МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

А.О. Ключев, Д.Р. Ковязина,
Е.В. Петров, А.Е. Платунов

ИНТЕРФЕЙСЫ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ

Учебное пособие

Санкт-Петербург

2010

Ключев А.О., Ковязина Д.Р., Петров Е.В., Платунов А.Е. Интерфейсы периферийных устройств. – СПб.: СПбГУ ИТМО, 2010. – 290 с.

Учебное пособие посвящено вопросам организации интерфейсов периферийных устройств. В пособии рассматриваются: организация системы ввода-вывода, способы обмена информацией между устройствами вычислительной системы, аппаратные интерфейсы вычислительных систем. В приложениях к учебному пособию помещены справочные данные и задания к выполнению лабораторных работ с использованием учебного лабораторного стенда SDK-1.1.

Для подготовки бакалавров и магистров по направлению 23.01.00 «Информатика и вычислительная техника»; по программам подготовки магистров 23.01.00.33 «Проектирование встроенных вычислительных систем» и 23.01.00.34 «Системотехника интегральных вычислителей. Системы на кристалле».

Рекомендовано к печати ученым советом факультета КТиУ, протокол №15 от 16.11.2010 г.

В 2009 году Университет стал победителем многоэтапного конкурса, в результате которого определены 12 ведущих университетов России, которым присвоена категория «Национальный исследовательский университет». Министерством образования и науки Российской Федерации была утверждена Программа развития государственного образовательного учреждения высшего профессионального образования «Санкт-Петербургский государственный университет информационных технологий, механики и оптики» на 2009–2018 годы.

Оглавление

Введение

При изучении систем ввода-вывода и интерфейсов периферийных устройств необходимо представлять основные принципы построения средств вычислительной техники, которые во многом определяются той элементной базой, на которой строятся вычислительные системы общего и специализированного назначения. С этой точки зрения весь период развития вычислительной техники от первого компьютера до современных вычислительных машин можно разбить на два этапа. Первый – это этап до появления современных интегральных схем и микропроцессоров. Второй – после их появления и начала выпуска персональных компьютеров (ПК).

На первом этапе компьютеры разрабатывались и изготавливались на своей собственной элементной базе, их устройства (процессор, ОЗУ, устройства управления и т.п.) имели архитектуру и структуру, присущую только данному компьютеру или одному семейству, связь между устройствами и узлами осуществлялась с помощью интерфейсов, используемых только этим типом вычислительной машины. Стандарты применялись, но в основном касались ПУ, а не внутренних устройств. Поэтому машины разных фирм были не совместимы по элементной базе, устройствам и конструктивам. Это относилось как к большим, так и к малым вычислительным машинам. Это машины типа БЭСМ-1 (2,6), «Урал», «Наири», IBM 360 (370) , ЕС ЭВМ-1033 (1040, 1060), DEC PDP-11, СМ-1 (2,3,4) и т.д. Каждый тип компьютера был в определенной степени уникален [96].

На втором этапе изменился принцип построения вычислительной техники. Она стала основываться на правиле трех «М»: модульность, микропрограммируемость и магистральность. Модуль представляет собой функционально полное и конструктивно законченное устройство, серийно выпускаемое и программно (микропрограммно) управляемое (настраиваемое). Вычислительные системы собираются на основе этих модулей с помощью каналов связи – универсальных или специализированных интерфейсов [96].

На втором этапе существенно возросла роль стандартизации, без которой разработка и выпуск вычислительных систем общего назначения стал просто невозможен, а специализированного назначения – практически невозможен.

В последнее время существенно возросли роль и значение систем ввода-вывода и интерфейсов вследствие усиления значимости коммуникационной составляющей по сравнению с вычислительной составляющей в информационно-управляющих системах.

Учебное пособие посвящено рассмотрению роли и места систем ввода-вывода и интерфейсов в вычислительных системах, изложению принципов их построения и функционирования. Оно содержит материалы о современных стандартных внутрисистемных, системных, периферийных интерфейсах, а также лабораторные работы, способствующие изучению этих материалов, приобретению студентами навыков проектирования и низкоуровневого

программирования подсистем ввода-вывода микропроцессорных систем различного назначения на примере программирования учебного лабораторного стенда SDK-1.1.

Учебное пособие предназначено для студентов, специализирующихся в области вычислительной техники и программирования. Знание интерфейсов позволит разработчику аппаратуры более грамотно подойти к выбору варианта, соответствующего поставленной задаче. Знания материалов этого пособия необходимы системным интеграторам. Без этих знаний они не смогут подобрать требуемый набор устройств и оптимально объединить их в систему. Сведения учебного пособия помогут системным программистам разрабатывать собственные драйверы периферийных устройств или адаптировать чужие разработки.

1 Организация системы ввода-вывода вычислительной системы

1.1 Основы архитектурной организации вычислительной системы

В основе архитектуры современных вычислительных машин (ВМ) лежит представление алгоритма решения задачи в виде программы последовательных вычислений. Согласно стандарту ISO 2382/1-84, программа для ВМ – это упорядоченная последовательность команд, подлежащая обработке.

Вычислительная машина, где определенным образом закодированные команды программы хранятся в памяти, известна под названием ВМ с хранимой в памяти программой. Идея принадлежит создателям вычислителя ENIAC Эккерту, Мочли и фон-Нейману. Еще до завершения работ над ENIAC они приступили к новому проекту – EDVAC, главной особенностью которого стала концепция хранимой в памяти программы, на долгие годы определившая базовые принципы построения последующих поколений вычислительных машин. Относительно авторства существует несколько версий, но поскольку в законченном виде идея впервые была изложена в 1945 году в статье Джона фон-Неймана (John von Neumann «First Draft of a Report on the EDVAC»), именно его фамилия фигурирует в обозначении архитектуры подобных машин, составляющих подавляющую часть современного парка ВМ и вычислительных систем (ВС).

Типичная фон-Неймановская вычислительная машина содержит: память, устройство управления, арифметико-логическое устройство и устройства ввода и вывода. На рис. ниже представлена модифицированная структура фон-Неймановской ВМ, так как первоначально память не разделялась на основную и вторичную (внешнюю) и не было портов ввода-вывода.

В любой ВМ имеются средства для ввода программ и данных к ним. Информация поступает из подсоединенных к ЭВМ периферийных устройств (ПУ) ввода. Результаты вычислений выводятся на периферийные устройства вывода. Связь и взаимодействие ВМ и ПУ обеспечивают порты ввода и порты вывода. Термином порт обозначают аппаратуру сопряжения периферийного устройства с ВМ и управления им.

Введенная информация сначала запоминается в основной памяти, а затем переносится во вторичную память, для длительного хранения. Чтобы программа могла выполняться, команды и данные должны располагаться в основной памяти (ОП), организованной таким образом, что каждое двоичное слово хранится в отдельной ячейке, идентифицируемой адресом, причем соседние ячейки памяти имеют следующие по порядку адреса.

Рис. 1. Структура фон-Неймановской вычислительной машины

Устройство управления (УУ) – важнейшая часть ВМ, организующая автоматическое выполнение программ (путем реализации функций управления) и обеспечивающая функционирование ВМ как единой системы. Для пояснения функций УУ ВМ следует рассматривать как совокупность элементов, между которыми происходит пересылка информации, в ходе которой эта информация может подвергаться определенным видам обработки. Основной функцией УУ является формирование управляющих сигналов, отвечающих за извлечение команд из памяти в порядке, определяемом программой, и последующее исполнение этих команд.

Еще одной неотъемлемой частью ВМ является арифметико-логическое устройство (АЛУ). АЛУ (или операционное устройство, ОУ) обеспечивает арифметическую и логическую обработку двух входных переменных, в результате которой формируется выходная переменная.

УУ и АЛУ (ОУ) тесно взаимосвязаны и их обычно рассматривают как единое устройство, известное как центральный процессор (ЦП) или просто процессор.

Фон-Нейманом были сформулированы так называемые основные принципы построения ЭВМ.

1.1.1 Принципы Фон-Неймановской архитектуры

1. Принцип использования двоичной системы счисления для представления данных и команд.
2. Принцип хранимой программы.
3. Однородность памяти и адресации; принцип линейности памяти.

Память машины рассматривается как вектор, состоящий из одинаковых ячеек, способных принимать любые значения. Значение в ячейке с точки зрения процессора является последовательностью битов фиксированной длины без каких-либо ограничений. Ячейки памяти идентифицируются адресами:

числами от нуля до максимально возможной для данной машины величины. Адреса служат указателями для процессора, откуда следует извлекать значение или куда помещать значение.

Отсутствие различий между командами и данными означает, что по двоичному коду, взятому из некоторой ячейки памяти невозможно определить, представляет ли он машинную команду или данные некоторого типа. В связи с этим, тип слова (команда или данные) определяется не способом кодирования, а способом использования (интерпретацией). Из однородности памяти следует, что команды и данные располагаются в единой общей памяти и одинаково адресуются. Принцип использования единой памяти принято называть Принстонской архитектурой по наименованию института, в котором она была разработана. В отличие от этого принципа при построении некоторых ЭВМ используется раздельная память для программ и данных. Архитектура с разделением памяти получила название Гарвардской архитектуры ЭВМ.

4. Принцип последовательного программного управления.

Программа состоит из набора команд, которые выполняются процессором друг за другом в определенной последовательности (в порядке возрастания их адресов). Естественная последовательность выполнения команд может нарушаться при выполнении команд переходов (условных и безусловных), циклов, вызовов, возвратов и т.п. Кроме того, последовательность команд может нарушаться особыми случаями, приводящими к прерыванию программы.

Управляющее устройство (может называться счетчиком команд) содержит адрес команды, назначаемой для выполнения процессором.

5. Пассивность памяти и активность процессора.

Ячейка памяти всегда содержит какое-то значение. Полученное ячейкой значение не может быть изменено иначе как при выполнении специальной команды процессора, предназначенной для этого действия. Процессор всегда выполняет некоторую команду, закодированную последовательностью битов в ячейке и извлеченную из памяти.

1.1.2 Достоинства и недостатки Принстонской архитектуры

Основное преимущество Принстонской архитектуры – упрощение устройства ВМ, так как реализуется обращение только к одной общей памяти. Кроме того, использование единой области памяти позволяло оперативно перераспределять ресурсы между областями программ и данных, что существенно повышало гибкость вычислительной системы с точки зрения разработчика программного обеспечения. Размещение стека в общей памяти облегчало доступ к его содержимому. В разные моменты времени одна и та же область памяти может использоваться и как память программ и как память данных. Для того, чтобы программа могла работать в произвольной области памяти, ее необходимо модернизировать перед загрузкой, т.е. работать с нею как

с обычными данными. Эта особенность архитектуры позволяет наиболее гибко управлять работой ВС, но создает принципиальную возможность искажения управляющей программы, что понижает надежность работы аппаратуры.

Экономичность неймановских (Принстонских) ВМ определяется минимальностью затрат оборудования на реализацию вычислительного процесса по сравнению с ВМ с ненеймановской архитектурой.

Неслучайно поэтому Принстонская (фон-Неймановская) архитектура стала основной архитектурой универсальных компьютеров, включая персональные компьютеры.

1.1.3 Достоинства и недостатки Гарвардской архитектуры

Основной особенностью Гарвардской архитектуры является использование раздельных адресных пространств для хранения команд и данных.

Первым компьютером, в котором была использована идея гарвардской архитектуры, был Марк I. Разработан и построен в 1941 году по контракту с IBM молодым гарвардским математиком Говардом Эйкеном и ещё четырьмя инженерами этой компании на основе идей англичанина Чарльза Бэббиджа. Для хранения инструкций в Марк I использовалась перфорированная лента, а для работы с данными – электромеханические регистры. И такое разделение объяснялось, прежде всего, существующими тогда технологиями производства памяти: память для хранения данных с возможностью их перезаписи была малой емкости и дорого стоила, поэтому ее невозможно было использовать для хранения программ. В некотором смысле так было проще (смотрите историю вычислительной техники).

Гарвардская архитектура почти не использовалась до конца 70-х годов, пока производители микроконтроллеров (МК) не поняли, что она дает определенные преимущества разработчикам автономных систем управления.

Дело в том, что, судя по опыту использования микропроцессорных систем (МПС) для управления различными объектами, для реализации большинства алгоритмов управления такие преимущества Принстонской архитектуры как гибкость и универсальность не имеют большого значения. Анализ реальных программ управления показал, что необходимый объем памяти данных МК, используемый для хранения промежуточных результатов, как правило, на порядок меньше требуемого объема памяти программ. В этих условиях использование единого адресного пространства приводило к увеличению формата команд за счет увеличения числа разрядов для адресации операндов. Применение отдельной небольшой по объему памяти данных способствовало сокращению длины команд и ускорению поиска информации в памяти данных.

Кроме того, Гарвардская архитектура обеспечивает потенциально более высокую скорость выполнения программы по сравнению с Принстонской за счет возможности реализации параллельных операций. Выборка следующей команды может происходить одновременно с выполнением предыдущей, и нет

необходимости останавливать процессор на время выборки команды. Этот метод реализации операций позволяет обеспечивать выполнение различных команд за одинаковое число тактов, что дает возможность более просто определить время выполнения циклов и критичных участков программы. Однако такая организация сложнее по сравнению с Принстонской, так как заставляет процессор обслуживать обмен по двум шинам одновременно.

Кроме того, такая схема реализации доступа к памяти имеет один очевидный недостаток – высокую стоимость. При разделении каналов передачи адреса и данных на кристалле процессора последний должен иметь почти в два раза больше выводов (так как шины адреса и данных составляют основную часть выводов микропроцессора). Способом решения этой проблемы стала идея использовать общую шину данных и шину адреса для всех внешних данных, а внутри процессора использовать шину данных, шину команд и две шины адреса. Такую концепцию стали называть модифицированной Гарвардской архитектурой. Разделение шин в модифицированной Гарвардской структуре осуществляется при помощи раздельных управляющих сигналов: чтения, записи или выбора области памяти (например, микроконтроллер ADuC812).

Большинство производителей современных 8-разрядных МК используют Гарвардскую архитектуру. Однако Гарвардская архитектура является недостаточно гибкой для реализации некоторых программных процедур. Поэтому сравнение МК, выполненных по разным архитектурам, следует проводить применительно к конкретному приложению.

В Гарвардской архитектуре принципиально невозможно производить операцию записи в память программ, что исключает возможность случайного разрушения управляющей программы в случае неправильных действий над данными. Эти особенности определили области применения этой архитектуры построения микропроцессоров. Гарвардская архитектура применяется в микроконтроллерах, где требуется обеспечить высокую надежность работы аппаратуры и в сигнальных процессорах, где эта архитектура кроме обеспечения высокой надежности работы устройств позволяет обеспечить высокую скорость выполнения программы.

Существуют гибридные модификации архитектур, сочетающие достоинства как Гарвардской, так и Принстонской (фон-Неймановской) архитектур. Современные CISC-процессоры обладают раздельной кэш-памятью 1-го уровня для инструкций и данных, что позволяет им за один такт получать одновременно как команду, так и данные для её выполнения, т.е. процессорное ядро, формально, является Гарвардским, но с программной точки зрения выглядит как Принстонское, что упрощает написание программ. Обычно в данных процессорах одна шина используется и для передачи команд, и для передачи данных, что упрощает конструкцию системы. Современные варианты таких процессоров могут иногда содержать встроенные контроллеры сразу нескольких разнотипных шин для работы с различными типами памяти – например, DDR RAM и Flash. Тем не менее, и в этом случае шины, как правило,

используются и для передачи команд, и для передачи данных без разделения, что делает данные процессоры еще более близкими к Принстонской архитектуре при сохранении плюсов Гарвардской архитектуры.

1.2 Организация вычислительной системы. Элементы вычислительного ядра и системы ввода-вывода

Рассмотрим обобщенную структуру некоторой гипотетической вычислительной системы.

Под вычислительной системой (ВС) в рамках данного курса понимается комплекс технических средств (аппаратно-программных), предназначенный для автоматизации обработки информации.

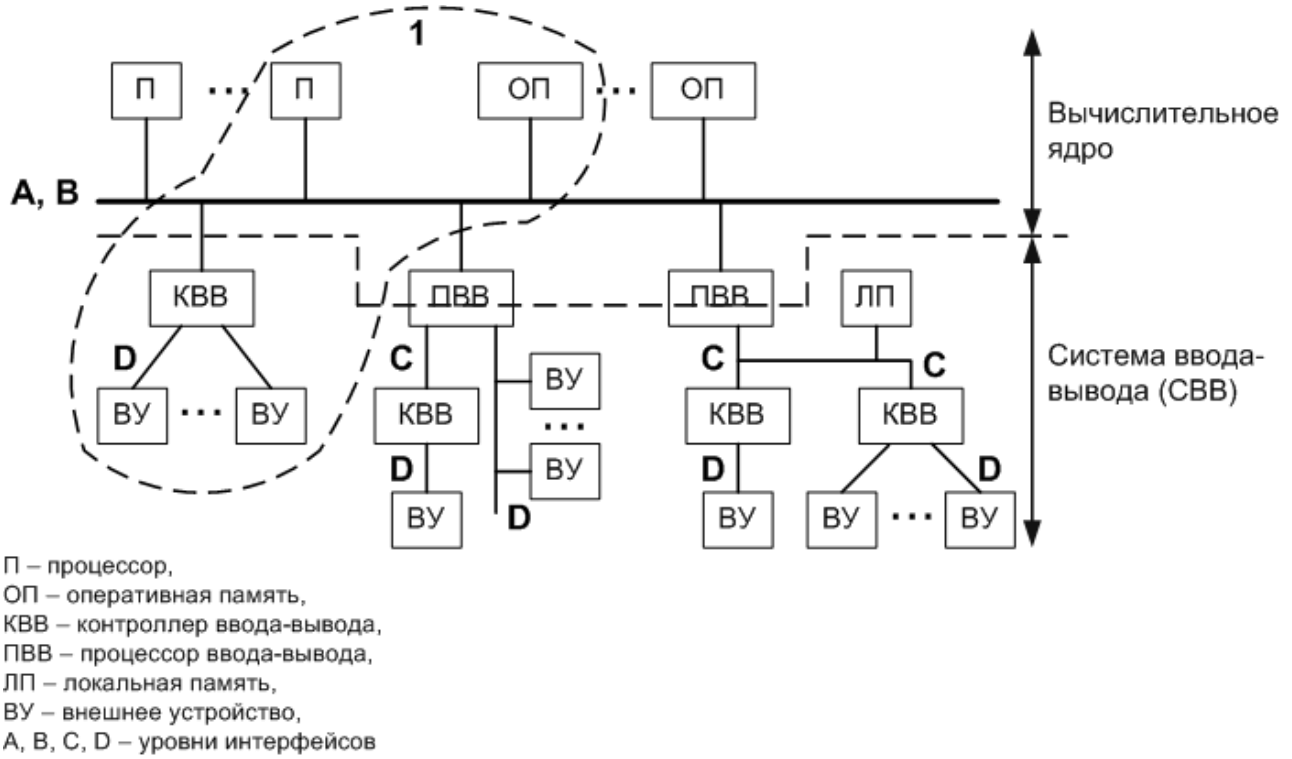


Рис. 2. Структура некоторой гипотетической вычислительной системы (ВС)

Выделенная система на рис. выше (обозначена «1») – простая однопроцессорная система.

Ввод/вывод организуется с помощью ЦП программным методом (с помощью программ ввода/вывода). Может быть несколько контроллеров ввода/вывода. Количество внешних устройств, подключаемых к контроллеру ввода/вывода, может быть различным; характерна шинная организация, где присутствует универсальный интерфейс, объединяющий элементы ядра и системы ввода/вывода.

Достоинства: простота, прозрачность, дешевизна.

Недостатки: надежность, время, низкая производительность.

Архитектура такой системы (простая однопроцессорная система) применялась в первых минимашинах. Сейчас применяется в несложных микроконтроллерах (с 8-разрядными или 16-разрядными ядрами).

Вычислительная система делится на две части:

1. Вычислительное ядро (то оборудование и программные средства, которые непосредственно участвуют в решении заданной задачи).
2. Система ввода-вывода (элементы, обеспечивающие общение вычислительного ядра с внешней средой).

К ядру относятся те элементы, которые непосредственно выполняют вычислительную работу. Поскольку в оперативной памяти обычно хранятся данные, непосредственно использующиеся в вычислительном процессе, этот элемент вычислительной системы («ОП» на рис. выше) однозначно можно отнести к ее ядру. Также к ядру ВС бесспорно относятся центральные процессоры («П» на рис. выше), так как они производят основные вычисления. Кроме того, к ядру относятся те элементы ВС, которые обеспечивают взаимосвязь процессора и оперативной памяти (кэш-память), специальные процессоры (математический сопроцессор), специализированные блоки, отвечающие за синхронизацию, диспетчеризацию, обеспечивающие безопасность рабочих блоков.

Процессоры ввода-вывода («ПВВ» на рис. выше) занимают пограничное положение, и в одних случаях их целесообразно включить в ядро, а в других – в систему ввода-вывода. Например, если в системе процессор ввода-вывода имеет собственную (локальную) память, из которой он выбирает команды и в которой содержатся промежуточные результаты его вычислений, а с центральным процессором он непосредственно не сопряжен, то такой процессор, безусловно, относится к подсистеме ввода-вывода. В случае, когда процессор ввода-вывода разделяет одну и ту же память с центральным вычислителем, его нельзя однозначно отнести к подсистеме ввода-вывода.

К системе ввода-вывода также обычно относят контроллеры ввода-вывода и внешние устройства, которыми они управляют.

Определим понятия, обозначения которых даны на рис. выше.

1.2.1 Процессор и память

Процессор – элемент вычислительной системы, устройство для выборки команд из памяти и выполнения действий, предписанных командами; устройство, осуществляющее процесс обработки информации. В ряде случаев процессором также называют программные средства, предназначенные для обработки информации (например, текстовый процессор, языковой процессор). Процессоры (в смысле устройств) можно классифицировать по разным критериям, например: по способу организации функционирования

(конвейерные, матричные), характеру обрабатываемой информации, по назначению и т.д. Например, различают:

- Универсальные (общего назначения) и специализированные (ПВВ, графические, процессоры обработки сигналов) процессоры. Первые призваны решать различные задачи и имеют широкую область применения, тогда как вторые ориентированы на решение узкого круга задач. Универсальные процессоры характеризуются: способностью обрабатывать большое число команд; системой команд (СК): если система команд позволяет решить любую задачу, то процессор универсальный. Нужно анализировать систему команд на ее сбалансированность, формы данных, способы адресации. Хотя универсальность процессора – вопрос относительный.
- Центральные, периферийные и сервисные процессоры. Центральный процессор осуществляет общее управление вычислительной системой: производит основную обработку данных, обмен ими с другими элементами ВС, а также управляет работой элементов ВС. Периферийный процессор выполняет лишь часть функций вычислительной системы: управляет и обменивается данными с устройствами ввода-вывода (процессор ввода-вывода), может участвовать в вычислительном процессе (обрабатывать часть данных). Сервисный (обслуживающий) процессор обычно не участвует в основном вычислительном процессе и выполняет функции контроля и обслуживания: выполняет инструментальные функции (доставка и отладка программного обеспечения, настройка оборудования), осуществляет контроль правильности функционирования системы, измерение параметров окружающей среды (температура, влажность), напряжения питания и т.п. В ВС один и тот же процессор может выполнять функции как периферийного, так и сервисного процессора.

Программируемость процессора – не обязательное свойство.

Процессоры могут быть:

- Непрограммируемые, не программно реализованные.
- Программируемые и программно реализованные.
- Непрограммируемые, но программно реализованные.
- Программируемые, но не программно реализованные.

Программируемый процессор – процессор, у которого есть система команд. Его можно настроить на решение той или иной задачи.

Функции непрограммируемого процессора раз и навсегда зафиксированы.

Процессоры могут строиться как аппаратные блоки или по принципу программно управляемых устройств.

Микропроцессор – процессор в интегральном исполнении, реализованный в рамках одной или нескольких интегральных микросхем; программируемый процессор в интегральном исполнении.

Микроконтроллер следует понимать как контроллер, построенный на основе микропроцессорной элементной базы. Микроконтроллеры могут быть однокристальными, одноплатными, программируемыми, логическими, промышленными, универсальными и т.д. Микроконтроллер в одном кристалле содержит микропроцессор и набор периферийных устройств и контроллеров: контроллер прерываний, таймеры, контроллер сети, контроллер последовательного канала, контроллер памяти, контроллер ПДП и т.д.

Память – совокупность устройств, предназначенных для хранения программ, обрабатываемой информации (данных), промежуточных или окончательных результатов вычислений.

Важнейшие характеристики памяти – емкость, быстродействие и стоимость. Емкость ЗУ определяется предельным количеством информации, размещаемым в ЗУ, и исчисляется в кило-, мега- и гигабайтах. Быстродействие ЗУ характеризуется затратами времени на чтение запись информации при обращении к ЗУ. Стоимость ЗУ – это затраты средств в денежном выражении на хранение всего объема информации, определяемого емкостью ЗУ. Для сравнения качества ЗУ различных типов используется характеристика, называемая удельной стоимостью и равная стоимости ЗУ, деленной на емкость ЗУ. Удельная стоимость имеет размерность, например, доллар/МБ.

В зависимости от назначения и особенностей реализации устройств памяти, по-разному подходят и к вопросам их классификации.

Критерии классификации:

1. По назначению.
2. По виду физического носителя (технология производства).
3. По организации доступа (адресный: произвольный, прямой (циклический), последовательный; ассоциативный доступ).
4. По возможности записи и перезаписи.
5. По энергозависимости/энергонезависимости.
6. По типу интерфейса.
7. По типу организации адресного пространства.
8. По удалённости и доступности для центрального процессора (первичная, вторичная, третичная память).

Для системы ввода/вывода характерно то, что:

- Память проявляется как категория внешних устройств.
- В составе обеспечивающих устройств, применяется буферная память, чтобы можно было выравнивать скорость работы различных составов системы ввода/вывода.

1.2.2 Контроллер ввода-вывода

Контроллер – устройство, управляющее функционированием отдельных блоков вычислительной системы и внешних устройств, например: вводом-выводом информации, доступом к памяти, к накопителям на магнитных дисках, дисплеям.

Контроллеры ввода-вывода (контроллеры периферийных устройств, КВВ) делятся на:

1. Устройства сопряжения стандартного интерфейса ВС с интерфейсом ВУ (функция преобразования), которые называются адаптерами;
2. Локальные устройства управления конечным оборудованием ВУ (функция управления).

Адаптер является средством сопряжения какого-либо устройства с какой-либо шиной или интерфейсом вычислительной системы. Контроллер служит тем же целям сопряжения, но при этом подразумевается его некоторая активность – способность к самостоятельным действиям после получения команд от обслуживающей его программы.

Внешние (периферийные) устройства – устройства обработки информации, управляемые процессорами или контроллерами ввода-вывода. К ним относятся устройства подготовки данных, устройства ввода и вывода информации, внешние накопители информации (ВЗУ), аппаратура передачи данных, различные преобразователи информации и т.д.

В смысле данного определения к контроллерам можно также отнести и периферийные процессоры. По сути, это они и есть, так как процессоры мы определили как любые устройства – элементы ВС, - обрабатывающие информацию. А поскольку контроллеры находятся в функциональном плане на периферии (см. определение), то их совершенно правильно было бы отнести к периферийным процессорам. Однако мы все-таки будем различать термины контроллер и процессор ввода-вывода. Сузим последнее понятие и, в рамках данного курса, процессорами ввода-вывода будем называть только те устройства обработки информации, которые могут самостоятельно выбирать команды из памяти (программируемые устройства, которые могут самостоятельно организовывать вычислительный процесс). Примеры ПВВ: акселераторы 2D- и 3D-графики; элементы внешней памяти; звуковая карта; сетевая карта. Процессоры ввода-вывода определяются по критерию сложности, но эта граница тоже расплывчата. Если устройство имеет программу, которая выполняется независимо от ЦП или является устройством с множеством сложных функций, то его можно отнести к ПВВ.

Контроллерами ввода-вывода будем называть устройства (или обособленные сложные блоки устройств), не способные самостоятельно избирать команды и для осуществления своих функций требующие управления извне. Простым примером такого устройства может быть контроллер последовательного канала. Для того чтобы он начал передавать посылку по

линии связи, обычно требуется отдать ему определенную команду (в большинстве случаев это запись передаваемых данных в порт контроллера). Далее, несмотря на то, что контроллер самостоятельно принимает посылки, поступающие по линии связи, выделяет из них данные и сохраняет в своем внутреннем буфере, дальнейшие действия по приему посылок он не способен выполнять, пока из его буфера не будут забраны принятые ранее данные. Режимы работы этого контроллера (скорость, количество стоп-бит, контроль четности и т.п.), опять-таки, задаются извне: путем записи в порты контроллера соответствующих значений.

Программируемый КВВ – это устройство, режим работы которого задается программно с помощью управляющих слов (например, устройство, обеспечивающее работы COM-портов), т.е. они конфигурируются извне (задания режимов работы извне).

1.2.3 Процессор ввода-вывода

В вычислительной технике под процессором понимается устройство обработки информации. Несмотря на то, что в основе современных процессоров могут лежать различные модели вычислений исторически сложилось так, что чаще всего используется модель на базе машины Фон-Неймана.

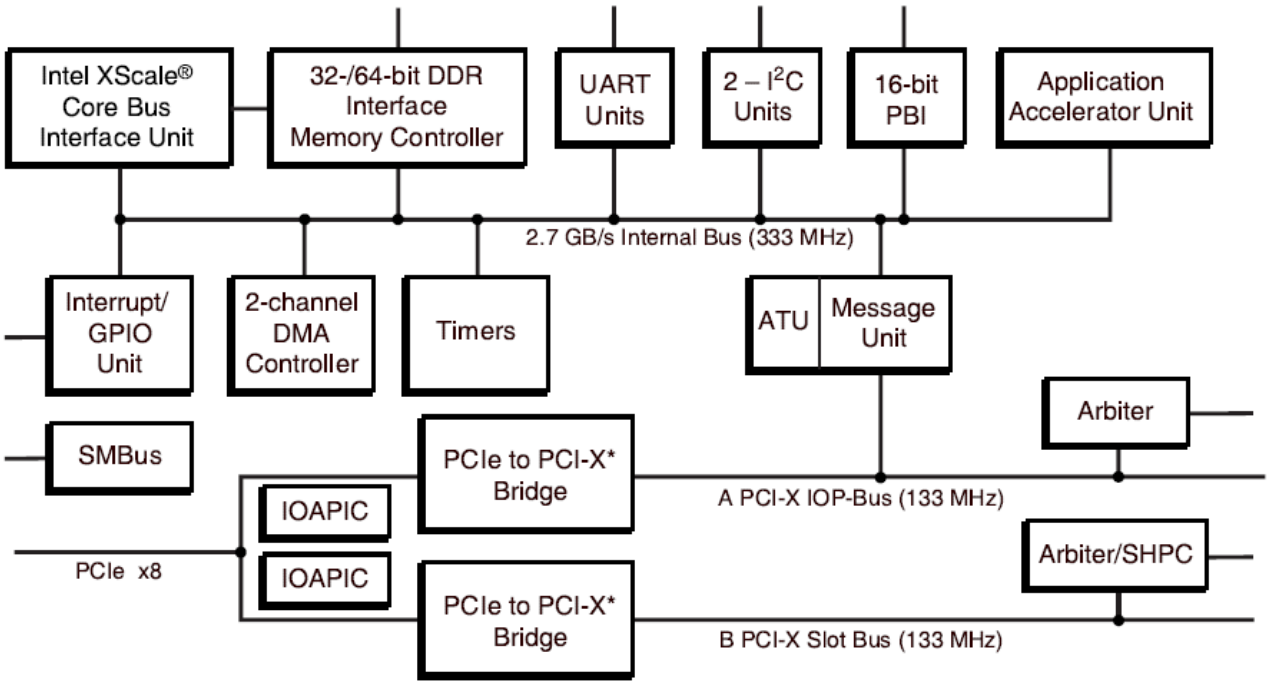


Рис. 3. Структура процессора ввода-вывода Intel 80333

В отличии от обычного процессора, процессор ввода-вывода предназначен для работы в рамках системы ввода-вывода с целью увеличения производительности системы. Увеличение производительности происходит за счет разгрузки центрального процессора и организации параллельной работы СВВ и основного процессора.

В качестве примеров процессоров ввода-вывода можно представить:

- Сетевой контроллер.
- Контроллер жесткого диска.

В качестве примера процессора ввода-вывода можно рассмотреть Intel 80333. В состав этого процессора входит:

- Ядро Intel XScale.
- Контроллер памяти.
- Контроллер UART.
- Контроллер I2C.
- Контроллер DMA.
- Таймеры.
- Мосты PCIe – PCI-X.
- Контроллер и арбитр периферийной шины.
- Контроллер прерываний.
- Порты ввода-вывода (GPIO).

Для сравнения, в большинстве современных микроконтроллеров (таких например, как Freescale Coldfire V2 или NXP LPC1700) выделенного процессора ввода-вывода нет, а есть только контроллеры ввода-вывода.

1.2.4 Интерфейс и протокол

Интерфейс – в широком смысле стык, граница раздела двух взаимодействующих систем, устройств или программ. Также интерфейс можно определить как соглашение о взаимодействии объектов: перечень средств взаимодействия, их параметры, в случае аппаратных интерфейсов – параметры сигналов, способы доступа к средствам взаимодействия, правила взаимодействия и т.д.

Выделяют интерфейсы:

1. Аппаратный (устройство-устройство) – совокупность алгоритмов обмена и технических средств, обеспечивающих обмен между устройствами. Примеры: PCI, RS-232, I^2C, Ethernet.
2. Программный – соглашение о связях в программной среде между программными модулями. Примеры: Win32, POSIX, API любого программного модуля (интерфейс прикладного программирования – набор функций, предоставляемый для использования в прикладных программах).
3. Пользовательский (ВС – пользователь) – сценарии, по которым строится общение оператора с вычислительной системой, и стиль их реализации. Примеры: «дружественный интерфейс человек-компьютер», WIMP

(window, icon, menu, pointers), стиль организации работы в программном комплексе Microsoft Office.

Аппаратный интерфейс – совокупность алгоритмов обмена и технических средств, обеспечивающих обмен между устройствами. В семиуровневой сетевой модели OSI аппаратный интерфейс соответствует физическому и частично канальному уровню, которые определяют физическую и логическую организацию аппаратного интерфейса.

Все множество аппаратных интерфейсов в рамках структуры, изображенной на рисунке выше, можно поделить по их назначению (типу сопрягаемых объектов) на 6 иерархических уровня:

1. Внутрисистемный («A» на рис. выше) – это группа интерфейсов, которая обеспечивает взаимодействие компонент ядра ВС. Интерфейсы этого уровня должны, очевидно, удовлетворять критерию максимальной производительности, например, интерфейс между процессором и памятью, в интерфейсной системе AMBA шина AHB.
2. Системный («B» на рис. выше) – группа интерфейсов, сопрягающих как элементы ядра ВС, так и элементы подсистемы ввода-вывода. Служат для развития системы (ISA, PCI, PCI Express), т.е. наращивания характеристик вычислительного ядра. Является компромиссом при создании дешевой вычислительной структуры.
3. Уровень стандартных интерфейсов ввода-вывода («C» на рис. выше) – группа интерфейсов, объединяющая контроллеры ввода-вывода с процессорами ввода-вывода. Характеристики этих интерфейсов сильно отличаются от характеристик первых двух групп: критерием является удобство и эффективность управления большим числом периферийных устройств. Рассматривается протокол обмена между ведущим и ведомым (интерфейс SCSI, SAS).
4. Уровень малых периферийных интерфейсов («D» на рис. выше), которые сопрягают контроллеры (процессоры) ввода-вывода непосредственно с внешними устройствами (RS-232, SPI, Centronics, SATA), из этого вытекают особенности организации этой группы интерфейсов. Для каждого внешнего устройства требуется свой оптимальный интерфейс.
5. Контроллерные сети. Примеры: 1-Wire, I²C, USB. На рисунке выше не отображены.
6. Сети передачи данных (СПД) систем обработки данных (СОД). На рисунке выше не отображены.

Протокол (от греч. protókollon – первый лист, приклеенный к свитку манускрипта) – документ, содержащий запись всего происходившего. Протоколирование – ведение записей, с информацией о произошедших событиях.

Протокол – правила взаимодействия двух и более систем при передаче данных.

1.2.5 Порт ввода-вывода

Порт можно определить как точку, через которую осуществляется взаимодействие с каким-либо блоком в системе ввода-вывода, многоразрядный вход или выход устройства. Порт ввода-вывода – это логическая адресуемая единица СВВ, которая характеризуется: адресом, форматом данных и набором операций, которые к этому порту можно применять. Взаимодействие может осуществляться как программным путем, так и аппаратным (порт – разъем устройства).

В случае программного взаимодействия совокупность портов нумеруется и представляет собой адресное пространство (т.е. к каждому порту доступ осуществляется по его адресу). Различают порты ввода, вывода и двунаправленные (ввода-вывода). Управление блоками СВВ через порты осуществляется путем записи в них или чтения из них данных. При обращении к порту на линии системного интерфейса выставляется его адрес, который распознается специальным блоком – адресным декодером (или селектором адресов, что то же самое), – расположенным в устройстве, к которому приписан данный порт. Адресный декодер затем инициирует процесс обмена данными (запись или чтение, в зависимости от управляющих сигналов), см. первый пример на рис. выше. Надо сказать, что кроме наличия «нужного» адреса на линиях системного интерфейса для начала процесса обмена с устройством необходимы еще определенные значения управляющих сигналов («чтение», «запись», «Chip-Select» и т.п.). Одному и тому же устройству может соответствовать несколько портов, идущих друг за другом (диапазон адресов) или иначе (вразброс по адресному пространству портов), через которые осуществляется доступ к разным механизмам устройства или к различным частям одного механизма (например, один порт представляет собой регистр адреса внутренней памяти устройства, а через другой пересылаются данные).

Примеры: COM-порт в PC/AT, пространство портов ввода-вывода, порт контроллера ПДП (DMA).

На рис. ниже изображены два примера взаимодействия с устройством через порты. Первый пример иллюстрирует механизм распознавания номеров (адресов) портов, к которым происходит обращение через системный интерфейс. Этот механизм был описан в предыдущем параграфе. Второй пример показывает, как через один и тот же порт может осуществляться доступ к разным ячейкам внутренней памяти устройства. В примере по каждому факту записи через порт адрес ячейки внутренней памяти, в которую будет производиться следующая запись, увеличивается на единицу. Факт записи устанавливается следующим образом: селектором адреса выделяется адрес порта устройства в виде активного сигнала на выходе (т.е. на выходе блока СА), этот сигнал объединяется по «И» с сигналом записи системного интерфейса. На рисунке довольно абстрактно изображена работа с блоком памяти: изображены не все сигналы, не конкретизированы их источники, показано, что из памяти осуществляется чтение из остальной части устройства и др. – это не

принципиально для примера, его цель – показать, как можно организовать доступ к разным ресурсам устройства через один и тот же порт. Вместо памяти можно с таким же успехом поставить несколько блоков, которые будут поочередно выбираться тем же счетчиком (СЧ на рисунке), а вместо доступа по записи можно организовать доступ по чтению или двусторонний обмен (чтение и запись).

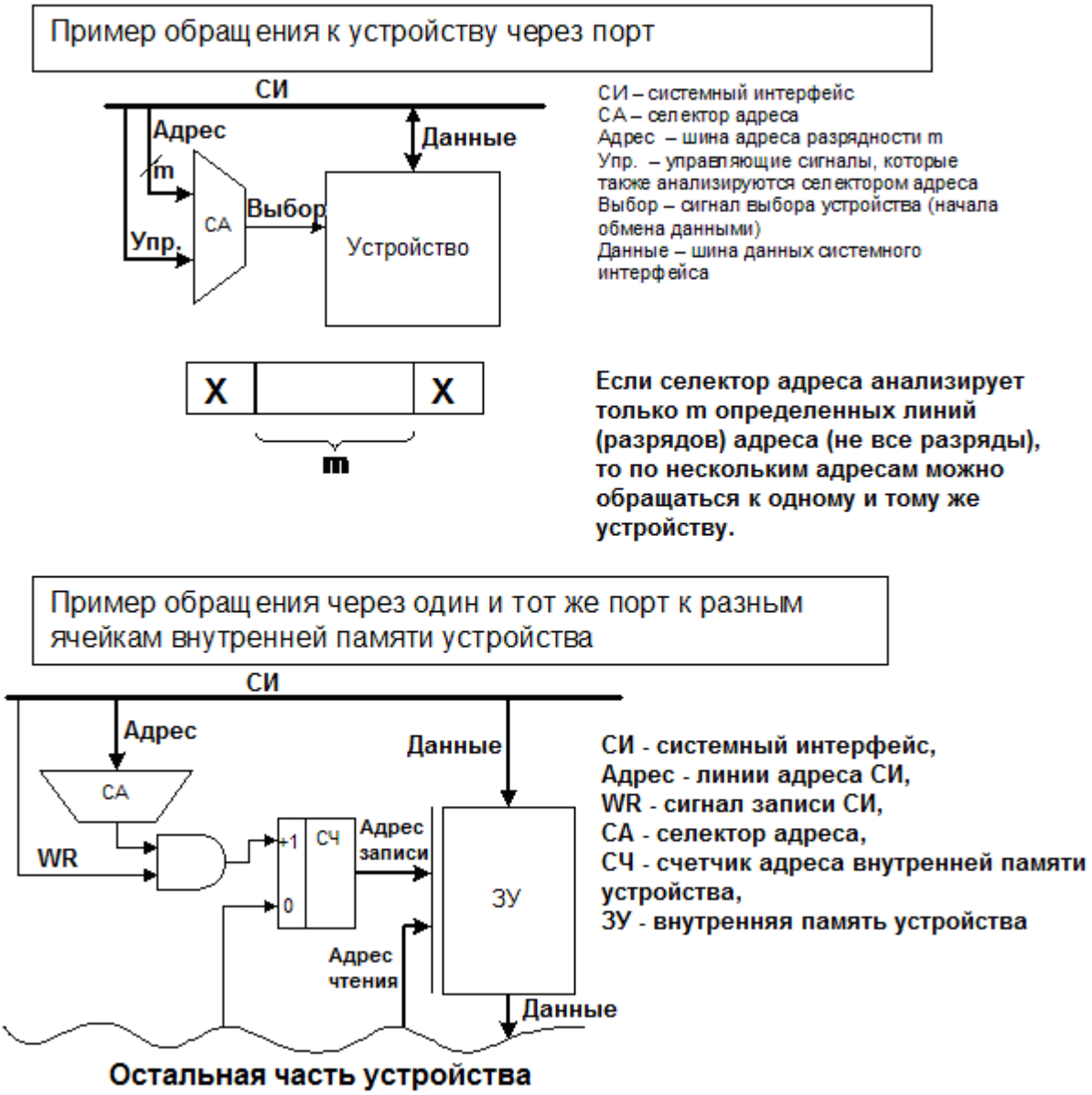


Рис. 4. Примеры аппаратной реализации портов ввода-вывода

Необходимо отметить, что формат данных, передаваемых через порт, не ограничивается форматом данных, пересылаемых через системный интерфейс. Под форматом данных подразумевается как их разрядность, так и положение значащих разрядов. Пусть, например, шина данных системного интерфейса имеет разрядность 8 бит, а в порт записываются четырехбитные данные, причем эти четыре бита могут быть выделены маской 00111100b (единицы соответствуют выделяемой тетраде). В этом случае необходимо считать

данными при обмене через порт только эти разряды. Или пусть у нас 14-разрядный порт, а шина данных - 8-разрядная: данные будут считаться записанными в порт только тогда, когда по одному и тому же адресу будут записаны сначала младшие 8 разрядов (одна операция с 8-битной шиной данных), а затем – старшие 6 (в виде записи байта, из которого только 6 бит принимаются за данные).

1.2.5.1 Адресное пространство портов ввода-вывода: единое с оперативной памятью и раздельное

Устройства ввода-вывода могут быть отображены в адресном пространстве оперативной памяти (memory-mapped I/O). Это называется вводом-выводом, управляемым памятью. Если устройства реагируют на обращения, как обычные компоненты памяти, они могут быть использованы для ввода-вывода, управляемого памятью. Единое адресное пространство портов ввода-вывода и оперативной памяти используется в большинстве современных микроконтроллеров с RISC-ядром: NXP LPC2000, LPC1700, Freescale Coldfire V2, Fujitsu и т. д.

Используется одна и та же адресная шина для обращения к основной памяти и устройствам ввода-вывода. Ввод-вывод, управляемый памятью, обеспечивает дополнительную гибкость программирования. Для доступа к порту ввода-вывода, расположенному в адресном пространстве памяти, могут использоваться любые работающие с памятью команды. Например, команда MOV позволяет пересылать данные между портом и любым регистром. Команды AND, OR и TEST могут использоваться для манипулирования отдельными битами в регистрах управления и состояния периферийных устройств. Ввод-вывод, управляемый памятью, может использовать для адресации портов ввода-вывода полный набор команд и режимов адресации памяти.

Достоинства

Не нужно во время схемотехнического проектирования кристалла процессора включать отдельные модули для организации и управления работой с портами ввода-вывода. Таким образом, микросхема процессора проще, производительнее, дешевле, может потреблять меньше электроэнергии и меньше по размерам.

При сегодняшних технологиях производства микроконтроллеров вопрос экономии памяти не стоит уже так остро, как несколько лет назад. Поэтому разделение по этой причине адресных пространств основной памяти и портов ввода-вывода не имеет такой значимости.

Недостатки

В случае микропроцессоров с единым адресным пространством операции ввода-вывода могут замедлять операции обращения к основной памяти.

В процессорах семейства Intel x86 используется раздельное адресное пространство оперативной памяти и портов ввода-вывода. В таком случае есть специальный класс инструкций в системе команд для взаимодействия с устройствами ввода-вывода. Например, в процессорах семейства Intel x86 это команды IN и OUT, при помощи которых можно читать и писать от 1 до 4 байтов из/в УВВ. Существует два вида команд ввода-вывода:

1. Команды, выполняющие пересылку отдельного элемента (байта, слова или двойного слова) в регистр или из регистра (для x86 IN, OUT). Они адресуют порты ввода-вывода либо напрямую, по адресу одного из портов, задаваемому непосредственно в команде, либо косвенно, при помощи адреса в регистре. Эти команды синхронизируют выполнение программы с работой внешнего аппаратного обеспечения. Буферы записи процессора очищаются, а выполнение программы откладывается до тех пор, пока не будет получен сигнал готовности (ready) последнего цикла шины.
2. Команды, пересылающие строки элементов (строки байт, слов или двойных слов), расположенных в памяти. Эти команды называются «строковыми командами ввода-вывода» или «блочными командами ввода-вывода» (для x86 INS, OUTS).

Кроме того, для отдельного адресного пространства может быть отдельная шина адреса.

Достоинства

Экономичное решение для процессоров со скромными емкостями памяти и адресными пространствами. Меньшая разрядность адреса дает меньшую разрядность команды. Раздельное адресное пространство – решение проблемы замедления обращения к основной памяти из-за активного ввода-вывода.

Недостатки

Ограничение в командах: чаще всего только загрузить из порта ввода-вывода в регистр процессора и обратно. Любая арифметическая, логическая операция над данными порта превращается как минимум в три инструкции. Недостатком дополнительного адресного пространства является уменьшение регулярности процессов обращения к памяти, что в свою очередь усложняет оптимизацию архитектуры микроконтроллера с точки зрения увеличения производительности.

1.2.5.2 Организация конфигурируемых параллельных портов ввода-вывода

Порты ввода-вывода (GPIO) современных микроконтроллеров являются сложными, многофункциональными устройствами. Практически каждый GPIO можно настроить как порт ввода (тогда он будет иметь высокое входное сопротивление) или порт вывода (тогда, мы получим возможность выдавать на

этот порт значения логического нуля и единицы). В некоторых микроконтроллерах есть возможность притягивать выход микроконтроллера к нулю или логической единице с помощью внутренних резисторов. Таким образом, мы можем выбирать схему подключения: с открытым коллектором, с закрытым коллектором, при этом подтянутую к логической «1» или логическому «0».

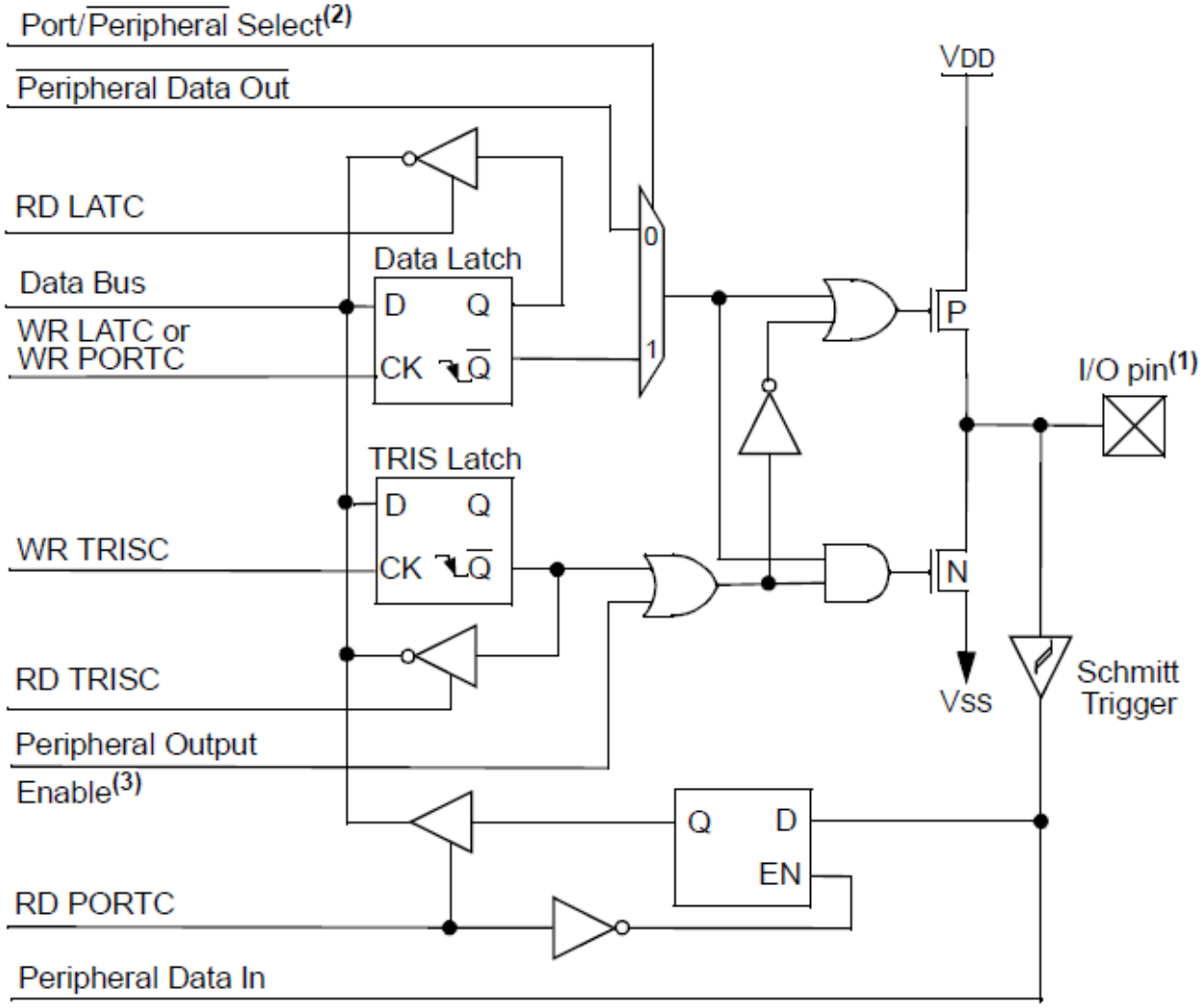


Рис. 5. Пример реализации одного из портов ввода-вывода в Microchip PIC 18

В некоторых микроконтроллерах есть возможность настраивать порты ввода-вывода таким образом, чтобы при возникновении фронта или спада импульса вырабатывалось прерывание.

Часть выводов микроконтроллера может работать не только как простые битовые порты ввода-вывода, но и как выходы и входы различных контроллеров (АЦП, ЦАП, UART, I2C, CAN, SPI и т.д), т.е. такие порты ввода-вывода обладают альтернативной функцией.

1.3 Принципы организации систем ввода-вывода

В основе организации систем ввода-вывода лежат следующие принципы:

1. Принцип программного управления элементами СВВ.
2. Адресуемость элементов СВВ (прозрачность доступа для программиста).
3. Многоуровневая организация СВВ.
4. Параллельность работы элементов СВВ.

Первое положение распространяет принцип программного управления на элементы системы ввода-вывода. Каждое устройство в системе способно выполнять команды, на основе которых строится алгоритм работы с ним и реализуется часть поставленной перед вычислительной системой задачи.

В соответствии со вторым принципом, устройства (элементы) СВВ можно адресовать, т.е. организовать доступ к конкретному устройству по его адресу (номеру). Совокупность устройств можно, таким образом, видеть как группу адресов, если угодно, адресное пространство. Это адресное пространство (или пространства) может быть отдельным, а может входить в другое адресное пространство (например, пространство адресов ячеек памяти).

Принцип многоуровневой организации СВВ, с одной стороны, обеспечивает гибкость системы, сбалансированность по сложности и стоимости ее компонент, облегчает изменение ее конфигурации: добавление и замену отдельных модулей без перепроектирования всей системы. С другой стороны, многоуровневая организация СВВ позволяет осуществить декомпозицию задачи ввода-вывода.

Смысл четвертого положения заключается в том, что различные элементы СВВ как отдельные устройства (группы устройств) могут работать параллельно. В некоторых случаях это ключевым образом влияет на производительность системы, позволяя выполнять больший объем вычислений (ввода-вывода, преобразования информации) за фиксированный промежуток времени. Однако в большинстве случаев значительного повышения производительности добиться не удается. Это объясняется тем, что зачастую выполнение задачи не удается распараллелить вследствие специфики алгоритма даже при наличии такой возможности. Пусть, например, два устройства ввода-вывода могут получать и обрабатывать данные одновременно, но данные, полученные через одно устройство, должны, вследствие алгоритма работы системы, предшествовать данным, полученным от другого устройства – пока не будут получены данные от первого устройства, результаты работы второго не будут востребованы. В этом случае эффективность параллельной работы этих устройств сильно снижается, так как позволяет устранить лишь потерю информации, введенной во второе устройство до завершения обработки данных в первом. Эту ситуацию стоит сравнить со случаем, когда алгоритм работы системы организован так, что для использования данных от второго устройства не требуется наличия данных от второго.

1.3.1 Организация СВВ универсальных ЭВМ

В основе создания СВВ универсальных ЭВМ лежат следующие критерии:

1. Обеспечение максимальной загрузки ЦП.
2. Обеспечение решения задач для широкого круга пользователей.
3. Обеспечение эффективной системы внешней памяти.
4. Обеспечение комфортных условий работы для операторов.

Для ПК несколько иной порядок: 2, 4, 1, 3. При этом добавляется еще один критерий – стоимость (точнее сказать доступность для широких слоёв населения). В настоящее время ПК позиционируется в основном как компьютер для офиса или домашний мультимедийный центр развлечений.

В состав периферийного оборудования типового ПК как правило входит:

1. Набор средств для взаимодействия оператора и компьютера (дисплей, манипуляторы и т. д.).
2. Устройства внешней памяти, их как минимум две группы:
   а) устройства быстродействующей памяти с несменным носителем (HDD, винчестеры);
   б) устройства со сменным носителем.
3. Устройства для получения жесткой копии, читаемой оператором (принтер).
4. Коммуникационные устройства (модемы, сетевые контроллеры)ю
5. Мультимедийные устройства.

Рассмотрим пример организации современного ПК на базе чипсета H57. СВВ на базе такого чипсета содержит:

1. До 8 портов PCIEx1 (PCI-E 2.0, но со скоростью передачи данных PCI-E 1.1).
2. До 4 слотов PCI.
3. 6 портов Serial ATA II на 6 устройств SATA300 (SATA-II, второе поколение стандарта), с поддержкой режима AHCI и функций вроде NCQ, с возможностью индивидуального отключения, с поддержкой eSATA и разветвителей портов.
4. Возможность организации RAID-массива уровней 0, 1, 0+1 (10) и 5 с функцией Matrix RAID (один набор дисков может использоваться сразу в нескольких режимах RAID: например, на двух дисках можно организовать RAID 0 и RAID 1, под каждый массив будет выделена своя часть диска).
5. 14 устройств USB 2.0 (на двух хост-контроллерах EHCI) с возможностью индивидуального отключения.

6. MAC-контроллер Gigabit Ethernet и специальный интерфейс (LCI/GLCI) для подключения PHY-контроллера (i82567 для реализации Gigabit Ethernet, i82562 для реализации Fast Ethernet).
7. High Definition Audio.

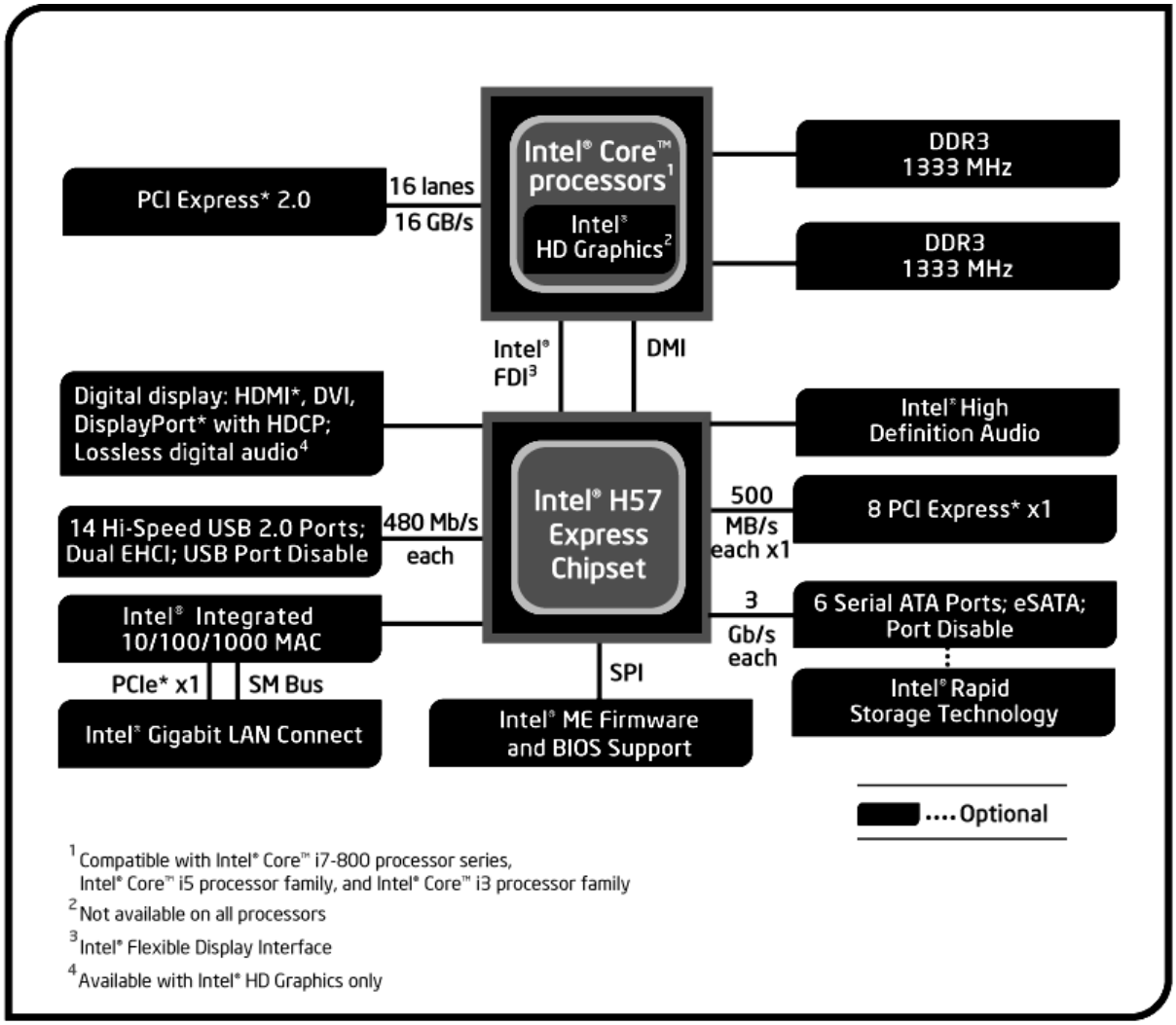


Рис. 6. Чипсет H57 фирмы Intel

1.3.2 Организация СВВ управляющих ЭВМ

Управляющие ЭВМ имеют следующую специфику:

1. Ориентация на задачи управления.
2. Минимально возможное энергопотребление.
3. Повышенная надёжность.
4. Сравнительно небольшие вычислительные ресурсы (память, быстродействие).
5. Интерфейс оператора минимален или отсутствует вообще.

В управляющих ЭВМ, как правило, используют специализированные процессоры для встраиваемых применений и устройства сопряжения с

объектом (УСО) для работы с реальным объектом в условиях агрессивной окружающей среды.

1.3.2.1 Порты ввода-вывода

Каждый процессор для встраиваемых применений имеет некоторое количество внешних линий ввода-вывода, подключенных к внешним выводам микросхемы и называемых внешними портами. Одиночные (одноразрядные, состоящие из одной линии) порты ввода-вывода объединяются в группы обычно по 4, 8 или 16 линий, которые называются параллельными портами. Разрядность параллельных портов может быть нестандартной, например, 5-разрядный порт у микроконтроллера PIC16F84.

Через порты процессорное ядро взаимодействует с различными внешними устройствами: считывает значения входных сигналов и устанавливает значения выходных сигналов.

Во встраиваемых системах в качестве внешних устройств чаще всего рассматриваются датчики, исполнительные устройства, устройства ввода-вывода данных оператором, устройства внешней памяти.

По типу сигнала различают порты:

1. Дискретные (цифровые) – используются для ввода-вывода дискретных значений логического «0» или «1».

В большинстве современных процессоров для встраиваемых применений поддерживается как независимое управление каждой линией параллельного порта, так и групповое управление всеми разрядами. Так как схемотехника отдельных линий в рамках одного 4-, 8- или 16-разрядного порта одинакова, то дальше будут рассматриваться устройство и функционирование одиночного разряда.

2. Аналоговые – через них вводятся сигналы на вход АЦП или других аналоговых схем и выводятся выходные сигналы ЦАП или других аналоговых схем.

Аналоговые порты (или перестраиваемые порты в аналоговом режиме) используются подключения внешних сигналов к ЦАП, АЦП или аналоговым компараторам, встроенным приемопередатчикам. В режиме работы с ЦАП, АЦП или компаратором порты обычно позволяют вводить сигнал в диапазоне от 0В- до Uпит+ (индексы + и – означают чуть больше и чуть меньше, примерно на 200-300мВ). В режиме приемопередатчика параметры сигналов определяются конкретным интерфейсом. В большинстве случаев аналоговые или цифровые линии подключения к приемопередатчикам вообще не называют портами, хотя они по схемотехнике и по месту в структуре процессора близки к универсальным портам ввода-вывода. Реализация входных и выходных каскадов зависит от схемы АЦП, компаратора, ЦАП или приемопередатчика.

3. Перестраиваемые – настраиваются на аналоговый или цифровой режим работы.

По направлению передачи сигнала различают:

1. Однонаправленные порты, предназначенные только для ввода (входные порты, порты ввода) или только для вывода (выходные порты, порты вывода).
2. Двунаправленные порты, направление передачи которых определяется в процессе программно управляемой настройки схемы.
3. Порты с альтернативной функцией. Отдельные линии этих портов связаны со встроенными периферийными устройствами, такими, как таймер, контроллеры последовательных приемопередатчиков. Если соответствующий периферийный модуль не задействован, то линии можно использовать как обычные порты, если модуль активизирован, то связанные с ним линии автоматически или «вручную» (программно) конфигурируются в соответствии с функциональным назначением и не могут быть использованы в качестве универсальных портов ввода-вывода. В некоторых случаях порты могут использоваться только для связи с периферийным модулем (например, входы АЦП в некоторых процессорах).

По алгоритму обмена различают порты:

1. С программно управляемым (программным) вводом-выводом: установка и считывание данных определяется только ходом вычислительного процесса. Нет защиты от повторного считывания-записи одного и того же (неизменившегося) значения на выводе и считывания-записи во время переходного процесса на выводе.
2. Со стробированием: каждая операция ввода-вывода подтверждается импульсом синхронизации (стробом) со стороны источника сигнала (при выводе – процессор, при вводе – внешнее устройство). Считывание информации приемником происходит только по стробу, что позволяет защититься от приема данных во время переходного процесса входного сигнала. Пример: порт PSP (Parallel slave port) в ОКМЭВМ PICmicro.
3. С полным квитированием. Данный режим чаще всего используется для обмена данными с другой вычислительной системой по параллельной шине. Кроме сигналов синхронизации со стороны передатчика используются сигналы подтверждения (готовности к следующему обмену) со стороны приемника. Это позволяет управлять интенсивностью обмена обеим взаимодействующим сторонам и предотвращает потерю данных, когда одна из них перегружена. Пример порта с квитированием – порт LPT персонального компьютера. Во встроенных модулях процессоров данный режим чаще всего реализуется программно-аппаратно.

1.3.2.2 Дискретные порты ввода-вывода

В большинстве современных процессоров для встраиваемых применений поддерживается как независимое управление каждой линией параллельного порта, так и групповое управление всеми разрядами. Так как схемотехника отдельных линий в рамках одного 4-, 8- или 16-разрядного порта одинакова, то дальше будут рассматриваться устройство и функционирование одиночного разряда.

1.3.2.3 Однонаправленные порты

Схема однонаправленного порта ввода представлена на рис. ниже.

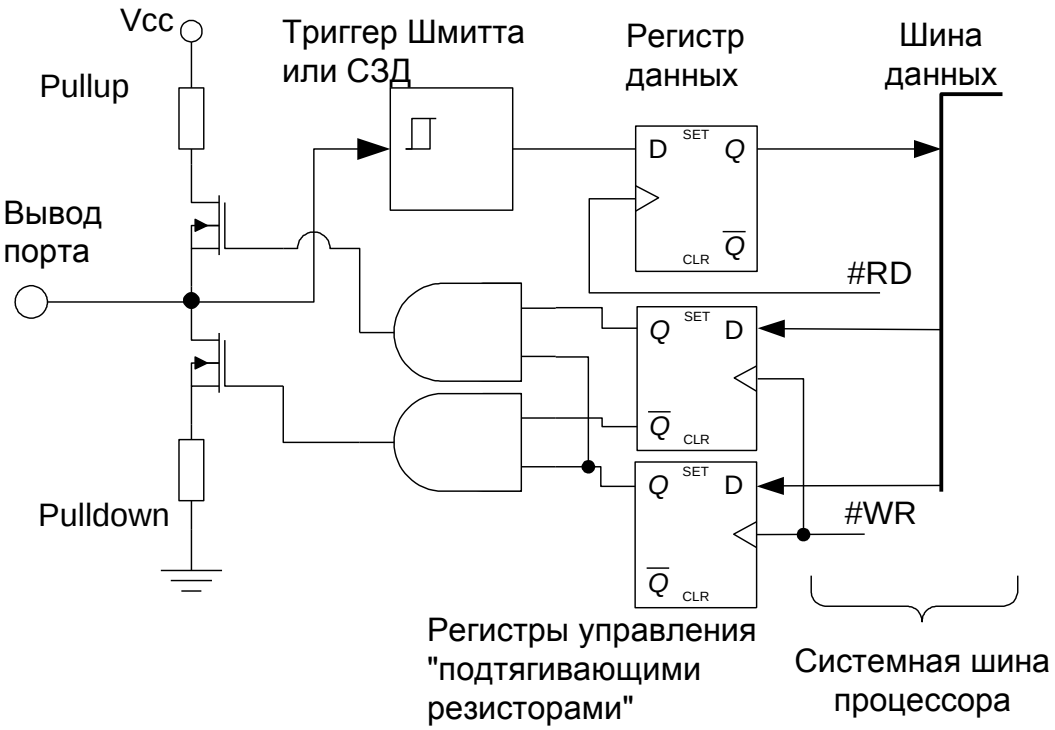


Рис. 7. Однонаправленный порт ввода

Внешние данные считываются через вывод порта (ножку микросхемы), проходят через триггер Шмитта (ТШ) или схему защиты от дребезга (СЗД) и по внутреннему сигналу чтения фиксируются в регистре данных, с выхода которого, в свою очередь, данные считываются процессором.

ТШ (используется в большинстве процессоров для встроенного применения) имеет гистерезис по уровню входного напряжения и предотвращает многократное переключение входных схем при пологом фронте сигнала или помехах.

СЗД (например, в семействе Zilog Z8) вводит инерционность переключения и отсекает реакцию на короткие по длительности импульсы. Используется для защиты от помех.

К входу также могут подключаться так называемые «резисторы поддержки» логической «1» (Pull-up) или логического «0» (Pull-down). Эти резисторы предназначены для переведения входов в устойчивое состояние «0»

или «1» и предотвращения произвольных переключений от помех в моменты, когда на них (входы) не подается внешний сигнал, например, неиспользуемых и не подключенных к внешним схемам входов («открытых входов»). Через специальные управляющие регистры «схемы поддержки» могут быть отключены полностью или включены в режим Pull-up или Pull-down.

Все перечисленные блоки: триггер Шмитта, СЗД и «схемы поддержки» – используются для защиты от случайных переключений в результате помех и помогают снизить энергопотребление, которое резко возрастает в момент переключений входных схем.

Порты вывода бывают:

- С двухтактной выходной схемой (комплементарные).
- С однотактной выходной схемой и внутренней нагрузкой.
- С открытым выходом (открытым коллектором или стоком).

Порты вывода с двухтактной выходной схемой являются самыми распространенными и реализованы, например, в семействах Atmel AVR, Microchip PICmicro, AMD AM186, Motorola HC08, HC11 и многих других.

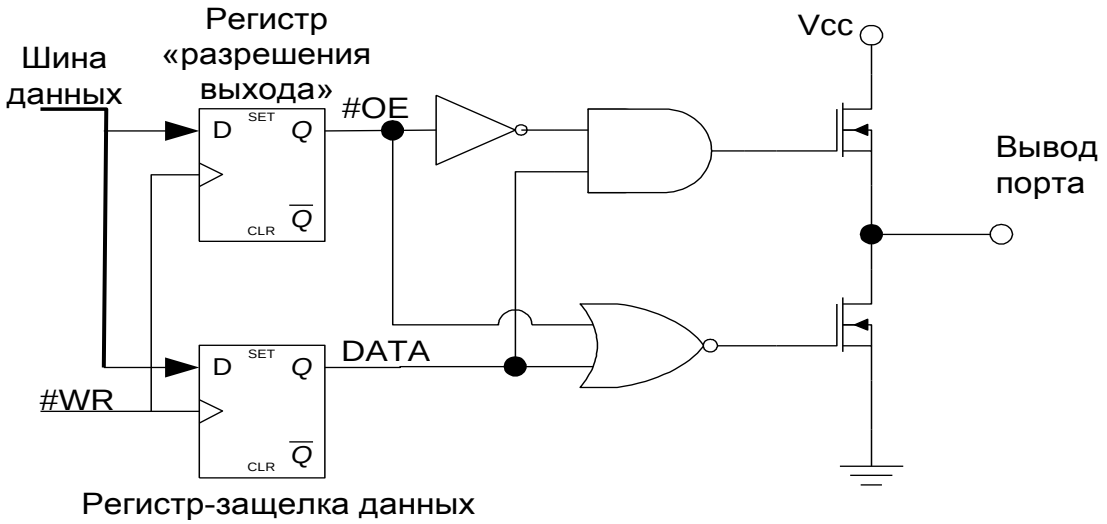


Рис. 8. Порт вывода с двухтактной схемой

Рассмотрим функционирование данной схемы.

Выходные данные записываются в регистр-защелку данных по внутреннему сигналу записи #WR и через простейшую логическую схему управляют выходными транзисторами. Если в регистр записано значение «1», то открыт верхний по схеме транзистор, а нижний закрыт: на выводе порта Vcc (логическая «1»). Если в регистр записано значение «0», то открыт нижний по схеме транзистор, а верхний закрыт: вывод порта соединен с минусовой шиной питания, т.е. там установлен «0».

На схеме верхний регистр управляет сигналом #OE – «разрешение выходов». Если в регистр записан «0», то схема работает, как было описано выше. Если записана «1», то оба транзистора закрываются и схема переводится в «высокоомное» состояние (Z-состояние). В этом состоянии выходное сопротивление порта очень высокое и он фактически «оторван» от микропроцессора. Это необходимо:

- Если к выходному порту подключены выходы других схем и необходимо разделять линии передачи данных с этими устройствами. Например: наш процессор используется как периферийный контроллер и его выходной порт подключен к периферийной шине другого процессора (мастера), к этой же шине подсоединены еще несколько периферийных контроллеров.
- В схемах двунаправленных портов (см. ниже).

Достоинство:

Максимальные значения втекающего (в состоянии «0») и вытекающего (в состоянии «1») тока выхода составляют 2-6мА для каскадов с нормальной нагрузочной способностью (например, Fujitsu MB90) и 5-30мА для каскадов с повышенной нагрузочной способностью (например, PICmicro, AVR). Встречаются отдельные микросхемы со сверхвысокой нагрузочной способностью – до 60-90мА (например, PIC17). Большой выходной ток позволяет непосредственно с ножки, без схем усиления и согласования сигнала, управлять достаточно мощной нагрузкой: светодиодами, реле, мощным электронным ключом (транзистор, тиристор). Это значительно упрощает схему устройств.

Недостатки:

- При программировании необходимо управлять дополнительным битовым регистром «разрешение выхода».
- Значительное энергопотребление и уровень помех при переключении. Последний особенно зависит от скорости переключения. Для ограничения токов в момент переключений иногда используют специальные демпфирующие схемы. Однако они снижают быстродействие портов. Наибольшее применение демпфирующие схемы находят в портах ПЛИС в силу их особо высокого быстродействия.
- Относительно сложная внутренняя схема, повышающая сложность и стоимость микросхемы в целом. Однако на нынешнем этапе, в связи с успехами технологии производства микросхем, это уже не является проблемой.

Порты вывода с однотактной выходной схемой и внутренней нагрузкой применяются, например, в семействе MCS-51. Они имеют более простую внутреннюю схему.

Когда в регистр-защелку записано значение «1», транзистор закрыт и на выходе через резистор R_L устанавливается Vcc – логическая «1». Когда же в регистр-защелку записан «0», открывается транзистор и соединяет выход с минусовой шиной питания, т.е. там устанавливается «0». При этом резистор R_L оказывается подключенным между шинами питания. Во избежание высокого тока через резистор и его перегрева сопротивление делают достаточно высоким 10-100кОм. Высокое сопротивление резистора позволяет непосредственно соединять несколько выходов, не опасаясь их встречного включения, так как если «0» на одном из выходов «подсадит» «1» на другом, то мощность, выделяемая на «подсаженном» резисторе будет мала, он не перегреется и каскад не выйдет из строя.

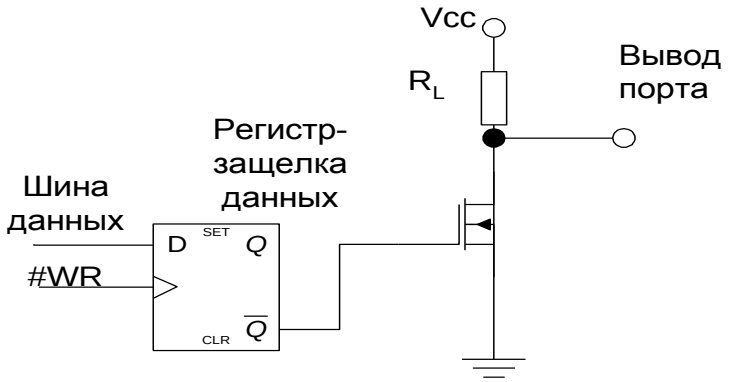


Рис. 9. Порт вывода с однотактной схемой

Достоинства:

- Необходимо управлять только одним регистром.
- Простая схема.
- Возможность без дополнительных схем организовать подключение на одну внешнюю шину несколько таких выходов. Легко построить квазидвунаправленный порт ввода-вывода (см. ниже).

Недостаток:

Малый вытекающий ток (в состоянии «1»), ограниченный резистором R_L – сотни мкА. Это не дает управлять относительно мощными нагрузками без дополнительных каскадов усиления либо требует обеспечивать, чтобы активным был сигнал со значением «0» («управление нулем»).

Порты вывода с открытым выходом (открытым коллектором или стоком)

Применяются во многих семействах микропроцессоров, например, AMD Am186 (там это один из режимов порта), PICmicro. Выходной каскад построен

по однотактной схеме с внешней нагрузкой. Принцип функционирования аналогичен описанному для однотактного выходного каскада.

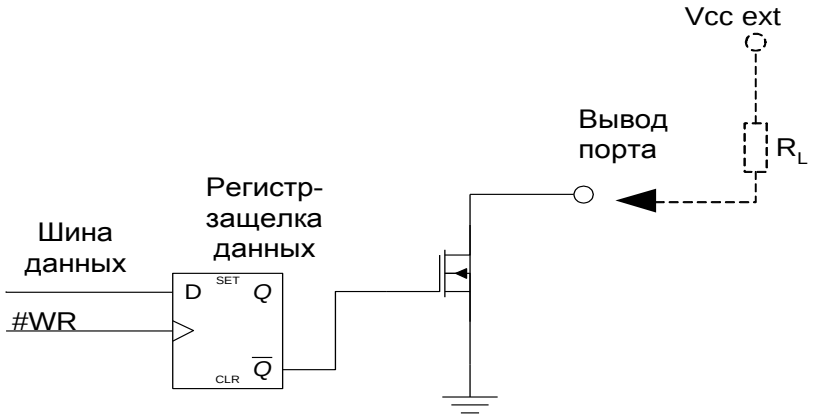


Рис. 10. Порт вывода с открытым выходом

Достоинства:

- Внешнее напряжение питания нагрузки Vcc ext может быть иным – большим или меньшим, чем питание микропроцессора. Это может быть удобным для сопряжения схем с различными уровнями логической «1», например, 3,3В и 5В. Если внешнее напряжение достаточно высокое, то можно непосредственно управлять высоковольтной нагрузкой. Например, анонсирован микроконтроллер семейства PICmicro допускающий подключение внешнего напряжения Vcc ext до 15В при питании ядра 2-6В.
- Необходимо управлять только одним регистром.
- Простая схема.
- Возможность без дополнительных схем организовать подключение на одну внешнюю шину несколько таких выходов. При этом можно подбирать требуемое сопротивление RL, например, стандарт I2C требует чтобы сопротивление было 2.2кОм. Легко построить квазидвунаправленный порт ввода-вывода (см. ниже).

Недостатки:

- Требуется внешняя нагрузка.
- Малый вытекающий ток (в состоянии «1»), ограниченный внешним нагрузочным резистором.

1.3.2.4 Двунаправленные порты и порты с альтернативной функцией

Самой простой схемой двунаправленного порта является квазидвунаправленный порт со схемой, аналогичной схеме порта вывода с однотактным выходным каскадом.

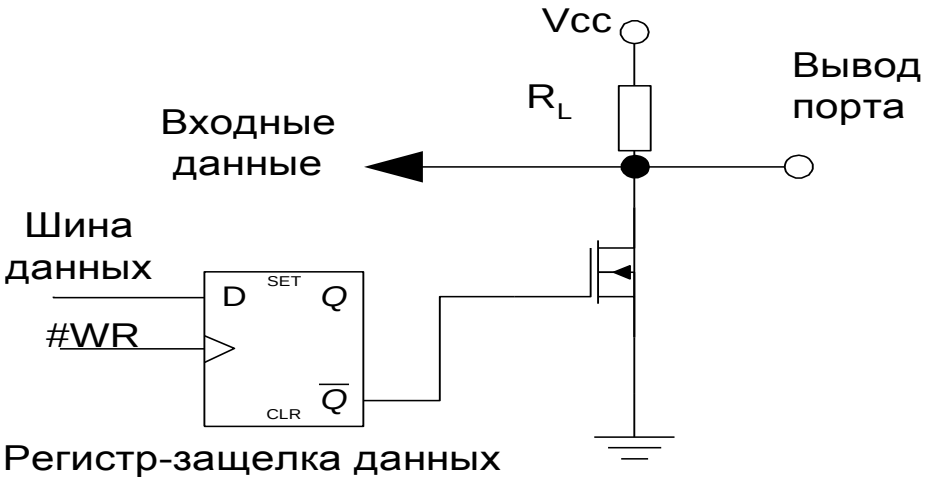


Рис. 11. Квазидвунаправленный порт

Регистр входных данных (на схеме не показан) подключен к внешнему выводу порта. Перед считыванием входных данных необходимо предварительно записать «1» в регистр-защелку выходных данных. Это закроет транзистор и исключит влияние порта вывода на входной сигнал. Резистор R_L останется подключенным к входному сигналу и будет являться для него дополнительной нагрузкой, но, так как сопротивление резистора велико (10-100кОм), даже на маломощный входной сигнал данная нагрузка не окажет заметного влияния. Схема квазидвунаправленного порта применяется в семействе MCS-51.

Более часто используется схема переключаемого двунаправленного порта с комплементарным выходным каскадом.

Она объединяет схемы порта ввода и порта вывода с двухтактной выходной схемой, описанные выше. Переключение порта в режим ввода осуществляется записью «1» в регистр «вход/выход». В этом случае (как было указано при описании порта вывода) оба транзистора переводятся в закрытое состояние и порт вывода не влияет на входной сигнал. В двунаправленных портах резисторы pull-up и pull-down подключаются только в режиме ввода, для чего на вход соответствующей схемы управления подключается выход регистра «вход/выход» («1» - ввод).

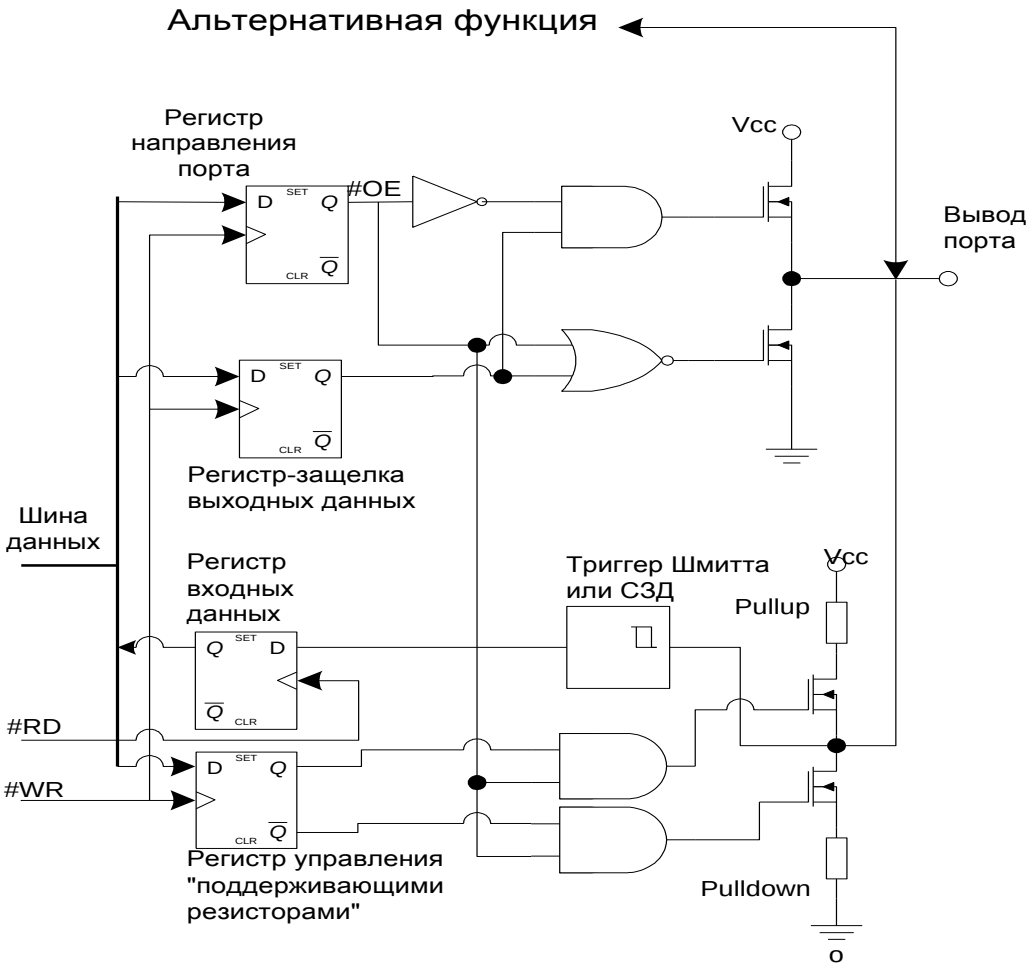


Рис. 12. Переключаемый двунаправленный порт с комплементарным выходным каскадом

Кроме исполнения функции порта ввода-вывода внешние выводы микросхемы могут быть задействованы для связи с внутренними периферийными модулями микропроцессора, а также с подсистемами процессорного ядра, схем памяти и управления (с контроллером прерываний, блоком интерфейса внешней памяти и т.п.). Данные функции называются альтернативными. Обычно, когда вывод порта используется для выполнения альтернативной функции, основные схемы переводятся в состояние ввода или вообще отключаются.

1.3.2.5 Аналого-цифровой преобразователь

Модуль аналого-цифрового преобразования (АЦП, Analog-to-digital converter, ADC) предназначен для ввода в процессор аналоговых сигналов с датчиков физических величин и преобразования значения напряжения этих сигналов в двоичный код с целью дальнейшей программной обработки [28, 32, 77, 91, 100]. Простейшим одноразрядным двоичным АЦП является компаратор.

Характеристики:

1. Разрядность АЦП характеризует количество дискретных значений, которые преобразователь может выдать на выходе.
2. Разрешение АЦП – минимальное изменение величины аналогового сигнала, которое может быть преобразовано данным АЦП. Обычно измеряется в вольтах, поскольку для большинства АЦП входным сигналом является электрическое напряжение.
3. Частота дискретизации.
4. Скорость преобразования.
5. Точность.

Пример:

Диапазон входных значений = от 0 до 10В (опорное напряжение)

Разрядность двоичного АЦП 12 бит: $2^{12} = 4096$ уровней квантования

Разрешение двоичного АЦП по напряжению: (10-0)/4096 = 0,00244В = 2,44мВ

На практике разрешение АЦП ограничено отношением сигнал/шум входного сигнала. При большой интенсивности шумов на входе АЦП различение соседних уровней входного сигнала становится невозможным, т.е. ухудшается разрешение. При этом реально достижимое разрешение описывается эффективной разрядностью (effective number of bits – ENOB), которая меньше, чем реальная разрядность АЦП. При преобразовании сильно зашумлённого сигнала младшие разряды выходного кода практически бесполезны, так как содержится шум.

Наиболее часто путаемыми характеристиками АЦП являются разрешающая способность и точность, хотя эти две характеристики реального АЦП крайне слабо связаны между собой. Разрешение не идентично точности, 12-разрядный АЦП может иметь меньшую точность, чем 8-разрядный. Для АЦП разрешение представляет собой меру того, на какое количество сегментов может быть поделен входной диапазон измеряемого аналогового сигнала (например, для 8-разрядного АЦП это 256 сегментов). Точность же характеризует суммарное отклонение результата преобразования от своего идеального значения для данного входного напряжения. Т.е., разрешающая способность характеризует потенциальные возможности АЦП, а совокупность точностных параметров определяет реализуемость такой потенциальной возможности. АЦП преобразует входной аналоговый сигнал в выходной

цифровой код. Для реальных преобразователей, изготавливаемых в виде интегральных микросхем, процесс преобразования не является идеальным: на него оказывают влияние как технологический разброс параметров при производстве, так и различные внешние помехи. Поэтому цифровой код на выходе АЦП определяется с погрешностью. В спецификации на АЦП указываются погрешности, которые дает сам преобразователь. Их обычно делят на статические и динамические. При этом именно конечное приложение определяет, какие характеристики АЦП будут считаться определяющими, самыми важными в каждом конкретном случае.

В большинстве применений АЦП используют для измерения медленно изменяющегося, низкочастотного сигнала (например, от датчика температуры, давления, от тензодатчика и т.п.), когда входное напряжение пропорционально относительно постоянной физической величине. Здесь основную роль играет статическая погрешность измерения. В спецификации АЦП этот тип погрешности определяют аддитивная погрешность (Offset), мультипликативная погрешность (Full-Scale), дифференциальная нелинейность (DNL), интегральная нелинейность (INL), погрешность квантования и апертурная погрешность.

Аналоговый сигнал является непрерывной функцией времени, в АЦП он преобразуется в последовательность цифровых значений. Следовательно, необходимо определить частоту выборки цифровых значений из аналогового сигнала. Частота, с которой производятся цифровые значения, получила название частоты дискретизации АЦП.

