЩАНИЕ ¦ ЛОВЯЩИЙ_ДАННЫЕ), проанализировать данные на AD[]★ и D[]★,

10) проанализировать информацию на BP★, TG[]★ и TP★,

11) активировать st6★, st1★ и st0★ соответствующим статусом,

12) затем снять di★,

13) ждать, пока -AS★¦ -DS★,

14) если ФАЗА_РАССОЕДИНЕНИЯ, снять ad[]★, d[]★, bp[]★, tg[]★ и tp★ и перейти к фазе рассоединения, как описано в 6.2.3.23,

15) если ФАЗА_ДАННЫХ, перейти к четному обмену, как описано в 6.2.3.6.

6.2.3.5 **Принудительная фаза данных задатчика — первый четный обмен**

1) Если ЧАСТНЫЙ & МАСКИРОВАНИЕ_ОБМЕН, ЗАДАТЧИК должен активировать ad[]★ соответствующим операндом маски и перейти к 7).

2) Если ЧАСТНЫЙ & СРАВНЕНИЕ_ОБМЕН, ЗАДАТЧИК должен активировать ad[]★ соответствующим операндом сравнения и перейти к 7).

3) Если ЧАСТНЫЙ & (ВЫБОРКА_СЛОЖЕНИЕ_СО_СТАРШЕГО ¦ ВЫБОРКА_СЛОЖЕНИЕ_С_МЛАДШЕГО), ЗАДАТЧИК должен активировать ad[]★ соответствующим операндом сложения и перейти к 7).

4) Если СРАВНЕНИЕ_ОБМЕН, ЗАДАТЧИК должен активировать ad[]★ соответствующим операндом обмена и перейти к 7).

5) Если ЗАПИСЬ_СМД, ЗАДАТЧИК должен активировать ad[]★, d[]★ и bp[]★ соответствующими данными и перейти к 7).

6) Если -ЗАПИСЬ_СМД, ЗАДАТЧИК должен снять ad[]★, d[]★, bp[]★, tg[]★, tp★ и перейти к 8).

7) ЗАДАТЧИК должен активировать tg[]★ и tp★ соответствующей информацией.

8) Затем ЗАДАТЧИК должен снять ds★.

9) ЗАДАТЧИК должен ждать, пока ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН ¦ ТАЙМ-АУТ_ПЕРЕДАЧИ.

10) Если установлен ТАЙМ-АУТ_ПЕРЕДАЧИ, ЗАДАТЧИК должен перейти к восстановлению тайм-аута передачи, как описано в 6.2.3.24.

11) Если -ЗАПИСЬ_СМД, ЗАДАТЧИК должен проанализировать информацию на AD[]★, D[]★, BP[]★, TG[]★ и TP★.

12) Если ЧАСТНЫЙ & (ВЫБОРКА_СЛОЖЕНИЕ_СО_СТАРШЕГО ¦ ВЫБОРКА_СЛОЖЕНИЕ_С_МЛАДШЕГО) ¦ -ЧАСТНЫЙ & (МАСКИРОВАНИЕ_ОБМЕН ¦ СРАВНЕНИЕ_ОБМЕН), ЗАДАТЧИК должен перейти к фазе рассоединения, как описано в 6.2.3.22.

13) ЗАДАТЧИК должен перейти к нечетному обмену, как описано в 6.2.3.7, если -НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ, или к фазе рассоединения, как описано в 6.2.3.22, если НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ.

6.2.3.6 **Принудительная фаза данных исполнителя — первый четный обмен**

Исполнитель должен:

1) если —СТАТУС_ЗАПИСИ & (ВЫБРАННЫЙ & ЗАПИСЬ_ИСПОЛНИТЕЛЯ ¦ ШИРОКОВЕЩАНИЕ ¦ ЛОВЯЩИЙ_ДАННЫЕ), ждать выставления DI★, выставить di★, проанализировать данные на AD[]★, D[]★ и перейти к 10),

2) если —СТАТУС_ЗАПИСИ & (ВНЕДРЕННОСТЬ ¦ ВЫБРАННЫЙ & ШИРОКОВЕЩАТЕЛЬНЫЙ_СТАТУС), активировать ad[]★, d[]★, bp[]★, tg[]★ и tp★ соответствующей информацией и перейти к 11),

3) выставить di★,

4) если —СТАТУС_ЗАПИСИ & ВЫБРАННЫЙ, активировать ad[]★, d[]★, bp[]★, tg[]★ и tp★ соответствующей информацией и перейти к 11),

5) если ЧАСТНЫЙ & МАСКИРОВАННЫЙ_ОБМЕН & ВЫБРАННЫЙ, проанализировать операнд маски на AD[]★, D[]★ и перейти к 10),

6) если ЧАСТНЫЙ & СРАВНЕНИЕ_ОБМЕН & ВЫБРАННЫЙ, проанализировать операнд сравнения на AD[]★, D[]★ и перейти к 10),

7) если ЧАСТНЫЙ & (ВЫБОРКА_СЛОЖЕНИЕ_СО_СТАРШЕГО ¦ ВЫБОРКА_СЛОЖЕНИЕ_С_МЛАДШЕГО) & ВЫБРАННЫЙ, проанализировать операнд сложения на AD[]★, D[]★ и перейти к 10),

8) если МАСКИРОВАНИЕ_ОБМЕН ¦ СРАВНЕНИЕ_ОБМЕН, проанализировать операнд обмена на AD[]★, D[]★ и перейти к 10),

9) если СТАТУС_ЗАПИСИ & (ВЫБРАННЫЙ ¦ ШИРОКОВЕЩАНИЕ ¦ ЛОВЯЩИЙ_ДАННЫЕ), проанализировать данные на AD[]★, D[]★,

10) проанализировать информацию на BP[]★, TG[]★ и TP★,

11) активировать st6★, st1★ и st0★ соответствующим статусом,

12) затем снять dk★,

13) ждать, пока —AS★ ¦ DS★,

14) если ФАЗА_РАССОЕДИНЕНИЯ, снять ad[]★, d[]★, bp[]★, tg[]★, tp★ и перейти к фазе рассоединения, как описано в 6.2.3.23,

15) если ФАЗА_ДАННЫХ, перейти к нечетному обмену, как описано в 6.2.3.8.

6.2.3.7 **Принудительная фаза данных задатчика — второй нечетный обмен**

1) Если ЧАСТНЫЙ & (МАСКИРОВАНИЕ_ОБМЕН & СРАВНЕНИЕ_ОБМЕН), ЗАДАТЧИК должен активировать ad[]★ соответствующим операндом обмена и перейти к 4).

2) Если ЗАПИСЬ_СМД, ЗАДАТЧИК должен активировать ad[]★, d[]★ и bp[]★ соответствующими данными и перейти к 4).

3) Если -ЗАПИСЬ_СМД, ЗАДАТЧИК должен снять ad[]★, d[]★, bp[]★, tg[]★, tp★ и перейти к 5).

4) ЗАДАТЧИК должен активировать tg[]★ и tp★ соответствующей информацией.

5) Затем ЗАДАТЧИК должен снять ds★.

6) ЗАДАТЧИК должен ждать, пока ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН ¦ ТАЙМ-АУТ_ПЕРЕДАЧИ.

7) Если установлен ТАЙМ-АУТ_ПЕРЕДАЧИ, ЗАДАТЧИК должен перейти к восстановлению тайм-аута передачи, как описано в 6.2.3.24.

8) Если -ЗАПИСЬ_СМД, ЗАДАТЧИК должен проанализировать информацию на AD[]★, D[]★, BP[]★, TG[]★ и TP★.

9) Если ЧАСТНЫЙ & (МАСКИРОВАНИЕ_ОБМЕН ¦ СРАВНЕНИЕ_ОБМЕН), ЗАДАТЧИК должен перейти к фазе рассоединения, как описано в 6.2.3.22.

10) ЗАДАТЧИК должен перейти к четному обмену, как описано в 6.2.3.11, если -НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ, или к фазе рассоединения, как описано в 6.2.3.22, если НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ.

6.2.3.8 **Принудительная фаза данных исполнителя — второй нечетный обмен**

Исполнитель должен:

1) если -СТАТУС_ЗАПИСИ & (ВЫБРАННЫЙ & ЗАПИСЬ_ИСПОЛНИТЕЛЯ ¦ ШИРОКОВЕЩАНИЕ ¦ ЛОВЯЩИЙ_ДАННЫЕ), ждать выставления DK★, проанализировать данные на AD[]★, D[]★ и перейти к 7),

2) если -СТАТУС_ЗАПИСИ & (ВНЕДРЕННОСТЬ ¦ ВЫБРАННЫЙ & ШИРОКОВЕЩАТЕЛЬНЫЙ_СТАТУС), активировать ad[]★, d[]★, bp[]★, tg[]★ и tp★ соответствующей информацией и затем выставить dk★ и перейти к 11),

3) выставить dk★,

4) если -СТАТУС_ЗАПИСИ & ВЫБРАННЫЙ, активировать ad[]★, d[]★, bp[]★, tg[]★ и tp★ соответствующей информацией и перейти к 8),

5) если ЧАСТНЫЙ & (МАСКИРОВАНИЕ_ОБМЕН & СРАВНЕНИЕ_ОБМЕН) & ВЫБРАННЫЙ, проанализировать операнд обмена на AD[]★, D[]★ и перейти к 8),

6) если СТАТУС_ЗАПИСИ & (ВЫБРАННЫЙ ¦ ШИРОКОВЕЩАНИЕ ¦ ЛОВЯЩИЙ_ДАННЫЕ), проанализировать данные на AD[]★ и D[]★,

7) проанализировать информацию на BP[]★, TG[]★ и TP★,

8) активировать st6★, st1★ и st0★ соответствующим статусом,

9) затем снять di★,

10) ждать, пока -AS★ ¦ -DS★,

11) если ФАЗА_РАССОЕДИНЕНИЯ, снять ad[]★, d[]★, bp[]★, tg[]★, tp★ и перейти к фазе рассоединения, как описано в 6.2.3.23,

12) если ФАЗА_ДАННЫХ, перейти к четному событию, как описано в 6.2.3.12.

6.2.3.9 **Принудительная фаза данных задатчика — третий и последующие нечетные обмены**

1) Если ЗАПИСЬ_СМД, ЗАДАТЧИК должен активировать ad[]★, d[]★, bp[]★, tg[]★ и tp★ соответствующей информацией и перейти к 3).

2) Если -ЗАПИСЬ_СМД, ЗАДАТЧИК должен снять ad[f]★, d[f]★, bp[]★, tg[]★ и tp★.

3) Затем ЗАДАТЧИК должен установить ds★.

4) ЗАДАТЧИК должен ждать, пока ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН ¦ ТАЙМ-АУТ_ПЕРЕДАЧИ.

5) Если установлен ТАЙМ-АУТ_ПЕРЕДАЧИ, ЗАДАТЧИК должен перейти к восстановлению тайм-аута передачи, как описано в 6.2.3.24.

6) Если -ЗАПИСЬ_СМД, ЗАДАТЧИК должен проанализировать информацию на AD[]★, D[]★, BP[]★, TG[]★ и TP★.

7) ЗАДАТЧИК должен перейти к четному обмену, как описано в 6.2.3.11, если -НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ, или к фазе рассоединения, как описано в 6.2.3.22, если НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ.

6.2.3.10 **Принудительная фаза данных исполнителя — третий и последующие нечетные обмены**

Исполнитель должен:

1) если -СТАТУС_ЗАПИСИ & (ВЫБРАННЫЙ & ЗАПИСЬ_ИСПОЛНИТЕЛЯ ¦ ШИРОКОВЕЩАНИЕ ¦ ЛОВЯЩИЙ_ДАННЫЕ), ждать выставления DK★, проанализировать данные на AD[]★ и D[]★ и перейти к 6),

2) если -СТАТУС_ЗАПИСИ & (ВНЕДРЕННОСТЬ ¦ ВЫБРАННЫЙ & ШИРОКОВЕЩАТЕЛЬНЫЙ_СТАТУС), активировать ad[]★, d[]★, bp[]★, tg[]★ и tp★ соответствующей информацией, выставить dk★ и перейти к 7),

3) выставить dk★,

4) если -СТАТУС_ЗАПИСИ & ВЫБРАННЫЙ, активировать ad[]★, d[]★, bp[]★, tg[]★ и tp★ соответствующей информацией и перейти к 7),

5) если СТАТУС_ЗАПИСИ & (ВЫБРАННЫЙ ¦ ШИРОКОВЕЩАНИЕ ¦ ЛОВЯЩИЙ_ДАННЫЕ), проанализировать данные на AD[]★ и D[]★,

6) проанализировать информацию на BP[]★, TG[]★ и TP★,

7) активировать st6★, st1★ и st0★ соответствующим статусом,

8) затем снять di★,

9) ждать, пока -AS★ ¦ -DS★,

10) если ФАЗА_РАССОЕДИНЕНИЯ, снять ad[]★, d[]★, bp[]★, tg[]★, tp★ и перейти к фазе рассоединения, как описано в 6.2.3.23,

11) если ФАЗА_ДАННЫХ, перейти к четному обмену, как описано в 6.2.3.12.

6.2.3.11 **Принудительная фаза данных задатчика — второй и последующие четные обмены**

1) Если ЗАПИСЬ_СМД, ЗАДАТЧИК должен активировать ad[]★, d[]★, bp[]★, tg[]★ и tp★ соответствующей информацией и перейти к 3).

2) Если -ЗАПИСЬ_СМД, ЗАДАТЧИК должен снять ad[]★, d[]★, bp[]★, tg[]★ и tp★.

3) Затем ЗАДАТЧИК должен снять ds★.

4) ЗАДАТЧИК должен ждать, пока ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН ¦ ТАЙМ-АУТ_ПЕРЕДАЧИ.

5) Если установлен ТАЙМ-АУТ_ПЕРЕДАЧИ, ЗАДАТЧИК должен перейти к восстановлению тайм-аута передачи, как описано в 6.2.3.24.

6) Если -ЗАПИСЬ_СМД, ЗАДАТЧИК должен оценить информацию на AD[]★, D[]★, BP[]★, TG[]★ и TP★.

7) ЗАДАТЧИК должен перейти к нечетному обмену, как описано в 6.2.3.9, если -НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ, или к фазе рассоединения, как описано в 6.2.3.22, если НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ

6.2.3.12 **Принудительная фаза данных исполнителя — второй и последующие четные обмены**

Исполнитель должен:

1) если -СТАТУС_ЗАПИСИ & (ВЫБРАННЫЙ & ЗАПИСЬ_ИСПОЛНИТЕЛЯ ¦ ШИРОКОВЕЩАНИЕ ¦ ЛОВЯЩИЙ_ДАННЫЕ), ждать выставления DI★, проанализировать данные на AD[]★, D[]★ и перейти к 6),

2) если -СТАТУС_ЗАПИСИ & (ВНЕДРЕННОСТЬ ¦ ВЫБРАННЫЙ & ШИРОКОВЕЩАТЕЛЬНЫЙ_СТАТУС), активировать ad[]★, d[]★, bp[]★, tg[]★ и tp★ соответствующей информацией, выставить di★ и перейти к 7),

3) выставить di★,

4) если -СТАТУС_ЗАПИСИ & ВЫБРАННЫЙ, активировать ad[]★, d[]★, bp[]★, tg[]★ и tp★ соответствующей информацией и перейти к 7),

5) если СТАТУС_ЗАПИСИ & (ВЫБРАННЫЙ , ШИРОКОВЕЩАНИЕ ¦ ЛОВЯЩИЙ_ДАННЫЕ), проанализировать данные на AD[]★ и D[]★,

6) проанализировать информацию на BP[]★, TG[]★ и TP★,

7) активировать st6★, st1★ и st0★ соответствующим статусом,

8) затем снять dk★,

9) ждать, пока -AS★ ¦ -DS★,

10) если ФАЗА_РАССОЕДИНЕНИЯ, снять ad[]★, d[]★, bp[]★, tg[]★, tp★ и перейти к фазе рассоединения, как описано в 6.2.3.23,

11) если ФАЗА_ДАННЫХ, перейти к нечетному обмену, как описано в 6.2.3.10.

6.2.3.13 **Пакетная фаза данных задатчика — одиночный пакетный режим**

1) Задатчик должен активировать cm[]★ и cp★ соответствующей информацией

2) Если -ЗАПИСЬ_СМД, ЗАДАТЧИК должен снять ad[]★, d[]★, bp[]★, tg[]★ и tp★.

3) Затем ЗАДАТЧИК должен выставить ds★.

4) Если ЗАПИСЬ_СМД, ЗАДАТЧИК должен ждать минимум один битовый период и затем активировать ad[]★, d[]★, bp[]★, tg[]★ и tp★ соответствующей пакетной передачей и, когда будет передан бит четности, перейти к 8).

5) Если -ЗАПИСЬ_СМД, ЗАДАТЧИК должен проанализировать пакетную передачу на AD[]★, D[]★, BP[]★, TG[]★, TP★ и, когда будет принят бит четности, перейти к 6).

6) ЗАДАТЧИК должен ждать, пока ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН ¦ ТАЙМ-АУТ_ПЕРЕДАЧИ.

7) Если установлен ТАЙМ-АУТ_ПЕРЕДАЧИ, ЗАДАТЧИК должен перейти к восстановлению тайм-аута передачи, как описано в 6.2.3.24.

8) ЗАДАТЧИК должен перейти к фазе рассоединения, как описано в 6.2.3.22.

6.2.3.14 **Пакетная фаза данных исполнителя — одиночный пакетный режим**

Исполнитель должен:

1) выставить dk★, активировать st1★ и st0★ соответствующим статусом и затем снять di★. Завершение 1) может быть продлено до начала 7),

2) если СТАТУС_ЗАПИСИ, проанализировать пакетную передачу на AD[]★, D[]★, BP[]★, TG[]★, TP★ и перейти к 7),

3) если -СТАТУС_ЗАПИСИ & (ВЫБРАННЫЙ & ЗАПИСЬ_ИСПОЛНИТЕЛЯ ¦ ШИРОКОВЕЩАНИЕ), проанализировать пакетную передачу на AD[]★, D[]★, BP[]★, TG[]★, TP★ и перейти к 7),

4) если ВЫБРАННЫЙ & -СТАТУС_ЗАПИСИ & -ЗАПИСЬ_ИСПОЛНИТЕЛЯ, активировать ad[]★, d[]★, bp[]★ tg[]★ и tp★ соответствующей пакетной передачей и перейти к 6),

5) если -СТАТУС_ЗАПИСИ & ВНЕДРЕННОСТЬ, активировать ad[]★, d[]★, bp[]★, tg[]★ и tp★ соответствующей пакетной передачей и перейти к 6),

6) снять ad[]★, d[]★, bp[]★ tg[]★ и tp★,

7) затем ждать, пока -AS★, и перейти к фазе рассоединения, как описано в 6.2.3.23.

6.2.3.15 **Пакетная фаза данных задатчика — режим многократных пакетов — нечетная пакетная очередь**

1) Задатчик должен активировать cm[]★ и cp★ соответствующей информацией.

2) Если –ЗАПИСЬ_СМД, ЗАДАТЧИК должен снять ad[]★, d[]★, bp[]★, tg[]★, tp★ и приготовить очередь для принятия пакета.

3) Затем ЗАДАТЧИК должен выставить ds★.

4) Если ЗАПИСЬ_СМД, ЗАДАТЧИК должен поставить в очередь соответствующий пакет для передачи.

5) ЗАДАТЧИК должен ждать, пока ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН ¦ ТАЙМ-АУТ_ПЕРЕДАЧИ.

6) Если установлен ТАЙМ-АУТ_ПЕРЕДАЧИ, ЗАДАТЧИК должен перейти к восстановлению тайм-аута передачи, как описано в 6.2.3.24.

7) ЗАДАТЧИК должен перейти к четному пакету в очереди, как описано в 6.2.3.17, если -НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ, или к фазе рассоединения, как описано в 6.2.3.22, если НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ.

6.2.3.16 **Пакетная фаза данных исполнителя — режим многократных пакетов — нечетная пакетная очередь**

Исполнитель должен:

1) если СТАТУС_ЗАПИСИ, подготовить очередь для принятия пакета,

2) если ВЫБРАННЫЙ & -СТАТУС_ЗАПИСИ & -ЗАПИСЬ_ИСПОЛНИТЕЛЯ, поставить в очередь соответствующую пакетную передачу,

3) если -СТАТУС_ЗАПИСИ & ВНЕДРЕННОСТЬ, поставить в очередь соответствующую пакетную передачу,

4) выставить dk★, активировать st7★, st5★, st4★, st2★, st1★ и st0★ соответствующим статусом, затем снять di★,

5) ждать, пока -AS★¦ -DS★,

6) если ФАЗА_РАССОЕДИНЕНИЯ, ждать, пока не будет принят или передан бит четности последнего пакета в очереди, снять ad[]★, d[]★, bp[]★, tg[]★, tp★ и перейти к фазе рассоединения, как описано в 6.2.3.23,

7) если ФАЗА_ДАННЫХ, перейти к четному пакету в очереди, как описано в 6.2.3.18.

6.2.3.17 **Пакетная фаза данных задатчика — режим многократных пакетов — четная пакетная очередь**

1) Задатчик должен активировать cm[]★ и cp★ соответствующей информацией.

2) Если -ЗАПИСЬ_СМД, ЗАДАТЧИК должен снять ad[]★, d[]★, bp[]★, tg[]★, tp★ и приготовить очередь для принятия пакета.

3) Затем ЗАДАТЧИК должен снять ds★.

4) Если ЗАПИСЬ_СМД, ЗАДАТЧИК должен поставить в очередь соответствующую пакетную передачу.

5) ЗАДАТЧИК должен ждать, пока ПРИЗНАК_ДАННЫХ_ЗАФИКСИРОВАН ¦ ТАЙМ-АУТ_ПЕРЕДАЧИ.

6) Если установлен ТАЙМ-АУТ_ПЕРЕДАЧИ, ЗАДАТЧИК должен перейти к восстановлению тайм-аута передачи, как описано в 6.2.3.24.

7) ЗАДАТЧИК должен перейти к нечетному пакету в очереди, как описано в 6.2.3.15, если -НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ, или к фазе рассоединения, как описано в 6.2.3.22, если НУЛЬ_СЧЕТЧИКА_ПЕРЕДАЧ.

6.2.3.18 **Пакетная фаза данных исполнителя — режим многократных пакетов — четная пакетная очередь**

Исполнитель должен:

1) если СТАТУС_ЗАПИСИ, подготовить очередь для принятия пакета,

2) если ВЫБРАННЫЙ & -СТАТУС_ЗАПИСИ & -ЗАПИСЬ_ИСПОЛНИТЕЛЯ, поставить в очередь соответствующую пакетную передачу,

3) если -СТАТУС_ЗАПИСИ & ВНЕДРЕННОСТЬ, поставить в очередь соответствующую пакетную передачу,

4) выставить di★, активировать st7★, st5★, st4★, st2★, st1★ и st0★ соответствующим статусом, затем снять dk★,

5) ждать, пока -AS★¦ DS★,

6) если ФАЗА_РАССОЕДИНЕНИЯ, ждать, пока не будет принят или передан бит четности последнего пакета в очереди, снять ad[]★, d[]★, bp[]★, tg[]★, tp★ и перейти к фазе рассоединения, как описано в 6.2.3.23,

7) если ФАЗА_ДАННЫХ, перейти к нечетному пакету в очереди, как описано в 6.2.3.16.

6.2.3.19 **Пакетная фаза данных задатчика — режим многократных пакетов — пакетная передача**

ЗАДАТЧИК должен считать каждую передачу так, что он знает начало и конец каждого пакета.

Перед инициализацией передачи первого пакета ЗАДАТЧИК должен ждать по крайней мере один битовый период после начального выставления ds★.

Чтобы инициализировать передачу любого последовательного пакета, ЗАДАТЧИК должен:

1) ждать, когда он осуществит смену сигнального уровня ds★, после смены ds★ при запросе пакета, затем отметить готовность пакетного состояния,

2) если в данное время не передается пакет и существует только один готовый к передаче пакет, ждать по крайней мере один битовый период с момента последнего изменения ds★ уровня,

3) если существует пакет, передаваемый в настоящее время, или пакеты в очереди перед пакетом, готовым к передаче, ждать, пока не будет передан бит четности всех предыдущих пакетов в очереди.

6.2.3.20 **Пакетная фаза данных исполнителя — режим многократных пакетов — пакетная передача**

Все участвующие исполнители должны считать каждую передачу так, что каждый участник знает начало и конец каждого пакета.

Чтобы инициализировать передачу любого пакета, исполнитель должен:

1) если в данное время не передается пакет и существует только один готовый к передаче от любого модуля на логической магистрали пакет, исполнитель может начать передачу пакета немедленно,

2) если существует пакет, передаваемый в настоящее время, или пакеты в очереди перед ним, готовые к передаче по этой логической магистрали, исполнитель должен ждать, пока не будет передан бит четности всех предыдущих пакетов в очереди. Затем исполнитель может начать передавать новый пакет, когда пройдет по крайней мере один полный битовый период после приема исполнителем последнего бита четности.

Любой участвующий модуль может принимать любой выбранный пакет без какого-либо взаимодействия с протоколом, изложенным выше, если пакеты не являются частью когерентно разделенной памяти. Если пакеты являются частью когерентно разделенной памяти, модули должны активировать соответствующую статусную информацию во время подтверждения, следующего за запросом пакета.

6.2.3.21 **Пакетный протокол данных**

Каждое локальное тактирование должно иметь отклонение от номинальной частоты не хуже, чем 0,01% при всех условиях работы. Модуль сообщает о своей пакетной скорости, как описано в 7.2.4.14.

Данные кодируются, используя стандарт NRZ-1, где логическая единица представлена переходом в начале битового периода, а логический ноль представлен отсутствием такого перехода.

Модуль, посылающий данные, должен генерировать переходы таким образом, чтобы любой край сигнала на магистрали не отличался от номинального положения края более чем на 30% длительности битового периода. Номинальное положение сигнала определяется по отношению к начальному (или синхронизирующему) переходу от отсутствия сигнала к его наличию.

Пакет не должен начинаться, пока AD[]★, D[]★, BP[]★, TG[]★ и TP★ не окажутся снятыми в течение хотя бы одного битового периода. Чтобы начать передачу, по всем активным AD[]★, D[]★, BP[]★, TG[]★ и TP★ линиям посылается синхронизирующее слово, в котором установлены все активные линии. Синхрослово следует перед словами данных. Длина данных 2, 4, 8, 16, 32 или 64 может быть установлена, используя РУС(CSR)-механизм, и вызывается командным сигналом DL★. После того как данные переданы, должно быть послано продольное слово четности. Четность подсчитывается в единицах серийного битового базиса, включая биты синхронизации и четности. Пакет с правильной четностью всегда заканчивается переходом всех AD[]★, D[]★, BP[]★, TG[]★ и TP★ линий в снятое состояние.

6.2.3.22 **Фаза рассоединения задатчика**

1) ЗАДАТЧИК должен ждать, пока ФАЗА_РАССОЕДИНЕНИЯ ¦ТАЙМ-АУТ_ПЕРЕДАЧИ.

2) Если -СТАТУС_ПРИНУЖДЕНИЯ, ЗАДАТЧИК должен ждать, пока не будет передан бит четности последнего пакета в очереди или пока не будет установлен ТАЙМ-АУТ_ПЕРЕДАЧИ.

3) Если установлен ТАЙМ-АУТ_ПЕРЕДАЧИ, ЗАДАТЧИК должен перейти к восстановлению тайм-аута передачи, как описано в 6.2.3.24.

4) Если ЗАПИСЬ_СМД ¦СТАТУС_РАСЩЕПЛЕНИЯ, ЗАДАТЧИК должен активировать AD[]★ соответствующей информацией рассоединения.

5) Если установлены ЗАПИСЬ_СМД ¦ СТАТУС_РАСЩЕПЛЕНИЯ, ЗАДАТЧИК должен активировать tg[]★ и tp★ соответствующей информацией.

6) ЗАДАТЧИК должен активировать cm[]★ и cp★ соответствующей информацией.

7) ЗАДАТЧИК должен снять as★ и выставить ai★.

8) Если СТАТУС_ПРИНУЖДЕНИЯ & ds★, ЗАДАТЧИК должен ждать, пока -ШИРОКОВЕЩАТЕЛЬНЫЙ_СТАТУС & -DK★ ¦ ШИРОКОВЕЩАТЕЛЬНЫЙ_СТАТУС & —DKf ¦ ТАЙМ-АУТ_ПЕРЕДАЧИ.

9) Если установлен ТАЙМ-АУТ_ПЕРЕДАЧИ, ЗАДАТЧИК должен перейти к восстановлению тайм-аута передачи, как описано в 6.2.3.24.

10) ЗАДАТЧИК должен снять ds★ и ak★.

11) Ждать, пока -AKf.

12) Снять ad[]★, d[]★, bp[]★, tg[]★, tp★, cm[]★ и cp★.

13) Если TE★ снят, обновить соответствующую информацию о состоянии модуля.

14) Затем установить ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

15) Если ЗАДАТЧИК собирается выполнить другую передачу, перейти к фазе соединения, как описано в 6.2.3.1.

16) Снять et★.

6.2.3.23 **Фаза рассоединения исполнителя**

1) Если -СТАТУС_ПРИНУЖДЕНИЯ, ждать, пока не будет передан бит четности последнего пакета в очереди.

2) Если ЗАПИСЬ_СМД СТАТУС_РАСЩЕПЛЕНИЯ, исполнитель должен проанализировать AD[]★, BP[]★, TG[]★ и TP★.

3) Проанализировать CM[]★ и CP★.

4) Выставить ai★.

5) Если выставлен, снять dk★.
6) Если -УДЕРЖАНИЕ_БЛОКИРОВКИ, сбросить БЛОКИРОВАНИЕ_РЕСУРСОВ.
7) Если во время передачи была обнаружена ошибка, установить te★.
8) Снять ak★.
9) Ждать, пока КОНЕЦ_РАССОЕДИНЕНИЯ.
10) Снять st[7 . . . 1]★ и ca[]★.
11) Если ТЕ★ снят, обновить соответствующую информацию о состоянии модуля.
12) Затем установить ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ.

6.2.3.24 **Восстановление тайм-аута передачи**

Когда установлен ТАЙМ-АУТ_ПЕРЕДАЧИ, ЗАДАТЧИК должен выполнить операцию Инициализации Магистрали, как описано в 7.2.3.3. За избежание и исправление ошибок, вызывающих тайм-аут передачи, несет ответственность разработчик системы.

7 СИСТЕМНОЕ УПРАВЛЕНИЕ МАГИСТРАЛЬЮ

7.1 Описание

7.1.1 Управление магистралью

В таблице 7.1 приведены соотношения между продолжительностью выставления сигнала RE★ и последующими за этим событиями. Детали представлены в нижеследующих разделах.

Таблица 7.1 — **Соотношения событий и длительностей RE★**

Событие	Событие считается случившимся, если RE★ выставлен в течение:	Для инициализации события выставить RE★ в течение:
Включение питания	30 мс (внутреннее состояние, указывающее на включение питания)	100—200 мс (плюс некое самотестирование)
Системный сброс	30—500 мс	100—200 мс
Инициализация магистрали	2—30 мс	4—12 мс
Выравнивание модуля, внедренного в действующую систему	1—129 мкс	1 мкс после завершения текущей передачи
Зависание линии	500—1000 мс (за исключением включения питания)	Не применяется

7.1.1.1 **Включение питания**

Когда происходит первоначальное включение питания системы, устройство питания выставляет сигнал сброса re★. Устройство питания должно выставить re★ раньше, чем любое из напряжений питания магистрали достигнет 40 % номинальной величины. Устройство питания поддерживает re★ выставленным в течение 100—200 мс после того, как все напряжения питания магистрали достигнут необходимого уровня. Это позволяет всем модулям на магистрали определить, что произошел системный сброс. Последовательность включения питания показана на рис. 7—1.

7.1.1.2 **Системный сброс**

Любой модуль, инициирующий системный сброс, выставляет re★ на 100—200 мс. Все модули в системе отслеживают сигнал RE★. Если модуль обнаруживает, что RE★ выставлен дольше, чем на 30 мс, то происходит системный сброс. Когда модуль обнаруживает, что произошел системный сброс, он выставляет re★. Затем модуль выполняет внутренние операции сброса, необходимые ему для участия в передачах по магистрали. Когда модуль завершил эти операции, он снимает re★, показывая, что он готов принять участие в передачах. После снятия re★ модули ожидают до тех пор, пока они не определят, что REf в логическом нуле. Это означает, что все модули готовы для проведения действий на магистрали.

Операции внутреннего сброса могут включать в себя самотестирование или другие операции, которые могут занимать длительное время. Поэтому было бы желательно проектировать модули, которые снимают re★ после того, как они полностью работоспособны, но до того, как они завершат самотестирование. В этом случае модули лишь нуждаются в способности выставлять признак занятости во время передач с последующим снятием re★, но после завершения самотестирования. Временная диаграмма системного сброса показана на рис. 7—2.

Если линия сброса не отпускается в пределах 500—1000 мс, за исключением первого включения питания, все модули отпускают re★, чем разрешают определение ситуации ошибочного зависания линии сброса. Любые последующие действия зависят от конкретной системы.

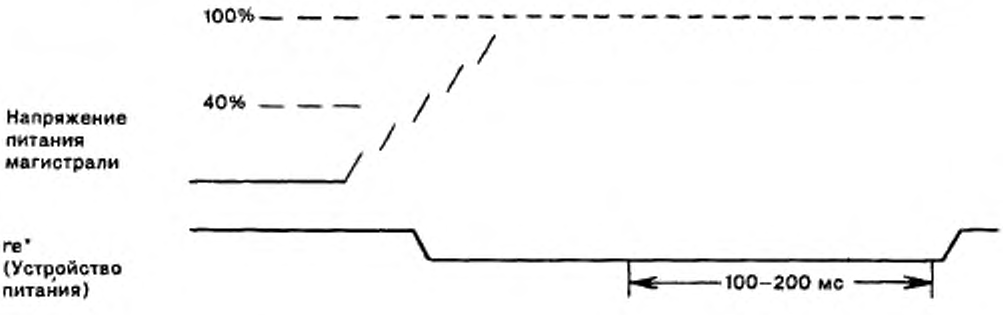

Рисунок 7—1 — Последовательность включения питания

Напряжение питания магистрали показано как однополярное положительное только для иллюстрации процесса. Действительные напряжения определены другими стандартами и могут быть как положительными, так и отрицательными и двуполярными

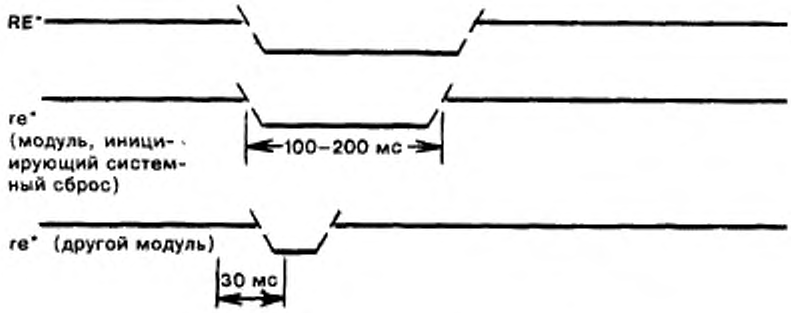

Рисунок 7—2 — Последовательность системного сброса

7.1.1.3 **Инициализация магистрали**

Если модули обнаруживают RE★ выставленным более 2 мс, происходит операция инициализации магистрали. Когда модуль определяет операцию инициализации магистрали, он инициализирует свои интерфейсные схемы и находится в ожидании, определяя, не является ли операция также и системным сбросом. Если модуль определяет REf в состоянии логического нуля до истечения 30 мс, он возобновляет операцию. Операция инициализации магистрали сбрасывает интерфейс магистрали, но она должна быть незаметной для функционирования остальной части системы.

7.1.1.4 **«Живое» вставление**

Логический протокол ФБ+ обеспечивает для модулей, вставленных в действующую систему с включенным питанием, возможность приведения их логического состояния в соответствие с состоянием других модулей на магистрали и начало работы. Когда модуль был вставлен в магистраль и определил, что он только что был включен, он проверяет, не выставлен ли RE★. Если сигнал не выставлен, то модуль предполагает, что он был внедрен в «живую» систему. Если линия сброса выставлена, то модуль ждет ее освобождения, чтобы установить, является ли это событие полным системным сбросом, а не инициализацией магистрали или «живым» вставлением. В конце концов модуль устанавливает факт вставления в действующую систему.

Модуль, вставленный в действующую систему, выставляет re★ для того, чтобы сообщить всем другим модулям, что он нуждается в подстройке. Модули не начинают никаких новых передач, что приводит к установлению на магистрали состояния «холостого хода». Внедренный модуль после выставления сигнала сброса ожидает до тех пор, пока магистраль не будет находиться в состоянии «холостого хода» как минимум 1 мкс, после чего выставляет ai★ и ar★ для завершения подстройки. После этого модуль отпускает re★ и все модули продолжают работу после того, как обнаружат, что REf находится в логическом нуле.

После последовательности подстройки все модули на магистрали «осведомлены» о том, что произошло «живое» вставление.

7.1.1.5 **«Живое» удаление**

Логический протокол ФБ+ также обеспечивает модулям возможность быть удаленными из включенной действующей системы без помех для операций, выполняемых другими модулями. Модули, подлежащие вынниманию, должны снять все сигналы таким образом, чтобы другие модули ошибочно не восприняли это как обычное снятие сигналов при нормальной работе.