Непрерывно меняющийся сигнал с ограниченной спектральной полосой подвергается оцифровке (т.е. значения сигнала измеряются через интервал времени T – период дискретизации), и исходный сигнал может быть точно восстановлен из дискретных во времени значений путём интерполяции. Точность восстановления ограничена ошибкой квантования. Однако в соответствии с теоремой Котельникова-Шеннона точное восстановление возможно, только если частота дискретизации выше, чем удвоенная максимальная частота в спектре сигнала.

Поскольку реальные АЦП не могут произвести аналого-цифровое преобразование мгновенно, входное аналоговое значение должно удерживаться постоянным, по крайней мере, от начала до конца процесса преобразования (этот интервал времени называют «время преобразования»). Эта задача решается путём использования специальной схемы на входе АЦП – устройства выборки-хранения – УВХ. УВХ, как правило, хранит входное напряжение в конденсаторе, который соединён со входом через аналоговый ключ: при замыкании ключа происходит выборка входного сигнала (конденсатор заряжается до входного напряжения), при размыкании – хранение. Многие АЦП, выполненные в виде интегральных микросхем, содержат встроенное УВХ.

Полученное в результате преобразования значение записывается в регистр данных (РД). АЦП, интегрированные в кристалл процессора, обычно строят по схеме последовательного приближения. Время преобразования обычно составляет несколько десятков микросекунд, в зависимости от частоты тактирования АЦП. Завершение процесса преобразования отмечается установкой флага, и если разрешено, вырабатывается запрос прерывания. В современных управляющих процессорах и микроконтроллерах наиболее распространены АЦП с разрядностью 8, 10, реже 12 и совсем редко 14 и 16 бит.

Аналоговый коммутатор выбирает один из возможных аналоговых входов (выводов) и подключает его к входу внутреннего АЦП для преобразования. При последовательной выборке каналов создается имитация многоканального АЦП. Применение действительно многоканальных АЦП резко повышает энергопотребление и стоимость процессора и обычно не используется (Если требуется несколько каналов и высокая скорость преобразования, то используют микросхему внешнего АЦП).

Код выбора канала может формироваться программно, т.е. программист «вручную» переключается между каналами, или аппаратно (автоматически), последовательно перебирая каналы (режим сканирования).

Для большего удобства использования модуля АЦП в режиме сканирования могут быть реализованы несколько регистров данных (Fujitsu MB90, Intel 8051GB) по одному на канал. Программисту будет достаточно считывать данные из регистра, соответствующего требуемому каналу. При этом код выбора канала параллельно подается на адресные входы блока регистров данных.

Источник опорного напряжения Vref и коммутатор Vref

Опорное напряжение Vref определяет диапазон значений напряжения на аналоговых входах и разрешающую способность АЦП, равную $Vref/2^n$, где n – разрядность АЦП. Если значение напряжения на входе невелико, то точность преобразования может быть увеличена путем уменьшения Vref. Диапазон допустимых значений Vref обычно находится в рамках значения напряжения питания процессора.

Могут быть использованы опорные источники следующего типа:

1. Внешний, подключаемый через специальный вывод микросхемы.
2. Внутренний, фиксированный или программируемый с помощью встроенного ЦАП.

Подключение к АЦП внешнего или внутреннего источников выполняется с помощью коммутатора Vref.

Коммутатор сигнала запуска АЦП позволяет выбрать способ запуска процесса преобразования, а также определяет один из возможных режимов работы АЦП:

1. Периодического преобразования. В этом режиме АЦП запускается периодическим сигналом от основного тактового генератора или встроенного таймера.
2. Если сигнал запуска подать на двоичный счетчик, выходами подключенный к управляющим входам аналогового коммутатора и адресным линиям блока регистров данных, то таким образом легко реализовать режим последовательного сканирования каналов.
3. Внешнего запуска. Запуск осуществляется внешним сигналом, что позволяет четко определить момент считывания значения аналогового напряжения со входа.
4. Программно управляемого запуска, по установке специального бита.

Блок управления модулем АЦП конфигурирует и синхронизирует функционирование других (вышеперечисленных) блоков, управляется программно через регистры специального назначения.

Аналоговый компаратор

Аналоговый компаратор используется для сравнения напряжения двух внешних аналоговых сигналов или для сравнения напряжения внешнего аналогового сигнала с образцовым напряжением, вырабатываемым внутри процессора. Могут быть запрограммированы различные уровни образцового напряжения. Результат сравнения кодируется битом в регистре специального назначения, например, «1» – вход А больше или равен В, «0» – вход А меньше В. В случае изменения соотношения изменяется значение бита, а также может быть установлен флаг и выработан запрос прерывания.

Структура блока аналогового компаратора приведена ниже.

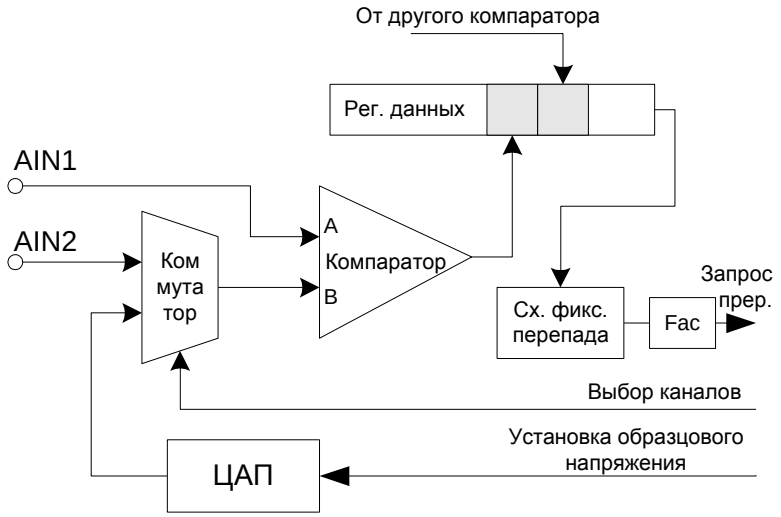


Рис. 13. Модуль аналогового компаратора

Аналоговый коммутатор входов выбирает аналоговые сигналы для сравнения. Первый сигнал берется с внешнего входа AIN1, в качестве второго

берется сигнал с внешнего входа AIN2 или эталонное внутреннее напряжение, которое вырабатывается с помощью ЦАП.

ЦАП – программируемый генератор эталонного напряжения.

Регистр данных – программно доступный регистр, в битах которого сохраняются результаты сравнения одного или нескольких компараторов.

Схема фиксации перепада определяет изменение одного из бит в регистре данных (выхода одного из компараторов) и вырабатывает по этому событию запрос прерывания.

Пример использования аналогового компаратора:

1. Контроль превышения допустимых значений температуры, давления, тока, напряжения и других физических величин. Физическая величина преобразуется в напряжение с помощью датчика и контролируется с помощью аналогового компаратора. Порог сравнения устанавливается встроенным генератором образцового напряжения.
2. Обнаружение (формирование) фронтов внешних сигналов.
3. Встроенные схемы контроля напряжения питания системы.

В настоящее время известно большое число методов преобразования напряжение-код. Эти методы существенно отличаются друг от друга потенциальной точностью, скоростью преобразования и сложностью аппаратной реализации.

Рис. 14. Классификация АЦП по методам преобразования

В основу классификации АЦП положен признак, указывающий на то, как во времени разворачивается процесс преобразования аналоговой величины в

цифровую. В основе преобразования выборочных значений сигнала в цифровые эквиваленты лежат операции квантования и кодирования. Они могут осуществляться с помощью либо последовательной, либо параллельной, либо последовательно-параллельной процедур приближения цифрового эквивалента к преобразуемой величине.

Параллельные АЦП

АЦП этого типа осуществляют квантование сигнала одновременно с помощью набора компараторов, включенных параллельно источнику входного сигнала. На рисунке показана реализация параллельного метода АЦ-преобразования для 3-разрядного числа. С помощью трех двоичных разрядов можно представить восемь различных чисел, включая нуль. Необходимо, следовательно, семь компараторов. Семь соответствующих эквидистантных опорных напряжений образуются с помощью резистивного делителя.

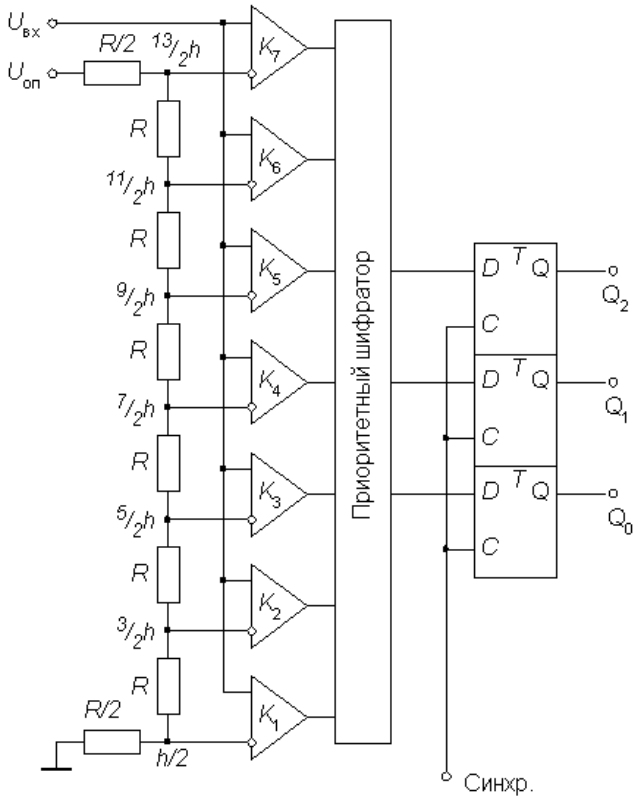


Рис. 15. Схема параллельного АЦП

Если приложенное входное напряжение не выходит за пределы диапазона от 5/2h, до 7/2h, где h=Uоп/7 – квант входного напряжения, соответствующий единице младшего разряда АЦП, то компараторы с 1-го по 3-й устанавливаются в состояние «1», а компараторы с 4-го по 7-й – в состояние «0».

Подключение приоритетного шифратора непосредственно к выходу АЦП может привести к ошибочному результату при считывании выходного кода. Рассмотрим, например, переход от трех к четырем, или в двоичном коде от 011 к 100. Если старший разряд вследствие меньшего времени задержки изменит

свое состояние раньше других разрядов, то временно на выходе возникнет число 111, т.е. семь. Величина ошибки в этом случае составит половину измеряемого диапазона.

Так как результаты АЦ-преобразования записываются, как правило, в запоминающее устройство, существует вероятность получить полностью неверную величину. Решить эту проблему можно, например, с помощью устройства выборки-хранения (УВХ). Некоторые интегральные микросхемы (ИМС) параллельных АЦП, например, MAX100, снабжаются сверхскоростными УВХ, имеющими время выборки порядка 0,1 нс. Другой путь состоит в использовании кода Грея, характерной особенностью которого является изменение только одной кодовой позиции при переходе от одного кодового значения к другому. Наконец, в некоторых АЦП (например, MAX1151) для снижения вероятности сбоев при параллельном АЦ-преобразовании используется двухтактный цикл, когда сначала состояния выходов компараторов фиксируются, а затем, после установления состояния приоритетного шифратора, подачей активного фронта на синхровход выходного регистра в него записывают выходное слово АЦП.

Благодаря одновременной работе компараторов параллельный АЦП является самым быстрым. Например, восьмиразрядный преобразователь типа MAX104 позволяет получить 1 млрд. отсчетов в секунду при времени задержки прохождения сигнала не более 1,2 нс. Недостатком этой схемы является высокая сложность. Действительно, N-разрядный параллельный АЦП сдержит 2N-1 компараторов и 2N согласованных резисторов. Следствием этого является высокая стоимость и значительная потребляемая мощность.

АЦП последовательного приближения

АЦП последовательного приближения (successive approximation architecture, SAR), или АЦП с поразрядным уравновешиванием содержит компаратор, вспомогательный ЦАП и регистр последовательного приближения. АЦП преобразует аналоговый сигнал в цифровой за N шагов, где N – разрядность АЦП.

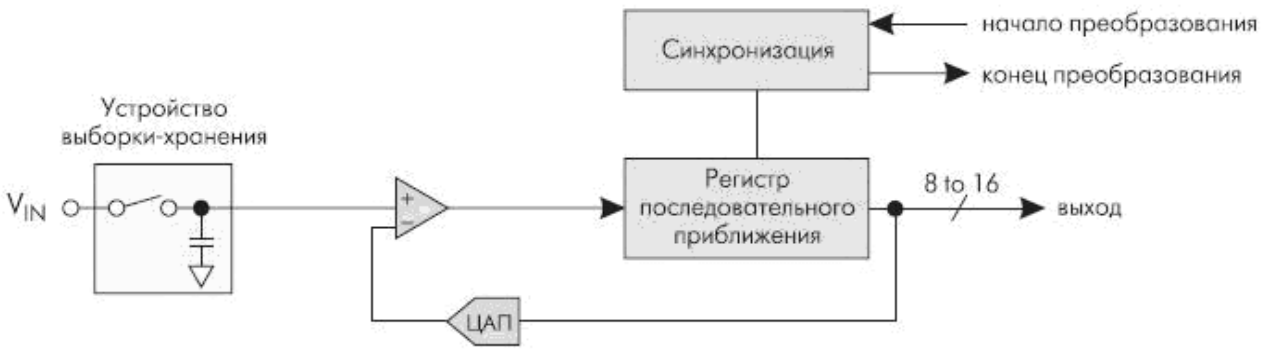


Рис. 16. АЦП последовательного приближения

На каждом шаге определяется по одному биту искомого цифрового значения, начиная от старшего значащего разряда (СЗР) и заканчивая младшим значащим разрядом (МЗР). Последовательность действий по определению очередного бита заключается в следующем. На вспомогательном ЦАП выставляется аналоговое значение, образованное из битов, уже определённых на предыдущих шагах; бит, который должен быть определён на этом шаге, выставляется в 1, более младшие биты установлены в 0. Полученное на вспомогательном ЦАП значение сравнивается с входным аналоговым значением. Если значение входного сигнала больше значения на вспомогательном ЦАП, то определяемый бит получает значение 1, в противном случае 0. Таким образом, определение итогового цифрового значения напоминает двоичный поиск. АЦП этого типа обладают одновременно высокой скоростью и хорошим разрешением. Однако при отсутствии устройства выборки хранения погрешность будет значительно больше (представьте, что после оцифровки самого большого разряда сигнал начал меняться).

Характеристика АЦП последовательного приближения: невысокая скорость преобразования, невысокая цена и низкое энергопотребление.

1.3.2.6 Цифро-аналоговый преобразователь

Цифро-аналоговый преобразователь (digital-analog converter, DAC) предназначен для преобразования числа, представленного, как правило, в виде двоичного кода, в напряжение или ток, пропорциональные этому числу. Схемотехника цифро-аналоговых преобразователей весьма разнообразна. На рисунке ниже представлена общая классификация ЦАП по способам преобразования входного кода и схемам формирования выходного сигнала [28, 32, 77, 91, 100].

Дальнейшую классификацию цифро-аналоговых преобразователей можно провести по ряду специфических признаков, например:

1. По роду выходного сигнала: преобразователи с токовым выходом или с выходом по напряжению.
2. По типу цифрового интерфейса: с последовательным вводом или параллельным вводом.
3. По числу ЦАП на кристалле: одноканальные и многоканальные.
4. По быстродействию: низкого, среднего и высокого быстродействия.
5. По разрядности.

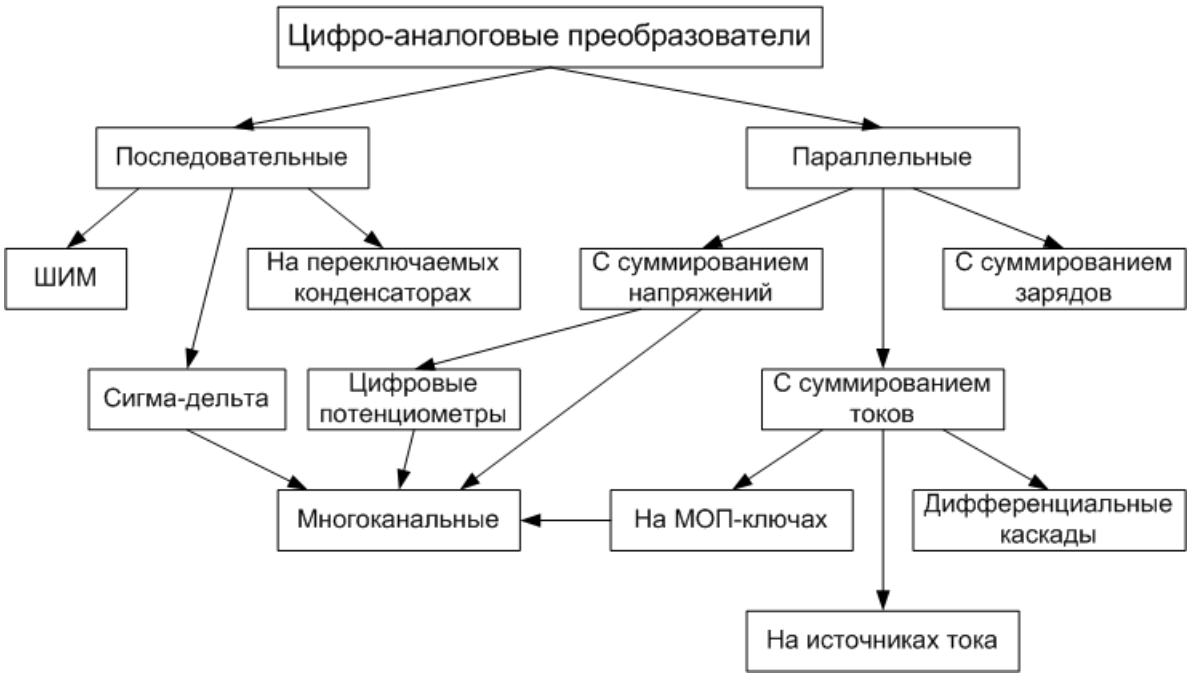


Рис. 17. Обобщенная классификация ЦАП

Матрица R-2R – самый распространенный метод цифро-аналогового преобразования. Матрица работает по принципу деления входного напряжения на входах. Матрица имеет число входов по числу разрядов регистра данных. На каждый вход через ключ может быть подано опорное напряжение Vref или 0В. Ключи управляются разрядами регистра данных: «1» – на матрицу подается Vref, «0» – подается 0В.

Коммутатор опорного напряжения Vref позволяет выбрать внешний или встроенный источник опорного напряжения.

В регистр данных записывается цифровой код. Регистр данных определяет разрядность ЦАП.

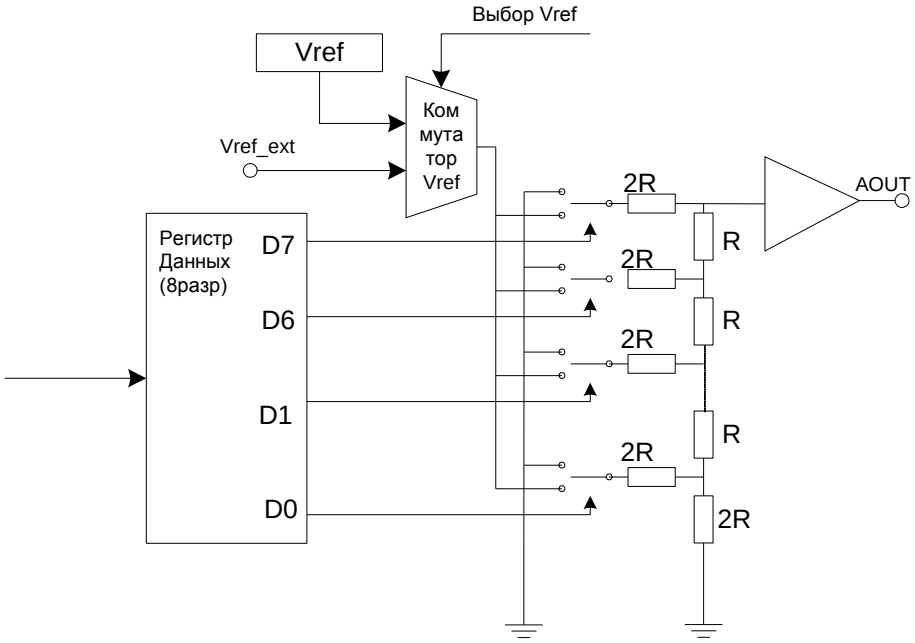


Рис. 18. Модуль ЦАП с типом преобразования «Матрица R-2R»

На практике ЦАП применяется для управления различными исполнительными устройствами (приводами) и системами: электродвигателями постоянного тока с переменной скоростью вращения, источниками питания с управляемым напряжением, различными индикаторами и т.п. С помощью ЦАП можно синтезировать аналоговые сигналы различной формы, например, синусоидальной.

1.3.2.7 Устройства сопряжения с объектом (УСО) управляющих ЭВМ

Устройства сопряжения с объектом (УСО), которые также называют модулями ввода-вывода, выполняют функции адаптера датчиков и исполнительных устройств. Они имеют специальные аппаратные каскады сопряжения с оконечными устройствами и поддерживают алгоритмы управления ими. УСО могут выполнять функции первичной обработки данных с датчиков: фильтрацию, усреднение и накопление. УСО являются подчиненными по отношению к логическим контроллерам и самостоятельно не реализуют каких-либо алгоритмов контроля объекта управления. По способу взаимодействия с логическим контроллером УСО делятся на:

- Локальные, конструктивно совмещенные с логическим контроллером.
- Удаленные, или сетевые, взаимодействующие с логическим контроллером по сетевому каналу, конструктивно независимые от логических контроллеров.

В отличие от УВВ УСО работает с реальным объектом в условиях помех, высоких напряжений и больших токов (например, управление двигателем переменного тока). Как правило, это устройство гальванически развязано с объектом управления. В качестве УСО могут выступать дискретные (битовые) входы и выходы, аналоговые входы и выходы и т.п. Без гальванической развязки управляющий вычислительный комплекс будет постоянно сбоить из-за помех или сгорит при появлении повышенного напряжения на входе УСО.

Устройство сопряжения с объектом является водоразделом, отделяющим «тепличный» мир вычислительного устройства от достаточно агрессивной окружающей среды. УСО должно не только сопрягать различные уровни сигналов, например 24В, необходимые для включения реле, и 5В (TTL-уровень) на выходе битового порта ввода-вывода микроконтроллера, но и не пропускать высокие напряжения во внутренние цепи контроллера.

В некоторых системах (например, в медицинских) главным является сохранение объекта управления, так как от этого напрямую зависит жизнь человека. Такие системы тестируются на степень защиты человека от электрической травмы из-за некорректной работы УСО.

2 Способы обмена информацией между устройствами вычислительной системы

В данной главе рассматривается обмен между ядром вычислительной системы и элементами системы ввода-вывода. Обмен производится с точки зрения программы, выполняющейся на центральных процессорах, которая взаимодействует с блоками СВВ через порты ввода вывода.

Все многообразие способов такого обмена можно разделить на несколько видов:

1. Программно управляемые:
   - Синхронный.
   - Асинхронный с программной проверкой готовности (программный полинг, «по опросу»).
   - Асинхронный с аппаратной проверкой готовности (обмен по прерыванию).
2. В режиме прямого доступа (без участия центральных процессоров).

2.1 Синхронный обмен данными

Синхронный обмен данными предполагает отсутствие ситуации неготовности обменивающихся сторон. Например, при чтении данных из порта предполагается, что устройство всегда готово передать их читающей стороне. При записи в порт, наоборот, устройство всегда готово принять данные.

При синхронном обмене им полностью управляет программа, а элемент СВВ, с которым происходит взаимодействие, никак не может повлиять на ход обмена. Т.е., даже если устройство работает с задержками, то эти задержки учитывает программа, которая с ним взаимодействует, но само устройство не имеет никакой возможности сообщить программе о своей готовности или неготовности.

Основные достоинства:

- Потенциально, синхронный обмен – самый быстрый из всех рассматриваемых в данном разделе.
- Синхронный обмен требует минимум аппаратного обеспечения.

Основной минус: синхронный обмен сложно (или вообще невозможно) организовать с асинхронными устройствами (т.е. с устройствами, имеющими разное время выполнения операций и/или множество производимых операций с сильно различающимися временами выполнения).

2.2 Асинхронный обмен данными с программной проверкой готовности

Асинхронный обмен с программной проверкой готовности предполагает возможность программно оценить степень готовности элемента СВВ, к которым происходит взаимодействие. Обычно для этих целей служит программно доступный (через порт) регистр состояния устройства. Перед тем, как передать данные устройству или забрать их из него, программа имеет возможность определить, готово ли само устройство к этой операции, прочитав значение из порта состояния.

Простым примером может служить работа с контроллером последовательного канала (UART) «по опросу»: перед тем, как прочитать данные из порта данных контроллера, необходимо проверить, являются ли эти данные результатом приема посылки и не забирались ли они программой ранее. Проще говоря, необходимо проверить данные на достоверность. Перед тем же, как записывать данные для передачи в буфер контроллера, необходимо убедиться, что в буфере есть место, т.е. что запись новых данных в буфер не приведет к уничтожению ранее помещенных и еще не переданных данных.

Очевидно, что такой способ обмена требует дополнительных усилий со стороны программы на опрос готовности. Потенциально возможна ситуация выхода устройства из строя («вечная неготовность»), поэтому необходимо соответствующим образом строить алгоритм работы с ним, чтобы программа не «зависала» в бесконечном цикле, дожидаясь готовности со стороны устройства (см. рисунок).

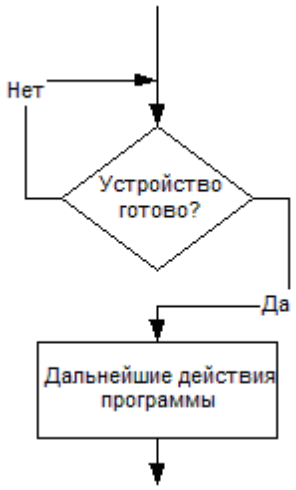


Рис. 19. Нерекомендуемый подход к обмену с устройством

Типичный алгоритм асинхронного обмена с программным опросом готовности (фрагмент программы) изображен на рисунке. Очевидно, такой способ позволяет избежать ситуации «зависания» программы, а также провести диагностику причины (длительной) неготовности устройства и установить факт его выхода из строя.

Несомненным достоинством асинхронного обмена с программной проверкой готовности является то, что программа способна определять степень готовности устройства самостоятельно и учитывать факты отказа (их частоту, длительность неготовности и т.п.) в своих дальнейших действиях.

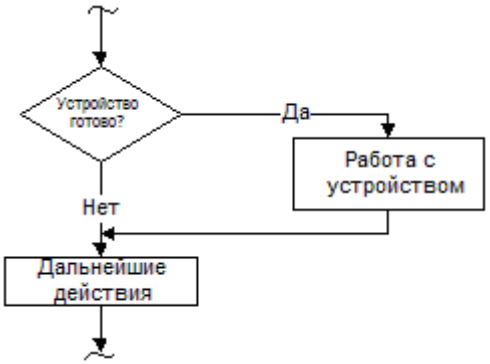


Рис. 20. Рекомендуемый способ асинхронного обмена с устройствами

При этом очевидными «минусами» являются:

- Наличие программных усилий по определению готовности. Инструкции по опросу отнимают процессорное время, в течение которого могли бы выполняться другие части алгоритма. Если бы процедура опроса выполнялась параллельно с основным алгоритмом (скажем, аппаратно), то это бы позволило существенно ускорить работу всей программы.
- Неэффективность при обмене с большим количеством устройств и/или высокой частоте процедуры обмена. Из-за того, что опрос готовности осуществляется программой, накладные расходы на опрос при большом количестве обращений становятся очень высокими и снижают производительность всей системы.

На рисунке приведен типичный вариант программного полинга (опрос флагов прерывания ПУ). Первым проверяется флажок готовности ПУ1 с наибольшим приоритетом. Если оно не запрашивало обслуживание, опрашивается следующее ПУ и т.д.

Когда встречается первое устройство готовое к операциям ввода-вывода (ВВ), управление передается подпрограмме обслуживания этого устройства. При завершении обслуживания в полинге может быть запрограммировано одно из следующих действий:

- Управление возвращается в основную программу без проверки готовности остальных ПУ. Здесь гарантируется обязательная проверка в каждом цикле полинга ПУ с высоким приоритетом, так как обслуживание их блокирует обслуживание устройств с меньшим приоритетом.
- Управление возвращается к программе полинга, т.е. в точку проверки прерывания следующего ПУ (на рисунке ниже это показано цифрой).

Этот способ гарантирует проверку в каждом цикле полинга всех устройств.

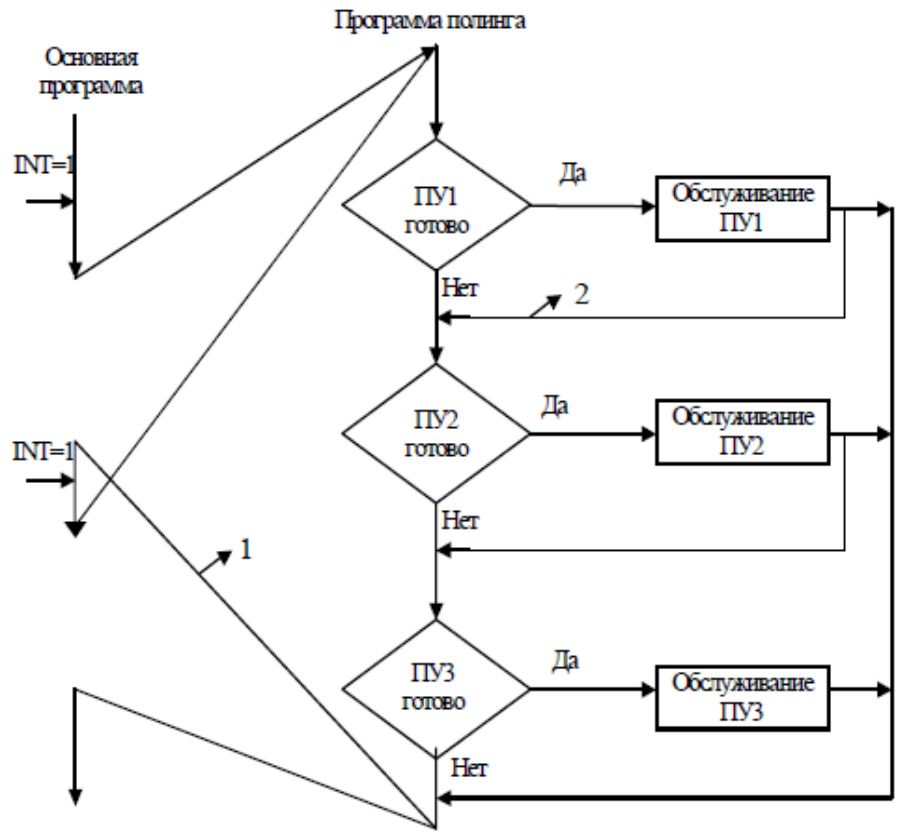


Рис. 21. Программный полинг

Основное достоинство программного полинга заключается в простоте его реализации, почти не требующей дополнительных аппаратных средств.

2.3 Асинхронный обмен данными с аппаратной проверкой готовности

2.3.1 Система прерываний

Прерывание в классической трактовке – прекращение выполнения текущей команды или последовательности команд для обработки некоторого события обработчиком прерывания, с последующим возвратом к выполнению прерванной программы.

Прерывание можно определить как механизм межпроцессного взаимодействия, включающий в себя механизмы передачи данных (тип прерывания) и управления (пуск/останов текущего процесса).

Система прерываний является неотъемлемой частью любой вычислительной системы и предназначена для обеспечения быстрой реакции процессора на ряд ситуаций, требующих его внимания, которое может возникать как в самом процессоре, так и за его пределами.

Прерывания следует рассматривать не только и не столько как реакцию процессора на аномальные ситуации, а как естественный процесс, с помощью которого реализуется поддержка большинства необходимых механизмов, таких как виртуальная память, ввод/вывод и т. п. По образному выражению Питера Нортона: "Прерывание – это движущая сила компьютера".

Система прерываний представляет собой комплекс аппаратных и программных средств. Аппаратные средства системы прерываний обычно называются блоком или контроллером прерываний. В ПК это PIC (Programmable Interrupt Controller), т.е. отдельная микросхема 8259A. В некоторых случаях контроллер прерываний интегрируется в кристалл микропроцессора. Программные средства систем прерываний представляют собой специальные программы – обработчики прерываний (interrupt handler).

Назначение системы прерывания – реагировать на определенные события путем прерывания работы процессора по выполнению программы и переключения процессора на выполнение другой программы, обслуживающей соответствующую ситуацию. В момент возникновения определенного события (причины) формируется сигнал прерывания, который поступает в процессор и инициирует специальную операцию – операцию прерывания, обеспечивающую прерывание одной программы и переключение процессора на выполнение другой программы.

2.3.2 Классификация прерываний

В зависимости от источника возникновения сигнала прерывания делятся на:

- Асинхронные или внешние (аппаратные) – события, которые исходят от внешних источников (например, периферийных устройств) и могут произойти в любой произвольный момент: сигнал от таймера, внешнего интерфейса, АЦП и др.
- Внутренние – события в самом процессоре как результат нарушения каких-то условий при исполнении машинного кода: деление на ноль или переполнение, обращение к недопустимым адресам или недопустимый код операции. Такого рода прерывания еще называются исключительными ситуациями (exceptions).
- Программные (частный случай внутреннего прерывания) – инициируются исполнением специальной инструкции в коде программы. Программные прерывания как правило используются для обращения к функциям встроенного программного обеспечения драйверов и операционной системы.

Аппаратные прерывания могут возникать в произвольные моменты времени и являются асинхронными по отношению к выполняемой программе.

С помощью аппаратных прерываний осуществляется взаимодействие процессора с периферийными устройствами, а также сообщается о различных

ошибках аппаратуры (например, ошибка памяти, ошибка передачи по шине и т. п.) или об аварийном отключении питания. Во втором случае иногда такие прерывания называют исключительными ситуациями (exceptions).

Реагируя на аппаратное прерывание, процессор должен идентифицировать его источник, сохранить минимальный контекст прерываемой программы и переключаться на специальную программу – обработчик прерывания (interrupt handler), который может быть оформлен как процедура или задача.

Действия обработчика прерывания, называемые обслуживанием прерывания, заключается в том, чтобы правильно отреагировать на прерывание конкретного источника (например, поместить символ нажатой клавиши в буфер, произвести инкремент системных часов и т. п.).

После завершения обслуживания прерывания процессор возвращается к выполнению прерванной программы, и она должна продолжиться таким образом, как будто прерывания не было.

К основным ситуациям, возникающим вне процессора и приводящим к прерыванию, относятся:

1. Запросы от управляемого объекта (являются типичными для управляющих систем), т.е. запросы от ВУ:
   - ВУ, готовые к обмену, требуют реакции процессора для организации программно управляемой передачи данных.
   - Завершение работы ВУ или КВВ по передаче данных.
   - Особая (аварийная) ситуация в ВУ или КВВ.
2. Запросы прерываний от других процессоров для обеспечения синхронизации вычислительных процессов, протекающих в рамках многопроцессорной системы.

Программные прерывания, в отличие от аппаратных, появляются синхронно по отношению к выполняемой программе.

Причинами программных прерываний могут служить особые ситуации, возникающие при выполнении программы, препятствующие нормальному продолжению программы и требующие специального обслуживания (переполнение, нарушение защиты памяти, отсутствие нужной страницы в оперативной памяти и т.п.), а также специальные команды типа INT n (n – номер прерывания), являющиеся генераторами программных прерываний. Эти команды обычно используются для вызова определенных функций ОС.

Обработка исключительных ситуаций (exception handling) – механизм языков программирования, предназначенный для описания реакции программы на ошибки времени выполнения и другие возможные проблемы (исключения), которые могут возникнуть при выполнении программы и приводят к невозможности (бессмысленности) дальнейшей отработки программой её

базового алгоритма. В русском языке также применяется более короткая форма термина: «обработка исключений».

Во время выполнения программы могут возникать ситуации, когда состояние данных, УВВ или компьютерной системы в целом делает дальнейшие вычисления в соответствии с базовым алгоритмом невозможным или бессмысленными. Классические примеры подобных ситуаций:

- Нулевое значение знаменателя при выполнении операции целочисленного деления. Результата у операции быть не может, поэтому ни дальнейшие вычисления, ни попытка использования результата деления не приведут к решению задачи.
- Ошибка при попытке считать данные с внешнего устройства. Если данные не удаётся ввести, любые дальнейшие запланированные операции с ними бессмысленны.
- Исчерпание доступной памяти. Если в какой-то момент система оказывается не в состоянии выделить достаточный для прикладной программы объём оперативной памяти, программа не сможет работать нормально.
- Появление сигнала аварийного отключения электропитания системы. Прикладную задачу, по всей видимости, решить не удастся, в лучшем случае (при наличии какого-то резерва питания) прикладная программа может озаботиться сохранением данных.
- Появление на входе коммуникационного канала данных, требующих немедленного считывания. Чем бы ни занималась в этот момент программа, она должна перейти к чтению данных, чтобы не потерять поступившую информацию.

2.3.3 Функции системы прерываний и их реализация

Функции системы прерываний:

1. Прием и хранение запросов прерываний от многих источников.
2. Выделение наиболее приоритетного запроса из множества поступивших.
3. Проверка возможности обработки запросов центральным процессором (проверка замаскированности запросов или сравнение уровня приоритетности запросов с так называемым порогом прерываний).
4. Сохранение состояния (контекста) прерываемой программы.
5. Вызов обработчика прерываний.
6. Собственно обработка прерываний (выполнение программы обработки прерываний).
7. Восстановление состояния (контекста) прерванной программы и возобновление ее выполнения.

Этапы 1-5 выполняются аппаратными средствами компьютера автоматически при появлении запроса прерывания. Этап 7 также выполняется аппаратно по команде возврата из обработчика прерывания.

Процедура опроса источников прерываний с целью выделения наиболее приоритетного (полинг/polling) может быть реализована как на аппаратном, так и на программном уровнях.

Программный полинг (уже был рассмотрен ранее) реализуется специальной программой, которая последовательно опрашивает триггеры запросов, объединенных, как правило, в единый регистр с целью поиска первого установленного бита.

Аппаратный полинг может быть реализован либо на основе многотактной схемы, в основу которой положен двоичный счетчик, либо с помощью однотактной схемы, которую обычно называют дейзи-цепочка (будет рассмотрен далее).

Отношение процессора к поступившим запросам прерываний может быть выражено с помощью одного из двух механизмов:

- Механизм масок.
- Порог прерываний.

Механизм масок основан на использовании специального бита для каждого запроса прерываний, с помощью которого разрешается или запрещается обработка прерываний, связанных с этим запросом. В процессоре семейства Intel функцию маски выполняет бит IF, с помощью которого разрешается (IF=1) или запрещается (IF=0) обработка запросов внешних прерываний (как правило, от ВУ). При сброшенном флаге IF принято говорить, что прерывание замаскировано.

Запросы прерываний от ВУ формируются с помощью PIC (Programmable Interruption Controller), который связан линией запроса с CPU. Запросы от PIC поступают в CPU на внешний вход INTR.

Для управления прерываниями используется маска прерываний (см. рис. ниже), представляющая собой двоичное слово M = m1 m2 … mk с числом разрядов, равным числу маскируемых причин прерывания. Если разряд маски mk = 0, то прерывание по причине k запрещено (замаскировано), если разряд маски mk = 1, то прерывание по причине k разрешено (не замаскировано). Маска прерываний хранится в процессоре, куда она загружается командой УСТАНОВИТЬ МАСКУ A, где A – адрес. По этой команде слово с адресом A загружается в качестве маски в процессор и определяет отношение процессора к сигналам прерывания. Если все разряды маски равны нулю, процессор не реагирует ни на одну причину прерывания.

В простейших процессорах используется следующий способ маскирования прерываний. В систему команд компьютера вводятся две системные команды ЗАПРЕТИТЬ ПРЕРЫВАНИЯ и РАЗРЕШИТЬ ПРЕРЫВАНИЯ, выполнение

которых приводит к запрещению и разрешению прерываний одновременно по всем причинам.

Команды, маскирующие прерывания, относятся к группе привилегированных команд.

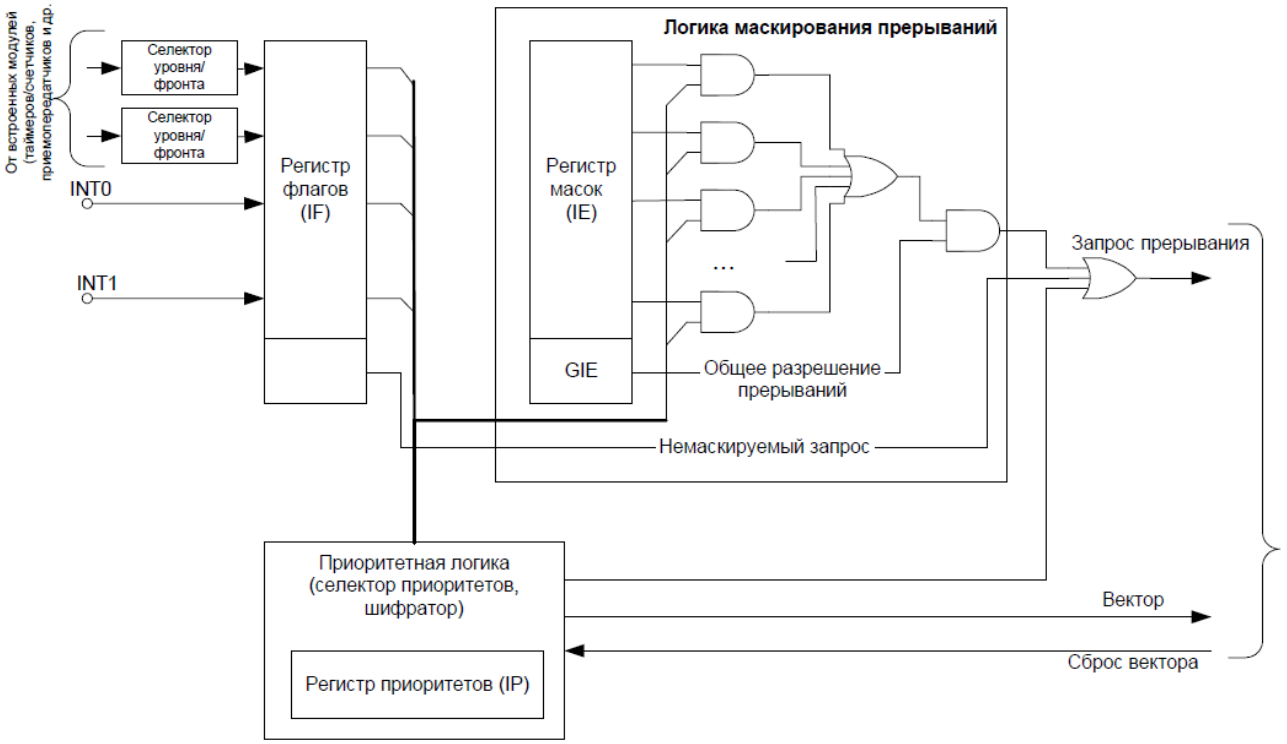


Рис. 22. Обобщенная схема блока обработки запросов прерываний

На рисунке селектор уровня/фронта внешнего сигнала запроса прерывания выбирает событие, по которому вырабатывается запрос прерывания от внешнего сигнала («внешнее прерывание»). Возможны следующие настройки: по перепаду (фронт или спад сигнала) или по уровню. При возникновении запроса прерывания в регистре флагов (Interrupt Flag, IF) устанавливается бит, соответствующий источнику. Логика маскирования прерываний разрешает или запрещает выработку запросов от определенных источников или от всех источников сразу. Для разрешения прерывания необходимо установить в «1» соответствующий бит регистра масок (Interrupt Enable, IE) и бит общего разрешения прерываний (Global Interrupt Enable, GIE). Логика маскирования никак не влияет на немаскируемый запрос прерывания (Non-Maskable Interrupt, NMI). Приоритетная логика (Interrupt Priority, IP) вырабатывает вектор для наиболее приоритетного запроса и передает его вычислительному ядру синхронно с сигналом запроса прерывания; отслеживает приоритет запроса, находящегося в обработке и вытесняет (прерывает) данных запрос, если пришел другой запрос с более высоким приоритетом.

Порог прерываний представляет собой собственный приоритет процессора, точнее, уровень приоритета выполняемой им программы, и отражается с помощью специального поля в слове состояний процессора.

В интерфейсе «общая шина» типа Unibus выделяются специальные линии запросов прерывания от ВУ и линии разрешения прерываний, которые являются однонаправленными. По линии запросов сигналы передаются от ВУ к CPU, по линии разрешения прерываний – от CPU к ВУ. (В данном случае под ВУ понимается не столько само внешнее устройство, сколько его контроллер (блок управления)). Эти линии обычно называются линиями арбитража. Для каждого уровня приоритета используются две линии.

Линии арбитража связаны со специальным блоком процессора, называемым арбитром. Основной функцией арбитра является выделение наиболее приоритетного запроса и сравнение его уровня приоритетности с собственным приоритетом CPU (порогом прерываний). В случае, если уровень приоритета запроса от ВУ оказывается ниже порога прерываний, обслуживание этого запроса откладывается до момента снижения порога прерывания до требуемого уровня.

Обработчик прерывания должен выполнять действия, связанные с появлением запроса данного типа, и поместить адрес начала этой программы в специальной таблице адресов прерываний. Программа-обработчик, как правило, должна начинаться с сохранения состояния тех регистров процессора, которые будут ею изменяться, и заканчиваться восстановлением состояния этих регистров. Программа-обработчик должна завершаться специальной командой, указывающей процессору на необходимость возврата в прерванную программу.

2.3.4 Аппаратный полинг

В случае аппаратного полинга (дейзи-цепочка, приоритетная цепочка, гирляндное или каскадное включение устройств) микропроцессор и все ПУ соединяются таким образом, что микропроцессор может осуществить автоматический запрос с целью идентификации прерывающего устройства, как показано на рисунке.

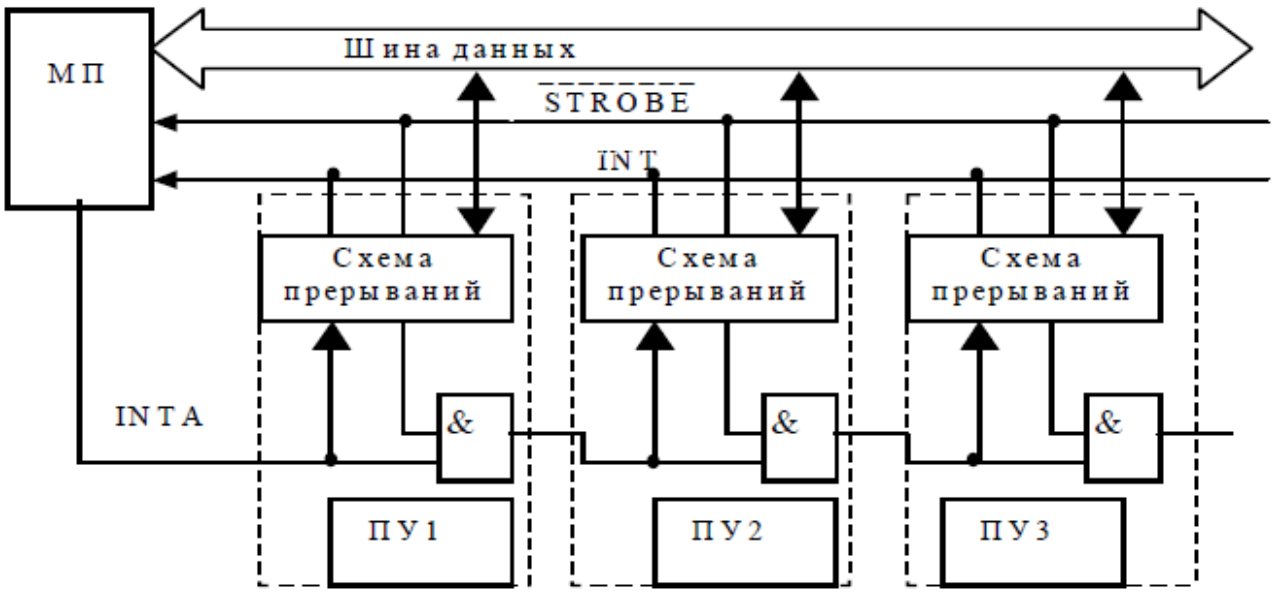


Рис. 23. Аппаратный полинг

Когда микропроцессор реагирует на запрос прерывания, он формирует сигнал подтверждения прерывания INTA, на линии, которая последовательно проходит через все устройства. При прохождении сигнала по цепочке проверяется состояние флажков готовности ПУ. Если ПУ не формирует сигнал прерывания, сигнал INTA проходит в следующее ПУ, пока не встретит активное ПУ. Это ПУ блокирует дальнейшее распространение сигнала INTA по цепочке.

Приоритет ПУ задается их физической близостью к микропроцессору по линии INTA.

Затем активное ПУ передает по шине данных свой адрес (вектор прерывания), сопровождая его импульсом STROBE. Этот адрес имеет однозначное соответствие с начальным адресом подпрограммы обслуживания прерывания данного устройства.

Время реакции микропроцессора на запрос прерывания определяется временем распространения сигнала INTA в цепочке и намного превышает времени реакции относительно программного полинга, но он требует дополнительные аппаратные средства для определения приоритета, а также для формирования адреса подпрограммы обслуживания.

Адрес (вектор прерывания), возвращаемый прерывающим устройством, обычно встраивается в интерфейсную плату, и его с помощью перемычек и переключателей может изменять пользователь. Приоритет ПУ определяется размещением его интерфейсной платы в разъеме, занимающее фиксированное положение в схеме.

2.3.5 Характеристики систем прерываний

Для оценки эффективности систем прерывания рассмотрим следующие характеристики.

Если управление сохранением состояния и возвратом возложено на саму прерывающую программу, то она состоит из трех частей: подготовительной и заключительной, обеспечивающих переключение программ, и собственно прерывающей программы, выполняющую затребованную запросом передачу информации.

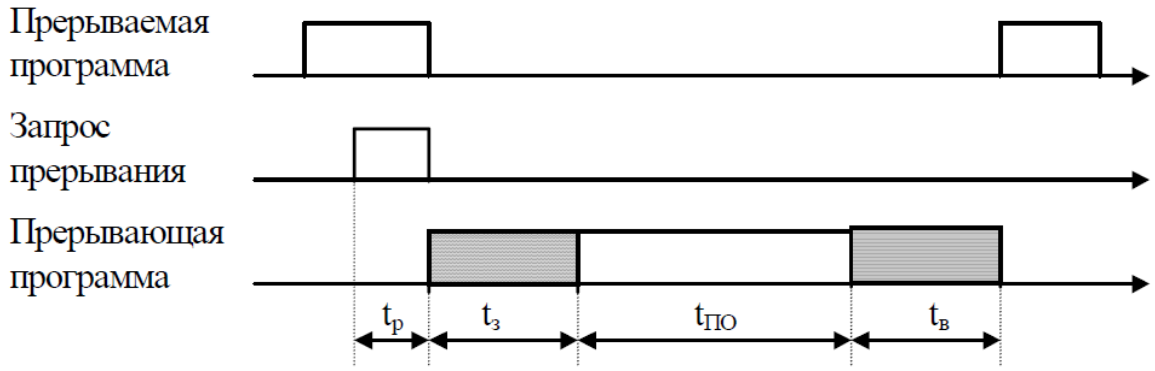


Рис. 24. Временная диаграмма процесса прерывания

tр – время реакции;

$t_з$ – сохранение состояния прерванной программы;

$t_в$ – время восстановления состояния прерванной программы;

$t_{ПО}$ – время выполнения прерывающей программы.

Время реакции $t_р$ – время между появлением сигнала прерывания и началом выполнения прерывающей программы (включая время выполнения цикла прерывания микропроцессора).

Время обслуживания $t_о$ есть сумма времени, затраченной на сохранение состояния прерванной программы, и времени на возврат к ней:

$t_о = t_з + t_в$

Глубина прерывания – максимальное число программ, которые могут прерывать друг друга. Глубина равна К, если допускается последовательное прерывание К программ. Глубина прерывания обычно совпадает с числом уровней приоритетов, распознаваемых системой прерываний.

2.3.6 Контроллер прерываний 8259A

В IBM PC-совместимых компьютерах обработка сигналов запросов прерывания выполняется контроллером прерываний (Programmable Interruption Controller, PIC), программно совместимым с микросхемой Intel 8259A. На рисунке ниже представлена типовая схема подключения контроллера 8259A к процессору.

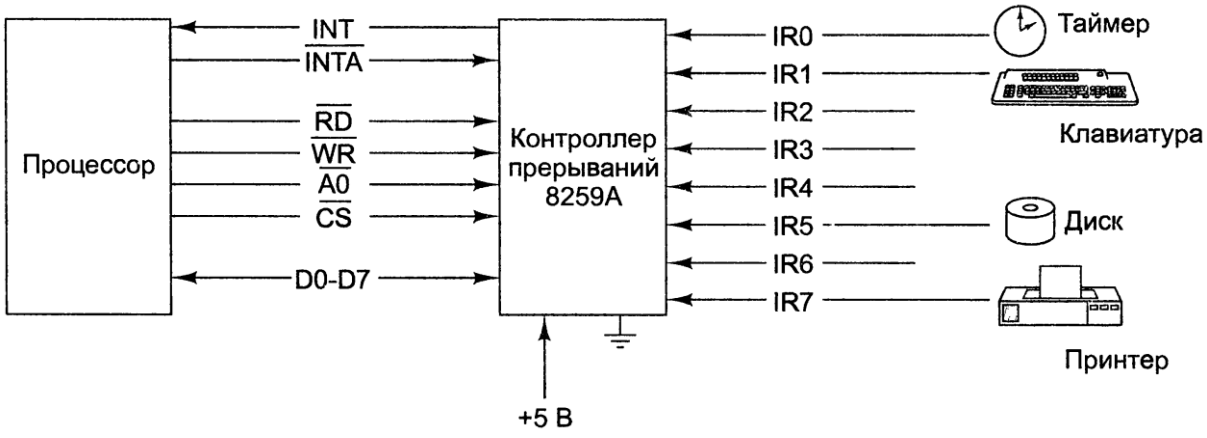


Рис. 25. Контроллер прерываний 8259A

До восьми контроллеров ввода-вывода могут быть непосредственно связаны с восемью входами IRx (Interrupt Request – запрос прерывания, IRQ) микросхемы 8259A. Когда любое из этих устройств решит выполнить прерывание, оно запускает свою линию входа. Если активизируется один или несколько входов, контроллер 8259A выдает сигнал INT (INTerrupt – прерывание), который подается на соответствующий вход центрального процессора. Если центральный процессор способен обработать прерывание, он посылает микросхеме 8259A импульс через вывод INTA (INTerrupt Acknowledge – подтверждение прерывания). В этот момент микросхема 8259A должна определить, на какой именно вход поступил сигнал прерывания. Для этого она

помещает номер входа на информационную шину. Эта операция требует особого цикла шины. Центральный процессор использует этот номер для обращения к таблице указателей, которую называют таблицей векторов прерываний, чтобы найти адрес процедуры обработки этого прерывания.

Микросхема 8259A содержит несколько регистров, которые центральный процессор может считывать и записывать, используя обычные линии шины и сигналы управления (чтение, запись и т.д.). Когда программное обеспечение обработало прерывание и готово получить следующее, оно записывает специальный код в один из регистров, который вызывает сброс сигнала INT микросхемой 8259A, если не появляется другое прерывание. Регистры также могут записываться для того, чтобы перевести микросхему 8259A в один из нескольких режимов, и для выполнения некоторых других функций.

При наличии более 8 устройств ввода-вывода микросхемы 8259A могут соединяться каскадом. В самой экстремальной ситуации все 8 входов могут быть связаны с выходами еще 8 микросхем 8259A, соединяя до 64 устройств ввода-вывода в двухступенчатую систему прерывания. Микросхема 8259A содержит несколько выводов для каскадного соединения, но мы их опустили ради простоты.

2.4 Организация обмена в режиме прямого доступа

Обмен данными микропроцессора с медленнодействующими ПУ обычно организуется по прерываниям или по программному опросу. Однако при передаче между основной и внешней памятью микро-ЭВМ больших блоков данных (десятки байт и более) производительность процессора в этих режимах является недостаточной.

Скорость передачи данных в режиме программного ввода-вывода ограничивается только процессором. Поэтому для передачи данных между внешними устройствами и ОЗУ разработан специальный метод передачи данных без участия процессора, получившего название прямого доступа к памяти (Direct Memory Access, DMA). Аппаратные средства реализации канала ПДП называются контроллером прямого доступа к памяти (КПДП).

DMA-контроллер содержит несколько регистров, доступных центральному процессору для чтения и записи. Обычно эти регистры задают порт (или канал), который должен быть использован; адрес памяти; направление переноса данных (чтение/запись); единицу переноса (побайтно/пословно); число байтов, которое следует перенести.

Необходимо отметить, что контроллер ПДП используется не только для передачи данных между УВВ и памятью, но и из памяти в память, и из УВВ в УВВ.

В идеальном случае режим ПДП совершенно не должен влиять на действия процессора, но для этого потребуется сложный и дорогой тракт в основную память вычислительной системы. Поэтому в большинстве систем

используется временное разделение (мультиплексирование) общей системной шины между процессором и КПДП.

Разработано две разновидности ПДП: режим без пропусков тактов микропроцессора и режим с пропуском тактов микропроцессора.

В первом режиме реализации прямой доступ осуществляется без участия процессора (параллельно микропроцессору). Для этого используются те интервалы машинных циклов, в течение которых микропроцессор не обращается к основной памяти. Процессор (или дополнительная схема) идентифицирует эти интервалы для КПДП специальным сигналом, означающим доступность системной шины. Производительность процессора в этом режиме не уменьшается, но для каждого типа процессора потребуется свой контроллер ПДП. С другой стороны, сами передачи будут носить нерегулярный характер ввиду отсутствия у некоторых команд этих интервалов, что приведет к уменьшению скорости передачи данных в режиме ПДП.

Во втором способе реализации КПДП полностью «захватывает» системную шину на время передачи, при этом процессор отключается от системной шины и переходит в режим холостого хода. Таким образом, передачи ПДП осуществляются путем пропуска тактов процессора в выполняемой программе. При выполнении передач ПДП содержимое внутренних регистров процессора не модифицируются, поэтому его не нужно запоминать в памяти, а затем восстанавливать, как при обработке прерываний. Выполнение программы осуществляется сразу после окончания ПДП. Тем не менее, в условиях интенсивных передач ПДП эффективная производительность процессора уменьшается.

Аппаратная реализация каналов ПДП определятся особенностями ЭВМ и устройств внешней памяти, но можно сформулировать общие принципы работы каналов ПДП.

2.4.1 Общие принципы организации ПДП

Центральный процессор программирует контроллер DMA, устанавливая его регистры. Затем процессор дает команду устройству (например, АЦП) прочитать данные во внутренний буфер. DMA-контроллер начинает работу, посылая устройству запрос чтения (при этом устройство даже не знает, пришёл ли запрос от процессора или от контроллера DMA). Адрес памяти уже находится на адресной шине, так что устройство знает, куда следует переслать следующее слово из своего внутреннего буфера. Когда запись закончена, устройство посылает сигнал подтверждения контроллеру DMA. Затем контроллер увеличивает используемый адрес памяти и уменьшает значение своего счётчика байтов. После чего запрос чтения повторяется, пока значение счётчика не станет равно нулю. По завершении цикла копирования устройство инициирует прерывание процессора, означающее завершение переноса данных. Контроллер может быть многоканальным, способным параллельно выполнять несколько операций.

Для осуществления режима ПДП контроллер должен выполнить ряд последовательных операций для передачи данных в этом режиме, называемых также циклами ПДП:

1. Принять запрос на ПДП от ПУ (DMA Request, DREQ);
2. Сформировать запрос процессору для перехода в режим ПДП (Hold Request, HRQ).
3. Принять сигнал (Hold Acknowledge, HLDA), подтверждающий переход процессора в режим ПДП (ШД, ША, ШУ в z-состояние), т. е. переход в режим ПДП.
4. Сформировать сигнал (DMA Acknowledge, DACK), сообщающий ПУ о начале выполнения циклов ПДП.
5. Сформировать на ША адрес ячейки памяти, предназначенной для обмена.
6. Выработать сигналы чтения из памяти, записи в ПУ (MEMR1, IOW) и чтение из ПУ, запись в память (IOR, MEMW), обеспечивающие управление обменом.
7. По окончании ПДП либо повторить цикл ПДП, изменив адрес, либо прекратить ПДП, сняв запросы ПДП.

На рисунке показана структурная схема микропроцессорной системы с контроллером ПДП. На рисунке сигнал MEMR назван MR, а сигнал MEMW – MW.

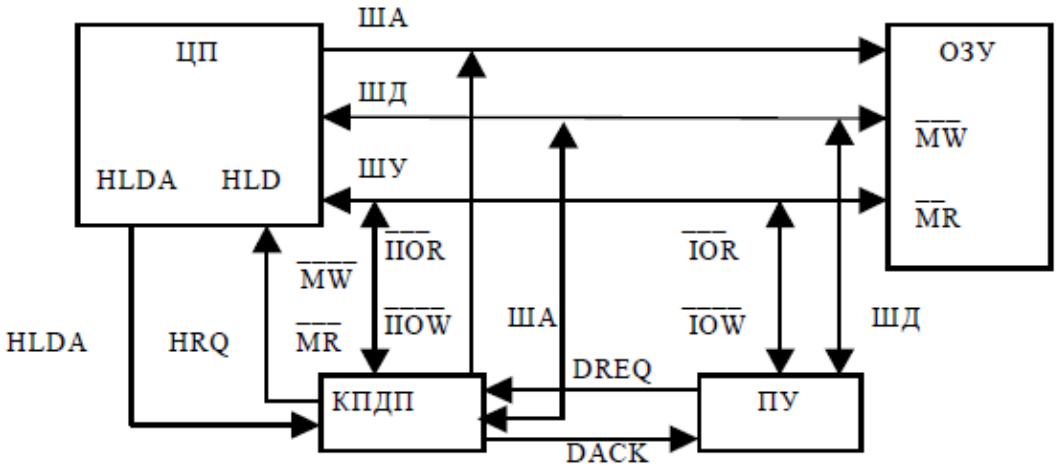


Рис. 26. Структурная схема МПС с контроллером ПДП

Циклы ПДП выполняются с последовательно расположенными ячейками памяти, поэтому КПДП должен иметь счетчик адреса ОЗУ. Число циклов ПДП определяется специальным счетчиком. Управление обменом осуществляется специальной логической схемой, формирующей в зависимости от типа обмена пары управляющих сигналов: MEMR, IOW (циклы чтения) и IOR, MEMW (циклы записи).

Из изложенного следует, что контроллер ПДП по запросу (DREQ) от устройства внешней памяти ПУ должен взять на себя управление системными шинами и выполнять совмещенные циклы чтения или записи до тех пор, пока

содержимое счетчика циклов ПДП не будет равно нулю. После этого, устройство внешней памяти снимает запрос ПДП (DREQ), что приводит к снятию соответствующего запроса в процессор (HRQ), и он возобновляет приостановленную программу.

2.4.2 DMA-контроллеры персонального компьютера

В оригинальной архитектуре IBM PC реализация режима ПДП была возможна при наличии аппаратного DMA-контроллера – микросхемы Intel 8237. Эта микросхема могла управлять четырьмя "каналами", каждый из которых связан со своим набором регистров DMA. Таким образом, четыре устройства могли сохранять свою DMA информацию в контроллере одновременно. Более поздние PC содержали эквивалент двух DMA-контроллеров: второй контроллер (master) подключен к системному процессору, а первый (slave) подсоединен к каналу 0 второго контроллера. Сейчас эти чипы являются частью чипсета, на котором строится материнская плата, но еще несколько лет назад это были два раздельных чипа 8237. Оригинальная архитектура PC имела только один DMA контроллер. Второй был добавлен в архитектуру PC на платформах i286. Однако второй контроллер был установлен как master, так как он управлял 16-битовой передачей, в то время как первый контроллер управлял 8-ью битами передачи и был оставлен для обратной совместимости.

Интерфейс КПДП 8237:

- HLDA (Hold Acknowledge) – входной сигнал подтверждения процессором освобождения системной шины (ШД, ША, ШУ в z-состояние), т. е. переход в режим ПДП.
- DREQ (DMA Request) – запрос на ПДП от ПУ.
- IOR (I/O Read) – входной сигнал управления, который используется процессором для чтения регистров управления контроллера ПДП, когда КПДП находится в режиме ожидания (Idle). В режиме ПДП этот сигнал является выходным и используется КПДП для чтения данных из ПУ.
- IOW (I/O Write) – входной сигнал управления, который используется процессором для конфигурирования регистров управления контроллера ПДП, когда КПДП находится в режиме ожидания (Idle). В режиме ПДП этот сигнал является выходным и используется КПДП для записи данных в ПУ.
- HRQ (Hold Request) – запрос от КПДП процессору на переход в режим ПДП, т. е. на освобождение системной шины.
- DACK (DMA Acknowledge) – сигнал подтверждения КПДП перехода в режим ПДП для определенного ПУ (которое прежде послало DREQ).
- MEMR (Memory Read) – сигнал чтения данных из памяти в режиме ПДП.

- MEMW (Memory Write) – сигнал записи данных в память в режиме ПДП.
- A0-A3 – младшие разряды шины адреса. В режиме ожидания эти разряды (входной сигнал) используются для адресации к регистрам управления КПДП. В режиме ПДП (выходной сигнал) служат 4 младшими разрядами адреса обращения к памяти.
- A4-A7 – старшие разряды шины адреса. Используются только в режиме ПДП (выходной сигнал) в качестве 4 старших разрядов адреса обращения к памяти.
- D0-D7 – шина данных. В режиме ожидания используется для конфигурирования регистров КПДП. В режиме ПДП – для передачи данных между ОП и ПУ.

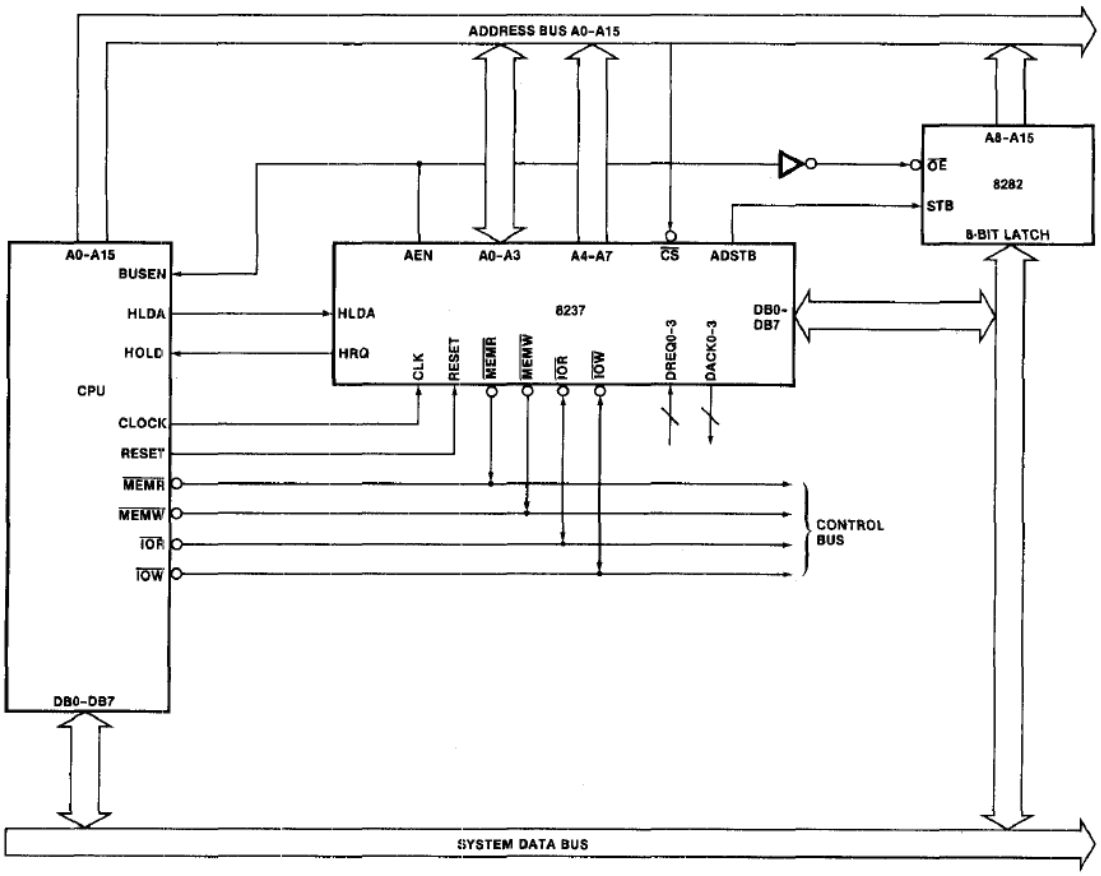


Рис. 27. Системный интерфейс контроллера ПДП 8237

Каналы пронумерованы от 0 до 7. Канал номер 4 не доступен периферии ISA, потому что он используется для внутренних целей каскадирования slave-контроллера к master-контроллеру. Доступные каналы имеют номера 0..3 на slave-контроллере для 8-ми битовой передачи, и номера 5..7 на master-контроллере для 16-битовой передачи. Обратите внимание, что 4-ый канал общей нумерации соответствует 0-му каналу master-контроллера. Размер DMA-передачи сохраняется в контроллере как 16-ти битовое число, и представляет собой число циклов шины необходимое для завершения передачи требуемого

количества данных. Таким образом, максимальный размер передачи для slave-контроллера равен 64Кб, и 128Кб для master-контроллера.

Стандартные каналы КПДП 8237 (архитектура ISA):

- Канал 2 – контроллер НГМД,
- Канал 3 – контроллер НЖМД,
- Канал 4 – каскадирование и др.

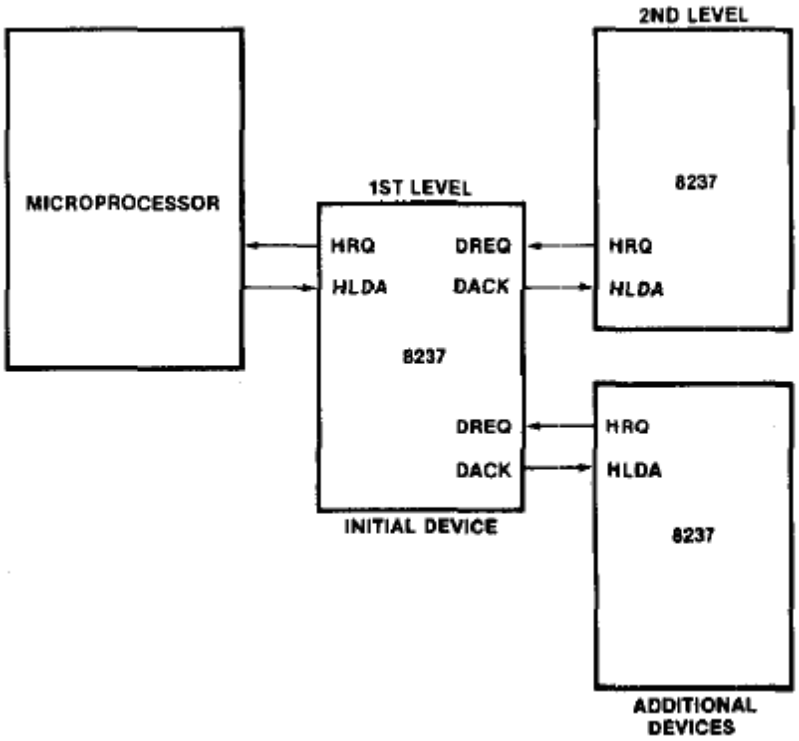


Рис. 28. Каскадирование контроллеров ПДП 8237

В архитектура IBM PC с шиной PCI как такового централизованного КПДП (микросхема 8237) нет, а есть режим прямого управления шиной (bus mastering) и КВУ со встроенными КПДП и возможностью работы в соответствующем режиме. В режиме bus mastering любому устройству возможно заявить о возникновении потребности к захвату шины, каковая потребность удовлетворяется так называемым арбитром при первой возможности. Устройство, успешно осуществившее захват шины, самостоятельно выставляет на шину сигналы адреса и управления, и исполняет в течение какого-то времени ту же ведущую роль на шине, что и ЦП. Доступ ЦП к шине при этом кратковременно блокируется.

3 Аппаратные интерфейсы вычислительных систем

3.1 Характеристики аппаратных интерфейсов

Аппаратные интерфейсы определяются двумя уровнями описания: логическая и физическая организация. **Логическая организация**: группы взаимодействующих объектов, характер взаимодействия, адресное пространство, система команд, информация о состоянии объектов, фазы в работе интерфейса, форматы данных, набор процедур по реализации взаимодействия и последовательность их выполнения для различных режимов функционирования. **Физическая организация** интерфейса определяется электрической и конструктивной совместимостью сопрягаемых устройств.

Под **электрической совместимостью** понимается согласованность статических и динамических параметров электрических сигналов в системе соединительных линий интерфейса с учетом ограничений на пространственное размещение частей интерфейса и техническую реализацию приемопередающих блоков интерфейса. Условия электрической совместимости определяют [63]:

- Тип приемопередающих элементов.
- Состав линий и схема их согласования.
- Соотношения между логическими и электрическими состояниями сигналов и пределы их изменения.
- Коэффициенты нагрузочной способности приемопередающих элементов и значения допустимой емкостной и резистивной нагрузки линии в устройстве.
- Допустимую длину линии и порядок подключения линий к соединительным элементам (разъемам).
- Требования к источникам и цепям электрического питания.
- Требования по помехоустойчивости и заземлению.

Конструктивная совместимость – согласованность конструктивных элементов интерфейса, предназначенных для обеспечения механического контакта электрических соединений и механической замены схемных элементов, блоков и устройств [63].

Условия совместимости определяют:

- Типы соединительных элементов (разъем, штекер, распределение соединительных линий внутри соединительного элемента).
- Конструкцию платы, каркаса, стойки.
- Конструкцию кабельного соединения.

К основным характеристикам аппаратных интерфейсов относятся:

1. Скорость передачи (пропускная способность, производительность).
2. Протяжённость.
3. Тип сопрягаемых устройств вычислительной системы (см. следующий раздел).
4. Топология.
5. Разрядность слова данных (последовательный или параллельный интерфейс).
6. Синхронный или асинхронный интерфейс.
7. Симплексный, полудуплексный, дуплексный обмен.

Производительность оценивается количеством информации (полезной), передаваемой в секунду. Избыточная информация может достигать 90%. Производительность связана с понятием тактовой частоты. Также на неё влияет разрядность шины данных.

Протяжённость связана и влияет на производительность интерфейса, определяется типом сопрягаемых устройств вычислительной системы.

По **топологии** выделяют:

- Радиальные интерфейсы.
- Шинные интерфейсы (моноканал).
- Цепочечные интерфейсы.
- Кольцо.
- Интерфейсы со сложной топологией (каждый с каждым, произвольная топология, гиперкуб и т.д.).

При **радиальной топологии** происходит соединение двух устройств (соединение типа «точка-точка») и более (топология «звезда»). При этом способе соединения устройств вычислительной системы имеется главный модуль, с которым связаны все остальные, и они могут взаимодействовать между собой только через главный модуль, что также снижает производительность, но меньше, чем при магистральном способе. Такой способ соединения эффективен для соединения модулей, которые в основном работают только с главным. Достоинство – простота каналов и протоколов связи каждого модуля (RS-232, Centronics), что позволяет увеличивать их длину намного больше, чем у магистральных. Радиальный способ позволяет к одному разъему подключать всего одно ПУ или, при соответствующей организации канала, несколько ПУ. Этот способ также нашел широкое применение в компьютерах.

Топология моноканал (общая шина) используется при соединении двух и более устройств. В такой топологии требуется система адресации и синхронизации (арбитраж, система доступа). Достоинства: возможность расширения (простая) теоретически до бесконечности, практически

ограничивается системой адресации и характеристиками электрических элементов, экономия ресурсов технических средств, в первую очередь, линий связи. Недостатки: невысокая производительность, так как все модули делят магистраль между собой, а высокоскоростные магистрали все же дороги и могут быть реализованы только при их ограниченной длине.

Цепочечное соединение обладает рядом полезных свойств: можно использовать узлы как усилители и тем самым увеличивать суммарную протяжённость интерфейса; можно обеспечить параллельную работу не перекрывающихся сегментов. Часто используются комбинации моноканала и цепочки. К недостаткам цепочечных интерфейсов можно отнести замедление передачи сигнала по мере увеличения количества узлов.

Кольцевой интерфейс является гибридом шинной топологии и цепочечной. Совмещает в себе достоинства шины и цепочки. За счет дополнительного пути обхода обладает более высокой надежностью, чем шина.

Интерфейсы со сложной топологией используются тогда, когда есть необходимость в дополнительных (резервных) каналах связи с целью увеличения надежности или производительности. Интерфейсы со сложной топологией можно встретить в глобальных сетях (Интернет), в многопроцессорных вычислительных системах (суперЭВМ) и в беспроводных сенсорных сетях. В интерфейсах со сложной топологией на первое место выходит проблема маршрутизации сообщений и организация системы приоритетов.

Одной из характеристик аппаратных интерфейсов является **разрядность слова данных**, которая позволяет делить интерфейсы на последовательные, последовательно-параллельные и параллельные. От этой характеристики зависит стоимость аппаратуры и кабельного соединения, а также производительность интерфейса, его помехозащищенность. Последовательный интерфейс предполагает для передачи данных в одном направлении единственную сигнальную линию, по которой информационные биты передаются друг за другом последовательно. При этом скорость изменения передатчиком состояния линии должна равняться скорости распознавания состояний приемником. Эта скорость измеряется в бодах (baud) – количестве изменений состояния линии за одну секунду. В простейшем случае в линии имеется всего два состояния сигнала, т.е. одним состоянием кодируется один бит, и тогда скорость изменения состояния в бодах совпадает со скоростью передачи двоичной информации, определяемой количеством передаваемых за секунду бит информации, bps (bits per second, бит/с). Однако, при использовании других методов модуляции возможны несколько состояний сигнала, что позволяет одним состоянием кодировать сразу несколько передаваемых бит, и здесь скорость передачи данных bps превышает скорость изменения сигнала baud. Примеры последовательных интерфейсов: RS-232, SPI, I^2C.

В параллельном интерфейсе для передачи данных в одном направлении используется несколько линий (8, 16, 24, 32, 64). Примеры параллельных интерфейсов: ISA, ATA, SCSI, PCI, IEEE 1284/Centronics. С понятием параллельного интерфейса соседствуют такие понятия, как шина и магистраль.

Шина – совокупность линий, сгруппированных по функциональному назначению (например, шина адреса, шина данных и т.д.).

Магистраль – совокупность всех линий аппаратного интерфейса. Выделяются две магистрали: информационного канала и управления информационным каналом. По информационной магистрали передаются коды адресов, команд, данных, состояния. Аналогичные наименования имеют соответствующие шины интерфейса.

Шины адреса предназначены для выборки в магистрали узлов устройства, ячеек памяти. Для логической адресации в основном используется двоичный код. В некоторых интерфейсах применяется позиционное или географическое кодирование, при котором каждой позиции (месту) выделяется отдельная линия выборки. В этом случае используется термин «географическая адресация».

Шины данных используются для передачи в основном двоичных кодов. Как правило, в параллельных интерфейсах шины данных кратны байту (8, 16, 24, 32 разряда).

Шины состояния используются для передачи сообщений, описывающих результат операции на интерфейсе или состояния устройств сопряжения. Коды формируются в ответ на действие команд или отображают состояние функционирования устройств, таких как готовность, занятость, наличие ошибки и т. д. В наиболее стандартизованных интерфейсах разряды состояния унифицированы для любых типов устройств, в других – носят рекомендательный характер или отсутствуют.

В большинстве параллельных интерфейсов коды адресов, данных, команд, состояний передаются по шинам интерфейса в режиме временного мультиплексирования сигнала по одним и тем же линиям с использованием дополнительных линий идентификации типа передаваемой информации. В таком случае их называют последовательно-параллельными интерфейсами. При этом существенно сокращается число линий информационной магистрали, однако происходит снижение быстродействия передачи информации.

Магистраль управления информационным каналом по функциональному назначению делится на следующие шины: управления обменом, передачи управления, прерывания, управления режимом работы, специальных сигналов.

Шина управления включает в себя линии синхронизации передачи информации. В зависимости от используемого принципа обмена (синхронного, асинхронного) число линий может меняться. Кроме того, данная шина используется для управления операциями на магистрали. По функциональному назначению различают следующие команды: адресации, управления обменом информацией, изменения состояния и режимов работы. Адресные команды

используются для задания режимов адресации: вторичной, широковещательной, групповой и т.п. Наиболее распространенными командами являются: чтение, запись, конец передачи, запуск.

Шины передачи управления используется для реализации операций приоритетного занятия магистрали информационного канала (арбитража ресурсов шины).

Шина прерывания применяется в основном в системных интерфейсах. Устройство идентифицируется либо адресом источника прерывания, либо адресом программы обслуживания прерывания, так называемым вектором прерывания.

Шины управления режимом работы и специальных управляющих сигналов содержат линии, обеспечивающие работоспособность интерфейса, в том числе приведение устройств в исходное состояние, контроль источников питания, контроль и службу времени и т. п.

Важнейшим моментом в работе аппаратных интерфейсов является **синхронизация** передачи информации.

Синхронизация – это согласование процессов взаимодействия между устройствами, заключающееся в передаче информации источником и ее приема приемником (одним или несколькими).

Существуют два основных режима синхронизации: синхронный и асинхронный.

В **синхронном** режиме для синхронизации используют специальную линию для передачи тактовых импульсов. Информация в канале данных считывается приемником только в те моменты, когда на линии синхронизации сигнал активный. Таким образом, смена состояний источника и приемника взаимонезависимы и выполняются через одинаковые фиксированные интервалы времени. В этом случае приемник должен успеть принять данные до момента времени, когда источник выставит новые данные. Величина фиксированного интервала времени синхронизации (τ) определяется суммой времен (T): распространения сигнала в линии связи, распознавания его приемником и временем фиксации данных в приемнике. Если источник взаимодействует с разными приемниками, то его частота работы определяется частотой работы самого медленного устройства, включая сам источник, что естественно замедляет общий процесс передачи данных. Для надежной передачи данных необходимо выполнение условия $\tau > T_{max}$, где τ – это период следования импульсов синхронизации, задающих моменты переключения сигналов в приемнике и источнике. Реализация контроллеров синхронного аппаратного интерфейса упрощается по сравнению с другими. Второе преимущество состоит в том, что удаётся максимально использовать скорость передачи аппаратуры. Повышение производительности происходит из-за

отсутствия необходимости обратной связи (подтверждения). Недостаток – отсутствие гибкости, необходимость использования дополнительных сигналов.

При **асинхронном** режиме смена состояний источника и приемника взаимозависимы, момент времени изменения состояния источника зависит от момента времени, когда приемник зафиксирует данные. Асинхронный режим реализуется с помощью обратной связи от приемника к источнику (рукопожатие, сигналы квитирования, подтверждения). В асинхронном последовательном интерфейсе RS-232 посылке очередного байта информации предшествует специальный старт-бит, сигнализирующий о начале передачи (обычно логический «0»). Затем следуют биты данных (их обычно 8), за которыми может следовать дополнительный бит (его наличие зависит от режима передачи, обычно этот бит выполняет функцию контроля четности). Завершается посылка стоп-битом (логическая «1»), длина которого (длительность единичного состояния линии) может соответствовать длительности передачи 1, 1.5 («полтора стоп-бита») или 2 бита. Стоп-бит гарантирует некоторую выдержку между соседними посылками, при этом пауза между ними может быть сколь угодно долгой (без учета понятия «тайм-аута»). Оптимальная загрузка сопрягающихся устройств.

Асинхронный режим при работе источника со многими приемниками, имеющими широкий диапазон скоростей передачи информации, обеспечивает большую общую пропускную способность, чем синхронный. Асинхронный режим не означает, что синхронизация отсутствует, при асинхронном принципе период синхронизации является переменным, а при синхронном этот период постоянен и определяется частотой работы самого медленного устройства. В асинхронных интерфейсах переключение сигналов на линиях шины не привязана к фронтам импульсов синхронизации. Эта шина позволяет гораздо проще приспособить широкое разнообразие устройств и удлинить шину без беспокойства о перекосе сигналов синхронизации и о системе синхронизации. Асинхронная шина легче масштабируется.

Если шина синхронная, то переключение всех сигналов на линиях шины производится по фронту импульсов синхронизации. Эти шины могут быть быстрыми и дешевыми. Но частота работы параллельных интерфейсов ограничивается из-за проблем перекоса сигналов на разных линиях шины, что накладывает серьезные ограничения на длину этих шин. Обычно внутрисистемные шины синхронные, системные шины (расширения) могут быть синхронные (PCI) или асинхронные (ISA). В последовательных интерфейсах такие ограничения существенно меньше.

Изохронные интерфейсы – компромиссный вариант. Работают по принципу асинхронной передачи, но все переключения привязаны к тактовой сетке (т.е. варьируются дискретно). Пример: I^2C.

Следует отметить, что в синхронных и асинхронных аппаратных интерфейсах могут быть как синхронные, так и асинхронные способы передачи данных (программно управляемые, ПДП).

Обмен данными может быть:

- Дуплексным: одновременный прием и передача данных.
- Полудуплексным: данные передаются в одном направлении с возможностью смены направления.
- Симплексным: данные передаются только в одном направлении.

3.2 Функции аппаратных интерфейсов

В настоящее время интерфейсы выполняют следующие основные функции [63]:

1. Проведение синхронизации интерфейса, используя синхронный или асинхронный принципы.
2. Передачу информации между источником и приемником с помощью операций чтения и записи.
3. Арбитраж активных устройств на шине и селекция ПУ при вводе-выводе в режимах прямого доступа к памяти и прерываний.
4. Контроль передачи информации и функционирования самой шины и устройств на ней.
5. Преобразование информации из параллельного в последовательное представление и обратно.
6. Поддержку режима автоконфигурации.
7. Управление питанием компьютера.
8. Поддержку режима горячего подключения ПУ к системному блоку.

Функция синхронизации

Синхронизация является той функцией, которая определяет скорость и надежность передачи информации. Функция синхронизации реализуется либо по синхронному, либо по асинхронному принципу, используя аппаратные или программные средства. При аппаратной синхронизации она осуществляется с помощью специальных сигналов синхронизации (синхронизирующих импульсов, сигналов стробирования). Программная синхронизация использует специальные маркеры и метки, представляющие собой либо коды синхронизации, либо пакеты-маркеры, содержащие соответствующую информацию.

Функция передачи информации

Передача информации осуществляется в режиме программно управляемого ввода-вывода или прямого доступа к памяти.

Передача информации между источником и приемником выполняется в виде циклов (команд) шины. Обычно используется четыре типа циклов обмена: циклы памяти, циклы ввода-вывода, циклы прямого доступа к памяти и цикла автоконфигурации. Управление передачей осуществляет активное устройство.

В процессорах с архитектурой x86 (Intel) активное устройство совместно с адресом выдает команду обмена данными. Эти команды различают пространства памяти, ввода-вывода и автоконфигурации. Инструкции ввода-вывода процессора порождают шинные циклы обмена, в которых вырабатываются сигналы IORD (Input-Output read, чтение порта) и IOWR (Input-Output write, запись в порт), которые отличают пространства ввода-вывода от пространства памяти, где вырабатывают соответствующие сигналы чтения и записи MEMRD (Memory Read, чтение памяти) и MEMWR (Memory Write, запись в память). В цикле обмена участвуют сигналы стробирования и квитирования. В случае кодирования команд в виде сигналов на линиях RD (Read) и WR (Write), они являются многофункциональными и обычно указывают направление передачи, адресное пространство и выполняют функции стробирования.

Перед передачей данных активное устройство указывает номера байт, передаваемых в цикле обмена, с помощью специальных сигналов и кодов.

Сигналы IORD, MEMWR и IOWR, MEMRD вырабатываются и в циклах прямого доступа к памяти. В этом случае активным является контроллер прямого доступа к памяти (КПДП), он выдает на шину адрес памяти, к которой производится доступ, а адрес порта не выдается.

Функции арбитража и селекции

Функции арбитража и селекции используются для выбора устройств с наибольшим приоритетом и предоставления им прав работать на шине. Эти функции обслуживают режим работы на шине нескольких активных устройств и ввод-вывод в режиме прерываний и прямого доступа к памяти.

Функция селекции при вводе-выводе в режиме прерываний включает также процесс идентификации периферийного устройства, получившего право работать с активным устройством.

Если на магистрали несколько устройств и они должны делить общие ресурсы, то надо решить задачу арбитража.

Виды арбитража: централизованный и децентрализованный [63].

Таблица 1. Централизованный арбитраж.

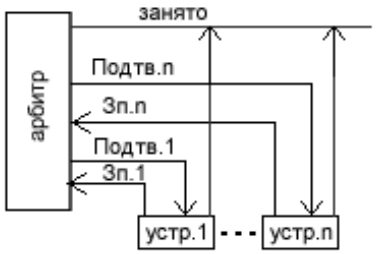
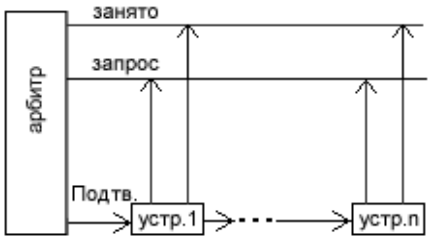
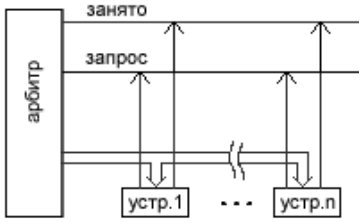
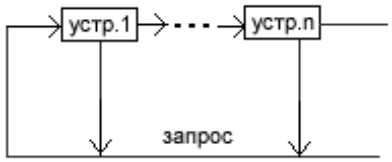
Схема 1	Схема 2	Схема 3
Арбитр анализирует индивидуальные линии запросов Зп.1..Зп.n и посылает сигнал Подтв. i с учетом приписанных устройствам приоритетов.	Сигнал «Подтверждение» последовательно переходит по всем устройствам.	Арбитр опрашивает устройства в соответствии с приоритетами. Для опроса готовности устройства используется вспомогательная линия или шина.
		

Таблица 2. Децентрализованный арбитраж.

Схема 1	Схема 2
Сигнал «Запрос», сформированный схемой ИЛИ из индивидуальных линий «Запрос», подается входным сигналом «Подтверждение» последовательно на Устройство1 и последующие Устройства, если предыдущему Устройству информационный канал не нужен.	Используется одна линия, определяющая состояние занятости информационного канала, по перемещению в линии маркерного импульса или серии импульсов. Устройство, запрашивающее информационный канал, не пропускает маркер к следующему Устройству.
	

В архитектуре процессоров x86 арбитраж реализуется с помощью специальных схем арбитра в главном мосту или с помощью котроллеров прямого доступа к памяти, построенных на микросхемах типа i8237A, имеющих 4 линии запросов ПДП. Функция селекции использует контроллеры прерываний типа i8259A с 8 линиями запросов прерывания (IRQi).

Функция контроля

Эта функция используется для контроля передачи адреса и данных, контроля выдачи сигналов обратной связи (квитирования) и улучшения ремонтопригодности компьютера при локализации неисправностей.

При параллельной передаче адрес и данные контролируются методом проверки на четность (нечетность) ECC, для чего вводится специальная линия контрольного разряда.

При последовательной передаче, как правило, используется избыточные циклические коды (метод CRC) и каждый блок данных сопровождается контрольным кодом.

Контроль выполнения циклов на шине осуществляется методом тайм-аут. При этом методе для каждого контролируемого цикла задается максимально возможное время длительности цикла, если цикл не завершается за это время, выдается сигнал ошибки.

Для проведения диагностики работоспособности устройств в интерфейсах используется специальная последовательная шина JTAG, предназначенная для тестирования PCI-устройств с помощью встроенного порта TAP (Test Access Port).

Функция преобразования информации

В компьютерах используются одновременно и параллельные, и последовательные интерфейсы, кроме того, применяются ПУ с последовательной записью и считыванием информации на носителе. Все это приводит к необходимости при передаче информации производить преобразование последовательного ее представления в параллельное и наоборот. Эти функции реализуются в соответствующих контроллерах ввода-вывода.

Функция автоконфигурации

Эта функция в интерфейсе реализуется специальными операциями конфигурационного чтения и записи (Configuration Read and Write), сигналами выбора устройств при конфигурации и выделенным адресным пространством автоконфигурации. Сигналы выбора являются индивидуальными для каждого устройства. С их помощью производится последовательная выборка устройств шины, подлежащих автоконфигурации. Конфигурируемые устройства сообщают блоку автоконфигурации о потребностях в ресурсах и возможных диапазонах памяти, эти данные хранятся в регистрах автоконфигурации. После распределения ресурсов, выполняемого программой конфигурирования, в устройство передаются параметры конфигурирования, которые записываются в пространство памяти автоконфигурации, расположенной в самих устройствах. ПУ, использующие автоконфигурацию, должны иметь соответствующие средства для проведения этих процедур.

Функция управлением питанием (Power Management)

В настоящее время многие компьютеры круглосуточно включены и работают. Поэтому в интерфейсах вводят специальные функции управления электропотреблением, работающие в компьютерах общего назначения в соответствии со спецификациями ACPI и PC97. В микропроцессорной технике

для специализированных применений есть аппаратные и программные решения по управлению питанием, энергопотреблением.

Функция горячего подключения ПУ

Эта функция позволяет отключать и подключать ПУ без остановки компьютера. При этом происходит автоконфигурирование включенного устройства без участия оператора.

3.3 Классификация аппаратных интерфейсов

По типу сопрягаемых объектов можно выделить следующие группы аппаратных интерфейсов:

1. Внутрисистемные интерфейсы.
2. Системные интерфейсы.
3. Стандартные периферийные интерфейсы.
4. Малые периферийные интерфейсы.
5. Интерфейсы систем передачи данных:
   - Магистральные интерфейсы для WAN.
   - Интерфейсы локальных вычислительных сетей.
6. Интерфейсы распределенных систем управления.

Внутрисистемный интерфейс является группой интерфейсов, которая обеспечивает взаимодействие элементов ядра вычислительной системы и должно удовлетворять критерию максимальной производительности.

Системный интерфейс служит для развития системы, т.е. наращивания характеристик ядра (например, ISA, PC-104, PCI, ASB). Является компромиссом при создании дешевой вычислительной структуры.

Стандартный периферийный интерфейс характерен только для систем ввода-вывода и позволяет объединить процессор ввода-вывода и контроллер ввода-вывода. Характеристики стандартного интерфейса отличаются от предыдущих 2-х групп: критерием является удобство и эффективность управления большим числом периферийных устройств.

Малые интерфейсы ввода-вывода объединяют контроллер ввода-вывода с внешним устройством, из этого вытекают особенности организации этого интерфейса. Для каждого внешнего устройства требуется свой оптимальный интерфейс.

Интерфейсы систем передачи данных предназначены для организации взаимодействия вычислительных систем общего назначения. По степени территориального охвата можно выделить локальные, городские и глобальные сети (например, Интернет). В зависимости от места применения меняются требования к интерфейсам. Магистральные интерфейсы, на базе которых реализована связь между странами и континентами организована так, чтобы за

единицу времени проходило максимальное количество информации. В таких интерфейсах превалирует пакетная передача данных, позволяющая наиболее эффективно использовать дорогие каналы связи. В локальных сетях основной целью является обеспечение достаточной пропускной способности, при которой у любого пользователя создается иллюзия монопольного использования ресурсов сети.

Интерфейсы локальных вычислительных сетей являются подмножеством интерфейсов систем передачи данных и предназначены для объединения вычислительных устройств, территориально сосредоточенных на сравнительно небольшой площади (десятки – сотни метров в поперечнике). Как правило, к локальным вычислительным сетям относят офисные сети, фрагменты корпоративных сетей (см. пирамиду автоматизации). Основная цель локальных вычислительных сетей – объединение разрозненных компьютеров общего назначения с целью передачи сравнительно больших объемов данных с высокой скоростью. В качестве примеров можно привести Fast Ethernet, Gigabit Ethernet, Wi-Fi.

Интерфейсы распределенных систем управления объединяют контроллеры с целью организации управления и сбора телеметрической информации с большого количества узлов сети. На первое место в таких системах выдвигается не объем и скорость передаваемой информации, а надёжность и реальное время. Как правило, в распределенных системах управления объем передаваемых данных и скорости передачи сравнительно небольшие.

По среде передачи выделяют:

- Проводные интерфейсы.
- Оптические интерфейсы.
- Беспроводные интерфейсы.
- Акустические интерфейсы.

Проводные интерфейсы используют в качестве носителя информации электрический сигнал. В настоящий момент это наиболее распространенный тип интерфейса. К недостаткам интерфейса можно отнести высокую стоимость материалов.

Оптические интерфейсы используют для передачи информации свет. Свет может передаваться как по специальным световодам (оптоволокну), так и в воздушной или безвоздушной среде, в пределах прямой видимости. Использование оптоволокна позволяет сравнительно недорого осуществлять связь на большие расстояния. Как правило, оптические интерфейсы используют в сетях передачи данных при организации магистральных каналов связи или в системах, работающих в условиях помех (например, для передачи Hi-Fi звука в музыкальных центрах).

Акустический интерфейс предполагает использование звука для передачи данных в слышимом (20Гц..20кГц) или ультразвуковом диапазонах. Этот интерфейс характеризуется низкой скоростью передачи и используется там, где по каким либо причинам невозможно использование обычных каналов связи (например, под водой).

Кроме того, аппаратные интерфейсы можно классифицировать по характеристикам, перечисленным в предыдущем разделе.

3.4 Понятие интерфейсных систем

Интерфейсная система это совокупность нескольких интерфейсов, обладающих различной функциональностью и объединенных в рамках единой архитектуры. Главное преимущество таких систем – концептуальная целостность, позволяющая строить сложные компьютерные системы в рамках некоего отлаженного и единого для разных разработчиков шаблона.

Примером современной интерфейсной системы можно назвать систему AMBA, объединяющую в себе внутрисистемный, системный и периферийный интерфейсы, используемые при построении СнК на базе компонентов фирмы ARM. Еще одним хорошим примером интерфейсной системы является Multibus фирмы Intel Corp., состоящая из системной шины (Multibus System Bus), шины расширения ввода-вывода (iSBX) , исполнительской шины (iLBX) и шины многоканального ввода-вывода (Multichannel I/O Bus). Эта интерфейсная система была спроектирована в 80-х годах 20 века и используется до сих пор в промышленной электронике.

3.5 Реализация аппаратных интерфейсов: проблемы и технические решения

3.5.1 Электромагнитные помехи

В качестве электромагнитных помех (ЭМП) может фигурировать практически любое электромагнитное явление в широком диапазоне частот, способное негативно влиять на работу аппаратуры. Введем краткую классификацию помех, которая широко используется инженерами, работающими в области ЭМС.

В зависимости от источника ЭМП можно разделить на естественные и искусственные. Наиболее распространенной естественной ЭМП является электромагнитный импульс при ударе молнии. Искусственные помехи можно разделить на создаваемые функциональными и нефункциональными источниками. Источник помехи является функциональным, если для него самого создаваемая ЭМП является полезным сигналом.

В зависимости от среды распространения ЭМП могут разделяться на индуктивные и кондуктивные. Индуктивными называются ЭМП, распространяющиеся в виде электромагнитных полей в непроводящих средах. Кондуктивные ЭМП представляют собой токи, текущие по проводящим

конструкциям и земле. Деление помех на индуктивные и кондуктивные является, условным. В реальности протекает единый электромагнитный процесс, затрагивающий проводящую и непроводящую среду. В ходе распространения многие помехи могут превращаться из индуктивных в кондуктивные и наоборот. Так, переменное электромагнитное поле способно наводить токи в кабелях, которые далее распространяются как классические кондуктивные помехи. С другой стороны, токи в кабелях и цепях заземления сами создают электромагнитные поля, т.е., индуктивные помехи. Деление помех на индуктивные и кондуктивные можно считать относительно строгим лишь в низкочастотной (до десятков кГц) области, когда емкостные и индуктивные связи обычно малы.

Кондуктивные помехи в цепях, имеющих более одного проводника, принято также делить на помехи "провод–земля" (синонимы – несимметричные, общего вида, Common Mode) и "провод–провод" (симметричные, дифференциального вида, Differential Mode). В первом случае ("провод–земля") напряжение помехи приложено, как следует из названия, между каждым из проводников цепи и землей (рис. ниже, а). Во втором – между различными проводниками одной цепи (рис. ниже, б). Обычно самыми опасными для аппаратуры являются помехи "провод–провод", поскольку они оказываются приложенными так же, как и полезный сигнал.

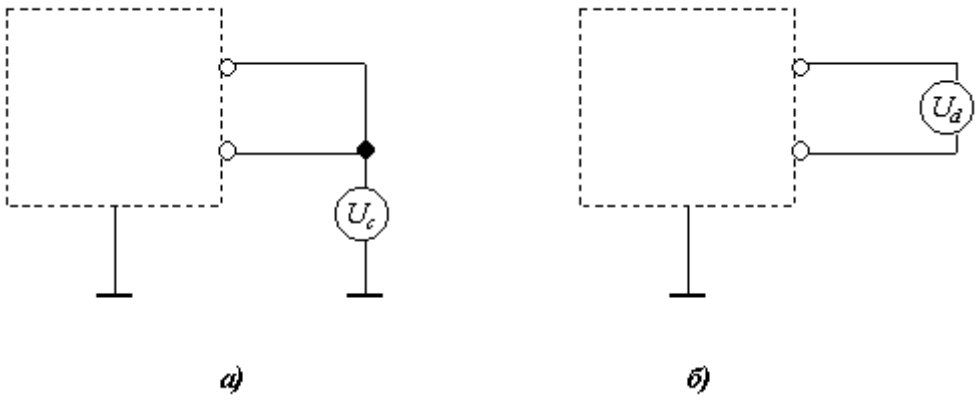


Рис. 29. Схема приложения помехи "провод–земля" (а) и "провод–провод" (б).

Реальные помехи обычно представляют собой комбинацию помех "провод–провод" и "провод–земля". Нужно учитывать, что несимметрия внешних цепей передачи сигналов и входных цепей аппаратуры может вызывать преобразование помехи "провод–земля" в помеху "провод–провод". На рис. ниже наглядно показан упрощенный процесс преобразования помехи: несимметрия внешних цепей (Zl1 не равно Zl2) и входных цепей аппаратуры приемника (Zi1 не равно Zi2) приводит к появлению помехи "провод–провод" величиной Ud = (Zi1/ Zl1 – Zi2/Zl2)Uc. В данном примере упрощение заключается в том, что внутреннее сопротивление приемника в режиме "провод–провод" принято равным бесконечности (т.е., в качестве измерителя полезного сигнала включен идеальный вольтметр).

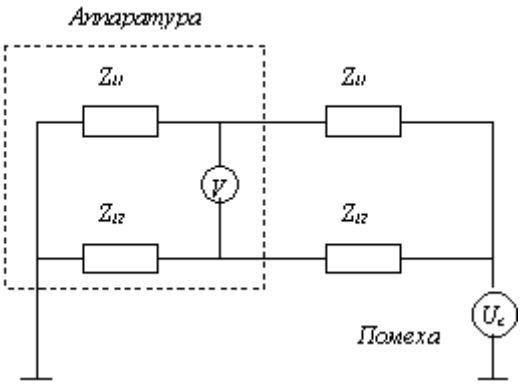


Рис. 30. Преобразование помехи "провод–земля" в помеху "провод–провод".

Следующие два способа классификации помех основываются на их спектральных характеристиках. Во-первых, ЭМП делятся на узкополосные и широкополосные. К первым относятся помехи спектр которых близок к линейчатому – максимальный уровень на основной частоте, пики меньшего уровня на частотах гармоник. Такие помехи обычно возникают от систем связи на несущей частоте, систем питания переменным током. Широкополосные помехи обычно проявляются в виде либо отдельных импульсов, либо их последовательности. Спектр периодических широкополосных помех состоит из большого набора пиков на частотах, кратных частоте основного сигнала. Для апериодических помех спектр является непрерывным и описывается спектральной плотностью [31].

3.5.2 Характеристики линии связи

Волновое сопротивление – это сопротивление, которое встречает электромагнитная волна при распространении вдоль однородной линии без отражения. В бесконечно длинной линии или линии конечной длины, но нагруженной на сопротивление, равное волновому сопротивлению, не происходит отражения электромагнитных волн и образования стоячих волн. В этом случае линия передаёт в нагрузку практически всю энергию от генератора (без потерь) [79].

$$\rho = \sqrt{\frac{L}{C}}$$

где L и C – индуктивность и ёмкость единицы длины линии. Формула справедлива для высоких частот.

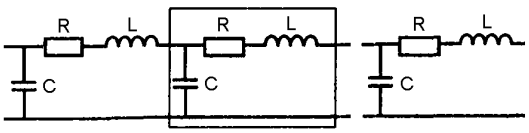


Рис. 31. Эквивалентная схема линии передачи

Волновое сопротивление является комплексной величиной и состоит из активной и реактивной части. Зависимость волнового сопротивления от частоты повышается в области низких частот и имеет емкостной характер ($2\pi fL \ll R$). В области высоких частот имеет место $2\pi fL > R$, $2\pi fC \gg (1/R)$ и значение волнового сопротивления стремится к постоянной величине [39].

Длинная линия – электрическая линия, образованная двумя параллельными проводниками тока, длина которых превышает длину волны передаваемых электромагнитных колебаний, а расстояние между проводниками значительно меньше длины волны. Длинная линия является системой с распределёнными постоянными (параметрами), так как каждый элемент её длины обладает одновременно некоторыми значениями индуктивности L и активного сопротивления R проводов, ёмкости C и проводимости тока G между проводами. Через эти параметры определяют основные характеристики длинной линии – волновое сопротивление ρ и скорость распространения v электромагнитных волн вдоль неё. Длинная линия математически связаны между собой так называемыми телеграфными уравнениями. Длинная линия называется **однородной**, если значения её параметров неизменны на всём протяжении; при отсутствии в ней электрических потерь, т. е. R = G = 0 (обычно на радиочастотах),

$$v = \frac{1}{\sqrt{LC}}$$

Длинная линия имеет в общем случае комплексный характер (содержит активную и реактивную составляющие) и зависит от длины линии и характера электрической нагрузки на её конце (выходе). Входное сопротивление длинная линия бесконечной длины равно ρ. Для максимальной передачи энергии от источника линии её входное сопротивление должно быть активным и равным внутреннему сопротивлению источника, т. е. согласованным с ним. Различают 3 режима работы длинной линия:

1. Режим бегущей волны, когда передаваемая энергия полностью поглощается нагрузкой (сопротивление нагрузки активное и равное W);
2. Режим стоячей волны, когда передаваемая энергия полностью отражается от конца линии к источнику (короткозамкнутая или разомкнутая на конце длинной линии),
3. Промежуточный режим (сопротивление нагрузки комплексное и не равное ρ). Длинная линия применяют для передачи информации в дальней телеграфно-телефонной связи, телевидении, радиолокации, а также для передачи энергии по проводам на далёкие расстояния [32],[35].

Стоячие волны – волны, возникающие вследствие интерференции волн, распространяющихся во взаимно противоположных направлениях. Практически стоячие волны возникают при отражениях волн от преград и неоднородностей в результате наложения отражённой волны на прямую.

Различные участки стоячие волны колеблются в одной и той же фазе, но с различной амплитудой. В стоячей волне, в отличие от бегущей, не происходит течения энергии. Бегущие волны отражаются от границ системы, и в результате наложения падающих и отражённых волн в системе устанавливаются стоячие волны [36].

Согласование электрических линий связи применяется для обеспечения нормального прохождения сигнала по длинной линии без отражений и искажений. Следует отметить, что в локальных сетях кабель работает в режиме длинной линии даже при минимальных расстояниях между компьютерами, так как скорости передачи информации и частотный спектр сигнала очень велики.

Принцип согласования кабеля прост: на его концах необходимо установить согласующие резисторы (терминаторы) с сопротивлением, равным волновому сопротивлению используемого кабеля.

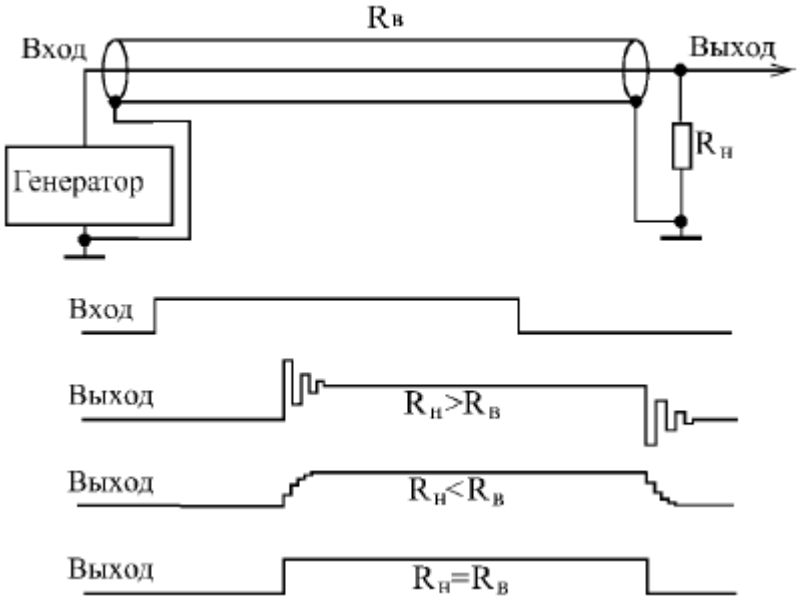


Рис. 32. Передача сигналов по электрическому кабелю

Как уже упоминалось, волновое сопротивление – это параметр данного типа кабеля, зависящий только от его устройства (сечения, количества и формы проводников, толщины и материала изоляции и т.д.). Величина волнового сопротивления обязательно указывается в сопроводительной документации на кабель и составляет обычно от 50-100 Ом для коаксиального кабеля, до 100-150 Ом для витой пары или плоского многопроводного кабеля. Точное значение волнового сопротивления легко можно измерить с помощью генератора прямоугольных импульсов и осциллографа как раз по отсутствию искажения формы передаваемого по кабелю импульса. Обычно требуется, чтобы отклонение величины согласующего резистора не превышало 10% в ту или другую сторону.

Если согласующее, нагрузочное сопротивление Rн меньше волнового сопротивления кабеля Rв, то фронт передаваемого прямоугольного импульса на

приемном конце будет затянут, если же Rн больше Rв, то на фронте будет колебательный процесс [65].

3.5.3 Виды линий связи

3.5.3.1 Коаксиальный кабель

Коаксиальный кабель – кабель, в котором оба проводника тока, образующие электрическую цепь, представляют собой 2 соосных цилиндра. Коаксиальный кабель применяется для передачи электрических сигналов в линиях дальней связи, в антенно-фидерных устройствах радиоэлектронной и телевизионной аппаратуры, между блоками радиотехнической аппаратуры и т.д. Электромагнитное поле коаксиального кабеля сосредоточено в пространстве между проводниками тока, т.е. внешнего поля нет, и поэтому потери на излучение в окружающее коаксиальный кабель пространство практически отсутствуют. Так как внешний проводник одновременно служит электромагнитным экраном, защищающим электрическую цепь тока от влияний извне, коаксиальный кабель обладает высокой помехозащищенностью. Коаксиальный кабель имеет относительно малые потери энергии передаваемых сигналов.

3.5.3.2 Витая пара

Витая пара – вид кабеля связи, представляет собой одну или несколько пар изолированных проводников, скрученных между собой (с небольшим числом витков на единицу длины), покрытых пластиковой оболочкой.

Скручивание проводов в витую пару преследует следующие цели:

- Обеспечить одинаковое влияние помехи на оба провода.
- Обеспечить компенсацию излучения электромагнитных полей, за счет близкого к 100% взаимного влияния проводников и использование дифференциального (противофазного) сигнала.

В кабелях, содержащих в себе несколько витых пар, для уменьшения взаимного влияния у каждой пары делают свой шаг витков [87].

3.5.3.3 Плоский кабель

Плоский кабель представляет собой несколько изолированных проводников, скреплённых друг с другом в виде плоской полосы. Разъемы к плоскому кабелю не паяются, а обжимаются специальным инструментом. Цель создания плоского кабеля – ускорение монтажа и минимизация затрат. Кабель такого типа хорошо подходит для передачи высокочастотных сигналов на короткие дистанции (обычно, не более 3 метров).

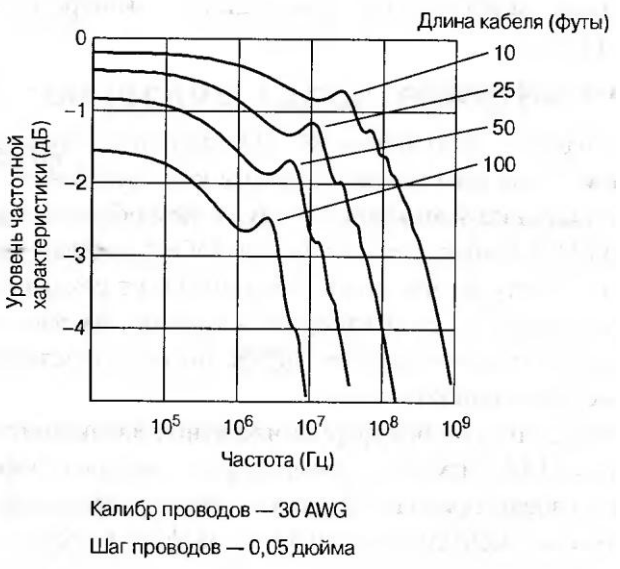


Рис. 33. Частотная характеристика плоского кабеля

Для плоского кабеля часто используют схему «земля-сигнал-земля», при которой сигнальные и земляные провода чередуются друг с другом. При такой схеме чередования, волновое сопротивление кабеля находится в диапазоне от 80 до 100 Ом.

3.5.3.4 Полосковые линии связи

Полосковая линия в технике сверхвысоких частот, плоскостная линия, канализирующая электромагнитные волны в воздушной или иной диэлектрической среде вдоль двух пли нескольких проводников, имеющих форму тонких полосок и пластин. Наряду с двухпроводными и коаксиальными линиями полосковая линия представляет собой разновидность радиоволновода. Электропроводящим материалом полосок и пластин служат медь, сплавы металлов, обладающие высокой проводимостью, серебро или (реже) золото, а в качестве диэлектрика выбирается фторопласт, полиэтилен, ситалл, керамика или др. материал с малыми потерями энергии на СВЧ и высокой диэлектрической проницаемостью (до 20). Существует много типов полосковых линий, которые подразделяют на симметричные и несимметричные линии. Полосковые линии характеризуют волновым сопротивлением (обычно 50-150 Ом), зависящим от типа диэлектрика и геометрических размеров линии, коэффициентом затухания на единицу длины (обычно 0,1-1,8 дб/м), рабочей полосой частот (практически 100 Мгц - 100 Ггц) [70].

3.5.4 Сбалансированная схема

Сбалансированные схемы – это схемы, в которых уменьшение помех осуществляется с использованием схемотехнических решений [47].

Рассмотрим обычную несимметричную схему с наведенным на земле шумом.

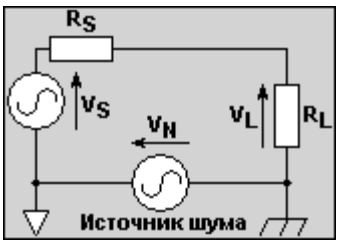


Рис. 34. Несбалансированная схема с "земляным" шумом

Отметим, что источник шума располагается между двумя частями схемы. В реальном устройстве он может находиться на печатной плате или в соединениях между отдаленным датчиком и измерительной схемой. Легко видеть, что выходное напряжение VL вычисляется простым способом

$$VL= ((VS+VN)RL) / (RS+RL)$$

Шум, добавляемый к сигналу, уменьшает отношение сигнал/шум. Во многих случаях это действие ухудшает качественные характеристики устройства и может быть недопустимым. На рисунке, показан альтернативный путь соединения этих двух частей схемы. В данном случае сигнал разделен на две равные части с заземлением общей точки. Источник входного сигнала и нагрузочный резистор дублированы, и выходной сигнал VL снимается с обоих нагрузок.

Основные преимущества сбалансированных схем:

1. Сигналы менее восприимчивы к помехам на шине земли;
2. Сигналы менее восприимчивы к помехам, вызванных емкостной связью;
3. Сигналы менее восприимчивы к помехам, вызванных индуктивной и электромагнитной связями;
4. Уменьшение собственного электромагнитного излучения по сравнению с несбалансированными схемами.

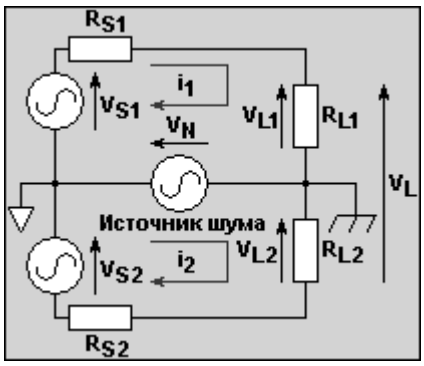


Рис. 35. Сбалансированная схема с "земляным" шумом

Главный недостаток сбалансированных схем заключается в том, что в них используется большее количество компонентов. Кроме того, возможно некорректное подключение сбалансированной системы с инвертированием сигнала. Решение о применении сбалансированной или несбалансированной структуры схемы должно быть в достаточной мере продуманным и никак не

случайным. Система всегда будет содержать части, не требующие сбалансированности или которые нельзя сбалансировать, и части, которые балансировать необходимо.

3.5.5 Симметричная и несимметричная схема передачи сигналов

3.5.5.1 Дифференциальный сигнал

Сигнал называется дифференциальным, когда по двум отдельным проводам передается два комплементарных (инверсных друг относительно друга) сигнала.

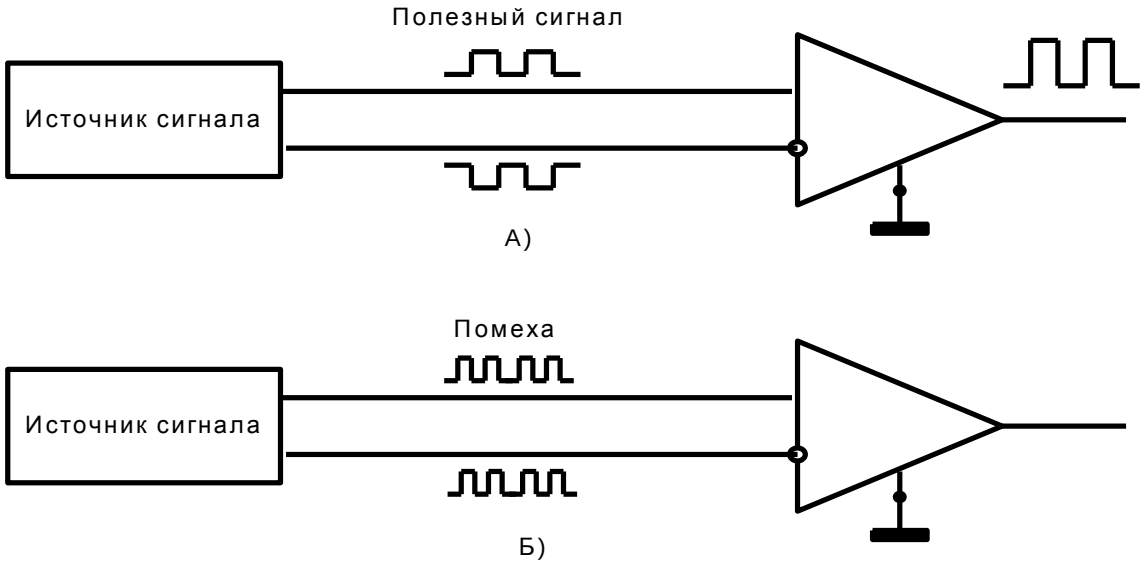


Рис. 36. Принцип использования дифференциального сигнала.

На рисунке А видно, что от источника, по двум проводам передается два сигнала инверсные друг относительно друга. Оба сигнала попадают на вычитатель (операционный усилитель). После вычитания инверсных сигналов результирующий сигнал усиливается. На рисунке Б показано влияние помех на схему с дифференциальным сигналом. Суть в том, что помеха, например, наведенная от сети 220В будет синфазной, т.е. в обоих проводах она будет с одинаковой фазой. При попадании такой помехи на вход вычитателя мы получим на выходе 0.

Таким образом, видно, что схемы передачи с дифференциальными сигналами обладают защитой от синфазных помех.

3.5.5.2 Несимметричная передача сигнала

Несимметричная или однопроводная схема (single-ended) предполагает, что сигнал передается от источника к приёмнику по одному проводу, а земли источника и приёмника связаны друг с другом.

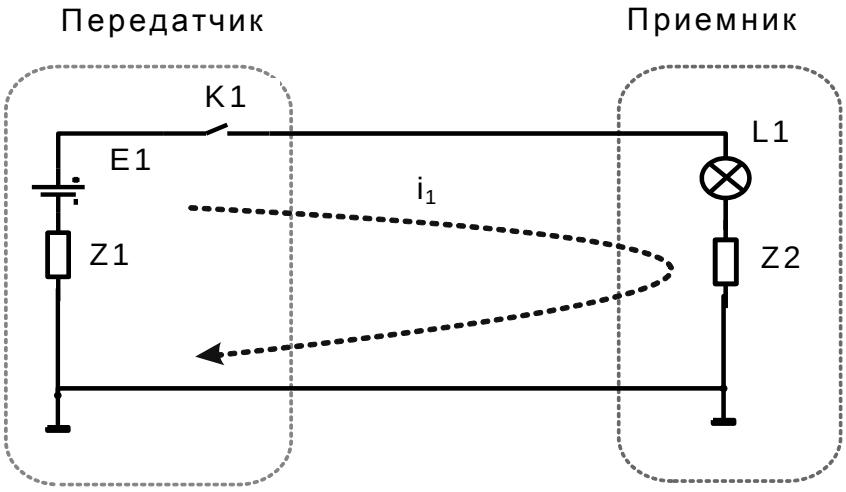


Рис. 37. Эквивалентная схема несимметричного способа передачи данных

На эквивалентной схеме видно, что земли источника и приемника связаны, между отрицательной клеммой источника питания E1 и лампы L1 есть импедансы Z1 и Z2, появляющиеся из-за того, что контакты и проводники имеют ненулевое сопротивление. На самом деле в однопроводной схеме проводов естественно два, просто общий провод по традиции учитывается по умолчанию.

Несимметричная схема передачи данных простая в реализации, но из-за целого ряда проблем не позволяет осуществлять передачу данных с большой скоростью на большие расстояния. Например, интерфейс RS-232C, работающий по несимметричной схеме позволяет передавать данные со скоростью 19200 бит в секунду на расстояние 15 метров, а интерфейс RS-485, работающий по симметричной схеме, позволяет на аналогичном расстоянии работать со скоростью 10 Мб/с.

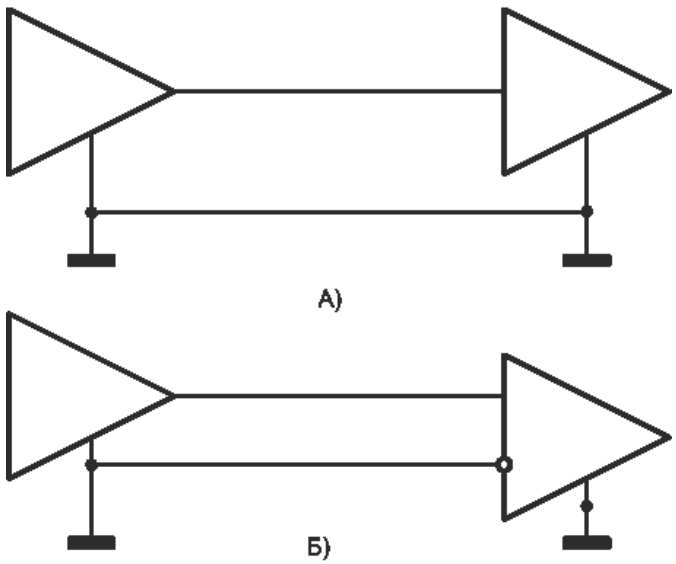


Рис. 38. Несимметричная передача сигнала. Вариант А используется в интерфейсах RS-232 (CCITT V.24/V.28, X.20bis/X.21bis и ISO IS2110), SPI, I2C, uLAN. Вариант Б, с дифференциальным входом, используется в интерфейсе RS-223A

К особенностям несимметричной схемы передачи сигналов можно отнести следующие:

- Для однопроводной (несимметричной) схемы передачи сигналов достаточно одного проводника.
- Однопроводная схема передачи сигналов чувствительна к нарушениям опорного напряжения.
- Однопроводная схема передачи сигналов восприимчива к дребезгу земли.

Для однопроводной передачи сигналов требуется, чтобы соединение на общую опорную шину (опорный слой) было низкоимпедансным [34].

Рассмотрим несимметричную схему, состоящую из двух источников напряжения E1 и E2. Источник напряжения E1, ключ K1 и импеданс корпусного вывода Z1 является источником сигнала, а лампочка L1 и импеданс корпусного вывода Z2 – приемником. К лампочке подключен минусовой провод второго источника питания E1, к которому подключен ключ K2, образующий второй передатчик, имеющий общую землю с первым.

В схеме протекает два тока i1 и i2. На схеме видно, что через импеданс Z2 протекает два тока i1 и i2, создавая там падение напряжения Z2i1 + Z2i2. Так как сумма напряжений в цепи есть сумма падений напряжений на всех участках цепи, получается, что напряжение на лампе зависит от наличия или отсутствия тока i2. Эффект влияния соседней цепи такого рода называется **влиянием через общий импеданс** или **дребезгом земли**. Для того, чтобы влияние соседней цепи было минимальным необходимо, чтобы импеданс Z2 был близок к нулю.

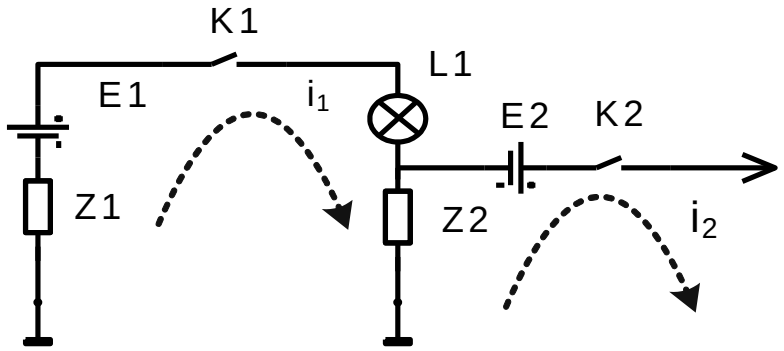


Рис. 39. Схема иллюстрирующая связь через общий импеданс.

Результатом дребезга влияния является помеха для полезного сигнала в приемнике L1.

Как уже говорилось выше, импедансы Z1 и Z2 появляются из-за того, что контакты и провода далеко не идеальны и имеют сопротивление. Вообще, необходимо помнить, что проводники и соединения имеют не только активное сопротивление, но и реактивное (индуктивность и емкость). Кроме того, они оказывают друг на друга существенное влияние через электрические и магнитные поля.

Однопроводные схемы очень чувствительны к электромагнитным наводкам, например, к помехам, вызываемым электрической сетью (220В, 50Гц). Для обеспечения передачи данных по однопроводной схеме часто используют коаксиальные и экранированные провода, в которых центральная жила является сигнальной, а экран (оплётка) – общим проводом.

3.5.5.3 Симметричная передача сигнала

Симметричными являются двухпроводные схемы, в которых оба проводника и все подключенные к ним цепи имеют одинаковый импеданс относительно земли и любого другого проводника. По одному проводу идет прямой ток сигнала, по второму – обратный [68].

Цель симметрирования состоит в том, чтобы сделать равными шумы, наводимые в обоих проводниках; в этом случае они будут представлять собой продольный, или синфазный, сигнал, который можно скомпенсировать в нагрузке.

Симметрирование – метод подавления шумов, который можно использовать в сочетании с экранированием там, где уровень шумов должен быть ниже уровня, достижимого при использовании только экранирования, или даже вместо экранирования.

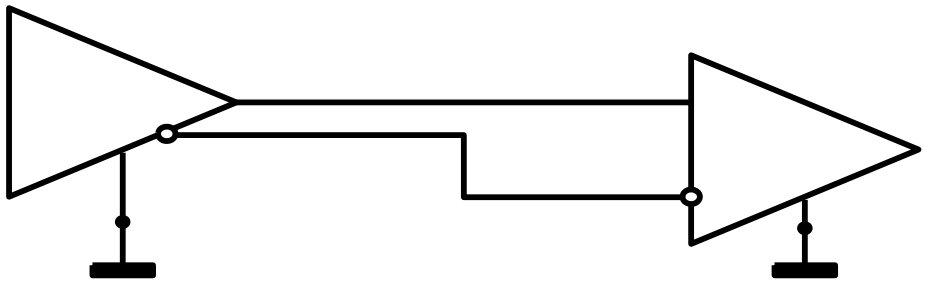


Рис. 40. Симметричная передача дифференциального сигнала. Используется в интерфейсах RS-422, RS-485, Ethernet, USB, LVDS, PCI Express.

- Двухпроводная (симметричная) схема передачи сигнала делает систему невосприимчивой к флуктуациям распределения общих опорных напряжений.
- В двухпроводной схеме передачи нейтрализуется любой вид помех, поражающий в равной степени оба проводника.
- В двухпроводной схеме передачи нейтрализуется дребезг земли (называемый также комбинационными коммутационными помехами) в приемнике.
- В двухпроводной схеме передачи нейтрализуются сдвиги земли, возникающие в высокочастотных разъемах.
- Двухпроводная схема передачи эффективно работает при условии ограничения паразитного возвратного тока сигнала [76].

3.5.6 Виды кодирования

При цифровом кодировании дискретной информации применяют потенциальные и импульсные коды [52].

В потенциальных кодах для представления логических единиц и нулей используется только значение потенциала сигнала, а его перепады, формирующие законченные импульсы, во внимание не принимаются. Импульсные коды позволяют представить двоичные данные либо импульсами определенной полярности, либо частью импульса - перепадом потенциала определенного направления.

При использовании прямоугольных импульсов для передачи дискретной информации необходимо выбрать такой способ кодирования, который одновременно достигал бы нескольких целей:

- Имел при одной и той же битовой скорости наименьшую ширину спектра результирующего сигнала.
- Обеспечивал синхронизацию между передатчиком и приемником.
- Обладал способностью распознавать ошибки.
- Обладал низкой стоимостью реализации.

Более узкий спектр сигналов позволяет на одной и той же линии (с одной и той же полосой пропускания) добиваться более высокой скорости передачи данных. Кроме того, часто к спектру сигнала предъявляется требование отсутствия постоянной составляющей, т.е. наличия постоянного тока между передатчиком и приемником. В частности, применение различных трансформаторных схем гальванической развязки препятствует прохождению постоянного тока.

Синхронизация передатчика и приемника нужна для того, чтобы приемник точно знал, в какой момент времени необходимо считывать новую информацию с линии связи. Эта проблема в сетях решается сложнее, чем при обмене данными между близко расположенными устройствами, например, между блоками внутри компьютера или же между компьютером и принтером. На небольших расстояниях хорошо работает схема, основанная на отдельной тактирующей линии связи, так что информация снимается с линии данных только в момент прихода тактового импульса. В сетях использование этой схемы вызывает трудности из-за неоднородности характеристик проводников в кабелях. На больших расстояниях неравномерность скорости распространения сигнала может привести к тому, что тактовый импульс придет настолько позже или раньше соответствующего сигнала данных, что бит данных будет пропущен или считан повторно. Другой причиной, по которой в сетях отказываются от использования тактирующих импульсов, является экономия проводников в дорогостоящих кабелях.

Поэтому в сетях применяются так называемые самосинхронизирующиеся коды, сигналы которых несут для передатчика указания о том, в какой момент

времени нужно осуществлять распознавание очередного бита (или нескольких бит, если код ориентирован более чем на два состояния сигнала). Любой резкий перепад сигнала – так называемый фронт – может служить хорошим указанием для синхронизации приемника с передатчиком.

При использовании синусоид в качестве несущего сигнала результирующий код обладает свойством самосинхронизации, так как изменение амплитуды несущей частоты дает возможность приемнику определить момент появления входного кода.

Распознавание и коррекцию искаженных данных сложно осуществить средствами физического уровня, поэтому чаще всего эту работу берут на себя протоколы, лежащие выше: канальный, сетевой, транспортный или прикладной. С другой стороны, распознавание ошибок на физическом уровне экономит время, так как приемник не ждет полного помещения кадра в буфер, а отбраковывает его сразу при распознавании ошибочных бит внутри кадра.

Требования, предъявляемые к методам кодирования, являются взаимно противоречивыми, поэтому каждый из рассматриваемых ниже популярных методов цифрового кодирования обладает своими преимуществами и своими недостатками по сравнению с другими.

3.5.6.1 Потенциальный код без возвращения к нулю

На рисунке показан уже упомянутый ранее метод потенциального кодирования, называемый также кодированием без возвращения к нулю (Non Return to Zero, NRZ). Последнее название отражает то обстоятельство, что при передаче последовательности единиц сигнал не возвращается к нулю в течение такта (как мы увидим ниже, в других методах кодирования возврат к нулю в этом случае происходит). Метод NRZ прост в реализации, обладает хорошей распознаваемостью ошибок (из-за двух резко отличающихся потенциалов), но не обладает свойством самосинхронизации. При передаче длинной последовательности единиц или нулей сигнал на линии не изменяется, поэтому приемник лишен возможности определять по входному сигналу моменты времени, когда нужно в очередной раз считывать данные. Даже при наличии высокоточного тактового генератора приемник может ошибиться с моментом съема данных, так как частоты двух генераторов никогда не бывают полностью идентичными. Поэтому при высоких скоростях обмена данными и длинных последовательностях единиц или нулей небольшое рассогласование тактовых частот может привести к ошибке в целый такт и, соответственно, считыванию некорректного значения бита.

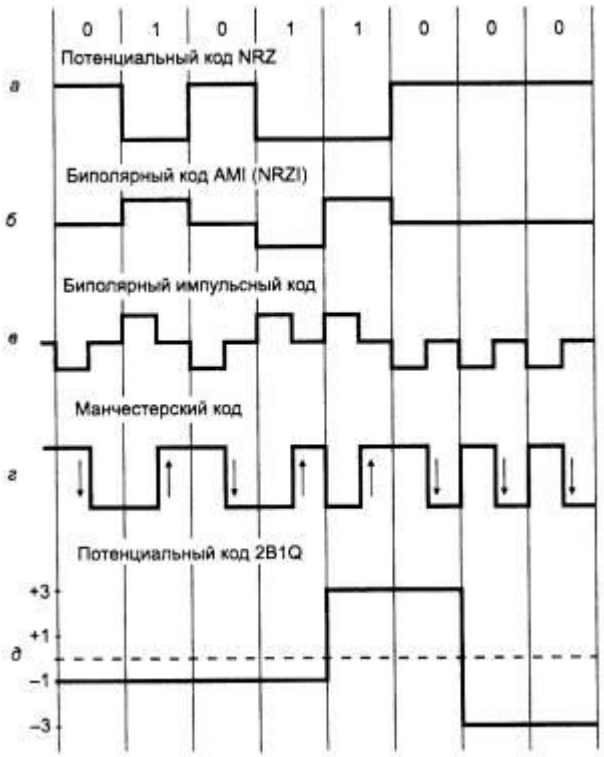


Рис. 41. Способы дискретного кодирования данных

Другим серьезным недостатком метода NRZ является наличие низкочастотной составляющей, которая приближается к нулю при передаче длинных последовательностей единиц или нулей. Из-за этого многие каналы связи, не обеспечивающие прямого гальванического соединения между приемником и источником, этот вид кодирования не поддерживают. В результате в чистом виде код NRZ в сетях не используется. Тем не менее используются его различные модификации, в которых устраняют как плохую самосинхронизацию кода NRZ, так и наличие постоянной составляющей. Привлекательность кода NRZ, из-за которой имеет смысл заняться его улучшением, состоит в достаточно низкой частоте основной гармоники f0, которая равна N/2 Гц, как это было показано в предыдущем разделе. У других методов кодирования, например, манчестерского, основная гармоника имеет более высокую частоту.

3.5.6.2 Метод биполярного кодирования с альтернативной инверсией

Одной из модификаций метода NRZ является метод биполярного кодирования с альтернативной инверсией (Bipolar Alternate Mark Inversion, AMI). В этом методе (рис. выше, б) используются три уровня потенциала - отрицательный, нулевой и положительный. Для кодирования логического нуля используется нулевой потенциал, а логическая единица кодируется либо положительным потенциалом, либо отрицательным, при этом потенциал каждой новой единицы противоположен потенциалу предыдущей.

Код AMI частично ликвидирует проблемы постоянной составляющей и отсутствия самосинхронизации, присущие коду NRZ. Это происходит при

передаче длинных последовательностей единиц. В этих случаях сигнал на линии представляет собой последовательность разнополярных импульсов с тем же спектром, что и у кода NRZ, передающего чередующиеся нули и единицы, т.е. без постоянной составляющей и с основной гармоникой N/2 Гц (где N – битовая скорость передачи данных). Длинные же последовательности нулей также опасны для кода AMI, как и для кода NRZ – сигнал вырождается в постоянный потенциал нулевой амплитуды. Поэтому код AMI требует дальнейшего улучшения, хотя задача упрощается – осталось справиться только с последовательностями нулей.

В целом, для различных комбинаций бит на линии использование кода AMI приводит к более узкому спектру сигнала, чем для кода NRZ, а значит, и к более высокой пропускной способности линии. Например, при передаче чередующихся единиц и нулей основная гармоника fo имеет частоту N/4 Гц. Код AMI предоставляет также некоторые возможности по распознаванию ошибочных сигналов. Так, нарушение строгого чередования полярности сигналов говорит о ложном импульсе или исчезновении с линии корректного импульса. Сигнал с некорректной полярностью называется запрещенным сигналом (signal violation).

В коде AMI используются не два, а три уровня сигнала на линии. Дополнительный уровень требует увеличение мощности передатчика примерно на 3 дБ для обеспечения той же достоверности приема бит на линии, что является общим недостатком кодов с несколькими состояниями сигнала по сравнению с кодами, которые различают только два состояния.

3.5.6.3 Потенциальный код с инверсией при единице

Существует код, похожий на AMI, но только с двумя уровнями сигнала. При передаче нуля он передает потенциал, который был установлен в предыдущем такте (т.е. не меняет его), а при передаче единицы потенциал инвертируется на противоположный. Этот код называется потенциальным кодом с инверсией при единице (Non Return to Zero with ones Inverted, NRZI). Этот код удобен в тех случаях, когда использование третьего уровня сигнала весьма нежелательно, например, в оптических кабелях, где устойчиво распознаются два состояния сигнала – свет и темнота.

Для улучшения потенциальных кодов, подобных AMI и NRZI, используются два метода. Первый метод основан на добавлении в исходный код избыточных бит, содержащих логические единицы. Очевидно, что в этом случае длинные последовательности нулей прерываются и код становится самосинхронизирующимся для любых передаваемых данных. Исчезает также постоянная составляющая, а значит, еще более сужается спектр сигнала. Но этот метод снижает полезную пропускную способность линии, так как избыточные единицы пользовательской информации не несут. Другой метод основан на предварительном «перемешивании» исходной информации таким образом, чтобы вероятность появления единиц и нулей на линии становилась

близкой. Устройства, или блоки, выполняющие такую операцию, называются трамблерами (scramble – свалка, беспорядочная сборка). При скремблировании используется известный алгоритм, поэтому приемник, получив двоичные данные, передает их на дескрэмблер, который восстанавливает исходную последовательность бит. Избыточные биты при этом по линии не передаются. Оба метода относятся к логическому, а не физическому кодированию, так как форму сигналов на линии они не определяют. Более детально они изучаются в следующем разделе.

3.5.6.4 Биполярный импульсный код

Кроме потенциальных кодов в сетях используются и импульсные коды, когда данные представлены полным импульсом или же его частью - фронтом. Наиболее простым случаем такого подхода является биполярный импульсный код, в котором единица представлена импульсом одной полярности, а ноль - другой. Каждый импульс длится половину такта. Такой код обладает отличными самосинхронизирующими свойствами, но постоянная составляющая, может присутствовать, например, при передаче длинной последовательности единиц или нулей. Кроме того, спектр у него шире, чем у потенциальных кодов. Так, при передаче всех нулей или единиц частота основной гармоники кода будет равна N Гц, что в два раза выше основной гармоники кода NRZ и в четыре раза выше основной гармоники кода AMI при передаче чередующихся единиц и нулей. Из-за слишком широкого спектра биполярный импульсный код используется редко.

3.5.6.5 Манчестерский код

В локальных сетях до недавнего времени самым распространенным методом кодирования был так называемый манчестерский код (рис. выше, г). Он применяется в технологиях Ethernet и Token Ring.

В манчестерском коде для кодирования единиц и нулей используется перепад потенциала, т.е. фронт импульса. При манчестерском кодировании каждый такт делится на две части. Информация кодируется перепадами потенциала, происходящими в середине каждого такта. Единица кодируется перепадом от низкого уровня сигнала к высокому, а ноль – обратным перепадом. В начале каждого такта может происходить служебный перепад сигнала, если нужно представить несколько единиц или нулей подряд. Так как сигнал изменяется по крайней мере один раз за такт передачи одного бита данных, то манчестерский код обладает хорошими самосинхронизирующими свойствами. Полоса пропускания манчестерского кода уже, чем у биполярного импульсного. У него также нет постоянной составляющей, а основная гармоника в худшем случае (при передаче последовательности единиц или нулей) имеет частоту N Гц, а в лучшем (при передаче чередующихся единиц и нулей) она равна N/2 Гц, как и у кодов AMI или NRZ. В среднем ширина полосы манчестерского кода в полтора раза уже, чем у биполярного импульсного кода, а основная гармоника колеблется вблизи значения 3N/4.

Манчестерский код имеет еще одно преимущество перед биполярным импульсным кодом. В последнем для передачи данных используются три уровня сигнала, а в манчестерском – два.

3.5.6.6 Потенциальный код 2B1Q

На рисунке показан потенциальный код с четырьмя уровнями сигнала для кодирования данных. Это код 2B1Q, название которого отражает его суть - каждые два бита (2B) передаются за один такт сигналом, имеющим четыре состояния (1Q), Паре бит 00 соответствует потенциал -2,5 В, паре бит 01 соответствует потенциал -0,833 В, паре 11 - потенциал +0,833 В, а паре 10 - потенциал +2,5 В. При этом способе кодирования требуются дополнительные меры по борьбе с длинными последовательностями одинаковых пар бит, так как при этом сигнал превращается в постоянную составляющую. При случайном чередовании бит спектр сигнала в два раза уже, чем у кода NRZ, так как при той же битовой скорости длительность такта увеличивается в два раза. Таким образом, с помощью кода 2B1Q можно по одной и той же линии передавать данные в два раза быстрее, чем с помощью кода AMI или NRZI. Однако для его реализации мощность передатчика должна быть выше, чтобы четыре уровня четко различались приемником на фоне помех.

3.5.7 Приемопередатчик последовательного интерфейса

В RS-232C используются уровни сигналов -15В...+15В. Зоной нечувствительности, т.е. отсутствия сигналов, считается напряжение -3В...+3В. При этом обратите внимание, что принимаемые/передаваемые данные инвертированы. Предельное напряжение на входе приёмника: -25В…+25В [89].

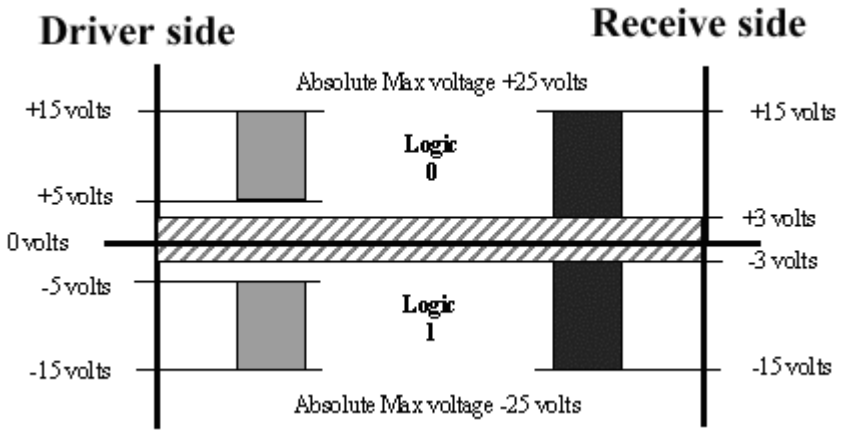


Рис. 42. Уровни сигналов UART по стандарту RS-232C.

Уровни сигналов UART по стандарту RS-232C (исходные состояния):

- Порт не инициализирован – на всех линиях напряжения находятся в диапазоне -3В...+3В.
- Напряжение на выходе передатчика при логическом «0» от +5 до +15 В, при логической «1» от -5 до 15В.

- Напряжение на входе приёмника при логическом «0» от +3 до +25 В, при логической «1» от -3 до 25В.

В RS-232C используется биполярный вариант кодирования NRZ (None Return to Zero). Метод NRZ прост в реализации, обладает хорошей распознаваемостью ошибок (из-за двух резко отличающихся потенциалов), но не обладает свойством самосинхронизации. При передаче длинной последовательности единиц или нулей сигнал на линии не изменяется, поэтому приемник лишен возможности определять по входному сигналу моменты времени, когда в очередной раз нужно считывать данные. Для синхронизации начала приема пакета используется стартовый служебный бит, например, единица.

Для NRZ требуется узкая полоса пропускания канала связи. Основная гармоника f0 имеет достаточно низкую частоту равную N/2 Гц, где N – битовая скорость передачи дискретных данных [бит/с] [18].

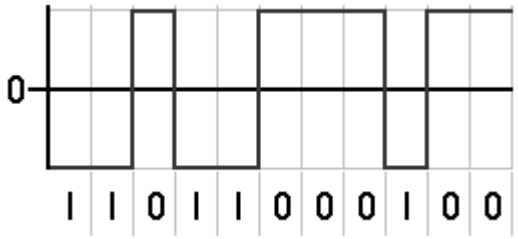


Рис. 43. Биполярное NRZ кодирование.

На рисунке ниже показана эквивалентная электрическая схема при обмене последовательными данными по стандарту RS-232C. Эта эквивалентная схема независима от того, где расположен генератор в DTE или DCE. По схеме видно, что опорное напряжение является общим для всей схемы, поэтому такая схема относится к классу однопроводных или несимметричных схем.

Характеристики сигнала обмена данными по стандарту RS-232C включены в международный стандарт ITU-T v.28 [101].

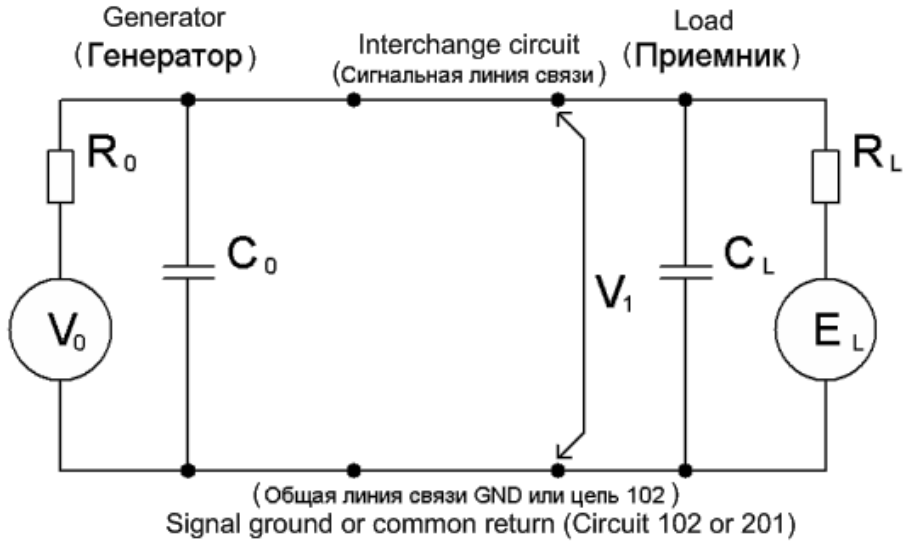


Рис. 44. Эквивалентная электрическая схема RS-232C.

- V0 – напряжение генератора при разомкнутой схеме.
- R0 – общее сопротивление генератора.
- C0 – общая ёмкость генератора.
- V1 – напряжение между сигнальной линией и общим проводом в месте стыка.
- CL – общая ёмкость приёмника.
- RL – общее сопротивление приёмника.
- EL – ЭДС приёмника при разомкнутой схеме.

Стыком интерфейса RS-232C считается линия соединения DTE плюс кабель с DCE, т.е. соединительный кабель интерфейса входит в состав DTE [101].

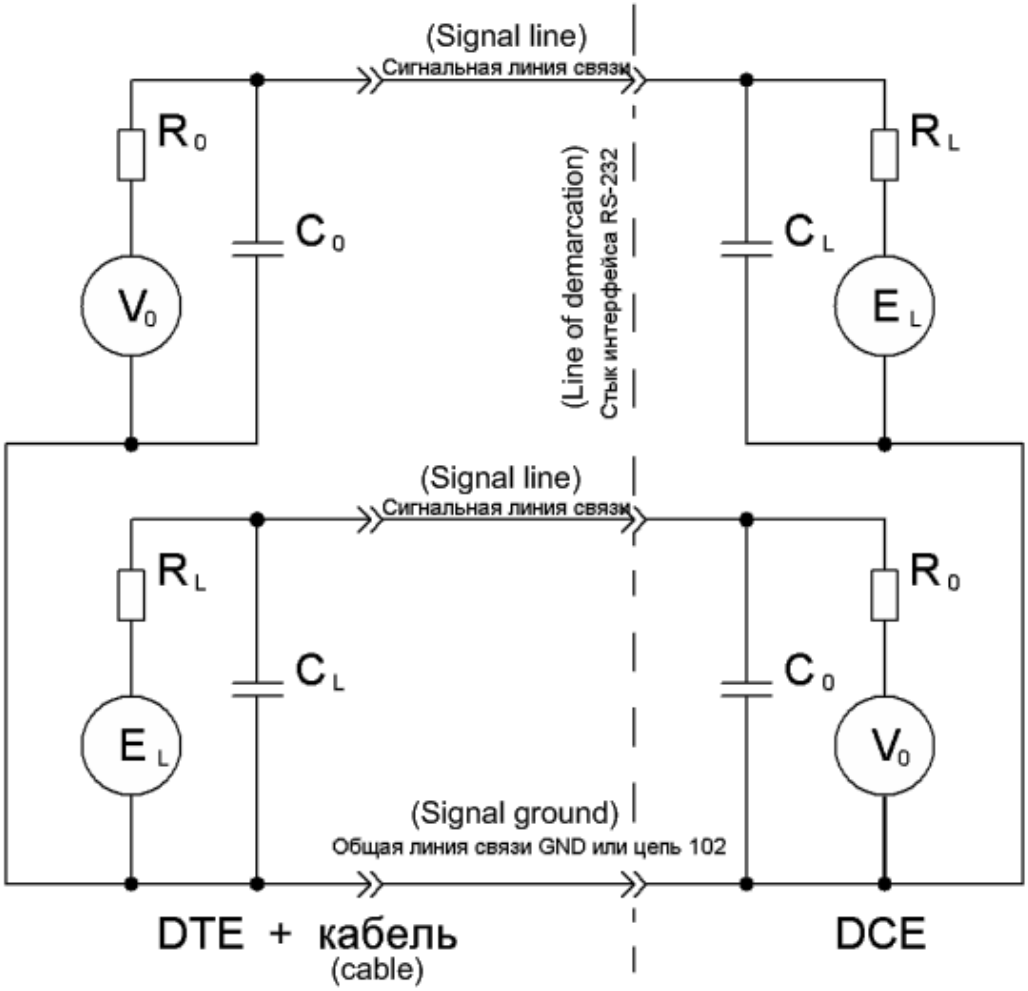


Рис. 45. Практическая схема стыка интерфейса RS-232C.

Электрические характеристики приёмника сигналов:

- RL – общее сопротивление приёмника должно находиться в пределах 3000...7000 Ом.
- V1 – напряжение на входе приёмника должно быть в пределах +3...+15В.

- EL – ЭДС приёмника при разомкнутой схеме должно быть не более +2В.
- CL – общая ёмкость цепей приёмника должна быть не более 2500 пФ.
- Входной импеданс приёмника не должен быть индуктивным.

Электрические характеристики генератора сигналов [101]:

- Допускается короткое замыкание сигналов.
- Допускается оставлять выход генератора без нагрузки.
- V0 – напряжение генератора при разомкнутой схеме должно быть не более +25В/+15 В (RS-232/ITU-T v.28).
- R0 и C0 для генератора не нормируются.
- Короткое замыкание цепей генератора не должно вызывать токи величиной более 0,5А.
- Если EL=0, то напряжение на входе приёмника должно быть V1=+5...+15 В, для любого диапазона нагрузки генератора RL=3000...7000 Ом.
- Генератор должен быть способен работать на ёмкостную нагрузку C0 плюс 2500 пФ.

Уровни сигналов для стандарта RS-232C [101]:

- Логической «1» считается информационный сигнал с напряжением V1 менее -3 В.
- Логическим «0» считается информационный сигнал с напряжением V1 более +3 В.
- Сервисный или синхронизирующий сигнал считается включенным «ON» («MARK») если V1 более +3 В.
- Сервисный или синхронизирующий сигнал считается выключенным «OFF» («SPACE») если V1 менее -3 В.
- Напряжение в диапазоне V1=-3 В...+3 В считается переходной областью.

Характеристики сигналов [101]:

- Все сигналы вошедшие в область перехода V1=-3В...+3В должны выйти в противоположный сигнал без повторного захода в эту область (т.е. монотонно).
- Не допускается колебания сигнала в области перехода.
- Сервисные и синхронизирующие сигналы должны проходить область перехода за время не более 1мс.
- Сигналы данных должны проходить область перехода за время не более 3% от времени одиночного элемента, но не более чем за 1 мс.

- Скорость нарастания фронта сигнала не должна превышать величины 30В за миллисекунду.
- Ограничения первых двух пунктов не относятся к электромеханическим устройствам размыкания и замыкания цепи [67], [53], [61], [27], [72].

3.5.8 Особенности параллельных интерфейсов

- Высокая стоимость погонного метра магистрали обусловлена большим количеством линий.
- Высокая скорость: удвоение количества линий для передачи данных способствует удвоению скорости канала. На практике это не совсем так, потому что присутствует разница в скорости распространения сигналов по параллельным линиям, т. е. разное время прихода сигналов (битов) на приемной стороне. В итоге скорость параллельного интерфейса снижается до скорости передачи сигнала по самой его медленной линии.
- Ограниченная длина интерфейса, которая обычно составляет от нескольких метров до десятков метров и в редких случаях достигает сотни. Объясняется это перекрестными помехами, наводками в соседних линиях, возникающими при передаче данных. Такие физические эффекты уменьшают не только длину кабеля, но и скорость передачи данных по нему (для минимизации помех).
- Простота схемотехнической реализации. Параллельный интерфейс на стороне передатчика и приемника должен иметь параллельные порты (буферы-защелки) для чтения/записи данных с шины. В случае последовательного интерфейса необходимым является преобразование параллельного кода в последовательный для передачи и обратное преобразование при приеме данных, которые выполняют специализированные микросхемы (например, UART в случае интерфейса RS-232). Кабели параллельных интерфейсов обычно имеют недорогую простую конструкцию, например, ленточную.

Широкое распространение параллельных интерфейсов связано с состоянием развития элементной базы в последней трети XX века. Тогда большая часть изделий базировалась на микросхемах малой и средней степени интеграции. Более простая реализация параллельного интерфейса по сравнению с последовательным выливалась в улучшение технических и экономических характеристик изделия. Кабельное соединение имело меньшую относительную стоимость. К настоящему времени получила развитие специальная элементная база. Порт интерфейса как последовательного, так и параллельного чаще всего может быть реализован на одной из специальных микросхем. Поэтому относительная стоимость кабельного соединения возросла. Применение последовательных интерфейсов стало более целесообразным на магистралях любой, но особенно большой длины. Таким образом, удешевление элементной базы, растущий спрос на высокоскоростную

передачу данных через большие расстояния, общая тенденция в увеличении плотности информационных потоков на единицу объема (или веса) аппаратуры привели к лавинообразному росту рынка последовательных интерфейсов. В результате имеют место быть следующие факты:

- Последовательный высокоскоростной обмен данными между узлами вычислительных систем/сетей с переходом к беспроводным технологиям связи.
- Большие потоки данных внутри вычислительных модулей (платы) и внутри кристаллов (микроконтроллеры, системы на кристалле) передаются по параллельным интерфейсам.

Параллельные интерфейсы стали уходить в прошлое в результате наступления последовательных интерфейсов: IEEE 1284 уступил USB, параллельный ATA – SATA, SCSI – SAS.

С другой стороны, идеология параллельных интерфейсов нашла свое проявление в современных технологиях беспроводной связи.

3.5.9 Мультиплексирование, конвейеризация, блочная передача

Для снижения общего количества линий связи магистрали часто применяется мультиплексирование шин адреса и данных, т.е. одни и те же линии связи используются в разные моменты времени для передачи как адреса, так и данных (в начале цикла – адрес, в конце цикла – данные). Для фиксации этих моментов (стробирования) служат специальные сигналы на шине управления. Понятно, что мультиплексированная шина адреса/данных обеспечивает меньшую скорость обмена, требует более длительного цикла обмена. По типу шины адреса и шины данных все магистрали также делятся на мультиплексированные и немультиплексированные [37], [98].

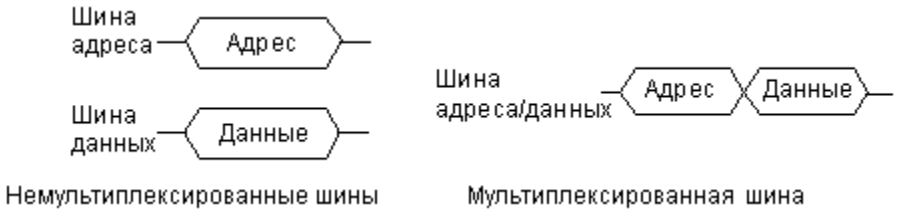


Рис. 46. Мультиплексирование шин адреса и данных.

Конвейеризация (pipelining) – обработка информации одновременно работающими модулями, каждый из которых выполняет заданную команду и передаёт результат другому модулю.

В PCI во время реакции памяти на запрос шина простаивает (но не свободна). Конвейерный доступ AGP позволяет в это время передавать следующие запросы, а потом получить поток ответов.

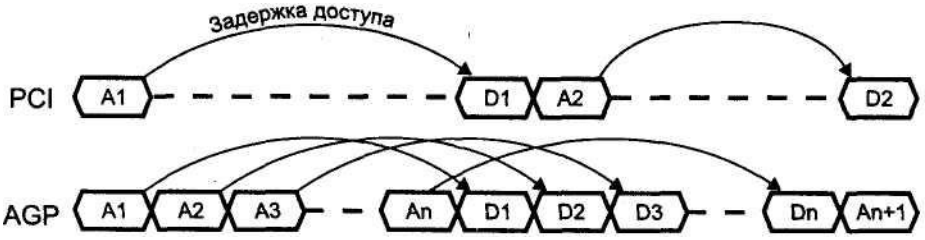


Рис. 47. Циклы обращения к памяти в шинах PCI и AGP.

Блочная передача данных – вариант передачи, при котором минимизируются расходы на передачу служебной информации с целью увеличения скорости передачи больших объемов данных. Например, при блочной передаче данных на шине VMEbus адрес выставляется только один раз в начале каждого блока, далее данные передаются подряд, до окончания блока.

3.5.10 Устройства гальванической изоляции в аппаратных интерфейсах

Гальваническая изоляция или гальваническая развязка – разделение электрических цепей посредством не проводящего ток материала. Для реализации гальванической изоляции можно использовать трансформаторы, конденсаторы, реле и оптроны [74].

Изоляция применяется для защиты от больших токов или напряжений, вызванных высоковольтными помехами и возникающих при наличии замкнутых цепей заземления. Такие замкнутые петли могут присутствовать в любой системе, где имеется несколько заземлений. Заземления в различных частях системы, связанных длинным кабелем, будут иметь различный потенциал, поэтому ток заземления будет проходить по соединительному кабелю. В отсутствие изоляции этот ток может создать дополнительные шумы, ухудшить качество канала или даже вывести из строя компоненты системы.

Токи, наводимые в длинных кабелях в условиях промышленности, например, при включении и выключении мощных электромоторов, при электростатических разрядах или при разрядах молнии, могут вызвать быстрые изменения потенциала заземления, величиной в сотни или тысячи вольт. При этом на информационный сигнал, передаваемый по каналу, накладывается высоковольтный импульс. При отсутствии изоляции этот высоковольтный импульс может нарушить передачу сигнала или даже вывести систему из строя. Подключение всех устройств, связанных общим интерфейсом, к одному заземлению сможет защитить систему от таких разрушающих воздействий, а изоляция устройств друг от друга позволяет избавиться от замкнутых "петлевых" заземлений.

3.5.10.1 DC/DC преобразователи

DC/DC преобразователи предназначены для преобразования одного уровня напряжения в другой. Преобразователи, имеющие гальваническую изоляцию, можно использовать для питания элементов гальванической изоляции интерфейсов.

3.5.10.2 Реализация гальванической изоляции дискретного выхода модуля ввода-вывода SDX-09

Дискретные выходы в модуле ввода-вывода SDX-09 реализованы на твердотельных реле (Solid State Relay, SSR) CPC1035N и позволяют коммутировать на выходе до 300В с максимальным током нагрузки 100 мА. Управление дискретными выходами программное с помощью линий DOUTx (где x – (0..8) номер дискретного выхода). Подача логического нуля на линии DOUT приводит к замыканию контактов соответствующего дискретного выхода. При подаче нуля на вход P5 на диоде D56 появляется разность потенциалов, достаточная для его зажигания.

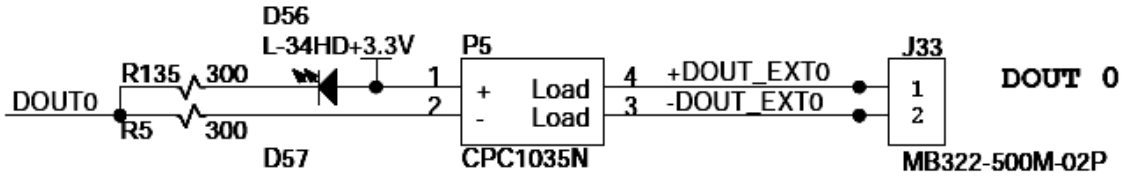


Рис. 48. Реализация гальванической изоляции дискретного выхода модуля ввода-вывода SDX-09.

Основными отличиями твердотельных реле от электромеханических являются:

- Отсутствие электромагнитных помех в момент переключения.
- Высокое быстродействие.
- Отсутствие акустического шума.
- Отсутствие дребезга контактов реле.
- Высокое сопротивление изоляции между входом и выходом.
- Большое количество переключений, не менее 109 раз.
- Малое энергопотребление.

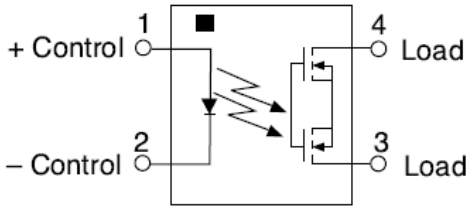


Рис. 49. Твердотельное реле CPC1035N

Напряжение изоляции CPC1035N – 1500В.

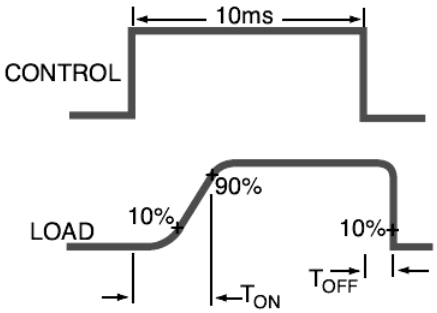


Рис. 50. Временные характеристики переключения твердотельного реле: Ton = 2мс, Toff = 1 мс.

Необходимо заметить, что твердотельные реле данного типа срабатывают достаточно медленно. Длительность фронта достигает 2 мс, длительность спада 1 мс.

3.5.10.3 Реализация гальванической изоляции дискретного входа модуля ввода-вывода SDX-09

Дискретные входы выполнены на базе оптронов KPC357NT. При подаче напряжения в диапазоне 0..24 В на дискретные входы на программно доступной линии DIN x (где x – (0..8) номер дискретного входа) формируется сигнал логического нуля.

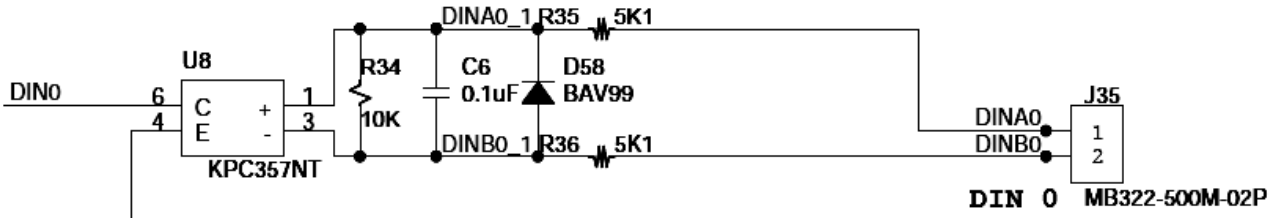


Рис. 51. Реализация гальванической изоляции дискретного входа модуля ввода-вывода SDX-09.

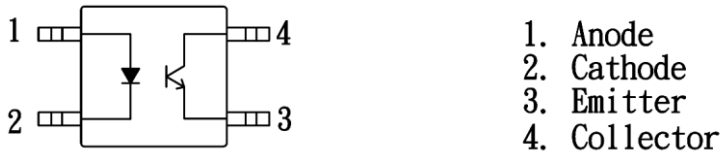


Рис. 52. Оптрон KPC357, состоящий из светодиода и фототранзистора.

Напряжение гальванической изоляции таких оптронов 600В (кратковременно до 3750В).

3.5.10.4 Реализация гальванической изоляции RS-232 в контроллере SDK-1.1

Гальваническая изоляция нужна для защиты ядра вычислительной системы от помех, от разности напряжений при коммутации (установке оборудования). Реализуется с помощью трансформаторной изоляции или с помощью оптоэлектронной схемы. Недостаток трансформаторов состоит в том, что они работают только на переменном токе. Оптоэлектронные схемы (оптопары) состоят из светоизлучающих приборов (диоды) и фотоприёмников (фоторезисторы, фототранзисторы). Оптопары работают хорошо только на

полярном подключении, что неудобно при передаче аналоговых сигналов. Гальваническая изоляция позволяет защитить SDK-1.1 от высоких напряжений, различных наводок и подключать его к ПК во время работы.

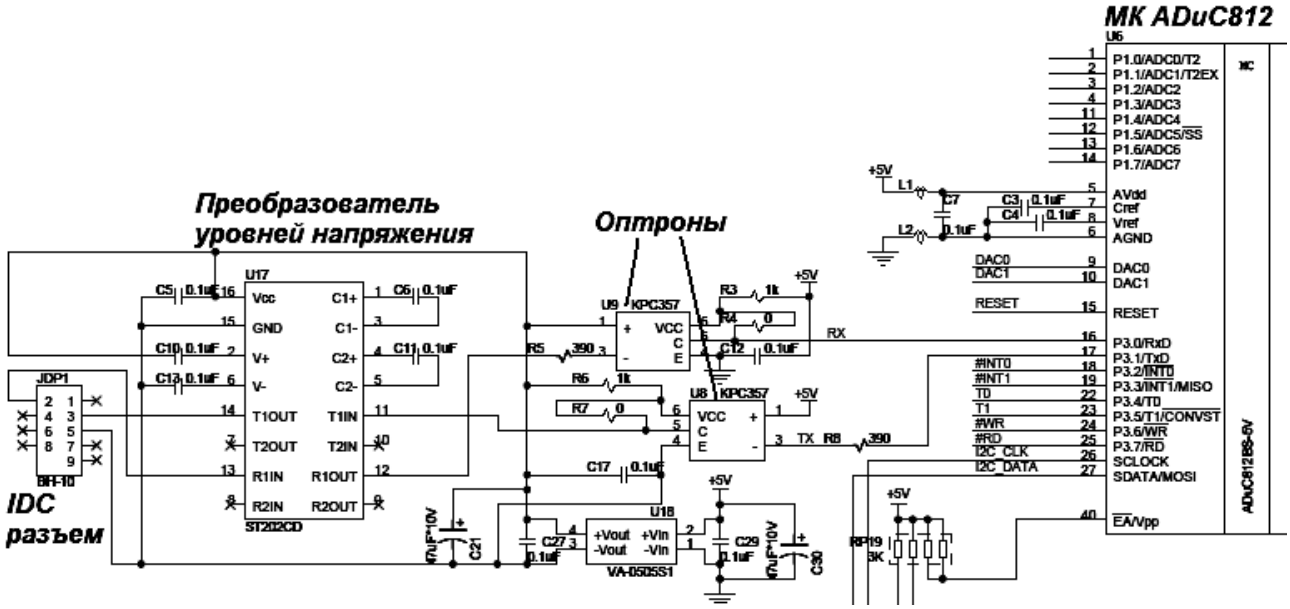
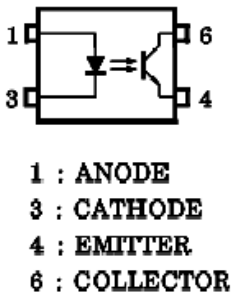


Рис. 53. Гальванически изолированный последовательный интерфейс SDK-1.1.



Реализована гальваническая изоляция на базе двух оптронов U8 и U9 (KPC357). Оптрон KPC357 состоит из светодиода (выводы 1,3) и фототранзистора (выводы 6,4). Если через светодиод пустить ток, то он начинает излучать свет. Свет падает на PN-переход фототранзистора и открывает его. Когда свет гаснет, фототранзистор закрывается. Гальваническая изоляция достигается как раз за счет того, что между двумя элементами оптрона нет никакой связи, кроме оптической.

VA-0505S1 (U18) – DC/DC преобразователь из 5 вольт в 5. Напряжение гальванической изоляции – 1000В (кратковременно до 3000В). На входы +Vin и –Vin поступает напряжение с внутренней шины питания SDK-1.1. С выходов +Vout и –Vout снимается напряжение для питания внешних цепей, находящихся за пределами барьера гальванической изоляции. Сигнальные линии Tx и Rx проходят через оптроны KPC357 (U8, U9).

Необходимо заметить, что оптроны срабатывают не мгновенно. В данном типе оптронов длительность фронта (t_r) и спада (t_f) выходного импульса может быть от единиц до десятков мкс.

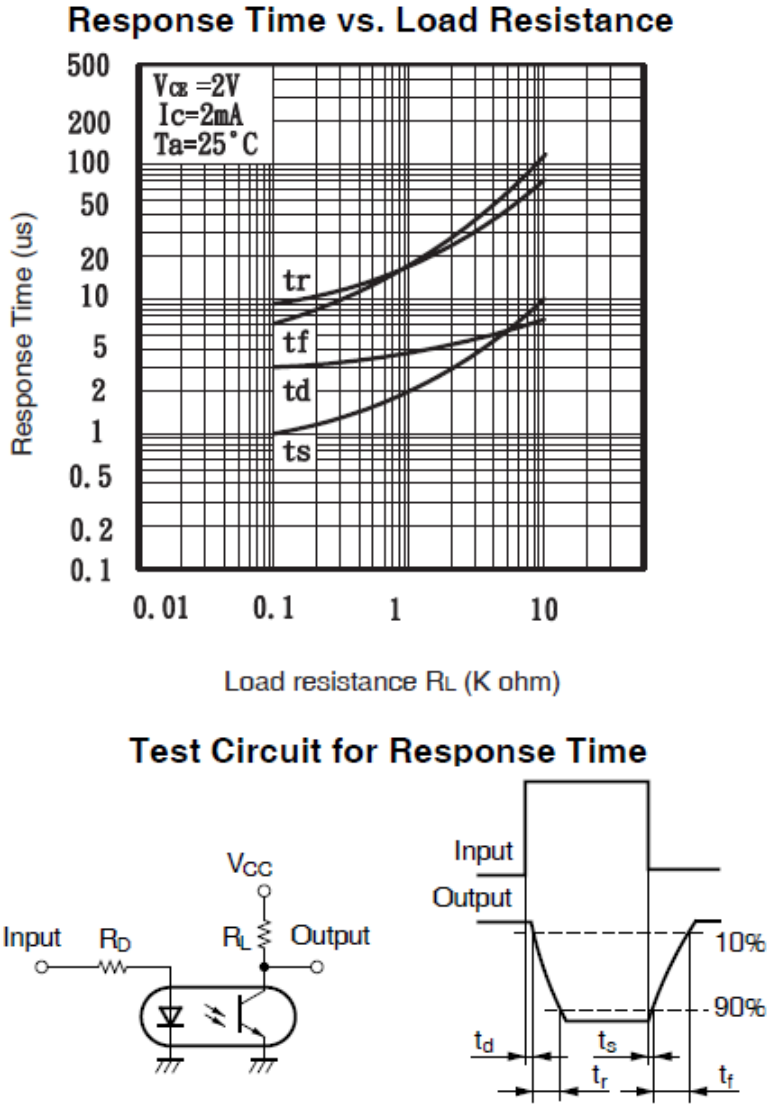


Рис. 54. Зависимость времени реакции оптрона от сопротивления нагрузки

Именно поэтому в учебных стендах SDK-1.1 скорость передачи данных по RS-232C ограничена скоростью 19200 бит/с. Длительность одного бита при такой скорости Тбит = 1/19200 = 52 мкс. Так как суммарная длительность фронта и спада (время реакции) у данного типа оптронов может достигать 40 мкс, на больших скоростях оптроны просто не будут успевать срабатывать.

ST202CD (U17) – приёмопередатчик, преобразующий уровень TTL (0..+5В) в уровни стандарта RS-232C.

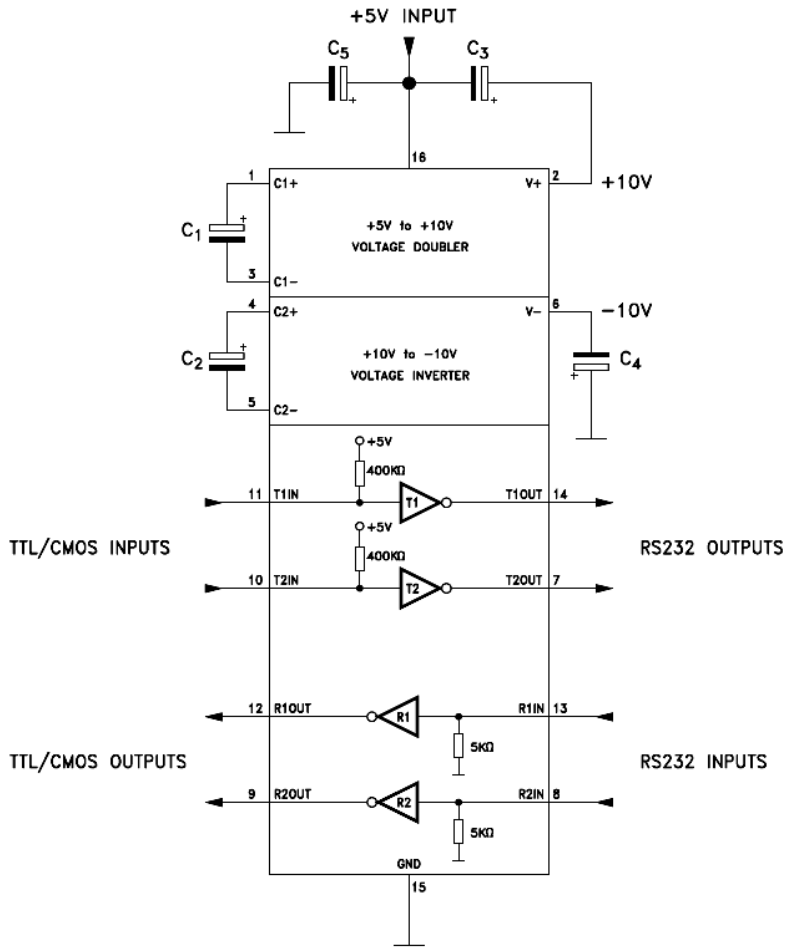


Рис. 55. Структурная схема приемопередатчика ST202

3.5.10.5 Технология iCoupler фирмы Analog Devices

Изоляторы iCoupler – это устройства гальванической развязки на основе трансформаторов, выполненных на кристалле кремния; эти трансформаторы играют ту же роль, что и пара светодиод/фотодиод в оптопаре. Планарный трансформатор изготовлен в ходе технологического процесса КМОП на этапе металлизации и имеет ещё один дополнительный слой осажденного золота. Одну «обмотку» трансформатора от другой изолирует слой электрически прочного синтетического полимера (полиимида). Эти две «обмотки» подключены к быстродействующим КМОП-схемам, которые обеспечивают интерфейс между трансформатором и внешними сигналами. Микроэлектронная технология дает возможность с минимумом затрат осуществить интеграцию нескольких каналов цифровой изоляции и других электронных схем в одном корпусе. Устройства iCoupler не имеют таких присущих оптопарам недостатков, как неопределенный коэффициент передачи тока, нелинейная передаточная функция и дрейф (температурный и временной); кроме того, устройство iCoupler позволяет уменьшить энергопотребление на 90% и для его работы не требуется внешних драйверов и дискретных компонентов.