Оператор, удаляющий модуль, должен вначале каким-либо образом известить плату о том, что она скоро будет удалена. Модуль, если он способен, завершает все задачи, находящиеся в данный момент в работе. Это может потребовать копирования хранящейся в модуле информации в другое место системы. Когда завершены передачи, требуемые для выполнения этих задач, модуль должен заблокировать свое участие в любых дальнейших передачах на магистрали. В это время модуль должен держать выставленными только ai★ и/или ak★ и, возможно, одну или более линий, синхронизирующих арбитражные сообщения.

Модулям разрешается освобождать ak★ в течение фазы данных или освобождать ar★ в течение «холостого хода» параллельной магистрали. Модулям разрешается вынимание из протокола арбитражных сообщений в одной из трех точек. В течение фазы 0 модулю разрешается вынимание освобождением ar★. В течение фазы 2 модулю разрешается вынимание освобождением ap★. В течение фазы 4 модулю разрешается вынимание освобождением aq★.

7.1.2 **Управляющие и статусные регистры ФБ+**

Модулю ФБ+ требуются три типа управляющих и статусных регистров. Регистры способностей модуля содержат информацию, указывающую на способности данного конкретного модуля. Регистры управления модулем используются для разрешения тех или иных его способностей и изменения управляющих параметров, действующих на режимы работы на магистрали. Регистры статуса модуля используются прежде всего для индикации ошибок и другой статусной информации. Определено только ограниченное число этих регистров, но есть зарезервированное пространство для других, определенных в других стандартах. Регистры подразумеваются 32-разрядные. Физические адреса регистров определены в стандартах более высокого уровня.

7.1.2.1 **Регистры возможностей модуля**

Все модули оборудованы следующими регистрами способностей:

★ ЛОГИЧЕСКИЕ_СПОСОБНОСТИ_МОДУЛЯ

★ ВНУТРЕННЯЯ_ЗАДЕРЖКА_СОРЕВНОВАНИЯ

★ РАЗМЕР_ФРАГМЕНТА_СООБЩЕНИЯ

★ СКОРОСТЬ_ПАКЕТА

Регистры способностей модуля доступны только для чтения. В большинстве систем данные для этих регистров будут расположены в ПЗУ способностей.

7.1.2.1.1 **Регистр логических способностей модуля**

Регистр логических способностей модуля показан на рис. 7—3. Каждое поле по отдельности описано в следующих параграфах.

Биты СПОСОБНОСТЬ_ПАКЕТНОЙ_ДЛИНЫ_n показывают, какую длину пакета поддерживает модуль. Действительные длины пакетов — 2, 4, 8, 16, 32 и 64 передачи.

Бит СПОСОБНОСТЬ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ, будучи установленным, показывает, что модуль поддерживает режим множественной передачи пакетов.

Бит ПАКЕТНАЯ_СПОСОБНОСТЬ, будучи установлен, показывает, что модуль поддерживает пакетный режим.

Бит ТЕГОВАЯ_СПОСОБНОСТЬ показывает, что модуль оборудован одной или более линией тега. Поле количества линий тега показывает, каким количеством теговых линий оборудован модуль

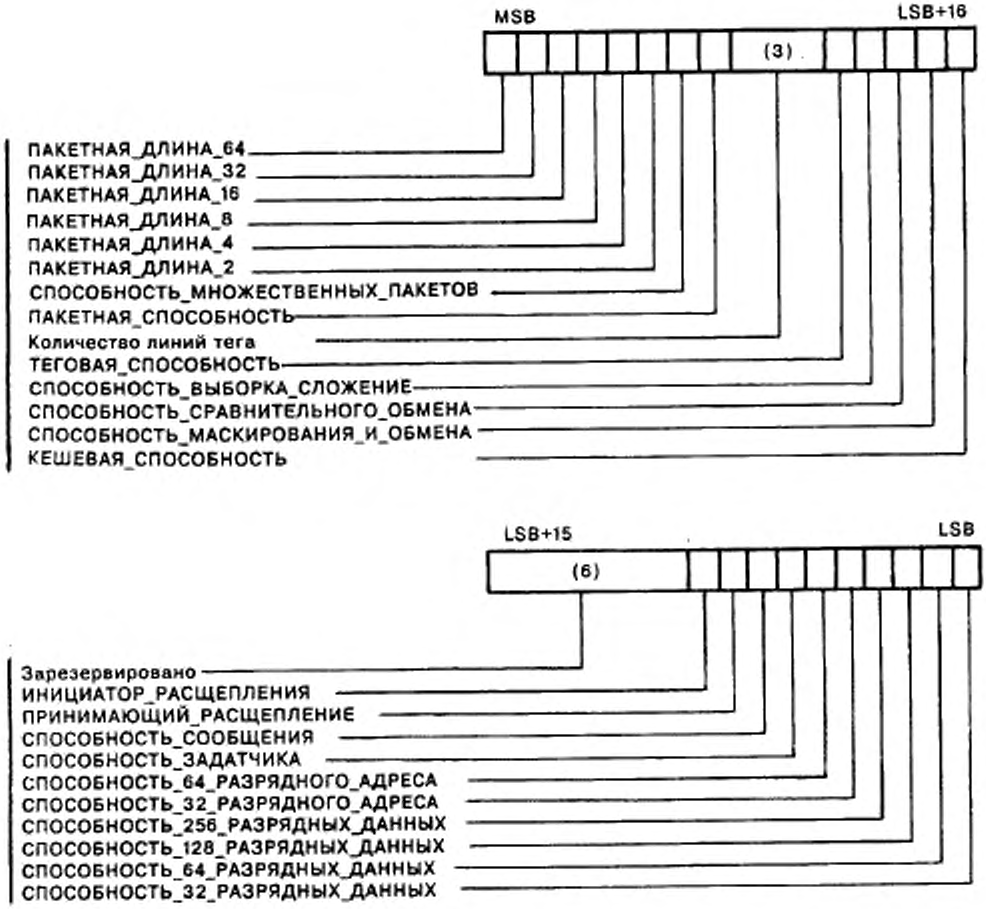

Рисунок 7— 3 — Регистр логических способностей модуля

Бит СПОСОБНОСТЬ_ВЫБОРКА_СЛОЖЕНИЕ показывает способность модуля выполнять защищенную команду ВЫБОРКА_СЛОЖЕНИЕ.

Бит СПОСОБНОСТЬ_СРАВНИТЕЛЬНОГО_ОБМЕНА показывает способность модуля выполнять защищенную команду СРАВНИТЕЛЬНЫЙ_ОБМЕН.

Бит СПОСОБНОСТЬ_МАСКИРОВАНИЯ_И_ОБМЕНА показывает способность модуля выполнять защищенную команду МАСКИРОВАННЫЙ_ОБМЕН.

Бит КЕШЕВАЯ_СПОСОБНОСТЬ показывает способность модуля принимать участие в передачах с кешированной когерентной разделяемой памятью.

Бит ИНИЦИАТОР_РАСЩЕПЛЕНИЯ показывает способность модуля инициировать расщепленные передачи. Инициатор расщепленной передачи способен выставлять сигнал SR★, после чего выполнять ответную передачу на магистрали.

Бит ПРИНИМАЮЩИЙ_РАСЩЕПЛЕНИЕ показывает, что модуль способен принимать расщепленные передачи. Когда принимающий расщепленную передачу является задатчиком и совершает передачу, он способен обнаружить выставление SR★, закончить передачу и ожидать ответную передачу от модуля, выставившего SR★.

Бит СПОСОБНОСТЬ_СООБЩЕНИЯ показывает способность модуля принимать участие в передаче прохождения сообщения, как описано в 9.2.

Бит СПОСОБНОСТЬ_ЗАДАТЧИКА показывает способность модуля становиться задатчиком

Бит СПОСОБНОСТЬ_64_РАЗРЯДНОГО_АДРЕСА показывает способность модуля поддерживать 64-разрядный адрес в дополнение 32-разрядному адресу, принятому по умолчанию.

Бит СПОСОБНОСТЬ_32_РАЗРЯДНОГО_АДРЕСА должен быть установлен, т. к. он показывает способность модуля работать с 32-разрядными адресами, которую должны обеспечивать все модули.

Биты СПОСОБНОСТЬ_32_РАЗРЯДНЫХ_ДАННЫХ, СПОСОБНОСТЬ_64_РАЗРЯДНЫХ_ДАННЫХ, СПОСОБНОСТЬ_128_РАЗРЯДНЫХ_ДАННЫХ, СПОСОБНОСТЬ_256_РАЗРЯДНЫХ_ДАННЫХ показывают разрядности данных, поддерживаемых данным модулем. Бит СПОСОБНОСТЬ_32_РАЗРЯДНЫХ_ДАННЫХ должен быть установлен, т. к. все модули поддерживают эту разрядность данных.

7.1.2.1.2 **Регистр внутренней задержки соревнования**

Система использует регистр внутренней задержки соревнования для вычисления времени решения наихудшего случая в течение времени соревнования арбитражного сообщения на магистрали (рис. 7—4).

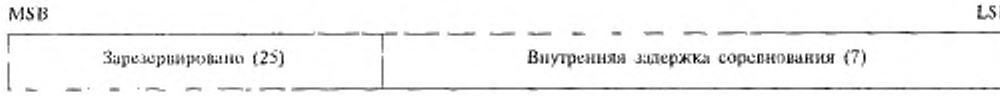

Рисунок 7—4 — Регистр внутренней задержки соревнования

7.1.2.1.3 **Регистр размера фрагмента сообщения**

Все модули, имеющие способность передачи сообщений, должны быть способны передавать фрагмент сообщения в 64 байта. Модули, имеющие способность передачи сообщений, могут указывать на свою способность передавать сообщения, размер фрагмента которых есть целое число, умноженное на 64 байта. Максимальный размер фрагмента указывается величиной в регистре размера фрагмента сообщения (рис. 7—5).

Рисунок 7—5 — Регистр размера фрагмента сообщения

7.1.2.1.4 **Регистр скорости пакета**

Регистр скорости пакета содержит четыре 8-разрядных поля, показывающих одну из четерых скоростей передачи, поддерживаемых модулем (рис. 7—6). Нулевая величина показывает, что поле не содержит поддерживаемой скорости передачи пакета.

Рисунок 7—6 — Регистр скорости пакета

7.1.2.2 **Регистры управления модулем**

Все модули оборудованы следующими регистрами управления:

* ЛОГИЧЕСКИЙ_РЕГИСТР_ОБЩЕГО_УПРАВЛЕНИЯ
* ЛОГИЧЕСКИЙ_РЕГИСТР_УПРАВЛЕНИЯ_МОДУЛЕМ
* РЕГИСТР_ЗАДЕРЖКИ_РАСПРОСТРАНЕНИЯ_МАГИСТРАЛИ
* РЕГИСТР_ВРЕМЕНИ_ЗАВЕРШЕНИЯ_СОРЕВНОВАНИЯ
* РЕГИСТР_ТАЙМ-АУТА_ПЕРЕДАЧИ
* МАСКА_ВЫБОРА_ПРОХОЖДЕНИЯ_СООБЩЕНИЯ

Регистры управления модулем доступны как для чтения, так и для записи.

7.1.2.2.1 **Логический регистр общего управления**

Логический регистр общего управления показан на рис. 7—7.

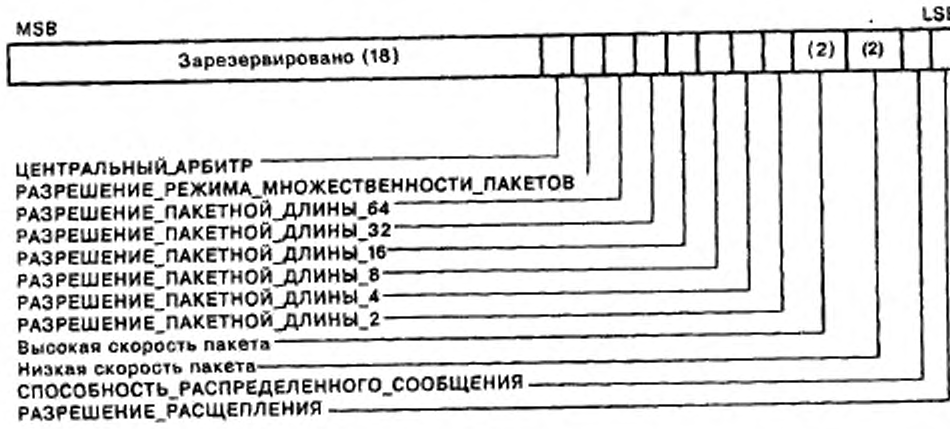

Рисунок 7—7 — Логический регистр общего управления

Каждое поле по отдельности описано в следующих параграфах.

Процесс конфигурации устанавливает разряд ЦЕНТРАЛЬНЫЙ_АРБИТР для выборки центрального арбитра вместо распределенного арбитра с целью определения задатчика магистрали. Этот разряд также устанавливается аппаратно, когда REf переходит в положение логического нуля после включения питания или системного сброса, если на магистрали есть модуль центрального арбитра.

Процесс конфигурации устанавливает разряд РАЗРЕШЕНИЕ_РЕЖИМА_МНОЖЕСТВЕННОСТИ_ПАКЕТОВ для указания на то, что модуль может производить передачи, используя множественный пакетный режим.

Процесс конфигурации устанавливает разряды РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_n для указания того, какие длины передачи пакетов допускается использовать модулю.

Процесс конфигурации записывает в поле «высокая скорость пакета» одну из четырех возможных величин для указания той скорости пакета, которую модуль может использовать во время передачи пакета с высокой скоростью.

Процесс конфигурации записывает в поле «низкая скорость пакета» одну из четырех возможных величин для указания той скорости пакета, которую модуль может использовать во время передачи пакета с низкой скоростью.

Процесс конфигурации устанавливает разряд СПОСОБНОСТЬ_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ для того, чтобы разрешить модулю посылать сообщения распределенного арбитража.

Процесс конфигурации устанавливает разряд РАЗРЕШЕНИЕ_РАСЩЕПЛЕНИЯ для указания того, что модуль может расщеплять передачи на магистрали, если он имеет такую способность.

7.1.2.2.2 **Логический регистр управления модулем**

Логический регистр управления модулем показан на рис. 7—8. Каждое поле по отдельности описано в следующих параграфах.

Процесс конфигурации устанавливает разряд РАЗРЕШЕНИЕ_ТЕГОВ для указания того, что теговые разряды активны и что данный модуль может активизировать теговые разряды и должен проверять четность тега.

Процесс конфигурации устанавливает разряд РАЗРЕШЕНИЕ_СООБЩЕНИЯ_ЧЕТНОСТИ для указания того, что модули будут сообщать об ошибке четности, выставлением be★ и te★.

Процесс конфигурации устанавливает разряд РАЗРЕШЕНИЕ_ВЫСОКОЙ_СКОРОСТИ для указания того, что модуль способен работать с высокой скоростью пакета

Процесс конфигурации устанавливает разряд РАЗРЕШЕНИЕ_ЗАДАТЧИКА — 8 для того, чтобы разрешить модулю становиться задатчиком (при наличии у модуля такой способности).

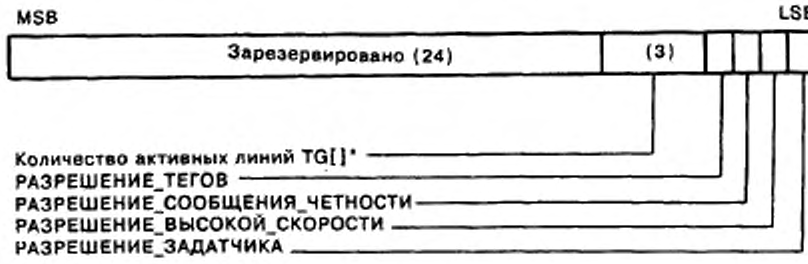

Рисунок 7 - 8 Логический регистр управления модулем

7.1.2.2.3 **Регистр задержки распространения магистрали**

Процесс конфигурации записывает в поле задержки распространения магистрали величину задержки распространения сигнала в одном направлении по магистрали (рис. 7—9). Модули могут использовать это значение для установки своих схем объединения по проводному ИЛИ и задержки арбитражного сообщения.

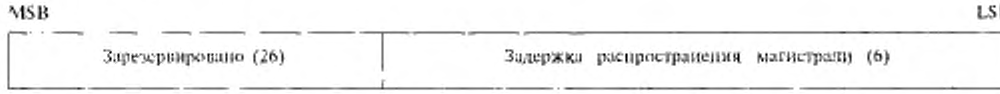

Рисунок 7 - 9 Регистр задержки распространения магистрали

7.1.2.2.4 **Регистр времени завершения соревнования**

Процесс конфигурации устанавливает регистр времени завершения соревнования в соответствии с наихудшим значением, которое модуль будет использовать для определения своего времени завершения соревнования в процессе соревнования арбитражных сообщений (рис. 7—10).

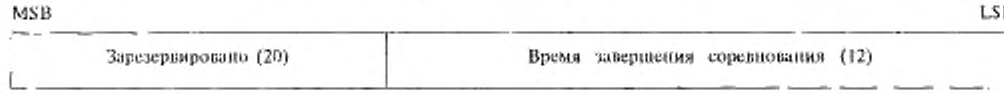

Рисунок 7—10 — Регистр времени завершения соревнования

7.1.2.2.5 **Регистр тайм-аута передачи**

Процесс конфигурации записывает в поле тайм-аута передачи длительность промежутка времени, в течение которого задатчик ждет перед тем, как установить свой атрибут тайм-аута передачи (рис. 7—11).

Рисунок 7—11 — Регистр тайм-аута передачи

7.1.2.2.6 **Маска выбора прохождения сообщения**

Биты в маске выбора прохождения сообщения устанавливаются для разрешения модулям получать определенные сообщения через свои почтовые ящики (рис. 7 - 12). Младший разряд не используется.

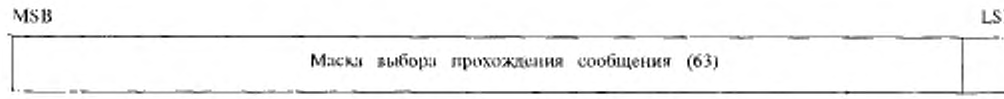

Рисунок 7 - 12 — Маска выбора прохождения сообщения

7.2 **Спецификация**

7.2.1 **Атрибуты управления магистралью**

УДЕРЖАНИЕ_МАГИСТРАЛИ

Модули должны устанавливать УДЕРЖАНИЕ_МАГИСТРАЛИ при условии: RE★ & -УСТРОЙСТВО МАГИСТРАЛИ & -СИСТЕМНЫЙ_СБРОС & —ВКЛЮЧЕНИЕ_ПИТАНИЯ & -ЖИВОЕ_ВСТАВЛЕНИЕ. Модули должны удерживать установленным УДЕРЖАНИЕ_МАГИСТРАЛИ до: ЖИВОЕ_ВСТАВЛЕНИЕ | УСТРОЙСТВО_МАГИСТРАЛИ | СИСТЕМНЫЙ_СБРОС.

НЕЗАНЯТОСТЬ_МАГИСТРАЛИ

Модули должны устанавливать НЕЗАНЯТОСТЬ_МАГИСТРАЛИ при условии: -AQf & AR★ & -AS★ & AI★ & КОНЕЦ_РАССОЕДИНЕНИЯ.

НЕЗАНЯТОСТЬ_МАГИСТРАЛИ_IUS

Модули должны устанавливать НЕЗАНЯТОСТЬ_МАГИСТРАЛИ_IUS при условии: НЕЗАНЯТОСТЬ_МАГИСТРАЛИ был установлен в течение минимум 1 мкс; и должны поддерживать его установленным до: НЕЗАНЯТОСТЬ_МАГИСТРАЛИ.

УСТРОЙСТВО_МАГИСТРАЛИ

Модули должны устанавливать УСТРОЙСТВО_МАГИСТРАЛИ при условии: REf & —СИСТЕМНЫЙ_СБРОС & —ВКЛЮЧЕНИЕ_ПИТАНИЕ в течение минимум 2 мс. Модули должны поддерживать УСТРОЙСТВО_МАГИСТРАЛИ установленным до: СИСТЕМНЫЙ_СБРОС | ВКЛЮЧЕНИЕ_ПИТАНИЯ | ~REf.

ВХОДЯЩИЙ

Модули должны устанавливать ВХОДЯЩИЙ при условии: ВКЛЮЧЕНИЕ_ПИТАНИЯ & -REf. Модули должны удерживать ВХОДЯЩИЙ установленным до: СИСТЕМНЫЙ СБРОС & (ai★ | ar★).

ИНИЦИАЛИЗАЦИЯ

Модули должны устанавливать ИНИЦИАЛИЗАЦИЯ при условии: ВКЛЮЧЕНИЕ_ПИТАНИЯ | СИСТЕМНЫЙ_СБРОС | УСТРОЙСТВО_МАГИСТРАЛИ.

«ЖИВОЕ»_ВСТАВЛЕНИЕ

Модули должны устанавливать «ЖИВОЕ»_ВСТАВЛЕНИЕ при условии: -ВКЛЮЧЕНИЕ_ПИТАНИЯ & -ВХОДЯЩИЙ & СОБЫТИЕ_СБРОСА & -REf. Модули должны удерживать «ЖИВОЕ»_ВСТАВЛЕНИЕ установленным до: ФАЗА_5.

КОМАНДА_«ЖИВОГО»_УДАЛЕНИЯ

КОМАНДА_«ЖИВОГО»_УДАЛЕНИЯ должна быть установлена в модуле, когда он предупрежден о том, что будет отключен.

ВКЛЮЧЕНИЕ_ПИТАНИЯ

Модули должны устанавливать ВКЛЮЧЕНИЕ_ПИТАНИЯ при подаче напряжения питания. Вставленные «вживую» модули не должны выставлять никаких сигналов на магистраль до установки ВКЛЮЧЕНИЕ_ПИТАНИЯ. Модули должны удерживать ВКЛЮЧЕНИЕ_ПИТАНИЯ установленным до: СИСТЕМНЫЙ_СБРОС | НЕЗАНЯТОСТЬ_МАГИСТРАЛИ_IUS & ai★ & ar★. Модули должны ожидать минимум 1 мкс после выставления re★ до того, как сбросить ВКЛЮЧЕНИЕ_ПИТАНИЯ. Модули, не способные управлять ar★, могут предполагать, что он выставлен.

ВРЕМЯ_ВКЛЮЧЕНИЯ_ПИТАНИЯ

Модули должны устанавливать ВРЕМЯ_ВКЛЮЧЕНИЯ_ПИТАНИЯ при подаче напряжения питания. Модули должны сбрасывать ВРЕМЯ_ВКЛЮЧЕНИЯ_ПИТАНИЯ минимум через 500 мс и не позднее чем через 1000 мс.

ГОТОВНОСТЬ_ДЛЯ_УДАЛЕНИЯ

Модули должны устанавливать ГОТОВНОСТЬ_ДЛЯ_УДАЛЕНИЯ при условии: КОМАНДА_ЖИВОГО_ОТКЛЮЧЕНИЯ & -ap★ & -aq★ & -ar★ & -as★ & -ak★ & -ai★ & -cm[]★ & -cp★ & -st[]★ & -ca[]★ & -re★ & -ab[]★ & -abp★ & -ac0★ & -ac1★ & -ad[]★ & -d[]★ & -bp[]★ & -rq0★ & -rq1★ & -et★.

ЗАВЕРШЕННЫЙ_СБРОС

Модуль должен устанавливать ЗАВЕРШЕННЫЙ_СБРОС при условии: СИСТЕМНЫЙ_СБРОС и если он завершил свою процедуру сброса, и готов для передач на магистрали. Модули должны удерживать ЗАВЕРШЕННЫЙ_СБРОС установленным до: —REf.

СОБЫТИЕ СБРОСА

Модули должны устанавливать СОБЫТИЕ_СБРОСА при условии: RE★. Модули должны удерживать СОБЫТИЕ_СБРОСА установленным до: СИСТЕМНЫЙ_СБРОС ¦ УСТРОЙСТВО_МАГИСТРАЛИ ¦ «ЖИВОЕ»_ВСТАВЛЕНИЕ.

СИСТЕМНЫЙ_СБРОС

Модули должны устанавливать СИСТЕМНЫЙ_СБРОС при условии: REf выставлен в течение минимум 30 мс. Модули должны удерживать СИСТЕМНЫЙ_СБРОС установленным до: -REf.

7.2.2 **Сигнал сброса RE★**

Модули должны выставлять re★ при условии: СИСТЕМНЫЙ_СБРОС & -ЗАВЕРШЕННЫЙ СБРОС ¦ ВХОДЯЩИЙ & -REf или инициировать системный сброс, инициализацию магистрали или «живое» вставление, как определено в 7.2.3. Модули должны удерживать re★ выставленным до: СИСТЕМНЫЙ_СБРОС & ЗАВЕРШЕННЫЙ_СБРОС ¦ ВХОДЯЩИЙ & НЕЗАНЯТОСТЬ_МАГИСТРАЛИ_IUS & ar★ & ai★.

7.2.3 **Определение протокола**

7.2.3.1 **Включение питания**

Система питания должна:

1) выставить re★ прежде, чем любое из питающих напряжений магистрали достигнет 40 % от номинала,

2) поддерживать re★ выставленным до тех пор, пока все питающие напряжения магистрали не установятся в пределах, определенных спецификацией,

3) продолжать удерживать re★ еще 100—200 мс,

4) отпустить re★.

Модули должны:

1) после определения наличия напряжения питания и появлении способности установить ВКЛЮЧЕНИЕ_ПИТАНИЯ,

2) ожидать до: СИСТЕМНЫЙ_СБРОС · --REf,

3) приступить к системному сбросу, как определено в 7.2.3.2 при условии: СИСТЕМНЫЙ_СБРОС,

4) приступить к процедуре «живого» вставления, как определено в 7.2.3.4 при условии. –REf.

7.2.3.2 **Системный сброс**

После обнаружения СИСТЕМНЫЙ_СБРОС модули должны:

1) выставить re★.

2) выполнить все внутренние операции сброса, необходимые для последующего участия в передачах на магистрали,

3) отпустить re★,

4) ожидать до: -REf или 500—1000 мс & ВРЕМЯ_ВКЛЮЧЕНИЯ_ПИТАНИЯ,

5) продолжить требуемые действия на магистрали.

Для инициализации операции системного сброса модуль должен выставить re★ на 100—200 мс, после чего перейти к 2).

7.2.3.3 **Инициализация магистрали**

После обнаружения УСТРОЙСТВО_МАГИСТРАЛИ модули должны:

1) инициализировать все схемы магистрального интерфейса,

2) ожидать до: -УСТРОЙСТВО_МАГИСТРАЛИ,

3) продолжить требуемые действия на магистрали при условии: -СИСТЕМНЫЙ_СБРОС.

Для инициализации операции инициализации магистрали модуль должен выставить re★ минимум на 4 мс и не дольше, чем на 12 мс, после чего перейти к 2).

7.2.3.4 **«Живое» вставление**

Модули должны:

1) совершить все внутренние операции сброса, необходимые для принятия участия в передачах на магистрали,

2) ожидать до. СИСТЕМНЫЙ_СБРОС ¦ -REf при условии RE★,

3) приступить к системному сбросу, как определено в 7.2.3.2 при условии: СИСТЕМНЫЙ_СБРОС,

4) ждать как минимум 1 мкс, после чего выставить re★ при условии : -REf,

5) ожидать до: НЕЗАНЯТОСТЬ_МАГИСТРАЛИ_IUS ¦ СИСТЕМНЫЙ_СБРОС,

6) приступить к системному сбросу, как определено в 7.2.3.2 при условии. СИСТЕМНЫЙ_СБРОС,

7) выставить ai★ и, если есть такая способность, выставить ar★ при условии. НЕЗАНЯТОСТЬ МАГИСТРАЛИ_1US.

8) отпустить re★,

9) продолжить требуемые действия на магистрали.

7.2.3.5 **«Живое» удаление**

После обнаружения КОМАНДА_«ЖИВОГО»_УДАЛЕНИЯ модули должны:

1) произвести любые зависящие от конкретной системы операции для сохранения состояния,

2) ожидать появление перечисленных ниже (а—д) событий и отключать соответствующие сигналы до появления условия: -ap★ & -aq★ & -ar★ & -as★ & -ak★ & -ai★ & -cm[]★ & -cp★ & -st[]★ & -ca[]★ & -re★ & -ab[]★ & -abp★ & -ac0★ & -ac1★ & -ad[]★ & -d[]★ & -bp[]★ & -rq0★ & -rq1★ & -et★:

а) после обнаружения выставления AS★ модули не должны выставлять ak★, а при условии: AS★ & AK★ & -ak★ должны отпустить ai★,

б) после обнаружения освобождения AS★ модули не должны выставлять ai★, а при условии: -AS★ & AI★ & -ai★ должны отпустить ak★,

в) отпустить ar★ при условии: -AQf & ar★ & -ap★,

г) отпустить ap★ при условии: -ARf & ap★ & -aq★,

д) отпустить aq★ при условии: -APf & aq★ & -ar★;

3) установить ГОТОВНОСТЬ_ДЛЯ_УДАЛЕНИЯ

7.2.4 **Управляющие и статусные регистры ФБ+**

Поля, обозначенные как «зарезервировано», должны возвращать нули при считывании. Данные, записываемые в поля, обозначенные как «зарезервировано», должны иметь нулевые значения. Все подключенные разряды управляющих регистров должны возвращать ранее записанные в них значения.

Все регистры временных интервалов программируются в 2★★(-32) долях секунды (233 пс). Временное разрешение таймеров не определяется этим форматом регистра.

7.2.4.1 **Регистры способностей модуля**

Модули должны быть оборудованы следующими регистрами способностей:

★ ЛОГИЧЕСКИЕ_СПОСОБНОСТИ_МОДУЛЯ

★ ВНУТРЕННЯЯ_ЗАДЕРЖКА_СОРЕВНОВАНИЯ

★ РАЗМЕР_ФРАГМЕНТА_СООБЩЕНИЯ

★ СКОРОСТЬ_ПАКЕТА

Регистры способностей модуля должны быть доступны только для чтения.

7.2.4.1.1 **Регистр логических способностей модуля**

СПОСОБНОСТЬ_ПАКЕТНОЙ_ДЛИНЫ_64

LSB+31 должен быть СПОСОБНОСТЬ_ПАКЕТНОЙ_ДЛИНЫ_64 и должен быть установлен, если модуль имеет способность принимать участие в передачах, используя при этом пакетный протокол ПЕРЕДАЧА_64.

СПОСОБНОСТЬ_ПАКЕТНОЙ_ДЛИНЫ_32

LSB+30 должен быть СПОСОБНОСТЬ_ПАКЕТНОЙ_ДЛИНЫ_32 и должен быть установлен, если модуль имеет способность принимать участие в передачах, используя при этом пакетный протокол ПЕРЕДАЧА_32.

СПОСОБНОСТЬ_ПАКЕТНОЙ_ДЛИНЫ_16

LSB+29 должен быть СПОСОБНОСТЬ_ПАКЕТНОЙ_ДЛИНЫ_16 и должен быть установлен, если модуль имеет способность принимать участие в передачах, используя при этом пакетный протокол ПЕРЕДАЧА_16.

СПОСОБНОСТЬ_ПАКЕТНОЙ_ДЛИНЫ_8

LSB+28 должен быть СПОСОБНОСТЬ_ПАКЕТНОЙ_ДЛИНЫ_8 и должен быть установлен, если модуль имеет способность принимать участие в передачах, используя при этом пакетный протокол ПЕРЕДАЧА_8.

СПОСОБНОСТЬ_ПАКЕТНОЙ_ДЛИНЫ_4

LSB+27 должен быть СПОСОБНОСТЬ_ПАКЕТНОЙ_ДЛИНЫ_4 и должен быть установлен, если модуль имеет способность принимать участие в передачах, используя при этом пакетный протокол ПЕРЕДАЧА_4.

СПОСОБНОСТЬ_ПАКЕТНОЙ_ДЛИНЫ_2

LSB+26 должен быть СПОСОБНОСТЬ_ПАКЕТНОЙ_ДЛИНЫ_2 и должен быть установлен, если модуль имеет способность принимать участие в передачах, используя при этом пакетный протокол ПЕРЕДАЧА_2.

СПОСОБНОСТЬ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ

LSB+25 должен быть СПОСОБНОСТЬ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ и должен быть установлен, если модуль имеет способность принимать участие в передачах, используя при этом пакетный протокол в режиме множественных пакетов.

ПАКЕТНАЯ_СПОСОБНОСТЬ

LSB+24 должен быть ПАКЕТНАЯ_СПОСОБНОСТЬ и должен быть установлен, если модуль имеет способность принимать участие в передачах, используя при этом пакетный протокол.

КОЛИЧЕСТВО ЛИНИЙ ТЕГА TG[]★

LSB+21, LSB+22 и LSB+23 должны быть трехразрядным полем, содержащим число линий TG[]★, которыми оснащен модуль. Значение 0 должно соответствовать одной линии тега TG[]★, а значение 7 должно соответствовать восьми линиям тега TG[]★.

ТЕГОВАЯ_СПОСОБНОСТЬ

LSB+20 должен быть ТЕГОВАЯ_СПОСОБНОСТЬ и должен быть установлен, если модуль оснащен одной или более линиями TG[]★.

СПОСОБНОСТЬ_ВЫБОРКА_СЛОЖЕНИЕ

LSB+19 должен быть СПОСОБНОСТЬ_ВЫБОРКА_СЛОЖЕНИЕ и должен быть установлен, если модуль имеет способность откликаться на защищенные команды ВЫБОРКА_СЛОЖЕНИЕ_БОЛЬШОЙ и ВЫБОРКА_СЛОЖЕНИЕ_МАЛЫЙ.

СПОСОБНОСТЬ_СРАВНИТЕЛЬНОГО_ОБМЕНА

LSB+18 должен быть СПОСОБНОСТЬ_СРАВНИТЕЛЬНОГО_ОБМЕНА и должен быть установлен, если модуль имеет способность откликаться на защищенную команду СРАВНИТЕЛЬНЫЙ_ОБМЕН.

СПОСОБНОСТЬ_МАСКИРОВАНИЯ_И_ОБМЕНА

LSB+17 должен быть СПОСОБНОСТЬ_МАСКИРОВАНИЯ_И_ОБМЕНА и должен быть установлен, если модуль имеет способность откликаться на защищенную команду МАСКИРОВАННЫЙ_ОБМЕН.

КЕШЕВАЯ_СПОСОБНОСТЬ

LSB+16 должен быть КЕШЕВАЯ_СПОСОБНОСТЬ и должен быть установлен, если модуль имеет способность совершать передачи в соответствии с 8.2.

ИНИЦИАТОР_РАСЩЕПЛЕНИЯ

LSB+9 должен быть ИНИЦИАТОР_РАСЩЕПЛЕНИЯ и должен быть установлен, если модуль имеет способность инициировать расщепленные передачи.

ПРИНИМАЮЩИЙ_РАСЩЕПЛЕНИЕ

LSB+8 должен быть ПРИНИМАЮЩИЙ_РАСЩЕПЛЕНИЕ и должен быть установлен, если модуль имеет способность поддерживать расщепленные передачи.

СПОСОБНОСТЬ_СООБЩЕНИЯ

LSB+7 должен быть СПОСОБНОСТЬ_СООБЩЕНИЯ и должен быть установлен, если модуль имеет способность принимать участие в передачах, используя протокол передачи сообщений.

СПОСОБНОСТЬ_ЗАДАТЧИКА

LSB+6 должен быть СПОСОБНОСТЬ_ЗАДАТЧИКА и должен быть установлен, если модуль имеет способность становиться ЗАДАТЧИКОМ.

СПОСОБНОСТЬ_64_РАЗРЯДНОГО_АДРЕСА

LSB+5 должен быть СПОСОБНОСТЬ_64_РАЗРЯДНОГО_АДРЕСА и должен быть установлен, если модуль имеет способность принимать участие в передачах с 64-разрядным адресом.

СПОСОБНОСТЬ_32_РАЗРЯДНОГО_АДРЕСА

LSB+4 должен быть СПОСОБНОСТЬ_32_РАЗРЯДНОГО_АДРЕСА и должен быть установлен.

СПОСОБНОСТЬ_256_РАЗРЯДНЫХ_ДАННЫХ

LSB+3 должен быть СПОСОБНОСТЬ_256_РАЗРЯДНЫХ_ДАННЫХ и должен быть установлен, если модуль имеет способность принимать участие в передачах с 256-разрядными данными.

СПОСОБНОСТЬ_128_РАЗРЯДНЫХ_ДАННЫХ

LSB+2 должен быть СПОСОБНОСТЬ_128_РАЗРЯДНЫХ_ДАННЫХ и должен быть установлен, если модуль имеет способность принимать участие в передачах с 128-разрядными данными.

СПОСОБНОСТЬ_64_РАЗРЯДНЫХ_ДАННЫХ

LSB+1 должен быть СПОСОБНОСТЬ_64_РАЗРЯДНЫХ_ДАННЫХ и должен быть установлен, если модуль имеет способность принимать участие в передачах с 64-разрядными данными.

СПОСОБНОСТЬ_32_РАЗРЯДНЫХ_ДАННЫХ

LSB должен быть СПОСОБНОСТЬ_32_РАЗРЯДНЫХ_ДАННЫХ и должен быть установлен.

7.2.4.1.2 **Регистр внутренней задержки соревнования**

Семь младших разрядов регистра внутренней задержки соревнования должны соответствовать наихудшему случаю задержки прохождения сигнала через логику соревнования между соседними парами линий арбитражных сообщений магистрали, как показано ниже.