Электрическая схема, подключенная к первичной «обмотке» трансформатора, преобразует переходы входного сигнала в импульсы длительностью 1 нс, эти импульсы подаются на трансформатор; схема, подключенная ко вторичной «обмотке», принимает эти импульсы и восстанавливает входной сигнал. Схема обновления сигнала (refresh) на входной стороне обеспечивает корректность выходного сигнала, даже если входной сигнал не меняет свое состояние. Это важно в ситуации включения питания, а также при передаче данных с низкой скоростью или при передаче постоянного сигнала.

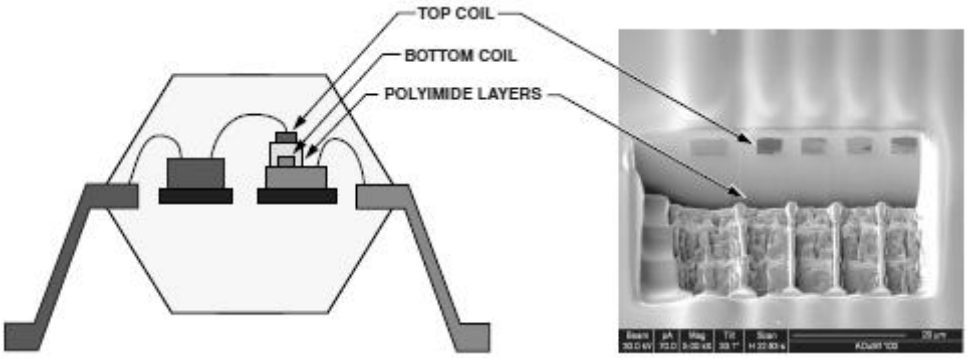


Рис. 56. Устройство и поперечное сечение изолятора iCoupler.

Так как назначение устройства iCoupler заключается в изоляции входа от выхода, входная и выходная схемы располагаются на различных кристаллах. Собственно трансформатор может быть расположен или на одном из этих кристаллов, или на третьем кристалле. Все кристаллы располагаются в стандартном пластиковом корпусе, в таких корпусах выпускаются многие современные микросхемы.

Особенностью многоканальных устройств iCoupler является наличие в одном корпусе каналов на передачу и на прием. Сами трансформаторы могут передавать сигнал в любую сторону, направление определяется схемами, подключенными к трансформатору. Поэтому многоканальные изоляторы поставляются с различными конфигурациями (с различными сочетаниями направлений передачи).

Примеры реализации гальванической изоляции различных интерфейсов при помощи изоляторов iCoupler демонстрируется на рисунках ниже.

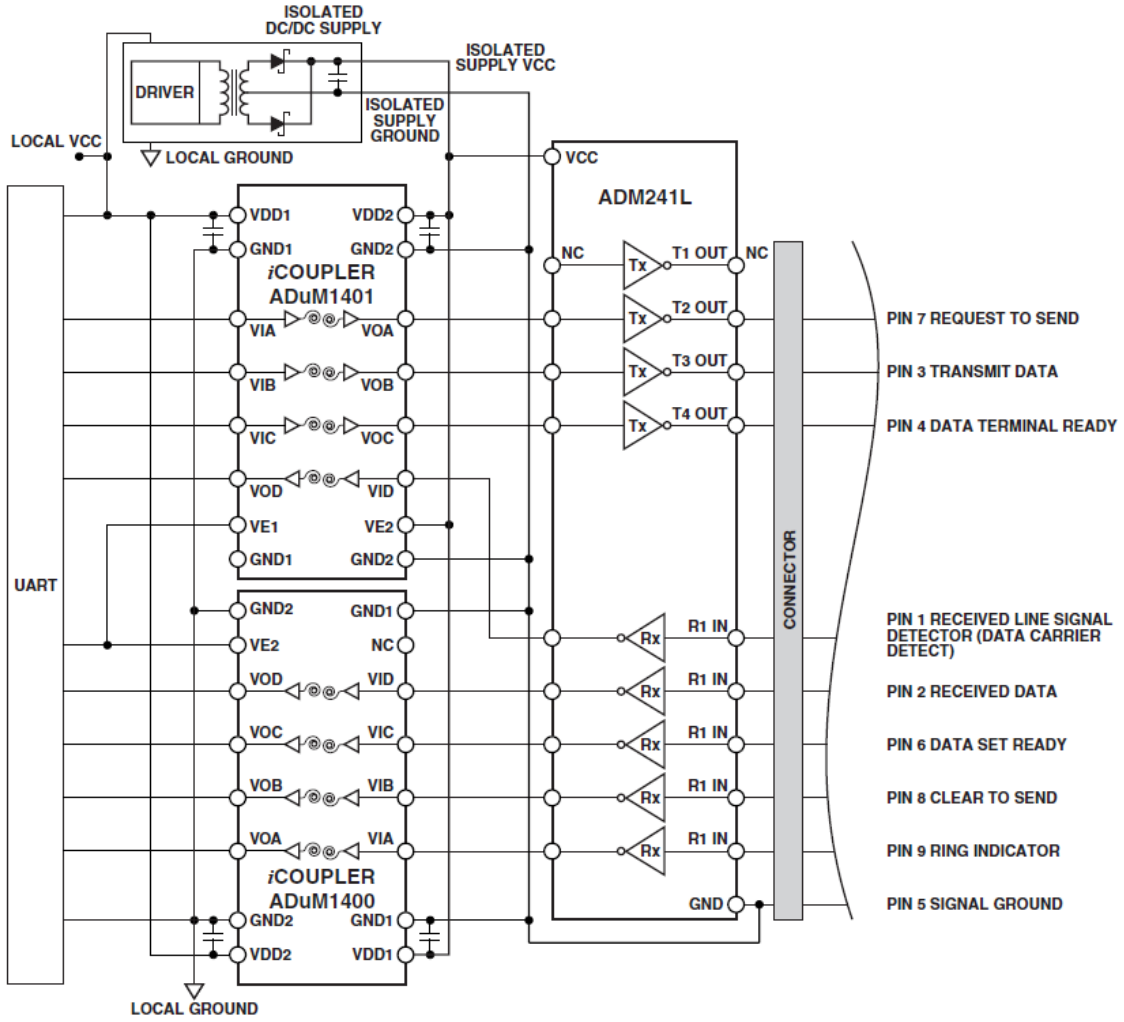


Рис. 57. Использование микросхемы ADuM1400 для реализации гальванической изоляции в интерфейсе RS-232.

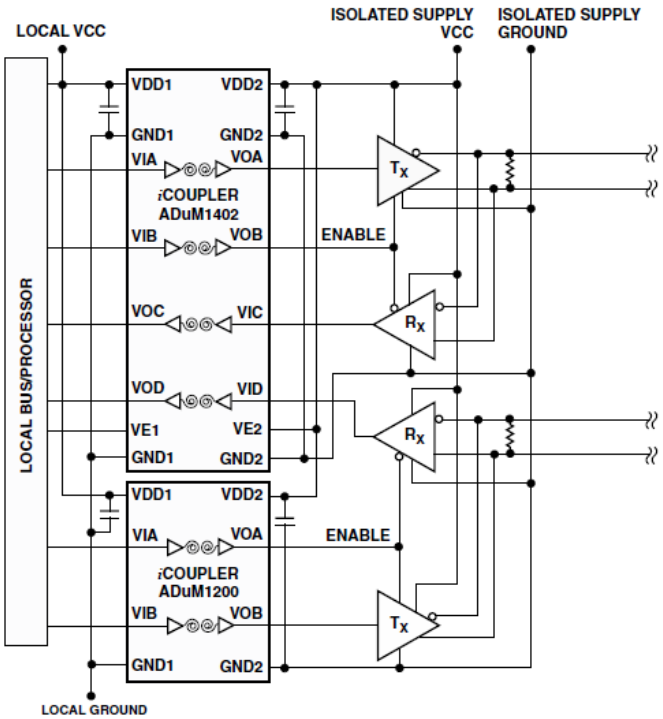


Рис. 58. Реализация гальванической изоляции в интерфейсе RS-485 с полным дуплексом.

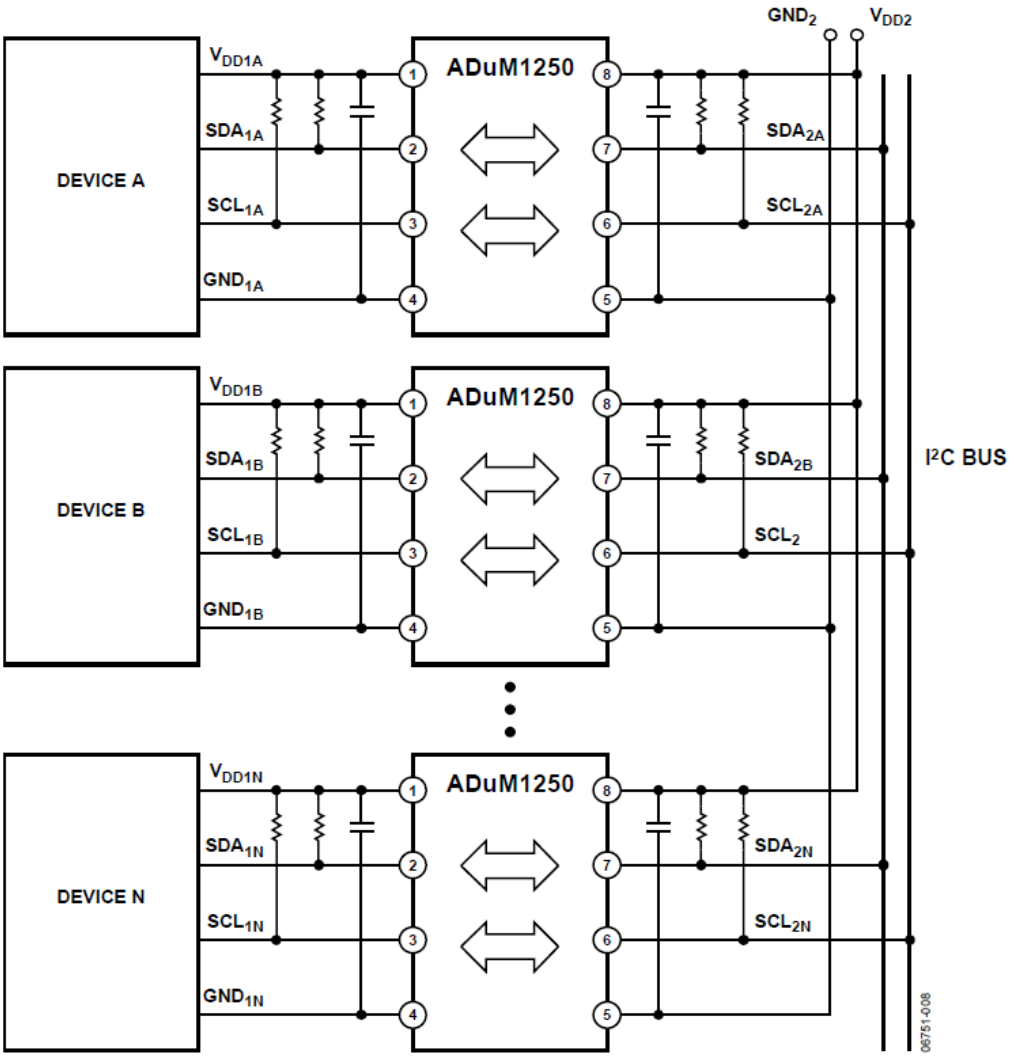


Рис. 59. Гальваническая изоляция интерфейса I2C на базе ADuM 1250.

3.5.11 Горячее подключение и автоконфигурирование

3.5.11.1 Горячее подключение

Горячее подключение (HotPlug) и горячая замена (HotSwap) – замена оборудования в компьютерной системе во время работы (без выключения питания и остановки процессора, системы). Горячее подключение, отключение и замена устройств предназначено для обеспечения следующих свойств интерфейса [38]:

- Во-первых, это безопасность переключений "на ходу" как для самих устройств и их интерфейсных схем, так и для целостности хранящихся и передаваемых данных и, наконец, для человека.
- Во-вторых, это возможность использования вновь подключенных устройств без перезагрузки системы, а также продолжения устойчивой работы системы при отключении устройств.

Далеко не все внешние интерфейсы поддерживают горячее подключение в полном объеме, так, например, зачастую сканер с интерфейсом SCSI должен быть подключен к компьютеру и включен до загрузки ОС, иначе он не будет доступен системе. С новыми шинами USB и Fire Wire проблем горячего подключения не возникает. Для внутренних интерфейсов горячее подключение несвойственно. Это касается и шин расширения, и модулей памяти, и даже большинства дисков ATA и SCSI. Горячее подключение поддерживается для шин расширения промышленных компьютеров, а также в специальных конструкциях массивов устройств хранения.

Горячее подключение обеспечивают интерфейсы: USB, FireWire, PCMCIA, PCI Express, Fibre Channel, eSATA, Bluetooth, ZigBee.

В большинстве современных устройств, допускающих горячую замену, используются дополнительные контакты сделанные таким образом, чтобы быть длиннее остальных и первыми входить в контакт с присоединяемой частью. Остальные контакты делаются короче, всего может быть несколько различных длин. Задержка между подключением первого контакта и последующих составляет от 25 до 250 миллисекунд.

Цепи питания подключаются в две стадии, в первой из которых с помощью более длинных контактов подключается цепь, ограниченная по току, а затем более короткими – питание полной мощности. Все цепи, участвующие в соединении, содержат защиту от статического электричества.

Вот пример типичной последовательности подключения:

1. Замыкаются наиболее длинные контакты (заземление). Тем самым достигается электрическая безопасность соединения и защита от статического заряда.
2. Замыкаются длинные или средние контакты предварительного питания. Заряжаются входные контуры цепей питания.
3. Задержка в десятки миллисекунд.
4. Подключаются короткие контакты питания.
5. Соединение считается установленным. Включается сигнал инициализации питания.
6. Цепь мягкого включения питания подает напряжение на устройство.
7. Задержка в десятки миллисекунд.
8. Цепь питания закончила мягкое подключение. Выключается сигнал инициализации питания.
9. Устройство начинает полноценную работу.

Особую трудность представляет соединение нескольких устройств. Подсоединение второго, третьего устройства может нарушать работу уже подключенного. Для борьбы с этим явлением используют фильтры в выходных цепях или временное логическое отключение передачи данных.

3.5.11.2 Plug and Play

Plug and Play (PnP) – технология, предназначенная для быстрого определения и конфигурирования устройств в компьютере и других технических устройствах.

В ряде интерфейсов заложены возможности PnP, которые предназначены для снятия с пользователей забот по конфигурированию подключаемых устройств. В современных интерфейсах эти возможности закладывались изначально (PCI, USB, Fire Wire, Bluetooth), и эти функции в большинстве случаев работают нормально. Однако для интерфейсов-ветеранов (например, ISA, SCSI) технология PnP является поздней искусственной надстройкой, работающей с переменным успехом (Plug and Pray - включай и молись). Часто побочные эффекты вызваны наследием "тяжелого прошлого" - соседством устройств PnP с традиционными (legacy) устройствами. На закате шины ISA ее система PnP в общем работала, но в SCSI от идей автоконфигурирования со временем отказались. При разработке собственных устройств встает вопрос выбора подходящего интерфейса подключения. Этот вопрос следует решать, исходя из принципа разумной достаточности, по возможности отдавая предпочтение внешним интерфейсам. Следует помнить, что разработка аппаратной части устройства (hardware) тесно связана и с программной поддержкой устройств – как модулями ПО, исполняемыми процессором компьютера (software), так и программами встроенного микроконтроллера (firmware), на базе которого, как правило, строятся современные устройства. Промышленностью выпускается множество моделей микроконтроллеров, имеющих популярные интерфейсы (USB, RS-232, PC и другие). Однако в ряде случаев приходится использовать и стандартизованные шины расширения ввода-вывода. Эти шины предоставляют более широкие возможности для взаимодействия процессора с аппаратурой, нескованные жесткими ограничениями внешних интерфейсов. Однако за универсальность и производительность внутренних шин расширения приходится расплачиваться более замысловатой реализацией интерфейсных схем и сложностями при обеспечении совместимости с другим установленным в компьютер оборудованием. Здесь ошибки могут приводить к потере работоспособности компьютера (хорошо если временной). Недаром серьезные производители компьютеров гарантируют работоспособность своих изделий только при установке сертифицированных (ими или независимыми лабораториями) карт расширения. При использовании внешних интерфейсов неприятности в случае ошибок чаще всего имеют отношение только к подключаемому устройству.

3.6 Внутрисистемный интерфейс AMBA

AMBA (Advanced Microcontroller Bus Architecture) – шина, разработанная фирмой ARM для организации эффективного взаимодействия компонентов устройств, построенных на базе ядер фирмы. Шина AMBA – стандартная встроенная ASIC-шина, обеспечивающая быстрое модульное проектирование систем при упрощении многократного использования схемотехники и тестов.

ARM также обеспечивает возможность использования библиотеки PrimeCell периферии, которая соответствует AMBA стандарту и обеспечивают простую разработку ASIC и ASSP. При использовании AMBA с синтезируемыми версиями периферийных устройств, аппаратные средства системы и программное обеспечение могут быть разработаны на начальном этапе проектирования и, следовательно, может быть снижен риск ошибок проектирования конечной системы [6].

Согласно спецификации AMBA Rev 2.0 (AMBA Specification (Rev 2.0)) типовая шина AMBA 2 содержит высокоскоростную системную магистральную шину (AHB или ASB) и шину периферии (APB).

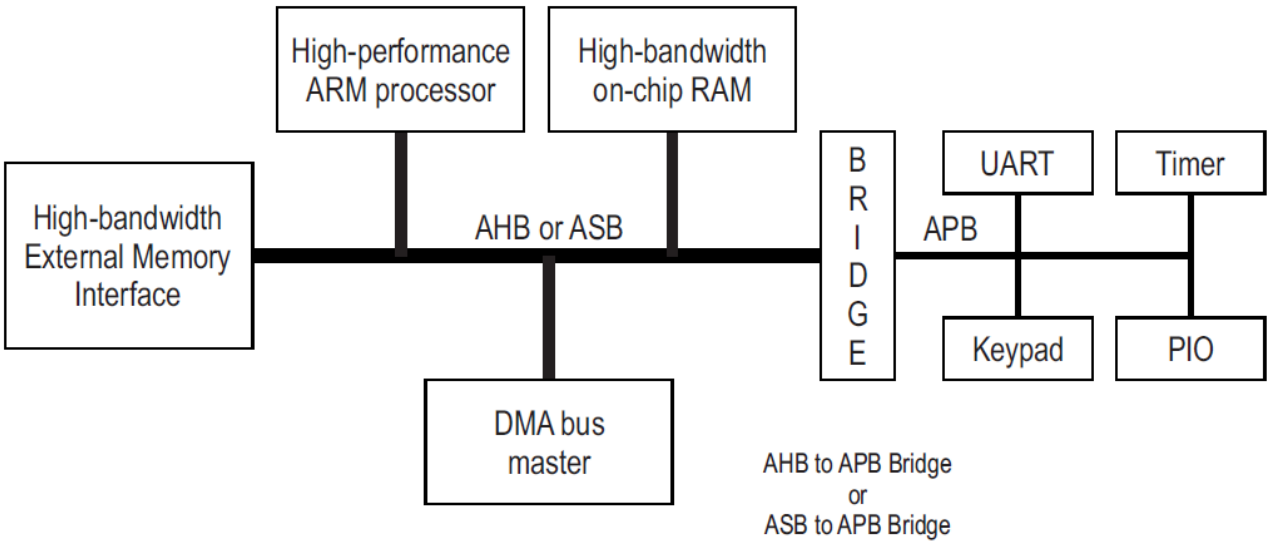


Рис. 60. Типовая вычислительная система на базе AMBA.

Системная шина соединяет встраиваемые процессоры, такие как ARM-ядра, с высокопроизводительной периферией, контроллерами DMA, встроенными памятью и интерфейсами.

Шина периферии работает с упрощенным протоколом и разработана для организации интерфейса с периферийными устройствами общего назначения или дополнительными периферийными устройствами. С системной шиной она соединяется через мост (bridge), способствующий снижению потребления системы.

В спецификации шины AMBA 2 определена методология тестирования, обеспечивающая быстрое тестирование модулей и кэш.

Фирмой ARM разработан набор макроячеек периферийных компонентов, которые фирма на основе лицензионных соглашений предоставляет заказчикам. Периферийные компоненты фирмы ARM, библиотека которых получила наименование PrimeCell, представляют собой готовые к применению программные макроячейки (IP-блоки), при разработке которых обращалось внимание на возможность многократного их использования. Применяя PrimeCell периферию разработчик существенно экономит время и стоимость

разработки за счет концентрации усилий на создании именно системы на кристалле, а не на разработке сначала необходимой периферии и лишь затем системы. В настоящее время в библиотеку входят:

- Контроллеры статической памяти (SRAM).
- Контроллеры динамической памяти (DDR, DDR2).
- Контроллеры прямого доступа к памяти.
- Контроллеры прерываний (VIC, Advanced VIC).
- UART, синхронные последовательные интерфейсы (SPI), часы реального времени, средства ввода-вывода общего назначения (GPIO), интерфейсы смарт-карт, контроллеры цветных ЖКИ.

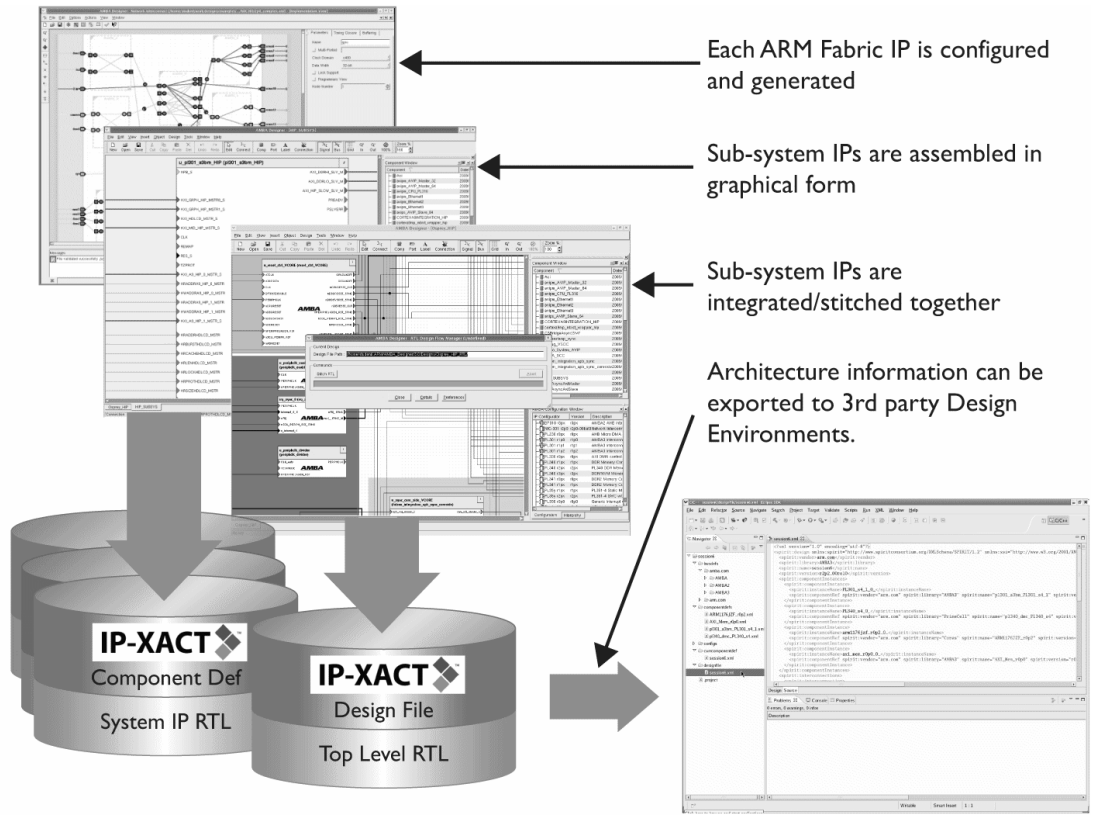


Рис. 61. Инструментальная система CoreLinkT AMBA Designer, предназначенная для разработки микроконтроллеров и СнК на базе ядра ARM с шинами AMBA AHB и AMBA APB.

Ведутся работы по дальнейшему расширению библиотеки.

Производительность устройств класса "система-на-кристалле" (СнК) в значительной мере зависит от эффективности взаимодействия всех встроенных компонентов и от эффективности их взаимодействия с внешним, относительно прибора, миром. В первую очередь это связано с различием в быстродействии встроенных компонентов, в особенностях организации интерфейсов.

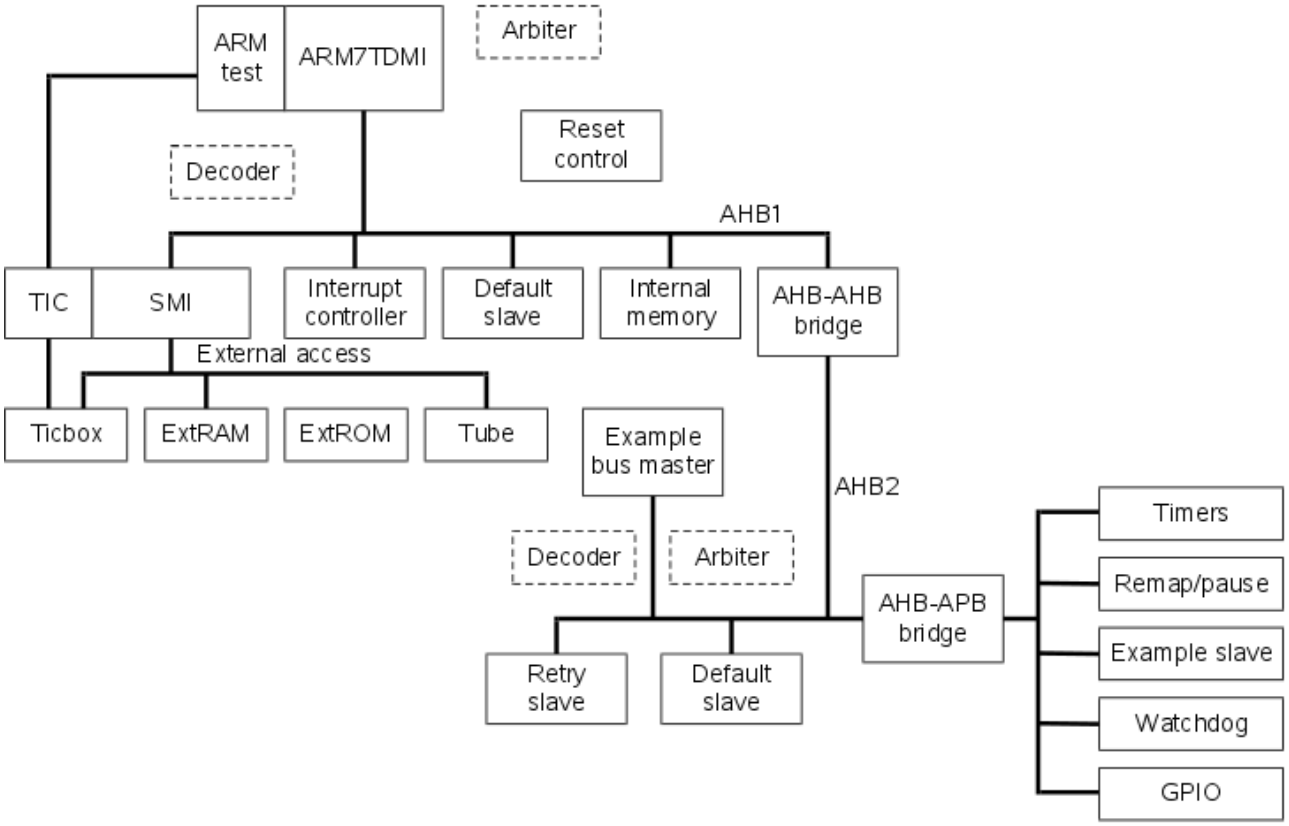


Рис. 62. Пример простой системы на базе ARM7 с шинами AHB и APB, прилагаемый к CoreLinkT AMBA Designer.

Для разработки микроконтроллеров и СнК на базе AMBA существуют специальные инструментальные средства. Например, CoreLinkT AMBA Designer [5] позволяет скомпоновать СнК из готовых IP компонентов. В состав IP-компонентов входят:

- Контроллер памяти.
- Арбитр.
- Матрица.
- Подчиненный контроллер (32 и 64 бита).
- Тестовый контроллер.
- Мост AHB-APB.
- Контроллер SRAM.
- Подчиненный контроллер для APB.
- Сторожевой таймер для APB.
- Таймеры для APB.
- Другие IP-компоненты.

3.6.1 Внутрисистемный интерфейс AMBA AHB

Внутрисистемный интерфейс AHB (Advanced High-performance Bus – расширенная высокопроизводительная шина) предназначен для объединения быстродействующих, высокопроизводительных модулей, работающих на высоких тактовых частотах.

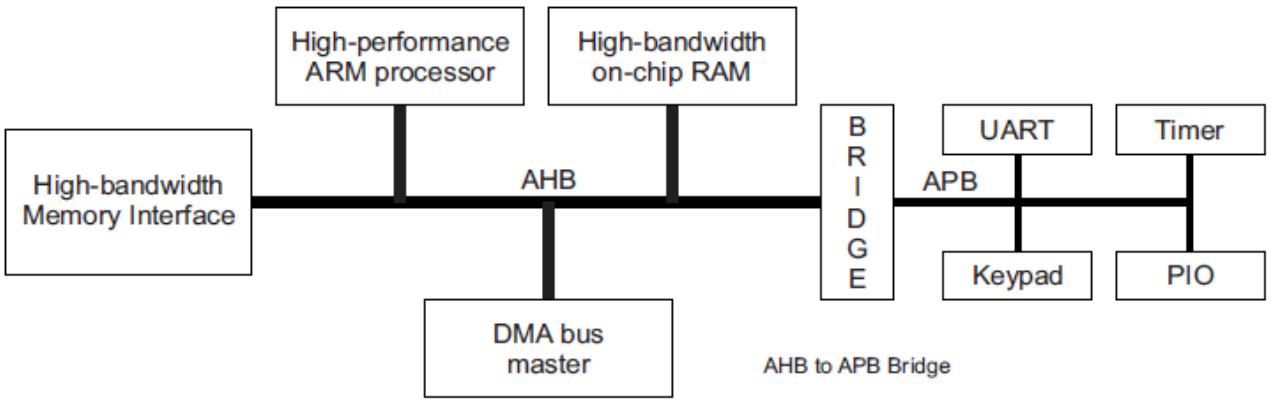


Рис. 63. Схема типичного микроконтроллера на базе шины AMBA-AHB.

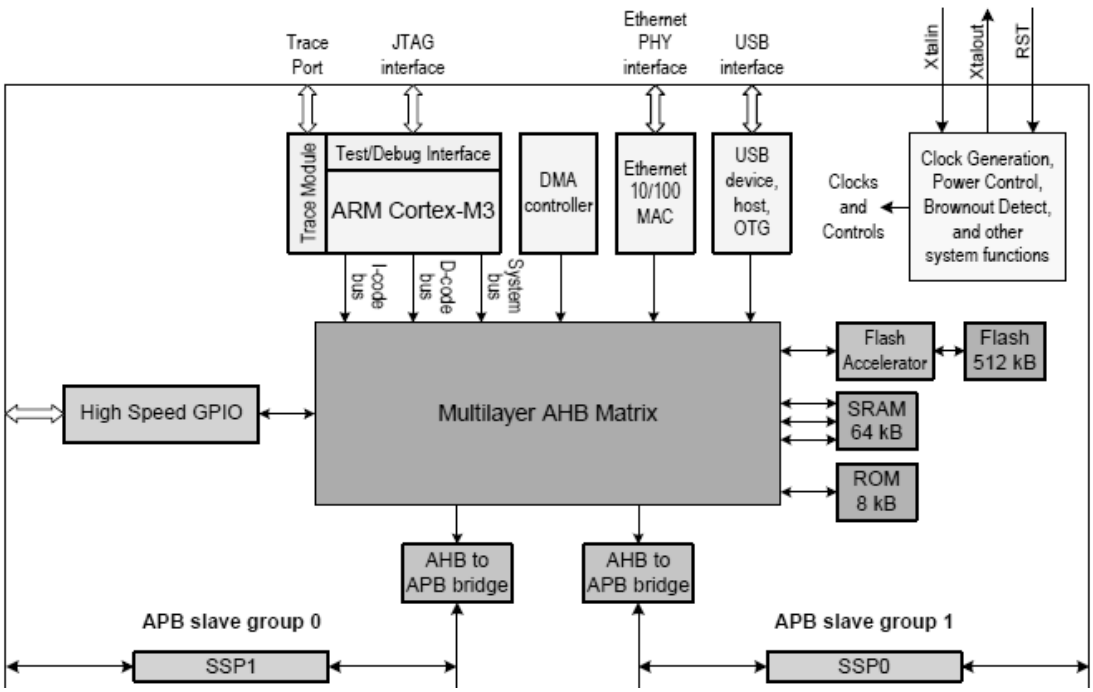


Рис. 64. Пример использования шины AMBA AHB в микроконтроллере NXP LPC1768 на базе Cortex-M3.

AHB позволяет соединить ядро микропроцессора с внутренними и внешними модулями памяти, контроллерами ПДП, быстродействующими сетевыми контроллерами Fast Ethernet, контроллерами USB-2.0 и т.д.

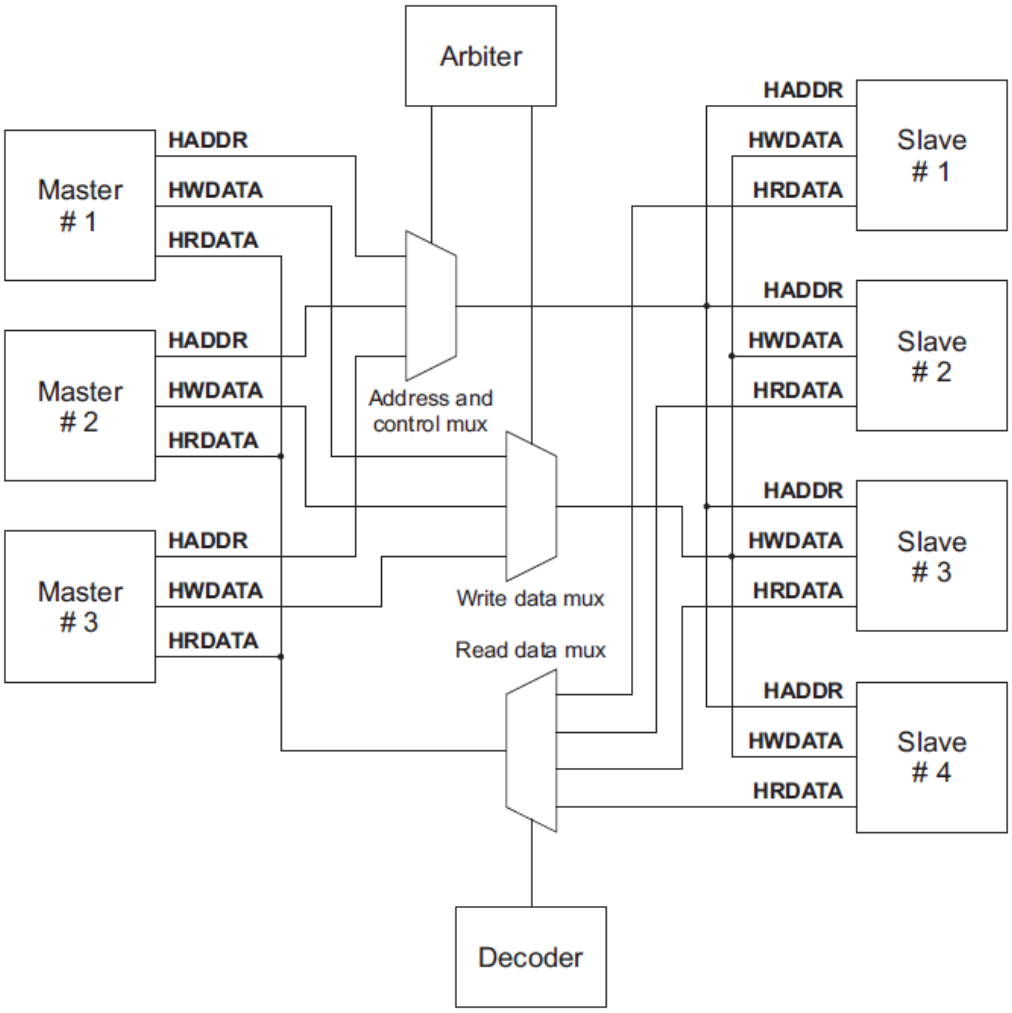


Рис. 65. Схема взаимодействия главных и подчиненных устройств, подключенных к шине AMBA AHB.

На рисунке показан фрагмент структуры микроконтроллера NXP LPC1768, выполненного на базе ядра ARM Cortex-M3. На схеме видно, что шиной AHB объединены вместе практически все элементы микроконтроллера, требующие большой скорости обмена данными. Через два моста, к шине AHB подключены две независимые шины APB, к которым подключаются сравнительно медленные периферийные устройства (UART, I2C, таймеры и так далее).

На рисунке шина AHB названа Matrix (матрица). Такое название справедливо, так как на самом деле шина является коммутатором, матрицей, связывающей несколько устройств попарно друг с другом в различные моменты времени. На рисунке ниже можно посмотреть реализацию шины AHB.

Мы видим, что арбитр шины управляет мультиплексорами, к которым подключены шины адреса и шины записи данных из главных устройств (Master) в подчиненные (Slave). Декодер управляет мультиплексором, через который считываются данные с подчиненных устройств. Хорошо видно, что для записи и чтения существуют раздельные шины.

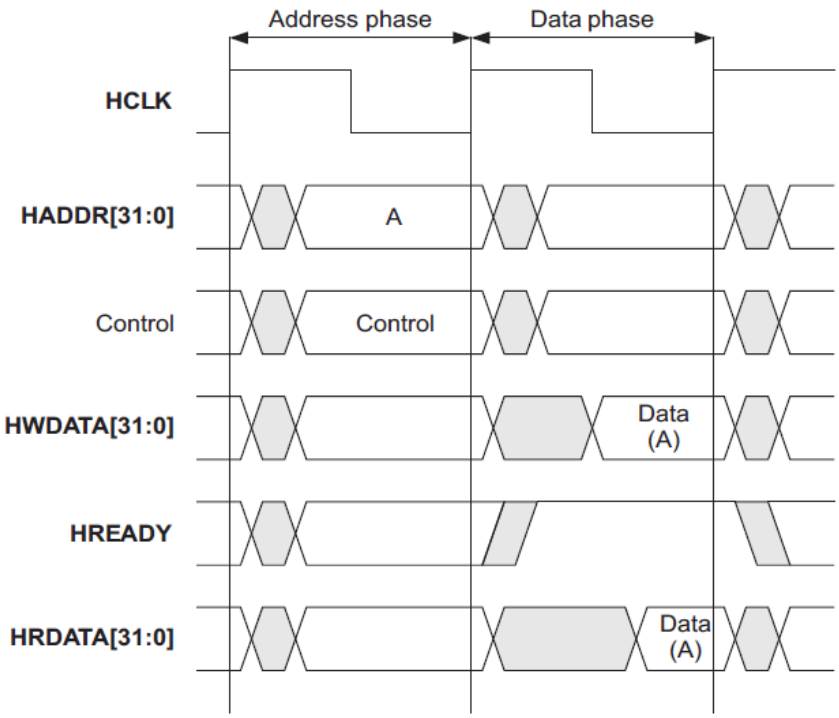


Рис. 66. Простой обмен данными на шине AMBA AHB.

Опишем простой вариант обмена данными на шине AMBA AHB:

1. Главное устройство выставляет адрес и управляющие сигналы по фронту сигнала HCLK.
2. Подчиненное устройство считывает значения с шины адресы (HADDR) и шины управления (Control) на следующем фронте HCLK.
3. После считывания адреса и данных подчиненное устройство может выдать ответ на шину в этом же цикле (сигнал HREADY активен).
4. Главное устройство считывает данные, выставленные подчиненным устройством в третьем цикле шины.

Подчиненное устройство может вставлять такты неготовности, используя сигнал HREADY (см. рисунок выше). Такт неготовности – это такой такт, во время которого обмен по шине не производится вследствие отсутствия готовых данных у одного из устройств.

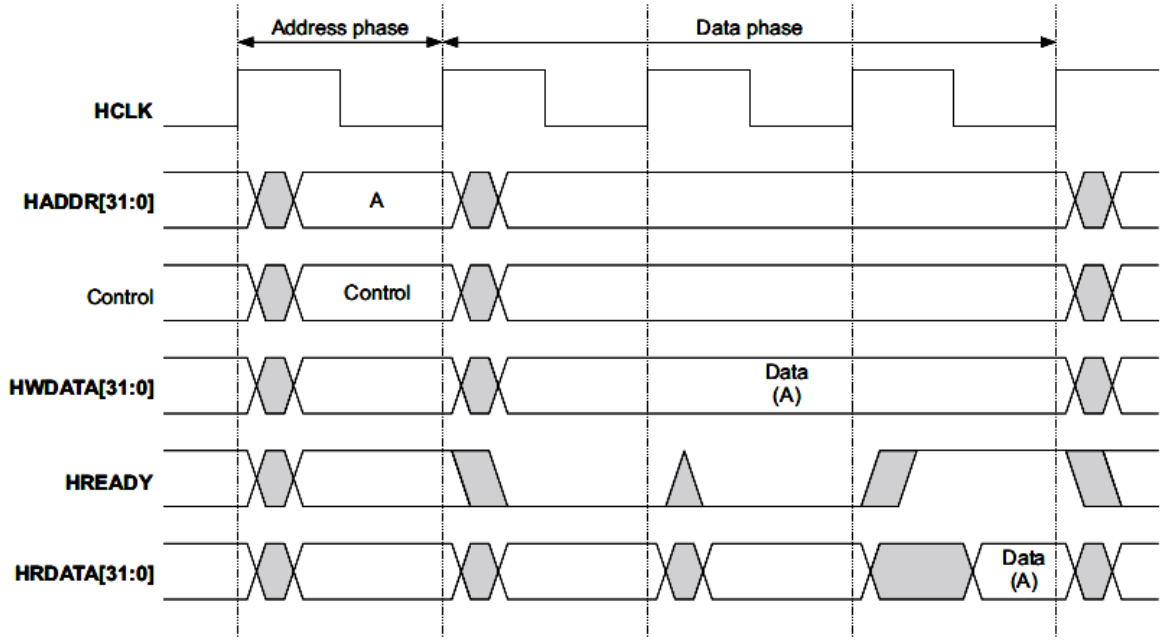


Рис. 67. Передача данных с тактами неготовности.

В отличие от предыдущего рисунка здесь хорошо видно, что во втором и третьем тактах шины сигнал HREADY равен нулю. Такты неготовности используются для того, чтобы синхронизировать быстрое главное устройство и более медленное подчиненное.

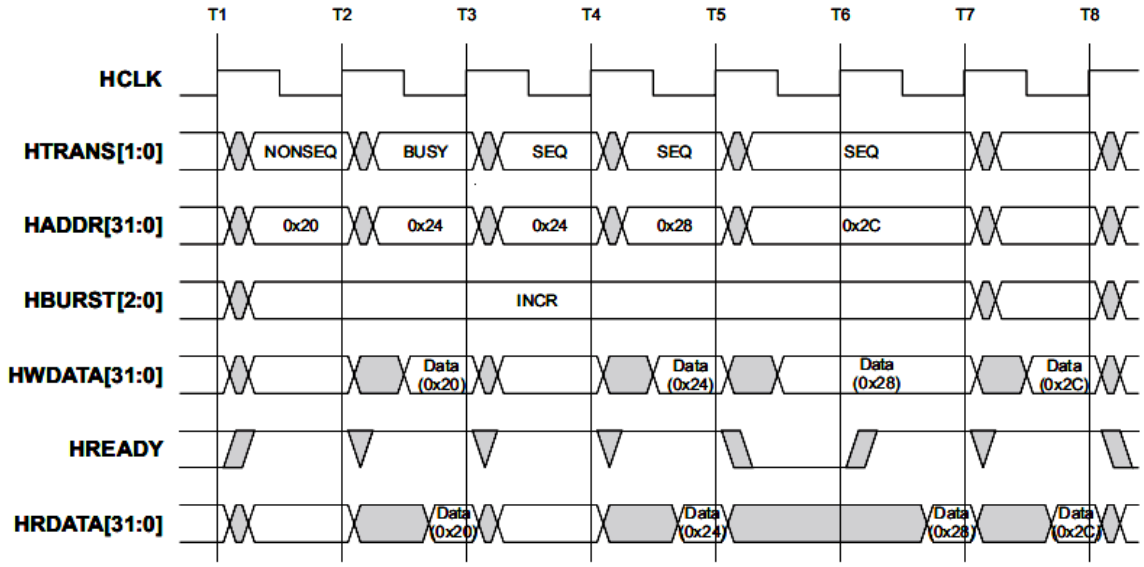


Рис. 68. Пример различных типов обмена на шине.

В зависимости от состояния сигнала HTRANS[1:0] может быть выделено четыре типа передач на шине:

1. IDLE. Передача данных не требуется.
2. BUSY. Позволяет вставлять IDLE циклы в середину пакетной передачи.
3. NONSEQ. Индицирует начало одиночной или пакетной передачи. Адрес установленный в предыдущем цикле шины игнорируется.

4. SEQ. Показывает, что адрес в новом цикле шины инкрементируется на 1, а сигналы на шине управления такие же.

Тип пакетной передачи определяется сигналом HBURST[2:0]. Существует 8 вариантов пакетной передачи:

1. SINGLE (000) – одиночная передача.
2. INCR (001) – инкрементная передача с неизвестной длиной.
3. WRAP4 (010) – циклическое окно на 4 адреса.
4. INCR4 (011) – инкрементная передача на 4 адреса.
5. WRAP8 (100) – циклическое окно на 8 адресов.
6. INCR8 (101) – инкрементная передача на 8 адресов.
7. WRAP16 (110) – циклическое окно на 16 адресов.
8. INCR16 (111) – инкрементная передача на 16 адресов.

Режим WRAP (циклическое окно) позволяет работать циклически, по кругу, в заданном окне адресов. Режим INCR предполагает простой инкремент (прибавление на единицу) адреса в каждом цикле шины.

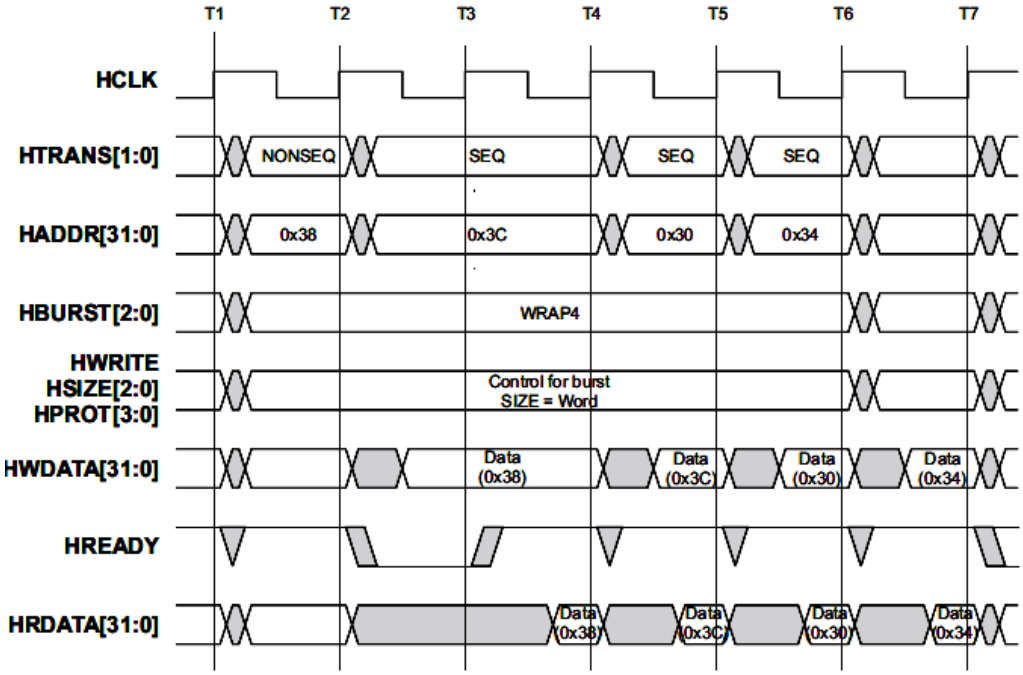


Рис. 69. Пример использования режима WRAP4.

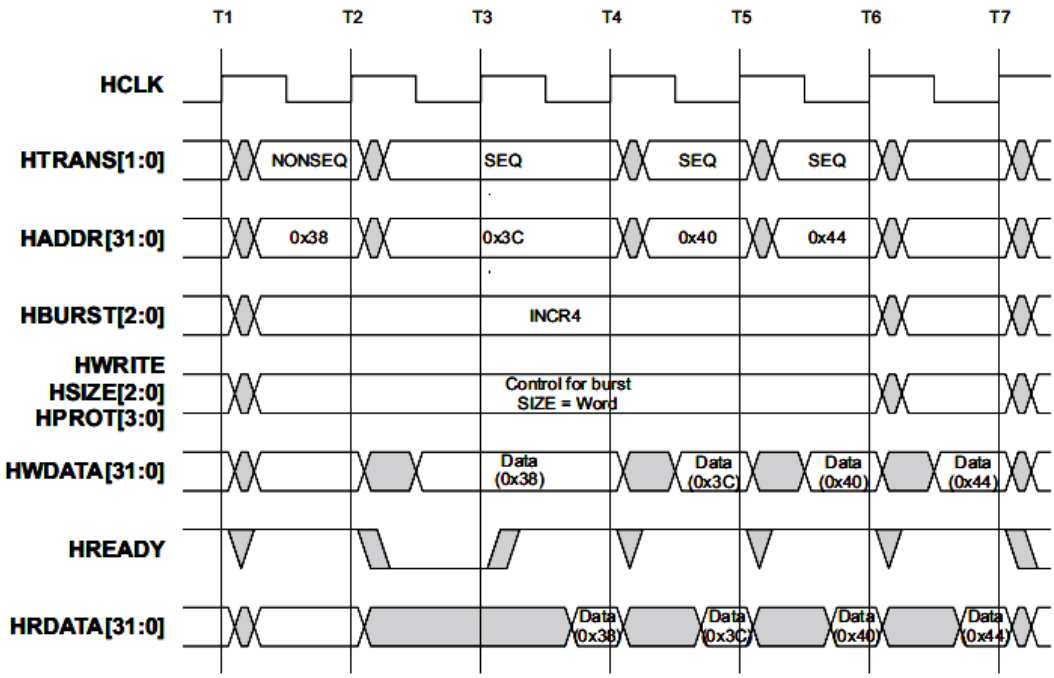


Рис. 70. Пример использования режима INCR4.

При передаче данных сигнал HWRITE становится активным (переводится в логическую «1»). В зависимости от состояния линий HSIZE[2:0] передаваться за один раз может 8, 16, 32, 64, 128, 256, 512 или 1024 бита данных. Эти биты (HSIZE) работают совместно с сигналами HBURST.

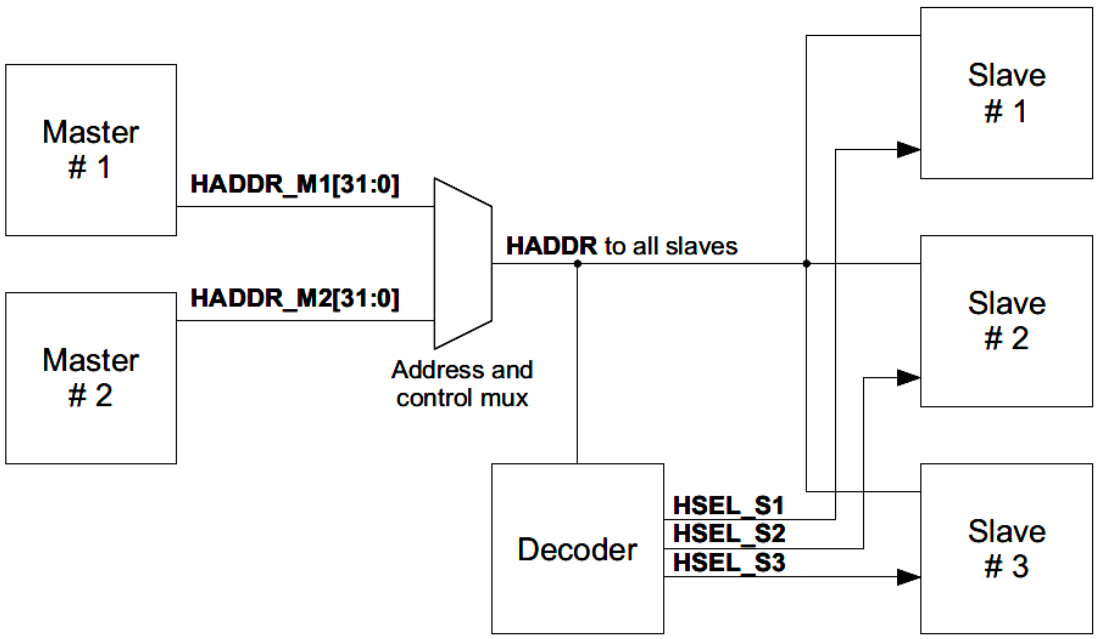


Рис. 71. Выбор подчиненного устройства с помощью сигналов HSEL.

Сигнал HPROT[3:0] определяют уровень защиты передаваемой информации: код операции, доступ к данным, пользовательский доступ, привилегированный доступ и так далее.

Выбор подчиненного устройство происходит с помощью декодера адреса, формирующего сигналы HSEL.

Сигнал HREADY используется для информирования главного устройства о завершении или продолжении передачи данных. Если уровень этого сигнала активный, то передача данных завершена.

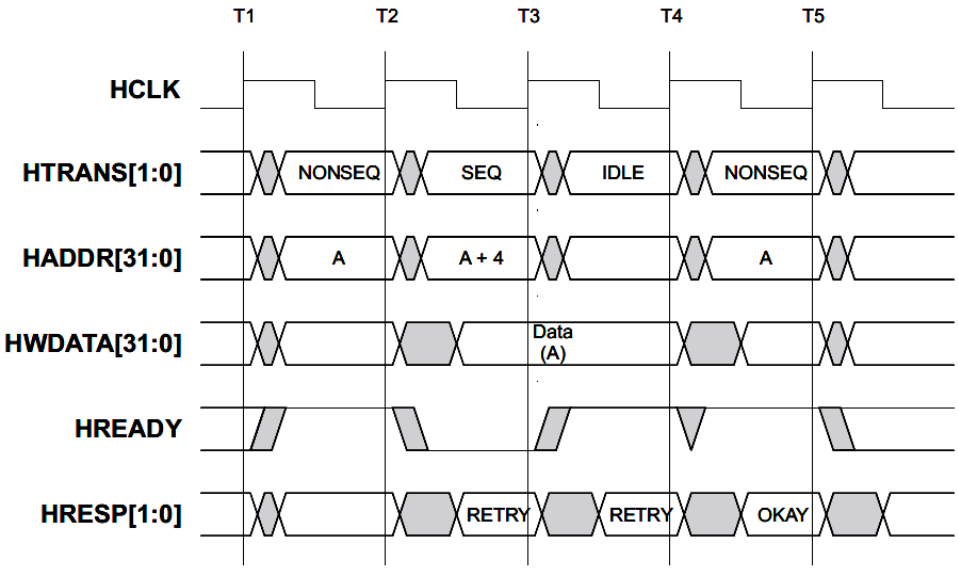


Рис. 72. Пример использования сигнала HRESP.

Сигнал HRESP[1:0] используется для подтверждения транзакции. Его значениями могут быть: OKAY – транзакция завершена успешно, ERROR – ошибка, RETRY – повторить попытку, SPLIT – данные переданы не полностью.

Сигналы HWDATA[31:0] используются для передачи данных от главного устройства к подчиненным.

Сигналы HRDATA[31:0] используются для передачи данных от подчиненных устройств к главным.

Арбитраж используется для того, чтобы только один мастер имел доступ к шине в один момент времени. Рассмотрим основные сигналы:

- HBUSREQx – запрос шины у арбитра шины.
- HLOCKx – сигнал, выставляемый мастером и означающий захват шины.
- HGRANTx – сигнал, выставляемый арбитром и обозначающий, что мастер получил доступ к шине.
- HMASTER[3:0] – сигнал показывающий номер мастера, захватившего шину.

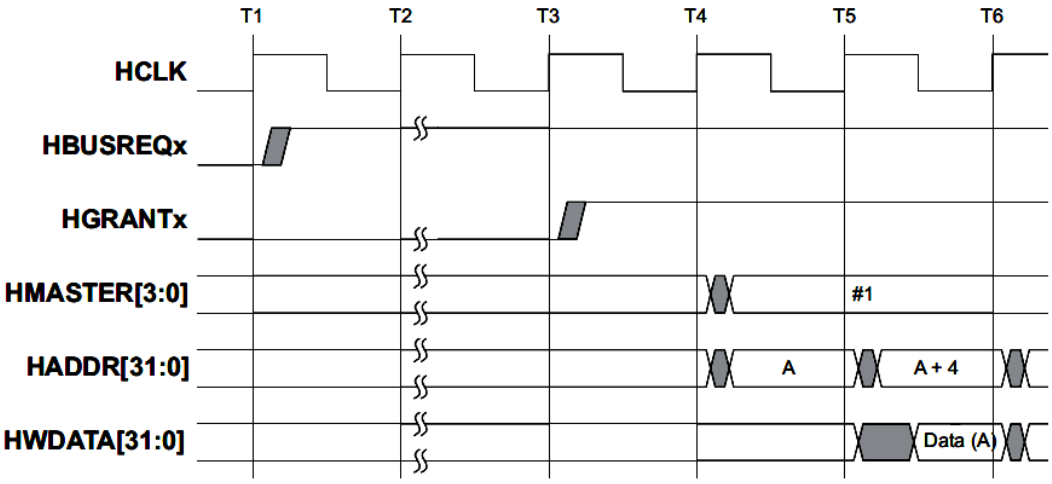


Рис. 73. Пример арбитража.

На рисунке видно, что в первом такте шины главное устройство выставляет сигнал запроса шины HBUSREQx. В третьем такте арбитр подтверждает захват шины сигналом HGRANTx. На четвертом такте арбитр выставляет сигнал HMASTER с номером главного устройства, захватившего шину.

3.6.2 Системный интерфейс AMBA ASB

AMBA ASB (Advanced System Bus) является системным интерфейсом и предназначен для использования в высокопроизводительных 16- и 32-разрядных микроконтроллерах. Интерфейс позволяет связать процессор, встроенную и внешнюю память. В AMBA ASB заложена тестовая инфраструктура. AMBA ASB использовался в микроконтроллерах с процессорными ядрами ARM7TDMI, ARM 920 и ARM940. В настоящее время этот интерфейс используется сравнительно редко, вместо него обычно используют более производительный AMBA AHB.

В AMBA ASB поддерживается множество ведущих устройств и пакетная передача. Шина ASB является более простой, по сравнению с AMBA AHB. Коренными отличиями является двусторонняя шина данных (в AHB для данных есть отдельные шины, предназначенные для записи и чтения), более узкая шина данных (32 разряда), не поддерживается раздельная (SPLIT) передача данных.

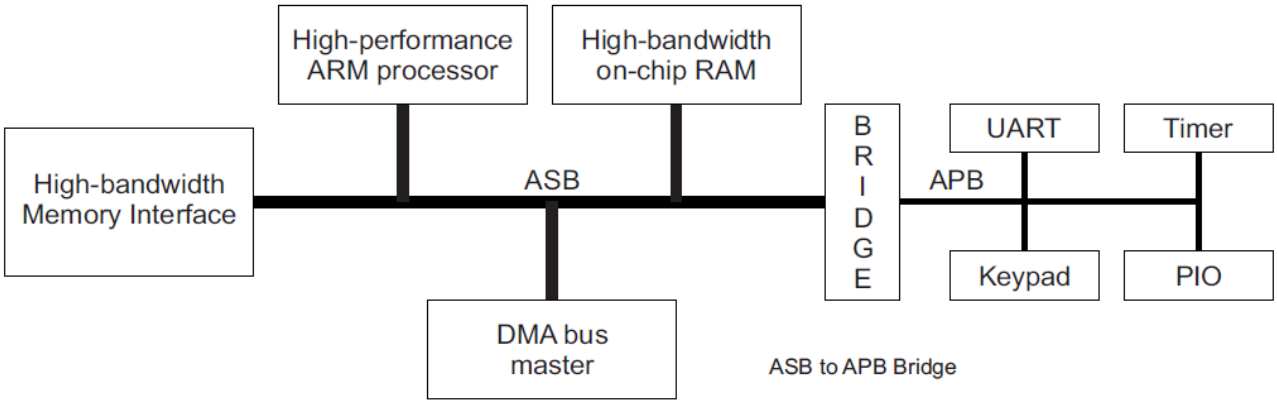


Рис. 74. Типичная система, построенная на базе AMBA ASB.

Система с шиной AMBA ASB (AHB) обычно содержит следующие компоненты:

- ASB-ведущий (мастер). Мастер инициирует операции чтения и записи посредством подачи адреса и управляющих сигналов. Только один мастер в определенный момент времени может быть активным.
- ASB-ведомый (слейв). Ведомый отвечает на операции чтения и записи в заданном адресном пространстве. Ведомый сигнализирует активному мастеру в случае успешного, ошибочного обмена данными или в случае ожидания.
- ASB-дешифратор. Выполняет дешифровку адресов и выбирает соответствующего ведомого. Дешифратор также гарантирует, что шина остается в рабочем состоянии, когда никакого обмена не производится.
- ASB-арбитр. Арбитр гарантирует, что только одному мастеру в данный момент времени позволяется инициировать обмен данными. И хотя протокол разрешения доступа к общей шине зафиксирован, любой алгоритм разрешения конфликтов может быть реализован в зависимости от требований области применения.

В шине возможны три основных состояния:

- NONSEQUENTIAL (N-TRAN)– используется для одиночных передач или первой передачи данных в пакете.
- SEQUENTIAL (S-TRAN) – используется при пакетной передаче данных.
- ADDRESS-ONLY – используется, если нет необходимости в передаче данных.

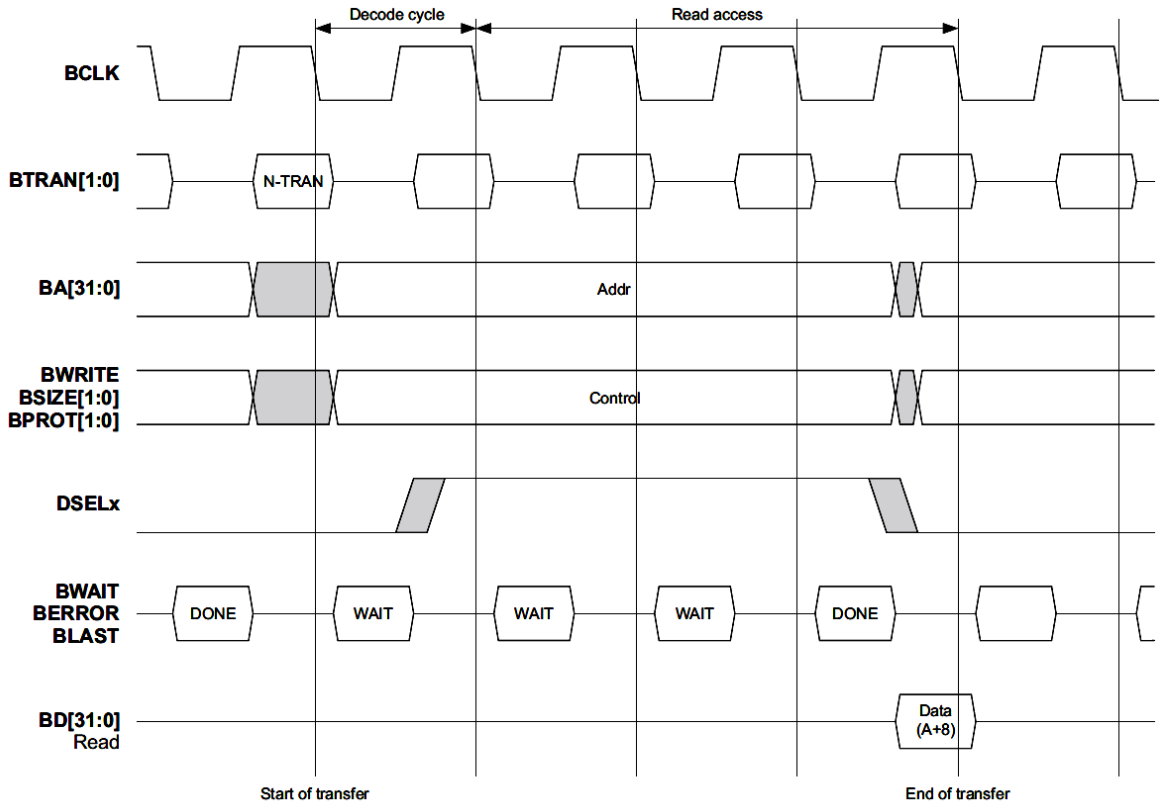


Рис. 75. Одиночная (NONSEQUENTIAL) передача данных.

Рассмотрим основные сигналы шины:

- BCLK – сигнал тактового генератора.
- BD[31:0] – шина данных.
- BA[31:0] – шина адреса.
- BWRITE – сигнал запись/чтение.
- BTRAN[1:0] – тип передачи (NONSEQUENTIAL, SEQUENTIAL, ADDRESS-ONLY).
- BSIZE[1:0] – размер передаваемых данных.
- DSELx – выбор устройства.

Рассмотрим взаимодействие нескольких главных устройств через арбитр шины:

- Главное устройство выставляет сигнал AREQx, означающий запрос на захват шины.
- Арбитр считывает запрос от главного устройства.
- Если сигнал BLOCK пассивен, то арбитр разрешает захват шины главному устройству, выставляя сигнал AGNTx. В противном случае, если сигнал BLOCK активен, разрешение на захват шины не выдается.

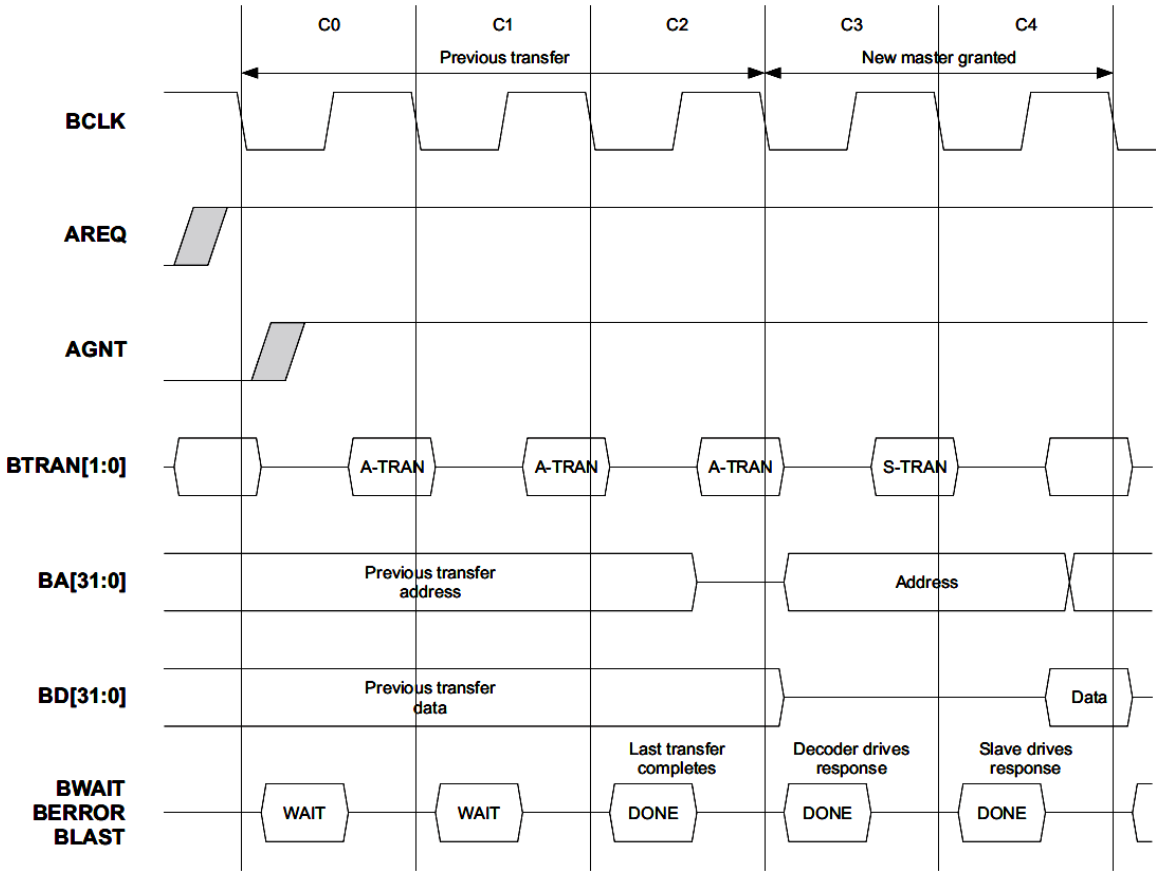


Рис. 76. Арбитраж в мультимастерном режиме работы шины AMBA ASB.

Интерфейс подчиненного устройства на шине AMBA ASB имеет следующий вид:

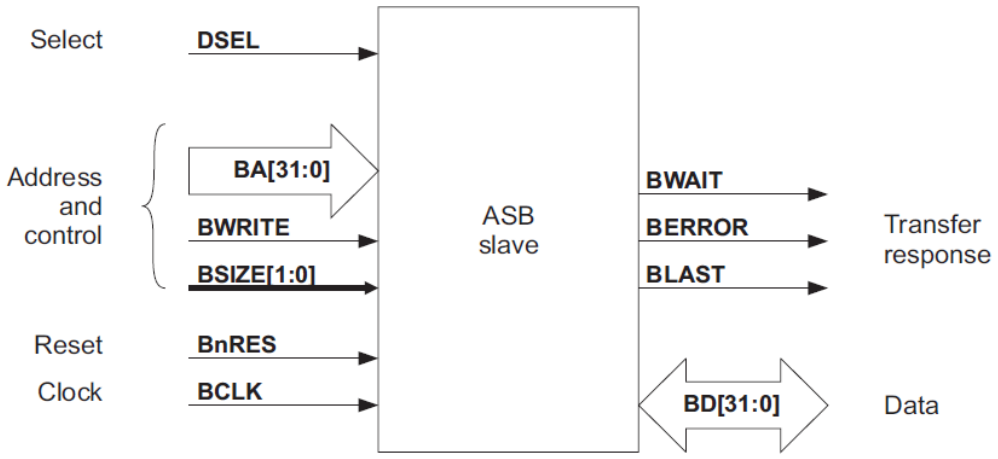


Рис. 77. Интерфейс подчиненного устройства для AMBA ASB.

3.6.3 Периферийный интерфейс AMBA APB

Интерфейс AMBA APB (Advanced Peripheral Bus) является частью иерархии интерфейсов AMBA и предназначен для объединения периферии, используемой в микроконтроллерах. Интерфейс AMBA APB используется практически во всех современных микроконтроллерах с ядром ARM.

Цель создания интерфейса – минимизация потребляемой мощности и упрощение архитектуры вычислительной системы.

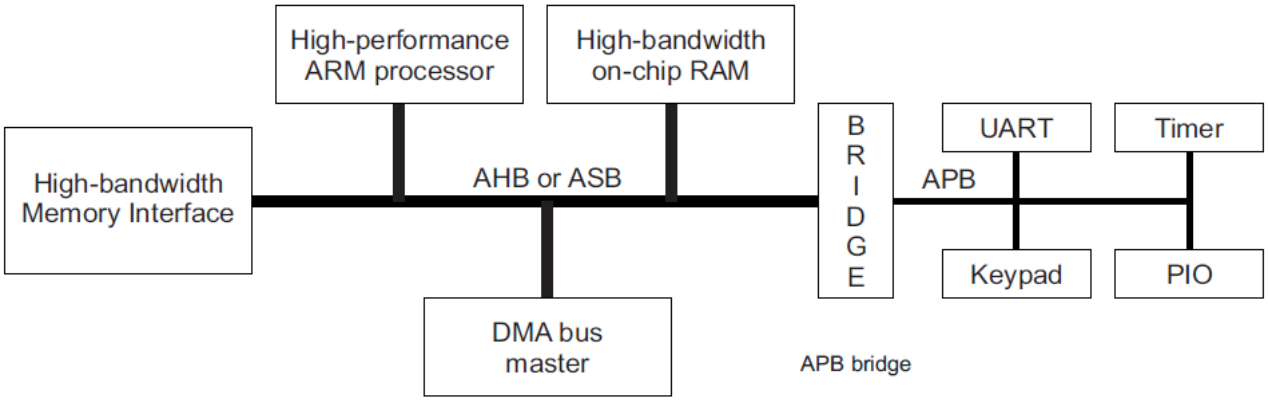


Рис. 78. Типичный микроконтроллер с периферийными контроллерами, подключенными к шине AMBA APB.

Интерфейс AMBA APB инкапсулирован в одном подчиненном устройстве шин AMBA AHB или AMBA ASB. При использовании шины APB потребление энергии значительно меньше, чем при прямом подключении контроллеров к системной шине. Интерфейс APB имеет смысл использовать с такими устройствами, в которых не требуется высокая пропускная способность шины и пакетный режим работы. Примерами таких устройств могут быть: контроллеры последовательного канала, таймеры, контроллеры I2C, SPI, ЦАП, АЦП, часов реального времени, сторожевого таймера и так далее.

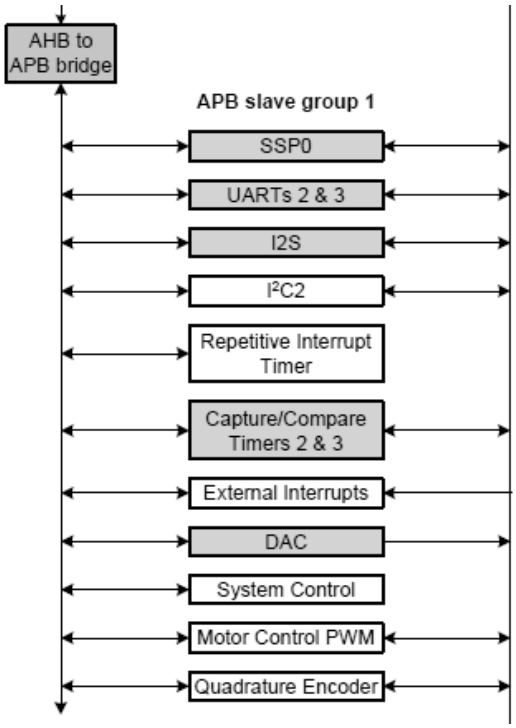


Рис. 79. Фрагмент одной из шин APB в микроконтроллере NXP LPC 1768 на базе Cortex-M3.

Единственным главным устройством (мастером) на шине APB является мост. Все остальные устройства (контроллеры) являются подчиненными устройствами.

Интерфейс AMBA APB может находиться в одном из трёх состояний:

- IDLE – устройство не готово, шина находится в исходном состоянии.
- SETUP – запущен процесс инициализации устройства.
- ENABLE – устройство готово к обмену.

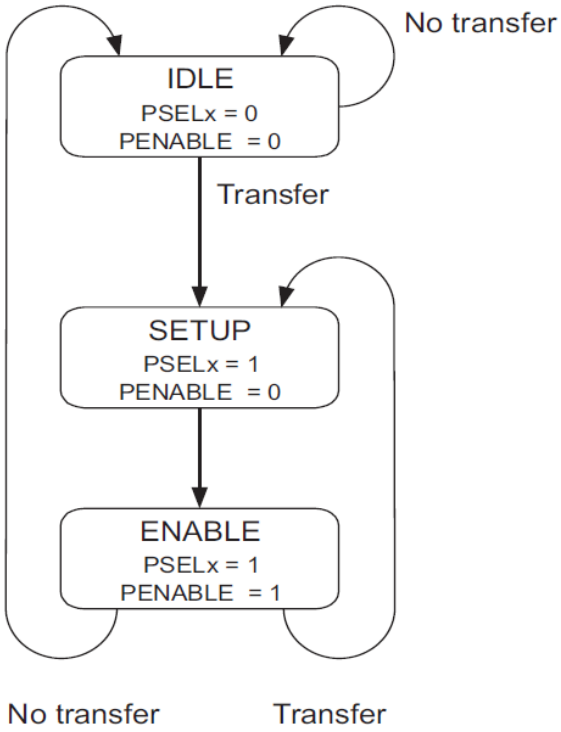


Рис. 80. Три состояния шины AMBA APB.

Если требуется что-то передать по шине, мы должны выбрать устройство, с которым мы будем общаться, с помощью сигнала PSELx. После получения сигнала устройство производит инициализацию и выставляет на шину сигнал PENABLE.

Циклы чтения и записи имеют вид, показанный на рисунке ниже.

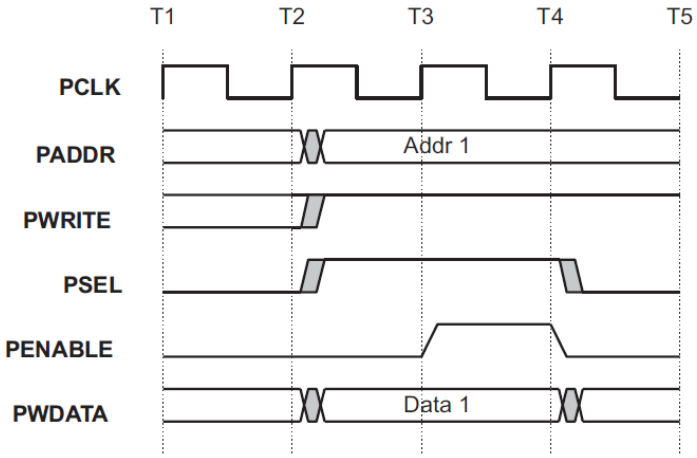


Рис. 81. Цикл записи данных в шине AMBA APB.

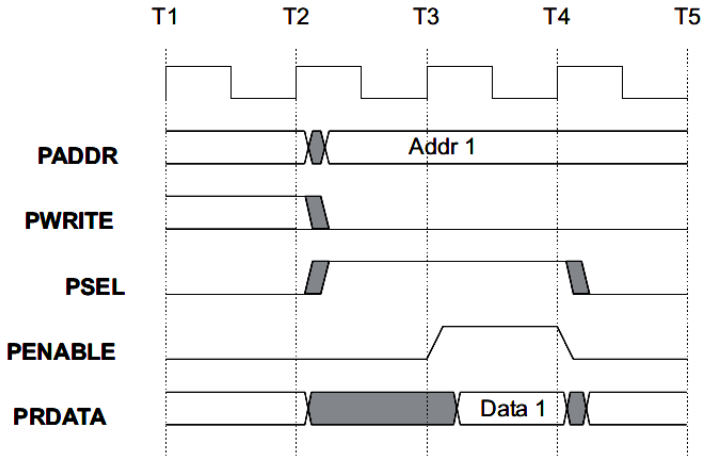


Рис. 82. Цикл чтения на шине AMBA APB.

Для взаимодействия с шинами AMBA AHB или AMBA ASB используется мост (APB bridge).

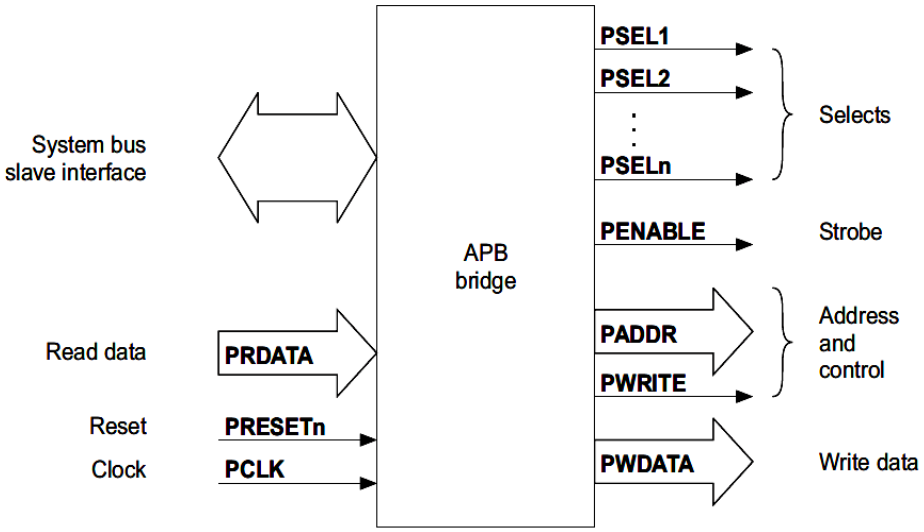


Рис. 83. Мост между периферийной шиной AMBA APB и шинами AMBA ASB или AMBA AHB.

Интерфейс для подключения подчиненных устройств к шине AMBA APB очень прост и имеет следующий вид:

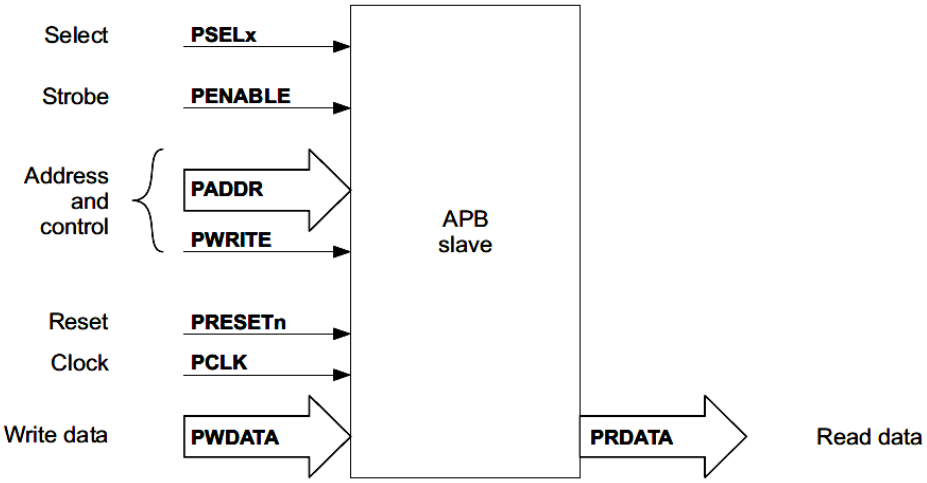


Рис. 84. Интерфейс подчиненного устройства в шине AMBA APB

3.7 Системные интерфейсы

3.7.1 Интерфейс PCI

PCI (Peripheral Component Interconnect) – мультплексированный, параллельный (разрядность 32 или 64 бита) системный интерфейс. Первая спецификация на этот интерфейс появилась в 1992 году. Первоначально интерфейс использовался в персональных компьютерах, работающих под управлением операционных

3.7.1.1 PCI-2.0

Спецификация PCI 2.0 (Peripheral Component Interconnect Local Bus Revision 2.0) определяет процессорно-независимую шину, предусматривающую подключение до шести устройств, в том числе контроллера шины PCI и дополнительного контроллера шины расширения ISA, EISA или MCA. Тактовая частота шины PCI достигает 33 МГц, причем обмен по ней может осуществляться 32- или 64-разрядными словами.

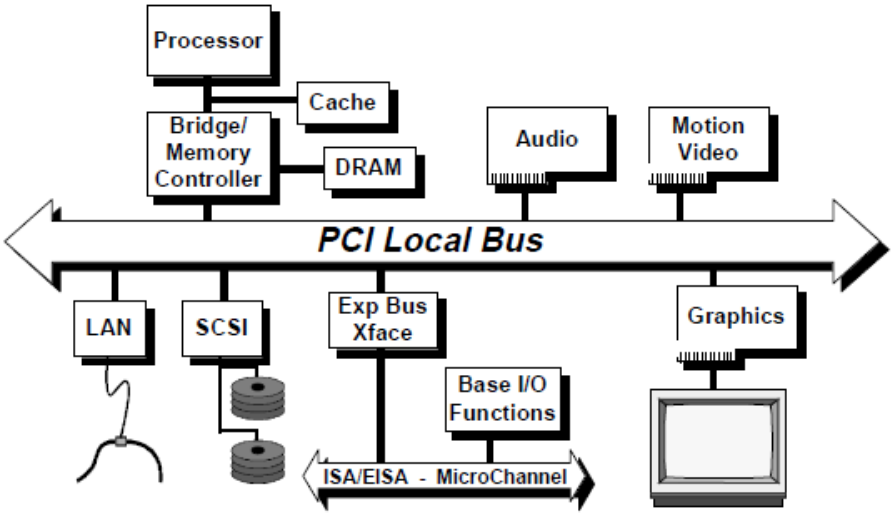


Рис. 85. Блок схема вычислительной системы на базе PCI.

Предусматривается одновременная поддержка нескольких главных устройств локальной шины, что важно для будущих мультимедиа- и других систем, обрабатывающих большие объемы графики, видео и данных других типов. Шина PCI поддерживает блочный обмен последовательными данными при выполнении операций чтения и записи данных в память.

В архитектуре PCI предусмотрен мост, развязывающий процессор и шину расширения при сохранении 32-разрядного тракта обмена данными с периферийными устройствами. Контроллер шины позволяет организовывать очередь операций записи и чтения.

Предусматривается возможность использования одной и той же системной платы для ЦП разных поколений, что, однако, требует более сложных схем управления контроллером шины PCI.

С целью сокращения числа выводов периферийных интегральных схем и снижения стоимости компонентов используется мультиплексирование сигналов.

Средняя номинальная скорость передачи данных по шине составляет 120 МБ/с, а пиковая – 132 МБ/с для 32-разрядного тракта передачи данных и 264 МБ/с – для 64-разрядного тракта.

Шина PCI поддерживает интерфейсы различных типов: MCA, ISA и EISA. Некоторые платы PCI могут использоваться в системах на различных платформах.

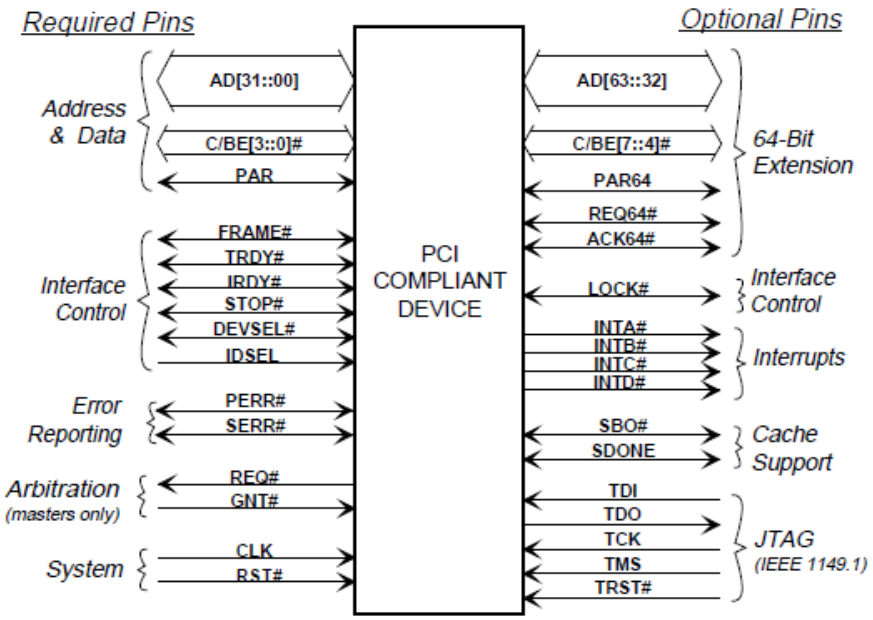


Рис. 86. Сигналы PCI-2.0.

В спецификации PCI 2.0 предусмотрены разные соединители для нормального (5 В) и пониженного (3,3 В) напряжения питания. Существуют соответственно 5- и 3,3-вольтовые платы, а также универсальные платы, которые могут устанавливаться в соединители обоих типов.

Имеются два дополнительных контакта PRSNT1# (B9) и PRSNT2# (B11). Они используются для индикации установки платы в гнездо и указывают величину необходимой для ее питания мощности. Благодаря отмеченным особенностям шина PCI может стать оптимальным выбором для компьютеров будущего. Ее внедрение обеспечит высокую производительность и хорошую совместимость.

Основой управления всеми передачами данных на PCI служат три сигнала:

- FRAME# ведётся задатчиком для отображения конца запроса.
- IRDY# ведётся задатчиком, позволяя вызывать циклы ожидания.
- TRDY# ведётся исполнителем, позволяя ему вызывать циклы ожидания.