1) От AB7★ к AB6★
2) От AB6★ к AB5★
3) От AB5★ к AB4★
4) От AB4★ к AB3★
5) От AB3★ к AB2★
6) От AB2★ к AB1★
7) От AB1★ к AB0★
8) От AB0★ к ABP★

Значение поля должно быть в шагах по 2★★(-32) с (≈233 пс). Надо отметить, что максимальная величина поля меньше, чем 30 нс.

7.2.4.1.3 **Регистр размера фрагмента сообщения**

Все модули, имеющие способность передачи сообщений, должны быть способны передавать и принимать фрагменты сообщений в 64 байта. Величина в регистре размера фрагмента сообщения должна показывать максимальное количество байт, которое имеющий способность к передаче сообщений модуль может передать в одном фрагменте сообщений. Величина в регистре размера фрагмента сообщений должна быть равной целому числу, умноженному на 64, т. е. шесть младших значащих разрядов должны быть нулевыми.

7.2.4.1.4 **Регистр скорости пакета**

Регистр скорости пакета содержит четыре 8-разрядных поля, определяющих от одной до четырех скоростей передачи пакета, поддерживаемых модулем. Скорость передачи должна быть определена как обратная величина от длительности одного битового периода. Величина в каждом из полей выражена в шагах по 1 МГц. Нулевая величина указывает, что данное поле не содержит поддерживаемой скорости передачи пакета.

Поле, заключающее в себе разряды от LSB до LSB+7 включ., соответствует Скорости Передачи Пакета A.

Поле, заключающее в себе разряды от LSB+8 до LSB+15 включ., соответствует Скорости Передачи Пакета B.

Поле, заключающее в себе разряды от LSB+16 до LSB+23 включ., соответствует Скорости Передачи Пакета C.

Поле, заключающее в себе разряды от LSB+24 до MSB включ., соответствует Скорости Передачи Пакета D.

7.2.4.2 **Регистры управления модулем**

Модули должны быть оборудованы следующими управляющими регистрами, как определено в стандарте более высокого уровня, таком, как P 896.2:

★ Логический регистр общего управления
★ Логический регистр управления модулем
★ Регистр задержки распространения магистрали
★ Регистр времени завершения соревнования
★ Регистр тайм-аута передачи
★ Маска выбора прохождения сообщения

7.2.4.2.1 **Логический регистр общего управления**

ЦЕНТРАЛЬНЫЙ_АРБИТР

LSB+13 должен быть ЦЕНТРАЛЬНЫЙ_АРБИТР. Модули должны использовать центрального арбитра для целей распределения параллельной магистрали в случае, если этот разряд установлен. Модули должны использовать для этих целей распределенный арбитраж в случае, если этот разряд сброшен. Модули должны устанавливать ЦЕНТРАЛЬНЫЙ_АРБИТР при условии: PE★ выставлен или сбрасывать ЦЕНТРАЛЬНЫЙ_АРБИТР при условии: PE★ отпущен и они обнаруживают переход REf в логический ноль, следующий за ВКЛЮЧЕНИЕ_ПИТАНИЯ ¦ СИСТЕМНЫЙ_СБРОС.

РАЗРЕШЕНИЕ_РЕЖИМА_МНОЖЕСТВЕННОСТИ_ПАКЕТОВ

LSB+12 должен быть РАЗРЕШЕНИЕ_РЕЖИМА_МНОЖЕСТВЕННОСТИ_ПАКЕТОВ. Модули, способные работать во множественном пакетном режиме, могут инициировать множественные пакетные передачи при условии РАЗРЕШЕНИЕ_РЕЖИМА_МНОЖЕСТВЕННОСТИ_ПАКЕТОВ установлен и также установлен один из описанных ниже разрядов РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ n. Модули должны сбрасывать РАЗРЕШЕНИЕ_РЕЖИМА_МНОЖЕСТВЕННОСТИ_ПАКЕТОВ при условии: СИСТЕМНЫЙ_СБРОС ¦ УСТРОЙСТВО_МАГИСТРАЛИ ¦ ВКЛЮЧЕНИЕ_ПИТАНИЯ.

РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_64

LSB+11 должен быть РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_64. Модули могут инициировать пакетные передачи при условии: РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_64 установлен. Модули должны сбрасывать РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_64 при условии: СИСТЕМНЫЙ_СБРОС ¦ УСТРОЙСТВО_МАГИСТРАЛИ ¦ ВКЛЮЧЕНИЕ_ПИТАНИЯ

РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_32

LSB+10 должен быть РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_32. Модули могут инициировать пакетные передачи при условии: РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_32 установлен. Модули должны сбрасывать РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_32 при условии: СИСТЕМНЫЙ_СБРОС ¦ УСТРОЙСТВО_МАГИСТРАЛИ ¦ ВКЛЮЧЕНИЕ_ПИТАНИЯ.

РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_16

LSB+9 должен быть РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_16. Модули могут инициировать пакетные передачи при условии: РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_16 установлен. Модули должны сбрасывать РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_16 при условии: СИСТЕМНЫЙ_СБРОС ¦ УСТРОЙСТВО_МАГИСТРАЛИ ¦ ВКЛЮЧЕНИЕ_ПИТАНИЯ.

РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_8

LSB+8 должен быть РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_8. Модули могут инициировать пакетные передачи при условии: РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_8 установлен. Модули должны сбрасывать РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_8 при условии: СИСТЕМНЫЙ_СБРОС ¦ УСТРОЙСТВО_МАГИСТРАЛИ ¦ ВКЛЮЧЕНИЕ_ПИТАНИЯ.

РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_4

LSB+7 должен быть РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_4. Модули могут инициировать пакетные передачи при условии: РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_4 установлен. Модули должны сбрасывать РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_4 при условии: СИСТЕМНЫЙ_СБРОС ¦ УСТРОЙСТВО_МАГИСТРАЛИ ¦ ВКЛЮЧЕНИЕ_ПИТАНИЯ.

РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_2

LSB+6 должен быть РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_2. Модули могут инициировать пакетные передачи при условии: РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_2 установлен. Модули должны сбрасывать РАЗРЕШЕНИЕ_ПАКЕТНОЙ_ДЛИНЫ_2 при условии: СИСТЕМНЫЙ_СБРОС ¦ УСТРОЙСТВО_МАГИСТРАЛИ ¦ ВКЛЮЧЕНИЕ_ПИТАНИЯ.

ВЫСОКАЯ_СКОРОСТЬ_ПАКЕТА

LSB+4 и LSB+5 должны быть двухразрядным полем, определяющим ту скорость пакета, которую модуль должен использовать при условии: -СТАТУС_НИЗКОЙ_СКОРОСТИ. Нулевая величина соответствует скорости пакета A. Величина, равная трем (двоичная), соответствует скорости пакета D. Модули должны очищать это поле при условии: СИСТЕМНЫЙ_СБРОС ¦ УСТРОЙСТВО_МАГИСТРАЛИ ¦ ВКЛЮЧЕНИЕ_ПИТАНИЯ.

НИЗКАЯ_СКОРОСТЬ_ПАКЕТА

LSB+2 и LSB+3 должны быть двухразрядным полем, определяющим ту скорость пакета, которую модуль должен использовать при условии: -СТАТУС_НИЗКОЙ_СКОРОСТИ Нулевая величина соответствует скорости пакета A. Величина, равная трем (двоичная), соответствует скорости пакета D. Модули должны очищать это поле при условии: СИСТЕМНЫЙ_СБРОС ¦ УСТРОЙСТВО_МАГИСТРАЛИ ¦ ВКЛЮЧЕНИЕ_ПИТАНИЯ.

СПОСОБНОСТЬ_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ

LSB+1 должен быть СПОСОБНОСТЬ_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ. Модули должны сбрасывать СПОСОБНОСТЬ_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ при условии: СИСТЕМНЫЙ_СБРОС ¦ УСТРОЙСТВО_МАГИСТРАЛИ ¦ ВКЛЮЧЕНИЕ_ПИТАНИЯ. Если этот бит сброшен системным процессором, модули должны отключиться от активного арбитражного соревнования, сбрасывая ЗАПРОС_РАСПРЕДЕЛЕННОГО_СООБЩЕНИЯ, ЗАПРОС_ЦЕНТРАЛЬНОГО_СООБЩЕНИЯ и ЗАПРОС_АРБИТРАЖНОГО_СООБЩЕНИЯ, как определено в 5.2.1 и 5.2.2.

РАЗРЕШЕНИЕ_РАСЩЕПЛЕНИЯ

LSB должен быть РАЗРЕШЕНИЕ_РАСЩЕПЛЕНИЯ. Модули могут расщеплять передачи на магистрали при условии: РАЗРЕШЕНИЕ_РАСЩЕПЛЕНИЯ установлен. Модули должны сбрасывать РАЗРЕШЕНИЕ_РАСЩЕПЛЕНИЯ при условии: СИСТЕМНЫЙ_СБРОС ¦ УСТРОЙСТВО_МАГИСТРАЛИ ¦ ВКЛЮЧЕНИЕ_ПИТАНИЯ. Процесс конфигурации не должен устанавливать РАЗРЕШЕНИЕ_РАСЩЕПЛЕНИЯ до: во всех модулях разряд ПРИНИМАЮЩИЙ_РАСЩЕПЛЕНИЕ установлен.

7.2.4.2.2 **Логический регистр управления модулем**

Количество теговых линий

LSB+4, LSB+5 и LSB+6 должны быть трехразрядным полем, определяющим количество линий TG[]★, которые должны быть активны. Нулевое значение поля должно соответствовать одной линии TG0★, а значение семь (двоичная), записанное в это поле, будет соответствовать восьми теговым линиям TG[7 . . . 0]★.

РАЗРЕШЕНИЕ_ТЕГОВ

LSB−3 должен быть РАЗРЕШЕНИЕ_ТЕГОВ. Модули должны разрешать проверку четности по TG[]★ при условии: РАЗРЕШЕНИЕ_ТЕГОВ установлен. Модули должны сбрасывать РАЗРЕШЕНИЕ_ТЕГОВ при условии: СИСТЕМНЫЙ_СБРОС ¦ УСТРОЙСТВО_МАГИСТРАЛИ ¦ ВКЛЮЧЕНИЕ_ПИТАНИЯ.

РАЗРЕШЕНИЕ_СООБЩЕНИЯ_ЧЕТНОСТИ

LSB+2 должен быть РАЗРЕШЕНИЕ_СООБЩЕНИЯ_ЧЕТНОСТИ. Модули должны выставлять be★ и te★ при возникновении ошибки четности при условии: РАЗРЕШЕНИЕ_СООБЩЕНИЯ_ЧЕТНОСТИ установлен. Модули должны сбрасывать РАЗРЕШЕНИЕ_СООБЩЕНИЯ_ЧЕТНОСТИ при условии: СИСТЕМНЫЙ_СБРОС ¦ УСТРОЙСТВО_МАГИСТРАЛИ ¦ ВКЛЮЧЕНИЕ_ПИТАНИЯ.

РАЗРЕШЕНИЕ_ВЫСОКОЙ_СКОРОСТИ

LSB+1 должен быть РАЗРЕШЕНИЕ_ВЫСОКОЙ_СКОРОСТИ. Модули должны сбрасывать РАЗРЕШЕНИЕ_ВЫСОКОЙ_СКОРОСТИ при условии: СИСТЕМНЫЙ_СБРОС ¦ УСТРОЙСТВО_МАГИСТРАЛИ ¦ ВКЛЮЧЕНИЕ_ПИТАНИЯ. Процесс конфигурации может установить РАЗРЕШЕНИЕ_ВЫСОКОЙ_СКОРОСТИ в данном модуле, если он (процесс) определяет высокую и низкую скорости пакета для сегмента магистрали и если модуль поддерживает высокую скорость пакета. Процесс конфигурации должен инициализировать поле высокой скорости пакета в логическом регистре общего управления для указания на принятую на магистрали высокую скорость пакета перед установкой РАЗРЕШЕНИЕ_ВЫСОКОЙ_СКОРОСТИ. Если РАЗРЕШЕНИЕ_ВЫСОКОЙ_СКОРОСТИ сброшен, модуль должен выставлять ts★ (ca0★) для передач в пакетном режиме.

РАЗРЕШЕНИЕ_ЗАДАТЧИКА

LSB должен быть РАЗРЕШЕНИЕ_ЗАДАТЧИКА. Если РАЗРЕШЕНИЕ_ЗАДАТЧИКА установлен, модуль может устанавливать как ЗАПРОС_РАСПРЕДЕЛЕННОГО_АРБИТРАЖА, так и (rq0★ ¦ rq1★). Модули, которые не имеют способности быть системными, должны сбрасывать РАЗРЕШЕНИЕ ЗАДАТЧИКА при условии: СИСТЕМНЫЙ_СБРОС ¦ УСТРОЙСТВО_МАГИСТРАЛИ ¦ ВКЛЮЧЕНИЕ_ПИТАНИЯ. Модули, имеющие способность быть системными, устанавливают и сбрасывают этот разряд, как определено в соответствующем профильном стандарте в Р 896.2.

7.2.4.2.3 **Регистр задержки распространения магистрали**

Шесть младших разрядов регистра задержки распространения магистрали должны соответствовать максимальному значению задержки распространения сигнала по всей длине логической магистрали. Значение, записанное в это поле, должно быть выражено в шагах по 2★★(-32) с (≈233 пс). Регистр должен быть установлен в значение 3F (шестнадцатиричное), что соответствует 14,7 нс при условии: СИСТЕМНЫЙ_СБРОС | УСТРОЙСТВО_МАГИСТРАЛИ | ВКЛЮЧЕНИЕ_ПИТАНИЯ.

Модули должны гарантировать, что изменение этой величины не повлечет за собой каких-либо нарушений протоколов арбитражных сообщений и параллельного обмена.

7.2.4.2.4 **Регистр времени завершения соревнования**

Двенадцать младших разрядов регистра времени завершения соревнования должны соответствовать времени для наихудшего случая, которую модуль использует для определения его времени завершения соревнования в течение передачи арбитражного сообщения t_pd» — величина, содержащаяся в регистре задержки распространения магистрали, как определено в 7.2.4.2.3. t_int — величина, содержащаяся в регистре внутренней задержки соревнования, как определено в 7.2.4.1.2. t_ext — наибольшая из величин в регистрах внутренней задержки соревнования всех других модулей системы. Следующее уравнение может быть использовано для вычисления времени завершения:

t_a=10 ★ t_pd» +3 ★ t_int +5 ★ t_ext

Необходимо заметить, что в зависимости от номера соревнования, используемого модулем, позиции его разъема и оборудования логики соревнования действительный наихудший случай времени завершения может быть значительно меньше, чем величина, данная этим уравнением. Система должна программировать этот регистр значением равным или большим, чем действительное время завершения, требуемое модулем в наихудшем случае.

Значение, записанное в это поле, должно быть выражено в шагах по 2★★(-32) с (≈233 пс). Регистр должен быть установлен в значение 600 (шестнадцатиричное) или 1536 (десятичное), что соответствует 358 нс при условии: СИСТЕМНЫЙ_СБРОС | УСТРОЙСТВО_МАГИСТРАЛИ | ВКЛЮЧЕНИЕ_ПИТАНИЯ.

7.2.4.2.5 **Регистр тайм-аута передачи**

Регистр тайм-аута передачи должен содержать величину, соответствующую времени тайм-аута передачи, используемую задатчиком для определения, когда он устанавливает ТАЙМ-АУТ_ПЕРЕДАЧИ, как определено в 6.2. Значение, записанное в это поле, должно быть выражено в шагах по 2★★(-32) с (≈233 пс). Регистр должен быть установлен в значение 80000 (шестнадцатиричное) или 524288 (десятичное), что соответствует 122 мкс при условии: СИСТЕМНЫЙ_СБРОС | УСТРОЙСТВО_МАГИСТРАЛИ | ВКЛЮЧЕНИЕ_ПИТАНИЯ. Нулевое значение должно запрещать тайм-аут передачи.

7.2.4.2.6 **Маска выбора прохождения сообщения**

Биты от LSB+1 до LSB+63 в регистре маски выбора прохождения сообщения устанавливаются для разрешения модулям получать определенные сообщения через их широковещательный почтовый ящик, как определено в 9.2. В LSB должен быть всегда записан ноль. При чтении LSB должен всегда возвращать ноль.

Регистр должен быть очищен при условии: СИСТЕМНЫЙ_СБРОС | УСТРОЙСТВО_МАГИСТРАЛИ. НЕ очищается при условии: ВКЛЮЧЕНИЕ_ПИТАНИЯ.

8 КЕШ-КОГЕРЕНТНОСТЬ

8.1 **Описание**

Архитектура с разделяемой памятью состоит из некоторого количества процессоров, которые разделяют память, доступную через использование ФБ+. Все процессоры имеют ассоциированные кеш-памяти для выполнения трех целей.

1) обеспечить процессор более высокой скоростью доступа к данным,

2) уменьшить конкуренцию использования магистрали,

3) увеличить эффективность магистрали.

Для достижения первой цели может быть использован практически любой тип кеш-памяти. Для достижения второй цели может быть использован кеш с обратным копированием. Кеш с обратным копированием имеет такое свойство, что данные, записанные из процессора, записываются только в кеш. Данные записываются в разделяемую память, только если строка кеша, содержащая модифицированный вход, требуется в ответ на непопадание в кеш. Таким образом, кеш с обратным

копированием уменьшает количество обращений к магистрали как на чтение, так и на запись. Для достижения третьей цели, в случае отсутствия в кеш, из памяти восстанавливается целый блок данных, хотя в действительности процессору было необходимо только одно слово. Таким образом, случайные обращения процессора превращаются в эффективные блоковые передачи. Этот блок данных называется строка кеша. Размер строки кеша в системах ФБ+ зафиксирован и равен 64 байтам. Строка кеша должна быть выровнена по модулю 64 в байтовом адресном пространстве и всегда передаваться одной блочной передачей. Передачи частей строки кеша не допускаются.

Говорят, что строка кеша действительна, если кеш содержит самую последнюю копию (соответствующую последнему изменению) системной области. Кеш-память с действительной строкой имеет системное разрешение для допуска процессора-владельца кеша к чтению этой действительной строки в частном порядке (без выхода на магистраль). Кеш с действительной строкой должен информировать остальные модули о существовании ее копии. Строка кеша может быть либо действительной, либо недействительной.

Строка кеша может быть действительна во многих кешах системы. Строка кеша называется исключительной, если эта строка кеша действительна и если этому кешу даны гарантии, что нет других действительных кешированных копий в любом другом кеше системы. Кеш с исключительной строкой имеет системное разрешение присваивать атрибут модифицирования в любое время Действительная строка кеша может быть либо исключительной, либо разделяемой.

Строка кеша называется модифицированной, если копия этой строки в кеше заменяет копию в разделяемой памяти. Кеш с модифицированной строкой имеет системное разрешение для допуска процессора-владельца кеша к чтению или записи строки кеша в частном порядке (без выхода на магистраль). Кеш с модифицированной строкой должен реагировать на передачи как посредник. Кеш с модифицированной строкой должен обновить содержимое разделяемой памяти до того момента, когда он может отказаться от атрибута модифицированный. Исключительная строка кеша может быть либо модифицированной, либо немодифицированной.

Протокол кеш-когерентности, описанный и определенный в данной главе, гарантирует, что все модули соблюдают один и тот же порядок при модификации данной строки кеша.

Не все адреса в системе могут быть кешированы. Модель, показанная на рис. 8—1, не включает в себя внутреннее содержание блоков и другие детали исполнения. Некоторое оборудование может иметь свою собственную память в дополнение к разделяемой памяти, несколько процессоров на один кеш, несколько кешей, все три типа модулей на одной плате и так далее. Процессорный модуль может быть любым активным пользователем памяти, таким как микропроцессор, несколько независимых портов памяти, обслуживающих один процессор, несколько независимых микропроцессоров, контроллер ввода/вывода с прямым доступом к памяти, еще один иерархический кеш-уровень и так далее.

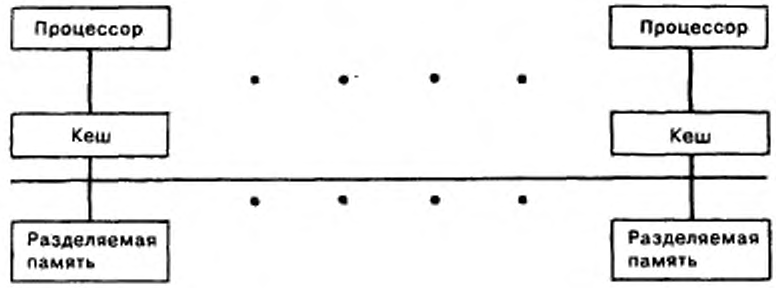

Рисунок 8—1 — Архитектура с разделяемой памятью

8.1.1 Атрибуты кеша

Когерентность данных в нескольких кешах основана на следующих атрибутах когерентности кеша: недействительный, разделяемый немодифицированный, исключительный немодифицированный и исключительный модифицированный. Только один из этих кеш-атрибутов может быть действителен для строки кеша в данном модуле и в данное время. Эти четыре атрибута заключают в себе состояние строки кеша.

Атрибут «недействительный» истинен для строки кеша, если нет обновленной копии системной памяти в кеше модуля. Все строки кеша модуля помечаются как недействительные, когда происходит системный сброс.

Атрибут «разделяемый немодифицированный» истинен для строки кеша, если кеш модуля содержит обновленную копию системной памяти и модуль предполагает, что такие копии содержатся также и в кешах других модулей. Модулям разрешается делать недействительными разделяемую немодифицированную копию строки кеша в любое время.

Атрибут «исключительный немодифицированный» истинен для строки кеша, если кеш модуля содержит обновленную копию системной памяти и модуль уверен, что действительной копии строки нет больше ни в одном кеше системы.

Атрибут «исключительный модифицированный» истинен для строки кеша, если кеш модуля содержит обновленную строку и эта копия — единственная в системе. Модуль несет ответственность за своевременное обновление содержимого памяти.

8.1.2 **Наблюдение за магистралью**

Всем кешам необходимо наблюдать, мониторировать все передачи когерентности кеша на системной магистрали. В определенных случаях модулям необходимо совершать совместные действия во время передач для поддержки когерентности кеша.

Кешу со строкой, которая не является «недействительной», требуется изменить статус его строки кеша, когда он отследит определенного рода передачи.

Кешу с «исключительной модифицированной» строкой требуется участвовать в доступе по чтению из модифицированной области, выставляя iv★ и обеспечивая строку кеша задатчику вместо системной памяти. После чего кеш должен изменить статус строки кеша на «разделяемый немодифицированный» либо «недействительный».

Кеш, который наблюдает передачу модифицированного чтения, перевода в «недействительность», или недействительной записи, должен изменить статус строки кеша на «недействительную».

В определенных случаях кешу разрешено преобразовывать передачи, происходящие на магистрали, в широковещательные операции и ловить (т. е. перехватывать) данные. Это позволяет кешу получить разделяемую копию строки кеша без дополнительной передачи. Модули могут перехватывать: передачи разделяемого чтения, обратного копирования, недействительного чтения и разделяемого ответа.

Для перехвата передач модули выставляют bc★ для преобразования передачи в широковещательный режим, if★ для оповещения других модулей о наличии у них копии и изменения состояния строки кеша на «разделяемый немодифицированный».

Если в кеше находится «исключительная модифицированная» копия строки кеша и он хочет скопировать строку обратно в память и перехватывает передачу чтения этой строки, он должен участвовать в передаче и «подставить» строку кеша. Это позволит кешу изменить статус строки на «недействительную» и избежать передачи обратного копирования.

8.1.3 **Когерентность кеша при использовании соединенных передач**

8.1.3.1 **Изменения состояния**

Этот раздел описывает изменения состояния, которые являются возможными для каждого из четырех атрибутов когерентности.

Как показано на рис. 8—2, если строка кеша является «недействительной», то возможен переход в любое из трех других состояний.

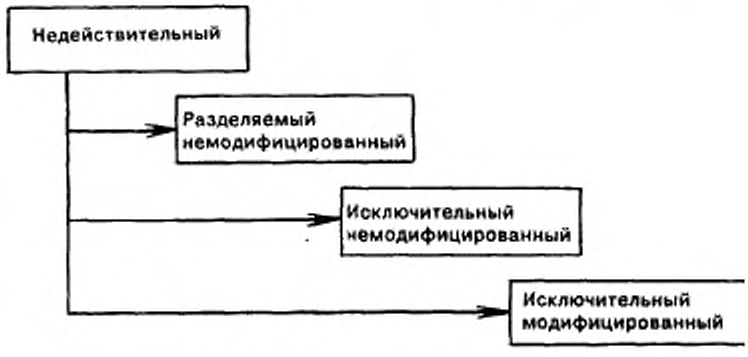

Рисунок 8—2 — Состояние «недействительный»

Модуль, инициирующий передачу типа «разделяемое чтение», которая успешно завершается выставлением TF★, переводит свою строку кеша в состояние «разделяемый немодифицированный». Если TF★ не был выставлен, состояние изменится на «исключительный немодифицированный».

Модуль, выставляющий bc★ и tf★ для перехвата разделяемого чтения, недействительного чтения или операции обратного копирования, изменит состояние строки своего кеша на «разделяемый немодифицированный».

Модуль, инициирующий передачу модифицированного чтения, в случае ее успешного завершения изменит состояние строки своего кеша на «исключительный модифицированный».

Как показано на рис. 8—3, если строка кеша является «разделяемой немодифицированной», то для нее возможен переход в состояния «недействительный» и «исключительный модифицированный».

Рисунок 8—3 — Состояние «разделяемый немодифицированный»

Любой модуль в любое время может изменить состояние строки кеша из «разделяемого немодифицированного» в «недействительный». Он делает это без использования магистральной передачи. Если модуль перехватил передачу разделяемого чтения или недействительного чтения и не выставил tf★, он изменяет состояние своей строки кеша на «недействительный». Если модуль перехватил передачу модифицированного чтения, недействительной записи или передачу недействительности, он изменяет состояние своей строки своего кеша на «недействительный».

Модуль, который инициирует передачу недействительности, в случае ее успешного завершения изменяет состояние своей строки кеша на «исключительный модифицированный».

Как показано на рис 8—4, если строка кеша является «исключительной немодифицированной», то возможен переход в любое их трех других состояний.

Рисунок 8—4 — Состояние «исключительный немодифицированный»

Любой модуль в любое время может изменить состояние строки кеша с «исключительного немодифицированного» на любое из трех других состояний без использования магистральной передачи.

Модуль, перехватывающий передачу разделяемого чтения или недействительного чтения и выставивший при этом tf★, изменяет состояние своей строки кеша на «разделяемый немодифицированный». Модуль, перехватывающий передачу разделяемого чтения или недействительного чтения и не выставивший при этом tf★, изменяет состояние своей строки кеша на «недействительный» Модуль, перехватывающий передачу модифицированного чтения, недействительной записи или передачи недействительности, изменяет состояние своей строки кеша на «недействительный».

Как показано на рис. 8—5, если строка кеша является «исключительной модифицированной», то возможен переход в состояния «недействительный» и «разделяемый немодифицированный». Модуль может инициировать изменение из «исключительного модифицированного» на «недействительный» или «разделяемый немодифицированный», используя передачу обратного копирования. Если модуль желает хранить копию этой строки кеша, он может перевести строку в состояние «разделяемый немодифицированный», если он выставит tf★ для оповещения других модулей о том, что он хранит копию.

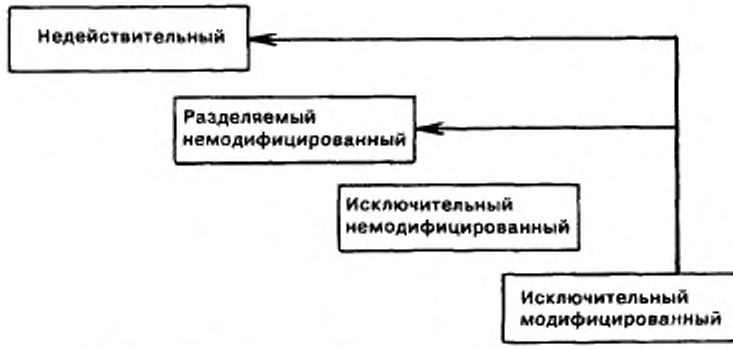

Рисунок 8—5 — Состояние «исключительный модифицированный»

Если модуль перехватывает передачу разделяемого чтения, недействительного чтения или модифицированного чтения, он должен выставить iv★ для того, чтобы участвовать в передаче и «подставить» данные вместо памяти. Модуль может хранить копию данных, если он выставляет tf★ во время передач разделяемого чтения или недействительного чтения.

Модули всегда изменяют состояние строки кеша на «недействительный», если они перехватывают передачу недействительного чтения.

8 1 3.2 **Процесс чтения с непопаданием**

Когда процессор производит операцию чтения, и требуемая область (адресного пространства) является «разделяемой немодифицированной», «исключительной немодифицированной» или «исключительной модифицированной», эту операцию называют чтение с попаданием. В случае чтения с попаданием нет необходимости в передачи по магистрали и процессор получает непосредственный доступ к данным. Если область «недействительна», эту операцию называют чтение с непопаданием

Когда происходит чтение с непопаданием, кеш запрашивает магистраль. Когда кеш получает магистраль, он производит передачу разделяемого чтения для запроса строки кеша, содержащей требуемые данные. Когда передача завершена, данные предоставляются процессору, сохраняются в кеше и для этой строки кеша устанавливается атрибут «разделяемый немодифицированный».

Есть специальный случай оптимизации доступов к памяти, которая не является на самом деле разделяемой. Если другой кеш не сигнализирует о том, что он содержит (аналогичную) строку кеша, выставляя во время передачи разделяемого чтения tf★, то для строки кеша может быть установлен атрибут «исключительный немодифицированный» Это позволяет процессору менять для этой строки атрибут на «исключительный модифицированный» без дальнейших передач по магистрали.

Атрибут «исключительный немодифицированный» для строки кеша может быть установлен только с использованием связанных передач во время выполнения передачи разделяемого чтения, заканчивающейся без выставления TF★. Модуль на магистрали может предупредить установление атрибута «исключительный немодифицированный», выставляя tf★ во время передач разделяемого чтения.

Если кеш наблюдает передачу чтения, которая может быть разделяемой и которая содержит данные для строки кеша, запрошенной его процессором, он может перехватить данные, и во время этого он должен выставить tf★. Полученные таким образом данные предоставляются процессору, сохраняются в кеше и для этой строки кеша устанавливается атрибут «разделяемый немодифицированный».

8.1.3.3 **Процесс записи с непопаданием**

Когда процессор производит операцию записи и требуемая область (адресного пространства) является «исключительной немодифицированной» или «исключительной модифицированной», эту операцию называют запись с попаданием, и нет необходимости в передаче по магистрали. Если эта область не маркирована как исключительная, эту операцию называют запись с непопаданием, даже если требуемая область является действительной. Когда происходит запись с непопаданием, кеш запрашивает магистраль. Когда кеш получает магистраль, возможны два случая. Если кеш содержит «разделяемую немодифицированную» копию строки кеша, он производит состоящую только из адреса передачу недействительности. Если кеш не содержит «разделяемую немодифицированную» копию строки кеша, он производит передачу типа модифицированное чтение. После передачи по магистрали данные берутся от процессора, объединяются с остальными в строке кеша, сохраняются в кеше и строка маркируется как «исключительная модифицированная».

Пока кеш ожидает становления задатчиком магистрали, он может перехватывать разделяемые передачи. Это позволит кешу использовать передачу только адреса в передаче недействительности вместо передачи модифицированного чтения.

8.1.3.4 **Процесс обратного копирования**

Для того, чтобы кеш мог обслуживать производимые процессором операции чтения и записи, необходимо наличие свободного места для хранения данных в кеш-памяти. После некоторого рабочего промежутка времени области адресов, доступные кешу для обслуживания отдельных операций чтения и записи с непопаданием, могут быть заняты. Когда это происходит, кеш должен будет освободить некоторое пространство кеш-памяти для его обслуживания непопадания в кеш. Области адресов, не являющиеся «исключительными модифицированными», могут быть снова использованы без передачи по магистрали. Области адресов, являющиеся «исключительными модифицированными», требуют передачу по магистрали для копирования содержания строки из кеш-памяти в разделяемую память для обновления системной информации до того, как строка кеша может быть использована для новой информации.

Процесс обратного копирования может быть использован в других случаях, где необходимо восстановление данных в памяти. Например, модуль может переписывать все содержимое кеша во время переключения контекста. Процедуры обслуживания, диагностики и «живого» удаления модуля также будут использовать процесс обратного копирования для обновления системной памяти.

Процесс начинается, когда кеш имеет «исключительную модифицированную» строку, которую он хочет сделать «недействительной» или «разделяемой немодифицированной». Кеш запрашивает магистраль и, когда становится задатчиком, инициирует передачу обратного копирования, которая копирует строку кеша обратно в память. Задатчик может держать копию строки, изменив ее «разделяемой немодифицированной», если он выставляет tf★ во время фазы соединения.

Пока кеш ждет момента, когда он станет задатчиком, какой-нибудь другой модуль может затребовать копию строки кеша. Если это произойдет, кеш должен участвовать и подставить эти данные во время передачи. В этот момент кеш может прекратить операцию обратного копирования.

8.1.3.5 **Оптимизация контроллера ввода/вывода**

Контроллеры ввода/вывода с прямым доступом к памяти могут читать и писать в когерентную память. Протокол требует, чтобы модули следовали протоколу когерентности кеша. Требуется, чтобы модули были оборудованы однострочным кешем, если им будет необходимо передавать отдельные кешевые строки. Два специальных типа передач, недействительное чтение и недействительная запись, доступны для снижения накладных расходов на передачи буфера ввода/вывода.

Передача недействительного чтения используется модулями тогда, когда у них нет действительной кешированной копии до передачи и не будет ее после. Передача недействительного чтения такая же, как и передача разделяемого чтения, за исключением того, что если TF★ не выставлен, то не будет разделяемых копий после передачи.

Передача недействительной записи используется модулями для записи новых данных во все байты строки кеша. Поскольку ни в одной строке кеша не останется старое содержимое, модулю не нужно предыдущее содержимое копии кеша. Передача недействительной записи делает все остальные копии строки кеша в системе недействительными и записывает новые данные в системную память. При успешном завершении такой передачи нет ни одного кеша в системе, содержащего действительную копию строки кеша.

Строго говоря, когерентность кеша не защищена при передаче типа «недействительная запись». Множество одновременных передач недействительной записи и смешивание передач недействительной записи с другими источниками модифицирования не будут представлять из себя какой-либо единообразный процесс для данной системы. Это означает, что чтение и запись областей в буфере ПДП во время действующего потока ПДП является программной ошибкой. Это означает, что передача недействительной записи предназначена для использования при инициализации системного буфера данных. Это означает, что она не предназначена для одновременного совместного обновления структуры данных разделяемой памяти.

8.1.3.6 **Примеры**

На рис. 8—6 показана модель системы, которая будет использоваться в различных примерах кешевой когерентности.

Следующие примеры показывают не типичную последовательность, а скорее детали протокола. Типичный случай не мог бы иметь одновременных передач по одному и тому же адресу, каждая передача могла бы завершиться без интерференции. Обычно разработчик системы не делал бы обновления одной строки одновременно несколькими процессорами без синхронизирующих переменных или других программных приемов для координации передач системного уровня.

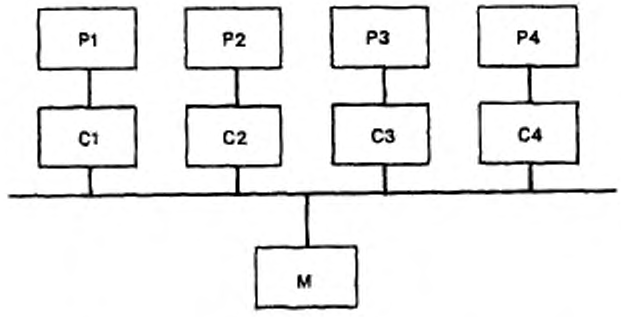

Рисунок 8—6 — Модель связанной когерентности кеша

8.1.3.6.1 **Один процессор читает и записывает область**

Этот пример иллюстрирует чтение с непопаданием, за которым следует запись с непопаданием. Пример начинается с состояния, когда все кеши недействительны. Действие происходит следующим образом:

1) процессор P1 читает область из кеша C1;

2) кеш C1 просматривает кешевый тег области и определяет, что строка кеша недействительна,

3) кеш C1 запрашивает магистраль, становится задатчиком и совершает операцию разделяемого чтения;

4) когда передача завершена, кеш C1 определяет, что передача была завершена без ошибки и что TF★ не был выставлен;

5) кеш C1 устанавливает атрибут «исключительный немодифицированный» для строки кеша;

6) кеш C1 предоставляет данные процессору P1;

7) процессор P1 может затем произвести любое число чтений, все из которых будут чтения с попаданием;

8) в конечном итоге процессор P1 производит запись в исключительную немодифицированную строку,

9) кеш C1 устанавливает атрибут когеренции для строки кеша «исключительный модифицированный»;

10) затем кеш C1 включает записанные данные в строку кеша;

11) все последующие записи, производимые процессором P1, являются записями с попаданием.

В конечном итоге кеш C1 возвратит данные в систему, совершив передачу обратного копирования или вклинившись в передачу чтения.

Как показано в этом примере, одна передача по магистрали — это все, что требуется для удовлетворения любого количества запросов на чтение от одного процессора к неразделяемой строке кеша. Только две передачи по магистрали требуются для удовлетворения любого количества записей от одного процессора в неразделяемую строку кеша. Статистически это наиболее частые операции в системе с разделяемой памятью.