Интерфейс не занят, если запрещены и FRAME# и IRDY#. Первый фронт синхроимпульса на котором разрешается FRAME# это адресная фаза, и

адресные и шинные коды команд передаются на этом фронте СИ. Следующий фронт СИ начинает первую из одной или нескольких фаз данных, в течение которой данные передаются между задатчиком и исполнителем по каждому фронту СИ и для этого разрешаются оба сигнала IRDY# и TRDY#. Состояния ожидания могут быть вставлены в фазу данных либо задатчиком либо исполнителем с помощью соответственно сигналов IRDY# и TRDY#.

Источнику данных необходимо разрешить свой RDY# сигнал без всяких условий, когда данные достоверны. Приёмник же может разрешить его, если ему нужно.

Как только задатчик разрешил IRDY#, он не может изменить IRDY# или FRAME# до тех пор, пока не завершится текущая фаза данных независимо от состояния TRDY#. Как только исполнитель разрешил TRDY# или STOP#, он уже не может изменить DEVSEL#, TRDY# или STOP# пока не завершится текущая фаза данных. Общий смысл в том, что ни задатчик, ни исполнитель не могут изменить свои действия, включившись в передачу данных.

В тот момент, когда задатчик собирается завершить передачу данных, запрещается FRAME# и разрешается IRDY#, указывая на то, что задатчик готов. После того, как испонитель укажет на последнюю передачу данных (TRDY# разрешён), интерфейс возвращается в состояние холостого хода с запрещёнными сигналами FRAME# и IRDY#.

Для облегчения процессов инициализации и конфигурации устройств на шине PCI, спецификация PCI поддерживает режим автоконфигурации, называемый режимом Plug and Play. Это существенно упрощает работу по подключению к компьютеру новых устройств.

Для осуществления режима конфигурации на шине PCI используются следующие аппаратно-программные средства.

В адресном пространстве, которое поддерживается шиной, выделяется специальное поле (пространство) конфигурации объемом 256 байт. Для выбора устройств в этом пространстве при операциях конфигурации на шине выделяются специальные линии IDSEL (Initialization Device Select). Эти линии индивидуальные для каждого устройства. В операциях на шине предусмотрены две специальные команды чтения и записи конфигурации. Спецификация не определяет универсальный механизм начала процесса конфигурации. Но для PC-AT совместимых компьютеров такой механизм специфицирован.

Шина PCI имеет две команды конфигурации: чтения и записи из адресного пространства конфигурации, емкость которого 256 байт. Команды конфигурации, подобно другим командам чтения и записи, разрешают доступ к байту, слову, двойному слову (32 р.) и пакетной передаче. Правила транзакций такие же как в других командах, включая все условия завершения транзакций.

Особенностями команд конфигурации являются следующие.

Доступ в область адресов конфигурации устройства на шине производится с помощью специальных, индивидуальных для каждого устройства, линий типа точка-точка, идущих от главного моста к каждому устройству отдельно. По этим линиям передается специальный сигнал выбора устройства IDSEL. Каждое устройство имеет свой вход IDSEL, который работает подобно классическому входу "выбор кристалла" (chip select). Устройство PCI становится исполнителем команды конфигурации только тогда, когда его сигнал IDSEL установлен, и два младших разряда адреса AD[1::0] содержат код 00 в течении фазы адреса команды конфигурации.

Активное значение сигнала IDSEL имеет высокий уровень, т.е. соответствует 1.

Адресация внутри пространства конфигурации устройства, содержащего 64 регистра по 32 разряда каждый, осуществляется разрядами [7::2] шины AD[7::2] и разрядами выбора байта C/B[3::0].

В командах конфигурации используется только 11 младших разряда на шине AD[10::0], а разряд AD[31::11] не используется (являются резервными).

Спецификацией не определено как формируется сигнал IDSEL главным мостом шины PCI. Однако применяется способ задания линий IDSEL путем использования старших линий шины AD[31::11]. Это позволяет иметь 21 такую линию. В этом случае, одна из линий AD[31::11] соединяется со входом IDSEL устройства через большое сопротивление. Это снижает нагрузку на линию при выполнении других операций на шине, но приводит к затягиванию фронта установки сигнала IDSEL в активное состояние. В связи с этим необходима предустановка адреса в фазе адреса транзакции конфигурации. Адрес может быть предустановлен за несколько импульсов до FRAME# (число импульсов для предустановки определяется из постоянной времени входной цепочки сигнала IDSEL).

Система, реализующая процедуру автоконфигурации, должна обеспечивать механизм разрешения цикла конфигурации на шине PCI, который генерируется программно. Этот механизм реализуется главным мостом. Для PC-AT совместимых систем этот механизм специфицирован и рассматривается ниже. Для систем другой архитектуры он не специфицирован.

Основной механизм шинных передач на PCI это "пакет". Пакет состоит из адресной фазы и одной или более фазы данных. PCI поддерживает пакеты и в пространстве памяти и в пространстве ввода/вывода. Хост-мост (который держит связь между хост-процессором и PCI) может объединять (или собирать) доступ к памяти на запись в единый запрос без посторонних эффектов. Устройство подтверждает, что их нет (позволяет предупреждать чтение данных и объединять записи данных в любом порядке) путём установки бита выборки в базовом адресном регистре. Мост может различать, где позволено объединение данных, а где нет путём адресного диапазона, который может быть предоставлен софтом конфигурации в процессе инициализации. Объединение

данных в такой буфер должно остановиться (а буфер очищен), когда происходит последовательная запись, которая не подпадает под выборку или чтение (в любом диапазоне).

Так как доступы к ресурсу на в/в не могут быть скомбинированы, они обычно имеют только одиночную фазу данных. Вообще, ни один процессор или задатчик на шине не генерирует пакеты в пространстве в/в. Все доступы в/в должны появляться на PCI, как только ЦП их сгенерировал. Почти все сигналы выровнены по фронту синхроимпульса.

Начало и продолжение транзакции

Транзакцию начинает задатчик, предварительно получив разрешение на работу на шине. Это разрешение задатчик получает от арбитра, послав ему сигнал запроса REQ# и получив разрешение GRN# от него.

После этого задатчик начинает транзакцию с адресной фазы путем установки по 1-ому импульсу транзакции сигналов FRAME#, адреса AD[31::0] и команды C/BE[3::0]#. Эти сигналы становятся достоверными ко 2-ому импульсу транзакции. На первом импульсе начинается адресная фаза.

Рассмотрим сигналы транзакции чтения.

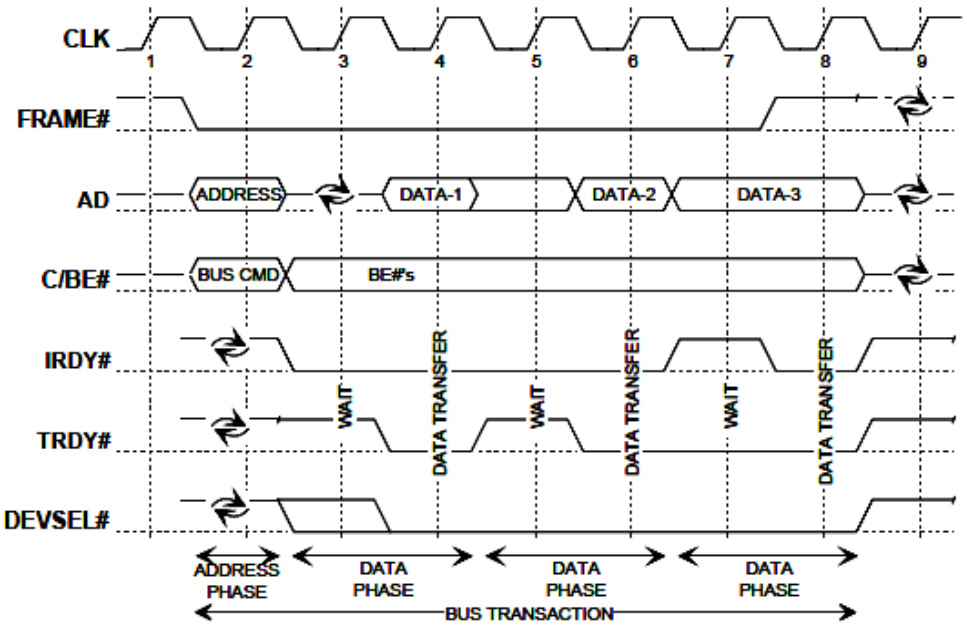


Рис. 87. Операция чтения.

По второму импульсу, в соответствии с установленным адресом определяется исполнитель и соответствующая команда. При операции чтения по этому импульсу задатчик выставляет сигнал IRDY#, который говорит исполнителю, что задатчик готов принять данные. Исполнитель ко 2-ому импульсу организует Т-цикл для сигналов AD[31::0].

По 2-ому импульсу он может установить, если успеет, сигнал DEVSEL#, сообщающий задатчику, что исполнитель найден и имеет право проводить транзакцию, если не успевает из-за Т-цикла на шине AD, то установка

DEVSEL# происходит на 3-м импульсе. На 2-м импульсе исполнитель устанавливает сигналы указателя байт при передаче данных C/BE[3::0]#. После 2-го импульса задатчик не управляет линиями AD[31::0] и C/BE[3::0]#, ими управляет исполнитель. На 2-ом импульсе заканчивается фаза адреса и начинается фаза данных.

На 3-ем импульсе исполнитель определяет, что задатчик готов (по сигналу IRDY#) к приему данных, и выставляет первые данные на AD[31::0] и устанавливает сигнал TRDY#, который сообщает задатчику, что на линиях AD[31::0] имеются первые достоверные данные. На этом такте может устанавливаться сигнал DEVSEL#, если исполнитель не успел его установить на втором импульсе.

Сигнал DEVSEL# должен устанавливаться после декодирования адреса и перед или вместе с сигналами IRDY#, STOP# и данных. Исполнитель не должен сбрасывать DEVSEL#, пока не закончится последняя фаза данных. Так как на 3-м импульсе фазы данных передачи данных от исполнителя к задатчику не произошло, то этот такт называется тактом ожидания.

Только на 4-м импульсе задатчик определяет (по сигналу TRDY#), что на шине AD[31::0] находятся достоверные данные и считывает их, завершая первую фазу данных. На 4-м импульсе начинается вторая фаза данных и т.д. Фазы данных могут занимать один такт, если нет тактов ожидания, или несколько тактов, если есть такты ожидания. Такты ожидания могут формироваться либо задатчиком (сбросом IRDY#), либо исполнителем (сбросом TRDY#).

Задатчик или исполнитель могут менять значения данных на шине AD только при активных значениях сигналов IRDY# и TRDY# соответственно. Если эти сигналы сброшены, то на шине AD удерживаются старые значения данных.

При операции записи, значения сигналов на шине C/BE# меняется на каждой фазе данных. При чтении значения C/BE# не меняется в течение всех фаз данных транзакции.

Окончание транзакции

На последней фазе данных обязательно сбрасывается FRAME# и устанавливается IRDY#. FRAME# может быть сброшен только тогда, когда IRDY# установлен. После тог как исполнитель на последней фазе данных установит TRDY#, может быть проведена последняя передача от исполнителя к задатчику, и транзакция закончится сбросом на последнем импульсе транзакции сигнала IRDY#. Так как сброшены сигналы FRAME# и IRDY#, то следующий такт будет тактом холостого хода (IDLE циклом).

По последнему импульсу транзакции также сбрасываются сигналы TRDY# и DEVSEL#.

Операция записи. Транзакция записи выполняется точно также, за исключением того, что в ней отсутствует Т-цикл на 3-м импульсе для сигналов AD[31::0]. Поэтому на 3-м импульсе отсутствует такт очищения, и исполнитель на 3-м импульсе считывает первые данные, выставленные задатчиком. При транзакции записи шинами AD и C/BE управляет задатчик.

Сигнал STOP# используется для завершения транзакции по инициативе исполнителя.

Способы завершения транзакций

Транзакция может быть завершена либо задатчиком, либо исполнителем. Пока ни тот ни другой не инициализируют останов, транзакция продолжается.

1-й способ завершения транзакции задатчиком заключается в следующем. Задатчик инициализирует завершение транзакции, когда сигнал FRAME# сброшен, а IRDY# установлен. Это указывает исполнителю, что наступает последняя фаза данных. Последняя передача данных происходит, когда установлены и IRDY#, и TRDY#. Транзакция завершается, когда и FRAME#, и IRDY# сброшены (состояние холостого хода шины).

Задатчик может прекращать транзакцию этим способом по двум причинам:

- Когда задатчик заканчивает начатую им транзакцию.
- Когда линия разрешения захвата шины задатчиком GRN сброшена и наступил момент Тайм-аут, когда исчерпано время, задаваемое Таймером задержки.

Модифицированная версия этого способа завершения транзакции задатчиком используется тогда, когда исполнитель не отвечает на его адресацию сигналом DEVSEL#.

2-й способ завершения транзакции исполнителем заключается в следующем. В этом случае используется сигнал STOP#. Исполнитель выдает сигнал STOP#, чтобы запросить завершение транзакции от задатчика.

После установки, STOP# сохраняет активное значение до момента сброса FRAME#. Взаимосвязь между IRDY# и TRDY# не зависит от взаимодействия между STOP# и FRAME#. Поэтому данные могут быть переданы или не переданы до конца в текущей транзакции. Это зависит единственно от состояния IRDY# и TRDY#. Однако, когда STOP# установлен, а TRDY# сброшен, это указывает на невозможность дальнейшей передачи данных исполнителя. В этом случае, задатчик не ждет последней передачи данных, а немедленно завершает транзакцию.

Арбитраж

Каждый задатчик на шине PCI получает доступ к шине только после разрешения на захват шины от Арбитра. Для этого используется специальные, индивидуальные для каждого задатчика линии (типа точка-точка) запроса REQ# к арбитру и разрешения GNT# к задатчику от арбитра.

Для доступа к шине, задатчик выдает арбитру сигнал запроса REQ# и может захватить шину только после получения от арбитра сигнала разрешения GNT#.

Процесс арбитража не требует дополнительных циклов шины, так как он совмещается с выполнением других операций, кроме случая, когда шина не занята и находится в состоянии холостого хода (IDLE цикл).

Захват шины задатчиком и выполнение транзакции идет в такой последовательности. Задатчик выдает запрос REQ#, через время задержки арбитража арбитр выдает задатчику сигнал разрешения GNT#, через время задержки ожидания захвата шины задатчик начинает транзакцию, выставив сигнал FRAME#. Идет передача одной или нескольких транзакций, в конце последней транзакции производится сброс сигнала REQ#, затем GNT#.

Такой процесс происходит тогда, когда нет запросов шины от задатчиков с большим приоритетом.

Если во время работы текущего задатчика появляется запрос от объекта с большим приоритетом, то либо идет прерывание транзакции, либо она выполняется до конца, а затем шина переходит к задатчику с большим приоритетом.

Рассмотрим процессы установки и сброса сигналов REQ# и GNT# в различных ситуациях.

- Задатчик выдает сигнал REQ# по фронту любого импульса синхронизации шины (CLK).
- Задатчик может сбросить сигнал REQ# на любом импульсе CLK, сброс этого сигнала арбитр интерпретирует как то, что задатчику больше шина не требуется и можно сбросить его сигнал разрешения GNT#.
- Если задатчику требуется выполнение только одной транзакции, то он сбрасывает REQ# на первом импульсе CLK, начинающем транзакцию, одновременно с установкой сигнал FRAME# этой транзакции.
- Если задатчику требуется передаче нескольких транзакций, то он должен удерживать свой REQ# до последней транзакции, так как арбитр дает разрешение только на одну транзакцию.
- Когда транзакция заканчивается по инициативе исполнителя установкой сигнала STOP#, задатчик должен сбросить свой REQ# минимум за два импульса CLK. Если задатчик намеревается продолжить прерванную исполнителем транзакцию, он должен переустановить свой сигнал REQ#.

Таким образом, в одно и то же время могут быть установлены REQ# от нескольких задатчиков.

В ответ на запрос арбитр выдает сигнал разрешения GNT#. При нескольких запросах REQ# арбитр выдает сигнал разрешения GNT# задатчику, имеющему в данный момент времени наивысший приоритет.

На шине в любой момент времени может быть установлен только один сигнал GNT#, так как шиной в каждый момент времени управляет только один задатчик.

Арбитр выдает сигнал разрешения GNT# в ответ на запрос REQ# с некоторой задержкой арбитража. Эта задержка измеряется от момента установки REQ# (т.е. от первого импульса CLK, которому соответствует стабильное значение REQ#) до момента получения стабильного значения сигнала GNT# задатчиком (т.е. до первого импульса CLK, которому соответствует стабильное значение GNT#). Это время измеряется в числе импульсов CLK.

Время задержки арбитража зависит от алгоритма арбитража, уровня приоритета устройства. Минимальная задержка арбитража получается в случае, когда шина находится в состоянии холостого хода (IDLE).

Типовая задержка арбитража для задатчика с наивысшим приоритетом, при наличие на шине текущего задатчика, равна двум импульсам CLK.

Если текущего задатчика нет, то задержка равна одному импульсу.

Для устройств с более низким приоритетом может требоваться большая задержка арбитража в зависимости от числа задействованных устройств с более высоким уровнем приоритетов.

Арбитр может сбросить сигнал GNT# задатчика на любом импульсе CLK для обслуживания объекта с большим приоритетом.

Сброс сигнала GNT# может производиться либо в ответ на сброс своего сигнала запроса REQ#, либо в случае, когда пришел запрос от задатчика с большим приоритетом.

Арбитр может сохранить установленный сигнал разрешения GNT# для данного задатчика и при сбросе этим задатчиком сигнала REQ#, если нет других требований на захват шины.

Если FRAME# сброшен, GNT# этого задатчика может быть снят в любой момент времени в порядке обслуживания задатчика с более высоким приоритетом или в ответ на сброс своего REQ#.

Арбитр может сбросить GNT# одного объекта в момент установки GNT# другого с более высоким приоритетом, если на шине нет состояния холостого хода. В противном случае, требуется задержка в один импульс между сбросом одного GNT# и установкой другого, чтобы исключить состязания на шине по сигналам AD и PAR.

Если шина находится в состоянии холостого хода (IDLE) и арбитр сбросил GNT#, то задатчик прекращает транзакцию и теряет доступ к шине, за

исключением случая когда происходит одновременно (на одном и том же импульсе CLK) сброс GNT# и установка FRAME# объекта. В этом случае транзакция продолжается.

Если задатчик не использует шину после получения сигнала разрешения GNT# (его REQ# установлен) в течение 16 импульсов, то арбитр отключает его от шины.

Получив сигнал GNT#, задатчик не сразу захватывает шину, а через некоторое время, называемое задержкой ожидания захвата шины (задержкой захвата). Это время от получения GNT# задатчиком до установки стабильного значения сигнала FRAME#. Оно также измеряется количеством импульсов CLK.

Задержка захвата определяется тем, что задатчик, получив разрешение GNT#, может начать свою транзакцию только тогда, когда шина будет находиться в состоянии холостого хода (сброшены FRAME# и IRDI#). Поэтому этот задатчик ждет, пока текущий задатчик завершит транзакцию и шина перейдет в состояние холостого хода.

Текущий задатчик завершает транзакцию в следующих случаях:

1. Нормальное завершение в конце транзакции.
2. Завершение по прерыванию транзакции:
   - Прерывание от задатчика, когда появляется сигнал тайм-аут (время таймера задержки истекло) и сброшен его сигнал GNT#.
   - Прерывание от исполнителя, когда исполнитель выдает сигнал STOP#.

3.7.1.2 PCI 2.1 - 3.0

Отличались от версии 2.0 возможностью одновременной работы нескольких шинных задатчиков (bus-master, так называемый «конкурентный режим»), а также появлением универсальных карт расширения, способных работать в слотах, использующих как с напряжением 5В, так и в слотах, использующих 3,3В (с частотой 33 и 66 МГц соответственно). Пиковая пропускная способность для 33 МГц – 133 МБ/с, а для 66 МГц – 266 МБ/с.

- Версия 2.1 – работа с картами, рассчитанными на напряжение 3,3 вольта, наличие соответствующих линий питания являлась опциональной.
- Версия 2.2 – сделанные в соответствии с этими стандартами карты расширения имеют универсальный ключ разъёма по питанию и способны работать во многих более поздних разновидностях слотов шины PCI, а также, в некоторых случаях, и в слотах версии 2.1.
- Версия 2.3 несовместима с картами PCI, рассчитанными на использование 5 вольт, несмотря на продолжающееся использование 32-битных слотов с 5 вольтовым ключом. Карты расширения имеют

универсальный разъём, но не способны работать в 5 вольтовых слотах ранних версий (до 2.1 включительно).

- Версия 3.0 завершает переход на карты PCI 3,3 вольт, карты PCI (5 вольт) больше не поддерживаются.

3.7.1.3 PCI 64

Расширение базового стандарта PCI, появившееся в версии 2.1, удваивающее число линий данных, и, следовательно, пропускную способность. Слот PCI64 является удлинённой версией обычного PCI-слота. Формально совместимость 32-битных карт с 64-битным слотами (при условии наличия общего поддерживаемого сигнального напряжения) полная, а совместимость 64-битной карты с 32-битным слотами является ограниченной (в любом случае произойдёт потеря производительности). Работает на тактовой частоте 33 МГц. Пиковая пропускная способность – 266 МБ/с.

- Версия 1 использует слот PCI 64-бита и напряжение 5В.
- Версия 2 использует слот PCI 64-бита и напряжение 3,3В.

3.7.1.4 PCI 66

Версия PCI 66 является работающим на тактовой частоте 66 МГц развитием PCI 64; использует напряжение 3,3В в слоте; карты имеют универсальный, либо 3,3 В форм-фактор. Пиковая пропускная способность – 533 МБ/с.

3.7.1.5 PCI 64/66

Комбинация PCI 64 и PCI 66, позволяет вчетверо увеличить скорость передачи данных по сравнению с базовым стандартом PCI; использует 64-битные 3,3 вольтовые слоты, совместимые только с универсальными и 3,3 вольтовые 32-битные карты расширения. Карты стандарта PCI64/66 имеют либо универсальный (но имеющий ограниченную совместимость с 32-битными слотами) либо 3,3 вольтовый форм-фактор (последний вариант принципиально не совместим с 32-битными 33 МГц слотами популярных стандартов). Пиковая пропускная способность – 533 МБ/с.

3.7.1.6 PCI-X

Развитие версии PCI 64. Для всех вариантов шины существуют следующие ограничения по количеству подключаемых к каждой шине устройств: 66 МГц – 4, 100 МГц – 2, 133 МГц – 1 (или 2, если одно или оба устройства не находятся на платах расширения, а уже интегрированы на одну плату вместе с контроллером), 266, 533 МГц и выше – 1.

В версии 1.0 введено две новые рабочие частоты: 100 и 133 МГц, а также механизм раздельных транзакций для улучшения производительности при одновременной работе нескольких устройств. Как правило, обратно совместима со всеми 3,3 В и универсальными PCI-картами. Карты обычно выполняются в

64-битном 3,3 В формате и имеют ограниченную обратную совместимость со слотами PCI64/66, а некоторые - в универсальном формате и способны работать (хотя практической ценности это почти не имеет) в обычном PCI 2.2/2.3. Пиковая пропускная способность – 1024 МБ/с.

В версии 2.0 введено две новые рабочие частоты: 266 и 533 МГц, а также коррекция ошибок чётности при передаче данных (ECC). Расширяет конфигурационное пространство PCI до 4096 байт и допускает расщепление на 4 независимых 16-битных шины, что применяется исключительно во встраиваемых и промышленных системах, сигнальное напряжение снижено до 1,5 В, но сохранена обратная совместимость разъёмов со всеми картами, использующими сигнальное напряжение 3,3В. Пиковая пропускная способность – 4096 МБ/с.

3.7.2 Интерфейс PCI Express

PCI Express, или PCIe, или PCI-E – компьютерная шина, использующая программную модель шины PCI и высокопроизводительный физический протокол, основанный на последовательной передаче данных.

Развитием стандарта PCI Express занимается организация PCI Special Interest Group.

В отличие от шины PCI, использовавшей для передачи данных общую шину, PCI Express, в общем случае, является пакетной сетью с топологией типа звезда, устройства PCI Express взаимодействуют между собой через среду, образованную коммутаторами, при этом каждое устройство напрямую связано соединением типа точка-точка с коммутатором.

Кроме того, шиной PCI Express поддерживается:

- Горячая замена карт.
- Гарантированная полоса пропускания (QoS).
- Управление энергопотреблением.
- Контроль целостности передаваемых данных.

Разработка стандарта PCI Express была начата фирмой Intel после отказа от шины InfiniBand. Официально первая базовая спецификация PCI Express появилась в июле 2002 года.

Шина PCI Express нацелена на использование только в качестве локальной шины. Так как программная модель PCI Express во многом унаследована от PCI, то существующие системы и контроллеры могут быть доработаны для использования шины PCI Express заменой только физического уровня, без доработки программного обеспечения. Высокая пиковая производительность шины PCI Express позволяет использовать её вместо шин AGP и тем более PCI и PCI-X.

Для подключения устройства PCI Express используется двунаправленное последовательное соединение типа точка-точка, называемое lane; это резко отличается от PCI, в которой все устройства подключаются к общей 32-разрядной параллельной двунаправленной шине.

Соединение между двумя устройствами PCI Express называется link, и состоит из одного (называемого 1x) или нескольких (x2, x4, x8, x12, x16 и x32) двунаправленных последовательных соединений lane. Каждое устройство должно поддерживать соединение x1.

На электрическом уровне каждое соединение использует низковольтную дифференциальную передачу сигнала (LVDS), приём и передача информации производится каждым устройством PCI Express по отдельным двум проводникам, таким образом, в простейшем случае, устройство подключается к коммутатору PCI Express всего лишь четырьмя проводниками.

Использование подобного подхода имеет следующие преимущества:

- Карта PCI Express помещается и корректно работает в любом слоте той же или большей пропускной способности (например, карта x1 будет работать в слотах x4 и x16).
- Слот большего физического размера может использовать не все lane'ы (например, к слоту x16 можно подвести линии передачи информации, соответствующие x1 или x8, и всё это будет нормально функционировать; однако, при этом необходимо подключить все линии «питание» и «земля», необходимые для слота x16).

В обоих случаях, на шине PCI Express будет использовать максимальное количество lane'ов доступных как для карты, так и для слота. Однако это не позволяет устройству работать в слоте, предназначенном для карт с меньшей пропускной способностью шины PCI Express. Например, карта x4 физически не поместится в стандартный слот x1, несмотря на то, что она могла бы работать в слоте x4 с использованием только одного lane. На некоторых материнских платах можно встретить нестандартные слоты x1 и x4, у которых отсутствует крайняя перегородка, таким образом, в них можно устанавливать карты большей длины, чем разъем. При этом не обеспечивается питание и заземление выступающей части карты, что может привести к различным проблемам.

PCI Express пересылает всю управляющую информацию, включая прерывания, через те же линии, что используются для передачи данных. Последовательный протокол никогда не может быть заблокирован, таким образом задержки шины PCI Express вполне сравнимы с таковыми для шины PCI (заметим, что шина PCI для передачи сигнала о запросе на прерывание использует отдельные физические линии IRQ#A, IRQ#B, IRQ#C, IRQ#D).

Во всех высокоскоростных последовательных протоколах (например, гигабитный Ethernet) информация о синхронизации должна быть встроена в передаваемый сигнал. На физическом уровне, PCI Express использует метод

канального кодирования 8B/10B (8 бит в 10-и, избыточность 20%) для устранения постоянной составляющей в передаваемом сигнале и для встраивания информации о синхронизации в поток данных. В PCI Express 3.0 используется более экономное кодирование 128b/130b с избыточностью 1,5%.

Некоторые протоколы (например, SONET/SDH) используют метод, который называется скремблинг для встраивания информации о синхронизации в поток данных и для "размывания" спектра передаваемого сигнала. Спецификация PCI Express также предусматривает функцию скремблинга, но скремблинг PCI Express отличается от такового для SONET.

3.8 Стандартные периферийные интерфейсы

3.8.1 Интерфейс SCSI

SCSI – интерфейс, разработанный для объединения на одной шине различных по своему назначению устройств, таких как жёсткие диски, накопители на магнитооптических дисках, приводы CD, DVD, стримеры, сканеры, принтеры и т. д. Раньше имел неофициальное название Shugart Computer Systems Interface в честь создателя Алана Ф. Шугарта.

Теоретически возможен выпуск устройства любого типа на шине SCSI.

После стандартизации в 1986 году SCSI начал широко применяться в компьютерах Apple Macintosh, Sun Microsystems. В компьютерах, совместимых с IBM PC, SCSI не пользуется такой популярностью в связи со своей сложностью и сравнительно высокой стоимостью и применяется преимущественно в серверах.

SCSI широко применяется на серверах, высокопроизводительных рабочих станциях. RAID-массивы на серверах часто строятся на жёстких дисках со SCSI-интерфейсом (однако, в серверах нижнего ценового диапазона всё чаще применяются RAID-массивы на основе SATA). В настоящее время устройства на шине SAS постепенно вытесняют устаревшую шину SCSI.

Система команд SCSI на уровне программного обеспечения употребляется в единых стеках поддержки устройств хранения данных в ряде операционных систем, таких как Microsoft Windows.

Существует реализация системы команд SCSI поверх оборудования (контроллеров и кабелей) IDE/ATA/SATA, называемая ATAPI – ATA Packet Interface. Все используемые в компьютерной технике подключаемые по IDE/ATA/SATA приводы CD/DVD/Blu-Ray используют эту технологию.

Также система команд SCSI реализована поверх протокола USB, что является частью спецификации класса Mass Storage device. Это позволяет подключать через интерфейс USB любые хранилища данных (от флеш-накопителей до внешних жёстких дисков), не разрабатывая для них собственного протокола обмена, а вместо этого используя имеющийся в операционной системе драйвер SCSI.

Существует три стандарта SCSI:

1. SE (single-ended) – асимметричный SCSI, для передачи каждого сигнала используется отдельный проводник.
2. LVD (low-voltage-differential) – интерфейс дифференциальной шины низкого напряжения, сигналы положительной и отрицательной полярности идут по разным физическим проводам. На один сигнал приходится по одной витой паре проводников. Используемое напряжение при передаче сигналов +1,8 В.
3. HVD (high-voltage-differential) – интерфейс дифференциальной шины высокого напряжения, отличается от LVD повышенным напряжением и специальными приемопередатчиками.

Первый стандарт SCSI имеет 50-контактный неэкранированный разъем для внутрисистемных соединений и аналогичный экранированный разъем типа Centronics (Alternative 2) для внешних подключений. Передача сигналов осуществляется 50 контактным кабелем типа A-50 на 8-разрядной (битной) шине. Но надо иметь ввиду, что до появления SCSI, имевшего 50-контактный разъём, и даже одновременно с ним был более старый SCSI, имевший 25-контактный разъём, почти такой, как разъём LPT (например, в теперь уже почти вышедшем из употребления сканере Mustek 1200 FS есть одновременно три разъёма: OPTION на 26 контактов, SCSI на 25 контактов, SCSI на 50 контактов).

В стандарте SCSI-2 для 8-битной шины предусматривался кабель типа A, который, как и в SCSI-1, поддерживал 50-контактными разъемами типа D с уменьшенным шагом выводов (Alternative 1). Разъемы типа Centronics (Alternative 2) в SCSI-2 построены 8- и 16-битной шине. Передача информации осуществляется по 68-контактным кабелям типа A-68 и P-68(Wide). Для 32-битной версии шины был предусмотрен тип кабеля B, который должен был параллельно подключаться одновременно с кабелем A в одно устройство. Однако кабель B не получил широкого признания и из стандарта SCSI-3 исключен.

В стандарте SCSI-3 кабеля A-68 и P-68 поддерживались экранированными, либо неэкранированными разъемами типа D. Кабеля в SCSI-3 снабжены фиксаторами-защелками, а не проволочными кольцами, как разъемы Centronics. Начиная с этой версии SCSI, в массивах накопителей используется 80-контактный разъем, называемый Alternative 4. Накопители с таким разъемом поддерживают горячее подключение устройств, т.е. устройства SCSI можно подключать и отключать при включенном питании.

3.8.1.1 SCSI-1

Стандартизован ANSI в 1986 г. Использовалась восьмибитная шина с пропускной способностью в 1,5 МБ/с в асинхронном режиме и 5 МБ/с в синхронном режиме. Максимальная длина кабеля – до 6 метров.

3.8.1.2 SCSI-2

Этот стандарт был предложен в 1989 году и существовал в двух вариантах – Fast SCSI и Wide SCSI:

1. Fast SCSI характеризуется удвоенной пропускной способностью (до 10 МБ/с).
2. Wide SCSI в дополнение к этому имеет удвоенную разрядность шины (16 бит), что позволяет достичь скорости передачи до 20 МБ/с.

При этом максимальная длина кабеля ограничивалась тремя метрами. Также в этом стандарте была предусмотрена 32-битная версия Wide SCSI, которая позволяла использовать два шестнадцатибитных кабеля на одной шине, но эта версия не получила распространения.

3.8.1.3 SCSI-3

Этот стандарт также известен под названием Ultra SCSI, предложен в 1992 году. Пропускная способность шины составила 20 МБ/с для восьмибитной шины и 40 МБ/с для шестнадцатибитной. Максимальная длина кабеля так и осталась равной трём метрам. Устройства, отвечающие этому стандарту, известны своей чувствительностью к качеству элементов системы (кабель, терминаторы).

3.8.1.4 Ultra-2 SCSI

Предложен в 1997 году. Использует LVDS. Максимальная длина кабеля – 12 метров, пропускная способность – до 80 МБ/с.

3.8.1.5 Ultra-3 SCSI

Этот стандарт также известен под названием Ultra-160 SCSI, предложен в конце 1999 года. Имеет удвоенную пропускную способность (по сравнению с Ultra-2 SCSI), которая составила 160 МБ/с. Увеличения пропускной способности удалось достичь за счёт одновременного использования фронтов и срезов импульсов.

В этот стандарт было добавлено использование CRC (Cyclic Redundancy Check), предупреждение ошибок.

3.8.1.6 Ultra-320 SCSI

Этот стандарт также известен под названием Fast Ultra-320. Ultra320 LVD SCSI диск Fujitsu MAP3735NC из состава RAID-массива подключается при помощи разъёма SCA-2. Развитие интерфейса Ultra-3 с удвоенной скоростью передачи данных (до 320 МБ/с).

3.8.1.7 Ultra-640 SCSI

Этот стандарт предложен в начале 2003 года. Удвоенная пропускная способность (640 МБ/с). В связи с резким сокращением максимальной длины

кабеля Ultra-640 SCSI неудобен для использования с более, чем двумя устройствами, поэтому не получил широкого распространения.

3.8.1.8 Команды SCSI

В терминологии SCSI взаимодействие идёт между инициатором и целевым устройством. Инициатор посылает команду целевому устройству, которое затем отправляет ответ инициатору.

Команды SCSI посылаются в виде блоков описания команды (Command Descriptor Block, CDB). Длина каждого блока может составлять 6, 10, 12 или 16 байт. В последних версиях SCSI блок может иметь переменную длину. Блок состоит из однобайтового кода команды и параметров команды.

После получения команды целевое устройство возвращает значение 00h в случае успешного получения, 02h в случае ошибки или 08h в случае, если устройство занято. В случае, если устройство вернуло ошибку, инициатор обычно посылает команду запроса состояния. Устройство возвращает Key Code Qualifier (KCQ).

Все команды SCSI делятся на четыре категории: N (non-data), W (запись данных от инициатора целевым устройством), R (чтение данных) и B (двусторонний обмен данными). Всего существует порядка 60 различных команд SCSI, из которых наиболее часто используются:

- Test unit ready – проверка готовности устройства, в т.ч. наличия диска в дисководе.
- Inquiry – запрос основных характеристик устройства.
- Send diagnostic – указание устройству провести самодиагностику и вернуть результат.
- Request sense – возвращает код ошибки предыдущей команды.
- Read capacity – возвращает ёмкость устройства.
- Read (4 варианта) – чтение данных из устройства.
- Write (4 варианта) – запись данных в устройство.
- Write and verify – запись и проверка.
- Mode select – установка параметров устройства.
- Mode sense – возвращает текущие параметры устройства.

Каждое устройство на SCSI-шине имеет как минимум один номер логического устройства (LUN, Logical Unit Number). В некоторых более сложных случаях одно физическое устройство может представляться набором LUN.

Семейство стандартов SCSI включает в себя ряд стандартов уровня аппаратуры, стандарты SAM и SPC, описывающие главнейшие команды и

структуры типа развернутой информации об ошибке, и специфичных для класса устройств стандартов.

Одним из последних является MMC (Multimedia Command Set), полностью описывающий систему команд приводов CD/DVD/Blu-Ray, в том числе их разновидностей с возможностью записи. Некоторые приводы, например, производства Asus и Pioneer, используют конкурирующий стандарт Mt. Fuji, отличающийся от MMC в некоторых нюансах.

3.8.2 Интерфейс SAS

Serial Attached SCSI (SAS) – компьютерный интерфейс, разработанный для обмена данными с такими устройствами, как жёсткие диски, накопители на оптическом диске и т.д. SAS использует последовательный интерфейс для работы с непосредственно подключаемыми накопителями (Direct Attached Storage (DAS) devices). SAS разработан для замены параллельного интерфейса SCSI и позволяет достичь более высокой пропускной способности, чем SCSI; в то же время SAS совместим с интерфейсом SATA. Хотя SAS использует последовательный интерфейс в отличие от параллельного интерфейса, используемого традиционным SCSI, для управления SAS-устройствами по-прежнему используются команды SCSI. Протокол SAS разработан и поддерживается комитетом T10. SAS поддерживает передачу информации со скоростью до 3 Гбит/с. Благодаря уменьшенному разъему SAS обеспечивает полное двухпортовое подключение как для 3,5-дюймовых, так и для 2,5-дюймовых дисковых накопителей (раньше эта функция была доступна только для 3,5-дюймовых дисковых накопителей с интерфейсом Fibre Channel).

Типичная система с интерфейсом SAS состоит из следующих компонентов:

- Инициаторы (Initiators) – устройства, которые порождают запросы на обслуживание для целевых устройств и получают подтверждения по мере исполнения запросов. Чаще всего инициатор выполняется в виде СБИС.
- Целевые устройства (Targets) содержат логические блоки и целевые порты, которые осуществляют приём запросов на обслуживание, исполняют их. После того, как закончена обработка запроса, инициатору запроса отсылается подтверждение выполнения запроса. Целевое устройство может быть как отдельным жёстким диском, так и целым дисковым массивом.
- Подсистема доставки данных (Service Delivery Subsystem) является частью системы ввода-вывода, которая осуществляет передачу данных между инициаторами и целевыми устройствами. Обычно подсистема доставки данных состоит из кабелей, которые соединяют инициатор и целевое устройство. Дополнительно, кроме кабелей в состав подсистемы доставки данных могут входить расширители SAS.

- Расширители (Expanders) – устройства, входящие в состав подсистемы доставки данных и позволяют облегчить передачи данных между устройствами SAS. Например, расширитель позволяет подключить несколько целевых устройств SAS к одному порту инициатора. Подключение через расширитель является абсолютно прозрачным для целевых устройств.

Спецификации на SAS регламентируют физический, канальный и логический уровни интерфейса.

3.8.3 Сравнение SAS и параллельного SCSI

- SAS использует последовательный протокол передачи данных между несколькими устройствами, и, таким образом, использует меньшее количество сигнальных линий.
- Интерфейс SCSI использует общую шину. Таким образом, все устройства подключены к одной шине, и с контроллером одновременно может работать только одно устройство. Интерфейс SAS использует соединения точка-точка, каждое устройство соединено с контроллером выделенным каналом.
- В отличие от SCSI, SAS не нуждается в терминации шины пользователем.
- В SCSI имеется проблема, связанная с тем, что время распространения сигнала по разным линиям, составляющим параллельный интерфейс, может отличаться. Интерфейс SAS лишён этого недостатка.
- SAS поддерживает большое количество устройств (> 16384), в то время как интерфейс SCSI поддерживает 8, 16, или 32 устройства на шине.
- SAS обеспечивает более высокую пропускную способность (1,5, 3,0 или 6,0 Гбит/с). Такая пропускная способность может быть обеспечена на каждом соединении инициатор–целевое устройство, в то время как на шине SCSI пропускная способность шины разделена между всеми подключёнными к ней устройствами.
- SAS поддерживает подключение устройств с интерфейсом SATA.
- SAS, также как и параллельный SCSI, использует команды SCSI для управления и обмена данными с целевыми устройствами.

3.8.4 Сравнение SAS и SATA

- SATA-устройства идентифицируются номером порта контроллера интерфейса SATA, в то время как устройства SAS идентифицируются их WWN-идентификаторами (World Wide Name). Для подключении SATA-устройства к домену SAS используется специальный протокол STP (Serial ATA Tunneled Protocol), описывающий согласование идентификаторов SAS и SATA.

- Устройства SATA 1 и SAS поддерживают тегированные очереди команд TCQ (Tagged Command Queuing). В то же время, устройства SATA версии 2 поддерживают как TCQ, так и Native Command Queuing (NCQ).
- SATA использует набор команд ATA, который позволяет работать с жёсткими дисками, в то время как SAS поддерживает более широкий набор устройств, в том числе жёсткие диски, сканеры, принтеры и др. (Накопители на оптическом диске, подключаемые через SATA, на самом деле являются целевыми устройствами SCSI, для доставки SCSI команд к которым используется SATA).
- Аппаратура SAS поддерживает связь инициатора с целевыми устройствами по нескольким независимым линиям: в зависимости от реализации можно повысить отказоустойчивость системы и/или увеличить скорость передачи данных. Интерфейс SATA версии 1 такой возможности не имеет. В то же время, интерфейс SATA версии 2 использует дубликаторы портов для повышения отказоустойчивости.
- Преимущество SATA состоит в низком энергопотреблении и невысокой стоимости оборудования, а интерфейса SAS – в большей надёжности.

3.9 Малые периферийные интерфейсы

3.9.1 Интерфейс RS-232

Интерфейс RS-232 – стандартный интерфейс, предназначенный для последовательной двоичной передачи данных между терминальным (DTE, Data Terminal Equipment) и связным (DCE, Data Communications Equipment) оборудованием [38, 37, 54, 7556, 89].

Ассоциация электронной промышленности (EIA) развивает стандарты по передаче данных. Стандарты EIA имеют префикс "RS". "RS" означает рекомендуемый стандарт, но сейчас стандарты просто обозначаются как "EIA" стандарты. RS-232 был введен в 1962, в 1969 была представлена третья редакция (RS-232C). Четвертая редакция была в 1987 (RS-232D, известная также под EIA-232D). RS-232 идентичен стандартам МККТТ (CCITT) V.24/V.28, X.20bis/X.21bis и ISO IS2110.

Рис. 88. Соединение двух удаленных терминалов при помощи модемов.

Чтобы не составить неправильного представления об интерфейсе RS-232, необходимо отчетливо понимать различие между этими видами оборудования. Терминальное оборудование, например микрокомпьютер, может посылать и

(или) принимать данные по последовательному интерфейсу. Оно как бы оканчивает (terminate) последовательную линию. Связное оборудование – устройства, которые могут упростить передачу данных совместно с терминальным оборудованием. Наглядным примером связного оборудования служит модем (модулятор–демодулятор). Он оказывается соединительным звеном в последовательной цепочке между компьютером и телефонной линией.

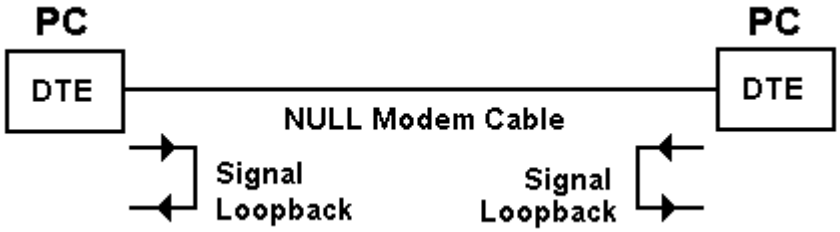


Рис. 89. Соединение двух терминалов при помощи null-modem кабеля.

Различие между терминальными и связными устройствами довольно расплывчато, поэтому возникают некоторые сложности в понимании того, к какому типу оборудования относится то или иное устройство. Рассмотрим ситуацию с принтером. К какому оборудованию его отнести? Как связать два компьютера, когда они оба действуют как терминальное оборудование.

Для ответа на эти вопросы следует рассмотреть физическое соединение устройств. Информация передается по проводам с уровнями сигналов, отличающимися от обычных уровней цифрового сигнала (5В, 3,3В и т.п.), для обеспечения большей устойчивости к помехам. Асинхронная передача данных осуществляется с установленной скоростью при синхронизации уровнем сигнала стартового импульса. RS-232 используется для передачи данных на небольшое расстояние (единицы - десятки метров) с небольшой скоростью (обычно, не быстрее 115200 бит/с). Для формирования уровня сигнала используются микросхемы приёмопередатчиков, а для формирования и распознавания посылок – микросхемы UART.

Модуль универсального синхронно-асинхронного приемопередатчика (Universal Synchronous/Asynchronous Receiver and Transmitter, USART) стал стандартом «де-факто» среди контроллеров последовательного обмена. В названии часто опускают слово «синхронный», и модуль не совсем корректно именуется UART (чисто асинхронные приемопередатчики сейчас встречаются достаточно редко). Характеристики последовательного порта UART не позволяют производить приём и передачу данных за пределы печатной платы. Для связи с другими устройствами сигнал от UART необходимо пропустить через приёмопередатчик, работающий в одном из стандартов: RS-232, RS-485, RS-422.

Обычно модули UART в асинхронном режиме поддерживают протокол обмена для интерфейса RS-232 (8N1 или 9N1); в синхронном режиме – нестандартные синхронные протоколы, в некоторых случаях – протокол SPI.

Приемопередатчик – преобразователь уровня, как правило, в интегральном исполнении. Предназначен для преобразования электрических сигналов из уровня ТТЛ в уровень, соответствующий физическому уровню определенного стандарта.

Контроллер UART обычно содержит:

1. Источник тактирования (обычно с увеличенной частотой тактирования по сравнению со скоростью обмена, чтобы иметь возможность отслеживать состояние линии передачи данных в середине передачи бита).
2. Входные и выходные сдвиговые регистры.
3. Регистры управления приемом/передачей данных, чтением/записью.
4. Буферы приема/передачи.
5. Параллельная шина данных для буферов приема/передачи.
6. FIFO-буферы памяти (опционально).

Ошибки UART:

1. Overrun Error (ошибка из-за повышенной скорости передачи, переполнение буфера приема).

Эта ошибка случается, когда приемник UART не успевает обрабатывать приходящие из канала символы, т. е. буфер переполняется.

2. Framing Error (ошибка кадрирования).

Эта ошибка случается, когда фиксируется некорректное состояние линии данных в момент передачи старт- или стоп-бита. Например, после передачи 8 бит данных приемник ожидает перехода линии в стоп-состояние, но этого не происходит.

3. Break Condition (сигнал прерывания передачи, разрыва связи).

Этот сигнал информирует о том, что входная линия данных находилась в неизменном нулевом состоянии в течение времени, больше передачи одного символа. В буфере приема нулевой байт. Некоторые устройства используют такую последовательность, чтобы сообщить передатчику, например, о переходе на другую скорость обмена данными.

Существуют и другие ошибки, отслеживаемые контроллером UART.

3.9.1.1 Сигнальные линии последовательного интерфейса

Произведя незначительные изменения в линиях интерфейса RS-232, можно заставить связное оборудование функционировать как терминальное. Чтобы разобраться в том, как это сделать, нужно проанализировать функции сигналов интерфейса RS-232C.

Таблица 3. Функции сигнальных линий интерфейса RS-232.

Номер контакта	Сокращение	Направление	Полное название
1	FG	-	Основная или защитная земля
2	TD (TXD)	К DCE	Передаваемые данные
3	RD (RXD)	К DTE	Принимаемые данные
4	RTS	К DCE	Запрос передачи
5	CTS	К DTE	Сброс передачи
6	DSR	К DTE	Готовность модема
7	SG	-	Сигнальная земля
8	DCD	К DTE	Обнаружение несущей данных
9	-	К DTE	(Положительное контрольное напряжение)
10	-	К DTE	(Отрицательное контрольное напряжение)
11	QM	К DTE	Режим выравнивания
12	SDCD	К DTE	Обнаружение несущей вторичных данных
13	SCTS	К DTE	Вторичный сброс передачи
14	STD	К DCE	Вторичные передаваемые данные
15	TC	К DTE	Синхронизация передатчика
16	SRD	К DTE	Вторичные принимаемые данные
17	RC	К DTE	Синхронизация приемника
18	DCR	К DCE	Разделенная синхронизация приемника
19	SRTS	К DCE	Вторичный запрос передачи
20	DTR	К DCE	Готовность терминала
21	SQ	К DTE	Качество сигнала
22	RI	К DTE	Индикатор звонка
23	-	К DCE	(Селектор скорости данных)
24	TC	К DCE	Внешняя синхронизация передатчика
25	-	К DCE	(Занятость)

Примечания:

1. Линии 11, 18, 25 обычно считают незаземленными. Приведенная в таблице спецификация относится к спецификациям Bell 113B и 208A.
2. Линии 9 и 10 используются для контроля отрицательного (MARK) и положительного (SPACE) уровней напряжения.
3. Во избежание путаницы между RD (Read – считывать) и RD (Received Data – принимаемые данные) будут использоваться обозначения RXD и TXD, а не RD и TD.

Терминальное оборудование обычно оснащено разъемом со штырьками, а связное – разъемом с отверстиями (но могут быть и исключения).

Рис. 90. Стандартный последовательный порт RS-232C имеет форму 25-контактного разъема типа D

Сигналы интерфейса RS-232C подразделяются на следующие классы:

- Последовательные данные (например, TXD, RXD). Интерфейс RS-232C обеспечивает два независимых последовательных канала данных: первичный (главный) и вторичный (вспомогательный). Оба канала могут работать в дуплексном режиме, т.е. одновременно осуществляют передачу и прием информации.
- Управляющие сигналы квитирования (например, RTS, CTS). Сигналы квитирования – средство, с помощью которого обмен сигналами позволяет DTE начать диалог с DCE до фактической передачи или приема данных по последовательной линии связи.
- Сигналы синхронизации (например, TC, RC). В синхронном режиме (в отличие от более распространенного асинхронного) между устройствами необходимо передавать сигналы синхронизации, которые упрощают синхронизм принимаемого сигнала в целях его декодирования.

На практике вспомогательный канал (вторичный) RS-232C применяется редко, и в асинхронном режиме вместо 25 линий используются 9 линий (таблица 2).

Таблица 4. Основные линии интерфейса RS-232C (EIA-574, DB9)

Номер контакта	Сигнал	Выполняемая функция
1	FG	Подключение земли к стойке или шасси оборудования

2	TXD	Последовательные данные, передаваемые от DTE к DCE
3	RXD	Последовательные данные, принимаемые DTE от DCE
4	RTS	Требование DTE послать данные к DCE
5	CTS	Готовность DCE принимать данные от DTE
6	DSR	Сообщение DCE о том, что связь установлена
7	SG	Возвратный тракт общего сигнала (земли)
8	DCD	DTE работает и DCE может подключиться к каналу связи

Таблица 5. Основные линии интерфейса RS-232C (EIA-561, RJ-45)

Номер контакта	Сигнал	Выполняемая функция
1	RI	Индикатор вызова
2	CD	Обнаружение несущей данных
3	DTR	Готовность терминала
4	GND	Возвратный тракт общего сигнала (земли)
5	RxD	Последовательные данные, принимаемые DTE от DCE
6	TxD	Последовательные данные, передаваемые от DTE к DCE
7	CTS	Готовность DCE принимать данные от DTE
8	RTS	Требование DTE послать данные к DCE

3.9.1.2 Управление потоком

Иногда устройство не может обработать принимаемые данные от компьютера или другого устройства. Устройство использует управление потоком для прекращения передачи данных. Могут использоваться аппаратное или программное управление потоком.

Аппаратное управление потоком

Аппаратный протокол управления потоком RTS/CTS. Он использует дополнительно два провода в кабеле, а не передачу специальных символов по линиям данных. Поэтому аппаратное управление потоком не замедляет обмен в отличие от протокола Xon-Xoff. При необходимости послать данные компьютер устанавливает сигнал на линии RTS. Если приемник (модем) готов к приему данных, то он отвечает установкой сигнала на линии CTS, и компьютер начинает посылку данных. При неготовности устройства к приему сигнал CTS не устанавливается.

Программное управление потоком

Программный протокол управления потоком Xon/Xoff использует два символа: Xon и Xoff. Код ASCII символа Xon – 17, а ASCII код Xoff – 19. Модем имеет маленький буфер, поэтому при его заполнении модем посылает символ Xoff компьютеру для прекращения посылки данных. При появлении возможности приема данных посылается символ Xon и компьютер продолжит пересылку данных. Этот тип управления имеет преимущество в том, что не требует дополнительных линий, так как символы передаются по линиям TXD/RXD. Но на медленных соединениях это может привести к значительному замедлению соединения, так как каждый символ требует 10 битов.

3.9.1.3 Разъемы и кабели

Устройства для связи по последовательному каналу соединяются кабелями с 9-ю или 25-ю контактными разъёмами типа D-sub. Обычно они обозначаются DE-9 (или некорректно: DB-9), DB-25, CANNON 9, CANNON 25.

Первоначально в RS-232 использовались DB-25, но, поскольку многие приложения использовали лишь часть предусмотренных стандартом контактов, стало возможно применять для этих целей 9-штырьковые разъёмы DE-9 (D-subminiature).

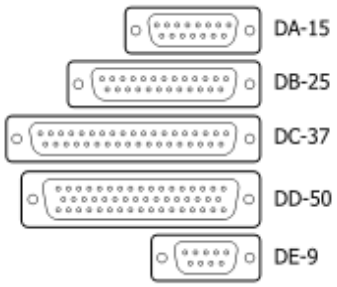


Рис. 91. Разъемы серии D-sub.

D-subminiature, или D-sub – название электрического разъёма, применяемого, в частности, в компьютерной технике. Название «субминиатюрный» было уместно тогда, когда эти разъёмы только появились, в наше же время эти разъёмы относятся к числу наибольших по размерам из используемых в компьютерах.

Разъём D-sub содержит два или более параллельных рядов контактов или гнёзд, обычно окружённых металлическим экраном в форме латинской D, который обеспечивает механическое крепление соединения и экранирует от электромагнитных помех. Форма разъёма в виде буквы D предохраняет от неправильной ориентации разъёма. Часть разъёма, содержащая контакты, называется по-английски male connector, или plug (по-русски штекер, или вилка, хотя чаще в данном контексте используется жаргонный термин «папа»), а часть, содержащая гнёзда — female connector, или socket (розетка или «мама»). Экран розетки плотно входит внутрь экрана вилки. Если используются

экранированные кабели, экраны разъёмов соединяются с экранами кабелей, обеспечивая, таким образом, непрерывное экранирование для всего соединения.

Разъёмы D-sub были изобретены и введены в употребление фирмой ITT Cannon, подразделением ITT Corporation в 1952 году. В принятой этой фирмой системе обозначений буква D обозначает всю серию разъёмов D-sub, а вторая буква используется для указания размера разъёма, исходя из числа стандартных контактов, которые могут разместиться внутри D-образного экрана (A = 15 контактов, B = 25, C = 37, D = 50, E = 9), после чего следует цифра, обозначающая фактическое число используемых контактов, и буква, обозначающая «пол» разъёма (M – male, «папа» , F – female, «мама», P – plug, штепсель или «папа», S – socket, розетка или «мама»).

Вероятно, потому, что в оригинальном PC как для параллельного, так и для последовательного портов использовались разъёмы DB-25, многие, не понимая, что «B» в данном случае означает размер экрана, стали сам разъём D-sub называть DB, вместо того, чтобы использовать обозначения «DA», «DC» или «DE». Когда для последовательного порта стали использовать 9-штырьковые разъёмы, их начали называть DB9 вместо DE9. Сейчас достаточно распространено, что разъёмы DE9 продаются, как DB9. Под DB9 в современном мире почти всегда подразумевают 9-штырьковый разъём с размером экрана E.

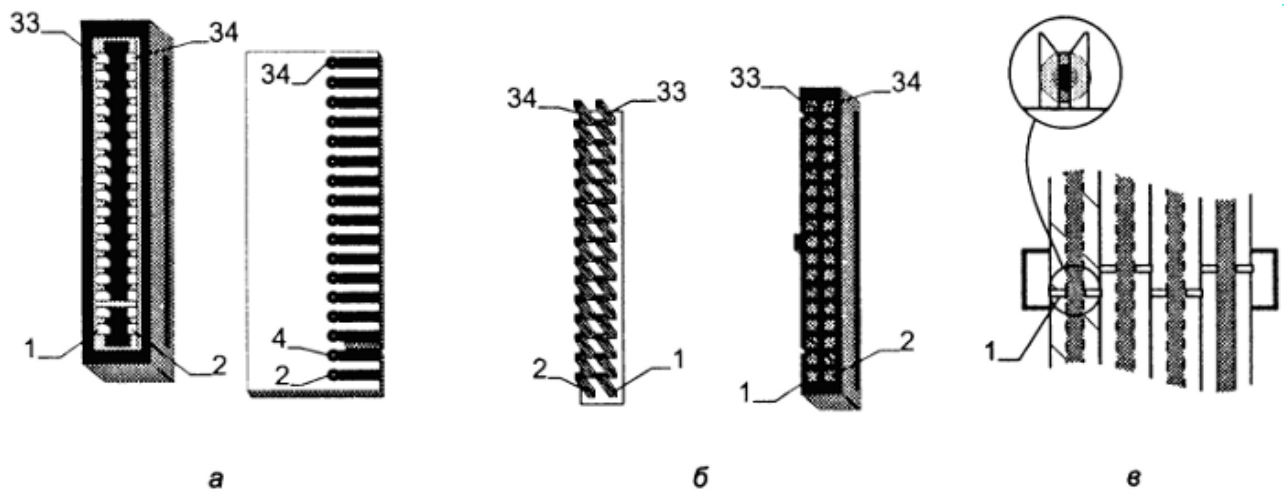


Рис. 92. Разъемы IDC: а - краевые, б - штырьковые, в - заделка проводов

Разъемы IDC (Insulation-Displacement Connector – разъем, смещающий изоляцию) получили название от способа присоединения кабеля. Контакты этих разъемов со стороны, обращенной к кабелю, имеют ножи, подрезающие и смещающие изоляцию проводников кабеля. Эти разъемы предназначены в основном для ленточных кабелей-шлейфов, хотя в них возможна заделка и одиночных проводников. Для заделки кабелей в эти разъемы существуют специальные инструменты-прессы, но при необходимости можно обойтись плоской отверткой (и умелыми руками). Разъемы IDC существуют как ответные части для краевых печатных разъемов (рис. выше, а) и штырьковых контактов (рис. выше, б). Разъемы могут иметь ключи: для печатных разъемов это прорезь и соответствующая ей перемычка, расположенная ближе к первым контактам.

Для штырьковых разъемов ключом является выступ на корпусе, но этот ключ сработает, только если ответная часть имеет пластмассовый бандаж с прорезью. Ключом может являться и отсутствующий штырек – на разъеме для него не оставляют отверстия. На ленточном кабеле крайний провод, соединяемы с контактом «1», маркируют цветной краской (например, красной). На печатной плате штырек «1» обычно имеет отличающуюся от формы других (квадратную) форму контактной площадки. Такого типа разъем используется в учебном лабораторном стенде SDK-1.1 для подключения по интерфейсу RS-232.

3.9.1.4 Формат последовательной передачи данных

Поскольку данные обычно представлены на шине микропроцессора в параллельной форме (байтами, словами), их последовательный ввод-вывод оказывается несколько сложным. Для последовательного ввода потребуется средства преобразования последовательных входных данных в параллельные данные, которые можно поместить на шину. С другой стороны, для последовательного вывода необходимы средства преобразования параллельных данных, представленных на шине, в последовательные выходные данные. В первом случае преобразование осуществляется регистром сдвига с последовательным входом и параллельным выходом (SIPO), а во втором – регистром сдвига с параллельным входом и последовательным выходом (PISO). Оба регистра обычно входят в состав приемопередатчика USART.

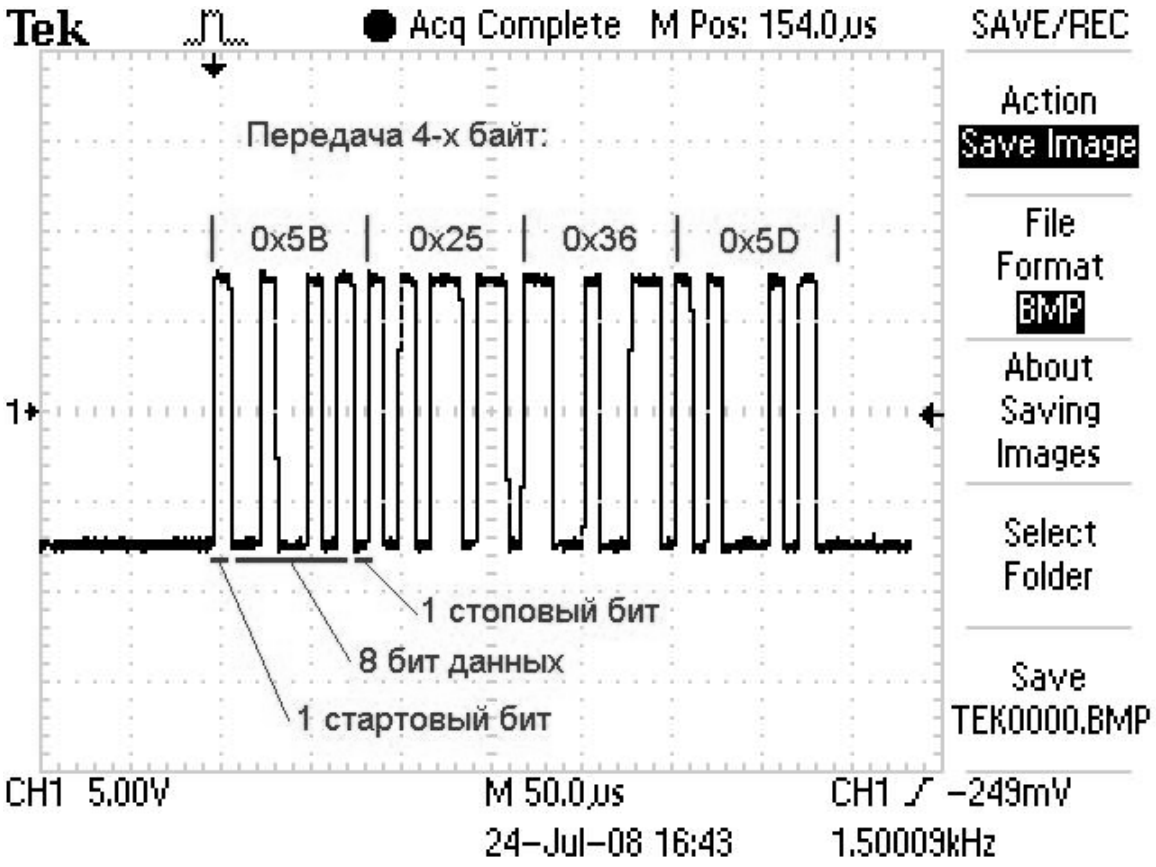


Рис. 93. Так выглядит на экране осциллографа последовательная передача данных по интерфейсу RS-232.

Последовательные данные передаются в синхронном или асинхронном режимах. В синхронном режиме все передачи осуществляются под управлением общего сигнала синхронизации, который должен присутствовать на обоих концах линии связи. Асинхронная передача подразумевает передачу данных пакетами; каждый пакет содержит необходимую информацию, требующуюся для декодирования содержащихся в нем данных. Конечно, второй режим сложнее, но у него есть серьезное преимущество: не нужен отдельный сигнал синхронизации.

В асинхронном режиме посылке очередного байта информации предшествует специальный старт-бит, сигнализирующий о начале передачи (обычно логический «0»). Затем следуют биты данных (их обычно 8), за которыми может следовать дополнительный бит (его наличие зависит от режима передачи, обычно этот бит выполняет функцию контроля четности). Завершается посылка стоп-битом (логическая «1»), длина которого (длительность единичного состояния линии) может соответствовать длительности передачи 1, 1.5 («полтора стоп-бита») или 2 бита (см. рис. выше). Стоп-бит гарантирует некоторую выдержку между соседними посылками, при этом пауза между ними может быть сколь угодно долгой (без учета понятия «тайм-аута»).

Контроль четности

Контроль четности может быть использован для обнаружения ошибок при передаче данных. При использовании контроля четности посылаются сообщения, подсчитывающие число единиц в группе бит данных. В зависимости от результата устанавливается бит четности. Приемное устройство также подсчитывает число единиц и затем сверяет бит четности. Для обеспечения контроля четности компьютер и устройство должны одинаково производить подсчет бита четности, т.е. определиться устанавливать бит при четном (even) или нечетном (odd) числе единиц. При контроле на четность биты данных и бит четности всегда должны содержать четное число единиц. В противоположном случае выполняется контрол на нечетность. Часто в драйверах UART RS-232 реализуются еще две опции на четность: Mark и Space. Эти опции не влияют на возможность контроля ошибок: Mark означает, что устройство всегда устанавливает бит четности в 1, а Space – всегда в 0.

Обнаружение ошибок

Проверка на четность – это простейший способ обнаружения ошибок. Он может определить возникновение ошибок в одном бите, но при наличии ошибок в двух битах уже не заметит ошибок. Также такой контроль не отвечает на вопрос, какой бит ошибочный. Другой механизм проверки включает в себя старт- и стоп-биты, циклические проверки на избыточность, которые часто применяются в соединениях Modbus.

Рассмотрим пример. В этом примере показана структура передаваемых данных со синхронизирующим тактовым сигналом. В этом примере

используется 8 бит данных, бит четности и стоп-бит. Такая структура также обозначается 8E1.

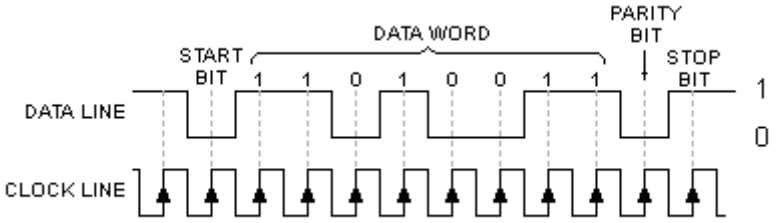


Рис. 94. Фомат передачи данных по интерфейсу RS-232.

Примечание: тактовый сигнал используется для асинхронной передачи (это внутренний сигнал UART).

Старт-бит

Сигнальная линия может находиться в двух состояниях: включена и выключена. Линия в состоянии ожидания всегда включена. Когда устройство или компьютер хотят передать данные, они переводят линию в состояние выключено – это установка старт-бита. Биты сразу после старт-бита являются битами данных.

Стоп-бит

Стоп-бит позволяет устройству или компьютеру произвести синхронизацию при возникновении сбоев. Например, помеха на линии скрыла старт-бит. Период между старт- и стоп-битами постоянен согласно значению скорости обмена, числу бит данных и бита четности. Стоп-бит всегда включен. Если приемник определяет выключенное состояние, когда должен присутствовать стоп-бит, фиксируется появление ошибки. Стоп-бит не просто один бит минимального интервала времени в конце каждой передачи данных. На компьютерах обычно он эквивалентен 1 или 2 битам, и это должно учитываться в программе драйвера. Хотя, 1 стоп-бит наиболее общий, выбор 2 бит в худшем случае немного замедлит передачу сообщения.

Есть возможность установки значения стоп бита равным 1,5. Это используется при передаче менее 7 битов данных. В этом случае не могут быть переданы символы ASCII, и поэтому значение 1,5 используется редко.

3.9.1.5 Работа с последовательным каналом

Простым примером асинхронного обмена с программной проверкой готовности может служить работа с контроллером последовательного канала (UART) «по опросу»: перед тем, как прочитать данные из порта данных контроллера, необходимо проверить, являются ли эти данные результатом приема посылки и не забирались ли они программой ранее. Проще говоря, необходимо проверить данные на достоверность. Перед тем же, как записывать данные для передачи в буфер контроллера, необходимо убедиться, что в буфере

есть место, т.е. что запись новых данных в буфер не приведет к уничтожению ранее помещенных и еще не переданных данных.

При организации асинхронного обмена по прерыванию при приеме байта с линии происходит прерывание и передача управления соответствующей программе-обработчику, который читает принятый байт из порта данных контроллера UART и, к примеру, помещает его в специальный буфер-очередь принятых байт, доступный прерванной программе. По завершении процедуры обработки прерывания управление передается в прерванную программу, которая при желании (в любом удобном месте алгоритма) может забрать принятый байт. С другой стороны, при завершении отправки контроллером очередного байта также происходит прерывание, сигнализирующее о том, что байт послан и в буфер UART можно поместить новые данные. Обработчик прерывания при наличии данных в исходящей очереди записывает очередной байт в порт данных контроллера и запускает посылку. Основная же программа может, не заботясь о готовности или неготовности контроллера, принять очередной байт, может спокойно помещать данные в исходящий буфер, а всю работу с устройством выполнит обработчик прерываний, когда оно (устройство) будет готово.

3.9.2 Интерфейс SPI

SPI (Serial Peripheral Interface) – последовательный синхронный стандарт передачи данных в режиме полного дуплекса, разработанный компанией Motorola для обеспечения простого и недорогого сопряжения микроконтроллеров и периферии. SPI также иногда называют четырёхпроводным (four-wire) интерфейсом.

Интерфейс SPI, наряду с I^2C, относится к самым широко используемым интерфейсам для соединения микросхем. Изначально он был придуман компанией Motorola, а в настоящее время используется в продукции многих производителей [71]. Шина SPI организована по принципу «ведущий-подчиненный». В качестве ведущего шины обычно выступает микроконтроллер, но им также может быть программируемая логика, DSP-контроллер или специализированная интегральная схема. В роли подчиненных устройств выступают различного рода микросхемы, в том числе запоминающие устройства (EEPROM, Flash-память, SRAM), часы реального времени (RTC), АЦП/ЦАП, температурные датчики, сенсорные экраны, цифровые потенциометры, коммуникационные контроллеры (Ethernet, USB, CAN, IEEE 802.15.4, IEEE 802.11), ЖКИ, мультимедийные карты (SD, MMC) и др.

Кроме того, интерфейс SPI является основой для построения ряда специализированных интерфейсов, в том числе, интерфейса JTAG и интерфейсов карт Flash-памяти (мультимедийные карты SD и MMC).

Главным составным блоком интерфейса SPI является обычный сдвиговый регистр, сигналы синхронизации и ввода-вывода битового потока, которые и образуют интерфейсные сигналы. Таким образом, протокол SPI правильнее

назвать не протоколом передачи данных, а протоколом обмена данными между двумя сдвиговыми регистрами, каждый из которых одновременно выполняет и функцию приемника, и функцию передатчика. Непременным условием передачи данных по шине SPI является генерация сигнала синхронизации шины. Этот сигнал имеет право генерировать только ведущий шины и от этого сигнала полностью зависит работа подчиненного шины. Принимающая периферия (ведомая) синхронизирует получение битовой последовательности с тактовым сигналом. К одному последовательному периферийному интерфейсу ведущего устройства-микросхемы может присоединяться несколько микросхем. Ведущее устройство выбирает ведомое для передачи, активируя сигнал «выбор кристалла» (chip select) на ведомой микросхеме. Периферия, не выбранная процессором, не принимает участие в передаче по SPI.

Частота работы интерфейса SPI составляет 1-70МГц.

3.9.2.1 Типы подключения к шине SPI

Существует три типа подключения к шине SPI, в каждом из которых участвуют четыре сигнала (см. таблицу).

Таблица 6. Цифровые сигналы шины SPI.

Ведущий шины			**Подчиненный шины**		
Основное обозначение	**Альтернативное обозначение**	**Описание**	**Основное обозначение**	**Альтернативное обозначение**	**Описание**
MOSI (Master Output Slave Input)	DO, SDO, DOUT	Выход последовательной передачи данных	MOSI	DI, SDI, DIN	Вход последовательной приема данных
MISO (Master Input Slave Output)	DI, SDI, DIN	Вход последовательного приема данных	MISO	DO, SDO, DOUT	Выход последовательной передачи данных
SCLK	DCLOCK, CLK, SCK	Выход синхронизации передачи данных	SCLK	DCLOCK, CLK, SCK	Вход синхронизации приема данных
SS (Slave Select)	CS (Chip Select)	Выход выбора подчиненного (выбор микросхемы)	SS	CS	Вход выбора подчиненного (выбор микросхемы)

Самое простое подключение, в котором участвуют только две микросхемы, показано на рис. ниже. Здесь ведущий шины передает данные по линии MOSI синхронно со сгенерированным им же сигналом SCLK, а подчиненный захватывает переданные биты данных по определенным фронтам принятого сигнала синхронизации. Одновременно с этим подчиненный отправляет свою посылку данных. Представленную схему можно упростить исключением линии MISO, если используемая подчиненная интегральная схема (ИС) не предусматривает ответную передачу данных или в ней нет потребности.

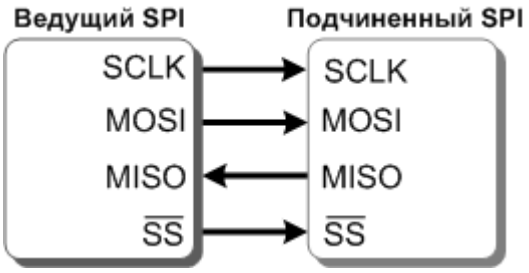


Рис. 95. Простейшее подключение к шине SPI.

Одностороннюю передачу данных можно встретить у таких микросхем, как ЦАП, цифровые потенциометры, программируемые усилители и драйверы. Таким образом, рассматриваемый вариант подключения подчиненной ИС требует 3 или 4 линии связи. Чтобы подчиненная ИС принимала и передавала данные, помимо наличия сигнала синхронизации необходимо также, чтобы линия SS была переведена в низкое состояние. В противном случае подчиненная ИС будет неактивна. Когда используется только одна внешняя ИС, может возникнуть соблазн исключения и линии SS за счет жесткой установки низкого уровня на входе выбора подчиненной микросхемы. Такое решение крайне нежелательно и может привести к сбоям или вообще невозможности передачи данных, так как вход выбора микросхемы служит для перевода ИС в её исходное состояние и иногда инициирует вывод первого бита данных.

При необходимости подключения к шине SPI нескольких микросхем используется либо независимое (параллельное) подключение, либо каскадное (последовательное). Независимое подключение более распространенное, так как достигается при использовании любых SPI-совместимых микросхем. Здесь все сигналы, кроме выбора микросхем, соединены параллельно, а ведущий шины переводом того или иного сигнала SS в низкое состояние задает, с какой подчиненной ИС он будет обмениваться данными. Главным недостатком такого подключения является необходимость в дополнительных линиях для адресации подчиненных микросхем (общее число линий связи равно 3+n, где n – количество подчиненных микросхем).

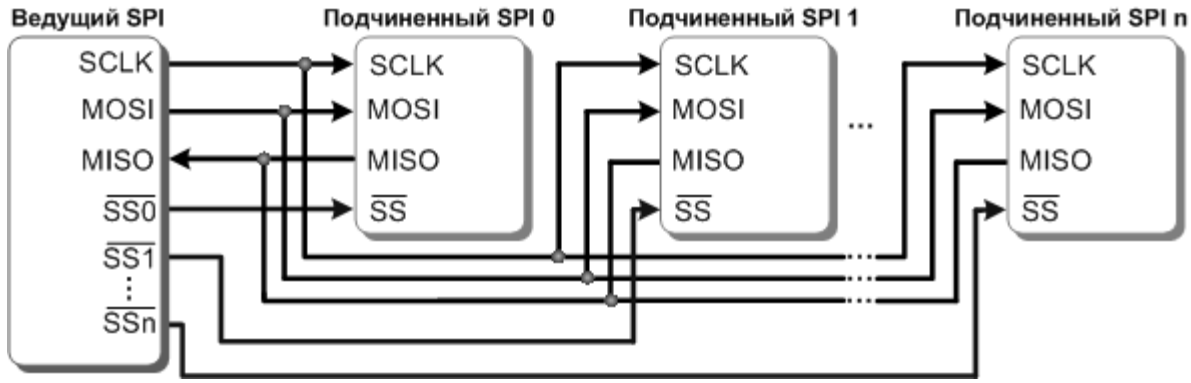


Рис. 96. Независимое подключение к шине SPI.

Каскадное включение избавлено от этого недостатка, так как здесь из нескольких микросхем образуется один большой сдвиговый регистр. Для этого выход передачи данных одной ИС соединяется со входом приема данных другой, как показано на рисунке ниже. Входы выбора микросхем здесь соединены параллельно и, таким образом, общее число линий связи сохранено равным 4. Однако использование каскадного подключения возможно только в том случае, если его поддержка указана в документации на используемые микросхемы. Чтобы выяснить это, важно знать, что такое подключение по-английски называется «daisy-chaining» (по-русски – «дейзи-цепочка»).

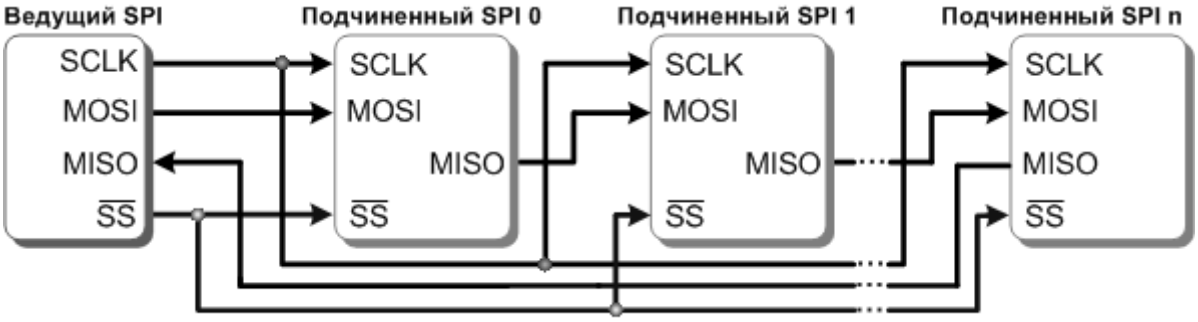


Рис. 97. Каскадное подключение к шине SPI.

3.9.2.2 Режимы работы шины SPI

Во время каждого цикла обмена по SPI происходит полнодуплексная передача данных:

- Ведущий выставляет бит данных на линию MOSI; ведомый читает его с этой же линии.
- Ведомый выставляет бит на линию MISO; ведущий читает его с этой же линии.

Не во всех случаях передачи данных все эти операции имеют значение, но они все равно выполняются.

Протокол передачи по интерфейсу SPI предельно прост и, по сути, идентичен логике работы сдвигового регистра, которая заключается в выполнении операции сдвига и, соответственно, побитного ввода и вывода данных по определенным фронтам сигнала синхронизации. Получается, что

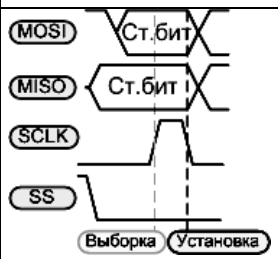
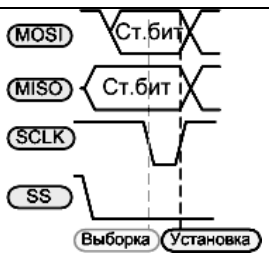
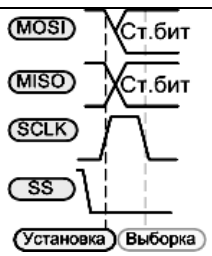
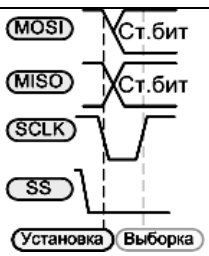
сдвиговые регистры ведущего и ведомого устройств образуют кольцо. Данные обычно выдвигаются на линию старшим битом вперед. После передачи 8 тактов по линии синхронизации сдвиговые регистры ведущего и ведомого полностью обменяются данными (байтами). Далее полученные данные, если они являются значимыми, сохраняются в памяти этих устройств. Если есть еще байты для передачи, цикл повторяется. Цикл обмена данными по SPI может занимать сколько угодно тактов и начинается/заканчивается выбором/отменой выбора подчиненного устройства.

В один и тот же момент времени ведущий может обмениваться данными только с одним ведомым.

Кроме установки скорости обмена ведущий должен определить, по каким фронтам сигнала синхронизации будет выполняться чтение, а по каким – запись данных. Установка данных при передаче и выборка при приеме всегда выполняются по противоположным фронтам синхронизации. Это необходимо для гарантирования выборки данных после надежного их установления. Если к этому учесть, что в качестве первого фронта в цикле передачи может выступать нарастающий или падающий фронт, то всего возможно четыре варианта логики работы интерфейса SPI. Эти варианты получили название режимов SPI и описываются двумя параметрами:

- CPOL (полярность синхронизации) – исходный уровень сигнала синхронизации. Если CPOL=0, то линия синхронизации до начала цикла передачи и после его окончания имеет низкий уровень (т.е. первый фронт нарастающий, а последний – падающий). Если CPOL=1, – высокий (т.е. первый фронт падающий, а последний – нарастающий).
- CPHA (фаза синхронизации). От этого параметра зависит, в какой последовательности выполняется установка и выборка данных. Если CPHA=0, то по переднему фронту в цикле синхронизации будет выполняться выборка данных, а затем, по заднему фронту, – установка данных. Если же CPHA=1, то установка данных будет выполняться по переднему фронту в цикле синхронизации, а выборка – по заднему. Информация по режимам SPI обобщена в таблице ниже.

Таблица 7. Режимы SPI.

Режимы SPI	**0**	**1**	**2**	**3**
CPOL	0	1	0	1
CPHA	0	0	1	1
Временная диаграмма первого цикла синхронизации				

3.9.2.3 Достоинства шины SPI

1. Скорость передачи выше, чем у I^2C, SMBus. Предельная простота протокола передачи на физическом уровне обуславливает высокую надежность и быстродействие передачи. Предельное быстродействие шины SPI измеряется десятками мегагерц и, поэтому, она идеальна для потоковой передачи больших объемов данных и широко используется в высокоскоростных ЦАП/АЦП, драйверах светодиодных дисплеев и микросхемах памяти.
2. Протяженность SPI сравнима с интерфейсами RS-232, RS-485, CAN.
3. Полнодуплексный обмен.
4. Протокол передачи битовый, а значит, гибкий для реализации и назначения:
   - Не ограничен передачей 8-битовых слов (можно и 12-битовые, и 16-битовые).
   - Можно выбирать размер, содержимое и назначение пакетов передачи, т.е. определяется прикладной задачей, а не ограничивается стандартом (в отличие от I^2C).
5. Очень простая аппаратная реализация интерфейса:
   - Пониженное энергопотребление по сравнению с I^2C (никаких pullup-регистров).
   - Никаких механизмов арбитража (разрешения конфликтных ситуаций) и соответствующих им состояний отказа.
   - Ведомые устройства используют сигналы тактирования ведущего и не нуждаются в точных источниках тактирования.
   - Ведомый не нуждается в уникальном адресе по сравнению с I^2C.
   - Все линии шины SPI являются однонаправленными, что существенно упрощает решение задачи преобразования уровней и гальванической изоляции микросхем.

3.9.2.4 Недостатки шины SPI

1. Больше линий, чем в I^2C.
2. Нет адресации в протоколе обмена, для каждого нового ведомого устройства требуется отдельная сигнальная линия выбора (SS).
3. Никакой аппаратной реализации подтверждения наличия ведомого устройства, т.е. ведущий может «разговаривать» с пустотой и не знать об этом.
4. Может быть только один мастер (нет режима мультимастера, в отличие от I^2C).
5. Никакой аппаратной поддержки управления потоком данных.
6. Как такового стандарта не существует, например, как у I^2C. Протокол I^2C является более стандартизованным, поэтому, пользователь I^2C-

микросхем более защищен от проблем несовместимости выбранных компонентов.

3.9.3 Интерфейс Centronics

Порт параллельного интерфейса был введен в персональный компьютер для подключения принтера – отсюда и пошло его название LPT-порт (Line PrinTer – построчный принтер). Традиционный, он же стандартный, LPT-порт называется стандартным параллельным портом (Standard Parallel Port, SPP), или SPP-портом, и является однонаправленным портом, через который программно реализуется протокол обмена Centronics. Название и назначение сигналов разъема порта соответствуют интерфейсу Centronics. SPP-порт ориентирован на вывод данных, хотя с некоторыми ограничениями позволяет и вводить данные. Скорость передачи данных может варьироваться и достигать 1,2 Мбит/с.

Существуют различные модификации LPT-порта: двунаправленный, EPP, ECP и др., расширяющие его функциональные возможности, повышающие производительность и снижающие нагрузку на процессор. Поначалу они являлись фирменными решениями отдельных производителей, позднее был принят стандарт IEEE 1284.

К LPT-портам подключают принтеры, плоттеры, сканеры, коммуникационные устройства и устройства хранения данных, а также электронные ключи, программаторы и прочие устройства. Иногда параллельный интерфейс используют для связи между двумя компьютерами.

Таблица 8. Упрощённая таблица сигналов интерфейса Centronics.

Контакты DB-25 IEEE 1284-A	Контакты Centronics IEEE 1284-B	Обозначение	Примечание	Функция
1	1	Strobe	Маркер цикла передачи (выход)	Управление
2	2	Data 1	Сигнал 1 (выход)	Данные
3	3	Data 2	Сигнал 2 (выход)	Данные
4	4	Data 3	Сигнал 3 (выход)	Данные
5	5	Data 4	Сигнал 4 (выход)	Данные
6	6	Data 5	Сигнал 5 (выход)	Данные
7	7	Data 6	Сигнал 6 (выход)	Данные
8	8	Data 7	Сигнал 7 (выход)	Данные
9	9	Data 8	Сигнал 8 (выход)	Данные
10	10	Acknowledge	Готовность принять (вход)	Состояние
11	11	Busy	Занят (вход)	Состояние
12	12	Paper End	Нет бумаги (вход)	Состояние

13	13	Select	Выбор (вход)	Состояние
14	14	Auto Feed	Автоподача (выход)	Управление
15	32	Error	Ошибка (вход)	Состояние
16	31	Init	Инициализация (выход)	Управление
17	36	Select In	Управление печатью (выход)	Управление
18-25	16-17, 19-30	GND	Общий	Земля

С внешней стороны порт имеет 8-битную шину данных, 5-битную шину сигналов состояния и 4-битную шину управляющих сигналов, выведенные на разъем-розетку DB-25S. В LPT-порте используются логические уровни ТТЛ, что ограничивает допустимую длину кабеля из-за невысокой помехозащищенности ТТЛ-интерфейса. Гальваническая развязка отсутствует, схемная земля подключаемого устройства соединяется со схемной землей компьютера. Из-за этого порт является уязвимым местом компьютера, страдающим при нарушении правил подключения и заземления устройств. Поскольку порт обычно располагается на системной плате, в случае его «выжигания» зачастую выходит из строя и его ближайшее окружение вплоть до выгорания всей системной платы.

Адаптер SPP-порта содержит три 8-битных регистра, расположенных по соседним адресам в пространстве ввода-вывода, начиная с базового адреса порта BASE (3BCh, 378h или 278h).

Data Register (DR) – регистр данных, адрес=BA5E. Данные, записанные в этот регистр, выводятся на выходные линии Data[7:0]. Данные, считанные из этого регистра, в зависимости от схемотехники адаптера соответствуют либо ранее записанным данным, либо сигналам на тех же линиях, что не всегда одно и то же.

Status Register (SR) – регистр состояния (только чтение), адрес=BA5E+1. Регистр отображает 5-битный порт ввода сигналов состояния принтера и флаг прерывания: состояние готовности принтера, сигнал о конце бумаги в принтере, сигнал о включении принтера, сигнал о любой ошибке принтера и др.

Control Register (CR) – регистр управления, адрес=BA5E+2, допускает запись и чтение. Регистр связан с 4-битным портом вывода управляющих сигналов, для которых возможно и чтение; выходной буфер обычно имеет тип «открытый коллектор». При помощи этого регистра можно подавать такие сигналы, как сигнал аппаратного сброса принтера, сигнал на автоматический перевод строки по приему байта и возврата каретки (CR), сигнал стробирования выходных данных и др.

Перечислим шаги процедуры вывода байта по интерфейсу Centronics с указанием требуемого количества шинных операций процессора:

1. Вывод байта в регистр данных (1 цикл IOWR#).
2. Ввод из регистра состояния и проверка готовности устройства (сигнал Busy). Этот шаг зацикливается до получения готовности или до срабатывания программного тайм-аута (минимум 1 цикл IORD#).
3. По получению готовности выводом в регистр управления устанавливается строб данных, а следующим выводом строб снимается. Обычно, чтобы переключить только один бит (строб), регистр управления предварительно считывается, что к двум циклам IOWR# добавляет еще один цикл IORD#.

Видно, что для вывода одного байта требуется 4-5 операций ввода-вывода с регистрами порта (в лучшем случае, когда готовность обнаружена по первому чтению регистра состояния). Отсюда вытекает главный недостаток вывода через стандартный порт – невысокая скорость обмена при значительной загрузке процессора. Порт удается разогнать до скоростей 100-150 Кбайт/с при полной загрузке процессора, что недостаточно для печати на лазерном принтере. Другой недостаток функциональный – сложность использования в качестве порта ввода.