8.1.3.6.2 **Все процессоры одновременно читают и записывают область**

Этот пример иллюстрирует, как четыре процессора одновременно читают область и потом записывают в нее. Пример начинается с состояния, когда все кеши недействительны. Действие происходит следующим образом:

1) все четыре процессора пытаются прочитать из своих кешей;

2) все четыре кеша просматривают кешевые тэги для интересующей области и определяют, что они недействительны;

3) четыре кеша запрашивают магистраль. Кеш C1 выигрывает арбитраж и становится задатчиком;

4) кеш C1 совершает передачу разделяемого чтения;

5) три остальных кеша выставляют tf★ и bc★ для получения данных, запрошенных кешем C1;

6) когда передача завершается, все кеши определяют, что передача была завершена без ошибки и что TF★ был выставлен;

7) затем все кеши маркируют строку кеша как «разделяемую немодифицированную» и предоставляют данные своим процессорам;

8) все процессоры одновременно записывают в строку кеша;

9) все четыре кеша запрашивают магистраль, C1 выигрывает арбитраж и становится задатчиком;

10) кеш C1 совершает передачу недействительности;

11) когда три остальных кеша отслеживают передачу недействительности, они делают строку кеша «недействительной»;

12) когда передача завершается и кеш C1 определяет, что операция была завершена без ошибок, он делает строку «исключительной модифицированной» и включает данные от процессора P1 в строку;

13) кеши C2—C4 после проигрыша арбитражного соревнования в (9) снова запрашивают магистраль, C2 выигрывает и становится задатчиком;

14) кеш C2 совершает передачу модифицированного чтения;

15) кеш C1 отслеживает передачу, вмешивается, маркирует строку как «недействительную» и выставляет свои данные вместо памяти;

16) когда передача завершается и кеш C2 определяет, что передача завершена без ошибки, он маркирует строку «исключительной модифицированной» и включает данные от процессора P2 в строку;

17) кеши C3 и C4 после проигрыша арбитражного соревнования в (13) снова запрашивают магистраль, C3 выигрывает и становится задатчиком;

18) кеш C3 совершает передачу модифицированного чтения;

19) кеш C2 отслеживает передачу, вмешивается, маркирует строку как «недействительную» и выставляет свои данные вместо памяти;

20) когда передача завершается и кеш C3 определяет, что передача завершена без ошибки, он маркирует строку «исключительной модифицированной» и включает данные от процессора P3 в строку;

21) кеш С4 после проигрыша арбитражного соревнования в (17) снова запрашивают магистраль, выигрывает и становится задатчиком;

22) кеш С4 совершает передачу модифицированного чтения;

23) кеш С3 отслеживает передачу, вмешивается, маркирует строку как «недействительную» и выставляет свои данные вместо памяти;

24) когда передача завершается и кеш С3 определяет, что передача завершена без ошибки, он маркирует строку «неключительной модифицированной» и включает данные от процессора Р4 в строку.

Пример показал, что возможны комбинированные разделяемые доступы, но что каждый исключительный доступ требует исключительной передачи по магистрали.

8.1.3.6.3 **Кеш-резидентная выборка и сложение**

Строка кеша с исключительным атрибутом может быть использована для элементарных, на базе кеша последовательностей типа считать-модифицировать-записать, для синхронизации без использования магистрали. Эти последовательности типа считать-модифицировать-записать могут включать в себя традиционные операции с памятью, такие как тестирование, установка или обмен, или более новые последовательности, такие как выборка-и-: сложение или сравнение, или обмен. Гарантируется, что все эти последовательности являются элементарными, если строка кеша, содержащая переменную синхронизации, имеет исключительный атрибут. Эти элементарные последовательности являются вопросом технологии локального модульного оборудования и не определены в этом стандарте.

Следующий пример иллюстрирует, как четыре процессора одновременно получают уникальное число из синхронизирующей переменной. Пример начинается с состояния, когда все кеши недействительны и область памяти содержит нули. Действие происходит следующим образом:

1) все четыре процессора пытаются прочитать из своих кешей требующих исключительное владение;

2) все четыре кеша просматривают кешевый тег области и определяют, что он недействительный;

3) четыре кеша запрашивают магистраль. Кеш С1 выигрывает арбитраж и становится задатчиком;

4) кеш С1 совершает передачу модифицированного чтения;

5) когда передача завершается и кеш С1 определяет, что передача была завершена без ошибки, он маркирует строку как «исключительную модифицированную» и дает запрошенное значение из памяти, которое для процессора Р1 равно 0. Процессор Р1 инкрементирует это значение;

6) кеш С1 включает 1 в строку кеша;

7) кеши С2—С4 после проигрыша арбитражного соревнования в (3) снова запрашивают магистраль, С2 выигрывает и становится задатчиком;

8) кеш С2 совершает передачу модифицированного чтения. Кеш С1 вмешивается, выставляет текущее значение и маркирует свою строку как «недействительную»;

9) когда передача завершается и кеш С2 определяет, что передача была завершена без ошибки, он маркирует строку как «исключительную модифицированную» и дает запрошенное значение из памяти, которое для процессора Р2 равно 1. Процессор Р2 инкрементирует это значение;

10) кеш С2 включает 2 в строку кеша;

11) кеши С3 и С4 после проигрыша арбитражного соревнования в (7) снова запрашивают магистраль, С3 выигрывает и становится задатчиком;

12) кеш С3 совершает передачу модифицированного чтения. Кеш С2 вмешивается, выставляет текущее значение и маркирует свою строку как «недействительную»;

13) когда передача завершается и кеш С3 определяет, что передача была завершена без ошибки, он маркирует строку как «исключительную модифицированную» и дает запрошенное значение из памяти, которое для процессора Р3 равно 2. Процессор Р3 инкрементирует это значение:

14) кеш С3 включает 3 в строку кеша;

15) кеш С4 после проигрыша арбитражного соревнования в (11) снова запрашивают магистраль, выигрывает и становится задатчиком;

16) кеш С4 совершает передачу модифицированного чтения. Кеш С3 вмешивается, выставляет текущее значение и маркирует свою строку как «недействительную»;

17) когда передача завершается и кеш С4 определяет, что передача была завершена без ошибки, он маркирует строку как «исключительную модифицированную» и дает запрошенное значение из памяти, которое для процессора Р4 равно 3. Процессор Р4 инкрементирует это значение;

18) кеш С4 включает 4 в строку кеша.

Как показано в этом примере, возможно координировать N процессоров с N передачами по магистрали.

8.1.4 **Когерентность кеша при использовании расщепленных передач в пределах одного сегмента магистрали**

Секция 8.1.4 описывает использование расщепленных передач в кеш-когерентной системе в пределах одного сегмента магистрали. Расщепленные передачи используются тогда, когда время доступа к модуль больше, чем время доступа к магистрали.

Нет необходимости в том, чтобы все модули были способны к расщепленным передачам. Расщепленные передачи должны быть разрешены установкой бита разрешения расщепления в логическом регистре общего управления, как это описано в 7.2. В общем случае, расщепленные передачи будут разрешены, если только все модули в системе способны работать с расщепленными передачами. Если модуль запрошен для поддержки расщепленной передачи, но он не способен ее поддержать, он выставляет te★.

Расщепленные передачи используются только тогда, когда в них есть необходимость. Каждый модуль, который может хотеть расщепить передачу, проверяет, является ли это необходимым, перед тем как сделать передачу расщепленной.

8.1.4.1 **Расщепленные передачи**

Модули кеша и модули памяти определяют, есть ли необходимость в расщеплении, декодируя адрес и команду в каждой кеш-когерентной операции чтения, отслеженной ими. Если модуль отвечает на этот адрес, но не может ответить немедленно, он выставляет sr★.

Модуль памяти может выставлять sr★ для того, чтобы передачи становились расщепленными. Если модуль памяти не наблюдает SW★, он должен сгенерировать разделяемый ответ для передачи разделяемого или недействительного чтения и модифицированный ответ для команды модифицированного чтения. Задатчик может наблюдать IV★ и выставлять sw★. Если модуль памяти наблюдает SW★, он не должен генерировать никакой ответной передачи. Кеш, выставивший iv★, может выставить также sr★ для того, чтобы вызвать расщепленное вмешательство. Кеш, выставивший iv★ и обнаруживший SR★, должен использовать расщепленное вмешательство или должен выставить te★.

8.1.4.1.1 **Разделяемый отклик**

Если модуль расщепляет передачу разделяемого или недействительного чтения, он должен в конце ответить передачей разделяемого ответа.

Передача разделяемого ответа может быть перехвачена любым кешем. Память должна перехватить данные из передачи разделяемого ответа.

Если нет других кешевых сигналов, от кеша, имеющего строку кеша, или от кеша, перехватывающего передачу разделяемого ответа выставлением tf★, строка в кеше запросчика может быть помечена как исключительная немодифицированная. Это позволяет процессору обновить строку кеша и сделать ее исключительной модифицированной без дальнейших передач по магистрали. Модули могут предотвратить перевод строки в исключительную модифицированную, выставив tf★ во время передачи разделяемого чтения или разделяемого ответа.

8.1.4.1.2 **Модифицированный ответ**

Если модуль расщепляет передачу модифицированного чтения, он должен в конце ответить передачей модифицированного ответа.

Передача модифицированного ответа не может быть перехвачена. Модуль, который первоначально генерирует передачу модифицированного чтения или недействительности и впоследствии получает модифицированный ответ, изменяет состояние своей кешевой строки на исключительное модифицированное.

8.1.4.2 **Одна незавершенная передача на строку кеша**

Модулям, расщепляющим передачу, необходимо выставить wt★, если какой-нибудь другой модуль инициирует запрос на передачу к той же строке кеша. Это налагает ограничение в одну незавершенную передачу на строку кеша. Когда модуль, инициирующий передачу, получает статус ожидания, ему необходимо ждать до тех пор, пока разделяемый или модифицированный ответ не будет отслежен строкой кеша, и тогда он может повторить свой запрос.

В случае передачи разделяемого ответа модуль может перехватить данные и удовлетворить свой запрос без передачи по магистрали.

Этот протокол может быть рассмотрен как очередь запросов к отдельной строке кеша. Модуль, возглавляющий очередь, получает расщепленный статус в ответ на свой запрос, и это гарантирует ему завершение. Все другие модули ждут завершения первой передачи модулями. Потом все ожидающие модули соревнуются за получение магистрали. Процесс арбитража выбирает один из модулей следующим «возглавляющим очередь». Все проигравшие модули отслеживают передачу выигравшего модуля и возвращаются в состояние ожидания без напрасной потери передачи по магистрали.

Этот протокол не обеспечивает определенного порядка в очереди. Он (порядок) зависит от стратегии арбитража. Неправильная временная последовательность может привести к неопределенному «голоданию» некоторых модулей.

Все исключительные передачи в очереди должны управляться индивидуально. Находясь в ожидании, модули должны производить операцию перехода в недействительность, если они отслеживают успешные передачи недействительности.

8.1.4.3 **Атрибуты запросчика и ответчика**

Модули, имеющие способность к расщеплению, используют набор атрибутов ответчика, который они связывают со строкой кеша, для которой у них есть ответ. Модули устанавливают «разделяемый ответчик», если они расщепляют передачи разделяемого или недействительного чтения. Модули устанавливают «исключительный ответчик», если они расщепляют передачи модифицированного чтения.

Модули, имеющие способность к расщеплению, используют набор атрибутов запросчика, который они связывают со строкой кеша, для которой должен быть дан ответ. Модули устанавливают «разделяемый запросчик», если они инициируют передачу разделяемого или недействительного чтения, которая расщепляется другим модулем. Модули устанавливают «исключительный запросчик», если они инициируют передачу модифицированного чтения, которая расщепляется другим модулем.

Модули устанавливают «ожидающий запросчик», если они получили во время передачи статус ожидания. Модули также могут устанавливать «ожидающий запросчик», если они отследили передачу, в которой выставлен SR★ или WT★.

8.1.4.4 **Оптимизация контроллера ввода/вывода**

Как в аналогичном случае, передача недействительной записи не гарантирует уникального, пригодного для всех систем порядка, который сохранял бы кешевую когерентность.

Передача недействительной записи отменяет любые атрибуты запросчика или ответчика, которые она встречает; любые такие модули должны повторить свои передачи.

Передача недействительной записи не может быть разделяемой, следовательно, не может быть отслежена никаким кешем.

Передача модифицированного чтения, за которой следует передача обратного копирования, используется для обновления определенных строк кеша по правилам кешевой когерентности. Если передача модифицированного чтения является расщепленной, данные возвращаются в передаче модифицированного ответа. Задатчик передачи модифицированного ответа не является контроллером ввода/вывода, так что другие передачи могут происходить между завершением передачи модифицированного отклика и передачей обратного копирования. Модуль должен отслеживать соответствие адресов для уверенности в соблюдении кешевой когерентности до завершения передачи обратного копирования.

8.1.4.5 **Примеры**

Рис. 8—6 вновь используется для показа модели системы, которая будет использована в примерах расщепленных передач кеш-когерентной системе в пределах одного сегмента магистрали. Память М расщепляет все передачи чтения. Кеши С1, С2 и С3 используют расщепленное вмешательство. Следующие примеры показывают не типичную последовательность, а скорее детали протокола.

8.1.4.5.1 **Один процессор читает и затем пишет область**

Этот пример иллюстрирует чтение с непопаданием, за которым следует запись с непопаданием. Пример начинается с состояния, когда все кеши недействительны. Действие происходит следующим образом:

1) процессор Р1 читает область из кеша С1;

2) кеш С1 просматривает кешевый тег этой области и определяет, что эта строка кеша недействительна;

3) кеш С1 запрашивает магистраль, становится задатчиком и затем совершает передачу разделяемого чтения;

4) память М расщепляет передачу и устанавливает ее атрибут «разделяемый ответчик» для запрошенной строки кеша;

5) когда передача завершается, кеш С1 определяет, что передача является расщепленной и устанавливает свой атрибут «разделяемый запросчик» для запрошенной строки кеша;

6) когда данные становятся доступны, память совершает передачу разделяемого ответа;

7) когда передача завершена, кеш С1 и память М определяют, что передача завершена без ошибки;

8) память М очищает свой атрибут «разделяемый ответчик»;

9) кеш С1 очищает свой атрибут «разделяемый запросчик» и, так как другие кеши не выставляли tf★, изменяет состояние строки кеша на исключительную немодифицированную;

10) кеш С1 предоставляет данные процессору Р1;

11) процессор может теперь совершить любое количество чтений, которые все будут чтениями с попаданием;

12) в конце концов процессор записывает в исключительную немодифицированную строку;

13) кеш С1 устанавливает состояние строки кеша в исключительную модифицированную;

14) кеш С1 присоединяет записанные данные к строке кеша;

15) все последующие записи процессора Р1 в строку являются записями с попаданием.

Как показано в этом примере, передача процессора расщепляется на две передачи по магистрали. Состояние строки кеша обновлено в завершении последовательности.

8.1.4.5.2 **Три процессора одновременно читают и пишут область**

Этот пример иллюстрирует одновременное чтение и затем запись области процессорами Р1, Р2 и Р3. Пример начинается с состояния, когда все кеши недействительны. Действие происходит следующим образом:

1) три процессора пытаются прочитать из своих кешей;

2) три кеша просматривают кешевый тег для области и определяют, что он недействителен;

3) три кеша потом запрашивают магистраль. Кеш С1 выигрывает арбитраж и становится задатчиком;

4) кеш С1 совершает передачу разделяемого чтения;

5) кеши С2 и С3 выставляют tf★ и bc★ для получения данных, запрошенных кешем С1. Память М выставляет sr★ для расщепления передачи;

6) когда передача завершается, все кеши определяют, что передача является расщепленной и поэтому не было передано никаких данных;

7) кеш С1 устанавливает свой атрибут «разделяемый запросчик», память М — атрибут «разделяемый ответчик», а остальные кеши — атрибуты «ожидающий запросчик»,

8) когда данные становятся доступными, память М совершает передачу разделяемого ответа. Кеши С2 и С3 выставляют tf★ и bc★ для получения данных, выставленных памятью М;

9) по окончании передачи модули определяют, что передача завершена без ошибки;

10) память М очищает свой атрибут «разделяемый ответчик»;

11) кеш С1 очищает свой атрибут «разделяемый запросчик» и изменяет состояние строки кеша на разделяемую немодифицированную;

12) другие кеши очищают свои атрибуты «ожидающий запросчик» и изменяют состояние своих кешевых строк на разделяемые немодифицированные;

13) кеши предоставляют данные своим процессорам;

14) три процессора одновременно пытаются записать в строку кеша;

15) три кеша потом запрашивают магистраль. Кеш С1 выигрывает арбитраж и становится задатчиком;

16) кеш С1 совершает передачу недействительности;

17) кеши С2 и С3 отслеживают передачу недействительности и изменяют состояние своих кешевых строк на недействительные;

18) когда передача завершается и кеш С1 определяет, что передача завершена без ошибки, он делает строку исключительной модифицированной и присоединяет записанные данные от процессора Р1 к строке кеша;

19) кеши С2 и С3 запрашивают магистраль, С2 выигрывает арбитраж и становится задатчиком;

20) кеш С2 совершает передачу модифицированного чтения,

21) память М выставляет sr★ для расщепления передачи. Кеш С1 отслеживает передачу, выставляет sr★ для расщепления передачи и выставляет iv★ для того, чтобы заставить кеш С2 выставить sw★ для оповещения памяти М о том, что память не обязана выставлять запрошенные данные. Кеш С2 устанавливает у себя атрибут «исключительный запросчик». Кеш С1 устанавливает у себя атрибут «исключительный ответчик». Кеш С3 оставляет свой атрибут «ожидающий запросчик»,

22) когда кеш С1 готов предоставить данные, он запрашивает магистраль и совершает передачу модифицированного ответа. Кеш С2 принимает данные;

23) кеш С2 устанавливает состояние строки кеша в исключительную модифицированную. Кеш С1 устанавливает состояние своей строки кеша в недействительную. Кеш С2 очищает свой атрибут «исключительный запросчик». Кеш С1 очищает свой атрибут «исключительный ответчик». Кеш С3 очищает свой атрибут «ожидающий запросчик»;

24) кеш С3 запрашивает шину и совершает передачу модифицированного чтения;

25) память М выставляет sr★ для расщепления передачи. Кеш С2 отслеживает передачу, выставляет sr★ для расщепления передачи и выставляет iv★ для того, чтобы заставить кеш С3 выставить sw★ для оповещения памяти М о том, что память не обязана выставлять запрошенные данные. Кеш С3 устанавливает у себя атрибут «исключительный запросчик». Кеш С2 устанавливает у себя атрибут «исключительный ответчик»;

26) когда кеш С2 готов предоставить данные, он запрашивает магистраль и совершает передачу модифицированного ответа. Кеш С3 принимает данные;

27) кеш С2 устанавливает состояние строки кеша в исключительную модифицированную. Кеш С1 устанавливает состояние своей строки кеша в недействительную. Кеш С2 очищает свой атрибут «исключительный запросчик». Кеш С1 очищает свой атрибут «исключительный ответчик»;

Пример показал, что возможно комбинировать разделяемые доступы, но что каждый исключительный доступ требует исключительной передачи по магистрали.

8.1.5 **Использование расщепленных передач для задержки окончаний недействительности**

Большие системы могут быть построены путем соединения сегмента магистрали с другими модулями через магистральные мосты. Эти магистральные мосты будут иметь медленный доступ к данным на удаленных сегментах магистрали, поэтому они используют расщепленные передачи для доступа к данным, как это было описано раньше. Кроме того, магистральные мосты имеют медленный доступ к кешевым тегам на удаленных сегментах магистрали. Этот раздел описывает использование расщепленных передач для того, чтобы гарантировать, что все удаленные кеши являются недействительными.

Архитектура с разделяемой памятью, использующая расщепленные передачи, представленная на рис. 8—7, состоит из архитектуры с разделяемой памятью, расширенной за счет модулей агентов кеша и агентов памяти.

Рисунок дан с намерением показать, что агенты кеша, агенты памяти и дополнительные модули подключены с использованием некоторой технологии, которая может быть весьма независимой от этой спецификации. Рисунок не претендует показать компоновку на платах или в корпусах, а так же другие детали оборудования. Некоторые виды оборудования могут иметь модули как агента кеша, так и агента памяти для оснащения кеш-когерентного магистрального моста. Некоторые виды оборудования могут использовать иерархическое кеширование с множеством уровней протоколов кешевой когерентности на множестве магистральных сегментах P896.1. Некоторые виды оборудования могут использовать кеш-когерентные мосты для других магистральных стандартов.

8.1.5.1 **Агенты кеша**

Говорят, что модуль является агентом кеша, если он использует расщепленные передачи для принятия на себя всех прав и ответственности некоторого количества удаленных кеш-модулей. В иерархической кеш-структуре агент кеша принимает на себя права и обязанности кеш-модулей, расположенных дальше от памяти в данном уровне иерархии.

Кеш-агенты должны поддерживать теги, содержащие состояние строки для каждой строки кеша, содержащейся в любом из кешей, за который агент отвечает. Ничего в этой спецификации не оговаривается по поводу того, что кеш-агент должен хранить копию данных, которые проходят через него или хранятся в кешах или в кеш-агентах, подключенных через него. Кеш-агент может иметь сегмент данных в своем кеше.

Кеш-агент расщепляет передачу, если она имеет модифицированный атрибут для строки кеша и есть необходимость вмешательства, чтобы передать обновленную копию данных от удаленного модуля кеш-памяти.

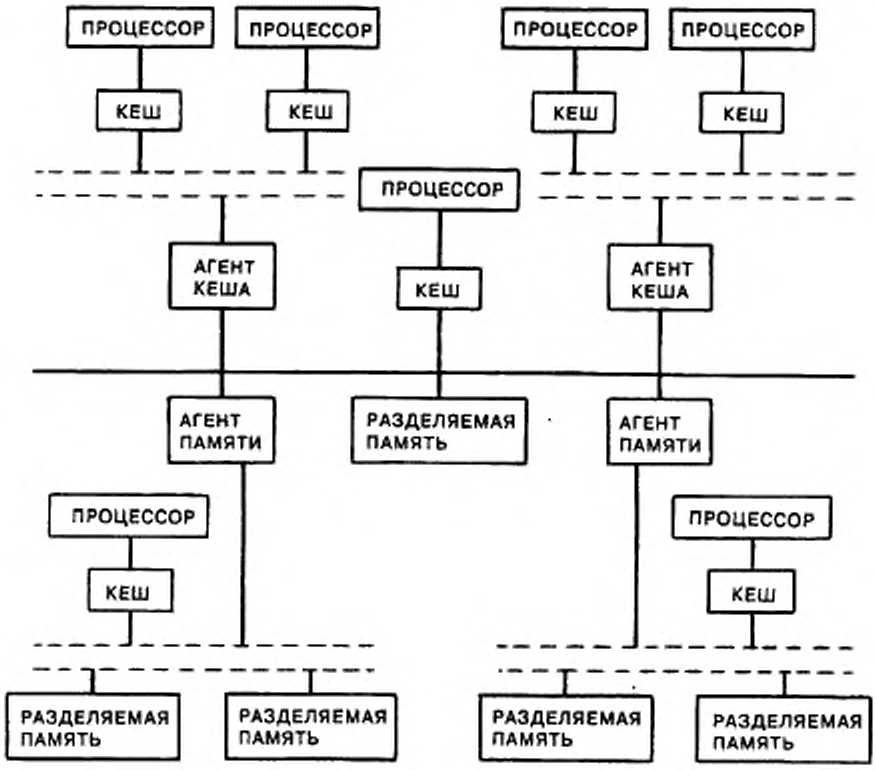

Рисунок 8—7 — Архитектура расщепленного ответа

Кеш-агент расщепляет передачу, когда он имеет действительный атрибут для строки и не может немедленно сделать недействительными строки в удаленных кеш-модулях, за которые он отвечает. Кеш-агент также выставляет tf★ для индикации, что некоторые кеши, иерархически расположенные дальше от памяти, могут еще быть действительными.

Передача, участвовавшая в арбитраже за сегмент магистрали и выигравшая его, будет иметь для кеш-агента приоритет над конфликтующей передачей по тому же адресу, пришедшей от любого удаленного кеша.

8 1.5.2 **Агенты памяти**

Говорят, что модуль является агентом памяти, если он использует расщепленные передачи для принятия на себя всех прав и ответственности некоторого количества удаленных модулей памяти. В иерархической кеш-структуре агент памяти принимает на себя права и обязанности модулей памяти, расположенных иерархически ближе к памяти.

Каждая область памяти, доступная протоколу кеш-когерентности, должна иметь только один выбранный модуль, который обслуживает доступы к этой области на каждом сегменте магистрали

Агент памяти должен быть способен получать данные из исключительной модифицированной строки кеша, когда удаленный кеш требует этого. Если строка является исключительной модифицированной, агент памяти должен использовать запрос чтения для получения данных. Если строка кеша не является исключительной модифицированной, любой запрос чтения агента памяти потерпит неудачу, поскольку не будет вмешательства. Если строка исключительная немодифицированная, агент памяти может изменить состояние строки в любое другое без видимой магистральной передачи. Агенты памяти хранят путь к каждой строке (из тех, за которые они отвечают), имеющей состояние исключительной модифицированной в иерархии выше нее, установкой атрибута строки памяти модифицированный для этой строки. Агент памяти должен предотвращать случаи появления строк кеша с исключительным немодифицированным состоянием, настолько чтобы он может отслеживать все изменения состояний на исключительный модифицированный.

Ничего в этой спецификации не оговаривается по поводу того, что агент памяти должен хранить копию данных, которые проходят через него или хранятся в кешах или в кеш-агентах, подключенных через него. Агент памяти может иметь сегмент данных в своем кеше для увеличения скорости передач.

Агент памяти расщепляет передачу для назначения разрешения чтения и записи области, которой он управляет в пределах магистрального сегмента, на котором он находится. Право доступа на запись запрашивается для модифицированного чтения или передачи недействительности. Право доступа на запись предоставляется успешным завершением передачи и любой ассоциированной передачей расщепленного ответа. Когда сегмент магистрали имеет права доступа на чтение и запись, все кеши и кеш-агенты должны быть с недействительными, разделяемыми немодифицированными и исключительными модифицированными состояниями. Право доступа на чтение запрашивается для разделяемого или модифицированного чтения, обратного копирования с выставленным tf★ или недействительного чтения с выставленным tf★. Право доступа на чтение предоставляется успешным завершением передачи и любой ассоциированной передачей расщепленного ответа. Когда сегмент магистрали имеет право доступа только на чтение, все кеши и кеш-агенты должны быть с недействительными и разделяемыми немодифицированными состояниями.

Агент памяти должен передавать внешние передачи недействительности к сегменту магистрали, если сегмент имеет право доступа на чтение. Агент памяти должен передавать внешние вмешательства к сегменту магистрали, если сегмент имеет право доступа на запись.

8.1 5.3 **Расщепленные передачи**

Модули кешевых агентов и агентов памяти определяют, необходимо ли им задерживать свои подтверждения недействительности, декодируя адрес и команду каждой кеш-когерентной передачи, которую они отслеживают. Если модуль является ответственным за адрес и не может ответить немедленно, он выставляет sr★ и tf★, как описано ниже.

Если более чем один модуль расщепляет какую-то конкретную передачу, все модули, расщепляющие передачу, расщепляют любые передачи ответа, которые они отслеживают, до тех пор, пока они не завершат свои ответы. В конце концов все эти модули выдают передачу ответа или позволяют завершиться отслеженным передачам ответа без дальнейшего расщепления.

Следующие кеш-когерентные передачи могут иметь задержанные подтверждения недействительности: модифицированное чтение, недействительность, недействительная запись и модифицированный ответ..

8.1.5.3.1 **Модифицированное чтение**

Любой модуль может задерживать подтверждение недействительности, выставляя tf ★для передачи модифицированного чтения.

Если SR★ также выставлен модулем, который будет выполнять копию данных, тогда модифицированное чтение является расщепленным как для подтверждения доступа, так и для подтверждения недействительности. В этом случае каждый модуль, выставивший tf★, должен в конечном итоге произвести передачу ответа или разрешить завершение передачи отслеженного ответа без дальнейшего расщепления.

Если SR★ не выставлен, но выставлен TF★, модифицированное чтение дает действительную копию связанных данных, но за этим должна следовать передача недействительности для получения гарантированного подтверждения недействительности. В этом случае каждый модуль, выставивший tf★, не имеет никаких обязательств до тех пор, пока не произойдет передача недействительности.

8.1.5.3.2 **Недействительность**

Любой модуль может задерживать подтверждение недействительности выставлением sr★ во время передачи недействительности.

8.1.5.3.3 **Недействительная запись**

Как и в бывшем ранее случае, передача недействительной записи не гарантирует уникального, пригодного для всех систем порядка, который сохранял бы кешевую когерентность. Однако все ранее действительные кеши обязаны обозначить завершение перехода в недействительность, так чтобы контроллер ПДП мог ждать, пока все данные, передаваемые ПДП, станут доступны для обозрения перед тем, как он пошлет блок их состояния.

Любой модуль может задерживать подтверждение недействительности для передачи недействительной записи, выставляя sr★.

8.1.5.3.4 **Модифицированный ответ**

Если модуль расщепляет передачу модифицированного чтения, недействительности или модифицированной записи, он в конечном итоге должен дать ответ, используя передачу модифицированного ответа, или разрешить завершиться модифицированному ответу других модулей без дальнейшего расщепления.

Кеш-агенты или агенты памяти могут расщеплять модифицированный ответ сигналом sr★ для строки кеша, про которую еще не известно, является ли она недействительной. Кеш-агенты, расщепляющие модифицированный ответ, также выставляют tf★.

Передача модифицированного ответа не может быть перехвачена. Модуль, который первоначально инициирует передачу модифицированного чтения или недействительности, и в завершении получивший модифицированный ответ без SR★, изменяет состояние своей строки кеша на исключительный модифицированный.

8.1.5.4 **Атрибуты запрашиваемого и отвечающего**

Кеш-агенты и агенты памяти используют атрибуты запрашивающего и отвечающего как это было описано ранее. В дополнение, модули устанавливают «недействительность отвечающего», если они расщепляют передачи недействительности. Модули устанавливают «недействительную запись отвечающего», если они расщепляют передачи недействительной записи. Модули устанавливают «исключительность запрашивающего», если они инициируют передачу недействительности, которая расщеплена другим модулем. Модули устанавливают «недействительную запись запрашивающего», если они инициируют передачу недействительной записи, которая расщеплена другим модулем.

8.1.5.5 **Множество незавершенных передач на строку кеша**

Система может состоять из нескольких сегментов магистрали, совместно использующих одно разделяемое пространство памяти и с оборудованием для кеш-когерентности с аппаратными протоколами с использованием модулей-мостов, с одним или более кеш-агентами и агентами памяти. В такой системе сегменты магистрали работают с независимой друг от друга синхронизацией. Несмотря на то, что протоколы допускают только одну незавершенную передачу на строку кеша в каждом сегменте магистрали, возможно наличие множества одновременно незавершенных передач на строку кеша во всей системе.

Использование иерархических кешей с принципом «включения» вызывает столкновения всех возможных конфликтующих, одновременных передачи. Система с иерархическими кешами проявляет принцип «включения», если для каждой строки кеша существует единственный путь от кеша к памяти, и каждый кеш-агент содержит сумму атрибутов для всех строк, за которые он ответственен. Во всех представленных примерах встречается принцип «включающий».

Каждая передача может быть представлена как приходящая от кеша или кеш-агента и направленная в сторону памяти, или от памяти или агента памяти и направленная в сторону кешей.

Некоторые типы передач могут быть пропущены кеш-агентами мимо памяти. Такими являются следующие передачи:

1) передача модифицированного или разделяемого чтения для чтения модифицированной, обновленной копии;

2) передача недействительности для перевода всех удаленных кешей в недействительное состояние;

3) передача недействительной записи для перевода всех удаленных кешей в недействительное состояние и всех требующих повторения незавершенных передач.

Передача любого типа может быть пропущена агентом памяти в сторону памяти. Все передачи, направленные в сторону памяти, подразумевают, что сегмент магистрали позади них не имеет больше исключительных атрибутов. Передачи, следующие по направлению к памяти, могут препятствовать разветвлению передач, следующих от памяти, для обеспечения того, чтобы все удаленные кеши были недействительны.

Передачи от множественных кешей или кеш-агентов к одному определенному сегменту магистрали управляются протоколами арбитража и ожидания. Передачи, следующие в одном и том же направлении, обслуживаются по порядку и никогда не могут сталкиваться. Передачи, следующие в противоположных направлениях, могут сталкиваться внутри магистральных мостов, что приводит к образованию столкновений недействительности. Такие столкновения всегда разрешаются в следующем порядке: направляющаяся от памяти передача пропускается первой, а передача, направляющаяся в сторону памяти, ждет до тех пор, пока не увидит завершения победительницы.

Передачи чтения, направляющиеся от памяти, всегда преследуют исключительный атрибут. Если они сталкиваются с передачей, направляющейся в сторону памяти, эта передача должна иметь обновленную копию данных во время передачи. Это обстоятельство завершает передачу разделяемого чтения. Передача модифицированного чтения получает копию данных, но должна продолжаться в направлении от памяти как передача недействительности.

Передача недействительности, направляющаяся от памяти, может натолкнуться на сегмент магистрали с кешами или/или кеш-агентами, находящимися в состоянии расщепленной передачи. Агент памяти инициирует передачу недействительности без выставления wt★. Все кеши становятся недействительными без влияния на состояние их расщепленной передачи. Кеш-агенты удостоверяются, что их удаленные кеши в текущий момент переводятся в недействительное состояние, выставляют sr★, если какой-нибудь удаленный кеш еще не является недействительным. Кеш-агенты докладывают о завершении передачей модифицированного ответа. Все кеш-агенты следят за магистралью и выставляют sr★ и tf★, если переход в недействительность еще не завершен. Агент памяти следит за магистралью и выставляет sr★, чтобы не дать первоначальному инициатору передачи ее завершить. Когда агент памяти наблюдает передачу модифицированного ответа без TF★, он знает, что все кеши, подключенные через сегмент магистрали, гарантированно являются недействительными.

Передача недействительной записи, направляющаяся от памяти, может натолкнуться на сегмент магистрали с кешами и/или кеш-агентами, находящимися в процессе расщепленной передачи. Агент памяти инициирует передачу недействительной записи как только адресный цикл, без выставления wt★. Все кеши переходят в недействительное состояние, сбрасывая состояния любых незавершенных расщепленных передач. Кеш-агенты удостоверяются, что их удаленные кеши переведятся в недействительное состояние дальнейшим прохождением недействительной записи, выставляя sr,★ если некоторые удаленные кеши еще не являются недействительными. Кеш-агенты докладывают о завершении передачей модифицированного ответа. Все кеш-агенты следят за магистралью и выставляют sr★, если переход в недействительность еще не завершен. Агент памяти следит за магистралью. Когда агент памяти наблюдает передачу модифицированного ответа без sr★, он знает, что все кеши, подключенные через сегмент магистрали, гарантированно являются недействительными. После передачи модифицированного ответа без sr★ всем кешам с отмененными расщепленными передачами разрешается сделать новую попытку.

Передача недействительной записи может быть расщеплена любым количеством магистральных мостов. Если мост расщепляет передачу, то в конечном итоге он должен закончить передачей модифицированного ответа или разрешить другой передаче модифицированного ответа, проводимую другим мостом, закончить без дальнейшего расщепления.

Кеш-агент или агент памяти, выполняющий задержанный переход в недействительность, выставляет sr★ на команду передачи недействительности и в дальнейшем становится ответственным за выполнение передачи модифицированного ответа или за разрешение другой передачи модифицированного отклика, проводимую другим агентом, завершиться без выставления sr★.

Кеш-агент или агент памяти, имеющий обновленную копию строки кеша, выставляет sr★ на команду модифицированного чтения. Он становится в дальнейшем ответственным за выполнение передачи модифицированного ответа с обновленной копией.

Кеш-агент, выполняющий задержанный переход в недействительность, выставляет tf★ на команду модифицированного чтения. Если агент памяти выставляет sr★, тогда кеш-агент выполняет расщепленный переход в недействительность, как описано выше. Если ни один агент памяти не выставляет sr★, тогда задатчик после передачи модифицированного чтения выполняет передачу недействительности, содержащую только адрес, разрешая кеш-агенту ее расщепить выставлением sr★. Если передача недействительности следует непосредственно в том же цикле владения магистралью, тогда не возникает возможности для зацикливания.

8.1.5.6 **Примеры**

Рис. 8—8 показывает одну строку в модуле М, разделяемую десятью процессорами через десять кешей, четыре кеш-агента, четыре агента памяти и пять сегментов магистрали. Для понимания примера всем модулям даны иерархические номера. Все сегменты магистрали имеют метку поддерживающих их памяти или агентов памяти.

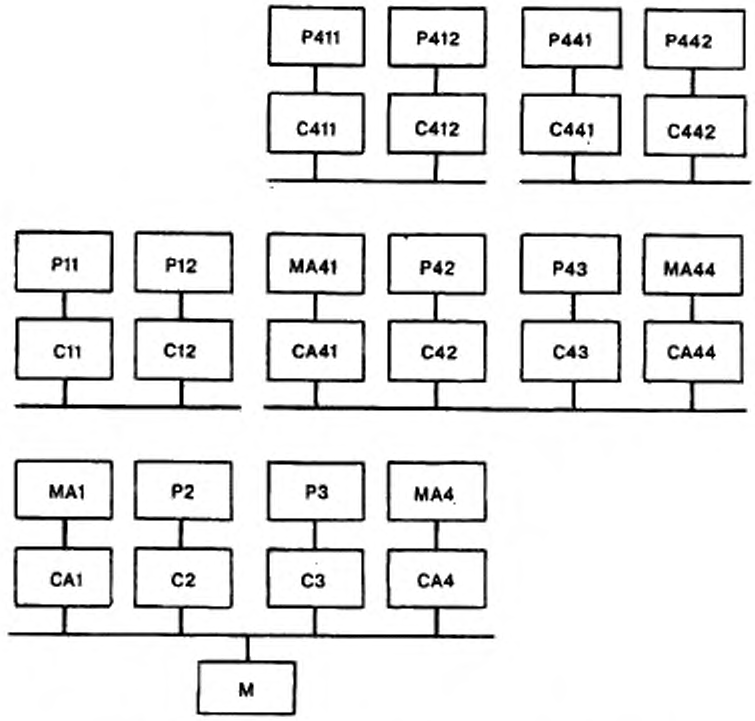

Рисунок 8—8 — Пример модели множества сегментов магистрали с расщепленными передачами

Топология примера должна быть достаточно большой для того, чтобы показать каждый случай неудачи в протоколе расщепленного ответа для разделяемой памяти В системе, представленной в примере, имеется множество копий любой области памяти, где бы она не находилась. Имеются конечный, средний и корневой сегменты магистрали.

Пример не предназначен для детального представления оборудования. Например, кеш-агент и агент памяти, являющиеся окончанием магистральных мостов, изображены соединенными связью от одной точки к другой, хотя возможны другие топологии.

Целью примера является показать не типичную последовательность, а детали протокола. Типичный случай не имел бы одновременные передачи по одному и тому же адресу; каждая передача должна завершиться без помех. Обычно разработчик системы не имел бы несколько процессоров, одновременно обновляющих содержимое строки без синхронизирующих переменных или других программных способов для координации передач системного уровня.

8.1.5.6.1 **Один процессор читает и записывает область**

Этот пример иллюстрирует чтение с непопаданием, за которым следует запись с непопаданием. В начале примера все кеши недействительны. Действие происходит следующим образом:

1) процессор P411 читает область из кеша C411. Кеш C411 проверяет тег строки кеша и определяет, что строка кеша недействительна;

2) кеш C411 запрашивает магистраль MA41, становится задатчиком и затем совершает передачу разделяемого чтения;

3) агент памяти MA41 расщепляет передачу, устанавливает свой атрибут разделяемого ответчика для требуемой строки кеша и передает запрос чтения кеш-агенту с неспецифицированным протоколом;

4) когда передача завершается, кеш C411 определяет, что передача была расщеплена и устанавливает свой атрибут разделяемого запросчика для требуемой строки кеша;

5) кеш-агент CA41 запрашивает магистраль MA4, становится задатчиком и затем совершает передачу разделяемого чтения;

6) агент памяти MA4 расщепляет передачу, устанавливает свой атрибут разделяемого ответчика для требуемой строки кеша и передает запрос чтения кеш-агенту CA4 с неспецифированным протоколом;

7) когда передача завершается, кеш CA41 определяет, что передача была расщеплена, и устанавливает свой атрибут разделяемого запросчика для требуемой строки кеша;

8) кеш-агент CA4 запрашивает магистраль M, становится задатчиком и затем совершает передачу разделяемого чтения. Память M расщепляет передачу, выставляя sr★. Кеш-агент CA4 устанавливает атрибут разделяемый запросчик. Память M устанавливает разделяемый ответчик;

9) память M получает данные, запрашивает магистраль и совершает передачу разделяемого ответа. Память M выставляет данные. Кеш-агент CA4 принимает данные, изменяет состояние строки кеша на исключительное немодифицированное и передает данные MA4. Кеш-агент сбрасывает атрибут разделяемого запросчика. Память M сбрасывает атрибут разделяемого ответчика;

10) агент памяти MA4 запрашивает магистраль MA4 и совершает передачу разделяемого ответа, выставляя tf★ для предотвращения состояния исключительного немодифицированного. Когда передача успешно завершается, он очищает атрибут разделяемого ответчика;

11) кеш-агент CA41 принимает данные, изменяет состояние строки кеша на разделяемый немодифицированный, очищает свой атрибут разделяемого запросчика и передает данные агенту памяти MA41,

12) агент памяти MA41 запрашивает магистраль MA41 и совершает передачу разделяемого ответа, выставляя tf★ для предотвращения состояния исключительного немодифицированного. Когда передача успешно завершается, он очищает атрибут разделяемого ответчика;

13) кеш-агент C411 принимает данные, изменяет состояние строки кеша на разделяемый немодифицированный, очищает свой атрибут разделяемого запросчика и передает данные процессору P411.

Таким образом удовлетворяется запрос на чтение, инициированный процессором P411. Процессор P411 может затем совершать любое число чтений, которые все будут чтениями с попаданием;

14) процессор P411 в конце концов записывает в эту строку кеша. Кеш C411 проверяет тег строки кеша и определяет, что строка кеша не является исключительной;

15) кеш C411 запрашивает магистраль MA41, становится задатчиком и затем совершает передачу недействительности;

16) агент памяти MA41 расщепляет передачу, устанавливает свой атрибут недействительного ответчика для требуемой строки кеша и передает запрос на недействительность кеш-агенту CA41;

17) когда передача завершается, кеш C411 определяет, что передача является расщепленной, и устанавливает свой атрибут исключительного запросчика для требуемой строки кеша;

18) кеш CA41 запрашивает магистраль MA4, становится задатчиком и затем совершает передачу недействительности;

19) агент памяти MA4 расщепляет передачу, устанавливает свой атрибут недействительного ответчика для требуемой строки кеша и передает запрос на недействительность кеш-агенту CA4;

20) когда передача завершается, кеш CA41 определяет, что передача является расщепленной, и устанавливает свой атрибут исключительного запросчика для требуемой строки кеша;

21) кеш-агент CA4 изменяет состояние своей строки с исключительного немодифицированного на исключительный модифицированный и передает модифицированный ответ к MA4. Необходимо заметить, что в случае целостного исполнения моста 4, MA4 мог раньше иметь доступ к тегам CA4 и мог бы не нуждаться в расщеплении передачи на шаге (19);

22) агент памяти MA4 запрашивает магистраль MA4 и совершает передачу модифицированного ответа. Когда передача завершается успешно, он очищает свой атрибут недействительного ответчика;

23) кеш-агент CA41 принимает модифицированный ответ, изменяет состояние строки кеша на исключительный модифицированный, очищает свой атрибут исключительного запросчика и передает ответ к MA41;

24) агент памяти MA41 запрашивает магистраль MA41 и совершает передачу модифицированного ответа. Когда передача завершается успешно, он очищает свой атрибут недействительного ответчика,

25) кеш-агент С411 принимает модифицированный ответ, изменяет состояние строки кеша на исключительный модифицированный, очищает свой атрибут исключительного запросчика и присоединяет данные из процессора Р411 к строке кеша и сохраняет ее в кеше.

Все последующие чтения и записи процессора Р411 являются попаданиями. Этот пример показал, как передачи производятся через множественные магистральные мосты и как атрибуты строки кеша накапливаются, используя «включающий» принцип.

8.1.5.6.2 **Четыре процессора одновременно читают и записывают в область**

Этот пример иллюстрирует одновременное чтение области четырьмя процессорами Р11, Р2, Р411 и Р412 с последующей записью в ту же область. В начале примера все кеши недействительны. Начальное значение области в памяти «0123». Четыре процессора записывают «А---», «-В---», «--С-» и «---D» соответственно, где только часть строки содержит записанную букву. Действие происходит следующим образом:

1) все четыре процессора пытаются прочитать из своих кешей;

2) четыре кеша проверяют кешевый тег области и определяют, что он недействительный;

3) четыре кеша запрашивают соответствующие магистрали. Кеш С411 выигрывает арбитраж и становится задатчиком на магистрали МА41. Кеш С411 совершает передачу разделяемого чтения, которую расщепляет агент памяти МА41. Кеш С411 устанавливает свой атрибут разделяемого запросчика. Агент памяти МА41 устанавливает свой атрибут разделяемого ответчика и передает запрос на чтение кеш-агенту СА41. В результате отслеживания передачи по магистрали кеш С412 устанавливает свой атрибут ожидающего запросчика;'

4) кеш-агент СА41 выигрывает арбитраж, становится задатчиком на магистрали МА4 и совершает передачу разделяемого чтения, которую расщепляет агент памяти МА4. Кеш-агент СА41 устанавливает свой атрибут разделяемого запросчика. Агент памяти МА4 устанавливает свой атрибут разделяемого ответчика и передает запрос на чтение кеш-агенту СА4;

5) тем временем кеш С11 выигрывает арбитраж и становится задатчиком на магистрали МА1. Кеш С11 совершает передачу разделяемого чтения, которую расщепляет агент памяти МА1. Кеш С11 устанавливает свой атрибут разделяемого запросчика. Агент памяти МА1 устанавливает свой атрибут разделяемого ответчика и передает запрос на чтение кеш-агенту СА1;

6) кеш-агенты СА1, СА4 и кеш С2 запрашивают магистраль М. Кеш-агент СА4 выигрывает и становится задатчиком на магистрали М, затем совершает передачу разделяемого чтения, которую память М расщепляет, выставляя sr★. Кеш-агент СА4 устанавливает свой атрибут разделяемого запросчика. Память М устанавливает свой атрибут разделяемого ответчика. Кеш-агент СА1 и кеш С2 устанавливают атрибут ожидающего запросчика;

7) память М получает данные, запрашивает магистраль, совершает передачу разделяемого ответа и выставляет данные «0123». Кеш-агенты СА1, СА4 и кеш С2 перехватывают данные и очищают свои атрибуты запросчика. Память М очищает свой атрибут ответчика. Кеш-агент СА4 изменяет состояние строки кеша на разделяемую немодифицированную и передает ответ чтения к МА4;

8) кеш-агент СА1 после перехвата данных изменяет состояние строки кеша на разделяемую немодифицированную и передает ответ чтения к МА1;

9) кеш С2 после перехвата данных изменяет состояние строки кеша на разделяемую немодифицированную, завершает свой запрос чтения и передает данные процессору Р2,

10) агент памяти МА1 запрашивает магистраль МА1, становится задатчиком и совершает передачу разделяемого ответа, передавая данные кешу С11. Агент памяти МА1 очищает свой атрибут разделяемого ответчика. Кеш С11 очищает свой атрибут разделяемого запросчика, изменяет состояние строки кеша на разделяемую немодифицированную и предоставляет данные процессору Р11;

11) агент памяти МА4 запрашивает магистраль МА4, становится задатчиком и совершает передачу разделяемого ответа, передавая данные кеш-агенту СА41. Агент памяти МА4 очищает свой атрибут разделяемого ответчика. Кеш-агент СА41 передает данные к МА41 и очищает свой атрибут разделяемого запросчика. С41 изменяет состояние строки кеша на разделяемую немодифицированную;

12) агент памяти МА41 запрашивает магистраль МА41, становится задатчиком и совершает передачу разделяемого ответа, передавая данные кешу С411. Кеш С412 перехватывает данные. Оба кеша изменяют у себя состояния строки кеша на разделяемую немодифицированную. Кеш С411 очищает свой атрибут разделяемого запросчика. Кеш С412 очищает свой атрибут ожидающего запросчика.

Таким образом удовлетворяются запросы на чтение, инициированные четырьмя процессорами. Процессоры могут теперь совершать любое число чтений, которые все будут попаданиями. Четыре кеша и кеш-агенты CA1, CA4 и CA41 имеют строку, помеченную как разделяемую немодифицированную;

13) все четыре процессора одновременно записывают в свои кеши;

14) четыре кеша проверяют кешевый тег для требуемой области и определяют, что они должны сделать недействительными все остальные копии в системе для того, чтобы изменить состояния своих строк на исключительные модифицированные;

15) четыре кеша запрашивают соответствующие магистрали. Кеш C2 выигрывает арбитраж и становится задатчиком на магистрали M. Кеш C2 совершает передачу недействительности, которую расщепляют кеш-агенты CA1 и CA4. Кеш C2 устанавливает свой атрибут исключительного запросчика, а кеш-агенты CA1 и CA4 устанавливают свои атрибуты недействительного ответчика и передают запрос на передачу недействительности своим агентам памяти;

16) агент памяти MA1 запрашивает магистраль MA1 и выигрывает соревнование у кеша C11. Когда агент памяти MA1 становится задатчиком, он совершает передачу недействительности, заставляя кеш C11 изменить состояние строки кеша на недействительное. Агент памяти MA1 передает завершение недействительности к CA1;

17) кеш C11 продолжает запрашивать магистраль MA1, становится задатчиком и совершает передачу модифицированного чтения, которую расщепляет агент памяти MA1. Кеш C11 устанавливает свой атрибут исключительного запросчика, а агент памяти MA1 устанавливает свой атрибут исключительного ответчика и передает запрос на модифицированное чтение кеш-агенту CA1;

18) кеш-агент C411 выигрывает арбитраж, становится задатчиком на магистрали MA41 и совершает передачу недействительности, которую расщепляет MA41. Кеш C411 устанавливает свой атрибут исключительного запросчика, агент памяти MA41 устанавливает свой атрибут исключительного ответчика и передает запрос на передачу недействительности кеш-агенту C41. Кеш C412 отслеживает передачу и устанавливает свой атрибут ожидающего запросчика, таким образом переходя в режим ожидания без потерь на доступ к магистрали.

В этой точке все четыре кеша ожидают ответы;

19) кеш-агент CA1 становится задатчиком на магистрали M и совершает передачу модифицированного ответа. Кеш-агент CA4 расщепляет передачу с выставлением сигналов sr★ и tf★, потому что он не завершил свою передачу недействительности. Кеш-агент CA1 очищает свой атрибут недействительного ответчика;

20) кеш-агент CA1 принимает модифицированное чтение от агента памяти MA1 и, так как он все еще остается задатчиком на магистрали M, совершает модифицированное чтение. Кеш-агент CA4 выставляет wt★, а кеш-агент CA1 устанавливает свой атрибут ожидающего запросчика;

21) кеш-агент CA41 становится задатчиком магистрали MA4 и совершает передачу недействительности, которую расщепляет MA4. Кеш-агент CA41 устанавливает свой атрибут исключительного запросчика, а агент памяти MA4 устанавливает свой атрибут недействительного ответчика.

В этой точке и кеш-агент CA4, и агент памяти MA4 имеют установленными свои атрибуты недействительного ответчика. Это означает, что произошло столкновение недействительностей в магистральном мосту CA4/MA4. Установление недействительности, пришедшее от кеша C2, обрабатывается первым, а установление недействительности от кеш-агента CA41 — вторым;

22) агент памяти MA4 очищает свой атрибут недействительного ответчика и устанавливает свой атрибут исключительного ответчика;

23) агент памяти MA4 становится задатчиком на магистрали MA4 и совершает передачу установления недействительности, которую расщепляет кеш-агент CA41. Агент памяти MA4 устанавливает свой атрибут исключительного запросчика. Кеш-агент CA41 устанавливает свой атрибут недействительного ответчика и продвигает запрос дальше, к агенту памяти MA41;

24) агент памяти MA41 становится задатчиком на магистрали MA41 и совершает передачу установления недействительности, которая заставляет кеш C411 изменить состояние своей строки кеша на недействительное. Агент памяти MA41 передает завершение установления недействительности кеш-агенту CA41;

25) кеш-агент CA41 становится задатчиком на магистрали MA4 и совершает передачу модифицированного ответа. Агент памяти MA4 передает модифицированный ответ кеш-агенту CA4 и очищает свой атрибут исключительного запросчика,

26) кеш-агент CA4 становится задатчиком на магистрали M, совершает передачу модифицированного ответа и очищает свой атрибут недействительного ответчика. Кешу C2 разрешается изменить состояния своей строки кеша на исключительное модифицированное и присоединить записанные данные от процессора P2. Величина, записанная в строку, теперь «OB23». Кеш-агент CA1 отслеживает модифицированный ответ и затем может очистить свой атрибут ожидающего запросчика.

В этой точке процессор P2 завершил свою запись. Строка имеет атрибут исключительная модифицированная в кеше C2 и недействительная в других кешах системы;

27) кеш-агент CA4 становится задатчиком на магистрали M и совершает передачу модифицированного чтения. Память M расщепляет ее, выставляя sr★. Кеш C2 вмешивается в передачу, выставляя sr★ и iv★. Кеш-агент CA4 выставляет sw★. Кеш C2 устанавливает атрибут исключительного ответчика. Кеш-агент CA4 устанавливает атрибут исключительного запросчика. Кеш-агент CA1 устанавливает свой атрибут ожидающего запросчика;

28) кеш C2 получает данные, запрашивает магистраль и совершает передачу модифицированного ответа. Кеш C2 предоставляет данные «OB23» кеш-агенту CA4 и памяти M. Кеш C2 изменяет состояние строки кеша на недействительное и сбрасывает свой атрибут исключительного ответчика. Кеш-агент CA4 изменяет состояние строки кеша на исключительное модифицированное, сбрасывает свой атрибут исключительного запросчика и передает данные «OB23» агенту памяти MA4. Кеш-агент CA1 сбрасывает свой атрибут ожидающего запросчика;

29) кеш-агент CA1 становится задатчиком на магистрали M и совершает передачу модифицированного чтения, которую кеш-агент CA4 расщепляет сигналами sr★ и iv★. Кеш-агент CA1 устанавливает свой атрибут исключительного запросчика, а кеш-агент CA4 устанавливает свой атрибут исключительного ответчика;

30) агент памяти MA4 становится задатчиком на магистрали MA4 и совершает передачу модифицированного ответа, предоставляя данные «OB23» кеш-агенту CA41. Агент памяти MA4 очищает свой атрибут исключительного ответчика и устанавливает свой атрибут модифицированный для строки памяти. Кеш-агент CA41 очищает свой атрибут исключительного запросчика и изменяет состояние строки кеша на исключительное модифицированное;

31) так как агент памяти MA4 является задатчиком на магистрали MA4 и его атрибут исключительного запросчика установлен, он немедленно совершает передачу модифицированного чтения. Кеш-агент CA41 расщепляет ее с выставлением сигналов sr★ и iv★. Агент памяти MA4 устанавливает свой атрибут исключительного запросчика, а кеш-агент CA41 устанавливает свой атрибут исключительного ответчика. Кеш-агент CA41 передает запрос модифицированного чтения агенту памяти MA41;

32) агент памяти MA41 становится задатчиком на магистрали MA41 и совершает передачу модифицированного ответа, предоставляя данные «OB23» кешу C411. Агент памяти MA41 очищает свой атрибут исключительного ответчика и устанавливает у себя модифицированный атрибут для строки памяти. Кеш C411 изменяет состояние своей строки кеша на исключительную модифицированную, очищает свой атрибут исключительного запросчика и присоединяет записанные данные от процессора P411. Величина, записанная в строку, теперь «OBC3».Кеш C412 очищает свой атрибут ожидающего запросчика;

33) кеш C412 становится задатчиком на магистрали MA41 и совершает передачу модифицированного чтения. Кеш C411 вмешивается в передачу с выставлением sr★ и iv★. Кеш C411 устанавливает свой атрибут исключительного ответчика. Агент памяти MA41 устанавливает свой атрибут ожидающего запросчика;

34) кеш C411 получает данные, запрашивает магистраль и совершает передачу модифицированного ответа. Кеш C411 предоставляет данные «OBC3» кешу C412. Кеш C412 присоединяет данные записи, изменяя их значение на «OBCD», и изменяет состояние строки на исключительное модифицированное. Кеш C411 изменяет состояние строки кеша на недействительное и сбрасывает свой атрибут исключительного ответчика. Кеш C412 сбрасывает свой атрибут исключительного запросчика. Агент памяти MA41 сбрасывает свой атрибут ожидающего запросчика;

35) агент памяти MA41 становится задатчиком на магистрали MA41 и совершает передачу модифицированного чтения. Кеш C412 вмешивается в передачу с выставлением сигналов sr★ и iv★. Кеш C411 устанавливает свой атрибут исключительного ответчика. Агент памяти MA41 устанавливает свой атрибут исключительного запросчика;

36) кеш С412 получает данные, запрашивает магистраль и совершает передачу модифицированного ответа. Кеш С412 предоставляет данные «OBCD», очищает свой атрибут исключительного ответчика и изменяет состояние своей строки кеша на недействительное. Агент памяти МА41 передает данные кеш-агенту СА41 и очищает свой модифицированный атрибут строки памяти и свой атрибут исключительного запросчика;

37) кеш-агент СА41 становится задатчиком на магистрали МА4 и совершает передачу модифицированного ответа, передавая данные агенту памяти МА4. Кеш-агент СА41 очищает свой атрибут исключительного ответчика и изменяет состояние своей строки кеша на недействительное. Агент памяти МА4 передает данные кеш-агенту СА4 и очищает свой модифицированный атрибут строки памяти и свой атрибут исключительного запросчика;

38) кеш-агент СА4 становится задатчиком на магистрали М и совершает передачу модифицированного ответа, передавая данные кеш-агенту СА1. Кеш-агент СА4 очищает свой атрибут исключительного ответчика и изменяет состояние своей строки на недействительное. Кеш-агент СА1 очищает свой атрибут исключительного запросчика, изменяет состояние своей строки на исключительное модифицированное и передает данные агенту памяти МА1;

39) агент памяти МА1 становится задатчиком на магистрали МА1 и совершает передачу модифицированного ответа, передавая данные «OBCD» кешу С11. Агент памяти МА1 очищает свой атрибут исключительного ответчика и устанавливает свой атрибут модифицированной строки памяти. Кеш С11 очищает свой атрибут исключительного запросчика, изменяет состояние своей строки кеша на исключительное модифицированное и присоединяет данные записи. Значение в строке кеша становится в итоге равным «ABCD».

В общем случае число передач, ожидаемых в случае среды, расщепляющей ответ, равняется (количество запросов) умноженное на (количество иерархических уровней) умноженное на два (запрос, затем ответ). Число передач может быть больше, если много передач должны обращаться к памяти и затем далее в исключительный модифицированный кеш; количество передач может быть меньше, если много передач локальны либо в пространстве, либо во времени, что позволяет сократить область распространения последовательностей передач или объединить множественные передачи. Производительность будет сильно зависеть от использования кластеров для доступов к данным в физической топологии системы.

В приведенном примере система обработала восемь запросов на обслуживание (четыре чтения и четыре записи) в 32 передачи. Этот пример с трехуровневой иерархией мог бы легко иметь 8★3★2 или 48 передач. Относительная локальность и объединение чтений были весьма успешны в смысле сокращения числа передач.

8.1.6 **Общий список команд и статусов кеш-когерентности**

	SR★	IV★	TF★					
Разделяемое чтение	0	0	0	Связ.	М,	пер.	ИН	
	0	0	1	Связ.	М,	пер.	РН	
	0	1	0	Связ.	К,	пер.	ИН,	М должна обн.
	0	1	1	Связ.	К,	пер.	РН,	М должна обн.
	1	0	X	Расщ.	М			
	1	1	X	Расщ.	К			
Модифицированное чтение	0	0	0	Связ.	М,	пер.	ИМ	
	0	0	1	Связ.	М,	пер.	РН,	затем недейст.
	0	1	0	Связ.	К,	пер.	ИМ	
	1	0	0	Расщ.	М			
	1	0	1	Расщ.	М,	расщ.	Н	
	1	1	0	Расщ.	К			
	X	1	1	недопустимо				
Передача недействительности	0	0	0	Пер.	ИМ			
	1	0	0	Расщ.	Н			
	X	X	1	недопустимо				
	X	1	X	недопустимо				

	SR★	IV★	TF★	
Обратное копирование	0	0	0	Все Н
	0	0	1	Некоторые РН
	X	1	X	недопустимо
	1	X	X	недопустимо
Недействительное чтение	0	0	0	Связ. М, все Н
	0	0	1	Связ. М, некоторые РН
	0	1	0	Связ. К, все Н, М должна обн.
	0	1	1	Связ. К, некоторые РН, М должна обн.
	1	0	X	Расщ. М
	1	1	X	Расщ. К
Недействительная запись	0	0	0	Все Н
	1	0	0	Расщ. Н
	X	1	X	недопустимо
	X	X	1	недопустимо
Разделяемый ответ	0	0	0	Пер. ИН, М должна обн.
	0	0	1	Пер. РН, М должна обн.
	1	X	X	недопустимо
	X	1	X	недопустимо
Модифицированный ответ	0	0	0	Пер. ИМ
	1	0	0	Расщ. Н, все КА Н
	1	0	1	Расщ. Н, некоторые КА не Н
	0	0	1	недопустимо
	X	1	X	недопустимо

Где сокращения обозначают следующее:

«X» — не имеет значения;

«Связ. М» — связанный ответ от памяти;

«Связ. К» — связанный ответ от вмешивающегося кеша;

«Расщ. М» — расщепленный ответ, совершенный памятью;

«Расщ. К» — расщепленный ответ, совершенный вмешивающимся кешем;

«Расщ. Н» — расщепленный ответ, совершенный по меньшей мере одной задержанной передачей недействительности;

«Пер. РН» — запросчик переводит свою строку кеша в состояние разделяемой немодифицированной, некоторые другие кеши также могут иметь разделяемые немодифицированные строки кеша;

«Пер. ИН» — запросчик переводит свою строку кеша в состояние исключительной немодифицированной, все другие кеши гарантированно имеют недействительные строки кеша;

«Пер. ИМ» — запросчик переводит свою строку кеша в состояние исключительной модифицированной, все другие кеши гарантированно имеют недействительные строки;

«М должна обн.» — память должна обновить свои данные (если память сама не является задатчиком в расщепленном ответе);

«затем недейст.» — после передачи модифицированного чтения задатчик должен совершить передачу недействительности;

«Все Н» — каждый кеш гарантированно имеет строку кеша в состоянии недействительности;

«Некоторые РН» — некоторые кеши могут иметь строку кеша в разделяемом немодифицированном состоянии;

«Все КА Н» — каждый кеш и кеш-агент гарантированно имеет строку кеша в недействительном состоянии, однако это не относится к агенту памяти;

«Некоторые КА не Н» — некоторые кеши или кеш-агенты могут не иметь недействительные строки кеша.

8.1.7 **Недопустимые комбинации атрибутов**

Если для строки кеша установлен исключительный атрибут, тогда ни один другой кеш не может иметь для этой строки действительный атрибут. Из этого следует несколько легко определяемых недопустимых последовательностей команд. Любое из этих условий указывает, что требования кеш-когерентности подвергнуты риску, в результате чего должен выставляться сигнал TE★.

8.1.7.1 **Разделяемый немодифицированный**

Если строка кеша разделяемая немодифицированная и кеш наблюдает передачу обратного копирования или передачу чтения с выставленным сигналом IV★, это значит, что другой кеш неправомерно имеет исключительный модифицированный атрибут для этой строки.

8.1.7.2 **Исключительный немодифицированный**

Если строка кеша исключительная немодифицированная и кеш наблюдает передачу обратного копирования, передачу чтения с выставленным сигналом IV★, передачу модифицированного чтения с выставленным TF★, но без tf★, или передачу недействительности, это значит, что другой кеш неправомерно имеет действительный атрибут для этой строки.

8.1.7.3 **Исключительный модифицированный**

Если строка кеша исключительная модифицированная и кеш наблюдает передачу обратного копирования, передачу модифицированного чтения с выставленным сигналом TF★ или передачу недействительности, это значит, что другой кеш неправомерно имеет действительный атрибут для этой строки.

8.1.7.4 **Модифицированная строка памяти**

Если агент памяти имеет для строки памяти атрибут модифицированной строки памяти и наблюдает передачу чтения без выставленного IV★, это значит, что какой-то кеш имеет неправомерно сброшенный атрибут исключительный модифицированный. Если агент памяти имеет для строки памяти атрибут модифицированной строки памяти и наблюдает передачу недействительности, тогда какой-то кеш неправомерно имеет разделяемый немодифицированный атрибут для этой строки.

Если агент памяти не имеет для строки памяти атрибут модифицированной строки памяти и наблюдает передачу чтения с выставленным IV★ или передачу обратного копирования, тогда какой-то кеш неправомерно имеет исключительный модифицированный атрибут для этой строки.

8.2 **Спецификация**

Уравнения в этом разделе предполагают, что передачи завершены успешно. Если выставлен сигнал BS★, это значит, что команда не должна произвести никакого действия. Если выставлен сигнал TE★, это значит, что команда не должна произвести никакого действия или, что на более верхний уровень должно быть послано сообщение об ошибке.

8.2.1 **Атрибуты модулей**

КЕШ

Модуль КЕШ должен обеспечивать хранение как минимум одной строки кеша. КЕШ должен обеспечивать для каждой своей строки атрибуты, определенные в 8.2.3.

АГЕНТ_КЕША

АГЕНТ_КЕША — это КЕШ-модуль, для которого истинно условие РАЗРЕШЕНИЕ_РАСЩЕПЛЕНИЯ & ПРИНИМАЮЩИЙ_РАСЩЕПЛЕНИЕ.

ПАМЯТЬ

Модуль ПАМЯТЬ должен быть конечным источником и/или назначением для данных, которые могут быть сохранены в нем.

АГЕНТ_ПАМЯТИ

АГЕНТ_ПАМЯТИ — это модуль, для которого истинно условие: РАЗРЕШЕНИЕ_РАСЩЕПЛЕНИЯ & ПРИНИМАЮЩИЙ_РАСЩЕПЛЕНИЕ и который действует на локальной магистрали вместо удаленной ПАМЯТИ.

8.2.2 **Атрибуты статуса**

КЕШ_OP_CMPLT

Модули должны устанавливать КЕШ_OP_CMPLT при условии: ЗАВЕРШЕНИЕ_ПЕРЕДАЧИ & -ЗАНЯТО & -ОШИБКА_ПЕРЕДАЧИ | РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ & ОЧЕРЕДЬ_АТРИБУТОВ.

ОЧЕРЕДЬ_АТРИБУТОВ

Модули должен устанавливать ОЧЕРЕДЬ_АТРИБУТОВ при условии, что все атрибуты, необходимые для совершения ожидающей своей очереди операции со строкой кеша, являются сохраненными таким образом, что они могут быть позднее применены к соответствующему пакету.

КЕШИРОВАННЫЙ

Модуль должен устанавливать КЕШИРОВАННЫЙ при условии: АДРЕС_ДЕКОДИРОВАН и адрес находится в диапазоне, установленном для пространства кеш-когерентности, и модуль способен отвечать на кеш-когерентные передачи и имеет тег, соответствующий строке кеша, ассоциирующейся с данным адресом. Модуль должен поддерживать КЕШИРОВАННЫЙ установленным, пока: -КЕШ_OP_CMPLT.

ПЕРЕХВАТ_ДАННЫХ

Модуль может устанавливать ПЕРЕХВАТ_ДАННЫХ при условии: КЕШИРОВАННЫЙ & (РАЗДЕЛЯЕМОЕ_ЧТЕНИЕ | ОБРАТНОЕ_КОПИРОВАНИЕ | НЕДЕЙСТВИТЕЛЬНОЕ_ЧТЕНИЕ | РАЗДЕЛЯЕМЫЙ_ОТВЕТ) и должен поддерживать ПЕРЕХВАТ_ДАННЫХ установленным, пока: -КЕШ_OP_CMPLT.

ПРЕВЕНТИВНЫЙ_ИН

Модуль может устанавливать ПРЕВЕНТИВНЫЙ_ИН при условии: (ЗАДАТЧИК | КЕШИРОВАННЫЙ) & (РАЗДЕЛЯЕМОЕ_ЧТЕНИЕ | РАЗДЕЛЯЕМЫЙ_ОТВЕТ) и должен поддерживать ПРЕВЕНТИВНЫЙ_ИН установленным, пока: -КЕШ_OP_CMPLT

СОХРАНЕНИЕ_КОПИИ

Модуль должен устанавливать СОХРАНЕНИЕ_КОПИИ при условии: (КЕШИРОВАННЫЙ & -СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ & (НЕДЕЙСТВИТЕЛЬНОЕ_ЧТЕНИЕ | РАЗДЕЛЯЕМОЕ_ЧТЕНИЕ | ЗАДАТЧИК & (ОБРАТНОЕ_КОПИРОВАНИЕ | РАЗДЕЛЯЕМЫЙ_ОТВЕТ)) & модуль удержит -СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ в конце передачи) и должен поддерживать СОХРАНЕНИЕ_КОПИИ установленным, пока: -КЕШ_OP_CMPLT.

ВЫСТАВЛЕНИЕ_CS

Модуль должен устанавливать ВЫСТАВЛЕНИЕ_CS при условии: (ЗАДАТЧИК | КЕШИРОВАННЫЙ) & (ПЕРЕХВАТ_ДАННЫХ | СОХРАНЕНИЕ_КОПИИ | ПРЕВЕНТИВНЫЙ_ИН) | АГЕНТ_КЕША & (МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ | МОДИФИЦИРОВАННЫЙ_ОТВЕТ) & -СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ & АГЕНТ_КЕША не может гарантировать СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ в конце передачи), и должен поддерживать ВЫСТАВЛЕНИЕ_CS установленным, пока: -КЕШ_OP_CMPLT

ВМЕШАТЕЛЬСТВО

Модуль должен устанавливать ВМЕШАТЕЛЬСТВО при условии: КЕШИРОВАННЫЙ & СТАТУС_ИСКЛЮЧИТЕЛЬНОГО_МОДИФИЦИРОВАННОГО & (РАЗДЕЛЯЕМОЕ_ЧТЕНИЕ | МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ | НЕДЕЙСТВИТЕЛЬНОЕ_ЧТЕНИЕ), и должен поддерживать ВМЕШАТЕЛЬСТВО установленным, пока: -КЕШ_OP_CMPLT.

ОЖИДАНИЕ_КЕШИРОВАННОЕ

Модуль должен устанавливать ОЖИДАНИЕ_КЕШИРОВАННОЕ на то время, пока: КЕШИРОВАННЫЙ & ОТВЕТЧИК & (РАЗДЕЛЯЕМОЕ_ЧТЕНИЕ | МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ | НЕДЕЙСТВИТЕЛЬНОСТЬ | НЕДЕЙСТВИТЕЛЬНОЕ_ЧТЕНИЕ).

КЕШИРОВАННОЕ_РАСЩЕПЛЕНИЕ

Модуль может устанавливать КЕШИРОВАННОЕ_РАСЩЕПЛЕНИЕ на то время, пока: КЕШИРОВАННЫЙ & (АГЕНТ_КЕША & (СТАТУС_ИСКЛЮЧИТЕЛЬНОГО_МОДИФИЦИРОВАННОГО & (РАЗДЕЛЯЕМОЕ_ЧТЕНИЕ | МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ | НЕДЕЙСТВИТЕЛЬНОЕ_ЧТЕНИЕ) | -СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ & (НЕДЕЙСТВИТЕЛЬНОСТЬ | НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ)) | АГЕНТ_ПАМЯТИ & (РАЗДЕЛЯЕМОЕ_ЧТЕНИЕ | МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ | НЕДЕЙСТВИТЕЛЬНОСТЬ | НЕДЕЙСТВИТЕЛЬНОЕ_ЧТЕНИЕ | НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ) | МОДИФИЦИРОВАННЫЙ_ОТВЕТ & СТАТУС_ОТВЕТЧИКА).

ФЛАГ_ОЧЕРЕДИ_ПЕРЕДАЧИ

Модули должны устанавливать ФЛАГ_ОЧЕРЕДИ_ПЕРЕДАЧИ при условии: УЧАСТИЕ & -РЕЖИМ_МНОЖЕСТВЕННЫХ_ПАКЕТОВ & ПРИЗНАК_РАССОЕДИНЕНИЯ_ОБНАРУЖЕН & ST4★ | ФЛАГ_ПОСЛЕДНЕЙ_ПЕРЕДАЧИ и должны поддерживать ФЛАГ_ОЧЕРЕДИ_ПЕРЕДАЧИ установленным, пока: -КЕШ_OP_CMPLT.

СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ

Модуль должен устанавливать СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ при условии: АДРЕС_ДЕКОДИРОВАН & КЕШИРОВАННЫЙ и статус адресуемой строки у данного кеша НЕДЕЙСТВИТЕЛЬНЫЙ. Модули должны поддерживать СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ установленным, пока: -КЕШ_OP_CMPLT.

СТАТУС_РАЗДЕЛЯЕМОГО_НЕМОДИФИЦИРОВАННОГО

Модуль должен устанавливать СТАТУС_РАЗДЕЛЯЕМОГО_НЕМОДИФИЦИРОВАННОГО при условии: АДРЕС_ДЕКОДИРОВАН & КЕШИРОВАННЫЙ и статус адресуемой строки кеша РАЗДЕЛЯЕМЫЙ_НЕМОДИФИЦИРОВАННЫЙ. Модули должны поддерживать СТАТУС_РАЗДЕЛЯЕМОГО_НЕМОДИФИЦИРОВАННОГО установленным, пока: -КЕШ_ОР_СМРLТ.

СТАТУС_ИСКЛЮЧИТЕЛЬНОГО_НЕМОДИФИЦИРОВАННОГО

Модуль должен устанавливать СТАТУС_ИСКЛЮЧИТЕЛЬНОГО_НЕМОДИФИЦИРОВАННОГО при условии: АДРЕС_ДЕКОДИРОВАН & КЕШИРОВАННЫЙ и статус адресуемой строки кеша ИСКЛЮЧИТЕЛЬНЫЙ_НЕМОДИФИЦИРОВАННЫЙ. Модули должны поддерживать СТАТУС_ИСКЛЮЧИТЕЛЬНОГО_НЕМОДИФИЦИРОВАННОГО установленным, пока: -КЕШ_ОР_СМРLТ.