Все перечисленные недостатки помог решить стандарт на параллельный интерфейс IEEE 1284, принятый в 1994 году. Он определяет 5 режимов обмена данными, метод согласования режима, физический и электрический интерфейсы. Согласно IEEE 1284, возможны следующие режимы обмена данными через параллельный порт:

- Режим совместимости (Compatibility Mode) – однонаправленный (вывод) по протоколу Centronics. Этот режим соответствует SPP-порту.
- Полубайтный режим (Nibble Mode) – ввод байта в два цикла (по 4 бита), используя для приема линии состояния. Этот режим обмена подходит для любых адаптеров, поскольку задействует только возможности стандартного порта.

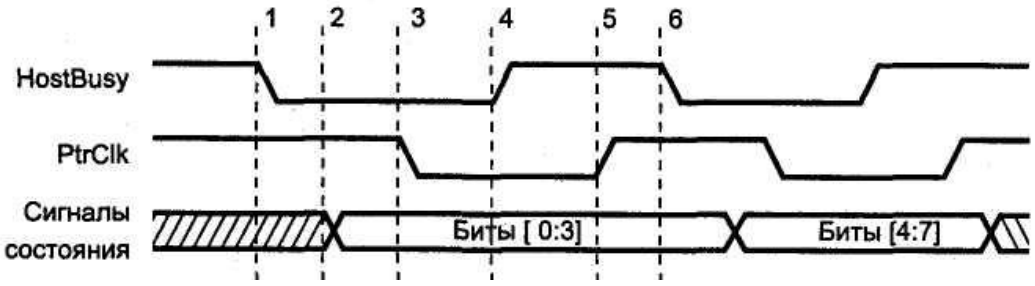


Рис. 98. Передача данных в Nibble Mode.

- Байтный режим (Byte Mode) – ввод байта целиком, используя для приема линии данных. Этот режим работает только на портах, допускающих чтение выходных данных (Bi-Directional или PS/2 Type 1).

Рис. 99. Передача данных в байтовом режиме.

- Режим EPP (EPP Mode) – двунаправленный обмен данными (EPP означает Enhanced Parallel Port). Управляющие сигналы интерфейса генерируются аппаратно во время цикла обращения к порту. Эффективен при работе с устройствами внешней памяти и адаптерами локальных сетей.
- Режим ECP (ECP Mode) – двунаправленный обмен данными с возможностью аппаратного сжатия данных по методу RLE (Run Length Encoding) и использования FIFO-буферов и DMA (ECP означает Extended Capability Port). Управляющие сигналы интерфейса генерируются аппаратно. Эффективен для принтеров и сканеров (здесь может использоваться сжатие) и различных устройств блочного обмена.

В настоящее время стандарт IEEE 1284 не развивается. Окончательная стандартизация параллельного порта совпала с началом внедрения интерфейса USB, который позволяет подключать также и комбинированные аппараты (сканер-принтер-копир) и обеспечивает более высокую скорость печати и надежную работу принтера. Также, альтернативой параллельному интерфейсу является сетевой интерфейс Ethernet.

Протокол EPP

Enhanced Parallel Port – улучшенный параллельный порт, который был разработан компаниями Intel, Xircom и Zenith Data Systems задолго до принятия стандарта IEEE 1284. Этот протокол предназначен для повышения производительности обмена по параллельному порту, впервые был реализован в чипсете Intel 386SL (микросхема 82360) и впоследствии принят множеством компаний как дополнительный протокол параллельного порта. Версии протокола, реализованные до принятия IEEE 1284, отличаются от нынешнего стандарта (см. ниже). Протокол EPP обеспечивает четыре типа циклов обмена: запись данных, чтение данных, запись адреса, чтение адреса.

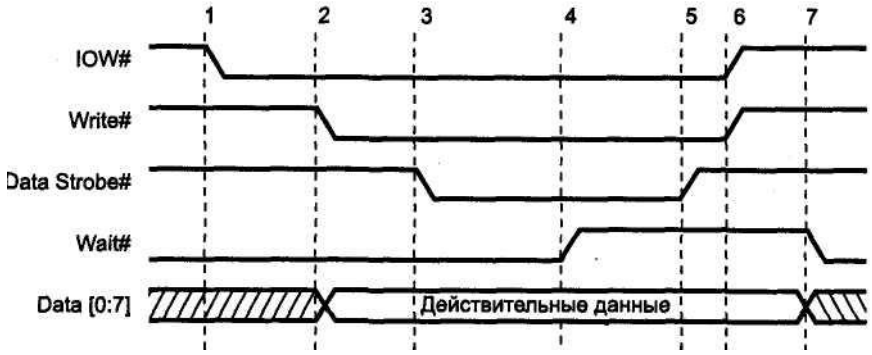


Рис. 100. Цикл записи данных EPP.

Назначение циклов записи и чтения данных очевидно. Адресные циклы используются для передачи адресной, канальной и управляющей информации. Циклы обмена данными отличаются от адресных циклов применяемыми стробирующими сигналами.

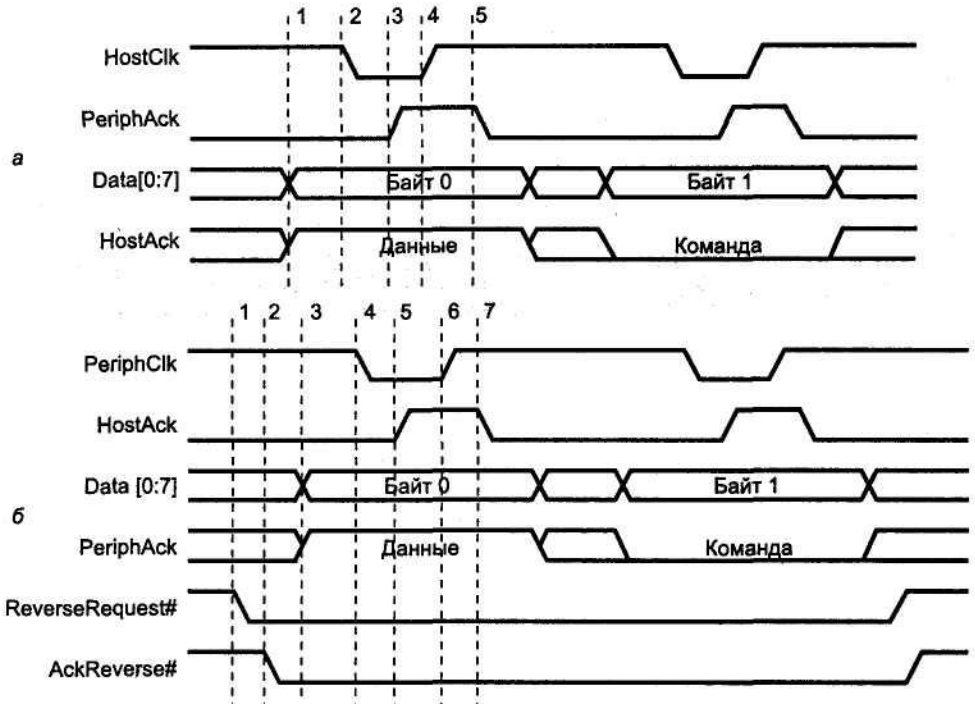


Рис. 101. Обратная передача и приём данных по протоколу ECP.

Протокол ECP

Extended Capability Port – порт с расширенными возможностями, который был предложен Hewlett Packard и Microsoft для связи с ПУ типа принтеров или сканеров. Как и EPP, данный протокол обеспечивает высокопроизводительный двунаправленный обмен данными хоста с ПУ. Протокол ECP в обоих направлениях обеспечивает два типа циклов:

- Циклы записи и чтения данных.
- Командные циклы записи и чтения.

Командные циклы подразделяются на два типа: передача канальных адресов и передача счетчика RLC (Run-Length Count).

В отличие от EPP вместе с протоколом ECP сразу появился стандарт на программную (регистровую) модель его адаптера, изложенный в документе "The IEEE 1284 Extended Capabilities Port Protocol and ISA Interface Standard" компании Microsoft. Этот документ определяет свойства протокола, не заданные стандартом IEEE 1284:

- Компрессия данных хост-адаптером по методу RLE.
- Буферизация FIFO для прямого и обратного каналов.
- Применение DMA и программного ввода-вывода.

3.9.4 Интерфейс SATA

SATA (Serial ATA) – последовательный интерфейс обмена данными с накопителями информации. SATA является развитием параллельного интерфейса ATA (IDE), который после появления SATA был переименован в PATA (Parallel ATA) [37].

Параллельный интерфейс ATA исчерпал свои ресурсы пропускной способности, достигшей 100 МБ/с в режиме UltraDM A Mode 5. Для дальнейшего повышения пропускной способности интерфейса (но, конечно же, не самих устройств хранения, которые имеют гораздо меньшие внутренние скорости обмена с носителем) было принято решение о переходе на последовательный интерфейс. Цель перехода – улучшение и удешевление кабелей и коннекторов, улучшение условий охлаждения устройств внутри системного блока (избавление от широкого шлейфа), обеспечение возможности разработки компактных устройств, облегчение конфигурирования устройств пользователем. Попутно расширяется адресация блоков (предельная емкость адресации ATA – 137 Гбайт). Сейчас уже ведутся работы над новой спецификацией Serial ATA II с большей пропускной способностью и специальными средствами для поддержки сетевых устройств хранения. Приведенная ниже информация относится к версии 1.0.

Интерфейс Serial ATA является хост-центрическим, в нем определяется только взаимодействие хоста с каждым из подключенных устройств, а взаимодействие между ведущим и ведомым устройствами, свойственное традиционному интерфейсу ATA, исключается. Программно хост видит множество устройств, подключенных к контроллеру, как набор каналов ATA, у каждого из которых имеется единственное ведущее устройство. Имеется возможность эмуляции пар устройств (ведущее–ведомое) на одном канале, если такая необходимость возникнет. Программное взаимодействие с устройствами Serial ATA практически совпадает с прежним, набор команд соответствует ATA/ATAPI-5. В то же время аппаратная реализация хост-адаптера Serial ATA сильно отличается от примитивного (в исходном варианте) интерфейса ATA. В параллельном интерфейсе ATA хост-адаптер был простым средством, обеспечивающим программное обращение к регистрам, расположенным в самих подключенных устройствах. В Serial ATA ситуация иная: хост-адаптер

имеет блоки так называемых «теневых» регистров (Shadow Registers), совпадающих по назначению с обычными регистрами устройств ATA. Каждому подключенному устройству соответствует свой набор регистров. Обращения к этим теневым регистрам вызывают процессы взаимодействия хост-адаптера с подключенными устройствами и исполнение команд.

В стандарте рассматривается многоуровневая модель взаимодействия хоста и устройства, где прикладным уровнем является обмен командами, информацией о состоянии и хранимыми данными. На физическом уровне для передачи информации между контроллером и устройством используются две пары проводов. Данные передаются кадрами, транспортный уровень формирует и проверяет корректность информационных структур кадров (Frame Information Structure, FIS). Для облегчения высокоскоростной передачи на канальном уровне данные кодируются по схеме 8B/10B (8 бит данных кодируются 10-битным символом) и скремблируются, после чего по физической линии передаются по простейшему методу NRZ (уровень сигнала соответствует передаваемому биту). Между канальным и прикладным уровнем имеется транспортный уровень, отвечающий за доставку кадров. На каждом уровне имеются свои средства контроля достоверности и целостности.

В первом поколении Serial ATA данные по кабелю передаются со скоростью 1500 Мбит/с, что с учетом кодирования 8B/10B обеспечивает скорость 150 МБ/с (без учета накладных расходов протоколов верхних уровней). В дальнейшем планируется повышать скорость передачи, и в интерфейсе заложена возможность согласования скоростей обмена по каждому интерфейсу в соответствии с возможностями хоста и устройства, а также качеством связи. Хост-адаптер имеет средства управления соединениями, программно эти средства доступны через специальные регистры Serial ATA.

В стандарте предусматривается управление энергорежимом интерфейсов. Каждый интерфейс кроме активного состояния может находиться в состояниях PARTIAL и SLUMBER с пониженным энергопотреблением, для выхода из которых требуется заметное время (10 мс).

Команды, требующие передачи данных, могут исполняться в различных режимах обмена. Обращение в режиме программно управляемого ввода-вывода и традиционный способ обмена по DMA (legacy DMA) выполняется аналогично привычному интерфейсу ATA. Однако внутренний протокол обмена между хост-адаптером и устройствами позволяет передавать между ними разноплановую информацию (структуры FIS определены не только для команд, состояния и собственно хранимых данных). В приложении D к спецификации описывается весьма своеобразный способ обмена по DMA, который предполагается основным (First-party DMA) для устройств Serial ATA. В традиционном контроллере DMA адаптера ATA для каждого канала имеется буфер, в который перед выполнением операции обмена загружают дескрипторы блоков памяти, участвующей в обмене. Теперь же предполагается, что адресная информация, относящаяся к оперативной памяти хост компьютера, будет

доводиться до устройства хранения, подключенного к адаптеру Serial ATA. Эта информация из устройства хранения при исполнении команд обмена выгружается в контроллер DMA хост-адаптера и используется им для формирования адреса текущей передачи. Мотивы и полезность этого нововведения не совсем понятны; расплатой за некоторое упрощение хост-адаптера (особенно многоканального) является усложнение протокола и расширение функций, выполняемых устройством хранения. Все-таки более привычно традиционное разделение функций, при котором задача устройств внешней памяти – хранить данные, не интересуясь тем, в каком месте оперативной памяти компьютера они должны находиться при операциях обмена.

3.9.4.1 Физический интерфейс Serial ATA

Последовательный интерфейс ATA, как и его параллельный предшественник, предназначен для подключений устройств внутри компьютера. Длина кабелей не превышает 1 м, при этом все соединения радиальные, каждое устройство подключается к хост-адаптеру своим кабелем. В стандарте предусматривается и непосредственное подключение устройств к разъемам кросс-платы с возможностью горячей замены. Стандарт определяет новый однорядный двухсегментный разъем с механическими ключами, препятствующими ошибочному подключению. Сигнальный сегмент имеет 7 контактов (S1-S7), питающий – 15 (P1-P15); все контакты расположены в один ряд с шагом 1,27 мм. Назначение контактов приведено в таблице. Малые размеры разъема (полная длина около 36 мм) и малое количество цепей облегчают компоновку системных плат и карт расширения. Питающий сегмент может отсутствовать (устройство может получать питание и от обычного 4-контактного разъема ATA). Вид разъемов приведен ниже. Для обеспечения горячего подключения контакты разъемов имеют разную длину, в первую очередь соединяются контакты «земли» P4 и P12, затем остальные «земли» и контакты предзаряда конденсаторов в цепях питания P3, P7 и P13 (для уменьшения броска потребляемого тока), после чего соединяются основные питающие контакты и сигнальные цепи.

Таблица 9. Разъем Serial ATA.

Контакт	Цепь	Назначение
51	GND	Экран
52	A+	Дифференциальная пара сигналов A
53	A-	Дифференциальная пара сигналов A
54	GND	Экран
55	B-	Дифференциальная пара сигналов B
56	B+	Дифференциальная пара сигналов B
57	GND	Экран

Ключи и свободное пространство		
P1	V33	Питание 3,3 В
P2	V33	Питание 3,3 В
P3	V33	Питание 3,3 В, предзаряд
P4	GND	Общий
P5	GND	Общий
P6	GND	Общий
P7	V5	Питание 5 В, предзаряд
P8	V5	Питание 5 В
P9	V5	Питание 5 В
P10	GND	Общий
P11	Резерв	
P12	GND	Общий
P13	V12	Питание 12В, предзаряд
P14	V12	Питание 12 В
P15	V12	Питание 12 В

Таблица 10. Сравнительная таблица интерфейсов из семейства ATA.

Наименование	Макс. пропускная способность (Мбит/с)	Скорость передачи (Мбайт/с)	Макс. длина кабеля	Количество подключаемых устройств
eSATA	3,000	300	2 with eSATA HBA (1 with passive adapter)	1 (15 с расширителем портов)
eSATAp				
SATA 600	6,000	600	1	
SATA 300	3,000	300		
SATA 150	1,500	150		1 на канал
PATA 133	1,064	133.5	0.46 (18 in)	2

3.10 Контроллерные сети

3.10.1Интерфейс RS-485

RS-485 (Recommended Standard 485, Electronics Industries Association 485, EIA-485) – стандарт передачи данных по двухпроводному полудуплексному многоточечному последовательному каналу связи.

Стандарт RS-485 совместно разработан двумя ассоциациями: Ассоциацией электронной промышленности (EIA – Electronics Industries Association) и Ассоциацией промышленности средств связи (TIA – Telecommunications

Industry Association). Ранее EIA маркировала все свои стандарты префиксом «RS» (Recommended Standard – Рекомендованный стандарт). Многие инженеры продолжают использовать это обозначение, однако EIA/TIA официально заменил «RS» на «EIA/TIA» с целью облегчить идентификацию происхождения своих стандартов. На сегодняшний день различные расширения стандарта RS-485 охватывают широкое разнообразие приложений. Этот стандарт стал основой для создания целого семейства промышленных сетей, широко используемых в промышленной автоматизации.

В стандарте RS-485 для передачи и приёма данных часто используется единственная витая пара проводов. Передача данных осуществляется с помощью дифференциальных сигналов. Разница напряжений между проводниками одной полярности означает логическую единицу, разница другой полярности — ноль.

RS-485 имеет следующие особенности:

- Возможность объединения несимметричных и симметричных цепей.
- Параметры качества сигнала, уровень искажений (%).
- Методы доступа к линии связи.
- Протокол обмена.
- Аппаратную конфигурацию (среда обмена, кабель).
- Типы соединителей, разъёмов, колодок, нумерацию контактов.
- Качество источника питания (стабилизация, пульсация, допуск).
- Отражения в длинных линиях.

Электрические и временные характеристики интерфейса RS-485:

- 32 приёмопередатчика при многоточечной конфигурации сети (на одном сегменте, максимальная длина линии в пределах одного сегмента сети 1200м).
- Только один передатчик активный.
- Максимальное количество узлов в сети — 250 с учётом магистральных усилителей.

Соответствие скорости обмена длине линии связи (зависимость экспоненциальная):

- 62,5 кбит/с 1200 м (одна витая пара).
- 375 кбит/с 300 м (одна витая пара).
- 2400 кбит/с 100 м (две витые пары).
- 10000 кбит/с 10 м.

Примечание: скорости обмена 62,5 кбит/с, 375 кбит/с, 2400 кбит/с оговорены стандартом RS-485. На скоростях обмена свыше 500 кбит/с рекомендуется использовать экранированные витые пары.

Тип приёмопередатчиков – дифференциальный, потенциальный. Изменение входных и выходных напряжений на линиях A и B: Ua (Ub) от −7В до +12В (+7В).

3.10.1.1 Согласование и конфигурация линии связи

При больших расстояниях между устройствами, связанными по витой паре, и высоких скоростях передачи начинают проявляться так называемые эффекты длинных линий. Причина этому – конечность скорости распространения электромагнитных волн в проводниках. Скорость эта существенно меньше скорости света в вакууме и составляет немногим больше 200 мм/нс. Электрический сигнал имеет также свойство отражаться от открытых концов линии передачи и ее ответвлений. Грубая аналогия – желоб, наполненный водой. Волна, созданная в одном конце, идет по желобу и, отразившись от стенки в конце, идет обратно, отражается опять и т.д., пока не затухнет. Для коротких линий и малых скоростей передачи этот процесс происходит так быстро, что остается незамеченным. Однако время реакции приемников десятки-сотни нс. В таком масштабе времени несколько десятков метров электрический сигнал проходит отнюдь не мгновенно. И если расстояние достаточно большое, фронт сигнала, отразившегося в конце линии и вернувшегося обратно, может исказить текущий или следующий сигнал. В таких случаях нужно каким-то образом подавлять эффект отражения [27].

У любой линии связи есть такой параметр, как волновое сопротивление Zв. Оно зависит от характеристик используемого кабеля, но не от длины. Для обычно применяемых в линиях связи витых пар Zв=120 Ом. Если на удаленном конце линии между проводниками витой пары включить резистор с номиналом, равным волновому сопротивлению линии, то электромагнитная волна, дошедшая до «тупика», поглощается на таком резисторе. Отсюда его названия – согласующий резистор или «терминатор».

Большой минус согласования на резисторах – повышенное потребление тока от передатчика, ведь в линию включается низкоомная нагрузка. Поэтому рекомендуется включать передатчик только на время отправки посылки. Есть способы уменьшить потребление тока, например, посредством включения последовательно с согласующим резистором конденсатора для развязки по постоянному току. Однако такой способ имеет свои недостатки. Для коротких линий (несколько десятков метров) и низких скоростей (меньше 38400 бод) согласование можно вообще не делать.

Эффект отражения и необходимость правильного согласования накладывают ограничения на конфигурацию линии связи.

Линия связи должна представлять собой один кабель витой пары. К этому кабелю присоединяются все приемники и передатчики. Расстояние от линии до

микросхем интерфейса RS-485 должно быть как можно короче, так как длинные ответвления вносят рассогласование и вызывают отражения.

В оба наиболее удаленных конца кабеля (Zв=120 Ом) включают согласующие резисторы Rt по 120 Ом (0,25 Вт). Если в системе только один передатчик и он находится в конце линии, то достаточно одного согласующего резистора на противоположном конце линии [27].

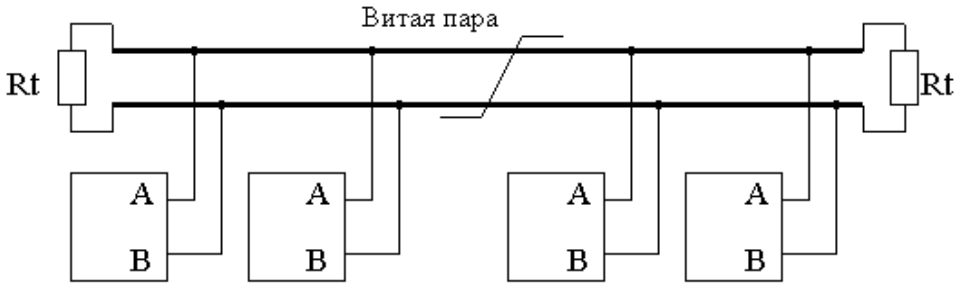


Рис. 102. Согласование линии связи.

3.10.1.2 Защитное смещение

Как уже упоминалось, приемники большинства микросхем RS-485 имеют пороговый диапазон распознавания сигнала на входах A-B - ±200мВ. Если |Uab| меньше порогового (около 0), то на выходе приемника RO могут быть произвольные логические уровни из-за несинфазной помехи. Такое может случиться либо при отсоединении приемника от линии, либо при отсутствии в линии активных передатчиков, когда никто не задает уровень. Чтобы в этих ситуациях избежать выдачи ошибочных сигналов на приемник UART, необходимо на входах A-B гарантировать разность потенциалов U_{ab} > +200мВ. Это смещение при отсутствии входных сигналов обеспечивает на выходе приемника логическую «1», поддерживая таким образом уровень стопового бита [27].

Добиться этого просто: прямой вход (A) следует подтянуть к питанию, а инверсный (B) к «земле».

Значения сопротивлений для резисторов защитного смещения ($R_{зс}$) нетрудно рассчитать по делителю. Необходимо обеспечить U_{ab} > 200мВ. Напряжение питания 5В. Сопротивление среднего плеча 120Ом/120Ом/12КОм, на каждый приемник примерно 57Ом (для 10 приемников). Таким образом, выходит примерно по 650Ом на каждый из двух $R_{зс}$. Для смещения с запасом сопротивление $R_{зс}$ должно быть меньше 650Ом. Традиционно ставят 560Ом.

Обратите внимание: в расчете номинала $R_{зс}$ учитывается нагрузка. Если на линии висит много приемников, то номинал $R_{зс}$ должен быть меньше. В длинных линиях передачи необходимо также учитывать сопротивление витой пары, которое может «съедать» часть смещающей разности потенциалов для удаленных от места подтяжки устройств. Для длинной линии лучше ставить два комплекта подтягивающих резисторов в оба удаленных конца рядом с терминаторами.

Функция безотказности

Многие производители приемопередатчиков заявляют о функции безотказности (failsafe) своих изделий, заключающейся во встроенном смещении. Следует различать два вида такой защиты [27]:

1. Безотказности в открытых цепях (Open circuit failsafe). В таких приемопередатчиках применяются встроенные подтягивающие резисторы. Эти резисторы, как правило, высокоомные, чтобы уменьшить потребление тока. Из-за этого необходимое смещение обеспечивается только для открытых (ненагруженных) дифференциальных входов. В самом деле, если приемник отключен от линии или она не нагружена, тогда в среднем плече делителя остается только большое входное сопротивление, на котором и падает необходимая разность потенциалов. Однако если приемопередатчик нагрузить на линию с двумя согласующими резисторами по 120Ом, то в среднем плече делителя оказывается меньше 60Ом, на которых, по сравнению с высокоомными подтяжками, ничего существенного не падает. Поэтому если в нагруженной линии нет активных передатчиков, то встроенные резисторы не обеспечивают достаточное смещение. В этом случае остается необходимость устанавливать внешние резисторы защитного смещения, как это было описано выше.
2. Истинная безотказность (True failsafe). В этих устройствах смещены сами пороги распознавания сигнала. Например: -50 / -200 мВ вместо стандартных порогов ±200 мВ, т.е. при U_{ab} > -50мВ на выходе приемника RO будет логическая «1», а при U_{ab} < -200 – на RO будет «0». Таким образом, и в разомкнутой, и в пассивной линиях при разности потенциалов U_{ab}, близкой к 0, приемник выдаст «1». Для таких приемопередатчиков внешнее защитное смещение не требуется. Тем не менее, для лучшей помехозащищенности все-таки стоит дополнительно немного подтягивать линию.

Сразу виден недостаток внешнего защитного смещения: через делитель постоянно будет протекать ток, что может быть недопустимо в системах малого потребления. В таком случае можно сделать следующее [27]:

1. Уменьшить потребление тока, увеличив сопротивления $R_{зс}$. Хотя производители приемопередатчиков и пишут о пороге распознавания в 200мВ, на практике вполне хватает 100мВ и даже меньше. Таким образом, можно сразу увеличить сопротивления $R_{зс}$ раза в два-три. Помехозащищенность при этом несколько снижается, но во многих случаях это некритично.
2. Использовать true failsafe приемопередатчики со смещенными порогами распознавания. Например, у микросхем MAX3080 и MAX3471 пороги -50мВ / -200мВ, что гарантирует единичный уровень на выходе приемника при отсутствии смещения (U_{ab}=0). Тогда внешние резисторы

защитного смещения можно убрать или значительно увеличить их сопротивление.

3. Не применять без необходимости согласование на резисторах. Если линия не будет нагружена на 2 резистора по 120Ом, то для обеспечения защитного смещения хватит «подтяжек» в несколько КОм в зависимости от числа приемников на линии.

3.10.1.3 Исключение приема при передаче в полудуплексном режиме

При работе с полудуплексным интерфейсом RS-485 (прием и передача по одной паре проводов с разделением по времени) можно забыть, что UART контроллера полнодуплексный, т.е. принимает и передает независимо и одновременно.

Обычно во время работы приемопередатчика RS-485 на передачу выход приемника RO переводится в третье состояние и ножка RX контроллера (приемник UART) «повисает в воздухе». В результате во время передачи на приемнике UART вместо уровня стопового бита («1») окажется неизвестно что и любая помеха будет принята за входной сигнал. Поэтому нужно либо на время передачи отключать приемник UART (через управляющий регистр), либо подтягивать RX к единице. У некоторых микроконтроллеров это можно сделать программно – активировать встроенные подтяжки портов [27].

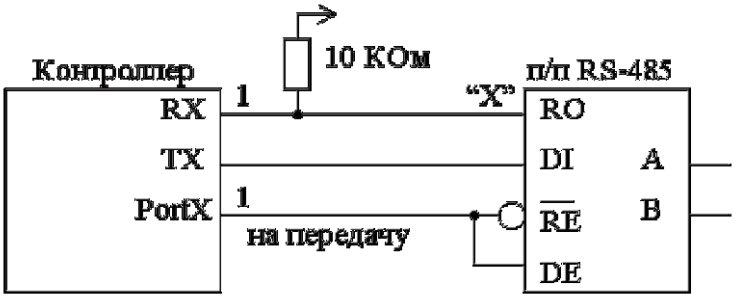


Рис. 103. Схема подключения приемопередатчика RS-485 к микроконтроллеру.

3.10.2Интерфейс 1-Wire

Сеть 1-Wire или MicroLan (MicroLan – торговая марка фирмы Dallas Semiconductor) предназначена для обеспечения питания и передачи данных с небольшой скоростью (обычно 16,4 Кбит/с) по одной сигнальной линии и одной возвратной (общий провод). Максимальное расстояние передачи – 300м. Тип линии связи – моноканал, тип передачи – полудуплекс. К интерфейсу возможно подключение одного мастера сети и нескольких подчиненных узлов с питанием от самой сети. Основное назначение сети – обеспечение связи микроконтроллера с различными датчиками (например, термодатчиками), устройствами Touch Memory (iButton), маркировка устройств уникальным идентификатором (каждая микросхема поддерживающая интерфейс 1-Wire имеет уникальный 64-битный идентификатор) и т.д. [99].

Последовательная передача осуществляется внутри дискретно определённых временных интервалов, называемых тайм-слотами. Master-

устройство инициирует передачу с помощью посылки командного слова на slave-устройство.

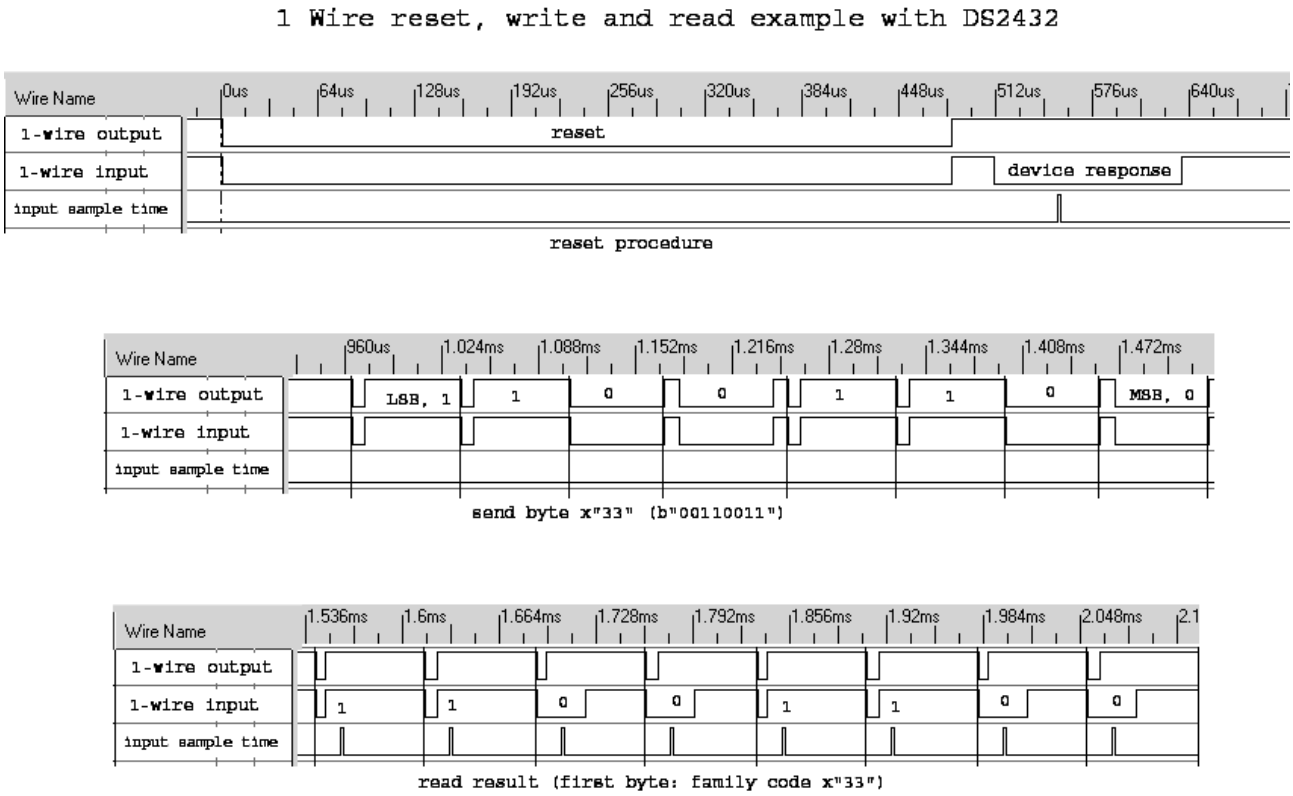


Рис. 104. Примеры временных диаграмм, иллюстрирующие работу интерфейса 1-Wire: сброс устройства (верхняя диаграмма), передача данных (средняя диаграмма), приём данных от устройства (нижняя диаграмма).

Команды и данные посылаются бит за битом, причём вначале передаётся наименее значащий бит LSB (Least Significant Bit). Синхронизация master и slave происходит по спадающему срезу сигнала, когда master замыкает стоком выходного транзистора порта линию данных на провод земли. Через определённое время после среза сигнала происходит анализ (выборка) состояния данных на линии (логический «0» или логическая «1») для получения одного бита информации. В зависимости от направления передачи информации в данный момент эту выборку делает либо устройство master, либо устройство slave. Этот метод обмена информацией называют передачей данных в тайм-слотах. Каждый тайм-слот отсчитывается независимо от другого, и в обмене данными могут иметь место паузы без возникновения ошибок.

Почти сразу после присоединения к считывающему устройству (через несколько микросекунд) slave-устройство выдаёт на линию импульс низкого уровня, чтобы сказать устройству master, что оно на линии и ожидаёт получения команды. Этот сигнал называется presence pulse (импульс присутствия, далее просто presence). Master может также давать запрос на slave устройство с целью получения presence, путём выдачи специального импульса, называемого импульсом сброса (reset pulse, далее просто reset). Если slave принял reset или если он был отсоединён от считывающего устройства, он будет анализировать

линию данных, и как только линия снова достигнет высокого уровня, slave сгенерирует presence.

3.10.3Интерфейс I^2C

В бытовой технике, телекоммуникационном оборудовании и промышленной электронике часто встречаются похожие решения в, казалось бы, никак не связанных изделиях. Например, практически каждая система включает в себя:

- Некоторый «умный» узел управления, обычно однокристальная микроЭВМ;
- Узлы общего назначения, такие как буферы ЖКИ, порты ввода-вывода, RAM, E^2PROM или преобразователи данных.
- Специфические узлы, такие как схемы цифровой настройки и обработки сигнала для радио- и видеосистем, или генераторы тонального набора для телефонии.

Для того чтобы использовать эти общие решения с выгодой для конструкторов и производителей (технологов), а также увеличить эффективность аппаратуры и упростить схемотехнические решения, компания Philips в 1980 году разработала простую двунаправленную двухпроводную шину для эффективного «межмикросхемного» (inter-IC) управления. Шина так и называется – Inter-Integrated Circuit, или IIC (I^2C) шина. В настоящее время ассортимент продукции Philips включает более 150 КМОП и биполярных I^2C-совместимых устройств, функционально предназначенных для работы во всех трех вышеперечисленных категориях электронного оборудования. Все I^2C-совместимые устройства имеют встроенный интерфейс, который позволяет им связываться друг с другом по шине I^2C. Это конструкторское решение разрешает множество проблем сопряжения различных устройств, которые обычно возникают при разработке цифровых систем [12, 13, 19, 23, 24, 46, 55, 56, 81].

Основной режим работы шины I^2C – 100 Кбит/с; 10 Кбит/с в режиме работы с пониженной скоростью. Заметим, что стандарт допускает тактирование с частотой вплоть до нулевой. Для адресации I^2C-устройств используется 7 бит.

После пересмотра стандарта в 1992 году становится возможным подключение ещё большего количества устройств на одну шину (за счёт возможности 10-битной адресации), а также большую скорость до 400 кбит/с в скоростном режиме. Соответственно, доступное количество свободных узлов выросло до 1008. Максимально допустимое количество микросхем, подсоединенных к одной шине, ограничивается максимальной емкостью шины в 400 пФ.

Версия стандарта 2.0, выпущенная в 1998 году представила высокоскоростной режим работы со скоростью до 3.4 Мбит/с с пониженным

энергопотреблением. Последняя версия 2.1 2000 года включила лишь незначительные доработки.

1 октября 2006 года были отменены лицензионные отчисления за использование протокола I²C. Однако отчисления сохраняются для выделения эксклюзивного подчинённого адреса на шине I²C.

Список возможных применений I²C:

- Доступ к модулям памяти (RAM, E2PROM, FLASH и др.).
- Доступ к низкоскоростным ЦАП/АЦП.
- Работа с часами реального времени (RTC).
- Регулировка контрастности, насыщенности и цветового баланса мониторов.
- Управление интеллектуальными звукоизлучателями (динамиками).
- Управление ЖКИ, в том числе в мобильных телефонах.
- Чтение информации с датчиков мониторинга и диагностики оборудования, например, термостат центрального процессора или датчик скорости вращения вентилятора охлаждения процессора.
- Информационный обмен между микроконтроллерами.

3.10.3.1 Концепция шины I²C

I²C использует две двунаправленные линии с открытым стоком: последовательная линия данных (SDA, Serial DAta) и последовательная линия тактирования (SCL, Serial CLock), обе нагруженные резисторами. Максимальное напряжение +5В, часто используется +3,3В, однако допускаются и другие напряжения (не менее +2В). Шина I²C поддерживает любую технологию изготовления микросхем (НМОП, КМОП, биполярную).

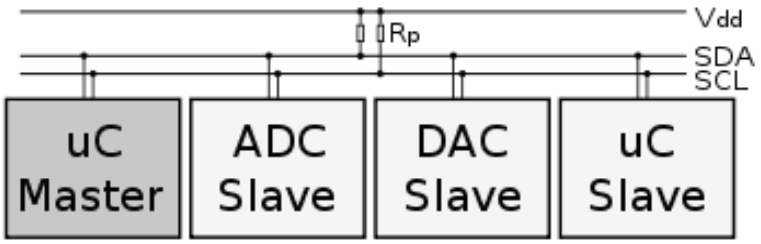


Рис. 105. Пример соединения устройств по шине I2C: один ведущий - микроконтроллер, три ведомых устройства - АЦП, ЦАП, МК.

Каждое устройство распознается по уникальному адресу, будь то микроконтроллер, ЖКИ-буфер, память или интерфейс клавиатуры, и может работать как передатчик или приёмник, в зависимости от назначения устройства. Обычно ЖКИ-буфер – только приёмник, а память может как принимать, так и передавать данные. Кроме того, устройства могут быть классифицированы как ведущие и ведомые при передаче данных. Ведущий –

это устройство, которое инициирует передачу данных и вырабатывает сигналы синхронизации. При этом любое адресуемое устройство считается ведомым по отношению к ведущему. Классическая адресация включает 7-битное адресное пространство с 16 зарезервированными адресами. Это означает до 112 свободных адресов для подключения периферии на одну шину.

Таблица 11. Термины, используемые в спецификации I2C.

Термин (англ.)	Термин (рус.)	Описание
Transmitter	Передатчик	Устройство, посылающее данные в шину
Receiver	Приемник	Устройство, принимающее с шины
Master	Ведущий	Начинает пересылку данных, вырабатывает синхроимпульсы, заканчивает пересылку данных
Slave	Ведомый	Устройство, адресуемое ведущим
Multi-master	–	Несколько ведущих могут пытаться захватить шину одновременно, без нарушения передаваемой информации
Arbitration	Арбитраж	Процедура, обеспечивающая Multi-master
Synchronization	Синхр.	Процедура синхронизации двух устройств

Возможность подключения более одного микроконтроллера к шине означает, что более чем один ведущий может попытаться начать пересылку в один и тот же момент времени. Для устранения хаоса, который может возникнуть в данном случае, разработана процедура арбитража. Эта процедура основана на том, что все I²C-устройства подключаются к шине по правилу монтажного И.

Генерация синхросигнала – это всегда обязанность ведущего: каждый ведущий генерирует свой собственный сигнал синхронизации при пересылке данных по шине. Сигнал синхронизации может быть изменен, только если он «вытягивается» медленным ведомым устройством (путем удержания линии в низком состоянии), или другим ведущим в случае столкновения.

3.10.3.2 Реализация монтажного И и монтажного ИЛИ

Схемотехника и функционирование выхода с открытым коллектором (ОК, для КМОП-схем правильно говорить о выходе с ОС, но часто данный тип выхода по аналогии с ТТЛ-схемами называют ОК) аналогична однотактному выходу с той разницей, что внутренний (на кристалле микросхемы) нагрузочный резистор отсутствует, а подключается внешний резистор [55]. Схема представлена на рисунке ниже, условное обозначение выхода данного типа – см. рисунок ниже, б.

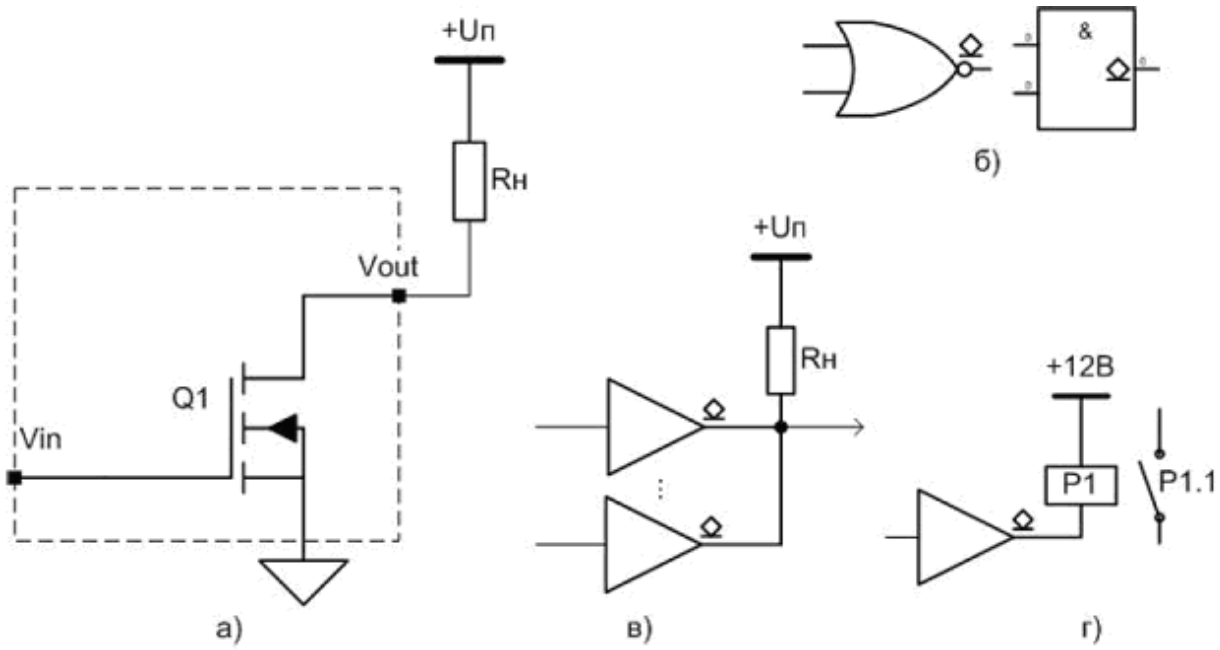


Рис. 106. Выход с открытым коллектором (стоком).

Особенностями выхода ОК являются:

1. Возможность подключать несколько выходов к одной шине (рис. выше, в). Эта схема включения может быть необходима для выполнения одной из двух функций:

   - **Реализация «монтажной логики»,** в частности операции монтажное И для положительной логики: если хотя бы на одном из выходов будет установлен НИЗКИЙ уровень, это значит, что общая выходная шина будет замкнута на 0В через открытый транзистор этого выхода и, вне зависимости от уровней на остальных выходах, на общей шине будет установлен НИЗКИЙ уровень, т.е. логический «0». Таким образом будет реализована операция И без какого-либо специального элемента. Если использовать отрицательную логику, т.е. ИСТИНОЙ (логической «1») считать НИЗКИЙ уровень, то данная схема реализует функцию монтажного ИЛИ. Именно в такой интерпретации чаще всего используется данная схема включения. Например, сигналы прерывания от нескольких источников с выходом ОК объединены и подключены к одному входу запроса прерывания у микропроцессора. Если хоть один из них перейдет в активное НИЗКОЕ состояние, то выработается запрос прерывания и далее процессор путем опроса найдет источник этого прерывания. Другой пример – сканирование матричной клавиатуры.

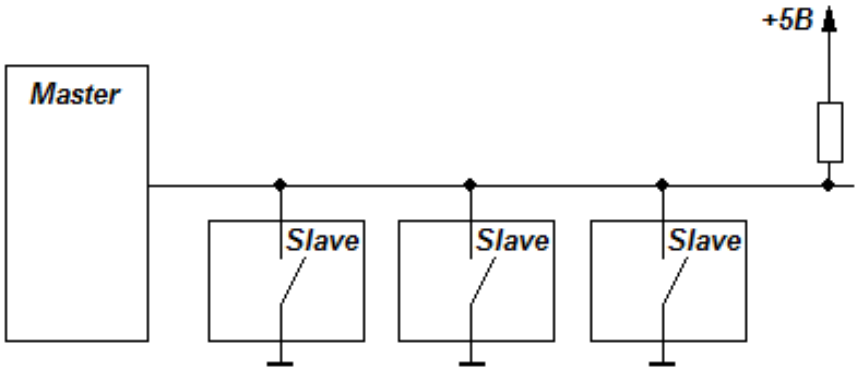


Рис. 107. Подключение устройств по принципу монтажного И.

Простейший пример – сигнал готовности. Есть несколько подчиненных устройств одного типа, каждое из которых работает по своему алгоритму и, например, требует начальную инициализацию. Чтобы главный понял, что все остальные устройства закончили инициализацию и подготовились, он «слушает» линию связи. В качестве сигнала готовности подчиненное устройство отпускает линию. Когда каждое устройство пройдет инициализацию, то линия примет высокий уровень и это будет сигналом полной готовности всех устройств.

- **Подключение нескольких источников к общей шине.** В данном случае активным является только один источник, а все неактивные обязаны установить на выходе ВЫСОКИЙ уровень. Далее, так как реализуется монтажное И (см. выше), то на шине присутствует только сигнал от одного – активного – источника. При этом, даже если одновременно активизируются два или более источников, выход не перегрузится и не «перегорит». Функция общей шины широко используется для периферийных интерфейсов микропроцессорных систем, например, в интерфейсах I2C, 1-Wire.

2. Возможность подключать нагрузку к напряжению большему или меньшему, чем напряжение питания микросхемы. Например, на рисунке (рис. выше, г) показан способ подключения обмотки реле с напряжением питания 12В. Аналогично можно подключать другие нагрузки: лампочки, линейки светодиодов, звукоизлучатели, двигатели постоянного тока и т.п. Использовать как большее, так и меньшее напряжение питания также может быть удобно для адаптеров логических уровней схем с различным напряжением питания. Например, при конверсии сигналов с ВЫСОКИМ уровнем, равным 5В, в сигналы с ВЫСОКИМ уровнем, равным 3,3В, и наоборот.

3.10.3.3 Принцип работы шины I²C

Вследствие различных технологий микросхем (КМОП, НМОП, биполярная), которые могут быть подключены к шине, уровни логического «0» (НИЗКИЙ) и логической «1» (ВЫСОКИЙ) не фиксированы и зависят от соответствующего уровня Vdd. Один синхроимпульс генерируется на каждый пересылаемый бит.

Данные на линии SDA должны быть стабильными в течение ВЫСОКОГО периода синхроимпульса. ВЫСОКОЕ или НИЗКОЕ состояние линии данных должно меняться, только если линия синхронизации в состоянии НИЗКОЕ.

Рис. 108. Пересылка бита по шине I²C.

Данные по линии SDA передаются байтами, при этом каждый байт должен оканчиваться битом подтверждения. Количество байт, передаваемых за один сеанс связи, не ограничено. Данные передаются, начиная со старшего бита. Если приёмник не может принять еще один целый байт, пока он не выполнит какую-либо другую функцию (например, обслужит внутреннее прерывание), он может удерживать линию SCL в НИЗКОМ состоянии, переводя передатчик в состояние ожидания. Пересылка данных продолжается, когда приёмник будет готов к следующему байту и отпустит линию SCL (опять срабатывает правило монтажного И).

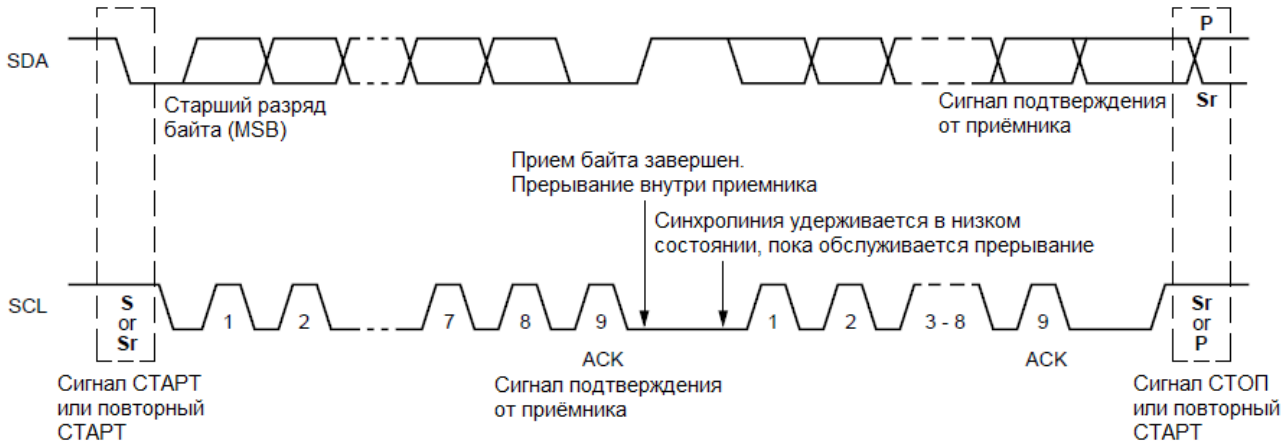


Рис. 109. Пересылка данных по шине I²C.

3.10.3.4 Сигналы СТАРТ и СТОП

Процедура обмена данными по шине I²C начинается с того, что ведущий формирует состояние СТАРТ – ведущий генерирует переход сигнала линии SDA из ВЫСОКОГО состояния в НИЗКОЕ при ВЫСОКОМ уровне на линии SCL [13, 81]. Этот переход воспринимается всеми устройствами, подключенными к шине как признак начала процедуры обмена. Процедура обмена завершается тем, что ведущий формирует состояние СТОП – переход состояния линии SDA из НИЗКОГО состояния в ВЫСОКОЕ при ВЫСОКОМ состоянии линии SCL. Состояния СТАРТ и СТОП всегда вырабатываются ведущим. Считается, что шина занята после фиксации состояния СТАРТ. Шина считается освободившейся через некоторое время после фиксации состояния СТОП.

Определение сигналов СТАРТ и СТОП устройствами, подключенными к шине, достаточно легко, если в них встроены необходимые цепи. Однако микроконтроллеры без таковых цепей должны осуществлять считывание значения линии SDA как минимум дважды за период синхронизации для того, чтобы определить переход состояния.

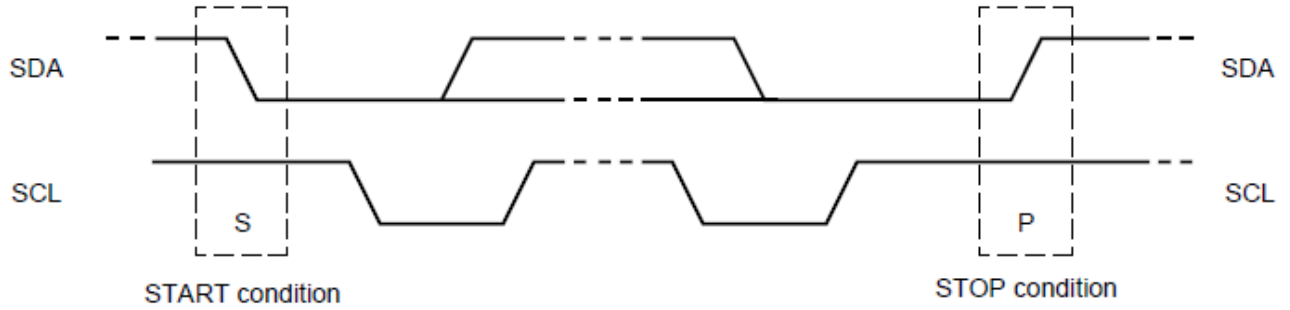


Рис. 110. СТАРТ и СТОП состояния.

3.10.3.5 Подтверждение

Подтверждение при передаче данных обязательно, кроме случаев окончания передачи ведомой стороной. Соответствующий импульс синхронизации генерируется ведущим [13, 81]. Передатчик отпускает (ВЫСОКОЕ) линию SDA в течение синхроимпульса подтверждения. Приёмник должен удерживать линию SDA в течение ВЫСОКОГО состояния синхроимпульса подтверждения в стабильно НИЗКОМ состоянии. Конечно, время установки и удержания также должны быть приняты во внимание (электрические и временные параметры).

Таким образом, передача 8 бит данных от передатчика к приемнику завершается дополнительным циклом (формированием 9-го тактового импульса линии SCL), при котором приемник выставляет НИЗКИЙ уровень сигнала на линии SDA, как признак успешного приема байта.

В том случае, когда ведомый приёмник не может подтвердить свой адрес (например, когда он выполняет в данный момент какие-либо функции реального времени), линия данных должна быть оставлена в ВЫСОКОМ

состоянии. После этого ведущий может выдать сигнал СТОП для прерывания пересылки данных. Если в пересылке участвует ведущий приёмник, то он должен сообщить об окончании передачи ведомому передатчику путем не подтверждения последнего байта. Ведомый передатчик должен освободить линию данных для того, чтобы позволить ведущему выдать сигнал СТОП или повторить сигнал СТАРТ.

3.10.3.6 Синхронизация

При передаче посылок по шине I²C каждый ведущий генерирует свой синхросигнал на линии SCL. Данные действительны только во время ВЫСОКОГО состояния синхроимпульса [13, 81].

Синхронизация выполняется с использованием подключения к линии SCL по правилу монтажного И. Это означает, что ведущий не имеет монопольного права на управление переходом линии SCL из НИЗКОГО состояния в ВЫСОКОЕ. В том случае, когда ведомому необходимо дополнительное время на обработку принятого бита, он имеет возможность удерживать линию SCL в НИЗКОМ состоянии до момента готовности к приему следующего бита. Таким образом, линия SCL будет находиться в НИЗКОМ состоянии на протяжении самого длинного НИЗКОГО периода синхросигналов.

Устройства с более коротким НИЗКИМ периодом будут входить в состояние ожидания на время, пока не кончится длинный период. Когда у всех задействованных устройств кончится НИЗКИЙ период синхросигнала, линия SCL перейдет в ВЫСОКОЕ состояние. Все устройства начнут проходить ВЫСОКИЙ период своих синхросигналов. Первое устройство, у которого кончится этот период, снова установит линию SCL в НИЗКОЕ состояние. Таким образом, НИЗКИЙ период синхролинии SCL определяется наидлиннейшим периодом синхронизации из всех задействованных устройств, а ВЫСОКИЙ период определяется самым коротким периодом синхронизации устройств.

Механизм синхронизации может быть использован приемниками как средство управления пересылкой данных на байтовом и битовом уровнях.

На уровне байта, если устройство может принимать байты данных с большой скоростью, но требует определенного времени для сохранения принятого байта или подготовки к приему следующего, то оно может удерживать линию SCL в НИЗКОМ состоянии после приема и подтверждения байта, переводя таким образом передатчик в состояние ожидания.

На уровне битов устройство, такое как микроконтроллер без встроенных аппаратных цепей I²C или с ограниченными цепями, может замедлить частоту синхроимпульсов путем продления их НИЗКОГО периода. Таким образом скорость передачи любого ведущего адаптируется к скорости медленного устройства.

3.10.3.7 Форматы обмена данными по шине I²C (7-битный адрес)

После сигнала СТАРТ посылается адрес ведомого. После 7 бит адреса следует бит направления данных (R/W), «0» означает передачу (запись), а «1» – прием (чтение). Пересылка данных всегда заканчивается сигналом СТОП, генерируемым ведущим [13, 46, 81]. Однако, если ведущий желает оставаться на шине дальше, он должен выдать повторный сигнал СТАРТ и затем адрес следующего устройства. При таком формате посылки возможны различные комбинации чтения/записи.

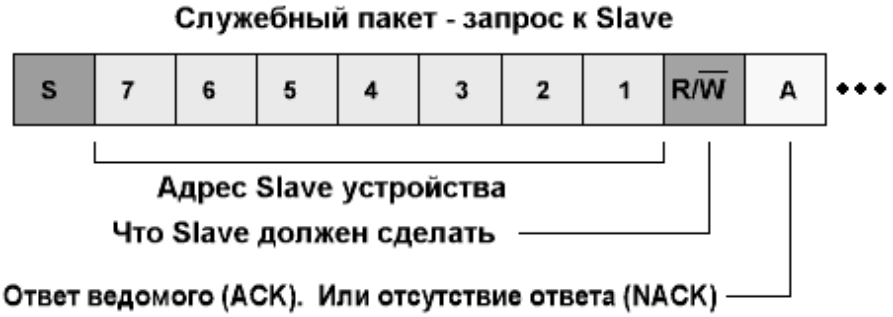


Рис. 111. Первый байт после СТАРТ-состояния (адресный байт).

Возможные форматы:

1. Ведущий-передатчик передает данные ведомому-приёмнику. Направление пересылки данных не изменяется (запись W = 0) [1346].

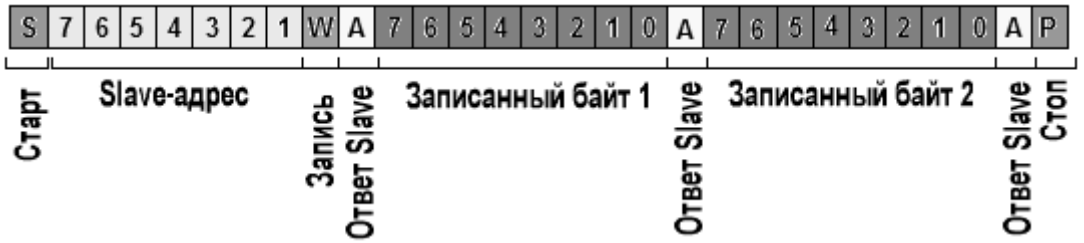


Рис. 112. Запись данных.

2. Ведущий читает ведомого немедленно после пересылки первого байта. В момент первого подтверждения ведущий-передатчик становится ведущим-приёмником, и ведомый-приёмник становится ведомым-передатчиком. Подтверждение тем не менее генерируется ведущим. Сигнал СТОП генерируется ведущим. В таком формате посылки есть тонкость: при приеме последнего байта надо дать ведомому понять, что больше ведущий читать не будет и отослать NACK на последнем байте. Если же отослать ACK, то после СТОП-сигнала ведущий не отпустит линию (объясняется работой автомата в контроллере I²C). Так что прием двух байтов будет выглядеть так (чтение R = 1):

Рис. 113. Чтение данных.

3. Комбинированный формат. При изменении направления пересылки данных повторяется сигнал СТАРТ и адрес ведомого, но бит направления данных инвертируется. Если ведущий-приёмник посылает повторный сигнал СТАРТ, он обязан предварительно послать сигнал неподтверждения [13, 46, 81].

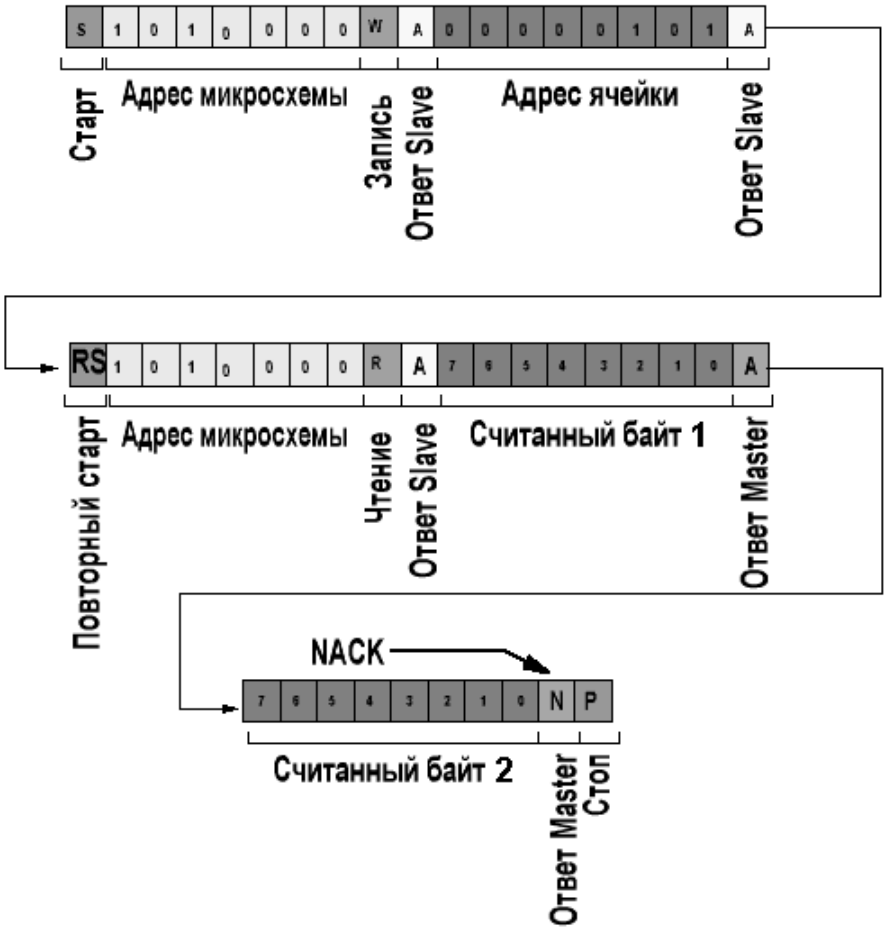


Рис. 114. Чтение данных с указанного адреса.

Такой формат пересылки используется, например, когда необходимо прочитать данные из памяти (EEPROM), часов реального времени и др. по определенному адресу. Таким образом, сначала в ведомое устройство записывается адрес чтения данных (на рисунке выше это адрес ячейки 0x05), т.е. выполняется инициализация счетчика адреса ведомого устройства; затем

делается повторный старт (RS), после чего читаются данные по заданному ранее адресу.

3.10.3.8 Арбитраж

Арбитраж помогает решать конфликтные ситуации во время передачи данных по I²C, когда присутствует несколько ведущих (режим мультимастера). Ведущий может начинать пересылку данных, только если шина свободна. Если один ведущий передает на линию данных НИЗКИЙ уровень, в то время как другой ВЫСОКИЙ, то последний отключается от линии, так как состояние SDA (НИЗКОЕ) не соответствует ВЫСОКОМУ состоянию его внутренней линии данных [13, 46, 81].

Вследствие того, что арбитраж зависит только от адреса и данных, передаваемых соревнующимися ведущими, не существует центрального ведущего, а также приоритетного доступа к шине.

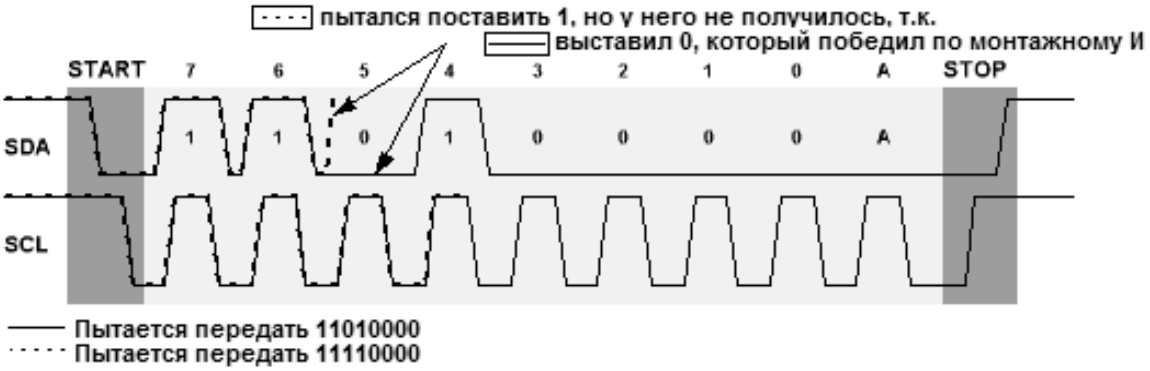


Рис. 115. Арбитраж между двумя ведущими (случай одновременной передачи данных).

Что же будет, когда два ведущих начнут передачу одновременно? Тут опять помогает свойство монтажного И: оба мастера бит за битом передают адрес ведомого, потом данные. Кто первый выставит на линию «0», тот и побеждает в этой конфликтной ситуации. Так что очевидно, что самый важный адрес должен начинаться с нулей, чтобы тот, кто к нему пытался обращаться, всегда выигрывал арбитраж. Проигравшая же сторона вынуждена ждать, пока шина не освободится.

Таким образом, арбитраж может продолжаться до окончания адреса, а если ведущие адресуют одно и то же устройство, то в арбитраже будут участвовать и данные. Вследствие такой схемы арбитража при столкновении данные не теряются.

Ведущему, проигравшему арбитраж, разрешается выдавать синхроимпульсы на шину SCL до конца байта, в течение которого был потерян доступ.

Если в устройство ведущего также встроены и функции ведомого и он проигрывает арбитраж на стадии передачи адреса, то он немедленно должен переключиться в режим ведомого, так как выигравший арбитраж ведущий мог адресовать его.

3.10.3.9 Достоинства шины I^2C

1. Требуются только две линии – линия данных (SDA) и линия синхронизации (SCL). Каждое устройство, подключённое к шине, может быть программно адресовано по уникальному адресу. В каждый момент времени существует простое отношение ведущий/ведомый: ведущие могут работать как ведущий передатчик и ведущий приёмник.
2. Шина позволяет иметь несколько ведущих, предоставляя средства для определения коллизий и арбитраж для предотвращения повреждения данных в ситуации, когда два или более ведущих одновременно начинают передачу данных. В стандартном режиме обеспечивается передача последовательных 8-битных данных со скоростью до 100 кбит/с и до 400 кбит/с в «быстром» режиме.
3. Встроенный в микросхемы фильтр подавляет всплески, обеспечивая целостность данных.
4. Максимально допустимое количество микросхем, подсоединённых к одной шине, ограничивается максимальной емкостью шины 400 пФ.

Преимущества для конструктора

I^2C-совместимые микросхемы позволяют ускорить процесс разработки от функциональной схемы до прототипа [13, 81]. Более того, поскольку такие микросхемы подключаются непосредственно к шине без каких-либо дополнительных цепей, появляется возможность модификации и модернизации системы прототипа путем подключения и отключения устройств от шины.

Вот некоторые достоинства I^2C-совместимых микросхем, которые касаются конструкторов:

1. Блоки на функциональной схеме соответствуют микросхемам, переход от функциональной схемы к принципиальной происходит быстро.
2. Нет нужды разрабатывать шинные интерфейсы, так как шина уже интегрирована в микросхемы.
3. Интегрированные адресация устройств и протокол передачи данных позволяют системе быть полностью программно определяемой.
4. Одни и те же типы микросхем могут быть часто использованы в разных приложениях.
5. Время разработки снижается, так как конструкторы быстро знакомятся с часто используемыми функциональными блоками и соответствующими микросхемами.
6. Микросхемы могут быть добавлены или убраны из системы без оказания влияния на другие микросхемы, подключенные к шине.
7. Простая диагностика сбоев и отладка; нарушения в работе могут быть немедленно отслежены.
8. Время разработки программного обеспечения может быть снижено за счет применения библиотеки повторно используемых программных модулей.

Помимо этих преимуществ, КМОП I^2C-совместимые микросхемы предоставляют для конструкторов специальные решения, которые в частности привлекательны для портативного оборудования и систем с батарейным питанием:

- Крайне низкое потребление.
- Высокая стойкость к помехам.
- Широкий диапазон питающего напряжения.
- Широкий рабочий температурный диапазон.

Преимущества для технолога (производителя)

I^2C-совместимые микросхемы не только помогают конструкторам, но и дают широкий диапазон преимуществ для технологов, потому что [13, 81]:

1. Простая двухпроводная последовательная шина I2C минимизирует соединения между микросхемами: микросхемы имеют меньше контактов и требуется меньше дорожек. В результате печатные платы становятся менее дорогими и меньше по размеру.
2. Полностью интегрированный I^2C-протокол устраняет нужду в дешифраторах адреса и другой внешней мелкой логике.
3. Возможность нескольких ведущих на I^2C-шине позволяет ускорить тестирование и настройку оборудования при помощи подключения шины к компьютеру сборочной линии.
4. Доступность I^2C-совместимых микросхем в SO- и VSO-корпусах, а также в DIL-корпусе снижает требования к размеру еще больше.

Это лишь некоторые преимущества. Кроме того, I^2C-совместимые микросхемы увеличивают гибкость системы, позволяя простое конструирование вариантов оборудования и легкую модернизацию для того, чтобы поддерживать разработки на современном уровне [13, 81]. Таким образом, целое семейство оборудования может быть разработано на основе базовой модели. Модернизация оборудования и расширение его функций (например, дополнительная память, дистанционное управление и т.п.) могут быть произведены путем простого подключения соответствующей микросхемы к шине. Если требуется большое ПЗУ, то дело лишь в выборе микроконтроллера с большим объемом ПЗУ. Поскольку новые микросхемы могут замещать старые, легко добавлять новые свойства в оборудование или увеличивать его производительность путем простого отсоединения устаревшей микросхемы и подключения к шине новой.

3.10.4Интерфейс USB

USB (Universal Serial Bus – «универсальная последовательная шина») – последовательный интерфейс передачи данных для среднескоростных и низкоскоростных периферийных устройств в вычислительной технике [37].

Разработка спецификаций на шину USB производится в рамках международной некоммерческой организации USB Implementers Forum (USB-IF), объединяющей разработчиков и производителей оборудования с шиной USB.

Для подключения периферийных устройств к шине USB используется четырёхпроводный кабель, при этом два провода (витая пара) в дифференциальном включении используются для приёма и передачи данных, а два провода – для питания периферийного устройства. Благодаря встроенным линиям питания USB позволяет подключать периферийные устройства без собственного источника питания (максимальная сила тока, потребляемого устройством по линиям питания шины USB, не должна превышать 500 мА).

К одному контроллеру шины USB можно подсоединить до 127 устройств по топологии «звезда», в том числе и концентраторы. На одной шине USB может быть до 127 устройств и до 5 уровней каскадирования хабов, не считая корневого.

USB обеспечивает обмен данными между хост-компьютером и множеством периферийных устройств (ПУ). Согласно спецификации USB, устройства (devices) могут являться хабами, функциями или их комбинацией. Устройство-хаб (hub) только обеспечивает дополнительные точки подключения устройств к шине. Устройство-функция (function) USB предоставляет системе дополнительные функциональные возможности, например: подключение к ISDN, цифровой джойстик, акустические колонки с цифровым интерфейсом и т.п. Комбинированное устройство (compound device), содержащее несколько функций, представляется как хаб с подключенными к нему несколькими устройствами. Устройство USB должно иметь интерфейс USB, обеспечивающий полную поддержку протокола USB, выполнение стандартных операций (конфигурирование и сброс) и предоставление информации, описывающей устройство. Работой всей системы USB управляет хост-контроллер (host controller), являющийся программно-аппаратной подсистемой хост-компьютера. Шина позволяет подключать, конфигурировать, использовать и отключать устройства во время работы хоста и самих устройств. Шина USB является хост-центрической: единственным ведущим устройством, которое управляет обменом, является хост-компьютер, а все присоединенные к ней периферийные устройства – исключительно ведомые. Физическая топология шины USB – многоярусная звезда. Ее вершиной является хост-контроллер, объединенный с корневым хабом (root hub), как правило, двухпортовым. Хаб является устройством-разветвителем, он может являться и источником питания для подключенных к нему устройств. К каждому порту хаба может непосредственно подключаться периферийное устройство или промежуточный хаб; шина допускает до 5 уровней каскадирования хабов (не считая корневого). Поскольку комбинированные устройства внутри себя содержат хаб, их подключения к хабу 6-го яруса уже недопустимо. Каждый промежуточный хаб имеет несколько нисходящих (downstream) портов для подключения

периферийных устройств (или нижележащих хабов) и один восходящий (upstream) порт для подключения к корневому хабу или нисходящему порту вышестоящего хаба. Логическая топология USB – просто звезда: для хост-контроллера хабы создают иллюзию непосредственного подключения каждого устройства. В отличие от шин расширения (ISA, PCI, PC Card), где программа взаимодействует с устройствами посредством обращений по физическим адресам ячеек памяти, портов ввода-вывода, прерываниям и каналам DMA, взаимодействие приложений с устройствами USB выполняется только через программный интерфейс. Этот интерфейс, обеспечивающий независимость обращений к устройствам, предоставляется системным ПО контроллера USB.

3.10.4.1 Модель передачи данных

Каждое устройство на шине USB при подключении автоматически получает свой уникальный адрес. Логически устройство представляет собой набор независимых конечных точек (endpoint, EP), с которыми хост-контроллер (и клиентское ПО) обменивается информацией. Каждая конечная точка имеет свой номер и описывается следующими параметрами:

- Требуемая частота доступа к шине и допустимые задержки обслуживания.
- Требуемая полоса пропускания канала.
- Требования к обработке ошибок.
- Максимальные размеры передаваемых и принимаемых пакетов.
- Тип передачи.
- Направление передачи (для передач массивов и изохронного обмена).

Каждое устройство обязательно имеет конечную точку с номером 0, используемую для инициализации, общего управления и опроса состояния устройства. Эта точка всегда сконфигурирована при включении питания и подключении устройства к шине. Она поддерживает передачи типа «управление». Кроме нулевой точки, устройства-функции могут иметь дополнительные точки, реализующие полезный обмен данными. Низкоскоростные устройства могут иметь до двух дополнительных точек, полноскоростные – до 15 точек ввода и 15 точек вывода (протокольное ограничение). Дополнительные точки (а именно они и предоставляют полезные для пользователя функции) не могут быть использованы до их конфигурирования (установления согласованного с ними канала).

Каналом (pipe) в USB называется модель передачи данных между хост-контроллером и конечной точкой устройства. Имеются два типа каналов: потоки и сообщения. Поток (stream) доставляет данные от одного конца канала к другому, он всегда однонаправленный. Один и тот же номер конечной точки может использоваться для двух поточных каналов – ввода и вывода. Поток может реализовывать следующие типы обмена: передача массивов, изохронный

и прерывания. Сообщение (message) имеет формат, определенный спецификацией USB. Хост посылает запрос к конечной точке, после которого передается (принимается) пакет сообщения, за которым следует пакет с информацией состояния конечной точки. Последующее сообщение нормально не может быть послано до обработки предыдущего, но при отработке ошибок возможен сброс необслуженных сообщений. Двусторонний обмен сообщениями адресуется к одной и той же конечной точке.

С каналами связаны характеристики, соответствующие конечной точке (полоса пропускания, тип сервиса, размер буфера и т.п.). Каналы организуются при конфигурировании устройств USB. Для каждого включенного устройства существует канал сообщений (Control Pipe 0), по которому передается информация конфигурирования, управления и состояния.

3.10.4.2 Протокол

Все обмены (транзакции) с устройствами USB состоят из двух-трех пакетов. Каждая транзакция планируется и начинается по инициативе контроллера, который посылает пакет-маркер (token packet). Он описывает тип и направление передачи, адрес устройства USB и номер конечной точки. В каждой транзакции возможен обмен только между адресуемым устройством (его конечной точкой) и хостом. Адресуемое маркером устройство распознает свой адрес и готовится к обмену. Источник данных (определенный маркером) передает пакет данных (или уведомление об отсутствии данных, предназначенных для передачи). После успешного приема пакета приемник данных посылает пакет квитирования (handshake packet). Хост-контроллер организует обмены с устройствами согласно своему плану распределения ресурсов. Контроллер циклически (с периодом 1,0+0,0005 мс) формирует кадры (frames), в которые укладываются все запланированные транзакции. Каждый кадр начинается с посылки маркера SOF (Start Of Frame), который является синхронизирующим сигналом для всех устройств, включая хабы. В конце каждого кадра выделяется интервал времени EOF (End Of Frame), на время которого хабы запрещают передачу по направлению к контроллеру. В режиме HS пакеты SOF передаются в начале каждого микрокадра (период 125+0,0625 мкс). Хост планирует загрузку кадров так, чтобы в них всегда находилось место для транзакций управления и прерываний. Свободное время кадров может заполняться передачами массивов (bulk transfers). В каждом (микро) кадре может быть выполнено несколько транзакций, их допустимое число зависит от длины поля данных каждой из них.

Для обнаружения ошибок передачи каждый пакет имеет контрольные поля CRC-кодов, позволяющие обнаруживать все одиночные и двойные битовые ошибки. Аппаратные средства обнаруживают ошибки передачи, а контроллер автоматически производит трехкратную попытку передачи. Если повторы безуспешны, сообщение об ошибке передается клиентскому программному обеспечению.

Все подробности организации транзакций от клиентского ПО изолируются контроллером USB и его системным программным обеспечением.

3.11 Сети передачи данных систем обработки данных. Беспроводные сенсорные сети

3.11.1Сети передачи данных

Сеть передачи данных – совокупность оконечных устройств (терминалов) связи, объединённых каналами передачи данных и коммутирующими устройствами (узлами сети), обеспечивающими обмен сообщениями между всеми оконечными устройствами. В вычислительной технике существует другой взгляд на определение сети: сеть определяется как сетевая подсистема вычислительной системы, которая включает технические средства и сервисы по организации взаимодействия компонентов (компьютеров) вычислительной системы.

3.11.2Беспроводные сенсорные сети

Сенсорная сеть (Wireless Sensor Network, WSN) – это инфраструктура, состоящая из сенсорных (измерительных), вычислительных и коммуникационных элементов, которая позволяет администратору осуществлять наблюдение за событиями и явлениями в определенной среде и реагировать на них. Под администратором понимается гражданский, правительственный, коммерческий или промышленный субъект. Средой в данном случае могут быть элементы физической природы, биологическая система или какая-нибудь информационная структура. Объединенная в сеть система сенсоров рассматривается как важная технология, которая в течение нескольких лет получит широкое распространение в многочисленных областях, в список которых будут входить не только системы безопасности. Типичными областями на сегодняшний день являются системы сбора информации (data collection), мониторинга, наблюдения и медицинской телеметрии.

В общем случае, беспроводная сенсорная сеть (БСС) – автономная самоорганизующаяся распределенная информационно-управляющая система, состоящая из малогабаритных интеллектуальных сенсорных устройств, обменивающихся информацией по беспроводным каналам связи.

У сенсорной сети есть 4 базовых компонента:

1. Совокупность сенсоров (пространственно распределенных либо локализованных).
2. Соединяющая их сеть (обычно, но не всегда, беспроводная).
3. Центральный узел сбора информации.
4. Совокупность вычислительных ресурсов для сопоставления данных, анализа событий и извлечения информации.

Следует обратить внимание на то, что в данном контексте вычислительные элементы считаются частью сенсорной сети; фактически же некоторые

вычисления могут осуществляться непосредственно внутри узлов сети – самих сенсоров. Вычислительная и коммуникационная инфраструктура сенсорных сетей зачастую весьма специфична, заточена под конкретную область применения.

Сенсоры в БСС имеют большое разнообразие задач, функций и возможностей. Сегодня эта технология динамично развивается, благодаря, с одной стороны, последним научным разработкам, и, с другой стороны, появлению огромного количества потенциальных областей применения. Сенсорная сеть – это многофункциональная область, в которую, среди прочих, входят работа с сетью, обработка сигналов, работа с базами данных, искусственный интеллект и т.п. Принципы сетевой организации и сетевые протоколы для этой технологии еще только начинают разрабатываться.

БСС может включать в себя следующие типы сенсоров:

- Электромагнитные.
- Радиочастотные.
- Оптические.
- Оптоэлектронные.
- Инфракрасные.
- Радары.
- Лазеры.
- Навигационные.
- Сейсмические.
- и другие.

Беспроводные сенсорные сети и системы разделяют на две категории:

- БСС 1 категории (C1WSN) – практически без исключения это многосвязные системы с многозвенной (многоскачковой, ретрансляционной) радиосвязью внутри беспроводных узлов или между ними, использующие методы динамической маршрутизации как на беспроводных, так и на проводных участках сети. Они характеризуются большим числом узлов (сенсоров, датчиков), массивными потоками данных (информации). Находят применение в сильно распределенных системах, таких как мониторинг окружающей среды, системы национальной безопасности, военные системы, применяемые в рекогносцировке.
- БСС 2 категории (C2WSN) – это обычно системы типа «точка-точка» или звезда, с прямыми (односкачковыми) связями, использующие статическую маршрутизацию внутри беспроводной сети. Для них типичным является наличие только одного пути от источника к

совместимому наземному или проводному передающему узлу сети (обычно БСС состоит из подвесных (прикрепляемых) узлов). Для таких БСС характерны относительно небольшие потоки данных (информации) транзакционного типа. Обычно применяются в помещениях ограниченного размера, например: в домах, на фабриках, в других строениях. Это могут быть домашние сенсорные системы («умный дом», интеллектуальное здание), системы радиочастотного распознавания (RFID).

Традиционно сенсорные сети находили применение в контексте таких сложных областей, как системы обнаружения радиации и/или ядерной угрозы, системы наведения корабельного вооружения, биомедицинские устройства, сейсмический мониторинг. В последнее время появился интерес к их применению в качестве детекторов химической и биологической угрозы для систем национальной безопасности, а также на потребительском рынке. Ниже приведен краткий список существующих и потенциальных областей применения:

Применение в военных областях:

- Наблюдение за силами противника либо за своими войсками и снаряжением.
- Рекогносцировка.
- Целеуказание.
- Оценка нанесенного урона (повреждений).
- Системы обнаружения ядерных, биологических, химических и других атак.

Применение в природной среде:

- Контроль микроклимата.
- Предсказание погоды (разного рода осадков).
- Обнаружение лесных пожаров.
- Обнаружение наводнений.

Здравоохранение:

- Удаленное наблюдение за физиологическими показателями (сенсоры на теле человека либо внутри него).
- Определение местонахождения врачей и пациентов внутри больницы.
- Распределение лекарств.
- Помощь престарелым людям, людям с ограниченной подвижностью.
- Домашнее применение («умный дом»).

- Снятие показаний со счетчиков.
- Включение/выключение электроприборов.

Промышленные и коммерческие инфраструктуры:

- Средства контроля и наблюдения в промышленных и офисных постройках.
- Контроль оборудования.
- Отслеживание и обнаружение средств передвижения.
- Наблюдение за потоком машин.
- Метрополитен.

Прежде всего, необходимо понимание того, что сенсорные сети имеют дело с пространством и временем: месторасположением, зоной покрытия и синхронизацией информации. Обычно им приходится обрабатывать большое количество информации за ограниченное время. Поэтому сенсорные сети часто поддерживают внутрисетевые вычисления и обработку. Некоторые из них используют обработку в узле-источнике, другие имеют иерархическую архитектуру. Вместо отправки необработанных данных в узлы, ответственные за общее скопление информации, узел-источник зачастую задействует собственные мощности для произведения базовых расчетов, а потом передает уже частично обработанную информацию. При использовании иерархической архитектуры обработка происходит последовательно во всех узлах, через которые проходит отправленная источником порция информации, до того, как она достигает необходимого распределительного или административного узла. Все без исключения сенсоры имеют ограниченный запас энергии и полосу пропускания для передачи сигнала. Данные ограничения в совокупности с необходимостью размещения большого количества сенсоров вызывают много трудностей в разработке и управлении БСС.

Ниже приведены ключевые понятия, относящиеся к БСС:

1. Сенсоры:
   - Внутренняя функциональность.
   - Обработка сигнала.
   - Сжатие, обнаружение и исправление ошибок, шифрование.
   - Контроль/приведение в действие.
   - Кластеризация и внутрисетевые вычисления.
   - Самоорганизация.
2. Беспроводные радио технологии:
   - Software-defined radios.

- Дальность передачи.
- Искажения при передаче.
- Способы модуляции сигнала.
- Сетевые топологии.

3. Стандарты (де-юро):
   - IEEE 802.11 a/b/g с добавочными протоколами безопасности.
   - IEEE 802.15.1 PAN/Bluetooth.
   - IEEE 802.15.3 ultrawideband (UWB).
   - IEEE 802.15.4/ZigBee.
   - IEEE 802.16 WiMax.
   - IEEE 1451.5 (Wireless Sensor Working Group).
   - Mobile IP.
4. Стандарты (де-факто):
   - TinyOS (программная платформа с открытым кодом, разрабатываемая в университете Беркли, Калифорния).
   - TinyDB (система обработки запросов для извлечения информации из сети сенсоров, оснащенных TinyOS).

Приложение А. Система ввода-вывода учебного лабораторного стенда SDK-1.1

А.1 Назначение стенда

Учебный стенд представляет собой микропроцессорный контроллер, построенный на базе однокристальной микроЭВМ ADuC812 [3] (SDK-1.1/S на ADuC842) и имеющий в своем составе разнообразные устройства, типичные для современных встроенных систем и предназначенные для ввода, обработки и вывода информации в цифровом и аналоговом виде. SDK-1.1 можно применять в качестве аппаратной базы для обучения основам современной микропроцессорной техники и программируемой логики в университетах, колледжах, физико-математических школах и на предприятиях [88].

Необходимость в подобных стендах возникает из-за того, что современные персональные компьютеры, к сожалению, достаточно плохо позволяют демонстрировать студентам все тонкости организации вычислительного процесса. Во-первых, вся «начинка» современного компьютера скрыта от пользователя операционной системой. Во-вторых, аппаратная база компьютеров часто меняется, что сильно затрудняет поддержку учебных материалов в актуальном состоянии. В-третьих, аппаратура современных компьютеров общего назначения весьма сложна и на подробное её изучение может просто не хватить времени, отведенного на курс учебной программой.

А.2 Состав стенда

В состав учебного стенда SDK-1.1 входят:

1. Микроконтроллер ADuC812 (Analog Devices), 8 Кб FLASH, 256 байт ОЗУ, 640 байт E2PROM.
2. Внешнее ОЗУ 128 Кб (с возможностью расширения до 512 Кб), подключение к МК ADuC812 по системной шине; используется для хранения пользовательских программ и данных.
3. Расширитель портов ввода–вывода – ПЛИС MAX3064 (Altera), подключение к МК ADuC812 по системной шине.
4. Внешняя память E2PROM 256 байт, подключение к МК ADuC812 по интерфейсу I2C.
5. Часы реального времени – PCF8583 (Philips), подключение по интерфейсу I2C.
6. Консоль оператора (подключение через ПЛИС к МК ADuC812):
   - Символьный жидкокристаллический индикатор (ЖКИ) WH1602B-YGK-CP (Winstar Display), 16 * 2.
   - Матричная клавиатура, 4 * 4.
   - Звуковой излучатель – 1 шт.

- Управляемые светодиоды – 8 шт.
- Ручные переключатели тестовых сигналов для аналоговых и дискретных портов ввода: коммутатор аналоговых каналов (подключен напрямую к МК ADuC812) и стимулятор дискретных портов.

7. Интерфейсы:
   - Оптически развязанный приемопередатчик инструментального канала RS–232C (для связи с персональным компьютером).
   - Интерфейс JTAG (IEEE 1149.1) для контроля периферийной шины и портов, реализованных в ПЛИС MAX3064.