СТАТУС_ИСКЛЮЧИТЕЛЬНОГО_МОДИФИЦИРОВАННОГО

Модуль должен устанавливать СТАТУС_ИСКЛЮЧИТЕЛЬНОГО_МОДИФИЦИРОВАННОГО при условии: АДРЕС_ДЕКОДИРОВАН & КЕШИРОВАННЫЙ и статус адресуемой строки кеша ИСКЛЮЧИТЕЛЬНЫЙ_МОДИФИЦИРОВАННЫЙ. Модули должны поддерживать СТАТУС_ИСКЛЮЧИТЕЛЬНОГО_МОДИФИЦИРОВАННОГО установленным, пока: -КЕШ_ОР_СМРLТ.

СТАТУС_ЗАПРОСЧИКА

Модуль должен устанавливать СТАТУС_ЗАПРОСЧИКА при условии: АДРЕС_ДЕКОДИРОВАН & КЕШИРОВАННЫЙ и адресуемая строка КЕША имеет атрибут ЗАПРОСЧИК. Модули должны поддерживать СТАТУС_ЗАПРОСЧИКА установленным, пока: -КЕШ_ОР_СМРLТ.

СТАТУС_ИСКЛЮЧИТЕЛЬНОГО_ЗАПРОСЧИКА

Модуль должен устанавливать СТАТУС_ИСКЛЮЧИТЕЛЬНОГО_ЗАПРОСЧИКА при условии: АДРЕС_ДЕКОДИРОВАН & КЕШИРОВАННЫЙ и адресуемая строка кеша имеет атрибут ИСКЛЮЧИТЕЛЬНЫЙ_ЗАПРОСЧИК. Модули должны поддерживать СТАТУС_ИСКЛЮЧИТЕЛЬНОГО_ЗАПРОСЧИКА установленным, пока: -КЕШ_ОР_СМРLТ.

СТАТУС_РАЗДЕЛЯЕМОГО_ЗАПРОСЧИКА

Модуль должен устанавливать СТАТУС_РАЗДЕЛЯЕМОГО_ЗАПРОСЧИКА при условии: АДРЕС_ДЕКОДИРОВАН & КЕШИРОВАННЫЙ и адресуемая строка кеша имеет атрибут РАЗДЕЛЯЕМЫЙ_ЗАПРОСЧИК. Модули должны поддерживать СТАТУС_РАЗДЕЛЯЕМОГО_ЗАПРОСЧИКА установленным, пока: -КЕШ_ОР_СМРLТ.

СТАТУС_ЗАПРОСЧИКА_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ

Модуль должен устанавливать СТАТУС_ЗАПРОСЧИКА_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ при условии: АДРЕС_ДЕКОДИРОВАН & КЕШИРОВАННЫЙ и адресуемая строка кеша имеет атрибут ЗАПРОСЧИК_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ. Модули должны поддерживать СТАТУС_ЗАПРОСЧИКА_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ установленным, пока: -КЕШ_ОР_СМРLТ.

СТАТУС_ОТВЕТЧИКА

Модуль должен устанавливать СТАТУС_ОТВЕТЧИКА при условии: АДРЕС_ДЕКОДИРОВАН & КЕШИРОВАННЫЙ и адресуемая строка кеша имеет атрибут ОТВЕТЧИК. Модули должны поддерживать СТАТУС_ОТВЕТЧИКА установленным, пока: -КЕШ_ОР_СМРLТ.

СТАТУС_ИСКЛЮЧИТЕЛЬНОГО_ОТВЕТЧИКА

Модуль должен устанавливать СТАТУС_ИСКЛЮЧИТЕЛЬНОГО_ОТВЕТЧИКА при условии: АДРЕС_ДЕКОДИРОВАН & КЕШИРОВАННЫЙ и адресуемая строка кеша имеет атрибут ИСКЛЮЧИТЕЛЬНЫЙ_ОТВЕТЧИК. Модули должны поддерживать СТАТУС_ИСКЛЮЧИТЕЛЬНОГО_ОТВЕТЧИКА установленным, пока: -КЕШ_ОР_СМРLТ.

СТАТУС_НЕДЕЙСТВИТЕЛЬНОГО_ОТВЕТЧИКА

Модуль должен устанавливать СТАТУС_НЕДЕЙСТВИТЕЛЬНОГО_ОТВЕТЧИКА при условии: АДРЕС_ДЕКОДИРОВАН & КЕШИРОВАННЫЙ и адресуемая строка кеша имеет атрибут НЕДЕЙСТВИТЕЛЬНЫЙ_ОТВЕТЧИК. Модули должны поддерживать СТАТУС_НЕДЕЙСТВИТЕЛЬНОГО_ОТВЕТЧИКА установленным, пока: -КЕШ_ОР_СМРLТ.

СТАТУС_РАЗДЕЛЯЕМОГО_ОТВЕТЧИКА

Модуль должен устанавливать СТАТУС_РАЗДЕЛЯЕМОГО_ОТВЕТЧИКА при условии: АДРЕС_ДЕКОДИРОВАН & КЕШИРОВАННЫЙ и адресуемая строка кеша имеет атрибут РАЗДЕЛЯЕМЫЙ_ОТВЕТЧИК. Модули должны поддерживать СТАТУС_РАЗДЕЛЯЕМОГО_ОТВЕТЧИКА установленным, пока: -КЕШ_ОР_СМРLТ.

СТАТУС_ОТВЕТЧИКА_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ

Модуль должен устанавливать СТАТУС_ОТВЕТЧИКА_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ при условии: АДРЕС_ДЕКОДИРОВАН & КЕШИРОВАННЫЙ и адресуемая строка кеша имеет атрибут ОТВЕТЧИК_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ. Модули должны поддерживать СТАТУС_ОТВЕТЧИКА_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ установленным, пока: -КЕШ_OP_CMPLT.

СТАТУС_ОЖИДАЮЩЕГО_ЗАПРОСЧИКА

Модуль должен устанавливать СТАТУС_ОЖИДАЮЩЕГО_ЗАПРОСЧИКА при условии: АДРЕС_ДЕКОДИРОВАН & КЕШИРОВАННЫЙ и адресуемая строка кеша имеет атрибут ОЖИДАЮЩИЙ_ЗАПРОСЧИК. Модули должны поддерживать СТАТУС_ОЖИДАЮЩЕГО_ЗАПРОСЧИКА установленным, пока: -КЕШ_OP_CMPLT.

8.2.3 **Атрибуты кеш-модуля для каждой строки**

НЕДЕЙСТВИТЕЛЬНЫЙ

КЕШ или АГЕНТ_КЕША должен устанавливать НЕДЕЙСТВИТЕЛЬНЫЙ и очищать РАЗДЕЛЯЕМЫЙ_НЕМОДИФИЦИРОВАННЫЙ | ИСКЛЮЧИТЕЛЬНЫЙ_НЕМОДИФИЦИРОВАННЫЙ | ИСКЛЮЧИТЕЛЬНЫЙ_МОДИФИЦИРОВАННЫЙ для всех строк при условии: СИСТЕМНЫЙ_СБРОС.

КЕШ или АГЕНТ_КЕША должен устанавливать НЕДЕЙСТВИТЕЛЬНЫЙ и очищать РАЗДЕЛЯЕМЫЙ_НЕМОДИФИЦИРОВАННЫЙ | ИСКЛЮЧИТЕЛЬНЫЙ_НЕМОДИФИЦИРОВАННЫЙ | ИСКЛЮЧИТЕЛЬНЫЙ_МОДИФИЦИРОВАННЫЙ при условии: ЗАДАТЧИК & (МОДИФИЦИРОВАННЫЙ_ОТВЕТ | НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ | НЕДЕЙСТВИТЕЛЬНОЕ_ЧТЕНИЕ | (ОБРАТНОЕ_КОПИРОВАНИЕ | РАЗДЕЛЯЕМЫЙ_ОТВЕТ) & – СОХРАНЕНИЕ_КОПИИ) | КЕШИРОВАННЫЙ & -СОХРАНЕНИЕ_КОПИИ & (РАЗДЕЛЯЕМОЕ_ЧТЕНИЕ | НЕДЕЙСТВИТЕЛЬНОЕ_ЧТЕНИЕ) | КЕШИРОВАННЫЙ & -РАСЩЕПЛЕНИЕ & (МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ | НЕДЕЙСТВИТЕЛЬНОСТЬ | НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ).

КЕШ или АГЕНТ_КЕША может устанавливать НЕДЕЙСТВИТЕЛЬНЫЙ и очищать РАЗДЕЛЯЕМЫЙ_НЕМОДИФИЦИРОВАННЫЙ, ИСКЛЮЧИТЕЛЬНЫЙ_НЕМОДИФИЦИРОВАННЫЙ при условии: -ИСКЛЮЧИТЕЛЬНЫЙ_МОДИФИЦИРОВАННЫЙ.

КЕШ или АГЕНТ_КЕША не должны разрешать доступ на чтение или модификацию данных в строке кеша с установленным атрибутом НЕДЕЙСТВИТЕЛЬНЫЙ.

РАЗДЕЛЯЕМЫЙ_НЕМОДИФИЦИРОВАННЫЙ

КЕШ или АГЕНТ_КЕША должен устанавливать РАЗДЕЛЯЕМЫЙ_НЕМОДИФИЦИРОВАННЫЙ и очищать НЕДЕЙСТВИТЕЛЬНЫЙ | ИСКЛЮЧИТЕЛЬНЫЙ_НЕМОДИФИЦИРОВАННЫЙ | ИСКЛЮЧИТЕЛЬНЫЙ_МОДИФИЦИРОВАННЫЙ при условии: ЗАДАТЧИК & (СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ & -ТОЛЬКО_АДРЕС & (РАЗДЕЛЯЕМОЕ_ЧТЕНИЕ | МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ) | СОХРАНЕНИЕ_КОПИИ & (ОБРАТНОЕ_КОПИРОВАНИЕ | РАЗДЕЛЯЕМЫЙ_ОТВЕТ)) | КЕШИРОВАННЫЙ & (РАЗДЕЛЯЕМЫЙ_ЗАПРОСЧИК & РАЗДЕЛЯЕМЫЙ_ОТВЕТ & СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ & -ТОЛЬКО_АДРЕС & СТАТУС_ФЛАГА_ПЕРЕДАЧИ | ПЕРЕХВАТ_ДАННЫХ & –ТОЛЬКО_АДРЕС | ИСКЛЮЧИТЕЛЬНЫЙ_ЗАПРОСЧИК & МОДИФИЦИРОВАННЫЙ_ОТВЕТ & -ТОЛЬКО_АДРЕС & СТАТУС_РАСЩЕПЛЕНИЯ | -СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ & СОХРАНЕНИЕ_КОПИИ & (РАЗДЕЛЯЕМОЕ_ЧТЕНИЕ | НЕДЕЙСТВИТЕЛЬНОЕ_ЧТЕНИЕ))

КЕШ или АГЕНТ_КЕША может устанавливать РАЗДЕЛЯЕМЫЙ_НЕМОДИФИЦИРОВАННЫЙ и очищать ИСКЛЮЧИТЕЛЬНЫЙ_НЕМОДИФИЦИРОВАННЫЙ при условии: ИСКЛЮЧИТЕЛЬНЫЙ_НЕМОДИФИЦИРОВАННЫЙ

КЕШ или АГЕНТ_КЕША не должен разрешать доступ на модификацию данных в строке кеша с установленным атрибутом РАЗДЕЛЯЕМЫЙ_НЕМОДИФИЦИРОВАННЫЙ. Кеш или АГЕНТ_КЕША может разрешать доступ на чтение данных из строки кеша с установленным атрибутом РАЗДЕЛЯЕМЫЙ_НЕМОДИФИЦИРОВАННЫЙ.

ИСКЛЮЧИТЕЛЬНЫЙ_НЕМОДИФИЦИРОВАННЫЙ

КЕШ или АГЕНТ_КЕША должен устанавливать ИСКЛЮЧИТЕЛЬНЫЙ_НЕМОДИФИЦИРОВАННЫЙ и очищать НЕДЕЙСТВИТЕЛЬНЫЙ | РАЗДЕЛЯЕМЫЙ_НЕМОДИФИЦИРОВАННЫЙ | ИСКЛЮЧИТЕЛЬНЫЙ_МОДИФИЦИРОВАННЫЙ при условии: СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ & -ТОЛЬКО_АДРЕС & -ФЛАГ_ОЧЕРЕДИ_ПЕРЕДАЧИ & (ЗАДАТЧИК & РАЗДЕЛЯЕМОЕ_ЧТЕНИЕ | КЕШИРОВАННЫЙ & РАЗДЕЛЯЕМЫЙ_ЗАПРОСЧИК & РАЗДЕЛЯЕМЫЙ_ОТВЕТ).

КЕШ или АГЕНТ_КЕША не должен разрешать доступ на модификацию данных в строке кеша с установленным атрибутом ИСКЛЮЧИТЕЛЬНЫЙ_НЕМОДИФИЦИРОВАННЫЙ. Кеш или АГЕНТ_КЕША может разрешать доступ на чтение данных из строки кеша с установленным атрибутом ИСКЛЮЧИТЕЛЬНЫЙ_НЕМОДИФИЦИРОВАННЫЙ.

ИСКЛЮЧИТЕЛЬНЫЙ_МОДИФИЦИРОВАННЫЙ

КЕШ или АГЕНТ_КЕША должен устанавливать ИСКЛЮЧИТЕЛЬНЫЙ_МОДИФИЦИРОВАННЫЙ и очищать НЕДЕЙСТВИТЕЛЬНЫЙ | РАЗДЕЛЯЕМЫЙ_НЕМОДИФИЦИРОВАННЫЙ | ИСКЛЮЧИТЕЛЬНЫЙ_НЕМОДИФИЦИРОВАННЫЙ при условии: ЗАДАТЧИК & -СТАТУС_РАСЩЕПЛЕНИЯ & -СТАТУС_ОЖИДАНИЯ & (СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ & МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ & -ФЛАГ_ОЧЕРЕДИ_ПЕРЕДАЧИ | СТАТУС_РАЗДЕЛЯЕМОГО_НЕМОДИФИЦИРОВАННОГО & НЕДЕЙСТВИТЕЛЬНОСТЬ) | КЕШИРОВАННЫЙ & ИСКЛЮЧИТЕЛЬНЫЙ_ЗАПРОСЧИК & МОДИФИЦИРОВАННЫЙ_ОТВЕТ & -СТАТУС_РАСЩЕПЛЕНИЯ & (СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ & -ТОЛЬКО_АДРЕС | СТАТУС_РАЗДЕЛЯЕМОГО_НЕМОДИФИЦИРОВАННОГО).

КЕШ или АГЕНТ_КЕША может устанавливать ИСКЛЮЧИТЕЛЬНЫЙ_МОДИФИЦИРОВАННЫЙ и очищать ИСКЛЮЧИТЕЛЬНЫЙ_НЕМОДИФИЦИРОВАННЫЙ при условии: ИСКЛЮЧИТЕЛЬНЫЙ_НЕМОДИФИЦИРОВАННЫЙ.

КЕШ или АГЕНТ_КЕША может разрешать доступ на чтение или модификацию данных в строке кеша с установленным атрибутом ИСКЛЮЧИТЕЛЬНЫЙ_МОДИФИЦИРОВАННЫЙ

8.2.4 **Атрибуты запросчика для каждой строки кеша**

ЗАПРОСЧИК

Модуль должен устанавливать для строки кеша атрибут ЗАПРОСЧИК при условии: ИСКЛЮЧИТЕЛЬНЫЙ_ЗАПРОСЧИК | РАЗДЕЛЯЕМЫЙ_ЗАПРОСЧИК | ЗАПРОСЧИК_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ.

ИСКЛЮЧИТЕЛЬНЫЙ_ЗАПРОСЧИК

Модуль должен устанавливать ИСКЛЮЧИТЕЛЬНЫЙ_ЗАПРОСЧИК при условии: ЗАДАТЧИК & СТАТУС_РАСЩЕПЛЕНИЯ & (МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ | НЕДЕЙСТВИТЕЛЬНОСТЬ) и очистить ИСКЛЮЧИТЕЛЬНЫЙ_ЗАПРОСЧИК при условии: КЕШИРОВАННЫЙ & (-СТАТУС_РАСЩЕПЛЕНИЯ & МОДИФИЦИРОВАННЫЙ_ОТВЕТ | НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ).

Модуль должен очистить ИСКЛЮЧИТЕЛЬНЫЙ_ЗАПРОСЧИК для всех строк при условии: СИСТЕМНЫЙ_СБРОС.

РАЗДЕЛЯЕМЫЙ_ЗАПРОСЧИК

Модуль должен устанавливать РАЗДЕЛЯЕМЫЙ_ЗАПРОСЧИК при условии: ЗАДАТЧИК & СТАТУС_РАСЩЕПЛЕНИЯ & (РАЗДЕЛЯЕМОЕ_ЧТЕНИЕ | НЕДЕЙСТВИТЕЛЬНОЕ ЧТЕНИЕ) и очистить РАЗДЕЛЯЕМЫЙ_ЗАПРОСЧИК при условии: КЕШИРОВАННЫЙ & (РАЗДЕЛЯЕМЫЙ_ОТВЕТ | НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ | ОБРАТНОЕ_КОПИРОВАНИЕ).

Модуль должен очистить РАЗДЕЛЯЕМЫЙ_ЗАПРОСЧИК для всех строк при условии: СИСТЕМНЫЙ_СБРОС.

ЗАПРОСЧИК_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ

Модуль должен устанавливать ЗАПРОСЧИК_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ при условии: ЗАДАТЧИК & СТАТУС_РАСЩЕПЛЕНИЯ & НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ и очистить ЗАПРОСЧИК_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ при условии: КЕШИРОВАННЫЙ & (-СТАТУС_РАСЩЕПЛЕНИЯ & МОДИФИЦИРОВАННЫЙ_ОТВЕТ | НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ).

Модуль должен очистить ЗАПРОСЧИК_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ для всех строк при условии: СИСТЕМНЫЙ_СБРОС.

ОЖИДАЮЩИЙ_ЗАПРОСЧИК

Модуль должен устанавливать ОЖИДАЮЩИЙ_ЗАПРОСЧИК при условии: -ЗАДАТЧИК & СТАТУС_ОЖИДАНИЯ & (РАЗДЕЛЯЕМОЕ_ЧТЕНИЕ | МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ | НЕДЕЙСТВИТЕЛЬНОСТЬ | НЕДЕЙСТВИТЕЛЬНОЕ_ЧТЕНИЕ).

Модуль может устанавливать ОЖИДАЮЩИЙ_ЗАПРОСЧИК при условии: -ЗАДАТЧИК & (СТАТУС_РАСЩЕПЛЕНИЯ | СТАТУС_ОЖИДАНИЯ) & (РАЗДЕЛЯЕМОЕ_ЧТЕНИЕ | МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ | НЕДЕЙСТВИТЕЛЬНОСТЬ | НЕДЕЙСТВИТЕЛЬНОЕ_ЧТЕНИЕ | МОДИФИЦИРОВАННЫЙ_ОТВЕТ).

Модуль должен очистить ОЖИДАЮЩИЙ_ЗАПРОСЧИК при условии: КЕШИРОВАННЫЙ & (РАЗДЕЛЯЕМЫЙ_ОТВЕТ | МОДИФИЦИРОВАННЫЙ_ОТВЕТ & -СТАТУС_РАСЩЕПЛЕНИЯ | НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ).

Модуль должен очистить ОЖИДАЮЩИЙ_ЗАПРОСЧИК для всех строк кеша при условии: СИСТЕМНЫЙ_СБРОС. Модули могут очистить ОЖИДАЮЩИЙ_ЗАПРОСЧИК при условии: происходит определенный пользователем тайм-аут.

8.2.5 **Атрибуты ответчика для каждой строки кеша**

ОТВЕТЧИК

Модуль должен устанавливать ОТВЕТЧИК при условии: ИСКЛЮЧИТЕЛЬНЫЙ_ОТВЕТЧИК ¦ НЕДЕЙСТВИТЕЛЬНЫЙ_ОТВЕТЧИК ¦ РАЗДЕЛЯЕМЫЙ_ОТВЕТЧИК ¦ ОТВЕТЧИК_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ.

ИСКЛЮЧИТЕЛЬНЫЙ_ОТВЕТЧИК

Модуль должен устанавливать ИСКЛЮЧИТЕЛЬНЫЙ_ОТВЕТЧИК при условии: КЕШИРОВАННЫЙ & (РАСЩЕПЛЕНИЕ & СТАТУС_ИСКЛЮЧИТЕЛЬНОГО_МОДИФИЦИРОВАННОГО & МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ ¦ РАСЩЕПЛЕНИЕ & АГЕНТ_ПАМЯТИ & -ST5★ МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ) ¦ АГЕНТ_ПАМЯТИ & СТАТУС_НЕДЕЙСТВИТЕЛЬНОГО_ОТВЕТЧИКА & СТОЛКНОВЕНИЕ_НЕДЕЙСТВИТЕЛЬНОСТИ.

Модуль должен очищать ИСКЛЮЧИТЕЛЬНЫЙ_ОТВЕТЧИК при условии: ЗАДАТЧИК & МОДИФИЦИРОВАННЫЙ_ОТВЕТ ¦ КЕШИРОВАННЫЙ & НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ.

Модуль должен очистить ИСКЛЮЧИТЕЛЬНЫЙ_ОТВЕТЧИК для всех строк при условии: СИСТЕМНЫЙ_СБРОС.

НЕДЕЙСТВИТЕЛЬНЫЙ_ОТВЕТЧИК

Модуль должен устанавливать НЕДЕЙСТВИТЕЛЬНЫЙ_ОТВЕТЧИК при условии. КЕШИРОВАННЫЙ & (АГЕНТ_КЕША ¦ АГЕНТ_ПАМЯТИ) & (НЕДЕЙСТВИТЕЛЬНОСТЬ ¦ МОДИФИЦИРОВАННОЕ_ЧТЕНИЕ & СТАТУС_РАСЩЕПЛЕНИЯ) & -СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ & АГЕНТ_КЕША не может гарантировать СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ в конце передачи.

Модуль должен очищать НЕДЕЙСТВИТЕЛЬНЫЙ_ОТВЕТЧИК при условии: (ЗАДАТЧИК ¦ КЕШИРОВАННЫЙ & -РАСЩЕПЛЕНИЕ) & МОДИФИЦИРОВАННЫЙ_ОТВЕТ & (АГЕНТ_КЕША ¦ АГЕНТ_ПАМЯТИ) & СТАТУС_НЕДЕЙСТВИТЕЛЬНОСТИ ¦ КЕШИРОВАННЫЙ & НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ ¦ АГЕНТ_ПАМЯТИ & СТОЛКНОВЕНИЕ_НЕДЕЙСТВИТЕЛЬНОСТИ.

Модуль должен очистить НЕДЕЙСТВИТЕЛЬНЫЙ_ОТВЕТЧИК для всех строк при условии: СИСТЕМНЫЙ_СБРОС.

РАЗДЕЛЯЕМЫЙ_ОТВЕТЧИК

Модуль должен устанавливать РАЗДЕЛЯЕМЫЙ_ОТВЕТЧИК при условии: КЕШИРОВАННЫЙ & РАСЩЕПЛЕНИЕ & (РАЗДЕЛЯЕМОЕ_ЧТЕНИЕ ¦ НЕДЕЙСТВИТЕЛЬНОЕ_ЧТЕНИЕ).

Модуль должен очищать РАЗДЕЛЯЕМЫЙ_ОТВЕТЧИК при условии: ЗАДАТЧИК & РАЗДЕЛЯЕМЫЙ_ОТВЕТ ¦ НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ.

Модуль должен очистить РАЗДЕЛЯЕМЫЙ_ОТВЕТЧИК для всех строк при условии: СИСТЕМНЫЙ_СБРОС.

ОТВЕТЧИК_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ

Модуль должен устанавливать ОТВЕТЧИК_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ при условии: КЕШИРОВАННЫЙ & РАСЩЕПЛЕНИЕ & НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ.

Модуль должен очищать ОТВЕТЧИК_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ при условии: (ЗАДАТЧИК ¦ КЕШИРОВАННЫЙ & -РАСЩЕПЛЕНИЕ) & МОДИФИЦИРОВАННЫЙ_ОТВЕТ ¦ НЕДЕЙСТВИТЕЛЬНАЯ_ЗАПИСЬ.

Модуль должен очистить ОТВЕТЧИК_НЕДЕЙСТВИТЕЛЬНОЙ_ЗАПИСИ для всех строк при условии: СИСТЕМНЫЙ_СБРОС.

СТОЛКНОВЕНИЕ_НЕДЕЙСТВИТЕЛЬНОСТИ

Случается в магистральных мостах и поэтому не рассматривается в этой спецификации. СТОЛКНОВЕНИЕ_НЕДЕЙСТВИТЕЛЬНОСТИ должен быть установлен, когда передача недействительности с одной магистрали сталкивается с передачей недействительности с другой магистрали. СТОЛКНОВЕНИЕ_НЕДЕЙСТВИТЕЛЬНОСТИ должен быть очищен, когда столкновение разрешается.

8.2.6 **Определение протокола**

8.2.6.1 **Размер строки кеша**

Строка кеша должна состоять из 64 непрерывных байтов памяти, выровненных по модулю 64.

8.2.6.2 **Передачи строки кеша**

Все передачи строк кеша должны начинаться с первого слова строки и заканчиваться, когда строка передана целиком. Частичная передача строки не допускается и строка должна быть передана целиком в пределах одной передачи.

9 ПЕРЕДАЧА СООБЩЕНИЙ

9.1 Описание

Процедура передачи сообщений ФБ+ обеспечивает простую и эффективную связь между процессорами в системах ФБ+. На прикладном (или системном) уровне процессы в одном модуле системы ФБ+ посылают сообщения или принимают сообщения от процессов в других модулях. Уровни передачи сообщений, необходимые для поддержки большого числа аппаратных средств, являются объектом спецификации передачи сообщений.

Передача сообщений ФБ+ определена на двух уровнях:

1) уровень фрагментов обеспечивает основной протокол для передачи логического объекта, называемого фрагментом сообщения между модулями в пределах ФБ+ -системы;

2) уровень сообщений, использующий фрагменты сообщений определенные на уровне фрагментов, передает сообщения переменной длины между модулями.

Модули, обладающие способностью передачи сообщений, используют несколько предопределенных адресов внутри пространства управляющих и статусных регистров (РУС) в качестве почтовых ящиков, через которые производится передача фрагментов сообщений. Так как только фрагменты сообщений ФБ+ записываются в почтовые ящики в пространстве сообщений УСР, они могут быть просто отделены и являются совместимыми со всеми остальными передачами ФБ+, как показано на рис. 9—1.

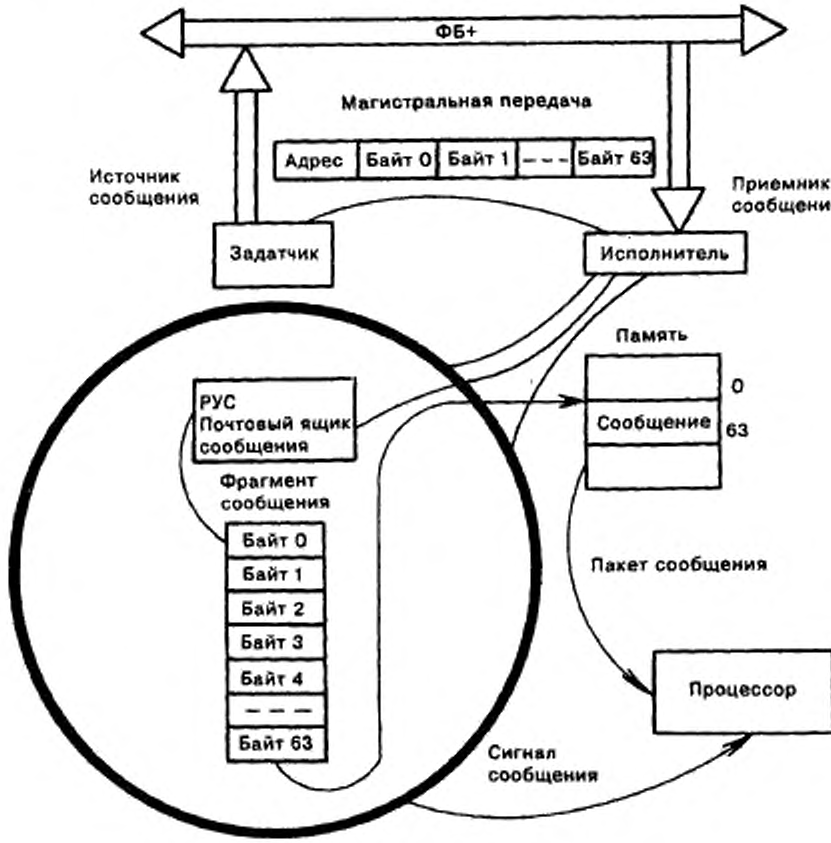

Рисунок 9—1 — Передача сообщений ФБ+

9.1.1 **Уровень фрагментов**

Фрагмент сообщения представляет собой блок данных, посылаемых задатчиком в почтовый ящик исполнителя, используя передачу записи. Могут использоваться передачи неблокированной записи или передачи записи без подтверждения. Команда передачи с записью без подтверждения - - это расширенная особенность передачи сообщений ФБ+. Почтовый ящик может быть представлен как «магическая» область. Это не область памяти, а окно, через которое передаются сообщения. Вышеописанная схема адресации позволяет модулям, передающим сообщения, связываться друг с другом независимо от типа процессора или аппаратных средств. Как только фрагмент сообщения был принят, модуль обрабатывает этот фрагмент, основываясь на информации, содержащийся в его заголовке.

9.1.1.1 **Структура фрагмента**

Так как фрагмент сообщения может передаваться между различными типами процессоров с различными внутренними представлениями данных, фрагмент сообщения будет определен так, как он появляется на магистрали. Как показано на рис. 9—2, различные квадлеты и байты, из которых состоит фрагмент сообщения, определяются так, как бы они выглядели на 32, 64, 128 и 256-разрядных магистралях.

Как показано на рисунке 9—3, 64-разрядный фрагмент сообщения составлен из заголовка и тела и передается через ФБ+ в почтовый ящик сообщений. Первые четыре байта фрагмента содержат заголовочную информацию, являющуюся общей для всех фрагментов сообщения ФБ+, и определяют протокол более высокого уровня, который будет использоваться. Тело фрагмента состоит из 60-байтного фрагмента данных.

5 полей имеются в заголовке фрагмента.

1) Тип команды записи выбирают команду передачи записи, которая будет использоваться. Если это поле нулевое, то используется полно связанная команда (передача незащищенной записи). Если это поле установлено в 1, то используется несвязанная команда (запись без подтверждения).

а) Порядок передач неблокированной записи сохраняется в пределах одной магистрали. Порядок передачи неблокированной записи не может быть сохранен в случае магистрального моста, если этот мост расщепляет передачу.

б) Порядок передач с записью без подтверждения не гарантирован. Гарантия целостности сообщения возлагается на протоколы передачи сообщения.

2) Поле протокола сообщения выбирает протокол более высокого уровня, используемый для обработки фрагмента. Поле протокола сообщения определяется следующим образом:

а) если величина поля протокола сообщения равна нулю, то фрагмент сообщения не является частью протокола более высокого уровня;

б) если величина поля протокола сообщения равна единице, то фрагмент сообщения является частью сообщения, которое согласуется с протоколом уровня сообщений ФБ+;

в) если величина поля протокола сообщения равна 2—63, то этот протокол уровня сообщений не определен этим стандартом, но зарезервирован для использования в будущем;

7) если величина поля протокола сообщения равна 64—79, то сообщение согласуется с протоколом уровня сообщений, определенным одним из подобных стандартов;

д) если величина поля протокола сообщения равна 80—127, то сообщение согласуется с одним из протоколов уровня зависимых сообщений производителя.

3) Поле приоритета сообщения определяет приоритет, связанный с фрагментом сообщения. Поле приоритета сообщения представляет собой то же самое, что может быть найдено в фазе рассоединения передачи записи фрагмента сообщения. Рассмотрение действительных значений, используемых в поле приоритета, выходит за рамки этой спецификации.

4) Поле глобальной идентификации (глобальная идентификация младшая, глобальная идентификация старшая) определяет модуль, посылающий фрагмент сообщения. Как только фрагмент появляется на магистрали, поле глобальной идентификации есть то же самое, что и глобальная идентификация, описанная в 6.1.17.11.

Так как каждый фрагмент сообщения имеет величину протокола сообщения, несколько протоколов передачи сообщения могут быть использованы внутри одной и той же системы без влияния друг на друга. Если модуль получает фрагмент сообщения с протоколом сообщения, который не поддерживается этим модулем, модуль может игнорировать этот фрагмент сообщений.

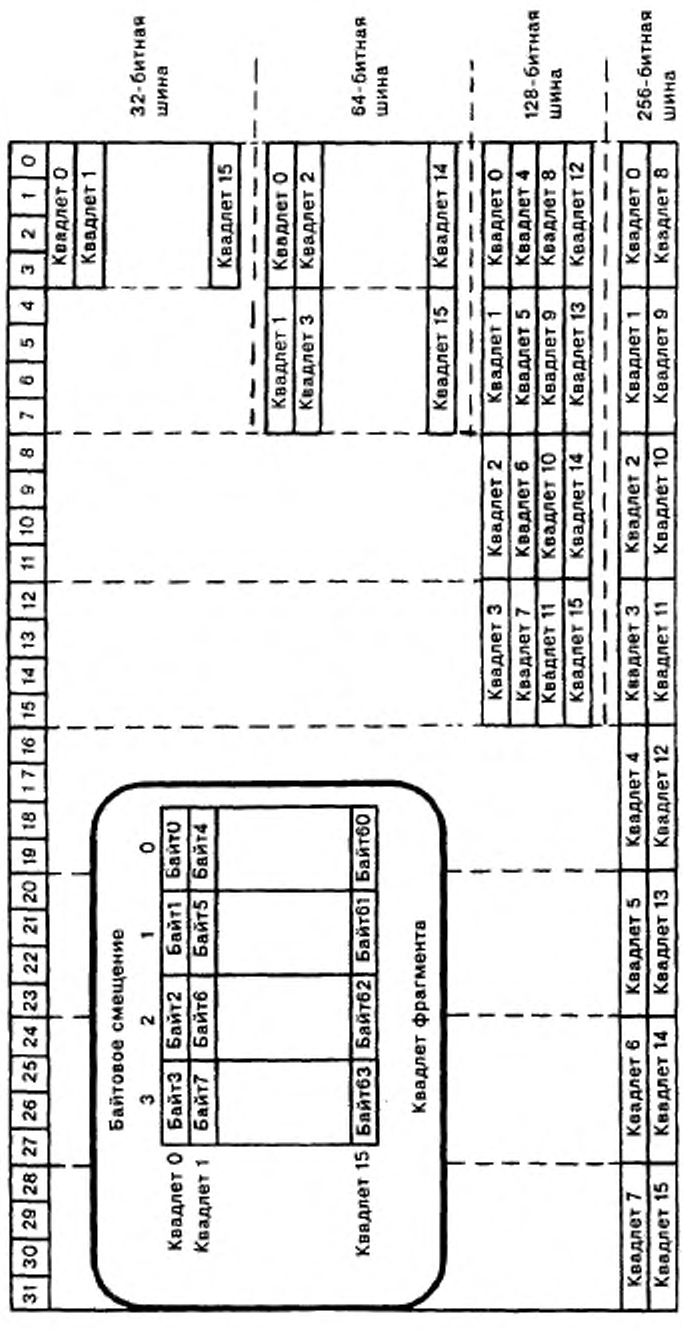

Рисунок 9—2 — Нотация фрагмента

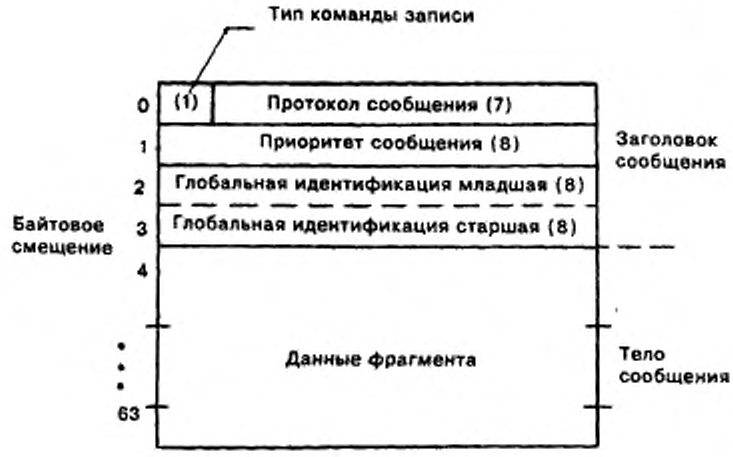

Рисунок 9—3 — Формат 64-байтного фрагмента

Размер фрагмента по умолчанию (РАЗМЕР_ФРАГМЕНТА_СООБЩЕНИЯ)-64 байта. Все ФБ+ модули, передающие сообщения, смогут передавать и принимать 64-байтные сообщения, используя 32-разрядную магистраль и вынужденные передачи неблокированной записи. Размеры фрагмента, разрядность магистрали и режимы передачи, поддерживаемые обоими, посылающим и принимающим, модулями, отличающиеся от принятых по умолчанию, поддерживаются при условии их поддержки самими модулями. Модули, способные работать с фрагментами больших размеров или большей разрядностью магистрали, совместимы снизу с модулями, имеющими размер фрагмента и разрядность магистрали по умолчанию. Фрагмент сообщения с размером меньше, чем 64 байта, может передаваться между модулями, когда протокол сообщений поддерживает эту особенность. Модуль должен избегать пересылки к модулям фрагментов, размерами большим, чем они поддерживают, так как возникнет ошибка передачи. Фрагменты сообщений могут быть переданы в фрагментном режиме, если размер фрагмента согласуется с допустимым размером фрагмента и исполнитель способен участвовать в фрагментной передаче.

Максимальный размер фрагмента по умолчанию, равный 64 байтам, был выбран для поддержания совместимости с другими аспектами ФБ+ (размер строки кеша и передачи в фрагментном режиме). Большие фрагменты выборочно могут быть использованы, если возникает необходимость. Так как буфер для больших фрагментов может быть использован для получения маленьких фрагментов, но не наоборот, соответствие операций между модулями, имеющими различные размеры фрагментов, должно быть обеспечено. Как описано в 7.2, каждый модуль, который поддерживает передачу сообщений, имеет регистр совместимого размера фрагмента сообщения, так что можно определить, какие размеры фрагментов будут совместимы.

9.1.1.2 **Структура почтового ящика**

Каждый модуль, поддерживающий передачу сообщений, имеет структуру почтового ящика такую же, которая показана на рис. 9—4. Во избежание тупиковых ситуаций, возникающих при использовании одиночного почтового ящика, каждый модуль, передающий сообщения, имеет почтовый ящик для запросов и ответов. Почтовые ящики могут принимать как широковещательные, так и адресованные к одному исполнителю передачи.

Почтовые ящики для сообщений предоставляют механизм для различения передач сообщения от остальных передач ФБ+. Рассмотрение адресов почтовых ящиков для сообщений выходит за рамки этого документа.

Исполнительные модули, передающие сообщения, имеет средства для приема фрагмента с магистрали и передачи его на запоминающее устройство, такое как аппаратная очередь, память, строка кеша или регистр.

Аппаратуре разрешено объединять почтовые ящики для запросов и ответов, если могут поддерживаться операции без тупиков. Модуль, который объединяет почтовые ящики запросов и ответов, должен распознавать оба адреса.

Аппаратура, требующая сохранения порядка для передач сообщений, может делать это посредством использования одного почтового ящика (запроса или ответа). Протоколы уровней сообщений, которые сохраняют порядок передач сообщений, посредством использования одного почтового ящика сообщений, должны также обеспечивать операции без тупиков.

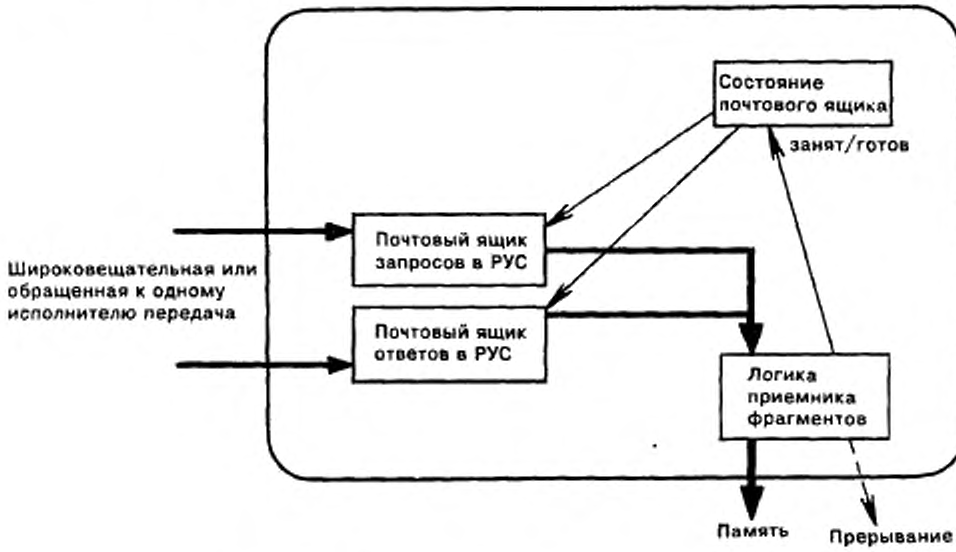

Рисунок 9—4 — Почтовые ящики для передачи сообщений

9.1.1.2.1 **Почтовый ящик запросов**

Почтовый ящик запросов и связанная с ним очередь используются для фрагментов сообщений, которые требуют возвращения подтверждающего сообщения в запрашивающий модуль. Если задатчик посылает фрагмент сообщения в почтовый ящик запросов исполнителя, то ожидается, что исполнитель вернет сообщение ответа в почтовый ящик ответов задатчика, как требует протокол.

9.1.1.2.2 **Почтовый ящик ответов**

Фрагменты подтверждения посылаются в почтовый ящик ответов запрашивающего модуля.

Почтовый ящик ответов и связанная с ним очередь используются для фрагментов сообщений, которые не требуют подтверждения. Если задатчик посылает фрагмент сообщения в почтовый ящик ответов исполнителя, то обратное сообщение не требуется. Сообщения, пересылаемые таким образом, являются по характеру сообщениями от точки к точке и имеют единственное место назначения.

9.1.1.2.3 **Широковещательные передачи**

Могут посылаться в почтовые ящики ответов и запросов всех передающих сообщения модулей на магистрали. Широковещательные передачи не должны подтверждаться. Требуется, чтобы модули, получающие широковещательное сообщение, переходили в широковещательный режим, выставляя bc★. Широковещательные передачи посылаются по уникальному адресу модуля.

Фрагменты сообщений, передаваемые в широковещательном режиме, имеют атрибуты, которые позволяют им посылаться выборочно, как показано на рис. 9—5. Когда декодируется адрес широковещательной передачи, модули используют шесть младших разрядов адреса. Если значение шести младших разрядов адреса равно нулю, исполнитель всегда принимает сообщение. Если значение шести младших разрядов адреса не равно нулю, они используются для адресации одного из 63 атрибутов выбора широковещательного почтового ящика, содержащихся в регистре селектируемой маски, как описано в 7.1 (СЕЛЕКТИРУЕМАЯ_МАСКА_ПРОХОЖДЕНИЯ_СООБЩЕНИЯ). Если адресованный атрибут выбора очищен, модуль игнорирует эту передачу и не выставляет sl★ в фазе соединения. При помощи установки соответствующего атрибута выбора, фрагменты сообщения эффективно фильтруются перед тем, как они могут поступить в модуль. Эта особенность передачи сообщений ФБ+ позволяет сообщениям передаваться широковещательно через ФБ+ и в то же время еще позволяет исполнителям выборочно принимать эти сообщения.

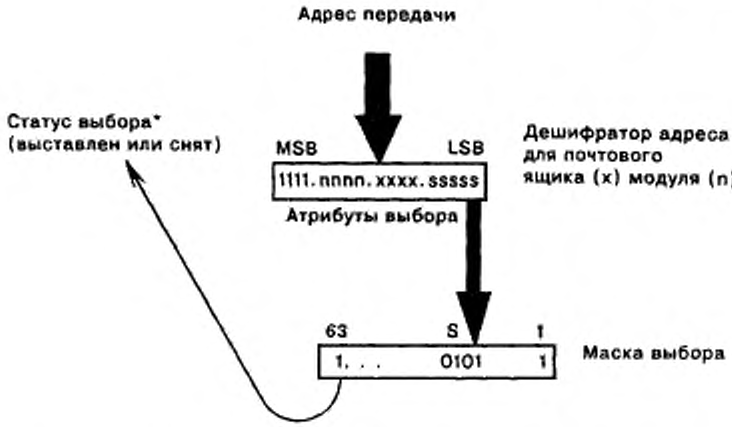

Рисунок 9—5 — Атрибуты выбора широковещательного почтового ящика

Некоторые исполнители, чей почтовый ящик занят, вызывают окончание передачи с be★ статусом. Широковещательный фрагмент является очень эффективным для тех применений, в которых требуется, чтобы много модулей получали одни и те же сообщения в одно и то же время. Однако неразборчивое использование широковещательных фрагментов может вызвать проблемы с производительностью и чрезмерные повторения из-за занятости почтовых ящиков. При помощи использования атрибутов выбора вероятность столкновения с занятым почтовым ящиком может быть уменьшена.

Рекомендуется, чтобы широковещание сообщения использовалось умеренно, если специальное обеспечение не используется для уменьшения возможности занятости почтового ящика.

9.1.1.3 **Протокол уровня фрагментов**

Неблокированная запись или запись без подтверждения может использоваться для передачи фрагмента сообщения в почтовый ящик сообщений. Неблокированная адресная передача может использоваться для определения текущего статуса почтового ящика. Адресные передачи не изменят состояние почтового ящика (занятость или конфликт сообщений). Модули выставляют be★ для индикации ошибки, если любой другой тип передач получен в передаче к почтовому ящику.

Модули, которые поддерживают передачу сообщений, выставляют sl★ в ответ на все передачи к почтовым ящикам запросов и ответов. Модули, передающие сообщения, выставляют sl★ и bc★ в ответ на все передачи к широковещательным почтовым ящикам, если соответствующий атрибут выбора установлен или AD[5 . . . 0]★ освобождены.

Если задатчик пытается послать фрагмент, который больше чем поддерживает исполнительный модуль, исполнитель выставит te★, а задатчик, увидя выставленным TE★, должен воспринимать это как ошибку. Исполнители выставляют ed★ в течение последней передачи, которую они могут разместить. Если задатчик получает ED★ статус, он должен закончить передачу, т. к. исполнительный модуль не может больше принимать данные. Если модуль принимает передачу в почтовый ящик, но временно не способен получать данные, он выставляет bs★. Первоначальный модуль должен позднее инициировать повтор. Модули используют статус занятости для предотвращения записи входного фрагмента поверх необработанного фрагмента.

Модули, передающие сообщения, будут выставлять или отпускать tf★ посредством установки или сброса КОНФЛИКТ_СООБЩЕНИЯ.

После системного сброса модуль, передающий сообщения, установит КОНФЛИКТ_СООБЩЕНИЯ для всех передач к любому из почтовых ящиков сообщений этого модуля до тех пор, пока почтовый ящик сообщений модуля не инициализирован должным образом.

После того как почтовый ящик сообщений модуля должным образом инициализирован, он установит КОНФЛИКТ_СООБЩЕНИЯ, когда он не будет способен принять фрагмент сообщения вследствие ограниченности ресурсов, которые вероятнее всего не будут доступны своевременно (время гораздо больше, чем задержка повтора занятости).

После получения TF★ статуса для операции передачи сообщения посылающий модуль прервет передачу. Протокол восстановления более высокого уровня в посылающем модуле ответственен за попытку повторения передачи в то время, когда наиболее вероятно, что передача сообщения будет успешна (время гораздо больше, чем задержка повтора занятости).

9.1.1.4 **Приоритет сообщения**

Каждый фрагмент сообщений имеет поле приоритета. Процедура использования приоритета фрагмента для арбитража за магистраль выходит за рамки этого описания.

Где используются очереди, фрагменты сообщений в обоих передающей и принимающей очередях должны устанавливаться в приоритетном порядке. Фрагменты с одинаковым приоритетом должны быть упорядочены по принципу первым вошел, первым вышел. Если используется короткая аппаратная очередь передач, и эта очередь полностью заполнена фрагментами с низшими приоритетами (что ведет к изменению порядка приоритетов), то должны быть предусмотрены механизмы, разрешающие предочистку фрагмента с низким приоритетом в пользу фрагмента с высоким приоритетом.

9.1.1.5 **Пример уровня фрагментов**

Пример возможного исполнения логики приемника фрагментов показан на рис. 9—6. Она основана на компоновке блоков сообщений, использующих механизм списка связей, обеспечивающий эффективный метод приема последовательностей фрагментов. Процесс приема фрагментов происходит следующим образом.

Рисунок 9—6 — Передающий сообщения по списку связей

1) До приема любых сообщений регистр указателя фрагмента и указатель связанного списка указывают на одну и ту же область памяти.

2) Другой модуль затем передает фрагмент сообщения во входной буфер фрагментов модуля.

3) Логика контроля передачи сообщений передает данные из входного буфера фрагментов в блок памяти, адресуемый посредством указателя связанного списка.

4) Используя указатель на следующий фрагмент, содержимое регистра указателя фрагмента затем изменяется, чтобы указывать на следующий вход в связанном списке.

5) Логика контроля передачи сообщений затем извещает процессор, что фрагмент сообщения принят, и сбрасывает bs★.

6) Посылающий модуль затем передает другой фрагмент сообщения во входной буфер сообщений модуля.

7) В то время, когда логика передачи сообщений передает данные из входного буфера фрагментов в блок памяти, адресуемый указателем связанного списка, какой-либо другой модуль шлет другое сообщение в этот модуль.

8) Принимающий модуль выставляет bs★, индицируя, что модулю следует повторить передачу позднее.

9) Когда передача из входного буфера фрагментов завершена, регистр указателя фрагмента изменяется для того, чтобы указывать на следующий вход в связанный список.

10) Логика контроля передачи сообщений затем извещает (т. е. прерывает) процессор, что другой фрагмент сообщений принят.

11) Посылающий модуль пытается снова и передает фрагмент сообщений в входной буфер фрагментов модуля.

12) Логика контроля передачи сообщений передает данные из входного буфера фрагментов в блок памяти, адресуемый указателем связанного списка.

13) Указатель следующего фрагмента равен нулю в этот момент, поэтому регистр указателя фрагмента установлен в нуль.

14) Логика контроля передачи сообщений затем извещает процессор, что другой фрагмент сообщений принят.

15) На любом шаге после шага (5) процессор может удалить сообщение из связанного списка, начиная с сообщения, адресуемого указателем связанного списка. После того как сообщение удалено, указатель связанного списка изменяется на указатель следующего фрагмента. Этот процесс может продолжаться до тех пор, пока указатель связанного списка не равен регистру указателя фрагмента. В дополнение к этому, процессор может еще добавлять пустые фрагменты в конец связанного списка.

9.1.2 **Уровень сообщений**

Следующие ниже описывают тот случай, когда поле протокола сообщения, определенное в заголовке фрагмента, равно единице.

9.1.2.1 **Структура фрагмента уровня сообщений**

Сообщение представляет собой логическую единицу, которая состоит из одного или нескольких фрагментов сообщения. Фрагмент уровня сообщений состоит из 8- или 12-байтного заголовка и следующего за ним фрагмента данных переменной длины, как показано на рис. 9—7 (в целях иллюстрации, пакет уровня сообщений показан как 64-байтный фрагмент). Содержимое фрагмента уровня сообщений определяется типом фрагмента.

Поле протокола сообщений фрагмента уровня сообщений такое же, как и в уровне фрагментов. Поле протокола сообщений равно единице для сообщений уровня сообщений ФБ—. Приоритет сообщения и поля глобальной идентификации фрагмента такие же как, и в уровне фрагментов.

Следующий раздел объясняет оставшуюся часть заголовка уровня сообщений.

9.1.2.1.1 **Тип фрагмента**

Поле типа фрагмента показывает, что фрагмент имеет один из следующих типов:

1) фрагмент неподтвержденного события,
2) фрагмент подтвержденного события;
3) фрагмент положительного подтверждения события;
4) фрагмент отрицательного подтверждения события;
5) фрагмент запроса многофрагментного сообщения;
6) фрагмент положительного ответа многофрагментного сообщения;
7) фрагмент отрицательного ответа многофрагментного сообщения;
8) фрагмент упорядоченных данных многофрагментного сообщения;
9) фрагмент последовательных данных многофрагментного сообщения;
10) последний упорядоченный фрагмент многофрагментного сообщения;
11) последний последовательный фрагмент многофрагментного сообщения;
12) фрагмент положительного подтверждения многофрагментного сообщения;
13) фрагмент отрицательного подтверждения многофрагментного сообщения.

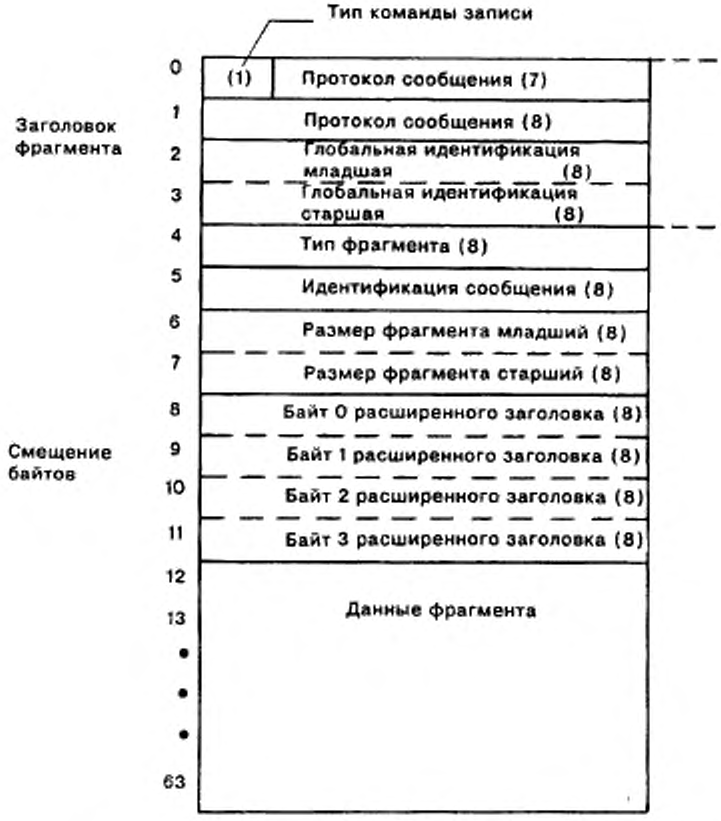

Рисунок 9—7 — Структура фрагмента уровня сообщений ФВ+

Другие типы фрагментов зарезервированы для применения в будущем или для фрагментов, определяемых пользователем.

Если старший разряд типа фрагмента установлен, запрос внимания (т. е. прерывание) будет сгенерирован в модуле, принимающем фрагмент.

Все типы фрагментов совместимы с передачей неблокированной записи. Все типы фрагментов кроме фрагмента упорядоченных данных многофрагментного сообщения совместимы с передачей с не подтвержденной записью. Фрагменты последовательных данных требуются, только если используется тип команды с необязательной записью без подтверждения.

9.1.2.1.2 **Размер фрагмента**

Поля размера фрагмента (Размер Фрагмента Младший и Размер Фрагмента Старший) показывают общее число байтов, занимаемых фрагментом (заголовок фрагмента, заголовок сообщения и тело сообщения). Хотя максимальный размер фрагмента определяется аппаратным обеспечением модуля, более короткие фрагменты также могут передаваться. Минимальный размер фрагмента — восемь байт. Это позволяет модулям посылать полный диапазон размеров фрагментов другим модулям.

9.1.2.1.3 **Идентификация сообщения**

Поле идентификации сообщения обеспечивает уникальный идентификатор для отслеживания сообщений. С ним модуль может инициировать различные одно- и многофрагментные сообщения к одному или более конечным модулям. Конечный модуль использует идентификацию сообщения в сочетании с глобальной идентификацией для управления своими приемными буферами. Так как фрагмент в многофрагментном сообщении будет иметь ту же самую величину идентификации сообщения, смешивание фрагментов более чем от одного источника может быть коррелировано с их соответствующими буферами сообщений.

9.1.2.1.2 **Расширенный заголовок**

В заголовке уровня сообщений является не обязательным. Содержимое расширенного заголовка меняется в зависимости от типа фрагмента. Расширенный заголовок состоит из Байта 0 Расширенного Заголовка до Байта 3 Расширенного Заголовка в порядке старшинства, где Байт 0 Расширенного Заголовка является наименьшим значащим байтом.

Когда тип фрагмента не требует расширенного заголовка, поле расширенного заголовка становится частью поля данных фрагмента

9.1.2.1.5 **Размер сообщения**

Для некоторых типов фрагментов расширенный заголовок является размером сообщения. Размер сообщения есть сумма всех байтов сообщения, передаваемого на уровень сообщений из прикладного уровня. В многофрагментном сообщении упорядоченных данных размер сообщения представляет собой сумму всех байт тел фрагментов упорядоченных данных и последнего упорядоченного фрагмента. Для последовательных многофрагментных сообщений размер сообщения представляет собой сумму всех байт тел фрагментов последовательных данных и последнего последовательного фрагмента.

9.1.2.1.6 **Интервал фрагментов**

Для некоторых типов фрагментов расширенный заголовок есть интервал фрагментов. Интервал фрагментов посылается запрашивающему модулю для индикации минимального времени между фрагментами многофрагментного сообщения. Использование интервалов короче чем этот интервал может привести к занятости почтового ящика приемника. Минимальное время между фрагментами будет определяться в значительной степени существованием аппаратной поддержки для многофрагментных сообщений.

9.1.2.1.7 **Номер последовательности**

Для некоторых типов фрагментов расширенный заголовок является номером последовательности. Номер последовательности есть монотонно увеличивающийся номер. Первому фрагменту в серии назначается номер один (1).

9.1.2.1.8 **Тип исключения**

Для некоторых типов фрагментов расширенный заголовок является типом исключения. Следующие типы исключения определены для всех фрагментов с отрицательным ответом:

1) недопустимый протокол сообщения;
2) недопустимый приоритет;
3) недопустимый размер фрагмента;
4) недопустимая идентификация сообщения;
5) недопустимая глобальная идентификация;
6) модуль не поддерживает тип фрагмента последовательных данных;
7) потерянный фрагмент;
8) тайм-аут.

9.1.2.1.9 **Фрагмент неподтвержденного события**

Используется для посылки фиксированного количества данных в почтовый ящик ответа. Тело фрагмента используется для передачи данных. Приемник этого фрагмента не будет подтверждать его получение при помощи фрагмента ответа. Идентификация сообщения не является необходимой для этого типа фрагмента.

9.1.2.1.10 **Фрагмент подтвержденного события**

Используется для посылки фиксированного количества данных в почтовый ящик запросов. Тело фрагмента используется для передачи данных. Получатель этого фрагмента будет подтверждать его получение посылкой фрагмента положительного подтверждения события, если фрагмент подтвержденного события принят успешно. Получатель этого фрагмента будет подтверждать его получение посылкой фрагмента отрицательного подтверждения события, если фрагмент подтвержденного события не принят успешно.

9.1.2.1.11 **Фрагмент положительного подтверждения события**

Посылается в почтовый ящик ответа модуля первоначально посылающего фрагмент подтвержденного события, который был успешно принят и обработан. Идентификация сообщения фрагмента положительного подтверждения события должна быть такой же, как и идентификация события, полученная в фрагменте подтвержденного события. Остальная часть фрагмента не используется.

9.1.2.1.12 **Фрагмент отрицательного подтверждения события**

Посылается в почтовый ящик ответов модуля первоначально посылающего фрагмент подтвержденного события, который не был успешно обработан. Идентификация сообщения фрагмента отрицательного подтверждения события должна быть такой же, как и идентификация события, полученная в фрагменте подтвержденного события. Расширенный заголовок используется в этом типе сообщения для индикации причины отказа. Остальная часть фрагмента может быть использована для дополнительной информации об отказе.

9.1.2.1.13 **Фрагмент запроса многофрагментного сообщения**

Для передачи данных, больших чем может содержаться в одном фрагменте, модуль может инициировать последовательность многофрагментных сообщений. Фрагмент запроса многофрагментного сообщения посылается в почтовый ящик запросов для инициации многофрагментного сообщения. Расширенный заголовок содержит размер сообщения многофрагментного сообщения. Идентификация сообщения используется как уникальный идентификатор последовательной передачи многофрагментного сообщения. Модуль, первоначально посылающий запрос фрагмента многофрагментного сообщения, получит в ответ либо фрагмент положительного ответа многофрагментного сообщения, либо фрагмент отрицательного ответа многофрагментного сообщения. Остальная часть фрагмента не используется.

9.1.2.1.14 **Фрагмент положительного ответа многофрагментного сообщения**

Посылается в почтовый ящик ответов, если условия позволяют удовлетворить фрагмент запроса многофрагментного сообщения. Идентификация сообщения в фрагменте ответа должна быть такая же, как идентификация сообщения в фрагменте запроса. Расширенный заголовок используется в этом типе фрагмента для передачи интервала между фрагментами. Остальная часть фрагмента не используется.

9.1.2.1.15 **Фрагмент отрицательного ответа многофрагментного сообщения**

Посылается в почтовый ящик ответов, если условия не позволяют удовлетворить фрагмент запроса многофрагментного сообщения. Идентификация сообщения в фрагменте ответа должна быть такая же, как идентификация сообщения в фрагменте запроса. Расширенный заголовок используется в этом типе фрагмента для передачи кода исключения. Остальная часть фрагмента может быть использована для дополнительной информации об отказе.

9.1.2.1.16 **Фрагмент упорядоченных данных многофрагментного сообщения**

Когда модуль, инициировавший многофрагментное сообщение, получает фрагмент положительного ответа, он должен начать посылать фрагменты упорядоченных данных в почтовый ящик запросов принимающего модуля, если выбраны команды передачи неблокированной записи. Для предотвращения насыщенности принимающего модуля, внутрифрагментный интервал, определяемый интервалом фрагментов, будет посылаться до того, когда начнет посылаться последующий фрагмент упорядоченных данных. Каждый пакет упорядоченных данных будет содержать фрагмент сообщения. Получатель этих фрагментов сообщения будет собирать сообщение. Идентификация сообщения будет такая же, как в фрагменте запроса многофрагментного сообщения, который инициировал последовательность сообщения. Идентификация сообщения будет использоваться для связывания фрагмента сообщения с его буфером. Порядок фрагментов упорядоченных данных будет гарантирован по всей системе. Этот тип фрагмента совместим только с передачами неблокированной записи. Передача неблокированной записи оптимальна для локальных, по отношению к магистрали, сообщений.

9.1.2.1.17 **Фрагмент последовательных данных многофрагментного сообщения**

Когда модуль, инициирующий многофрагментное сообщение, принимает фрагмент положительного ответа, он должен начать посылать фрагменты последовательных данных в почтовый ящик запросов принимающего модуля, если выбрана команда без подверження. Для предотвращения насыщенности принимающего модуля, внутрифрагментный интервал, определяемый интервалом фрагментов, будет посылаться до того, когда начнет посылаться последующий фрагмент последовательных данных. Каждый фрагмент последовательных данных будет содержать фрагмент сообщения. Получатель этих фрагментов сообщения будет собирать сообщение. Идентификация сообщения будет такая же, как в фрагменте запроса многофрагментного сообщения, который инициировал последовательность сообщения. Идентификация сообщения будет использоваться для связывания фрагмента сообщения с его буфером. Монотонно увеличивающийся Номер Последовательности будет помещен в расширенный заголовок для каждого фрагмента последовательных данных. Первый фрагмент последовательных данных будет иметь номер один (1). Нет требований для гарантии порядка фрагментов последовательных данных.

Хотя этот тип фрагмента был специально разработан для использования с командой записи без подтверждения, он совместим с командой передачи неблокированной записи. В дополнение к этому, так как команда записи без подтверждения не является обязательной, последовательные фрагменты также не обязательны. Передача записи без подтверждения необязательна систем, использующих магистральные мосты.

9.1.2.1.18 **Последний упорядоченный фрагмент многофрагментного сообщения**

Посылается запрашивающим модулем в почтовый ящик запросов для индикации последнего упорядоченного фрагмента многофрагментного сообщения. После получения модулем всех фрагментов данных многофрагментного сообщения, он будет проверять, что было получено запрашиваемое число байт. Идентификация сообщения будет такая же, как в фрагменте запроса многофрагментного сообщения, который инициировал последовательность сообщения. Принимающий модуль пошлет фрагмент подтверждения в почтовый ящик ответов запрашивающего модуля, индицируя успех или ошибку многофрагментного сообщения.

9.1.2.1.19 **Последний последовательный фрагмент многофрагментного сообщения**

Посылается запрашивающим модулем в почтовый ящик запросов для индикации последнего фрагмента последовательного многофрагментного сообщения. Когда модуль получает этот фрагмент, он укажет окончательный последовательный номер последовательного многофрагментного сообщения в расширенном заголовке. Идентификация сообщения будет такая же, как в фрагменте запроса многофрагментного сообщения, который инициировал последовательность сообщения. Принимающий модуль пошлет пакет подтверждения в почтовый ящик ответов запрашивающего модуля, индицируя успех или ошибку многофрагментного сообщения.

9.1.2.1.20 **Фрагмент положительного подтверждения многофрагментного сообщения**

Посылается в почтовый ящик ответов, если многофрагментное сообщения было успешно принято. Идентификация сообщения фрагмента подтверждения должна быть такая же, как и идентификация сообщения, принятая в фрагменте запроса. Остальная часть фрагмента не используется.

9.1.2.1.21 **Фрагмент отрицательного подтверждения многофрагментного сообщения**

Посылается в почтовый ящик ответов, если многофрагментное сообщения не было успешно принято. Идентификация сообщения фрагмента подтверждения должна быть такая же, как и идентификация сообщения, принятая в фрагменте запроса. Расширенный заголовок используется в этом типе фрагмента для указания кода исключения. Остальная часть фрагмента может быть использована для дополнительной информации об отказе.

Фрагмент отрицательного подтверждения может быть послан в любое время в течение последовательности многофрагментного сообщения и будет приводить к окончанию этой последовательности.

9.1.2.2 **Протоколы сообщений**

9.1.2.2.1 **Протокол для одиночного неподтвержденного фрагмента**

Сообщение из одиночного неподтвержденного фрагмента просто состоит из фрагмента одиночного сообщения, передающегося в почтовый ящик ответов, как показано на рис. 9—8.

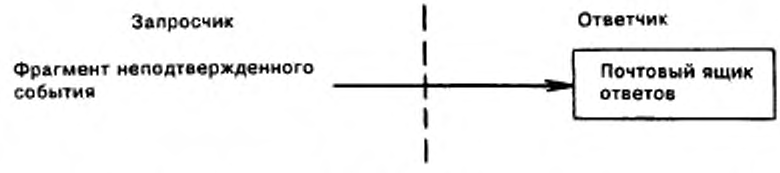

Рисунок 9—8 — Сообщение из одиночного неподтвержденного фрагмента

9.1.2.2.2 **Протокол для одиночного подтвержденного фрагмента**

Сообщение простого подтвержденного фрагмента состоит из двух шагов, как показано на рис. 9—9:

1) сначала фрагмент подтвержденного события передается в почтовый ящик запросов модуля;

2) затем модуль, получающий фрагмент подтвержденного события, отвечает фрагментом подтверждения, индицируя, что сообщение принято успешно.

Рисунок 9—9 — Сообщение из простого подтвержденного фрагмента

Как показано на рис. 9—10, посылка упорядоченного многофрагментного сообщения производится следующим образом

1) фрагмент запроса многофрагментного сообщения посылается в почтовый ящик запросов ответчика. Фрагмент запроса покажет, что запрашивается упорядоченное многофрагментное сообщение. Фрагмент запроса также индицирует размер сообщения;

2) ответчик затем посылает фрагмент положительного ответа в почтовый ящик запросчика для индикации того, что он может принять сообщение. Фрагмент ответа также показывает максимальную скорость, с которой запросчик должен посылать фрагменты;

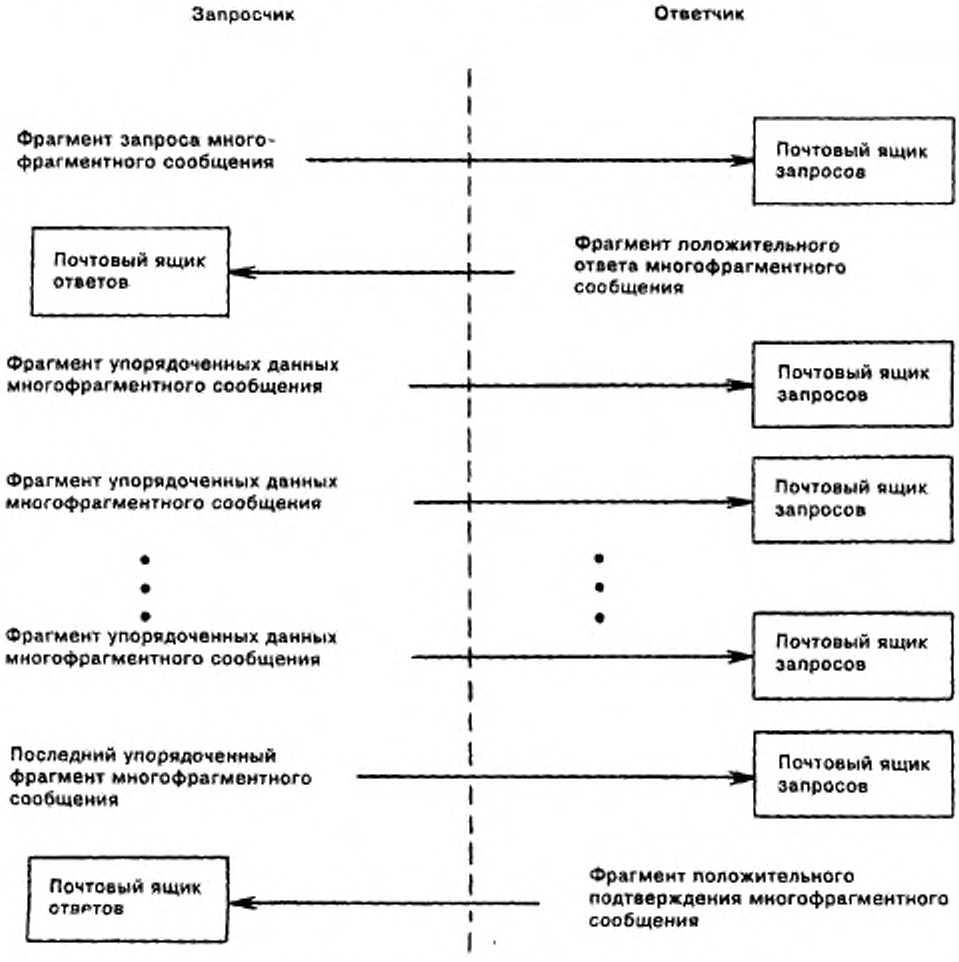

Рисунок 9—10 — Сообщение из множества упорядоченных фрагментов

3) запросчик затем посылает фрагменты упорядоченных данных ответчику;

4) запросчик посылает нуль или более полные фрагменты до тех пор, пока оставшиеся данные помещаются в один фрагмент. Тогда запросчик посылает последний упорядоченный фрагмент. Это указывает ответчику, что передача сообщения завершена;

5) ответчик затем посылает фрагмент положительного подтверждения запросчику для индикации того, что он принял сообщение правильно.

9.1.2.2.4 **Протокол для множественных последовательных фрагментов**

Как показано на рисунке 9—11, посылка последовательного многофрагментного сообщения производится следующим образом:

1) фрагмент запроса многофрагментного сообщения посылается в почтовый ящик запросов ответчика. Фрагмент запроса покажет, что запрашивается последовательное многофрагментное сообщение. Фрагмент запроса также индицирует размер сообщения;

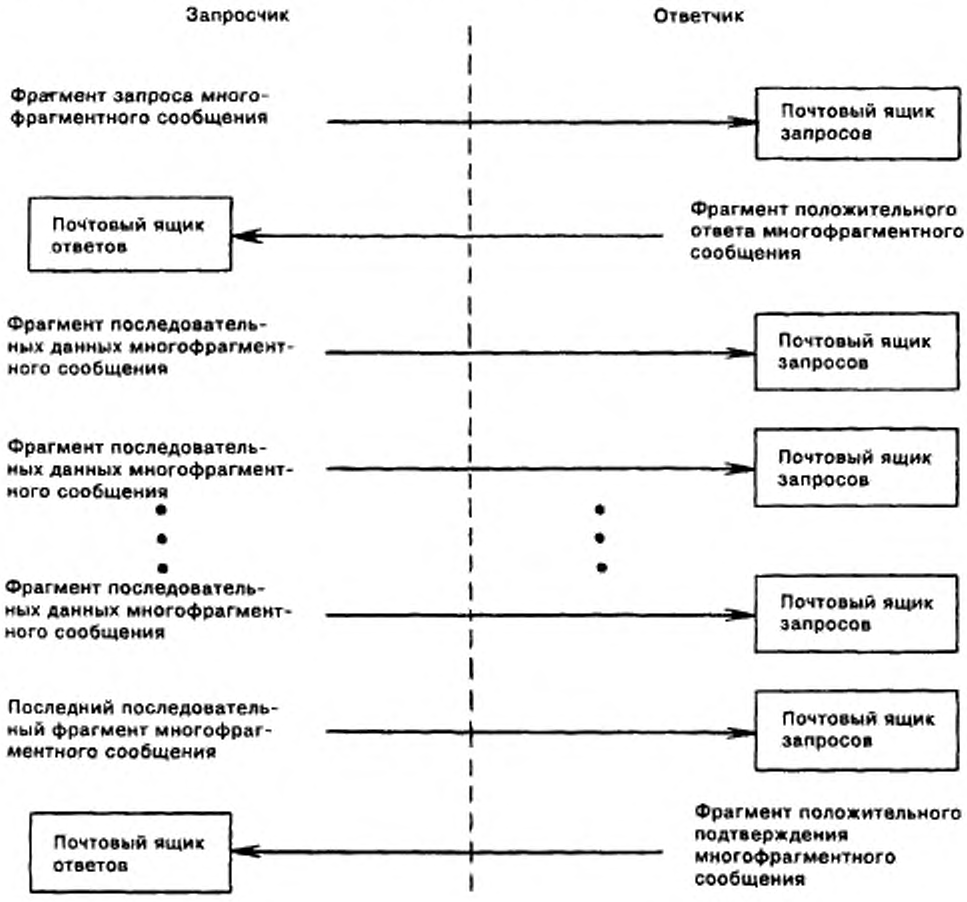

Рисунок 9—11 — Сообщение из множества последовательных фрагментов

2) ответчик затем посылает фрагмент положительного ответа в почтовый ящик запросчика для индикации того, что он может принять сообщение. Фрагмент ответа также показывает максимальную скорость, с которой запросчик должен посылать фрагменты;

3) запросчик затем посылает фрагменты последовательных данных ответчику. Запросчик будет включать в расширенный заголовок каждого фрагмента последовательный номер;

4) запросчик посылает нуль или более полные фрагменты до тех пор, пока оставшиеся данные помещаются в один фрагмент. Тогда запросчик посылает последний последовательный фрагмент. Это указывает ответчику, что передача сообщения завершена;

5) ответчик затем посылает фрагмент положительного подтверждения запросчику для индикации того, что он принял сообщение правильно.