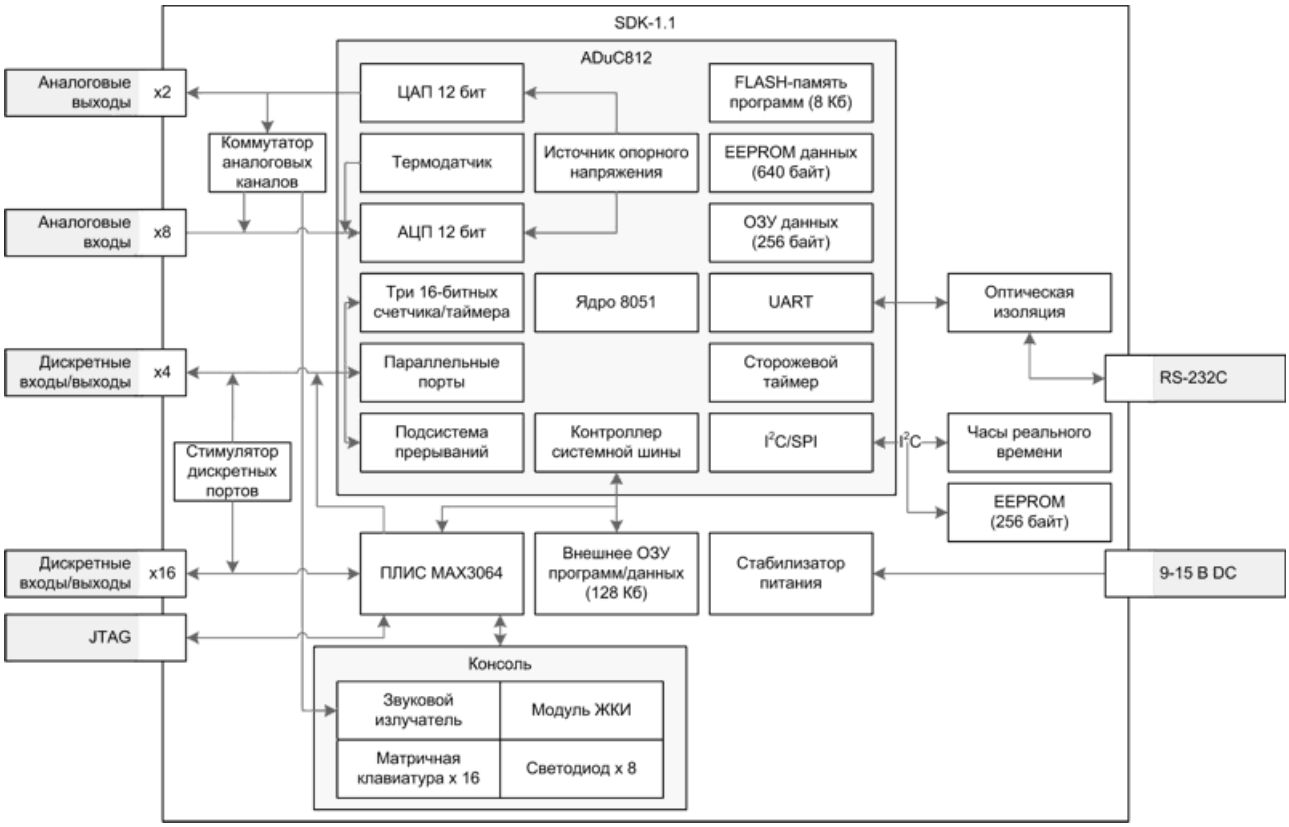


Рис. 116. Структура аппаратной части учебного стенда SDK-1.1.

A.3 Вычислительное ядро и система ввода-вывода

Вычислительное ядро – это оборудование и программные средства, которые непосредственно участвуют в решении заданной задачи (выполняют вычислительную работу). Таким образом, к вычислительному ядру контроллера SDK-1.1 относятся такие элементы, как процессор семейства MCS-51, внутреннее ОЗУ данных и FLASH-память программ (в ней располагаются резидентный загрузчик и системная таблица векторов прерываний) микроконтроллера ADuC812, внешняя память программ и данных SRAM, предназначенная для размещения пользовательских программ и данных.

Система ввода-вывода содержит элементы вычислительной системы, обеспечивающие общение вычислительного ядра с внешней средой. Таким

образом, к системе ввода-вывода контроллера SDK-1.1 относятся все контроллеры ввода-вывода и периферийные устройства, находящиеся в микроконтроллере ADuC812: порты ввода-вывода, таймеры/счетчики, универсальный асинхронный приемопередатчик (UART), контроллеры интерфейсов I^2C и SPI, АЦП, ЦАП и др. Доступ ко всей периферии в микроконтроллере осуществляется через регистры специального назначения (SFR). Также к системе ввода-вывода стенда относится расширитель портов ввода-вывода, выполненный на базе ПЛИС, и все подключенные к нему периферийные устройства: ЖКИ, матричная клавиатура, линейка светодиодов, звуковой излучатель, дискретные порты ввода-вывода. Необходимо отметить, что расширитель портов ввода-вывода разделяет единое внешнее пространство данных с внешним ОЗУ.

Данное разделение компонентов стенда SDK-1.1 по принципу принадлежности вычислительному ядру или системе ввода-вывода выполнено на логическом уровне. В следующих подразделах приведена более подробная информация о компонентах стенда, при этом «выделение» компонентов выполнено на физическом уровне, т.е. на основе принципиальной электрической схемы.

A.3.1 Микроконтроллер ADuC812

Микроконтроллер ADuC812 является клоном Intel 8051 (8052) со встроенной периферией, а значит, является представителем Гарвардской архитектуры.

Основные характеристики:

1. Рабочая частота 11,0592МГц.
2. 8 Кб FLASH-памяти (10000-50000 циклов «стирание-запись-чтение») для хранения программ. В стенде SDK-1.1 в этой памяти располагаются резидентный загрузчик и системная таблица векторов прерываний.
3. 256 байт ОЗУ данных.
4. 640 байт программируемого E2PROM со страничной организацией (160 страниц по 4 байта, 10000-50000 циклов доступа к памяти/стирание-запись-чтение) для хранения данных (например, различных настроек).
5. Адресное пространство памяти программ 64 Кб.
6. Адресное пространство внешней памяти данных 16 Мб.
7. Четыре 8-разрядных порта ввода-вывода (три двунаправленных, один порт ввода).
8. Три 16-битных таймера/счетчика и один таймер WatchDog.
9. 8-канальный 12-битный АЦП, который может работать в режиме (максимальная частота выборки-дискретизации 200 кГц).
10. 2-канальный 12-битный ЦАП.
11. Внутренний термодатчик.
12. Режим управления питанием.

13. Универсальный асинхронный приемопередатчик (UART).
14. Интерфейс I^2C (используется в стенде SDK-1.1), интерфейс SPI (не используется в стенде SDK-1.1).

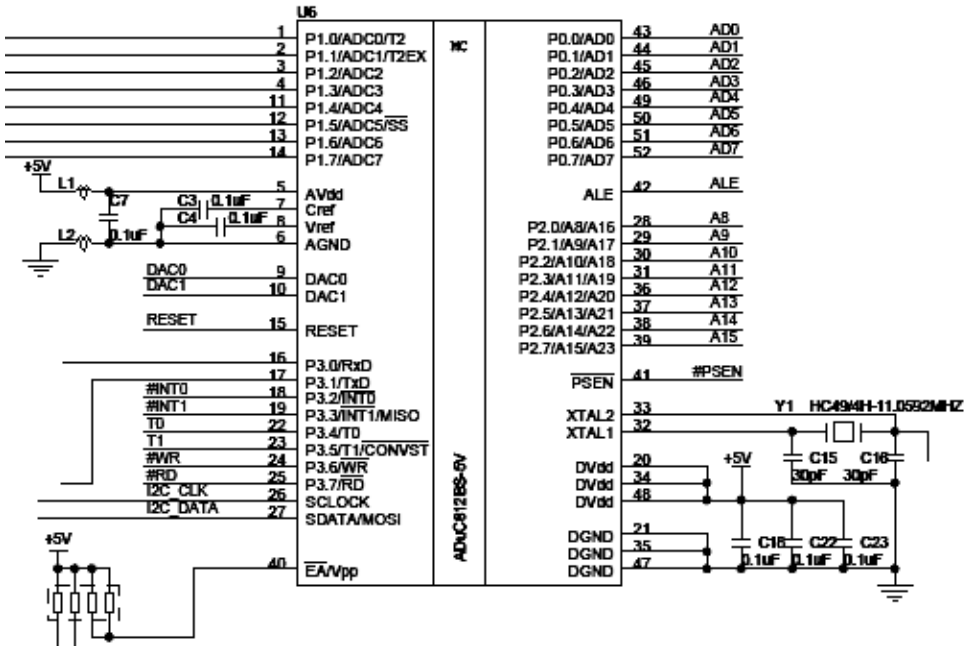


Рис. 117. Микроконтроллер ADuC812 на принципиальной электрической схеме стенда SDK-1.1.

A.3.2 Внешняя память программ и данных

Внешняя память программ и данных стенда SDK-1.1 – статическое ОЗУ (SRAM), имеющее страничную организацию и предназначенное для размещения пользовательских программ и данных. Размер страницы 64 Кб. Первая страница (страница 0) доступна для выборки и команд, и данных микроконтроллером ADuC812. Остальные страницы доступны только для размещения данных (логическое адресное пространство внешней памяти данных составляет 16 Мб). Однако реально для пользовательских программ доступно не 64 Кб, а 52 Кб, так как в младшие адреса отображается 8 Кб FLASH-памяти ADuC812 (адреса 0000h-1FFFh). Кроме того, 4 Кб зарезервировано резидентным загрузчиком МК (адреса F000h-FFFFh).

Если в пользовательской программе используются прерывания, то ее рекомендуется загружать с адреса 2100h, так как в пространстве адресов 2000h-2100h располагается пользовательская таблица векторов прерывания.

Физическое адресное пространство внешней памяти данных в стенде SDK-1.1 ограничено 512 Кб (т.е. не может быть 16 Мб), потому что, начиная с 8 страницы, располагается адресное пространство ПЛИС (MAX 3064).

Внешняя память программ и данных подключается к МК ADuC812 по системной шине (как и ПЛИС).

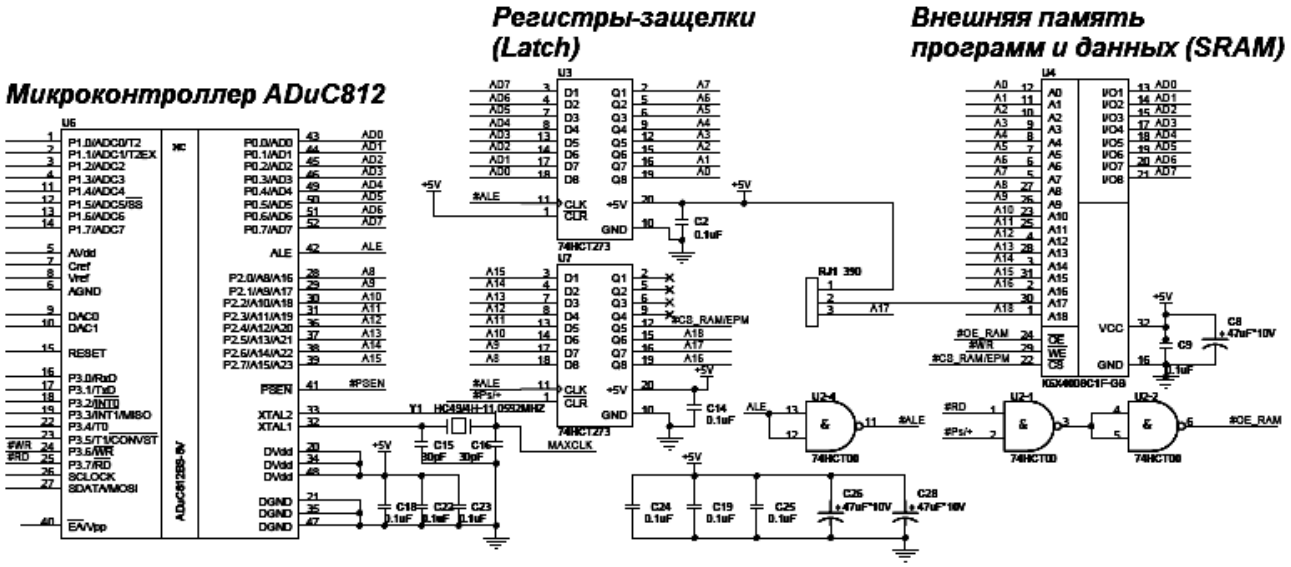


Рис. 118. Подключение внешней памяти программ и данных к МК ADuC812.

A.3.3 Порты ввода-вывода

Порт можно определить как точку, через которую осуществляется взаимодействие с каким-либо блоком в системе ввода-вывода, многоразрядный вход или выход устройства. **Порт ввода-вывода** – это логическая адресуемая единица системы ввода-вывода, которая характеризуется, в первую очередь, следующими тремя признаками:

- Адресом.
- Форматом данных, пересылаемых через него (под форматом данных подразумевается как их разрядность, так и положение значащих разрядов).
- Набором допустимых с ним операций (чтение, запись или и то, и другое).

В стенде SDK-1.1 ввод-вывод данных осуществляется с помощью портов микроконтроллера и микросхемы ПЛИС (расширитель портов ввода-вывода).

Порты P0, P1, P2, P3 микроконтроллера ADuC812 являются квазидвунаправленными портами ввода-вывода и предназначены для обеспечения обмена информацией микроконтроллера с внешними устройствами, образуя 32 линии ввода-вывода. Каждый из портов содержит восьмиразрядный регистр, имеющий байтовую и битовую адресацию для установки (запись «1») или сброса (запись «0») разрядов этого регистра с помощью программного обеспечения. Выходы этих регистров соединены с внешними ножками микросхемы.

Кроме работы в качестве обычных портов ввода-вывода внешние выводы портов P0-P3 могут выполнять ряд дополнительных (альтернативных) функций.

Порт P0 может быть использован для организации шины адреса/данных при работе микроконтроллера с внешней памятью данных или программ (см.

принципиальную схему стенда SDK-1.1), при этом через него выводится младший байт адреса (A0-A7), выдается из микроконтроллера или принимается в микроконтроллер байт данных.

Порт P1 – аналоговые входы.

Порт P2 может быть использован для организации шины адреса при работе микроконтроллера с внешней памятью данных или программ, при этом через него выводится старший байт адреса (A8–A15) для доступа к памяти программ; средний и старший байт адреса (A8 – A15, A16 – A23) для доступа к памяти данных.

Каждая линия **порта P3** имеет индивидуальную альтернативную функцию, которая может быть задействована простым обращением к устройству, соединенному с ножкой порта:

P3.0 RxD – вход последовательного порта (UART);
P3.1 TxD – выход последовательного порта (UART);
P3.2 INT0 используется как вход 0 внешнего запроса прерываний;
P3.3 INT1 используется как вход 1 внешнего запроса прерываний;
P3.4 T0 используется как вход счетчика внешних событий 0;
P3.5 T1 используется как вход счетчика внешних событий 1;
P3.6 WR – строб записи во внешнюю память данных;
P3.7 RD – строб чтения из внешней памяти данных.

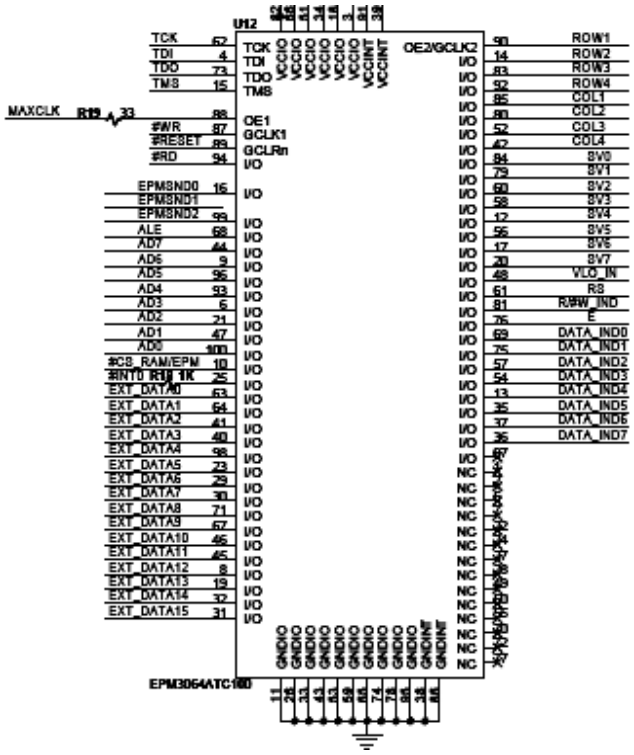
ПЛИС имеет 8 регистров, отображаемых во внешнее адресное пространство данных процессора. К ПЛИС в SDK-1.1 подключены:

- Клавиатура.
- ЖКИ.
- Линейка светодиодов.
- Звуковой излучатель.
- 16 дискретных портов ввода-вывода.

Таблица 12. Перечень регистров расширителя портов ввода-вывода.

Адрес	Регистр	Доступ	Назначение
080000H	KB	R/W	Регистр клавиатуры
080001H	DATA_IND	R/W	Регистр шины данных ЖКИ
080002H	EXT_LO	R/W	Регистр данных параллельного порта (разряды 0-7)
080003H	EXT_HI	R/W	Регистр данных параллельного порта (разряды 8-15)
080004H	ENA	W	Регистр управления портами ввода-вывода, звуком, сигналом INT0 и прерыванием от клавиатуры
080006H	C_IND	W	Регистр управления ЖКИ
080007H	SV	W	Регистр управления светодиодами

A.3.4 Расширитель портов ввода-вывода



В SDK-1.1 используется программируемая логическая интегральная схема (ПЛИС) семейства MAX3000A фирмы Altera (EPM3064A) как расширитель портов ввода-вывода.

К ПЛИС подключены:

- Клавиатура.
- ЖКИ.
- Линейка светодиодов.
- Звуковой излучатель.
- 16 дискретных портов ввода-вывода.

Для программиста расширитель портов представлен в виде нескольких однобайтовых регистров, находящихся в начале восьмой страницы внешней памяти данных.

Далее будет представлен обзор периферийных устройств, подключенных к расширителю портов ввода-вывода, их обозначение на принципиальной электрической схеме стенда SDK-1.1.

A.3.5 Периферийные устройства, подключенные к расширителю портов ввода-вывода

A.3.5.1 Матричная клавиатура

Клавиатура подключена через расширитель портов ввода-вывода.

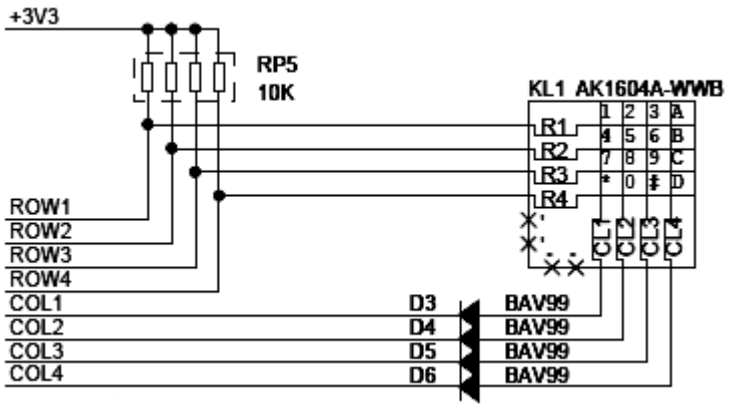
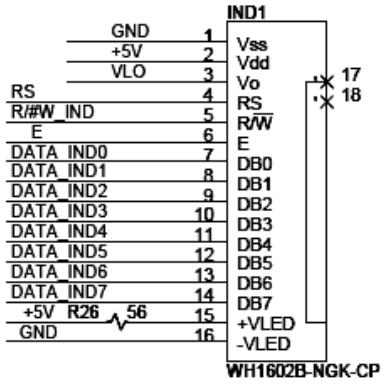


Рис. 119. Матричная клавиатура.

Клавиатура организована в виде матрицы 4x4. Доступ к колонкам и рядам организован как чтение/запись определенного регистра ПЛИС (4 бита соответствуют 4 колонкам, другие 4 бита – рядам). Ряды ROW1..ROW4

подключены к плюсу питания через резисторы. Это обеспечивает наличие логической единицы при отсутствии нажатия (чтение регистра ПЛИС). На столбцы клавиатуры COL1..COL4 подают логический ноль (запись через регистр ПЛИС). При нажатии на кнопку происходит изменение значения сигнала на входе соответствующего ряда с единицы на ноль.

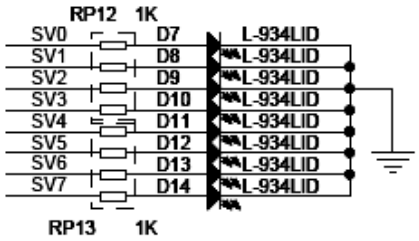
A.3.5.2 Жидкокристаллический индикатор



ЖКИ работает в текстовом режиме (2 строки по 16 символов), имеет подсветку (цвет желто-зеленый). Основные характеристики:

- Габариты: 80x36x13,2 мм.
- Активная область 56,21x11,5 мм.
- Размеры точки 0,56x0,66 мм; размеры символа 2,96x5,56 мм.
- Встроенный набор 256 символов (ASCII + кириллица).
- Генератор символов с энергозависимой памятью на 8 пользовательских символов.

A.3.5.3 Светодиодные индикаторы



Светодиодные индикаторы подключены к расширителю портов ввода-вывода. Так как все катоды светодиодов подключены к корпусу, для зажигания светодиодов необходимо подать напряжение +5В (логическая «1») на соответствующий анод. Резисторы R12, R13 ограничивают ток, текущий через порт ввода-вывода и светодиод. В данном случае приблизительный ток можно вычислить по закону Ома: I = U/R, I = 3,3/1000 = 3,3мА. От силы тока зависит яркость горения светодиода. Если ток сделать очень большим, то порт ввода-вывода или светодиод могут выйти из строя.

A.3.5.4 Звукоизлучатель

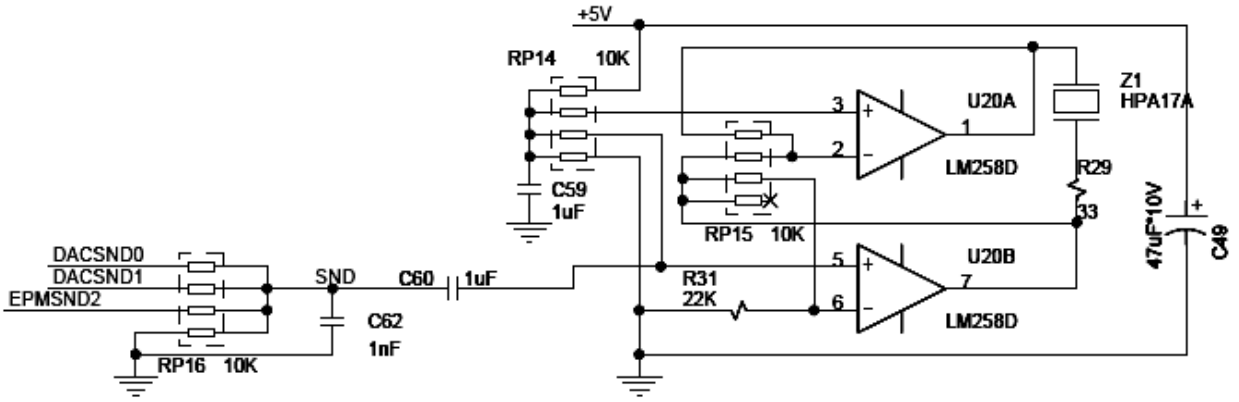


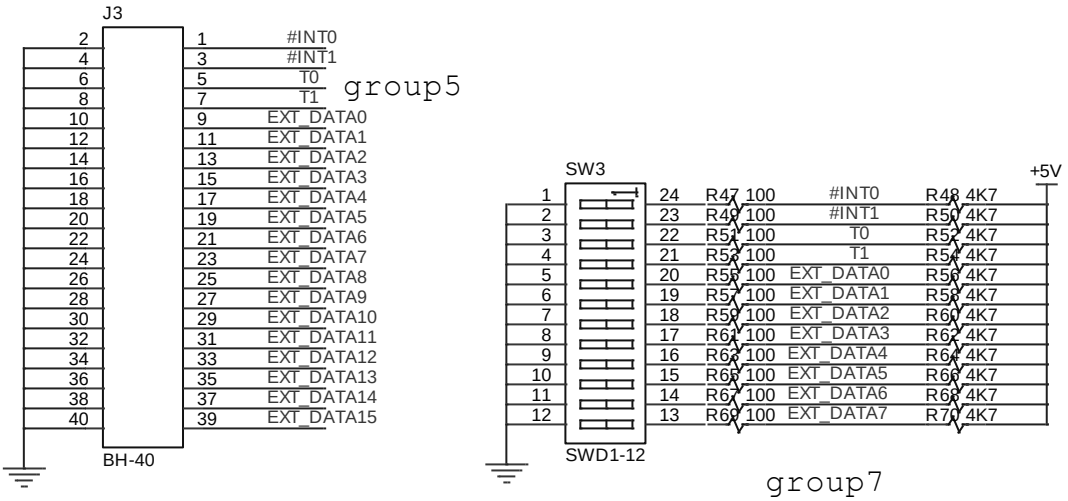
Рис. 120. Звукоизлучатель.

В SDK-1.1 используется пьезоэлектрический звукоизлучатель HPA17A (Z1). Выходы EPMSND0-EPMSND2 подключены к расширителю портов ввода-вывода (в SDK-1.1R3/R4/R5 сигналы EPMSND0 и EPMSND1 не подключены).

A.3.5.5 Дискретные входы-выходы

Дискретные входы-выходы предназначены для ввода и вывода информации, представленной в двоичном виде. Сигнал на входе или выходе дискретного порта может принимать значение логического нуля или единицы. В SDK-1.1 дискретные порты выведены на разъем J3. Эти порты можно использовать для подключения модулей ввода-вывода SDX-0.9 или каких-либо других внешних устройств. Кроме этого, к дискретным входам-выходам подключены DIP-переключатели (SW3), позволяющие задавать фиксированные значения сигналов на входах. По умолчанию все входы притянуты к логической единице (через резисторы на +5В). При замыкании переключателя SW3 на выбранном входе появляется логический ноль.

Дискретные входы-выходы не имеют гальванической изоляции. Логическому нулю соответствует 0В, а логической единице +5В (уровни TTL). Нагрузочная способность дискретных портов ввода-вывода, подключенных к разъему J3, невелика, так как на разъем выведены порты ADuC812 без каких-либо дополнительных усилителей.



A.3.6 Аналоговый ввод-вывод

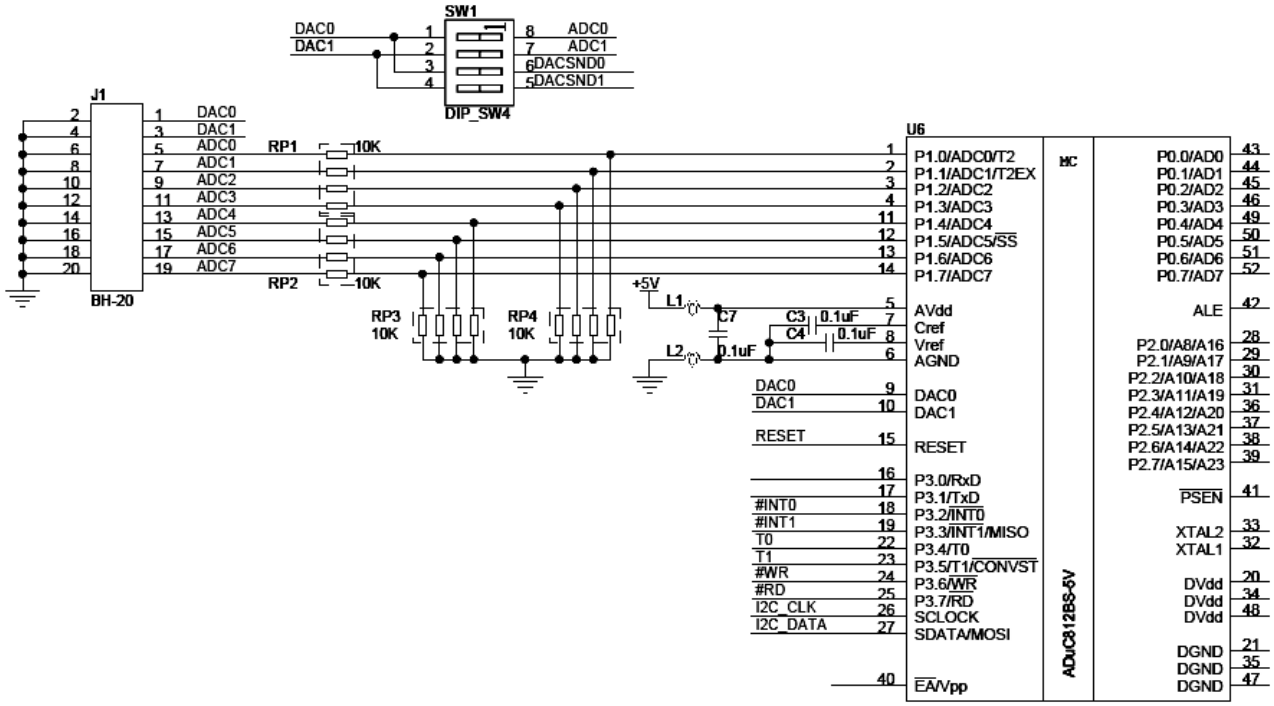


Рис. 121. Аналоговый ввод-вывод.

ADuC812 имеет в своем составе восемь быстродействующих 12-разрядных АЦП и два 12-разрядных ЦАП. Для коррекции зависимости параметров ЦАП и АЦП от температуры в ADuC812 встроен термодатчик. Все входы ЦАП и выходы АЦП выведены на разъем J1. Кроме того, выходы DAC0 и DAC1 можно замкнуть на входы ADC0 и ADC1 с помощью переключателя SW1.

A.3.7 I²C-устройства

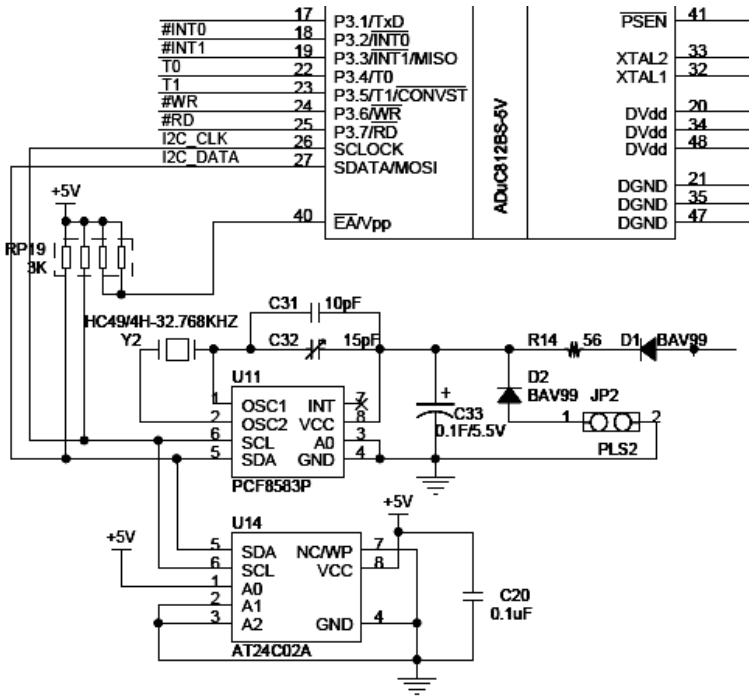


Рис. 122. I²C-устройства стенда SDK-1.1.

В стенде SDK-1.1 два устройства подключены по шине I²C к микроконтроллеру ADuC812: часы реального времени PCF8583 (U11) и E²PROM AT24C02A (U14).

E²PROM – электрически стираемое перепрограммируемое постоянное запоминающее устройство. Объем памяти E²PROM (AT24C02A, Atmel), установленной в стенде SDK-1.1, составляет 256 байт (2 Кбит).

Основные характеристики:

- Возможность перезаписи до 1 млн. раз.
- Возможность побайтовой и постраничной записи (в текущей конфигурации размер страницы составляет 8 байт).

Микросхема PCF8583 (Philips) – часы реального времени (часы/календарь) с памятью объемом 256 байт, работающие от кварцевого резонатора с частотой 32,768 кГц. Из 256 байт памяти собственно часами используются только первые 16 (8 постоянно обновляемых регистров-защелок на установку/чтение даты/времени и 8 на будильник), остальные 240 байт доступны для хранения данных пользователя. Точность измерения времени – до сотых долей секунды.

A.3.8 Последовательные и параллельные интерфейсы

В качестве примеров последовательных интерфейсов стенда SDK-1.1 можно назвать последовательный интерфейс RS-232, I^2C.

В качестве примеров параллельных интерфейсов (а точнее, последовательно-параллельных) стенда SDK-1.1 можно назвать системную шину, соединяющую микроконтроллер ADuC812 с внешней памятью программ и данных (SRAM) и ПЛИС; интерфейс подключения ЖКИ к ПЛИС.

Основными компонентами контроллера ЖКИ являются память DDRAM (Data Display RAM), память CGRAM (Character Generator RAM), память CGROM (Character Generator ROM), счетчик адреса, регистр команд IR (Instruction Register), регистр данных DR (Data Register). Регистр команд предназначен для записи в него таких операций, как очистка дисплея, перемещение курсора, включение/выключение дисплея, а также установка адреса памяти DDRAM и CGRAM, для последующего их выполнения. Регистр данных временно хранит данные, предназначенные для записи или чтения из DDRAM или CGRAM (символы). Эти два регистра можно выбрать с помощью регистрового переключателя RS (Register Select).

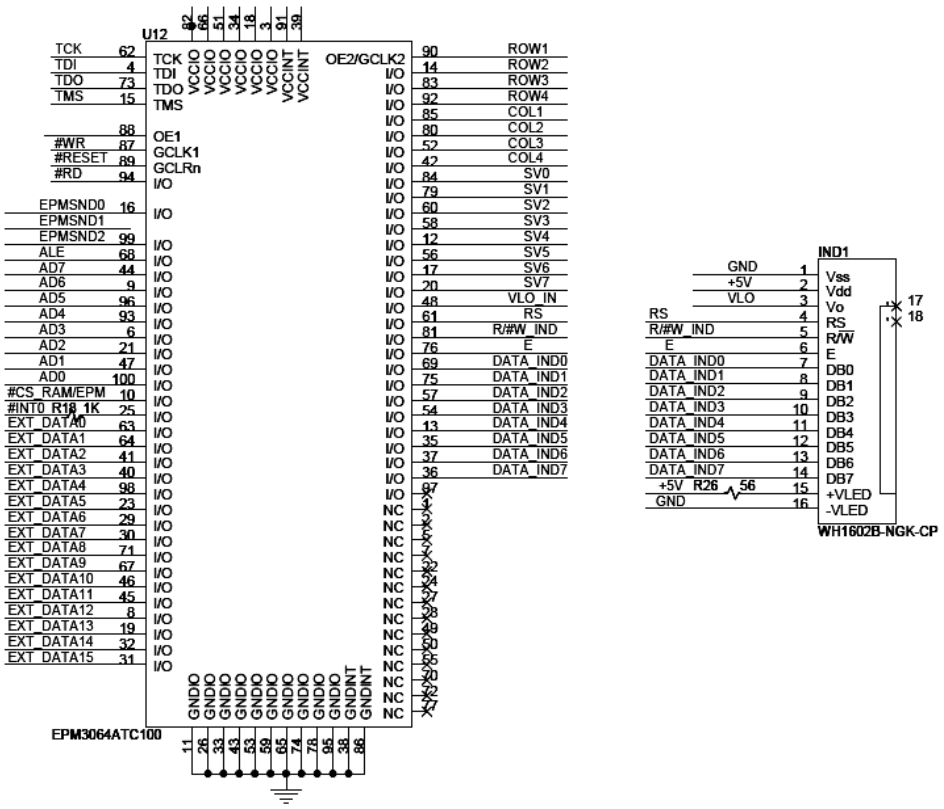


Рис. 123. Принципиальная схема стенда SDK-1.1: ПЛИС и ЖКИ.

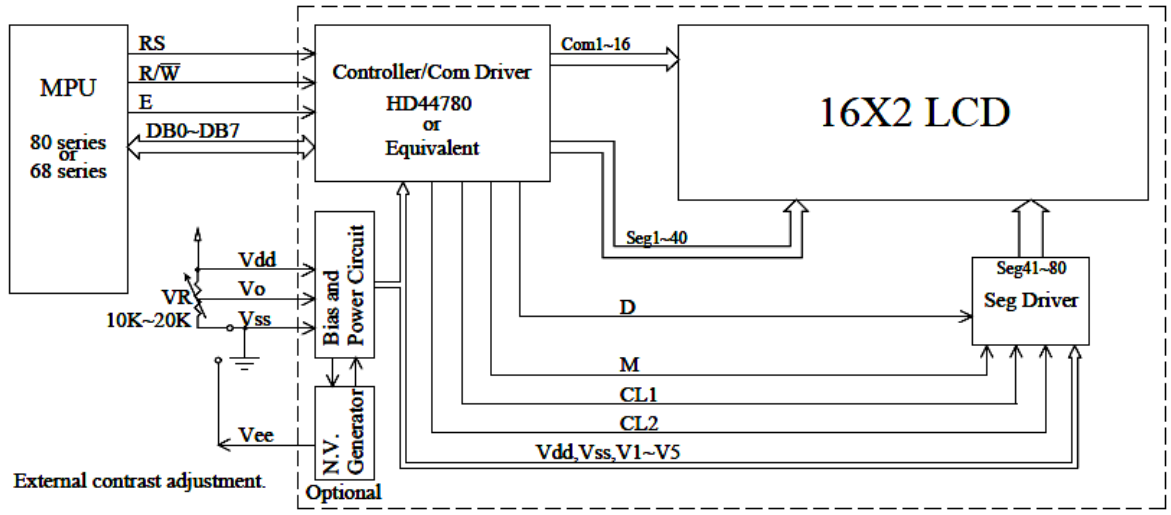


Рис. 124. Модуль ЖКИ

Таблица 13. Магистраль информационного канала и управления информационным каналом модуля ЖКИ.

Обозначение	Уровень	Описание
RS	H/L	H – данные, L – команды
R/W	H/L	H – чтение (с ЖКИ), L – запись (в ЖКИ)
E	H, H/L	Разрешающий сигнал (строб)
DB0-DB7	H/L	Шина данных/команд

Таблица 14. Варианты значений сигналов RS и R/W.

RS	R/W	Команда
0	0	IR используется для внутренних команд (очистка дисплея и т.д.).
0	1	Считывание флага занятости (DB7) и счетчика адреса (от DB0 до DB7).
1	0	Запись данных в DDRAM или CGRAM (из регистра данных в DDRAM или CGRAM).
1	1	Чтение данных из DDRAM или CGRAM (из DDRAM или CGRAM в регистр данных).

Приложение Б. Комплекс лабораторных работ для учебного лабораторного стенда SDK-1.1

Б.1 Лабораторная работа № 1 «Дискретные порты ввода-вывода»

Б.1.1 Задание

Разработать и реализовать драйверы светодиодных индикаторов и DIP-переключателей контроллера SDK-1.1. Написать тестовую программу с использованием разработанных драйверов по алгоритму, соответствующему варианту задания.

Б.1.2 Порты ввода-вывода

Каждый процессор для встраиваемых применений имеет некоторое количество внешних линий ввода-вывода, подключенных к внешним выводам микросхемы и называемых внешними портами. Одиночные (одноразрядные, состоящие из одной линии) порты ввода-вывода объединяются в группы обычно по 4, 8 или 16 линий, которые называются параллельными портами. Через порты процессорное ядро взаимодействует с различными внешними устройствами: считывает значения входных сигналов и устанавливает значения выходных сигналов. Во встраиваемых системах в качестве внешних устройств чаще всего рассматриваются датчики, исполнительные устройства, устройства ввода-вывода данных оператором, устройства внешней памяти.

По типу сигнала различают порты: дискретные (цифровые), аналоговые и перестраиваемые.

Дискретные (цифровые) порты используются для ввода-вывода дискретных значений логического «0» или «1». В большинстве современных процессоров для встраиваемых применений поддерживается как независимое управление каждой линией параллельного порта, так и групповое управление всеми разрядами.

Аналоговый порт ввода-вывода предназначен для работы с аналоговым (непрерывным) сигналом. В отличие от дискретного сигнала, принимающего всего два значения, напряжение аналогового сигнала может иметь любое значение (в определенных пределах) и меняться во времени. Пример аналогового сигнала – синусоида. Такую форму, например, имеет электрическое напряжение в сети 220 В 50 Гц.

Перестраиваемые порты ввода-вывода настраиваются на аналоговый или цифровой режим работы.

Б.1.3 Описание работы

В контроллере SDK-1.1 изучение дискретных портов ввода-вывода будет проводиться на примере работы с набором светодиодных индикаторов и DIP[1]-переключателей (см. рис. 125). Данные устройства вывода и ввода, соответственно, подключены к расширителю портов ввода-вывода, который в стенде SDK-1.1 выполнен на базе ПЛИС (программируемая логическая интегральная схема). Для программиста расширитель портов представлен в виде нескольких однобайтовых регистров, находящихся в начале восьмой страницы внешней памяти данных.

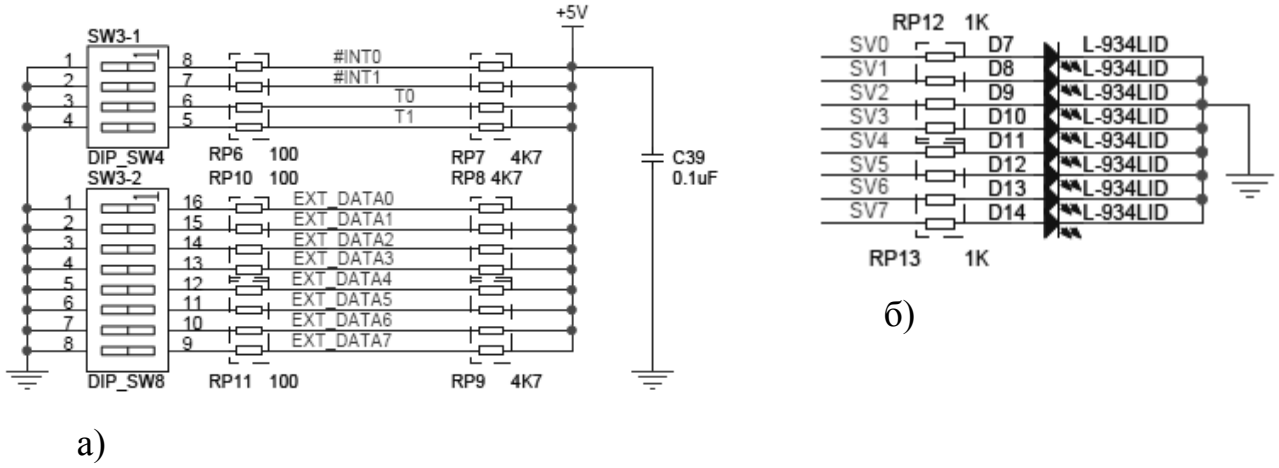
По умолчанию линии 0-7 дискретного параллельного порта ПЛИС притянуты к логической «1» (через резисторы на +5В). Для их принудительного «обнуления» в схему введен набор DIP-переключателей (на рис. 126 обозначение «SW3-2»), замыкающих соответствующие линии через резисторы 100 Ом на корпус. Для того чтобы принудительно «обнулить» соответствующую линию, необходимо установить соответствующий DIP-переключатель в положение «ON».

Рис. 125. Расположение программируемых в лабораторной работе светодиодных индикаторов и DIP-переключателей контроллера SDK-1.1

Для удобства, двоичный (шестнадцатеричный) код, задаваемый набором DIP-переключателей, выставляется следующим образом: первый переключатель

[1] DIP (Dual In-line Package) – тип корпуса микросхем, микросборок и некоторых других электронных компонентов. Имеет прямоугольную форму с двумя рядами выводов по длинным сторонам. Может быть выполнен из пластика (PDIP) или керамики (CDIP). В корпусе DIP могут выпускаться различные полупроводниковые или пассивные компоненты: микросхемы, сборки диодов, транзисторов, резисторов, малогабаритные переключатели.

соответствует младшему (0-му) разряду двоичного кода; восьмой переключатель соответствует старшему (7-му) разряду. При этом единичный разряд в коде – это соответствующий DIP-переключатель в положении «ON», нулевой разряд – это соответствующий DIP-переключатель в положении «OFF».



а) б)

Рис. 126. Обозначение DIP-переключателей (а) и светодиодных индикаторов (б) на принципиальной электрической схеме контроллера SDK-1.1

При установке DIP-переключателя в положение «ON» напряжение не сразу устанавливается на уровне 0В, а «скачет» в течение некоторого времени (1-10 мс), пока цепь надежно не замкнется. После того, как DIP-переключатель будет установлен в положение «OFF», напряжение также «скачет», пока не установится на уровне +5В. Такого рода переходные процессы называются дребезгом. Таким образом, при изменении положения DIP-переключателя («ON» ↔ «OFF») возникает эффект дребезга, отрицательное влияние которого в данной работе никак не устраняется.

Подробнее эта проблема будет исследоваться в лабораторной работе № 4 «Клавиатура».

Б.1.4 Требования к выполнению работы

1. Все программы должны быть написаны на языке Си.
2. Разрабатываемые драйверы устройств должны быть выполнены в виде отдельных программных модулей (файлов), содержащих функции по работе с заданным одним устройством. Например, в этой лабораторной работе драйвер светодиодных индикаторов должен содержать функцию установки состояния светодиодов, а драйвер DIP-переключателей – функцию чтения их состояния. В главном программном модуле должна решаться предлагаемая вариантом задания прикладная задача с использованием разработанных драйверов.
3. В тестовой программе для осуществления анимации запрещается использовать «покадровое» формирование картинки. Для реализации алгоритма анимации должны быть использованы логические, арифметические и бинарные операции, а также операции сдвигов.

4. Для задержек в программе следует использовать пустые циклы (тысячи – десятки тысяч итераций, в зависимости от длительности задержки).
5. В варианте задания представлен лишь фрагмент анимации, при этом все анимации являются циклическими. По приведенному фрагменту требуется определить алгоритм анимации и реализовать его.
6. Текст программы должен соответствовать правилам оформления программ на языке Си, приведенным в приложении (Приложение Г. Требования к оформлению программ на языке Си, [95]).

Б.1.5 Содержание отчета

1. Титульный лист.
2. Задание.
3. Блок-схема программы.
4. Исходный текст программы с комментариями.
5. Основные результаты.

Б.1.6 Литература

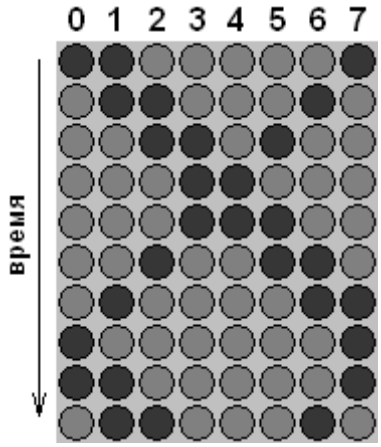
Литература к лабораторной работе: [45], [48], [49], [50], [51], [57], [75], [88], [93], [3], [8], [9], [11], [17], [22].

Б.1.7 Варианты заданий

Во всех вариантах задания единичный разряд в шестнадцатеричном коде – это соответствующий DIP-переключатель в положении «ON», нулевой разряд – это соответствующий DIP-переключатель в положении «OFF».

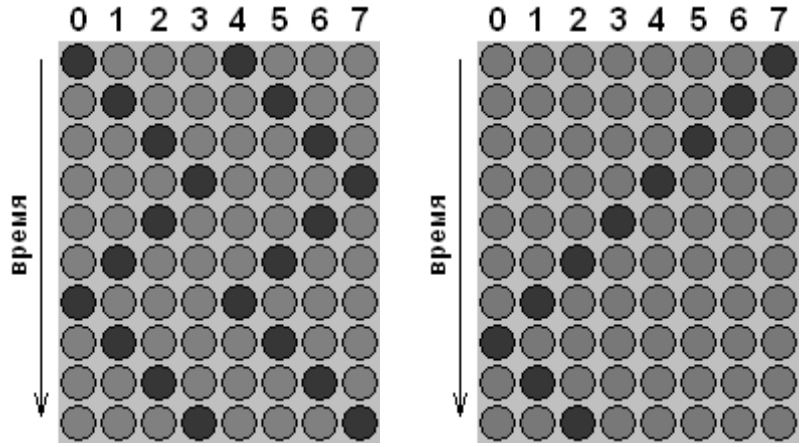
В варианте задания представлен лишь фрагмент анимации, при этом все анимации являются циклическими. По приведенному фрагменту требуется определить алгоритм анимации и реализовать его.

1. В случае установки на DIP-переключателях кода 0x11 (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться анимация, показанная ниже. Во всех остальных случаях светодиодные индикаторы отражают значение, выставленное на DIP-переключателях.

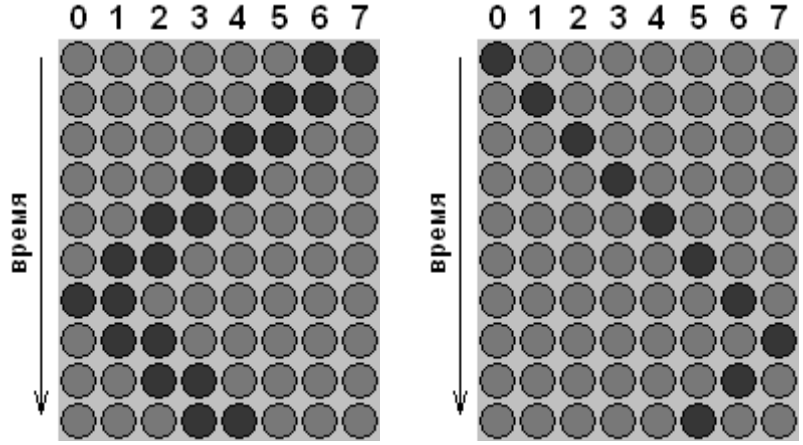


2. В случае установки на DIP-переключателях кода 0x22 (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться первая анимация, в случае установки кода

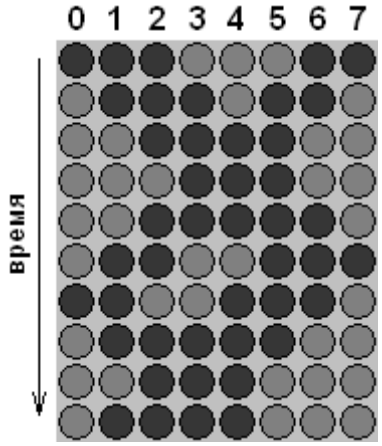
0xDD (шестнадцатеричное значение) – вторая анимация. Во всех остальных случаях светодиодные индикаторы отражают инвертированное значение, выставленное на DIP-переключателях.



3. В случае установки на DIP-переключателях кода 0x33 (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться первая анимация, в случае установки кода 0xCC (шестнадцатеричное значение) – вторая анимация. Во всех остальных случаях светодиодные индикаторы отражают значение, выставленное на DIP-переключателях.

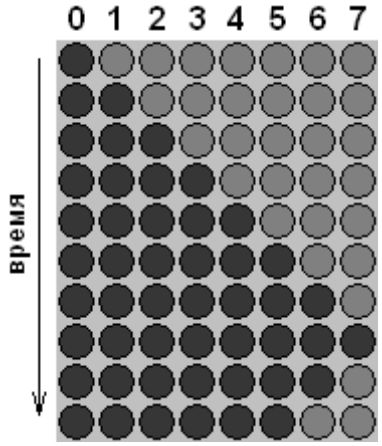


4. В случае установки на DIP-переключателях кода 0x44 (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться анимация, показанная ниже. Во всех остальных случаях светодиодные индикаторы отражают инвертированное значение, выставленное на DIP-переключателях.

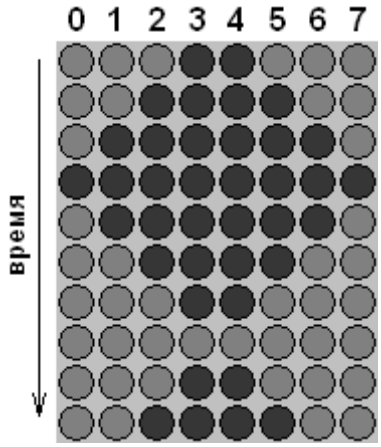


5. В случае установки на DIP-переключателях кода 0x55 (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться анимация, показанная ниже. Во всех

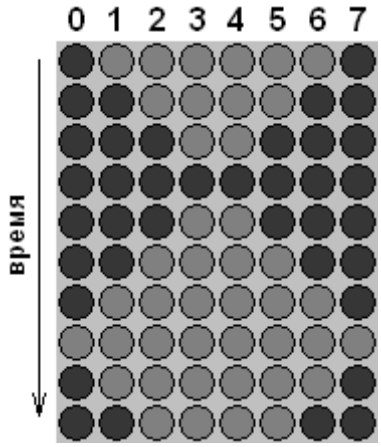
остальных случаях светодиодные индикаторы отражают значение, выставленное на DIP-переключателях.



6. В случае установки на DIP-переключателях кода 0x66 (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться анимация, показанная ниже. Во всех остальных случаях светодиодные индикаторы отражают инвертированное значение, выставленное на DIP-переключателях.

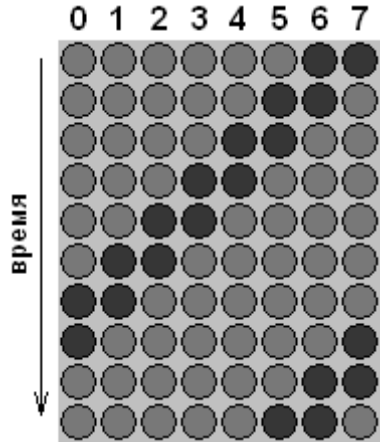


7. В случае установки на DIP-переключателях кода 0x77 (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться анимация, показанная ниже. Во всех остальных случаях светодиодные индикаторы отражают значение, выставленное на DIP-переключателях.

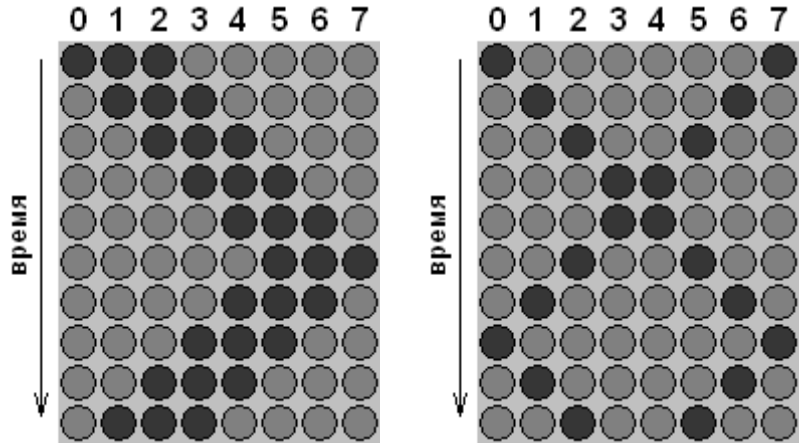


8. В случае установки на DIP-переключателях кода 0xAA (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться анимация, показанная ниже. Во всех

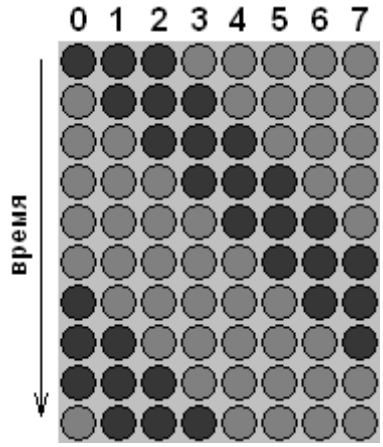
остальных случаях светодиодные индикаторы отражают инвертированное значение, выставленное на DIP-переключателях.



9. В случае установки на DIP-переключателях кода 0xBB (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться первая анимация, в случае установки кода 0x44 (шестнадцатеричное значение) – вторая анимация. Во всех остальных случаях светодиодные индикаторы отражают значение, выставленное на DIP-переключателях.



10. В случае установки на DIP-переключателях кода 0xCC (шестнадцатеричное значение) на светодиодные индикаторы должна выводиться анимация, показанная ниже. Во всех остальных случаях светодиодные индикаторы отражают инвертированное значение, выставленное на DIP-переключателях.



Б.2 Лабораторная работа № 2 «Таймеры. Система прерываний»

Б.2.1 Задание

Разработать и реализовать драйвер системного таймера микроконтроллера ADuC812. Написать тестовую программу с использованием разработанного драйвера по алгоритму, соответствующему варианту задания.

Б.2.2 Таймеры-счетчики

Таймер позволяет производить отсчет временных интервалов заданной продолжительности. Принцип действия таймера основан на двоичном счетчике с возможностью предварительной записи исходного значения. После каждого такта синхросигнала счетчик прибавляет или отнимает единицу от имеющегося у него значения. При достижении нуля (т.е. при переполнении), счетчик вырабатывает активный уровень на выходе. Как правило, выходной сигнал таймера заводят на вход запроса прерывания микропроцессора или контроллера прерываний.

В большинстве современных встроенных систем таймеры используются в качестве основы для организации системы разделения времени на базе переключателя задач. В данном случае таймер используется в паре с механизмом прерываний.

Микроконтроллер ADuC812 имеет три программируемых 16-битных таймера/счетчика: таймер 0, таймер 1, таймер 2. Каждый таймер состоит из двух 8-битных регистров THX и TLX. Все три таймера могут быть настроены на работу в режимах «таймер» или «счетчик».

В режиме «таймер» регистр инкрементируется каждый машинный цикл, т.е. можно рассматривать это как подсчет машинных циклов. Так как машинный цикл состоит из 12 перепадов напряжения на тактовом входе микроконтроллера, частота инкрементирования таймера в 12 раз меньше тактовой частоты микроконтроллера (соответствующего кварцевого резонатора).

В режиме «счетчик» регистр инкрементируется по перепаду из «1» в «0» внешнего входного сигнала, подаваемого на вывод микроконтроллера T0, T1 или T2 (см. принципиальную электрическую схему контроллера SDK-1.1). Когда опрос показывает высокий уровень в текущем машинном цикле и низкий уровень в следующем, счетчик увеличивается на 1. Таким образом, на распознавание периода требуются два машинных цикла, максимальная частота подсчета входных сигналов равна 1/24 частоты кварцевого резонатора. На длительность периода входных сигналов ограничений сверху нет. Для гарантированного прочтения входной сигнал должен удерживать значение «1», как минимум, в течение одного машинного цикла микроконтроллера.

Таким образом, отличие этих двух режимов работы таймера заключается в типе источника инкрементирования. В остальном функционирование в режимах счетчика и таймера аналогично.

Б.2.3 Описание работы

Данная лабораторная работа посвящена изучению таймера и системы прерываний микроконтроллера ADuC812. Основными функциями системного таймера являются: измерение интервалов времени и выполнение периодических задач. В данной работе с помощью таймеров требуется управлять светодиодными индикаторами (динамическая индикация) или звуковым излучателем (проигрывание мелодии), входящими в состав контроллера SDK-1.1.

Драйвер системного таймера должен состоять из двух частей: основной части, находящейся в обработчике прерывания и прикладной части – API-функций, используемых в программе:

Функция	Описание
`void InitTimer(void)`	Инициализация таймера.
`unsigned long GetMsCounter(void)`	Получение текущей метки времени в миллисекундах.
`unsigned long DTimeMs(unsigned long t0)`	Измерение количества миллисекунд, прошедших с временной метки `t0` и до текущего времени.
`void DelayMs(unsigned long t)`	Задержка на `t` миллисекунд.

Кроме того, могут быть реализованы функции работы с таймером в режиме «счетчик» (например, чтение счетчика).

Драйвер светодиодных индикаторов/звукового излучателя (зависит от варианта задания) должен быть реализован по тому же принципу, что и драйвер системного таймера. А именно: в обработчике прерывания от таймера должна выполняется сама динамическая индикация/проигрывание мелодии – примеры периодических задач, а через API-функции осуществляется настройка отображения анимации/звука и управление этими процессами.

Б.2.3.1 Пояснения для вариантов заданий с использованием светодиодов

Написать драйвер, позволяющий управлять яркостью свечения линейки светодиодов. Драйвер должен обеспечивать независимое управление яркостью каждого светодиода. Яркость свечения светодиодов должна задаваться в процентах.

Яркость свечения светодиода можно регулировать, меняя скважность сигнала управляющего питанием светодиода. **Скважность** – это отношение

периода следования (повторения) импульсов одной последовательности к их длительности. Величина, обратная скважности, называется **коэффициентом заполнения**. Сигнал со скважностью, равной двум, т.е. длительностью импульсов равной длительности пауз между ними, называется **меандром**. Внешне меандр представляет собой прямоугольное колебание.

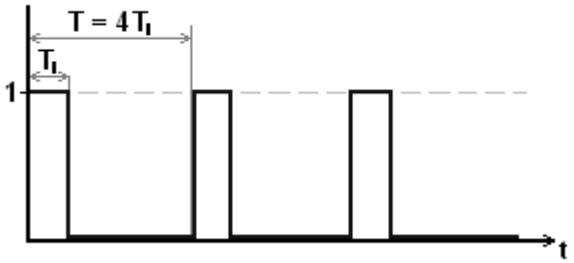


Рис. 127. Пример сигнала со скважностью 4

Таким образом, при скважности питающего сигнала равной 1, светодиод будет светиться с максимальной яркостью. При скважности сигнала управления равной 4 (см. рис. 127), светодиод будет светиться с яркостью 25% от максимальной и т.д. Кроме того, нужно стремиться уменьшить период и увеличить частоту сигнала управления, чтобы избежать эффекта мерцания светодиодов.

Б.2.3.2 Пояснения для вариантов заданий с использованием звукового излучателя

Требуется написать драйвер звукового излучателя, позволяющий задавать частоту (в герцах) и длительность звука (в миллисекундах).

Для управления звуковым излучателем используется регистр ПЛИС ENA (адрес 080004h). 2-4 биты регистра ENA управляют величиной напряжения на динамике, т.е. позволяют задавать громкость звука[2]: чем больше «единиц» выставлено в этих битах, тем громче звук. Частота звука задается частотой смены нулей и единиц в управляющих битах регистра ENA. Ниже, в табл. 15, приведены частоты нот первой октавы.

Таблица 15. Частоты музыкальных нот

Нота	Частота, Гц
До	261,63
Ре	293,67
Ми	329,63
Фа	349,22
Соль	391,99

[2] Такая возможность есть только в стендах SDK-1.1 ревизии R1 и R2.

Ля	440,00
Си	493,88

Частоты нот каждой соседней октавы отличаются в 2 раза. Например, частота ноты «ля» второй октавы 880 Гц.

Б.2.4 Особенности обработки прерываний в стенде SDK-1.1

В микроконтроллере ADuC812 девять источников прерываний с двумя уровнями приоритетов (см. табл. 16). Когда происходит прерывание, значение программного счетчика помещается в стек, а в сам счетчик загружается адрес соответствующего вектора прерывания.

Таблица 16. Адреса векторов прерывания микроконтроллера ADuC812

Прерывание	Наименование	Адрес вектора
PSMI	Монитор источника питания ADuC812	43H
IE0	Внешнее прерывание INT0	03H
ADCI	Конец преобразования АЦП	33H
TF0	Переполнение таймера/счетчика 0	0BH
IE1	Внешнее прерывание INT1	13H
TF1	Переполнение таймера/счетчика 1	1BH
I2CI/ISPI	Прерывание от I^2C/SPI	3BH
RI/TI	Прерывание от UART	23H
TF2/EXF2	Переполнение таймера/счетчика 2	2BH

Для каждого источника прерывания программист может задать один из двух уровней приоритета: высокий и низкий. Прерывание с высоким уровнем приоритета может прерывать обслуживание прерывания с низким уровнем приоритета, а если прерывания с разными уровнями происходят одновременно, то прерывание с высоким приоритетом будет обслужено первым. Обслуживание прерывания не может быть прервано прерыванием с таким же уровнем приоритета. Если два прерывания с одинаковым уровнем приоритета происходят одновременно, то порядок их обработки определяется с помощью следующей табл. 17:

Таблица 17. Порядок обработки прерываний с одинаковым уровнем приоритета

Источник	Очередность	Описание
PSMI	1 (Наивысшая)	Монитор источника питания ADuC812
IE0	2	Внешнее прерывание INT0
ADCI	3	Конец преобразования АЦП
TF0	4	Переполнение таймера/счетчика 0
IE1	5	Внешнее прерывание INT1

TF1	6	Переполнение таймера/счетчика 1
I2CI+ISPI	7	Прерывание от I^2C/SPI
RI+TI	8	Прерывание от UART
TF2+EXF2	9 (Низшая)	Переполнение таймера/счетчика 2

Прерывания ADuC812 имеют вектора в диапазоне 0003h-0043h, которые попадают в область младших адресов памяти программ. Это пространство соответствует 8Кб (0000h-2000h) FLASH-памяти.

В стенде SDK-1.1 пользователь не имеет возможности записи во FLASH-память (запись программ осуществляется во внешнюю память программ и данных), следовательно, не может подставить свои процедуры обработки прерываний (точнее, команды перехода к процедурам) по адресам, соответствующим векторам прерываний.

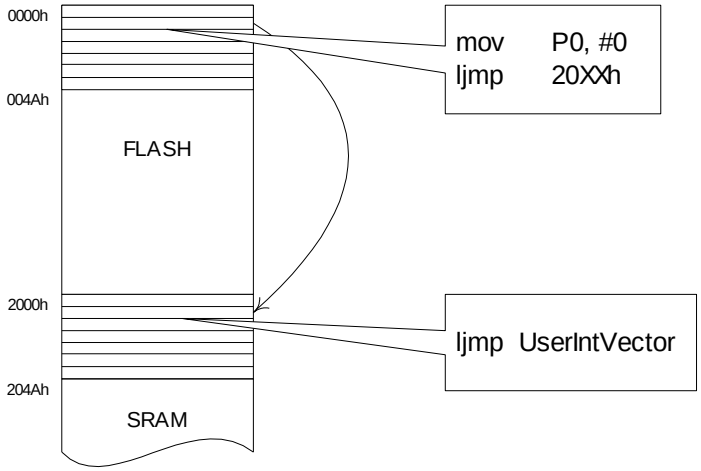


Рис. 128. Использование прерываний в SDK-1.1

Проблема использования прерываний в пользовательских программах решается следующим образом (см. рис. 128):

1. По адресам (0003h-0043h) векторов прерываний во FLASH-памяти SDK-1.1 располагаются команды переходов на вектора пользовательской таблицы, размещенной в адресах 2003h-2043h.
2. По адресам векторов пользовательской таблицы пользователем указываются команды переходов на процедуры обработки прерываний.

Приведем пример помещения собственного вектора в пользовательскую таблицу. Программа использует прерывание от таймера 0 (прерывание 0Bh) для зажигания линейки светодиодов. Данный пример адаптирован для компилятора SDCC. По сравнению с компилятором Keil C51 изменения коснулись функции SetVector (в связи с тем, что компилятор SDCC и микроконтроллер ADuC812 используют различные способы записи многобайтовых чисел в памяти; SDCC – little-endian, ADuC812 – big-endian) и заголовка функции обработчика прерывания (T0_ISR).

```
#include "aduc812.h"
#define MAXBASE          8

////////////////////////// WriteMax ////////////////////////////
// Запись байта в регистр ПЛИС
// Вход:
//   regnum – адрес регистра ПЛИС,
//   val – записываемое значение.
// Выход: нет.
// Результат: нет.
////////////////////////////////////////////////////////////////
void WriteMax (unsigned char xdata *regnum, unsigned char val)
{
 // Сохранение текущего значения регистра страниц
 unsigned char oldDPP = DPP;

 DPP = MAXBASE;    // Установка адреса страницы ПЛИС
 *regnum = val;    // Запись значения в регистр ПЛИС
 DPP = oldDPP;     // Восстановление сохраненного значения
                   // регистра страниц
}

////////////////////////// WriteLED  ///////////////////////////
// Функция установки состояния линейки светодиодов.
// Вход:
//   value – состояния светодиодов.
// Выход: нет.
// Результат: нет.
////////////////////////////////////////////////////////////////
void WriteLED(unsigned char value)
{
 // Запись состояния светодиодов в регистр 7-й регистр ПЛИС
 WriteMax( 7, value );
}

////////////////////////// T0_ISR //////////////////////////////
// Обработчик прерывания от таймера 0.
// Вход: нет.
// Выход: нет.
// Результат: нет.
////////////////////////////////////////////////////////////////
void T0_ISR( void ) __interrupt ( 1 )
{
 WriteLED( 0x55 ); // Зажигание светодиодов (через один)
}

////////////////////////// SetVector  //////////////////////////
// Функция, устанавливающая вектор прерывания в
// пользовательской таблице прерываний.
// Вход:
//   Vector – адрес обработчика прерывания,
//   Address – вектор пользовательской таблицы прерываний.
// Выход: нет.
// Результат: нет.
////////////////////////////////////////////////////////////////
void SetVector(unsigned char xdata * Address, void * Vector)
{
 unsigned char xdata * TmpVector;    // Временная переменная

 // Первым байтом по указанному адресу записывается
 // код команды передачи управления ljmp, равный 02h
 *Address = 0x02;
```

```
 // Далее записывается адрес перехода Vector
 TmpVector = (unsigned char xdata *) (Address + 1);
 *TmpVector = (unsigned char) ((unsigned short)Vector >> 8);
 ++TmpVector;
 *TmpVector = (unsigned char) Vector;
 // Таким образом, по адресу Address теперь
 // располагается инструкция ljmp Vector
}

////////////////////////// Main //////////////////////////////////
// Главная функция
//////////////////////////////////////////////////////////////////
void main( void )
{
 TH0 = 0xFF;        // Инициализация таймера 0
 TL0 = 0xF0;        //
 TMOD = 0x01;       //
 TCON = 0x10;       //

 // Установка вектора в пользовательской таблице
 SetVector( 0x200B, (void *)T0_ISR );
 // Разрешение прерываний от таймера 0
 ET0 = 1; EA = 1;

 while( 1 );
}
```

В ходе выполнения лабораторной работы производится ознакомление с организацией и принципом работы не только таймеров/счетчиков по прерываниям, но и внешних прерываний INT0/INT1. Необходимо отметить, что в случае последних возможны следующие настройки: по перепаду (фронт или спад напряжения) или по уровню напряжения на внешнем входе. В контроллере SDK-1.1 линии счетного входа таймера/счетчика 0/1 (T0/1) и внешних прерываний INT0/INT1 выведены на DIP-переключатели SW3-1 (см. рис. 129).

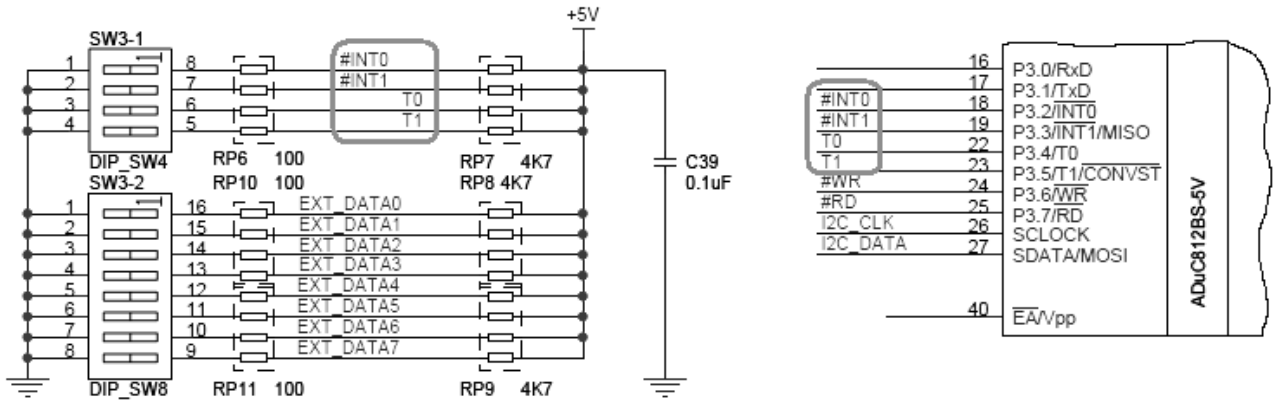


Рис. 129. Расположение линий T0/1 и INT0/1 МК ADuC812 (справа) и их вывод на DIP-переключатели (слева) на принципиальной электрической схеме контроллера SDK-1.1

Необходимо настроить внешнее прерывание INT0/1 так, чтобы оно работало по спаду, а не по уровню (регистр специального назначения TCON). Для корректной работы внешнего прерывания INT0 5-й бит регистра управления ENA (адрес 0x080004 ПЛИС) должен быть равен 1.

Б.2.5 Требования к выполнению работы

1. Все программы должны быть написаны на языке Си.
2. Разрабатываемые драйверы устройств должны быть выполнены в виде отдельных программных модулей (файлов), содержащих функции по работе с заданным одним устройством.
3. В программе должны быть использованы механизмы взаимного исключения (см. [51], IOS2003_lab4.pdf).
4. В варианте задания с использованием светодиодов представлен лишь фрагмент анимации, при этом все анимации являются циклическими. По приведенному фрагменту требуется определить алгоритм анимации и реализовать его. Смена яркости свечения светодиодов должна быть плавной, без видимых «рывков» и эффекта «мерцания».
5. Текст программы должен соответствовать правилам оформления программ на языке Си, приведенным в приложении (Приложение Г. Требования к оформлению программ на языке Си, [95]).

Б.2.6 Содержание отчета

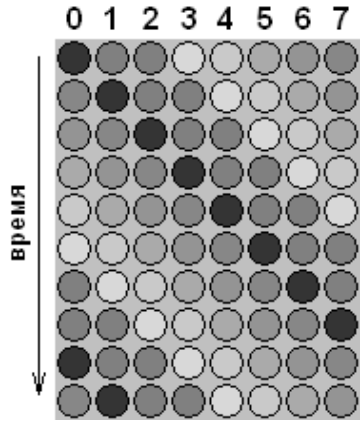
1. Титульный лист.
2. Задание.
3. Модель взаимодействия прикладной программы, прикладной части драйверов и системной части драйверов.
4. Исходный текст программы с комментариями.
5. Основные результаты.

Б.2.7 Литература

Литература к лабораторной работе: [45], [48], [49], [50], [51], [57], [75], [88], [93], [3], [8], [9], [11], [17], [22].

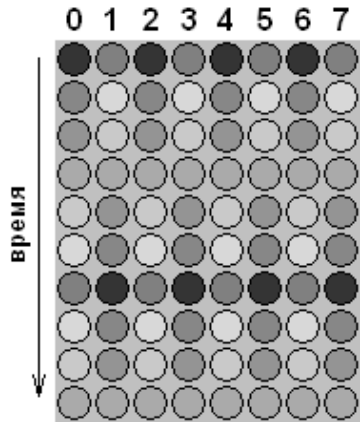
Б.2.8 Варианты заданий

1. Контроллер SDK-1.1 на линейку светодиодов циклически отображает приведенную ниже анимацию или выводит количество замыканий входа T0 (счетный вход таймера 0 на рис. 129). Подсчет количества замыканий входа должен быть реализован с помощью таймера-счетчика 0. Смена режима отображения производится по поступлению сигнала внешнего прерывания INT0 (рис. 129). В анимации должны использоваться не менее 6 градаций яркости свечения светодиодов (например, 0%, 20%, 40%, 60%, 80% и 100%). В результате выполнения работы должны быть разработаны драйверы системного таймера, таймера-счетчика, светодиодных индикаторов, внешнего прерывания.



2. Контроллер SDK-1.1 циклически проигрывает восходящую гамму нот первой октавы (длительность каждой ноты – 1 секунда) и на линейку светодиодов выводит количество замыканий входа T0 (счетный вход таймера 0 на рис. 129). Подсчет количества замыканий входа должен быть реализован с помощью таймера-счетчика 0. В результате выполнения работы должны быть разработаны драйверы системного таймера, таймера-счетчика, звукового излучателя, светодиодных индикаторов.

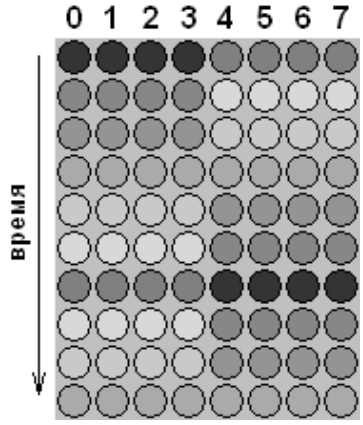
3. Контроллер SDK-1.1 на линейку светодиодов циклически отображает приведенную ниже анимацию или выводит количество замыканий входа T1 (счетный вход таймера 1 на рис. 129). Подсчет количества замыканий входа должен быть реализован с помощью таймера-счетчика 1. Смена режима отображения производится по поступлению сигнала внешнего прерывания INT1 (рис. 129). В анимации должны использоваться не менее 6 градаций яркости свечения светодиодов (например, 0%, 20%, 40%, 60%, 80% и 100%). В результате выполнения работы должны быть разработаны драйверы системного таймера, таймера-счетчика, светодиодных индикаторов, внешнего прерывания.



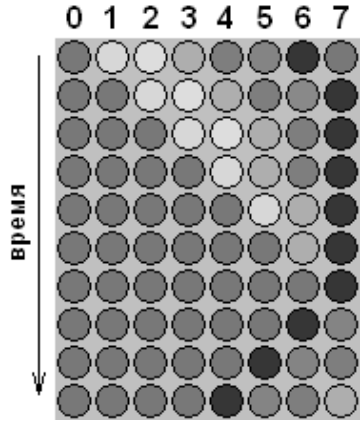
4. Контроллер SDK-1.1 циклически проигрывает нисходящую гамму нот второй октавы (длительность каждой ноты – 1 секунда) и на линейку светодиодов выводит количество замыканий входа T1 (счетный вход таймера 1 на рис. 129). Подсчет количества замыканий входа должен быть реализован с помощью таймера-счетчика 1. В результате выполнения работы должны быть разработаны драйверы системного таймера, таймера-счетчика, звукового излучателя, светодиодных индикаторов.

5. Контроллер SDK-1.1 на линейку светодиодов циклически отображает приведенную ниже анимацию или выводит количество замыканий входа T0 (счетный вход таймера 0 на рис. 129). Подсчет количества замыканий входа должен быть реализован с помощью таймера-счетчика 0. Смена режима отображения производится по поступлению сигнала внешнего прерывания INT1 (рис. 129). В анимации должны использоваться не менее 6 градаций яркости свечения светодиодов (например, 0%, 20%, 40%, 60%, 80% и 100%). В результате

выполнения работы должны быть разработаны драйверы системного таймера, таймера-счетчика, светодиодных индикаторов, внешнего прерывания.



6. Контроллер SDK-1.1 циклически проигрывает восходящую гамму нот первой октавы (длительность каждой ноты – 0,5 секунды) и на линейку светодиодов выводит количество замыканий входа INT0 (линия внешнего прерывания INT0 на рис. 129). В результате выполнения работы должны быть разработаны драйверы системного таймера, звукового излучателя, светодиодных индикаторов, счетчика срабатываний внешнего прерывания.
7. Контроллер SDK-1.1 на линейку светодиодов циклически отображает приведенную ниже анимацию или выводит количество замыканий входа T1 (счетный вход таймера 1 на рис. 129). Подсчет количества замыканий входа должен быть реализован с помощью таймера-счетчика 1. Смена режима отображения производится по поступлению сигнала внешнего прерывания INT0 (рис. 129). В анимации должны использоваться не менее 6 градаций яркости свечения светодиодов (например, 0%, 20%, 40%, 60%, 80% и 100%). В результате выполнения работы должны быть разработаны драйверы системного таймера, таймера-счетчика, светодиодных индикаторов, внешнего прерывания.



8. Контроллер SDK-1.1 циклически проигрывает нисходящую гамму нот второй октавы (длительность каждой ноты – 0,5 секунды) и на линейку светодиодов выводит количество замыканий входа INT1 (линия внешнего прерывания INT1 на рис. 129). В результате выполнения работы должны быть разработаны драйверы системного таймера, звукового излучателя, светодиодных индикаторов, счетчика срабатываний внешнего прерывания.

Б.3 Лабораторная работа № 3 «Последовательный интерфейс RS-232. UART»

Б.3.1 Задание

Разработать и написать драйверы последовательного канала для учебно-лабораторного стенда SDK-1.1 с использованием и без использования прерываний. Написать тестовую программу для разработанных драйверов, которая выполняет определенную вариантом задачу.

Б.3.2 Особенности последовательного интерфейса в микроконтроллере с ядром MCS-51

Последовательный порт в микроконтроллерах MCS-51 позволяет осуществлять последовательный дуплексный ввод-вывод в синхронном и асинхронном режимах с разными скоростями обмена. Помимо обычного ввода-вывода, в нем предусмотрена аппаратная поддержка взаимодействия нескольких микроконтроллеров.

Схематически контроллер последовательного порта представлен на рис. 130. Как видно из рисунка, регистр SBUF, доступный пользователю для чтения и записи, есть на самом деле два регистра, в один из которых можно только записывать, а из другого – читать. Контроллер позволяет дуплексный обмен данными, т.е. одновременно может передавать и принимать информацию по линиям TxD (P3.1) и RxD (P3.0) соответственно.

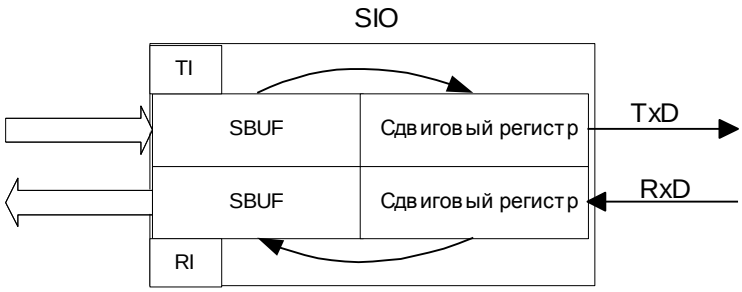


Рис. 130. Схема контроллера UART МК ADuC812

Передача инициируется записью в SBUF байта данных. Этот байт переписывается в сдвиговый регистр, из которого пересылается бит за битом. Как только байт будет переслан целиком, устанавливается флаг TI, сигнализирующий о том, что контроллер готов к передаче очередного байта.

Прием осуществляется в обратном порядке: после начала процесса приема принятые биты данных последовательно сдвигаются в сдвиговом регистре, пока не будет принято их установленное количество, затем содержимое сдвигового регистра переписывается в SBUF и устанавливается флаг RI. После этого возможен прием следующей последовательности бит. Необходимо отметить, что запись принятого байта из сдвигового регистра в SBUF происходит только при условии, что RI=0, поэтому при заборе пользовательской программой очередного байта из SBUF ей необходимо сбрасывать этот флаг, иначе

принятый в сдвиговый регистр, но не записанный в SBUF, следующий байт будет безвозвратно утерян.

Контроллер последовательного порта может работать в одном из четырех режимов (один синхронный и три асинхронных), различающихся скоростями обмена (бод) и количеством передаваемых/принимаемых бит:

- Режим 0 (синхронный): данные передаются и принимаются через RxD, TxD является синхронизирующим (выдает импульсы сдвига). Скорость в этом режиме фиксирована (1/12 частоты тактового генератора).
- Режим 1 (8 бит данных, асинхронный, переменная скорость). Передаются 10 бит: старт-бит, 8 бит данных (SBUF) и стоп-бит.
- Режим 2 (9 бит данных, асинхронный, фиксированная скорость). Передаются 11 бит: старт-бит, 8 бит (SBUF), бит TB8/RB8 (посылка/прием соответственно) и 1 стоп-бит. Биты TB8 и RB8 содержатся в регистре SCON (биты 3 и 2 соответственно), первый устанавливается программно, а второй содержит 9-й бит принятой комбинации. С помощью них можно организовать, например, контроль четности или обмен с двумя стоп-битами.
- Режим 3 (9 бит данных, асинхронный, переменная скорость). То же, что и режим 2, только скорость обмена переменная.

В режиме 1 и 3 используются Таймер 0 и/или Таймер 1 для настройки скорости обмена.

Б.3.3 Организация буферизированного последовательного ввода-вывода в стенде SDK-1.1

В микроконтроллере ADuC812 прерывание от последовательного канала имеет номер 4, адрес-вектор 0x23 и генерируется контроллером последовательного порта во время установки флага TI или RI. Чтобы разрешить это прерывание, нужно установить биты ES и EA регистра IEN0.

Управление обработчику прерывания передается тогда, когда UART либо готов к передаче очередного байта (TI=1), либо принял байт из канала (RI=1). При использовании этой схемы появляется проблема посылки первого байта.

Пусть прикладная программа желает послать байт, когда очередь WFIFO пуста и в порт данные не записывались длительное время (TI=0). Так как обработчик прерываний, обслуживающий очередь, вызывается только при установленных флагах RI или TI, прикладной программе придется либо самой записать первый байт в SBUF (начав тем самым процесс передачи), либо, записав байты для передачи в очередь WFIFO, искусственно установить флаг TI (вызвав тем самым обработчик, который обслужит очередь). Проблема разрешима, однако при ее решении необходимо учитывать, что TI также сброшен, когда контроллер занят посылкой байта, даже если очередь (WFIFO) пуста (посылка последнего байта в очереди).

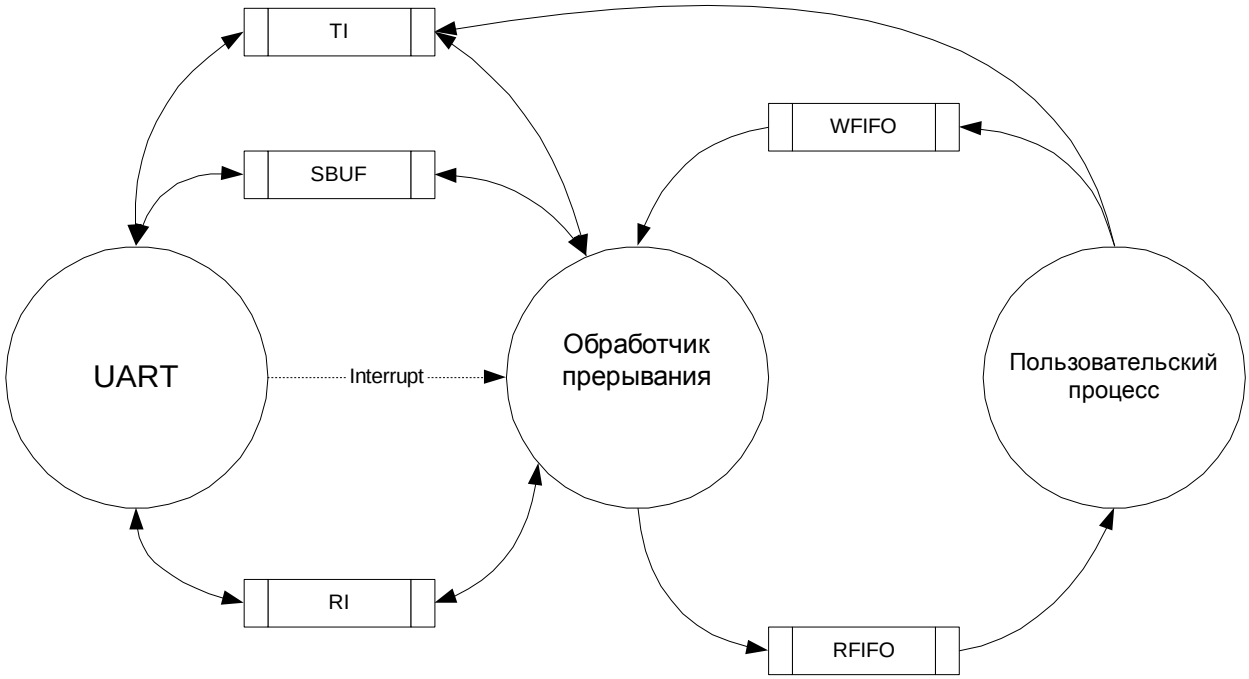


Рис. 131. Драйвер UART (диаграмма DFD)

Б.3.4 Организация программы

На рис. 132 изображены основные компоненты программы, которая должна быть результатом выполнения задания: драйвер последовательного канала; разборщик/преобразователь данных, переданных (принятых) по UART; пользовательский процесс, в котором выполняется прикладная задача.

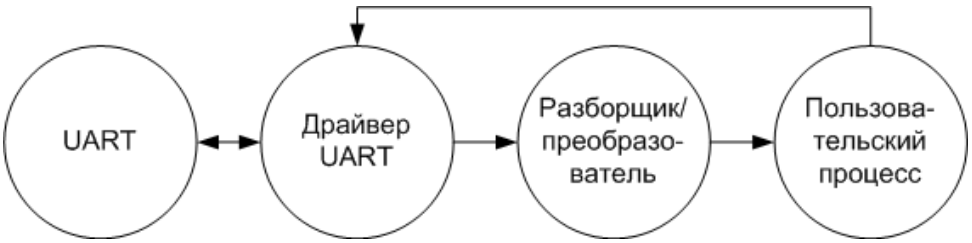


Рис. 132. Основные составляющие программы

Драйвер последовательного канала, организованного по опросу, должен содержать три функции: функция инициализации UART, функция передачи и приема байта данных по UART.

Драйвер последовательного канала по прерыванию состоит из функции инициализации UART, обработчика прерывания от UART, циклических буферов чтения и записи, API-функций: чтения байта из последовательного канала и записи байта в последовательный канал. Взаимодействие обработчика и API-функций осуществляется только через буфер. Обработчик взаимодействует с последовательным каналом напрямую. Если прерывание вызвано приходом байта по последовательному каналу, то обработчик записывает байт в буфер чтения RFIFO. Если прерывание вызвано окончанием передачи, то обработчик проверяет состояние буфера записи WFIFO, и, если он не пуст, считывает и пересылает следующий байт.