9.1.2.3 **Тайм-аут запроса/ответа**

На базе принципа передачи фрагмента за фрагментом, посылающий модуль должен установить максимальное время, которое может истекать между фрагментами запроса и ответа. Если время тайм-аута истечет, ошибка будет передана на прикладной уровень. Величина этого максимального времени находится за рамками данного документа.

9.1.2.4 **Тайм-аут многофрагментного сообщения**

Запрашивающий и отвечающий модули, участвующие в многофрагментной передаче, установят максимальное время, которое может пройти от начала многофрагментного сообщения до его успешного или неуспешного завершения. Если время тайм-аута истечет, многофрагментная последовательность будет отменена фрагментом, показывающим исключение по тайм-ауту. Все фрагменты, связанные с многофрагментным сообщением, которые вышли за границы по времени, должны быть очищены. Модуль, запрашивающий многофрагментное сообщение, может повторить последовательность. Такая же идентификация сообщения используется для повтора, какая была использована для первоначальной попытки многофрагментного сообщения. Величина этого максимального времени находится за рамками данного документа.

9.1.2.5 **Упорядочение фрагментов**

Протокол уровня сообщений предполагает, что упорядочение пакетов сохраняется, когда существует связь между фрагментами сообщения. Упорядочение пакетов не является необходимым, когда не существует связи между фрагментами сообщения. Когда связь существует, такая как и в случае «распрашивающих» сообщений, упорядочение фрагментов должно быть сохранено. Предпочтительный механизм для сохранения упорядочения — это назначение связанным фрагментам того же приоритета. Когда упорядочение важно, обычно используется передача неблокированной записи.

9.1.2.6 **Время владения магистралью**

Все пользователи передающих сообщения модулей должны остерегаться инициирования передач сообщений с длительным владением, так как другие передачи с более высоким приоритетом могут быть заблокированы для пересылок. Определением протокола более высокого уровня, который разбивает длинные сообщения на короткие фрагменты сообщений, последовательность фрагментов может передаваться между модулями. В заключении каждой фрагментной передачи магистраль может быть оставлена передаче с более высоким приоритетом. Если никакой другой задатчик не потребовал доступа к магистрали, следующий фрагмент может быть отправлен. Таким образом, большие сообщения могут передаваться эффективно без штрафов, связанных с долгими владениями магистралью.

9.2 **Спецификация**

9.2.1 **Описание атрибутов**

ИНИЦИАЛИЗАЦИЯ_ПОЧТОВОГО_ЯЩИКА

Модуль должен установить ИНИЦИАЛИЗАЦИЯ_ПОЧТОВОГО_ЯЩИКА при условии. ИНИЦИАЛИЗАЦИЯ и удерживать его установленным, пока: -КОНФЛИКТ_ПОЧТОВОГО_ЯЩИКА_ЗАПРОСОВ & -КОНФЛИКТ_ПОЧТОВОГО_ЯЩИКА_ОТВЕТОВ.

ПОЧТОВЫЙ_ЯЩИК_ЗАПРОСОВ

Модуль должен установить ПОЧТОВЫЙ_ЯЩИК_ЗАПРОСОВ при условии. АДРЕС_ДЕКОДИРОВАН и адрес соответствует почтовому ящику запросов модуля или ШИРОКОВЕЩАТЕЛЬНЫЙ, как определено в р 896.2. Модуль должен удерживать ПОЧТОВЫЙ_ЯЩИК_ЗАПРОСОВ установленным, пока: ИНИЦИАЛИЗАЦИЯ | ПЕРЕДАЧА_ЗАКОНЧЕНА.

ПОЧТОВЫЙ_ЯЩИК_ОТВЕТОВ

Модуль должен установить ПОЧТОВЫЙ_ЯЩИК_ОТВЕТОВ при условии. АДРЕС_ДЕКОДИРОВАН и адрес соответствует почтовому ящику ответов модуля или ШИРОКОВЕЩАТЕЛЬНЫЙ, как определено в р 896.2. Модуль должен удерживать ПОЧТОВЫЙ_ЯЩИК_ОТВЕТОВ установленным, пока: ИНИЦИАЛИЗАЦИЯ | ПЕРЕДАЧА_ЗАКОНЧЕНА.

ЗАНЯТОСТЬ_ПОЧТОВОГО_ЯЩИКА_ЗАПРОСОВ

Модуль должен установить ЗАНЯТОСТЬ_ПОЧТОВОГО_ЯЩИКА_ЗАПРОСОВ, если почтовый ящик запросов модуля не может принимать сообщения в настоящий момент времени, но будет способен сделать это, если ЗАДАТЧИК повторит попытку. Модуль должен удерживать ЗАНЯТОСТЬ_ПОЧТОВОГО_ЯЩИКА_ЗАПРОСОВ установленным, пока почтовый ящик запросов модуля не сможет принимать сообщения.

ЗАНЯТОСТЬ_ПОЧТОВОГО_ЯЩИКА_ОТВЕТОВ

Модуль должен установить ЗАНЯТОСТЬ_ПОЧТОВОГО_ЯЩИКА_ОТВЕТОВ, если почтовый ящик ответов модуля не может принимать сообщения в настоящий момент времени, но будет способен сделать это, если ЗАДАТЧИК повторит попытку. Модуль должен удерживать ЗАНЯТОСТЬ_ПОЧТОВОГО_ЯЩИКА_ОТВЕТОВ установленным, пока почтовый ящик ответов модуля не сможет принимать сообщения.

ВЫБРАННЫЙ_ПОЧТОВЫЙ_ЯЩИК_ЗАПРОСОВ

Модуль должен установить ВЫБРАННЫЙ_ПОЧТОВЫЙ_ЯЩИК_ЗАПРОСОВ при условии ПОЧТОВЫЙ_ЯЩИК_ЗАПРОСОВ & (НЕБЛОКИРОВАННАЯ_ЗАПИСЬ | ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ) и удержить его установленным, пока: ИНИЦИАЛИЗАЦИЯ | ПЕРЕДАЧА_ЗАКОНЧЕНА.

ВЫБРАННЫЙ_ПОЧТОВЫЙ_ЯЩИК_ОТВЕТОВ

Модуль должен установить ВЫБРАННЫЙ_ПОЧТОВЫЙ_ЯЩИК_ОТВЕТОВ при условии: ПОЧТОВЫЙ_ЯЩИК_ОТВЕТОВ & (НЕБЛОКИРОВАННАЯ_ЗАПИСЬ | ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ) и удержить его установленным, пока: ИНИЦИАЛИЗАЦИЯ | ПЕРЕДАЧА_ЗАКОНЧЕНА.

АТРИБУТ_ШИРОКОВЕЩАТЕЛЬНОЙ_СЕЛЕКЦИИ

Модуль должен установить АТРИБУТ_ШИРОКОВЕЩАТЕЛЬНОЙ_СЕЛЕКЦИИ при условии: ШИРОКОВЕЩАТЕЛЬНЫЙ и AD[5 . . . 0]★ отпущены или разряд регистра СЕЛЕКТИРУЕМАЯ_МАСКА_ПРОХОЖДЕНИЯ_СООБЩЕНИЯ, соответствующий величине AD[5 . . 0]★, установлен. Значение единицы (1) на AD[5 . . 0]★ соответствует LSB+1 регистра. Значение шестьдесят три (63) соответствует старшему значащему разряду регистра. Модуль должен удерживать АТРИБУТ_ШИРОКОВЕЩАТЕЛЬНОЙ_СЕЛЕКЦИИ установленным, пока: ИНИЦИАЛИЗАЦИЯ | ПЕРЕДАЧА_ЗАКОНЧЕНА.

КОНФЛИКТ_ПОЧТОВОГО_ЯЩИКА_ЗАПРОСОВ

Модуль должен установить КОНФЛИКТ_ПОЧТОВОГО_ЯЩИКА_ЗАПРОСОВ, если почтовый ящик запросов модуля не может принять сообщение в течение интервала повтора занятости, но способен сделать это позднее. Модуль должен удерживать КОНФЛИКТ_ПОЧТОВОГО_ЯЩИКА_ЗАПРОСОВ установленным до тех пор, пока почтовый ящик запросов модуля не сможет принимать сообщение.

КОНФЛИКТ_ПОЧТОВОГО_ЯЩИКА_ОТВЕТОВ

Модуль должен установить КОНФЛИКТ_ПОЧТОВОГО_ЯЩИКА_ОТВЕТОВ, если почтовый ящик ответов модуля не может принять сообщение в течение интервала повтора занятости, но способен сделать это позднее. Модуль должен удерживать КОНФЛИКТ_ПОЧТОВОГО_ЯЩИКА_ОТВЕТОВ установленным до тех пор, пока почтовый ящик ответов модуля не сможет принимать сообщение.

ВЫБРАННЫЙ_ПОЧТОВЫЙ_ЯЩИК

Модуль должен установить ВЫБРАННЫЙ_ПОЧТОВЫЙ_ЯЩИК при условии: ВЫБРАННЫЙ_ПОЧТОВЫЙ_ЯЩИК_ЗАПРОСОВ | ВЫБРАННЫЙ_ПОЧТОВЫЙ_ЯЩИК_ОТВЕТОВ и удерживать его установленным, пока: ИНИЦИАЛИЗАЦИЯ | ПЕРЕДАЧА_ЗАКОНЧЕНА.

ЗАНЯТОСТЬ_СООБЩЕНИЯ

Модуль должен установить ЗАНЯТОСТЬ_СООБЩЕНИЯ при условии: ПОЧТОВЫЙ_ЯЩИК_ЗАПРОСОВ & ЗАНЯТОСТЬ_ПОЧТОВОГО_ЯЩИКА_ЗАПРОСОВ | ПОЧТОВЫЙ_ЯЩИК_ОТВЕТОВ & ЗАНЯТОСТЬ_ПОЧТОВОГО_ЯЩИКА_ОТВЕТОВ. Модуль должен удерживать ЗАНЯТОСТЬ_СООБЩЕНИЯ установленным, пока: ИНИЦИАЛИЗАЦИЯ | ПЕРЕДАЧА_ЗАКОНЧЕНА.

КОНФЛИКТ_СООБЩЕНИЯ

Модуль должен установить КОНФЛИКТ_СООБЩЕНИЯ при условии: ИНИЦИАЛИЗАЦИЯ_ПОЧТОВОГО_ЯЩИКА | ПОЧТОВЫЙ_ЯЩИК_ЗАПРОСОВ & КОНФЛИКТ_ПОЧТОВОГО_ЯЩИКА_ЗАПРОСОВ | ПОЧТОВЫЙ_ЯЩИК_ОТВЕТОВ & КОНФЛИКТ_ПОЧТОВОГО_ЯЩИКА_ОТВЕТОВ. Модуль должен удерживать КОНФЛИКТ_СООБЩЕНИЯ установленным, пока: ИНИЦИАЛИЗАЦИЯ | ПЕРЕДАЧА_ЗАКОНЧЕНА.

9.2.2 **Спецификация форматов фрагментов**

9.2.2.1 **Формат фрагментов — уровень сообщений не ФБ+**

Формат фрагментов, не согласующихся со спецификацией уровня сообщений ФБ+, должен быть:

1) старший разряд нулевого байта должен быть установлен в нуль (0), если передача НЕБЛОКИРОВАННАЯ_ЗАПИСЬ будет использоваться для передачи фрагмента. Старший разряд нулевого байта должен быть установлен в единицу (1), если передача ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ используется для передачи фрагмента;

2) поле, состоящее из разрядов с 0 по 6 включ. в байте 0, должно быть установлено в 0 или 64 –127;

3) поле, состоящее из байта 1, должно содержать приоритет сообщения;

4) байты 2 и 3 должны содержать глобальную идентификацию посылающего модуля;

5) остальная часть фрагмента может содержать данные, определенные пользователем.

9.2.2.2 **Формат фрагмента неподтвержденного события**

1) Старший разряд нулевого байта должен быть установлен в нуль (0), если передача НЕБЛОКИРОВАННАЯ_ЗАПИСЬ_ будет использоваться для передачи фрагмента. Старший разряд нулевого байта должен быть установлен в единицу (1), если передача ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ будет использоваться для передачи фрагмента;

2) Поле, состоящее из разрядов с 0 по 6 включ. в байте 0, должно быть установлено в единицу (1).

3) Поле, состоящее из байта 1, должно содержать приоритет сообщения;

4) Байты 2 и 3 должны содержать глобальную идентификацию посылающего модуля;

5) Поле, состоящее из разрядов с 0 по 6 включ. в байте 4, должно содержать единицу (1). Разряд 7 байта 4 должен быть установлен для указания того, что событие запроса внимания будет посылаться принимающему модулю.

6) Байт 5 должен быть установлен в нуль (0).

7) Поле, состоящее из байтов 6 и 7, должно быть равно сумме всех байтов в фрагменте. Это число должно включать все байты в заголовке и теле фрагмента.

8) Остальная часть фрагмента может содержать данные, определенные пользователем.

Этот фрагмент должен посылаться только в почтовые ящики ответов.

9.2.2.3 **Формат фрагмента подтвержденного события**

1) Старший разряд нулевого байта должен быть установлен в нуль (0), если передача НЕБЛОКИРОВАННАЯ_ЗАПИСЬ будет использоваться для передачи фрагмента. Старший разряд нулевого байта должен быть установлен в единицу (1), если передача ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ будет использоваться для передачи фрагмента;

2) Поле, состоящее из разрядов с 0 по 6 включ. в байте 0, должно быть установлено в единицу (1).

3) Поле, состоящее из байта 1, должно содержать приоритет сообщения;

4) Байты 2 и 3 должны содержать глобальную идентификацию посылающего модуля;

5) Поле, состоящее из разрядов с 0 по 6 включ. в байте 4, должно содержать двойку (2). Разряд 7 байта 4 должен быть установлен для указания того, что событие запроса внимания будет посылаться принимающему модулю.

6) Байт 5 должен содержать уникальный идентификационный номер. Посылающий модуль должен использовать различные номера для каждого уникального сообщения, которые он посылает в один и тот же принимающий модуль.

7) Поле, состоящее из байтов 6 и 7, должно быть равно сумме всех байтов в фрагменте. Это число должно включать все байты в заголовке и теле фрагмента.

8) Остальная часть фрагмента может содержать данные, определенные пользователем.

Этот фрагмент должен посылаться только в почтовые ящики запросов.

9.2.2.4 **Формат фрагмента положительного подтверждения события**

1) Старший разряд нулевого байта должен быть установлен в нуль (0), если передача НЕБЛОКИРОВАННАЯ_ЗАПИСЬ будет использоваться для передачи фрагмента. Старший разряд нулевого байта должен быть установлен в единицу (1), если передача ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ будет использоваться для передачи фрагмента.

2) Поле, состоящее из разрядов с 0 по 6 включ. в байте 0, должно быть установлено в единицу (1).

3) Поле, состоящее из байта 1, должно содержать приоритет сообщения.

4) Байты 2 и 3 должны содержать глобальную идентификацию посылающего модуля.

5) Поле, состоящее из разрядов с 0 по 6 включ. в байте 4, должно содержать тройку (3). Разряд 7 байта 4 должен быть установлен для указания того, что событие запроса внимания будет посылаться принимающему модулю.

6) Байт 5 должен содержать идентификацию сообщения принятую в байте 5 фрагмента подтвержденного события, к которому этот ответ относится.

7) Поле, состоящее из байтов 6 и 7, должно быть равно сумме всех байтов в фрагменте. Это число должно включать все байты в заголовке и теле фрагмента.

Этот фрагмент должен посылаться только в почтовые ящики ответов.

9.2.2.5 **Формат фрагмента отрицательного подтверждения события**

1) Старший разряд нулевого байта должен быть установлен в нуль (0), если передача НЕБЛОКИРОВАННАЯ_ЗАПИСЬ будет использоваться для передачи фрагмента. Старший разряд нулевого байта должен быть установлен в единицу (1), если передача ЗАПИСЬ БЕЗ_ПОДТВЕРЖДЕНИЯ будет использоваться для передачи фрагмента.

2) Поле, состоящее из разрядов с 0 по 6 включ. в байте 0, должно быть установлено в единицу (1).

3) Поле, состоящее из байта 1, должно содержать приоритет сообщения.

4) Байты 2 и 3 должны содержать глобальную идентификацию посылающего модуля.

5) Поле, состоящее из разрядов с 0 по 6 включ. в байте 4, должно содержать четверку (4). Разряд 7 байта 4 должен быть установлен для указания того, что событие запроса внимания будет посылаться принимающему модулю.

6) Байт 5 должен содержать идентификацию сообщения принятую в байте 5 фрагмента подтвержденного события, к которому этот ответ относится.

7) Поле, состоящее из байтов 6 и 7, должно быть равно сумме всех байтов в фрагменте. Это число должно включать все байты в заголовке и теле фрагмента.

8) Поле, состоящее из байтов 8—11, должно содержать величину соответствующего типа исключения, как описано в 9.2.6.

9) Остальные байты фрагмента могут быть использованы для дополнительной информации об ошибке.

Этот фрагмент должен посылаться только в почтовые ящики ответов.

9.2.2.6 **Формат фрагмента запроса многофрагментного сообщения**

1) Старший разряд нулевого байта должен быть установлен в нуль (0), если передача НЕБЛОКИРОВАННАЯ_ЗАПИСЬ будет использоваться для передачи фрагмента. Старший разряд нулевого байта должен быть установлен в единицу (1), если передача ЗАПИСЬ_БЕЗ_ ПОДТВЕРЖДЕНИЯ будет использоваться для передачи фрагмента.

2) Поле, состоящее из разрядов с 0 по 6 включ. в байте 0, должно быть установлено в единицу (1).

3) Поле, состоящее из байта 1, должно содержать приоритет сообщения.

4) Байты 2 и 3 должны содержать глобальную идентификацию посылающего модуля.

5) Поле, состоящее из разрядов с 0 по 6 включ. в байте 4, должно содержать пятерку (5). Разряд 7 байта 4 должен быть установлен для указания того, что событие запроса внимания будет посылаться принимающему модулю.

6) Байт 5 должен содержать уникальный идентификационный номер. Посылающий модуль должен использовать различные номера для каждого уникального сообщения, которые он посылает в один и тот же принимающий модуль.

7) Поле, состоящее из байтов 6 и 7, должно содержать двенадцать (12), сумму байтов в фрагменте. Это число должно включать все байты в заголовке и теле фрагмента.

8) Поле, состоящее из байтов 8—11 включ., должно содержать размер сообщения, как описано в 9.2.3.

Этот фрагмент должен посылаться только в почтовые ящики запросов.

9.2.2.7 **Формат фрагмента положительного ответа многофрагментного сообщения**

1) Старший разряд нулевого байта должен быть установлен в нуль (0), если передача НЕБЛОКИРОВАННАЯ_ЗАПИСЬ будет использоваться для передачи фрагмента. Старший разряд нулевого байта должен быть установлен в единицу (1), если передача ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ будет использоваться для передачи фрагмента.

2) Поле, состоящее из разрядов с 0 по 6 включ. в байте 0, должно быть установлено в единицу (1).

3) Поле, состоящее из байта 1, должно содержать приоритет сообщения.

4) Байты 2 и 3 должны содержать глобальную идентификацию посылающего модуля.

5) Поле, состоящее из разрядов с 0 по 6 включ. в байте 4, должно содержать шестерку (6). Разряд 7 байта 4 должен быть установлен для указания того, что событие запроса внимания будет посылаться принимающему модулю.

6) Байт 5 должен содержать идентификацию сообщения, принятую в байте 5 фрагмента запроса многофрагментного сообщения, к которому этот ответ относится.

7) Поле, состоящее из байтов 6 и 7, должно содержать сумму байтов в фрагменте. Это число должно включать все байты в заголовке и теле фрагмента.

8) Поле, состоящее из байтов с 8 по 11 включ., должно содержать величину максимального интервала между фрагментами, как описано в 9.2.4.

Этот фрагмент должен посылаться только в почтовые ящики ответов.

9.2.2.8 **Формат фрагмента отрицательного ответа многофрагментного сообщения**

1) Старший разряд нулевого байта должен быть установлен в нуль (0), если передача НЕБЛОКИРОВАННАЯ ЗАПИСЬ будет использоваться для передачи фрагмента. Старший разряд бит нулевого байта должен быть установлен в единицу (1), если передача ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ будет использоваться для передачи фрагмента.

2) Поле, состоящее из разрядов с 0 по 6 включ. в байте 0, должно быть установлено в единицу (1).

3) Поле, состоящее из байта 1, должно содержать приоритет сообщения.

4) Байты 2 и 3 должны содержать глобальную идентификацию посылающего модуля.

5) Поле, состоящее из разрядов с 0 по 6 включ. в байте 4, должно содержать семерку (7). Разряд 7 байта 4 должен быть установлен для указания того, что событие запроса внимания будет посылаться принимающему модулю.

6) Байт 5 должен содержать идентификацию сообщения принятую в байте 5 фрагмента запроса многофрагментного сообщения, к которому этот ответ относится.

7) Поле, состоящее из байтов 6 и 7, должно содержать сумму байтов в фрагменте. Это число должно включать все байты в заголовке и теле фрагмента.

8) Поле, состоящее из байтов с 8 по 11 включ., должно содержать величину типа исключения, как описано в 9.2.6.

9) Оставшиеся байты в фрагменте могут использоваться для дополнительной информации об ошибке.

Этот фрагмент должен посылаться только в почтовые ящики ответов.

9.2.2.9 **Формат фрагмента упорядоченных данных многофрагментного сообщения**

1) Старший разряд нулевого байта должен быть установлен в нуль (0), если передача НЕБЛОКИРОВАННАЯ ЗАПИСЬ будет использоваться для передачи фрагмента. Старший разряд нулевого байта должен быть установлен в единицу (1), если передача ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ будет использоваться для передачи фрагмента.

2) Поле, состоящее из разрядов с 0 по 6 включ. в байте 0, должно быть установлено в единицу (1).

3) Поле, состоящее из байта 1, должно содержать приоритет сообщения.

4) Байты 2 и 3 должны содержать глобальную идентификацию посылающего модуля.

5) Поле, состоящее из разрядов с 0 по 6 включ. в байте 4, должно содержать восьмерку (8). Разряд 7 байта 4 должен быть установлен для указания того, что событие запроса внимания будет посылаться принимающему модулю.

6) Байт 5 должен содержать идентификацию сообщения принятую в байте 5 фрагмента запроса многофрагментного сообщения, к которому этот фрагмент относится.

7) Поле, состоящее из байтов 6 и 7, должно содержать сумму байтов в фрагменте. Это число должно включать все байты в заголовке и теле фрагмента.

8) Оставшаяся часть фрагмента содержит часть сообщения.

Этот фрагмент должен посылаться только в почтовые ящики запросов.

9.2.2.10 **Формат фрагмента последовательных данных многофрагментного сообщения**

1) Старший разряд нулевого байта должен быть установлен в нуль (0), если передача НЕБЛОКИРОВАННАЯ_ЗАПИСЬ будет использоваться для передачи фрагмента. Старший разряд нулевого байта должен быть установлен в единицу (1), если передача ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ будет использоваться для передачи фрагмента.

2) Поле, состоящее из разрядов с 0 по 6 включ. в байте 0, должно быть установлено в единицу (1).

3) Поле, состоящее из байта 1, должно содержать приоритет сообщения.

4) Байты 2 и 3 должны содержать глобальную идентификацию посылающего модуля.

5) Поле, состоящее из разрядов с 0 по 6 включ. в байте 4, должно содержать девятку (9). Разряд 7 байта 4 должен быть установлен для указания того, что событие запроса внимания будет посылаться принимающему модулю.

6) Байт 5 должен содержать идентификацию сообщения, принятую в байте 5 фрагмента запроса многофрагментного сообщения, к которому этот фрагмент относится.

7) Поле, состоящее из байтов 6 и 7, должно содержать сумму байтов в фрагменте. Это число должно включать все байты в заголовке и теле фрагмента.

8) Поле, состоящее из байтов 8—11, должно содержать монотонно увеличивающийся номер последовательности. Если это первый фрагмент последовательных данных, то номер последовательности должен быть равен единице (1).

9) Оставшаяся часть фрагмента может содержать часть сообщения.

Этот фрагмент должен посылаться только в почтовые ящики запросов.

9.2.2.11 **Формат последнего упорядоченного фрагмента многофрагментного сообщения**

1) Старший разряд нулевого байта должен быть установлен в нуль (0), если передача НЕБЛОКИРОВАННАЯ_ЗАПИСЬ будет использоваться для передачи фрагмента. Старший разряд нулевого байта должен быть установлен в единицу (1), если передача ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ будет использоваться для передачи фрагмента.

2) Поле, состоящее из разрядов с 0 по 6 включ. в байте 0, должно быть установлено в единицу (1).

3) Поле, состоящее из байта 1, должно содержать приоритет сообщения.

4) Байты 2 и 3 должны содержать глобальную идентификацию посылающего модуля.

5) Поле, состоящее из разрядов с 0 по 6 включ. в байте 4, должно содержать десятку (10). Разряд 7 байта 4 должен быть установлен для указания того, что событие запроса внимания будет посылаться принимающему модулю.

6) Байт 5 должен содержать идентификацию сообщения, принятую в байте 5 фрагмента запроса многофрагментного сообщения, к которому этот фрагмент относится.

7) Поле, состоящее из байтов 6 и 7, должно содержать сумму байтов в фрагменте. Это число должно включать все байты в заголовке и теле фрагмента.

8) Оставшаяся часть фрагмента может содержать заключительную часть сообщения.

Этот фрагмент должен посылаться только в почтовые ящики запросов.

9.2.2.12 **Формат последнего последовательного фрагмента многофрагментного сообщения**

1) Старший разряд нулевого байта должен быть установлен в нуль (0), если передача НЕБЛОКИРОВАННАЯ_ЗАПИСЬ будет использоваться для передачи фрагмента. Старший разряд нулевого байта должен быть установлен в единицу (1), если передача ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ будет использоваться для передачи фрагмента.

2) Поле, состоящее из разрядов с 0 по 6 включ. в байте 0, должно быть установлено в единицу (1).

3) Поле, состоящее из байта 1, должно содержать приоритет сообщения.

4) Байты 2 и 3 должны содержать глобальную идентификацию посылающего модуля.

5) Поле, состоящее из разрядов с 0 по 6 включ. в байте 4, должно содержать одиннадцать (11). Разряд 7 байта 4 должен быть установлен для указания того, что событие запроса внимания будет посылаться принимающему модулю.

6) Байт 5 должен содержать идентификацию сообщения, принятую в байте 5 фрагмента запроса многофрагментного сообщения, к которому этот фрагмент относится.

7) Поле, состоящее из байтов 6 и 7, должно содержать сумму байтов в фрагменте. Это число должно включать все байты в заголовке и теле фрагмента.

8) Поле, состоящее из байтов с 8 по 11 включ., должно содержать последовательный инкрементированный номер последнего фрагмента последовательных данных.

9) Оставшаяся часть фрагмента может содержать заключительную часть сообщения.

Этот фрагмент должен посылаться только в почтовые ящики запросов.

9.2.2.13 **Формат фрагмента положительного подтверждения многофрагментного сообщения**

1) Старший разряд нулевого байта должен быть установлен в нуль (0), если передача НЕБЛОКИРОВАННАЯ_ЗАПИСЬ будет использоваться для передачи фрагмента. Старший разряд нулевого байта должен быть установлен в единицу (1), если передача ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ будет использоваться для передачи фрагмента.

2) Поле, состоящее из разрядов с 0 по 6 включ. в байте 0, должно быть установлено в единицу (1).

3) Поле, состоящее из байта 1, должно содержать приоритет сообщения.

4) Байты 2 и 3 должны содержать глобальную идентификацию посылающего модуля.

5) Поле, состоящее из разрядов с 0 по 6 включ. в байте 4, должно содержать двенадцать (12). Разряд 7 байта 4 должен быть установлен для указания того, что событие запроса внимания будет посылаться принимающему модулю.

6) Байт 5 должен содержать идентификацию сообщения, принятую в байте 5 фрагмента запроса многофрагментного сообщения, к которому этот ответ относится.

7) Поле, состоящее из байтов 6 и 7, должно содержать восемь (8), сумму байтов в фрагменте. Это число должно включать все байты в заголовке и теле фрагмента.

Этот фрагмент должен посылаться только в почтовые ящики ответов.

9.2.2.14 **Формат фрагмента отрицательного подтверждения многофрагментного сообщения**

1) Старший разряд нулевого байта должен быть установлен в нуль (0), если передача НЕБЛОКИРОВАННАЯ_ЗАПИСЬ будет использоваться для передачи фрагмента. Старший разряд нулевого байта должен быть установлен в единицу (1), если передача ЗАПИСЬ_БЕЗ_ПОДТВЕРЖДЕНИЯ будет использоваться для передачи фрагмента.

2) Поле, состоящее из разрядов с 0 по 6 включ. в байте 0, должно быть установлено в единицу (1).

3) Поле, состоящее из байта 1, должно содержать приоритет сообщения.

4) Байты 2 и 3 должны содержать глобальную идентификацию посылающего модуля.

5) Поле, состоящее из разрядов с 0 по 6 включ. в байте 4, должно содержать тринадцать (13). Разряд 7 байта 4 должен быть установлен для указания того, что событие запроса внимания будет посылаться принимающему модулю.

6) Байт 5 должен содержать идентификацию сообщения, принятую в байте 5 фрагмента запроса многофрагментного сообщения, к которому этот ответ относится.

7) Поле, состоящее из байтов 6 и 7, должно содержать сумму байтов в фрагменте. Это число должно включать все байты в заголовке и теле фрагмента.

8) Поле, состоящее из байтов с 8 по 11 включ., должно содержать величину соответствующего типа исключения, как описано в 9.2.6.

9) Остальная часть фрагмента может быть использована для дополнительной информации об ошибке.

Этот фрагмент должен посылаться только в почтовые ящики ответов.

9.2.3 **Размер сообщения**

Это общее количество байтов, содержащееся в фрагментах многофрагментной серии (упорядоченные фрагменты или последовательные фрагменты).

9.2.4 **Интервал между фрагментами**

Это положительное число, измеряемое в единицах 2^{-31} с. Он показывает минимальную задержку между посылкой любых двух фрагментов в многофрагментном сообщении.

9.2.5 **Номер последовательности**

Зарезервирован для использования в последовательных многофрагментных сообщениях. Номер последовательности для каждого успешного фрагмента в серии будет увеличиваться на единицу (1).

9.2.6 **Поле типа исключения**

Следующие значения должны использоваться для параметров типа исключения для фрагмента отрицательного ответа:

Значение исключения	Описание исключения
1	Недопустимый протокол сообщения
2	Недопустимый размер фрагмента
3	Недопустимая идентификация сообщения (в использовании)
4	Недопустимая глобальная идентификация
5	Не поддерживает тип фрагмента последовательных данных многофрагментного сообщения
6	Тайм-аут
7 63	Резерв
остальные	Определяются пользователем

9.2.7 **Описание протоколов**

9.2.7.1 **Протокол задатчика для неподтвержденного однофрагментного сообщения**

Чтобы передать неподтвержденное однофрагментное сообщение, модуль должен.

1) сформатировать неподтвержденный фрагмент события, как определено в 9.2.2.2,

2) передать фрагмент, используя команду записи, определяемую старшим разрядом первого байта.

9 2 7.2 **Протокол исполнителя для неподтвержденного однофрагментного сообщения**

При получении однофрагментного сообщения без подтверждения модуль должен передать этот фрагмент на его уровень приложении.

9.2.7.3 **Протокол задатчика для подтвержденного однофрагментного сообщения**

Чтобы передать подтвержденное однофрагментное сообщение, модуль должен:

1) сформатировать подтвержденный фрагмент события, как определено в 9.2.2.3;

2) передать фрагмент, используя команду записи, определяемую старшим разрядом первого байта;

3) когда он принимает фрагмент положительного подтверждения события или фрагмент отрицательного подтверждения события или, если произошел тайм-аут, то известить свой уровень приложения

9.2.7.4 **Протокол исполнителя для подтвержденного однофрагментного сообщения**

При получении простого подтвержденного фрагмента события модуль должен:

1) если фрагмент принят успешно, то сформатировать фрагмент положительного подтверждения события, как определено в 9.2.2.4;

2) если фрагмент принят с ошибкой, то сформатировать фрагмент отрицательного подтверждения события, как определено в 9.2.2.5;

3) затем передать этот фрагмент запросчику.

9.2.7.5 **Протокол задатчика для упорядоченного многофрагментного сообщения**

Чтобы передать упорядоченное многофрагментное сообщение модуль должен.

1) сформатировать фрагмент запроса многофрагментного сообщения, как определено в 9.2.2.6;

2) передать фрагмент, используя команду записи, определяемую старшим рязрядом первого байта;

3) если он принимает фрагмент отрицательного ответа многофрагментного сообщения, или если происходит тайм-аут, то известить свой уровень приложения и прервать последовательность;

4) если он принимает фрагмент положительного ответа многофрагментного сообщения, то сформатировать фрагмент упорядоченных данных многофрагментного сообщения, как определено в 9.2.2.9;

5) затем передать этот фрагмент ответчику после того, как истечет межфрагментный интервал;

6) повторять шаги (4—5) до тех пор, пока не перешлет все данные, за исключением тех, которые умещаются в один фрагмент;

7) затем сформатировать последний фрагмент упорядоченных данных многофрагментного сообщения, как определено в 9.2.2.11;

8) затем передать этот фрагмент ответчику после того, как истечет межфрагментный интервал,

9) если он принимает фрагмент отрицательного подтверждения многофрагментного сообщения или если происходит тайм-аут, то известить свой уровень приложения и прервать последовательность;

10) если он принимает фрагмент положительного подтверждения многофрагментного сообщения, то известить свой уровень приложения об успешном завершении.

9.2.7.6 **Протокол задатчика для последовательного многофрагментного сообщения**

Чтобы передать последовательное многофрагментное сообщение модуль должен:

1) сформатировать фрагмент запроса многофрагментного сообщения, как определено в 9.2.2.6;

2) передать фрагмент, используя команду записи, определяемую старшим разрядом первого байта;

3) если он принимает фрагмент отрицательного ответа многофрагментного сообщения или если происходит тайм-аут, то известить свой уровень приложения и прервать последовательность;

4) если он принимает фрагмент положительного ответа многофрагментного сообщения, то сформатировать фрагмент последовательных данных многофрагментного сообщения, как определено в 9.2.2.10;

5) затем передать этот фрагмент ответчику после того, как истечет межфрагментный интервал;

6) повторять шаги (4—5) до тех пор, пока не перешлет все данные, за исключением тех, которые умещаются в один фрагмент;

7) затем сформатировать последний фрагмент последовательных данных многофрагментного сообщения, как определено в 9.2.2.12;

8) затем передать этот фрагмент ответчику после того, как истечет межфрагментный интервал;

9) если он принимает фрагмент отрицательного подтверждения многофрагментного сообщения или если происходит тайм-аут, то известить свой уровень приложения и прервать последовательность,

10) если он принимает фрагмент положительного подтверждения многофрагментного сообщения, то известить свой уровень приложения об успешном завершении.

9.2.7.7 **Протокол исполнителя для многофрагментного сообщения**

При получении фрагмента запроса многофрагментного сообщения модуль должен:

1) если фрагмент принят успешно, то сформатировать фрагмент положительного ответа многофрагментного сообщения, как определено в 9.2.2.13;

2) если фрагмент принят с ошибкой, то сформатировать фрагмент отрицательного ответа многофрагментного сообщения, как определено в 9.2.2.14;

3) затем передать этот фрагмент запросчику;

4).если сообщение является последовательным многофрагментным сообщением, то затем принять фрагменты последовательных данных многофрагментного сообщения и последний последовательный фрагмент от запросчика и перейти к пункту 6;

5) если сообщение является упорядоченным многофрагментным сообщением, то затем принять фрагменты упорядоченных данных многофрагментного сообщения и последний упорядоченный фрагмент от запросчика;

6) если он не корректно принял сообщение, то сформатировать фрагмент отрицательного подтверждения многофрагментного сообщения, как определено в 9.2.2.8, и передать его запросчику;

7) если он корректно принял сообщение, то сформатировать фрагмент положительного подтверждения многофрагментного сообщения, как определено в 9.2.2.7, и передать его запросчику.

УДК 681.327:006.354 ОКС 35.200 П85 ОКСТУ 0034

Ключевые слова: интерфейс, Фьючебас+, протокол, логический уровень, арбитраж, магистраль, передача данных.

Редактор *В. П. Огурцов*
Технический редактор *В. Н. Прусакова*
Корректор *Л. Я. Митрофанова*
Компьютерная верстка *А. Г. Хоменко*

Сдано в набор 01.02.96. Подписано в печать 21.03.96. Усл. печ. л. 21,39. Усл. кр.-отт. 21,63.
Уч.-изд. л. 21,20. Тираж 270 экз. С 3199. Зак. 154.

ИПК Издательство стандартов, 107076, Москва, Колодезный пер. 14.
ЛР № 021007 от 10.08.95
Набрано в Калужской типографии стандартов на ПЭВМ.
Калужская типография стандартов, ул. Московская, 256.
ПЛР № 040138