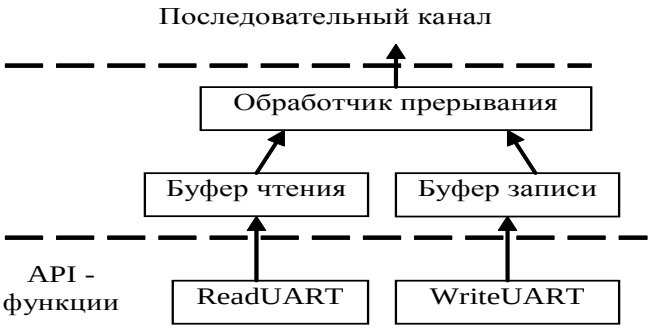


Рис. 133. Компоненты драйвера последовательного канала, организующего обмен данными по прерыванию

Функция чтения байта ReadUART() проверяет состояние буфера RFIFO, и если он не пуст, считывает из него байт. Функция WriteUART() записывает байт в буфер записи последовательного канала WFIFO. Если до записи буфер был пуст, необходимо аппаратно вызвать обработчик прерывания, чтобы инициировать передачу данных. В случае, когда к моменту записи в буфере уже находятся какие-либо данные, обработчик вызывать не надо, так как запись была инициирована раньше и в данный момент последовательный канал занят передачей данных, записанных в буфер ранее.

Принятые при помощи драйвера UART данные требуют обработки в соответствии с вариантом задания: смена нижнего регистра на верхний, фильтрация только по числовым или буквенным значениям, по количеству вводимых значений и др. Таким образом, требуется написать функцию (разборщик/преобразователь), которая будет отфильтровывать значения, неудовлетворяющие заданному алфавиту корректных символов. Далее над отфильтрованными данными производится указанная в варианте задания операция: арифметическое действие или конвертирование. Обработка ошибок в разборщике/преобразователе и прикладном алгоритме тоже должна быть предусмотрена.

Кроме того, для выполнения задания требуется использовать драйвер DIP-переключателей и светодиодных индикаторов.

Б.3.5 Организация обработчика прерывания UART

Окончание передачи байта и окончание приема байта по последовательному каналу вызывают одно и тоже прерывание и в обработчике два этих прерывания различаются программно. Обработчик логически разделен на две части: одна работает при установленном флаге RI (окончание приема), а другая при установленном флаге TI (окончание передачи). Ниже приведен шаблон обработчика прерывания последовательного канала:

```
////////////////////////// SIO_ISR ////////////////////////////////
// Обработчик прерывания UART.
// Вход: нет.
// Выход: нет.
// Результат: нет.
///////////////////////////////////////////////////////////////////
void SIO_ISR( void ) __interrupt ( 4 )
```

```
{
if(TI)
{
    // Передача байта
    // Читаем WFIFO (буфер передачи) и записываем его в SBUF
}

if(RI)
{
    // Прием байта
    // Читаем SBUF и записываем его в RFIFO (буфер приема)
}
}
```

Кроме того, функция SetVector должна быть такой (см. лабораторную работу № 2):

```
////////////////////// SetVector  ////////////////////////
// Функция, устанавливающая вектор прерывания в
// пользовательской таблице прерываний.
// Вход:
//   Vector – адрес обработчика прерывания,
//   Address – вектор пользовательской таблицы прерываний.
// Выход: нет.
// Результат: нет.
//////////////////////////////////////////////////////////
void SetVector(unsigned char xdata * Address, void * Vector)
{
 unsigned char xdata * TmpVector;

 *Address = 0x02;

 TmpVector = (unsigned char xdata *) (Address + 1);
 *TmpVector = (unsigned char) ((unsigned short)Vector >> 8);
 ++TmpVector;
 *TmpVector = (unsigned char) Vector;
}
```

Б.3.6 Описание работы

Данная лабораторная работа посвящена изучению организации и принципов работы контроллера последовательного канала UART микроконтроллера ADuC812 стенда SDK-1.1.

В рамках лабораторной работы необходимо разработать программу для контроллера SDK-1.1 (ведомый), который обменивается данными с персональным компьютером (ведущий). В качестве канала связи используется последовательный канал RS-232. На стороне персонального компьютера имеется инструментальное средство для обеспечения взаимодействия с SDK-1.1 – это терминальная программа MЗР. С описанием инструментальной системы (G)MЗР можно ознакомиться в соответствующем руководстве пользователя.

Программа должна выполнять две задачи в соответствии с вариантом задания. Первая задача выполняется при помощи драйвера последовательного канала, работающего по опросу, – это реализация так называемого “эха”: со стороны персонального компьютера передаются символы контроллеру SDK-1.1, на которые контроллер отвечает определенным образом (см. вариант задания). Вторая задача выполняется при помощи драйвера последовательного канала,

реализованного в режиме прерываний с буферизацией байтов в буфере FIFO (см. [51], IOS2003_lab4.pdf). Такой задачей является реализация устройства, которое выполняет одну арифметическую операцию над десятичными числами, или конвертора из одной системы счисления в другую. Переключение между двумя задачами в программе должно быть выполнено с использованием DIP-переключателей (лабораторная работа № 1).

Рекомендации к выполнению работы:

1. Написать драйвер последовательного канала по опросу и простую тестовую программу для него.
2. Написать драйвер последовательного канала, осуществляющий асинхронный обмен данными по прерыванию, и отладить при помощи уже написанной тестовой программы (см. п. 1).
3. Написать тестовую программу для разработанных драйверов, которая выполняет две задачи в соответствии с вариантом задания.

Б.3.7 Требования к выполнению работы

1. В тестовой программе должна быть продемонстрирована работа с последовательным интерфейсом и по опросу, и асинхронно по прерыванию, т.е. должны быть реализованы и использованы драйверы UART двух видов для решения двух разных задач, указанных в варианте задания.
2. Разрабатываемые драйверы устройств должны быть выполнены в виде отдельных программных модулей (файлов), содержащих функции по работе с заданным одним устройством.
3. Должен быть предусмотрен контроль ввода корректных значений, выполнения арифметических действий в рамках второй задачи программы.
4. В программе должны быть использованы механизмы взаимного исключения (см. [51], IOS2003_lab4.pdf).
5. Текст программы должен соответствовать правилам оформления программ на языке Си, приведенным в приложении (Приложение Г. Требования к оформлению программ на языке Си, [95]).

Б.3.8 Содержание отчета

1. Титульный лист.
2. Номер варианта, задание.
3. Иллюстрация организации и функционирования разработанного программного обеспечения (драйверы, тестовая программа) в виде блок-схемы, диаграммы процессов, потоков данных, диаграммы состояний автоматов и других способов структурного и поведенческого описания программы (по выбору студента).
4. Исходный текст программы с комментариями.
5. Основные результаты.

Б.3.9 Литература

Литература к лабораторной работе: [37], [38], [50], [51], [54], [75], [88], [3].

Б.3.10 Варианты заданий

Каждый вариант состоит из двух частей: первая выполняется с использованием драйвера последовательного канала по опросу, вторая – по прерыванию. Переключение между двумя частями в тестовой программе должно быть выполнено с использованием DIP-переключателей (лабораторная работа № 1).

1. Скорость последовательного канала – 9600 бит/с.

   - Каждый принятый по последовательному каналу символ (от персонального компьютера к SDK-1.1) передается в утроенном виде в обратную сторону (от SDK-1.1 к персональному компьютеру) и отображается в терминальной программе. Причем все символы русского алфавита отображаются в нижнем регистре, все символы английского алфавита – в верхнем регистре. Например, на символ ‘л’ (‘Л’) ответом является ‘ллл’, ‘1’ – ‘111’, ‘i’ (‘I’) – ‘III’ и т.д.

   - Конвертор из десятичной в двоичную систему счисления. Диапазон преобразуемых значений – от 010 до 25510 включительно. 8-разрядная сетка для отображения двоичных чисел. Контроллеру SDK-1.1 по последовательному каналу со стороны персонального компьютера с использованием терминальной программы передается десятичное число для конвертирования, причем число это отображается в терминале, а концом ввода является перевод на следующую строку (<CR><LF>). После чего контроллер возвращает результат преобразования числа в двоичную систему счисления, который отображается в терминале персонального компьютера и на светодиодных индикаторах стенда SDK-1.1 (лабораторная работа № 1). Каждое новое преобразование начинается с новой строки. Сигнализация в случае ввода некорректных значений – сообщение об ошибке в последовательный канал.

2. Скорость последовательного канала – 9600 бит/с.

   - Со стороны персонального компьютера с использованием терминальной программы контроллеру SDK-1.1 по последовательному каналу передается любой символ русского алфавита в любом регистре (‘а’, ‘Б’, ‘в’,…, ‘Я’). В ответ контроллер SDK-1.1 передает принятый символ в верхнем регистре и 5 символов русского алфавита, следующих за введенным, в нижнем регистре. Все остальные вводимые символы игнорируются контроллером SDK-1.1. Например, на символ ‘к’ (или ‘К’) ответом является ‘Клмноп’, ‘у’ (‘У’) – ‘Уфхцчш’, ‘э’ (‘Э’) – ‘Эюя’ и т.д. Каждому обмену данными между персональным компьютером и стендом SDK-1.1 назначается отдельная строка.

   - Сумматор десятичных чисел. Диапазон значений слагаемых – от 010 до 9910 включительно. Контроллеру SDK-1.1 по последовательному каналу со стороны персонального компьютера с использованием терминальной программы передаются два слагаемых (десятичные числа), причем разделителем слагаемых является символ сложения (‘+’), концом ввода является символ равенства (‘=‘), получившееся выражение отображается в терминале персонального компьютера. После чего контроллер возвращает результат сложения, который отображается в терминале. Каждое новое выражение начинается с новой строки. Сигнализация в случае ввода некорректных значений – сообщение об ошибке в последовательный канал.

3. Скорость последовательного канала – 4800 бит/с.

- Со стороны персонального компьютера с использованием терминальной программы контроллеру SDK-1.1 по последовательному каналу передается любой символ английского алфавита в нижнем регистре (‘a’, ‘b’, ‘c’,…, ‘z’). В ответ контроллер SDK-1.1 передает принятый символ в нижнем регистре и 5 символов английского алфавита, следующих за введенным, в верхнем регистре. Все остальные вводимые символы игнорируются контроллером SDK-1.1. Например, на символ ‘u’ ответом является ‘uVWXYZ’, ‘m’ – ‘mNOPQRS’, ‘w’ – ‘wXYZ’ и т.д. Каждому обмену данными между персональным компьютером и стендом SDK-1.1 назначается отдельная строка.
- Конвертор из шестнадцатеричной в двоичную систему счисления. Диапазон преобразуемых значений – от 016 до FF16 включительно. 8-разрядная сетка для отображения двоичных чисел. Контроллеру SDK-1.1 по последовательному каналу со стороны персонального компьютера с использованием терминальной программы передается шестнадцатеричное число для конвертирования, причем число это отображается в терминале, а концом ввода является перевод на следующую строку (<CR><LF>). После чего контроллер возвращает результат преобразования числа в двоичную систему счисления, который отображается в терминале персонального компьютера и на светодиодных индикаторах стенда SDK-1.1 (лабораторная работа № 1). Каждое новое преобразование начинается с новой строки. Сигнализация в случае ввода некорректных значений – сообщение об ошибке в последовательный канал.

4. Скорость последовательного канала – 4800 бит/с.
   - На каждый принятый по последовательному каналу символ (от персонального компьютера к SDK-1.1) в ответ передается этот же символ и 2 следующих за ним символа согласно таблице ASCII (от SDK-1.1 к персональному компьютеру) и отображается в терминальной программе. Причем все символы русского алфавита отображаются в верхнем регистре, все символы английского алфавита – в нижнем регистре. Например, на символ ‘л’ (‘Л’) ответом является ‘ЛМН’, ‘1’ – ‘123’, ‘i’ (‘I’) – ‘ijk’ и т.д.
   - Вычитатель десятичных чисел. Диапазон значений уменьшаемого и вычитаемого – от 010 до 9910 включительно. Разность может быть как положительной, так и отрицательной. Контроллеру SDK-1.1 по последовательному каналу со стороны персонального компьютера с использованием терминальной программы передаются уменьшаемое и вычитаемое (десятичные числа), причем разделителем введенных значений является символ вычитания (‘-’), концом ввода является символ равенства (‘=‘), получившееся выражение отображается в терминале персонального компьютера. После чего контроллер возвращает результат операции, который отображается в терминале. Каждое новое выражение начинается с новой строки. Сигнализация в случае ввода некорректных значений – сообщение об ошибке в последовательный канал.
5. Скорость последовательного канала – 1200 бит/с.
   - Со стороны персонального компьютера с использованием терминальной программы контроллеру SDK-1.1 по последовательному каналу передается любой числовой символ (‘0’, ‘1’, ‘2’,…, ‘9’). В ответ контроллер SDK-1.1 передает принятый символ и все остальные числовые символы, следующие за введенным. Все остальные вводимые символы игнорируются контроллером SDK-1.1. Например, на символ ‘4’ ответом является ‘456789’, ‘8’ – ‘89’, ‘1’ – ‘123456789’ и т.д. Каждому обмену данными между персональным компьютером и стендом SDK-1.1 назначается отдельная строка.

- Конвертор из десятичной в шестнадцатеричную систему счисления. Диапазон преобразуемых значений – от 010 до 25510 включительно. Контроллеру SDK-1.1 по последовательному каналу со стороны персонального компьютера с использованием терминальной программы передается десятичное число для конвертирования, причем число это отображается в терминале, а концом ввода является перевод на следующую строку (<CR><LF>). После чего контроллер возвращает результат преобразования числа в шестнадцатеричную систему счисления, который отображается в терминале персонального компьютера. Каждое новое преобразование начинается с новой строки. Сигнализация в случае ввода некорректных значений – сообщение об ошибке в последовательный канал и зажигание светодиодов (лабораторная работа № 1).

6. Скорость последовательного канала – 1200 бит/с.

   - На каждый принятый по последовательному каналу символ (от персонального компьютера к SDK-1.1) в ответ передается этот же символ и 2 предшествующих ему символа согласно таблице ASCII (от SDK-1.1 к персональному компьютеру) и отображается в терминальной программе. Причем все символы русского алфавита отображаются в верхнем регистре, все символы английского алфавита – в нижнем регистре. Например, на символ ‘л’ (‘Л’) ответом является ‘ЛКЙ’, ‘5’ – ‘543’, ‘i’ (‘I’) – ‘ihg’ и т.д.

   - Умножитель десятичных чисел. Диапазон значений множителей – от 010 до 9910 включительно. Контроллеру SDK-1.1 по последовательному каналу со стороны персонального компьютера с использованием терминальной программы передаются множители (десятичные числа), причем разделителем введенных значений является символ умножения (‘*’), концом ввода является символ равенства (‘=‘), получившееся выражение отображается в терминале персонального компьютера. После чего контроллер возвращает результат операции, который отображается в терминале. Каждое новое выражение начинается с новой строки. Сигнализация в случае ввода некорректных значений – сообщение об ошибке в последовательный канал и зажигание светодиодов (лабораторная работа № 1).

7. Скорость последовательного канала – 2400 бит/с.

   - Со стороны персонального компьютера с использованием терминальной программы контроллеру SDK-1.1 по последовательному каналу передается любой символ английского алфавита в нижнем регистре (‘a’, ‘b’, ‘c’,…, ‘z’). В ответ контроллер SDK-1.1 передает принятый символ в нижнем регистре и все символы английского алфавита, предшествующие введенному, в верхнем регистре. Все остальные вводимые символы игнорируются контроллером SDK-1.1. Например, на символ ‘f’ ответом является ‘fEDCBA’, ‘j’ – ‘jIHGFEDCBA’, ‘p’ – ‘pONMLKJIHGFEDCBA’ и т.д. Каждому обмену данными между персональным компьютером и стендом SDK-1.1 назначается отдельная строка.

   - Конвертор из шестнадцатеричной в десятичную систему счисления. Диапазон преобразуемых значений – от 016 до FF16 включительно. Контроллеру SDK-1.1 по последовательному каналу со стороны персонального компьютера с использованием терминальной программы передается шестнадцатеричное число для конвертирования, причем число это отображается в терминале, а концом ввода является перевод на следующую строку (<CR><LF>). После чего контроллер возвращает результат преобразования числа в десятичную систему счисления, который отображается в терминале персонального компьютера. Каждое новое преобразование начинается с новой строки. Сигнализация в случае ввода

некорректных значений – сообщение об ошибке в последовательный канал и зажигание светодиодов (лабораторная работа № 1).

8. Скорость последовательного канала – 2400 бит/с.

   - Со стороны персонального компьютера с использованием терминальной программы контроллеру SDK-1.1 по последовательному каналу передается любой числовой символ (‘0’, ‘1’, ‘2’,…, ‘9’). В ответ контроллер SDK-1.1 передает принятый символ и все остальные числовые символы, предшествующие введенному. Все остальные вводимые символы игнорируются контроллером SDK-1.1. Например, на символ ‘4’ ответом является ‘43210’, ‘8’ – ‘876543210’, ‘1’ – ‘10’ и т.д. Каждому обмену данными между персональным компьютером и стендом SDK-1.1 назначается отдельная строка.
   - Целочисленный делитель десятичных чисел. Диапазон значений делимого и делителя – от 010 до 9910 включительно. Контроллеру SDK-1.1 по последовательному каналу со стороны персонального компьютера с использованием терминальной программы передаются делимое и делитель (десятичные числа), причем разделителем введенных значений является символ деления (‘/’), концом ввода является символ равенства (‘=‘), получившееся выражение отображается в терминале персонального компьютера. После чего контроллер возвращает частное, которое отображается в терминале. Каждое новое выражение начинается с новой строки. Сигнализация в случае ввода некорректных значений – сообщение об ошибке в последовательный канал и зажигание светодиодов (лабораторная работа № 1).

Б.4 Лабораторная работа № 4 «Клавиатура»

Б.4.1 Задание

Разработать и написать драйвер клавиатуры для учебно-лабораторного стенда SDK-1.1. Написать тестовую программу для разработанного драйвера, которая выполняет определенную вариантом задачу.

Б.4.2 Матричная клавиатура

Нередко во встраиваемой технике предусмотрен ввод данных с использованием кнопок, переключателей или других контактных групп. Но простое подключение контактных групп к линиям ввода/вывода микроконтроллера может породить проблему нехватки этих самых линий, если таких контактных групп много. Решение проблемы есть – это использование матричной клавиатуры.

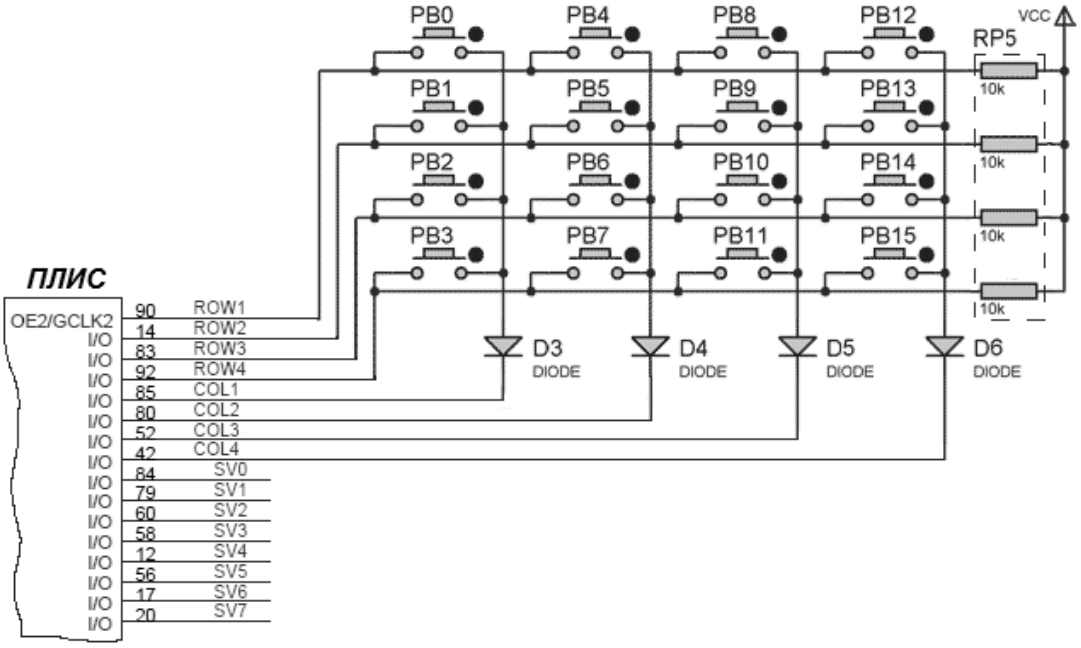


Рис. 134. Схема матричной клавиатуры контроллера SDK-1.1

Клавиатура в контроллере SDK-1.1 подключена к микроконтроллеру ADuC812 через расширитель портов ввода-вывода (ПЛИС). Схема клавиатурной матрицы представлена на рис. 134. Кнопки включены таким образом, что при нажатии кнопка замыкает строку на столбец. Из схемы видно, что часть линий ПЛИС используется в качестве сканирующих (столбцы), а часть в качестве считывающих (строки). Количество кнопок, подключенных таким образом, определяется как количество сканирующих линий, умноженное на количество считывающих. Отсюда следует, что использование матричной клавиатуры для случая, когда кнопок меньше или равно четырем, не имеет смысла, так как понадобятся те же четыре линии, а схема и программа усложнятся. Доступ к столбцам и строкам клавиатуры организован как

чтение/запись регистра ПЛИС (адрес 0x080000): младшие 4 бита соответствуют 4 столбцам (COL1..COL4), старшие 4 бита – строкам (ROW1..ROW4).

Как это работает? Линии сканирующего порта (столбцы) по умолчанию находятся в состоянии, когда на всех линиях, кроме одной, установлен высокий логический уровень. Линия, на которой установлен низкий логический уровень, является опрашиваемой в текущий момент, т.е. определяет опрашиваемый столбец. Если какая-либо кнопка этого столбца будет нажата, на соответствующей линии считывающего порта (строке) также будет низкий логический уровень, потому что замкнутая кнопка подтянет строку к потенциалу столбца, т.е. к земле. Если известен номер опрашиваемого столбца и номера линий считывающего порта, на которых установлен логический «0», можно однозначно определить, какие кнопки этого столбца нажаты.

Далее выбирается следующий опрашиваемый столбец путем установки логического «0» на соответствующей линии сканирующего порта и со считывающего порта снова снимаются данные. Цикл сканирования продолжается до тех пор, пока не будут перебраны таким образом все сканирующие линии. Традиционным решением является помещение процедуры опроса клавиатуры в обработчик прерывания от таймера. В контроллере SDK-1.1 есть еще другая возможность. В полной конфигурации ПЛИС работа с клавиатурой может быть организована по прерыванию: 6-й бит (KB) регистра ENA заведен на вход внешнего прерывания INT0 МК ADuC812. Внешнее прерывание будет возникать, как только на клавиатуре будет нажата хоть одна клавиша (любая).

Для случая, когда одновременно нажато несколько кнопок одного столбца все понятно. Будет установлено в логический «0» несколько битов считывающего порта одновременно. Но что произойдет, если будут замкнуты контакты нескольких кнопок из разных столбцов одной строки? Ведь в разных столбцах могут оказаться разные напряжения, так как на всех столбцах, кроме одного, логическая «1». Одновременное нажатие двух кнопок в одной строке привело бы к короткому замыканию и выжженным портам, если бы не диоды D3-D6 (рис. 134, рис. 135). Именно они защищают порты от короткого замыкания.

Резисторы RP5 на схеме являются подтягивающими. Если используемый микроконтроллер (в нашем случае используется ПЛИС) имеет в своем составе подтягивающие резисторы, от них можно отказаться.

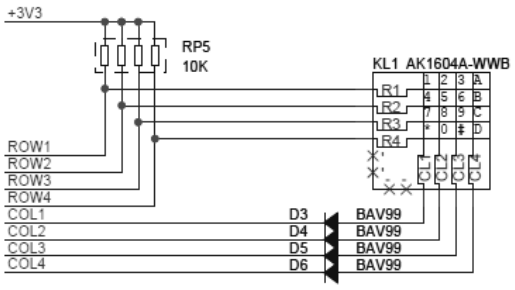


Рис. 135. Матричная клавиатура на принципиальной электрической схеме SDK-1.1

При работе с механическими кнопками возникает одна проблема – дребезг контактов. Суть его в том, что при нажатии на кнопку напряжение не сразу устанавливается на уровне 0В, а «скачет» в течение некоторого времени (десятки миллисекунд), пока цепь надежно не замкнется. После того, как кнопка будет отпущена, напряжение также «скачет», пока не установится на уровне логической «1» (рис. 136). Такое многократное замыкание/размыкание контактов вызвано тем, что контакты пружинят, обгорают и тому подобное.

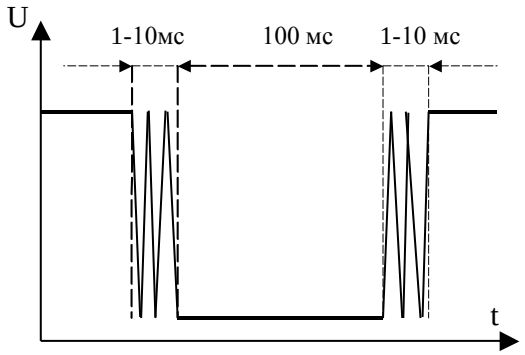


Рис. 136. Эффект дребезга контактов

Поскольку процессор обладает высоким быстродействием, то он может воспринять эти скачки напряжения за несколько нажатий. Решить эту проблему можно как аппаратно с помощью RS-триггера, триггера Шмитта, так и программно посредством небольшой задержки перед следующим опросом кнопки. Задержка подбирается такой, чтобы дребезг успел прекратиться к ее окончанию.

У приведенной схемы клавиатурной матрицы есть недостаток – эффект фантомного нажатия кнопки, который проявляется следующим образом: при нажатии определенной группы кнопок ПЛИС (микроконтроллер) считает еще несколько кнопок нажатыми, хотя никто этого не делал. На рис. 137 синими квадратами обозначены линии, на которых присутствует низкий логический уровень. Соответственно, красными квадратами обозначены линии с высоким логическим уровнем. В замкнутом состоянии находятся кнопки PB1, PB5 и PB6. В текущий момент сканируется столбец COL1, так как на выводе 85 ПЛИС присутствует низкий логический уровень, а на всех остальных сканирующих линиях высокий. В столбце COL1 в замкнутом состоянии находится только одна кнопка – PB1, и она подтягивает всю строку ROW2 к земле. Прочитав низкий логический уровень на линии ROW2, микроконтроллер через ПЛИС определит, что кнопка PB1 нажата, и это нормально. Однако в строке ROW2 есть еще одна нажатая кнопка – PB5. Она расположена в столбце COL2, а это значит, что и весь этот столбец окажется подтянутым к нулю. Сам по себе этот факт был бы не страшен, если бы в столбце COL2 не оказалась нажатой еще одна кнопка – PB6. Вот тут-то и начинаются неприятности. Эта кнопка подтянет к земле строку ROW3. Микроконтроллер через ПЛИС, прочитав со строки ROW3 низкий логический уровень, распознает кнопку PB2 нажатой (напомню, идет сканирование столбца COL1), а ее никто не нажимал.

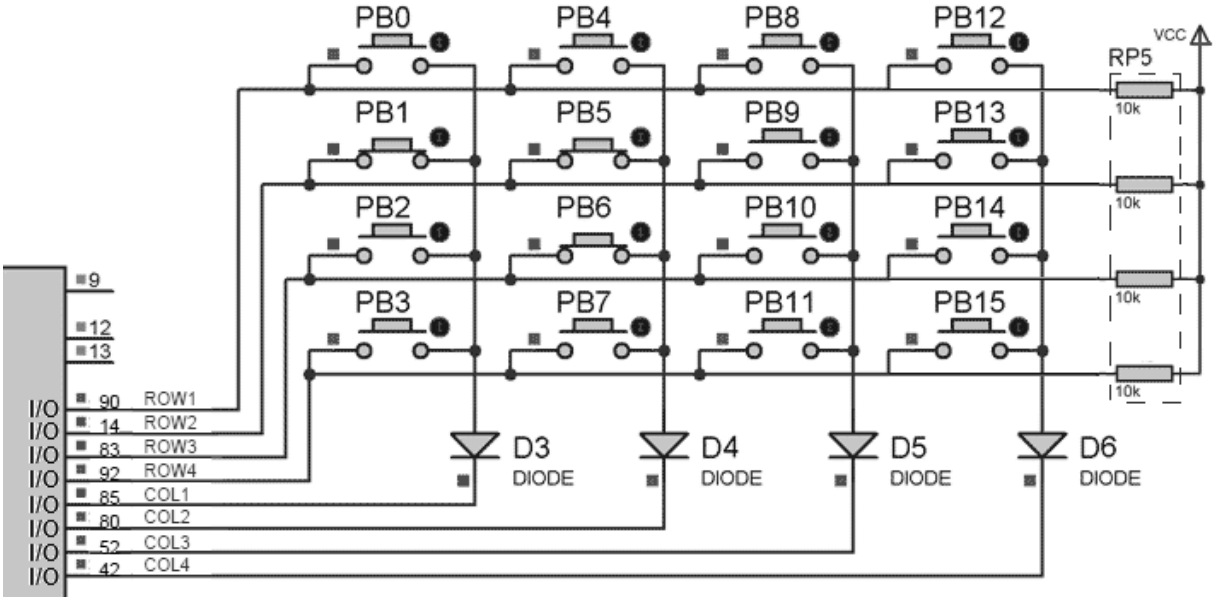


Рис. 137. Эффект фантомного нажатия кнопки

Итак, проблема проявляется при следующем условии: нажатыми должны быть минимум три кнопки, первая из которых находится в сканируемом столбце, вторая в том же ряду, что и первая, а третья в том же столбце, что и вторая.

Основным решением проблемы представляется ограничение на количество одновременно нажатых кнопок. Безопасным количеством являются две кнопки. Если нажато более двух кнопок, можно всячески предупреждать об этом пользователя (визуальная и/или звуковая сигнализация).

Второе решение проблемы вытекает из условия проявления проблемы. Ведь не любое сочетание клавиш приводит к фантомному нажатию. Можно реализовать алгоритм, проверяющий условие, при котором эта проблема проявляется, и в этом случае игнорировать ввод. Но вот нужна ли такая клавиатура, в которой не все комбинации клавиш допустимы, это вопрос конкретной разработки.

Если у нас кнопок не просто много, а очень много, то даже матрицирование не спасает от огромного расхода линий порта микроконтроллера. Тут приходится либо жертвовать несколькими портами, либо вводить дополнительную логику, например, дешифратор с инверсным выходом. Дешифратор принимает на вход двоичный код, а на выходе выдает «1» в выбранный разряд, а у инверсного дешифратора будет «0» (рис. 138).

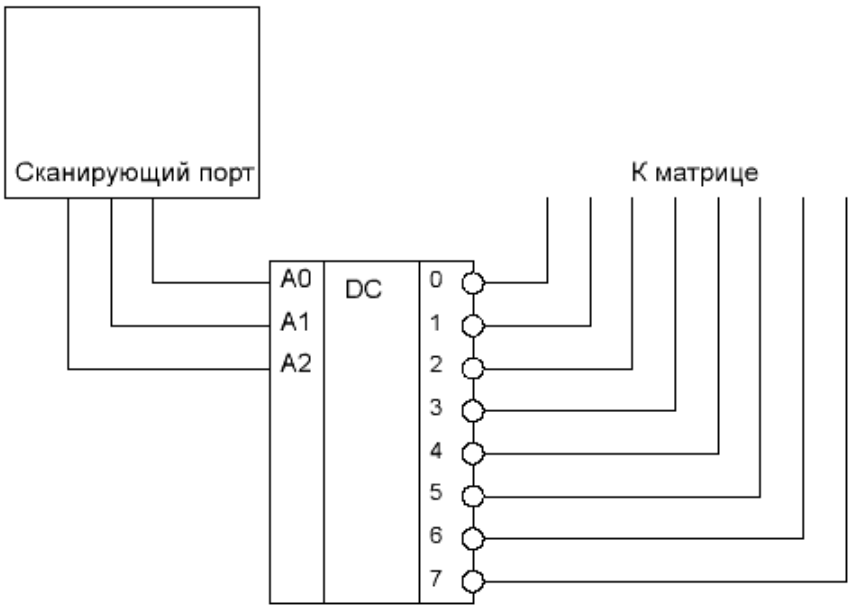


Рис. 138. Расширение разрядности клавиатурной матрицы дешифратором

Можно пойти еще дальше и поставить микросхему счетчика, который инкрементируется от импульсов с порта микроконтроллера. Значение же со счетчика прогоняется через дешифратор (рис. 139). Таким образом, можно заметно увеличить количество кнопок, только хватило бы разрядности дешифратора. Главное – учитывать на каком такте счетчика будет какой столбец.

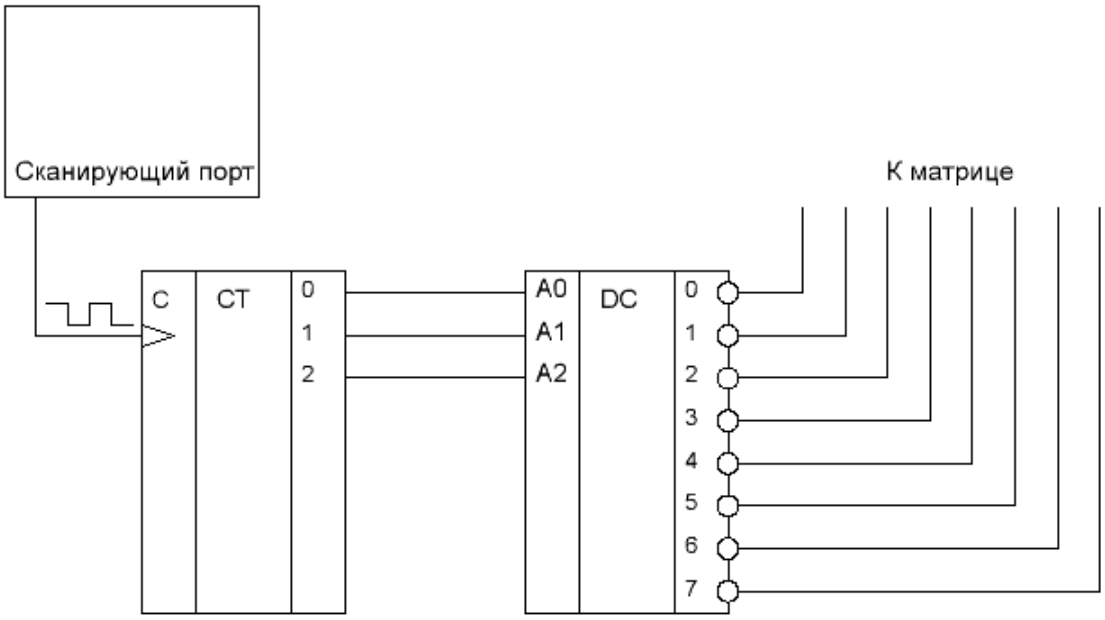


Рис. 139. Расширение разрядности клавиатурной матрицы счетчиком и дешифратором

Б.4.3 Описание работы

Данная лабораторная работа посвящена изучению клавиатуры в составе контроллера SDK-1.1.

В рамках лабораторной работы необходимо разработать программу для контроллера SDK-1.1 (ведомый), который обменивается данными с персональным компьютером (ведущий). В качестве канала связи используется последовательный канал RS-232. На стороне персонального компьютера имеется инструментальное средство для обеспечения взаимодействия с SDK-1.1 – это терминальная программа МЗР. Вариант задания такой же, как и в лабораторной работе № 3, с некоторыми изменениями, главным из которых является замена устройства ввода данных: вместо клавиатуры персонального компьютера используется клавиатура SDK-1.1. Устройством вывода является терминал персонального компьютера и светодиодные индикаторы SDK-1.1.

Программа должна выполнять две задачи. Первая задача – это тестирование клавиатуры (написанного драйвера): символы всех нажимаемых на клавиатуре SDK-1.1 кнопок выводятся в последовательный канал и отображаются в терминале. Вторая задача – выполнение варианта задания. Переключение между двумя задачами в программе должно быть выполнено с использованием DIP-переключателей (лабораторная работа № 1).

Драйвер клавиатуры должен работать по прерыванию от таймера, т.е. сканирование должно осуществляться в обработчике прерывания от таймера. В связи с этим необходимо следить, чтобы время выполнения обработчика не превышало времени между соседними прерываниями. Иначе это приведет к повторному входу в обработчик прерывания – последствия могут быть непредсказуемыми. Поэтому рекомендуется в каждом вызове обработчика опрашивать только один столбец клавиатуры, а не все сразу, так как увеличение времени выполнения обработчика приведет к уменьшению времени выполнения основной программы: может оказаться, что процессор большую часть времени будет занят обработкой прерываний от таймера со сканированием клавиатуры.

Драйвер клавиатуры должен содержать: функцию инициализации, функции сканирования клавиатуры и обработки нажатия кнопок (вызываются в обработчике прерывания от таймера), циклический буфер клавиатуры (нажатые кнопки), API-функцию чтения символа из буфера клавиатуры (см. [51], IOS2003_lab5.pdf). Взаимодействие обработчика и API-функции осуществляется только через буфер. Кроме того, работа с клавиатурой должна быть организована с переповторами, т.е. с отслеживанием длительного нажатия кнопки (как на клавиатуре персонального компьютера). Что это значит? Если после фиксирования нажатой кнопки (соответствующий символ занесен в буфер клавиатуры) проходит время, равное задержке перед повтором символа, а она все еще нажата, то в буфер клавиатуры повторно заносится этот же символ. После этого через промежутки времени, определяемые скоростью повтора символа, код кнопки заносится в буфер до тех пор, пока не будет зафиксировано

отпускание кнопки. При инициализации необходимо указать задержку перед повтором символа (первый параметр) и скорость повтора символа (второй параметр). Рекомендуемая величина задержки перед повтором – 1 секунда (не меньше).

Алгоритм опроса матричной клавиатуры рекомендуется реализовать в виде конечного автомата (Finite State Machine) – функции, которая в зависимости от своего состояния (значения определенной переменной) и входного воздействия, выполняет разную работу [30, 85, 96].

Драйвер клавиатуры должен адекватно обрабатывать одновременное нажатие нескольких кнопок.

Каждое нажатие кнопки на клавиатуре должно сопровождаться коротким звуковым сигналом, что требует использования драйвера звукового излучателя (генерация звука реализуется после попадания символа в буфер клавиатуры). Драйвер звукового излучателя должен работать по прерыванию от таймера. Длительность генерации звука – десятки миллисекунд. Таким образом, получается «музыкальная клавиатура».

Кроме того, должен быть предусмотрен контроль ввода корректных значений в рамках второй задачи программы (лабораторная работа № 3). Сигнализация в случае ввода некорректных значений – сообщение об ошибке в последовательный канал и зажигание светодиодов.

Рассмотрим применение конечных автоматов на более простом примере работы с одной клавишей без переповторов. Драйвер такой клавиши содержит:

- Функцию инициализации, в которой производится инициализация портов ввода-вывода микроконтроллера, к которым подключена клавиша; определение начальных значений буфера клавиатуры, счетчика нажатий клавиши и т.д.
- Функции опроса клавиши и определения ее нажатия/отпускания, которые вызываются в обработчике прерывания от таймера (рис. 140, а).
- Циклический буфер клавиатуры, в котором сохраняются сообщения о том, что клавиша нажата или не нажата.
- API-функцию чтения символа из буфера клавиатуры.

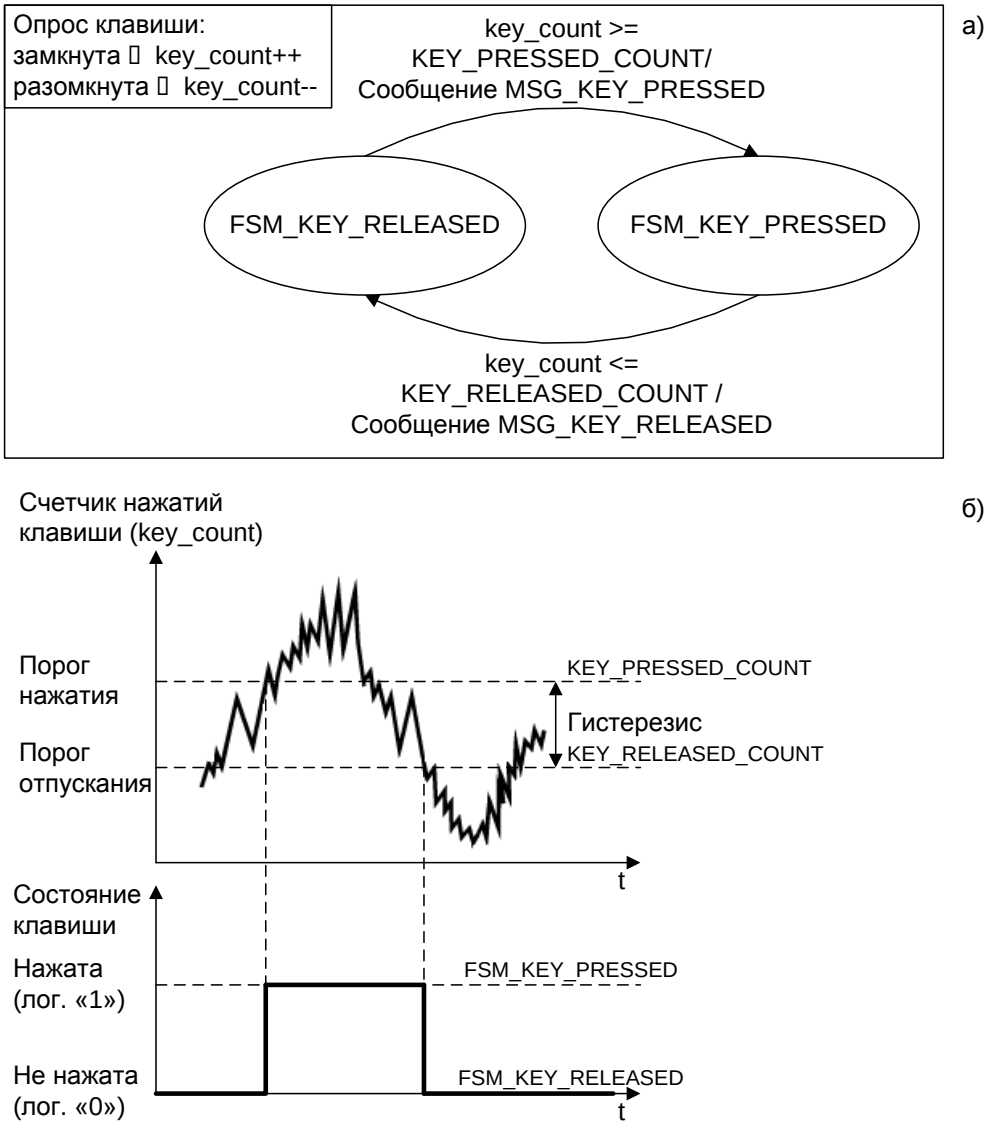


Рис. 140. Граф конечного автомата определения: нажата или отпущена клавиша? (а), временная диаграмма работы автомата (б)

Клавиша может находиться только в двух состояниях (рис. 140, а): клавиша нажата (FSM_KEY_PRESSED) и клавиша не нажата (FSM_KEY_RELEASED). Сначала в обработчике прерывания от таймера выполняется опрос клавиши на предмет ее замкнутости/разомкнутости. В процессе этого опроса счетчик нажатий клавиши key_count инкрементируется или декрементируется соответственно. Далее в виде конечного автомата реализуется алгоритм определения нажатия/отпускания клавиши на основе значения счетчика нажатий клавиши: если значение key_count превышает порог нажатия клавиши KEY_PRESSED_COUNT, то фиксируется ее нажатие (записывается сообщение MSG_KEY_PRESSED в буфер клавиатуры); если значение key_count падает ниже порога отпускания клавиши KEY_RELEASED_COUNT, то фиксируется ее отпускание (записывается сообщение MSG_KEY_RELEASED в буфер клавиатуры). Таким образом используется свойство гистерезиса: любой шум (дребезг), любые помехи с амплитудой, меньшей величины (KEY_PRESSED_COUNT − KEY_RELEASED_COUNT), отсекаются, а

состояние нажатой клавиши резко отличается от состояния отпущенной клавиши в плане определения и фиксации (рис. 140, б).

Данный принцип организации работы с одной клавишей можно перенести на набор клавиш, и в том числе, матричную клавиатуру. Особенностью такой схемы работы является то, что пороговые значения счетчика нажатий клавиши (KEY_PRESSED_COUNT и KEY_RELEASED_COUNT) прямо пропорциональны частоте прерываний от таймера, поэтому могут легко настраиваться при повторном использовании драйвера клавиатуры в других программных продуктах. Кроме того, данная схема абсолютно симметрично обрабатывает дребезг при нажатии и отпускании клавиши, а свойство гистерезиса позволяет регулировать чувствительность клавиатуры. Если ввести новые пороговые значения счетчика нажатий клавиши key_count, то можно организовать работу с клавиатурой с переповторами.

Б.4.4 Требования к выполнению работы

1. Разрабатываемые драйверы устройств должны быть выполнены в виде отдельных программных модулей (файлов), содержащих функции по работе с заданным одним устройством.
2. Переключение между двумя задачами в тестовой программе должно быть выполнено с использованием DIP-переключателей.
3. В тестовой программе должна быть продемонстрирована работа с клавиатурой и последовательным интерфейсом по прерыванию.
4. Работа с клавиатурой должна быть организована с переповторами, т.е. с отслеживанием длительного нажатия кнопки (как на клавиатуре персонального компьютера).
5. Драйвер клавиатуры должен адекватно обрабатывать одновременное нажатие нескольких кнопок.
6. Должен быть предусмотрен контроль ввода корректных значений в рамках выполнения прикладной задачи.
7. В программе должны быть использованы механизмы взаимного исключения (см. [51], IOS2003_lab4.pdf).
8. Текст программы должен соответствовать правилам оформления программ на языке Си, приведенным в приложении (Приложение Г. Требования к оформлению программ на языке Си, [95]).

Б.4.5 Содержание отчета

1. Титульный лист.
2. Номер варианта, задание.
3. Иллюстрация организации и функционирования разработанного программного обеспечения (драйверы, тестовая программа) в виде блок-схемы, диаграммы процессов, потоков данных, диаграммы состояний автоматов и других способов структурного и поведенческого описания программы (по выбору студента).
4. Исходный текст программы с комментариями.

5. Основные результаты.

Б.4.6 Литература

Литература к лабораторной работе: [38], [50], [51], [57], [58], [59], [75], [85], [88], [96], [3], [8], [9], [21].

Б.4.7 Варианты заданий

В случае установки на DIP-переключателях заданной комбинации (определяется студентом) контроллер SDK-1.1 входит в режим тестирования клавиатуры. В остальных случаях выполняется задача в соответствии с вариантом задания.

1. Скорость последовательного канала – 9600 бит/с.

   Конвертор из десятичной в двоичную систему счисления. Диапазон преобразуемых значений – от 0_{10} до 255_{10} включительно. 8-разрядная сетка для отображения двоичных чисел. При помощи клавиатуры SDK-1.1 вводится десятичное число для конвертирования. Кнопка перевода в двоичную систему счисления – «*». Вводимые с клавиатуры числа и результат должны выводиться в последовательный канал и отображаться в терминале персонального компьютера. Каждое новое преобразование начинается с новой строки. Должен быть предусмотрен контроль ввода корректных значений.

2. Скорость последовательного канала – 9600 бит/с.

   Сумматор десятичных чисел. Диапазон значений слагаемых – от 0_{10} до 99_{10} включительно. При помощи клавиатуры SDK-1.1 вводятся два слагаемых (десятичные числа), причем разделителем слагаемых является символ «+» (кнопка «A» на клавиатуре), символом начала вычисления – «=» (кнопка «#»). Вводимые с клавиатуры числа и результат должны выводиться в последовательный канал и отображаться в терминале персонального компьютера. Каждое новое выражение начинается с новой строки. Должен быть предусмотрен контроль ввода корректных значений.

3. Скорость последовательного канала – 4800 бит/с.

   Конвертор из шестнадцатеричной в двоичную систему счисления. Диапазон преобразуемых значений – от 0_{16} до FF_{16} включительно. 8-разрядная сетка для отображения двоичных чисел. При помощи клавиатуры SDK-1.1 вводится шестнадцатеричное число для конвертирования. Кнопка перевода в двоичную систему счисления – «#». Вводимые с клавиатуры числа и результат должны выводиться в последовательный канал и отображаться в терминале персонального компьютера. Каждое новое преобразование начинается с новой строки. Должен быть предусмотрен контроль ввода корректных значений.

4. Скорость последовательного канала – 4800 бит/с.

   Вычитатель десятичных чисел. Диапазон значений уменьшаемого и вычитаемого – от 0_{10} до 99_{10} включительно. Разность может быть как положительной, так и отрицательной. При помощи клавиатуры SDK-1.1 вводятся уменьшаемое и вычитаемое (десятичные числа), причем разделителем введенных значений является символ «-» (кнопка «B» на клавиатуре), символом начала вычисления – «=» (кнопка «#»). Вводимые с клавиатуры числа и результат должны выводиться в последовательный канал и отображаться в терминале персонального компьютера. Каждое новое выражение начинается с новой строки. Должен быть предусмотрен контроль ввода корректных значений.

5. Скорость последовательного канала – 1200 бит/с.

   Конвертор из десятичной в шестнадцатеричную систему счисления. Диапазон преобразуемых значений – от 0_{10} до 255_{10} включительно. При помощи клавиатуры SDK-1.1 вводится десятичное число для конвертирования. Кнопка перевода в шестнадцатеричную систему счисления – «*». Вводимые с клавиатуры числа и результат должны выводиться в последовательный канал и отображаться в терминале персонального компьютера. Каждое новое преобразование начинается с новой строки. Должен быть предусмотрен контроль ввода корректных значений.

6. Скорость последовательного канала – 1200 бит/с.

   Умножитель десятичных чисел. Диапазон значений множителей – от 0_{10} до 99_{10} включительно. При помощи клавиатуры SDK-1.1 вводятся множители (десятичные числа), причем разделителем введенных значений является символ «*» (кнопка «C» на клавиатуре), символом начала вычисления – «=» (кнопка «#»). Вводимые с клавиатуры числа и результат должны выводиться в последовательный канал и отображаться в терминале персонального компьютера. Каждое новое выражение начинается с новой строки. Должен быть предусмотрен контроль ввода корректных значений.

7. Скорость последовательного канала – 2400 бит/с.

   Конвертор из шестнадцатеричной в десятичную систему счисления. Диапазон преобразуемых значений – от 0_{16} до FF_{16} включительно. При помощи клавиатуры SDK-1.1 вводится шестнадцатеричное число для конвертирования. Кнопка перевода в десятичную систему счисления – «#». Вводимые с клавиатуры числа и результат должны выводиться в последовательный канал и отображаться в терминале персонального компьютера. Каждое новое преобразование начинается с новой строки. Должен быть предусмотрен контроль ввода корректных значений.

8. Скорость последовательного канала – 2400 бит/с.

   Целочисленный делитель десятичных чисел. Диапазон значений делимого и делителя – от 0_{10} до 99_{10} включительно. При помощи клавиатуры SDK-1.1 вводятся делимое и делитель (десятичные числа), причем разделителем введенных значений является символ «/» (кнопка «D» на клавиатуре), символом начала вычисления – «=» (кнопка «#»). Вводимые с клавиатуры числа и результат должны выводиться в последовательный канал и отображаться в терминале персонального компьютера. Каждое новое выражение начинается с новой строки. Должен быть предусмотрен контроль ввода корректных значений.

Б.5 Лабораторная работа № 5 «Жидкокристаллический индикатор»

Б.5.1 Задание

Разработать и написать драйвер жидкокристаллического индикатора (ЖКИ) для учебно-лабораторного стенда SDK-1.1. Написать программу для разработанного драйвера, которая выполняет определенную вариантом прикладную задачу.

Б.5.2 Описание работы

ЖКИ в контроллере SDK-1.1 подключен не напрямую к микроконтроллеру ADuC812, а через расширитель портов ввода-вывода, выполненный на базе ПЛИС. В ЖКИ есть специальный контроллер, формирующий необходимые напряжения на входах матрицы и осуществляющий динамическую индикацию. Для работы с этим контроллером реализован простейший интерфейс, описанный ниже (рис. 141, табл. 18). Матрица имеет 80 входов по горизонтали и 16 входов по вертикали[3].

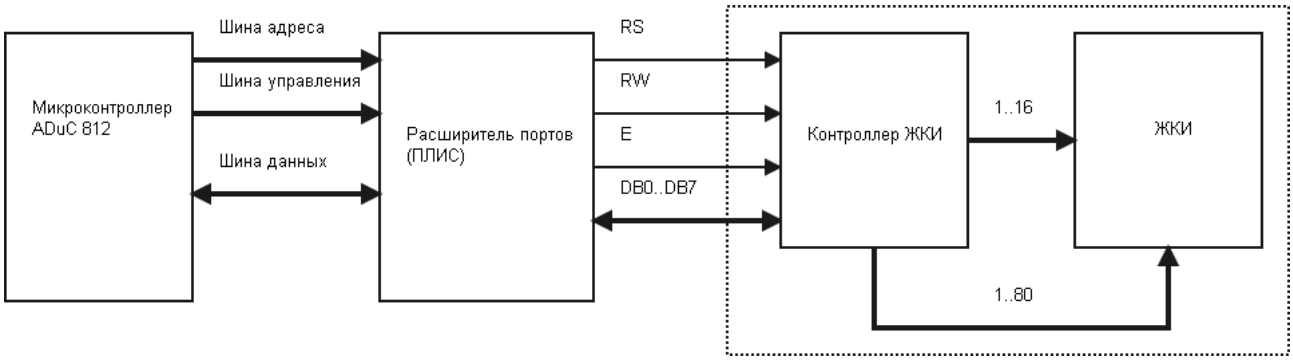


Рис. 141. Схема подключения контроллера ЖКИ к МК ADuC812

Матрица ЖКИ состоит из 32 знакомест (2 строки по 16 символов) размером 5 точек по горизонтали и 8 точек по вертикали. Для отображения различных символов внутри контроллера ЖКИ есть знакогенератор[4].

Таблица 18. Интерфейс ЖКИ

Обозначение	Описание
RS	Переключение между регистрами команд и данных: 1 – данные, 0 – команды
R/W	1– чтение (из контроллера ЖКИ), 0 – запись (в контроллер ЖКИ)
E	Разрешающий сигнал (1 – активный уровень). Если сигнал E = 0, то контроллер ЖКИ игнорирует все остальные сигналы
DB0	Бит данных 0

[3] Чтобы уменьшить количество проводников используется динамическая индикация. Принцип работы динамической индикации аналогичен принципу, используемому при опросе клавиатуры учебного стенда.

[4] Знакогенератор – специальное устройство, содержащее в себе ПЗУ (или ОЗУ) с битовыми картами с изображениями различных символов. Каждому изображению символа ставится в соответствие его код.

DB1	Бит данных 1
DB2	Бит данных 2
DB3	Бит данных 3
DB4	Бит данных 4
DB5	Бит данных 5
DB6	Бит данных 6
DB7	Бит данных 7

Основными компонентами контроллера ЖКИ являются память DDRAM (Data Display RAM), память CGRAM (Character Generator RAM), память CGROM (Character Generator ROM), счетчик адреса, регистр команд IR (Instruction Register), регистр данных DR (Data Register). Регистр команд предназначен для записи в него таких операций, как очистка дисплея, перемещение курсора, включение/выключение дисплея, а также установка адреса памяти DDRAM и CGRAM для последующего их выполнения. Регистр данных временно хранит данные, предназначенные для записи или чтения из DDRAM или CGRAM (символы). Эти два регистра можно выбрать с помощью регистрового переключателя RS (Register Select).

Работать с ЖКИ достаточно просто. За связь с ЖКИ в расширителе портов ввода-вывода отвечают два регистра:

1. DATA_IND (адрес 0x080001) отвечает за выдачу информации на шину данных (через этот регистр можно передавать команды контроллеру и данные).
2. C_IND (адрес 0x080006) отвечает за формирование сигналов E, R/W и RS, позволяющих регулировать обмен на шине между расширителем портов и контроллером ЖКИ.

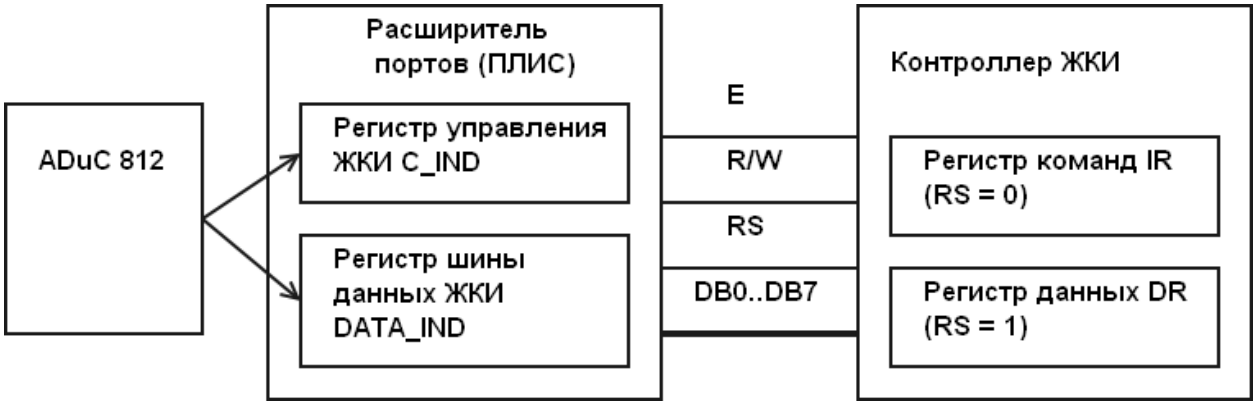


Рис. 142. Регистры, необходимые для работы с ЖКИ

На рис. 142 регистры расширителя портов изображены слева, регистры контроллера ЖКИ справа. Для доступа к регистрам контроллера ЖКИ вы должны сформировать на шине сигналы с помощью регистров расширителя портов.

Вся работа с индикатором сводится к нескольким простым вещам:

1. Первым шагом вы записываете команду или данные (коды выводимых символов) в регистр DATA_IND расширителя портов. После этого, содержимое этого регистра появляется на шине данных контроллера ЖКИ (DB0-DB7). Контроллер на эти данные, естественно, не реагирует, так как сигнал E (Enable) нами еще не выставлен в активный уровень (логическая «1»).
2. Вторым шагом вы должны разрешить работу с шиной с помощью сигнала E (логическая «1»), выставить сигнал записи (логический «0» на линии W) и указать тип регистра, с которым вы будете работать в контроллере ЖКИ на линии RS. Если вы передаёте данные, то на сигнал RS нужно подать «1», если команду, то «0».

Необходимо помнить, что в учебном стенде SDK-1.1 начальная инициализация контроллера ЖКИ уже выполнена в загрузчике. В реальной системе вам придется программировать ее самостоятельно.

Время выполнения команд контроллером ЖКИ не равно нулю, и это нужно учитывать в своем драйвере: опрашивать флаг готовности ЖКИ (BF). Таким образом, функции работы с ЖКИ НЕ нужно вызывать в обработчиках прерывания других периферийных устройств (например, таймера).

Данная лабораторная работа посвящена изучению жидкокристаллического индикатора (ЖКИ) стенда SDK-1.1.

В рамках лабораторной работы необходимо разработать программу для контроллера SDK-1.1, которая выполняет конкретную прикладную задачу (см. варианты задания). Реализация задачи требует знания материалов предыдущих лабораторных работ: таймеры микроконтроллера ADuC812, последовательный канал, светодиодные индикаторы, клавиатура и др.

Драйвер ЖКИ должен включать следующие функции:

Функция	**Описание**
`void InitLCD(void)`	Инициализация ЖКИ.
`void WriteControlLCD( unsigned char ch)`	Запись значения в регистр управления ЖКИ C_IND (ПЛИС): `ch` – значение, записываемое в C_IND.
`bit ReadBFLCD(void)`	Чтение флага BF (флаг занятости контроллера ЖКИ).
`unsigned char ClearLCD(void)`	Очистка дисплея с возвратом результата выполнения операции.
`unsigned char GotoXYLCD (unsigned char x, bit y)`	Переход в заданную позицию дисплея с возвратом результата выполнения операции: `x, y` – координаты позиции.
`unsigned char PrintCharLCD`	Вывод символа на дисплей с возвратом

`(unsigned char symbol)`	результата выполнения операции: `symbol` – выводимый символ.

Кроме того, может быть реализована функция вывода строки на ЖКИ, функция дополнительной настройки ЖКИ (отображение, мерцание курсора).

Драйвер клавиатуры использует прерывание таймера, в котором производится опрос состояния кнопок (сканирование клавиатуры). В данном случае можно применить автоматное программирование. В драйвер клавиатуры рекомендуется включить функцию чтения нажатых кнопок из буфера, связывающего ее с обработчиком прерывания таймера. Реакции на нажатия кнопок клавиатуры должны формироваться в главной программе. Вся обработка нажатий кнопок не должна быть локализована в обработчике прерываний таймера.

Драйвер таймера должен включать следующие функции (помимо обработчика прерывания):

Функция	Описание
`void InitTimer(void)`	Инициализация таймера.
`unsigned long GetMsCounter(void)`	Получение текущей метки времени в миллисекундах.
`unsigned long DTimeMs(unsigned long t0)`	Измерение количества миллисекунд, прошедших с временной метки `t0` и до текущего времени.
`void DelayMs(unsigned long t)`	Задержка на `t` миллисекунд.

Кроме того, могут быть реализованы функции работы с таймером в режиме «счетчик» (например, чтение счетчика).

Работа с последовательным каналом (приемопередатчиком UART) должна быть организована асинхронно по прерыванию. Драйвер последовательного канала включает следующие функции (помимо обработчика прерывания):

Функция	Описание
`void InitSerial(void)`	Инициализация последовательного канала.
`unsigned char WriteSerial(unsigned char data_buf)`	Передача байта данных `data_buf` с возвратом результата выполнения операции.
`unsigned char ReadSerial(unsigned char* data_buf)`	Прием байта данных `*data_buf` с возвратом результата выполнения операции.
`unsigned char StatusSerial(void)`	Чтение признака наличия байта в буфере приема.

Кроме того, может быть реализована функция вывода строки в последовательный канал.

Б.5.3 Требования к выполнению работы

1. Разрабатываемые драйверы устройств должны быть выполнены в виде отдельных программных модулей (файлов), содержащих функции по работе с заданным одним устройством.
2. На уровне драйверов (особенно обработчиков прерываний) НЕ рекомендуется смешивать работу с несколькими периферийными устройствами (например, в обработчике прерывания таймера выводить строку на ЖКИ, опрашивать DIP-переключатели и т.д.). Взаимодействие устройств ввода-вывода следует организовать на прикладном уровне с использованием API-функций их драйверов.
3. Должен быть предусмотрен контроль ввода корректных значений в рамках выполнения прикладной задачи.
4. В программе должны быть использованы механизмы взаимного исключения (см. [51], IOS2003_lab4.pdf).
5. Текст программы должен соответствовать правилам оформления программ на языке Си, приведенным в приложении (Приложение Г. Требования к оформлению программ на языке Си, [95]).

Б.5.4 Содержание отчета

1. Титульный лист.
2. Номер варианта, задание.
3. Модель написанной программы (см. Приложение В. Проектирование и разработка программы).
4. Разработанные протоколы, форматы данных и др.
5. Исходный текст программы с комментариями (можно не весь, но обязательно главная программа и полностью драйвер периферийного устройства, изучению которого была посвящена лабораторная работа).
6. Основные результаты.

Б.5.5 Литература

Литература к лабораторной работе: [50], [51], [57], [75], [80], [82], [88], [9], [21], [25].

Б.5.6 Варианты заданий

1. Секундомер.

Написать программу, реализующую функции электронного секундомера. Точность измерения времени – сотые доли секунды. В качестве устройства, измеряющего время, следует использовать один из внутренних таймеров микроконтроллера ADuC812. Управление секундомером должно осуществляться с клавиатуры стенда SDK-1.1:

- кнопка «*» («старт/пауза») – запускает процесс измерения времени либо приостанавливает его, не сбрасывая;
- кнопка «#» («сброс») – сбрасывает замеряемое время в ноль.

На ЖКИ должна отображаться четко следующая информация: замеряемое время (слева в верхней строке); минимальное из всех замеренных времен (слева в нижней строке); максимальное из всех замеренных времен (справа в нижней строке). Формат отображения времени: «SS:CC», где SS – секунды, CC – сотые доли секунды. После переполнения секундомер начинает отсчет с нуля, т.е. 99,99с → 0с. Замеренный интервал времени по нажатию кнопки «пауза» должен выводиться в последовательный канал в формате, описанном ранее. Каждый интервал с новой строки. Сообщения о сбросе и переполнении секундомера тоже должны выводиться в последовательный канал (формат этих сообщений определяется студентом).

В рамках задания необходимо реализовать:

- драйвер таймера;
- драйвер последовательного канала;
- драйвер клавиатуры;
- драйвер ЖКИ.

2. Калькулятор.

Написать программу, реализующую функции простого калькулятора. Управление калькулятором должно осуществляться с клавиатуры стенда SDK-1.1:

- кнопки «0 – 9» – ввод операндов;
- кнопка «A» – операция сложения;
- кнопка «B» – операция вычитания;
- кнопка «C» – операция умножения;
- кнопка «D» – операция деления;
- кнопка «*» – начало процесса вычисления;
- кнопка «#» – сброс. На ЖКИ должна выводиться следующая информация: вычисляемое выражение (например, «25+75», затем, после нажатия кнопки «*» – «25+75=100»).

Разрядная сетка операндов и результатов определяется студентом, однако должна содержать не меньше 2 десятичных разрядов и обладать возможностью легкого увеличения/уменьшения. После нажатия кнопки сброса, а также в начале работы калькулятора на индикаторе должен отображаться ноль. Каждое вычисленное выражение (по нажатию кнопки «*») должно выводиться в последовательный канал в формате, описанном ранее, и начинаться с новой строки.

В рамках задания необходимо реализовать:

- драйвер последовательного канала;
- драйвер клавиатуры;
- драйвер ЖКИ.

3. Текстовый редактор.

Написать программу, реализующую функции простого текстового редактора. Ввод текста должен осуществляться с клавиатуры стенда SDK-1.1 кнопками «0 – 9». Таким образом, текст может состоять только из цифровых символов. Объем вводимого текста не должен превышать 8 строк по 16 символов, при этом ЖКИ отображает только две соседние строки. Номер текущей строки должен отображаться на светодиодном индикаторе (первая строка – первый светодиод, вторая строка – второй светодиод и т.д.). Перемещение по тексту осуществляется с помощью курсора, управляемого клавишами «A» (влево), «B» (вправо), «C» (вверх) и «D» (вниз). Ввод текста производится в позицию, указываемую курсором, при этом все символы находящиеся правее введенного символа сдвигаются, курсор перемещается (как в обычном текстовом редакторе). В качестве клавиши перевода строки выступает кнопка «*», удаление символов должно производиться кнопкой «#».

В рамках задания необходимо реализовать:

- драйвер светодиодных индикаторов;
- драйвер клавиатуры;
- драйвер ЖКИ.

4. Бегущая строка.

Написать программу, реализующую эффект «бегущей строки». Эффект заключается в постепенном смещении отображаемой на ЖКИ строки по кругу (вправо или влево). Управление «бегущей строкой» должно осуществляться с клавиатуры стенда SDK-1.1:

- кнопка «A» – смена направления движения текста;
- кнопка «B» – смена строки ЖКИ, по которой двигается («бежит») текст, причем смена должна производиться немедленно;
- кнопки «C» и «D» – изменяют скорость движения текста (диапазон значений: 1 – 10 позиций в секунду).

Для формирования скорости движения строки обязательным является использование таймера микроконтроллера ADuC812. Движение текстовой строки реализуется таким образом, чтобы исчезающая из области видимости часть текста появлялась постепенно с противоположной стороны строки ЖКИ. Кроме того, отображение «бегущей строки» должно быть плавным, четким, без образованных движением текста «хвостов».

По нажатию кнопки «*» программа переходит в режим ввода новой текстовой строки. Ввод производится с помощью кнопок «0 – 9». Таким образом, вводимый текст может состоять только из цифровых символов. При вводе должна контролироваться длина текста (не менее 1 и не более 16 символов). Завершение ввода строки производится кнопкой «*». Отмена ввода – кнопка «#». Исполнение программы должно начинаться с прокручивания в верхней строке ЖКИ слева направо текста «SDK-1.1».

В рамках задания необходимо реализовать:

- драйвер таймера;
- драйвер клавиатуры;
- драйвер ЖКИ.

5. Монитор состояний устройств.

Написать программу, реализующую монитор состояний устройств стенда SDK-1.1. Программа должна отражать состояние следующих трех устройств:

- DIP-переключатели (линии 0-7 дискретного параллельного порта ПЛИС): отображается состояние DIP-переключателей в двоичной системе счисления;
- таймер-счетчик: отображается количество перепадов на счетном входе T0 или T1, вызываемых замыканием соответствующих DIP-переключателей (см. рис. 129);
- таймер: отображается системное время, прошедшее с момента старта программы в миллисекундах.

Программа должна работать в двух режимах: автоматическом и ручном. Смена режима работы должна производиться нажатием кнопки «*» на клавиатуре стенда SDK-1.1. В автоматическом режиме отображение состояний устройств производится циклически с интервалом 3 секунды по умолчанию. Данный интервал должен изменяться с помощью кнопок «C» и «D». На ЖКИ состояние каждого устройства должно выглядеть следующим образом: в верхней строке – название устройства, в нижней строке – текущее состояние устройства.

В ручном режиме смена отображения состояния устройств должна производиться нажатием кнопки «#».

Кроме того, каждые 2 секунды (может быть другой приемлемый интервал времени) в последовательный канал должно выводиться текущее состояние всех устройств в виде одной строки, формат строки – свободный.

В рамках задания необходимо реализовать:

- драйвер DIP-переключателей;
- драйвер таймера/счетчика;
- драйвер последовательного канала;
- драйвер клавиатуры (может быть реализован без переповторов, т.е. по принципу «кнопка нажата или нет»);
- драйвер ЖКИ.

Б.6 Лабораторная работа № 6 «Последовательный интерфейс I²C»

Б.6.1 Задание

Разработать и написать драйверы интерфейса I²C и I²C-устройств учебно-лабораторного стенда SDK-1.1. Написать программу для разработанных драйверов, которая выполняет определенную вариантом прикладную задачу.

Б.6.2 Описание работы

В бытовой технике, телекоммуникационном оборудовании и промышленной электронике часто встречаются похожие решения в, казалось бы, никак не связанных изделиях. Например, практически каждая система включает в себя:

- Некоторый «умный» узел управления, обычно однокристальная микроЭВМ.
- Узлы общего назначения, такие как буферы ЖКИ, порты ввода-вывода, RAM, E2PROM или преобразователи данных.
- Специфические узлы, такие как схемы цифровой настройки и обработки сигнала для радио- и видеосистем, или генераторы тонального набора для телефонии.

Для того чтобы использовать эти общие решения с выгодой для конструкторов и производителей (технологов), а также увеличить эффективность аппаратуры и упростить схемотехнические решения, компания Philips в 1980 году разработала простую двунаправленную двухпроводную шину для эффективного «межмикросхемного» (inter-IC) управления. Шина так и называется – Inter-Integrated Circuit, или IIC (I²C) шина. В настоящее время ассортимент продукции Philips включает более 150 КМОП и биполярных I²C-совместимых устройств, функционально предназначенных для работы во всех трех вышеперечисленных категориях электронного оборудования. Все I²C-совместимые устройства имеют встроенный интерфейс, который позволяет им связываться друг с другом по шине I²C. Это конструкторское решение разрешает множество проблем сопряжения различных устройств, которые обычно возникают при разработке цифровых систем.

Основной режим работы шины I²C – 100 кбит/с; 10 кбит/с в режиме работы с пониженной скоростью. Заметим, что стандарт допускает тактирование с частотой вплоть до нулевой. Для адресации I²C-устройств используется 7 бит (1980 год).

Список возможных применений I²C:

- Доступ к модулям памяти (RAM, E2PROM, FLASH и др.).
- Доступ к низкоскоростным ЦАП/АЦП.

- Работа с часами реального времени (RTC).
- Регулировка контрастности, насыщенности и цветового баланса мониторов.
- Управление интеллектуальными звукоизлучателями (динамиками).
- Управление ЖКИ, в том числе в мобильных телефонах.
- Чтение информации с датчиков мониторинга и диагностики оборудования, например, термостат центрального процессора или датчик скорости вращения вентилятора охлаждения процессора.
- Информационный обмен между микроконтроллерами.

I²C использует две двунаправленные линии с открытым стоком: последовательная линия данных (SDA, англ. Serial DAta) и последовательная линия тактирования (SCL, англ. Serial CLock), обе нагруженные резисторами (см. рис. 143). Максимальное напряжение +5В, часто используется +3,3В, однако допускаются и другие напряжения (не менее +2В). Шина I²C поддерживает любую технологию изготовления микросхем (НМОП, КМОП, биполярную).

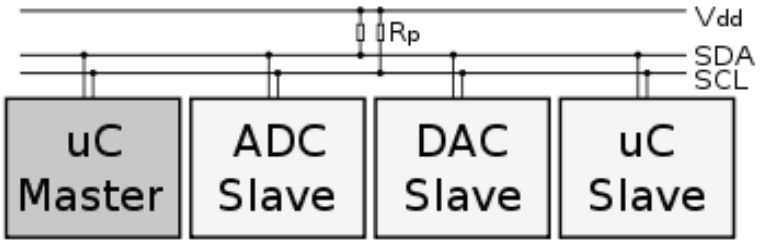


Рис. 143. Пример соединения устройств на шине I²C

Каждое устройство распознается по уникальному адресу, будь то микроконтроллер, ЖКИ-буфер, память или интерфейс клавиатуры, и может работать как передатчик или приёмник, в зависимости от назначения устройства. Обычно ЖКИ-буфер – только приёмник, а память может как принимать, так и передавать данные. Кроме того, устройства могут быть классифицированы как ведущие и ведомые при передаче данных. Ведущий – это устройство, которое инициирует передачу данных и вырабатывает сигналы синхронизации. При этом любое адресуемое устройство считается ведомым по отношению к ведущему. Классическая адресация включает 7-битное адресное пространство с 16 зарезервированными адресами (шина I²C 1980 года). Это означает до 112 свободных адресов для подключения периферии на одну шину.

Возможность подключения более одного микроконтроллера к шине означает, что более чем один ведущий может попытаться начать пересылку в один и тот же момент времени. Для устранения хаоса, который может возникнуть в данном случае, разработана процедура арбитража. Эта процедура основана на том, что все I²C-устройства подключаются к шине по правилу монтажного И.

Генерация синхросигнала – это всегда обязанность ведущего: каждый ведущий генерирует свой собственный сигнал синхронизации при пересылке данных по шине. Сигнал синхронизации может быть изменен, только если он «вытягивается» медленным ведомым устройством (путем удержания линии в низком состоянии), или другим ведущим в случае столкновения.

Данная лабораторная работа посвящена изучению последовательного интерфейса I^2C и устройств, подключенных по этому интерфейсу к микроконтроллеру ADuC812 стенда SDK-1.1, – энергонезависимая память EEPROM и часы реального времени (RTC).

В рамках лабораторной работы необходимо разработать программу для контроллера SDK-1.1, которая выполняет конкретную прикладную задачу (см. варианты задания). Реализация задачи требует знания материалов предыдущих лабораторных работ: таймеры микроконтроллера ADuC812, последовательный канал, светодиодные индикаторы, клавиатура, ЖКИ, звуковой излучатель и др.

В данной работе МК ADuC812 – ведущий, а EEPROM и RTC – ведомые на шине I^2C. К особенностям реализации контроллера последовательного двухпроводного интерфейса в ADuC812 можно отнести следующие: в режиме ведущего генерация сигналов на линиях данных и синхронизации является программной (через биты регистров специального назначения); в режиме ведущего (в отличие от ведомого) НЕ генерируются прерывания по приему/передаче данных. С учетом сказанного ниже приведены варианты реализации драйвера I^2C:

1. Простой вариант. В драйвере I2C реализуется синхронный обмен данными, который предполагает отсутствие ситуации неготовности I2C-устройств. Однако EEPROM и RTC не порты ввода-вывода и все-таки требуют определенное количество времени на выполнение циклов чтения/записи/стирания данных. При синхронной организации обмена эти задержки должна учитывать программа, т.е. драйвер I2C.
2. Усложненный вариант. Драйвер I2C должен использовать прерывание таймера, в котором работа с интерфейсом представляет собой периодический процесс генерации сигналов на линиях данных (SDA) и синхронизации (SCL) и организуется в виде конечного автомата. Состояниями данного автомата являются состояния шины I2C по передаче и приему данных (например, передача старт- или стоп-состояния, передача адреса ведомого устройства, передача или прием байта данных, передача или прием подтверждения и др.). Кроме того, необходимо отслеживать ошибочные состояния I2C и обрабатывать их. Частота настройки таймера определяет невысокую скорость передачи данных по шине I2C. Вся работа с интерфейсом НЕ должна быть локализована в обработчике прерываний таймера, так как процессом обмена данными необходимо управлять при помощи API-функций, которые взаимодействуют с обработчиком прерывания через буферы.

Независимо от варианта реализации драйвера I^2C он должен содержать такие API-функции, как инициализация I^2C, прием и передача блока данных и т.д. Взаимодействие с I^2C-устройствами (EEPROM, RTC) при помощи такого драйвера может быть выполнено по опросу (проверка готовности ведомых устройств к обмену).

Драйвер EEPROM должен включать следующие функции:

Функция	Описание
`unsigned char ReadEEPROM (unsigned long addr, unsigned long size, unsigned char *buf)`	Чтение данных из EEPROM с возвратом результата выполнения операции: `addr` – адрес ячейки памяти, `size` – размер буфера для чтения, `buf` – буфер.
`unsigned char WriteEEPROM (unsigned long addr, unsigned long size, unsigned char *buf)`	Запись данных в EEPROM с возвратом результата выполнения операции: `addr` – адрес ячейки памяти, `size` – размер буфера записи, `buf` – буфер.

Драйвер часов реального времени должен включать следующие функции:

Функция	Описание
`void InitRTC(void)`	Инициализация часов реального времени.
`unsigned char ReadRTC (TimeDate *td)`	Чтение даты и времени из RTC с возвратом результата выполнения операции: `td` – буфер для даты и/или времени в виде структуры специального формата.
`unsigned char WriteRTC (TimeDate *td)`	Запись даты и времени в RTC с возвратом результата выполнения операции: `td` – буфер для даты и/или времени в виде структуры специального формата.

Кроме того, в драйвер могут входить функции для работы с будильником и для дополнительной настройки RTC.

Драйвер клавиатуры тоже использует прерывание таймера, в котором производится опрос состояния кнопок (лабораторная работа № 4 «Клавиатура»). В данном случае можно применить автоматное программирование. В драйвер клавиатуры рекомендуется включить функцию чтения нажатых кнопок из буфера, связывающего ее с обработчиком прерывания. Реакции на нажатия кнопок клавиатуры должны формироваться в

главной программе. Вся обработка нажатий кнопок НЕ должна быть локализована в обработчике прерываний таймера.

Драйвер ЖКИ должен включать следующие функции (лабораторная работа № 5):

Функция	Описание
`void InitLCD(void)`	Инициализация ЖКИ.
`void WriteControlLCD( unsigned char ch)`	Запись значения в регистр управления ЖКИ C_IND (ПЛИС): `ch` – значение, записываемое в C_IND.
`bit ReadBFLCD(void)`	Чтение флага BF (флаг занятости контроллера ЖКИ).
`unsigned char ClearLCD(void)`	Очистка дисплея с возвратом результата выполнения операции.
`unsigned char GotoXYLCD (unsigned char x, bit y)`	Переход в заданную позицию дисплея с возвратом результата выполнения операции: `x, y` – координаты позиции.
`unsigned char PrintCharLCD (unsigned char symbol)`	Вывод символа на дисплей с возвратом результата выполнения операции: `symbol` – выводимый символ.

Кроме того, может быть реализована функция вывода строки на ЖКИ, функция дополнительной настройки ЖКИ (отображение, мерцание курсора).

Драйвер таймера должен включать следующие функции помимо обработчика прерывания (лабораторная работа № 2):

Функция	Описание
`void InitTimer(void)`	Инициализация таймера.
`unsigned long GetMsCounter(void)`	Получение текущей метки времени в миллисекундах.
`unsigned long DTimeMs(unsigned long t0)`	Измерение количества миллисекунд, прошедших с временной метки `t0` и до текущего времени.
`void DelayMs(unsigned long t)`	Задержка на `t` миллисекунд.

Работа с последовательным каналом (приемопередатчиком UART) должна быть организована по прерыванию. Драйвер последовательного канал включает следующие функции помимо обработчика прерывания (лабораторная работа № 3):

Функция	Описание
`void InitSerial(void)`	Инициализация последовательного канала.
`unsigned char WriteSerial(unsigned char data_buf)`	Передача байта данных `data_buf` с возвратом результата выполнения операции.
`unsigned char ReadSerial(unsigned char* data_buf)`	Прием байта данных `*data_buf` с возвратом результата выполнения операции.
`unsigned char StatusSerial (void)`	Чтение признака наличия байта в буфере приема.

Кроме того, может быть реализована функция вывода строки в последовательный канал.

Б.6.3 Требования к выполнению работы

1. Разрабатываемые драйверы устройств должны быть выполнены в виде отдельных программных модулей (файлов), содержащих функции по работе с заданным одним устройством.
2. На уровне драйверов (особенно обработчиков прерываний) НЕ рекомендуется смешивать работу с несколькими периферийными устройствами (например, в обработчике прерывания таймера выводить строку на ЖКИ, опрашивать DIP-переключатели и т.д.). Взаимодействие устройств ввода-вывода следует организовать на прикладном уровне с использованием API-функций их драйверов.
3. Должен быть предусмотрен контроль ввода корректных значений в рамках выполнения прикладной задачи.
4. В программе должны быть использованы механизмы взаимного исключения (см. [51], IOS2003_lab4.pdf).
5. Текст программы должен соответствовать правилам оформления программ на языке Си, приведенным в приложении (Приложение Г. Требования к оформлению программ на языке Си, [95]).

Б.6.4 Содержание отчета

1. Титульный лист.
2. Номер варианта, задание.
3. Модель написанной программы (см. Приложение В. Проектирование и разработка программы).
4. Разработанные протоколы, форматы данных и др.
5. Исходный текст программы с комментариями (можно не весь, но обязательно главная программа и полностью драйвер периферийного устройства, изучению которого была посвящена лабораторная работа).
6. Основные результаты.

Б.6.5 Литература

Литература к лабораторной работе: [46], [50], [51], [88], [3], [13], [19], [23], [24], [25].

Б.6.6 Варианты заданий

1. Тестирование EEPROM.

Контроллер SDK-1.1 организует по последовательному каналу систему меню, по которому можно перемещаться с помощью символов, передаваемых со стороны персонального компьютера с использованием терминальной программы. Прием неправильного символа по последовательному каналу приводит к перерисовке меню. Предлагается следующий вариант меню (как оно может выглядеть):

```
Тест EEPROM
------------------------------------------------------
1 – Запись данных
2 – Чтение данных
3 – Очистка памяти
4 – Автоматический тест
------------------------------------------------------
```

Кроме того, аналогичная система меню должна быть организована и на ЖКИ стенда SDK-1.1. Перемещение по этому меню реализуется при помощи клавиатуры SDK-1.1 (кнопки управления по выбору студента). Указанные 4 пункта меню должны быть выполнены обязательно, оформление может быть иным и разным для терминала и ЖКИ, но главное – понятным и удобным для использования.

При запуске теста в терминал и на ЖКИ SDK-1.1 должно выводиться меню.

По выбору пункта меню «Запись данных» EEPROM заполняется последовательностью случайных чисел (127/255 байт[5]) и CRC8, рассчитанного по этой последовательности. Результат выполнения операции (OK/ERR_I2C/ERR_CRC), записанная последовательность и CRC8 в шестнадцатеричном формате выводятся в терминал. Пример вывода:

```
Запись данных [OK]
[FA 11 45 67 23 21 CC E2 99 23 CB B5 5A 2F 25 81
 1F 44 …
 …          …          …          …          …          ]
CRC8 [0x28]
```

Вывод записанных данных для удобочитаемости может быть выполнен в виде 8/16 строк по 16 значений.

На ЖКИ SDK-1.1 отображается результат выполнения операции («`Test EEPROM Write OK`»). По нажатию специальной кнопки на клавиатуре SDK-1.1 на ЖКИ снова выводится меню.

По выбору пункта меню «Чтение данных» из EEPROM считывается записанная последовательность и рассчитывается CRC8 (все 128/256 байт). Результат выполнения операции (OK/ERR_I2C/ERR_CRC), прочитанная последовательность и CRC8 в шестнадцатеричном формате выводятся в терминал. Пример вывода:

```
Чтение данных [OK]
[FA 11 45 67 23 21 CC E2 99 23 CB B5 5A 2F 25 81
```

[5] Емкость EEPROM определяется установленной микросхемой в стенде SDK-1.1.

```
 1F 44 …
 …          …          …          …          …          ]
CRC8 [0x28]
```

На ЖКИ SDK-1.1 отображается результат выполнения операции («`Test EEPROM Read OK`»). По нажатию специальной кнопки на клавиатуре SDK-1.1 на ЖКИ снова выводится меню.

По выбору пункта меню «Очистка памяти» выполняется стирание всей памяти EEPROM (заполнение 0xFF). Результат выполнения операции (OK/ERR_I2C/ERR_CRC) выводится в терминал. Пример вывода:

```
Очистка памяти [OK]
```

На ЖКИ SDK-1.1 отображается результат выполнения операции («`Test EEPROM Erase all OK`»). По нажатию специальной кнопки на клавиатуре SDK-1.1 на ЖКИ снова выводится меню.

По выбору пункта меню «Автоматический тест» выполняется запись и чтение данных EEPROM как в пунктах меню 1 и 2. Результат выполнения операции (OK/ERR_I2C/ERR_CRC) выводится в терминал. Пример вывода:

```
Запись данных [OK]
Чтение данных [OK]
```

На ЖКИ SDK-1.1 отображается результат выполнения операции («`Test EEPROM Write & Read OK`»). По нажатию специальной кнопки на клавиатуре SDK-1.1 на ЖКИ снова выводится меню.

Необходимо отметить, что ошибка в качестве результата выполнения операции может быть двух видов: ошибка обмена по каналу I^2C (ERR_I2C) и ошибка в циклическом коде (ERR_CRC). Первая должна отслеживаться на уровне драйвера I^2C, вторая – на уровне расчета CRC8.

В рамках задания необходимо реализовать:

- драйвер последовательного канала;
- драйвер клавиатуры (может быть реализован без переповторов, т.е. по принципу «кнопка нажата или нет»);
- драйвер ЖКИ;
- драйвер I2C;
- драйвер EEPROM;
- функцию генерации случайного (псевдослучайного) числа в диапазоне от 0 до 255 на основе таймера ADuC812;
- функцию расчета CRC8 с использованием полинома x8 + x5 + x4 + 1.

2. Журнал событий.

Журнал событий реализуется на основе EEPROM: в нем сохраняется хронологическая информация о действиях пользователя, производимых со стендом SDK-1.1. Таким действием является:

- Изменение состояния одного из 12 DIP-переключателей (SW3-1 и SW3-2 на рис. 126) с указанием номера и положения вкл./выкл.;
- Нажатие кнопки клавиатуры (одной из 16) с указанием символа.

На запись о произведенном действии в EEPROM выделяется 4 байта, причем формат записи следующий:

- 1 и 2 байт – временная (секундная) метка события, выставляемая по таймеру;
- 3 байт – код события;
- 4 байт – CRC8 по предыдущим трем байтам.

При запуске системы таймер (переменная, отвечающая за системное время) инициализируется значением последней временной метки, присутствующей в журнале событий, если таковая есть. Должны быть реализованы механизмы по определению начала журнала событий (для его вычитывания) и конца (для дополнения новой записью). Необходимо предусмотреть возможность изменения размера журнала событий, т.е. журнал может занимать не всю память EEPROM. Журнал (буфер) является циклическим, т.е. в случае записи события в последние 4 байта журнала, следующая запись производится в его первые 4 байта. Кроме того, нужно предусмотреть возможность стирания всей памяти EEPROM.

Реализация системы меню из двух пунктов – чтение журнала событий и очистка EEPROM – на стороне персонального компьютера (терминальная программа). Перемещение по этому меню реализуется при помощи клавиатуры ПК.

Необходимо реализовать возможность передачи содержимого журнала событий по последовательному каналу в персональный компьютер в любой момент. Отображение считанной информации в терминальной программе должно быть выполнено в формате: временная метка – событие. Каждому событию – новая строка. При чтении журнала событий проверка CRC8 для каждой записи является обязательной. В случае ошибки обмена по каналу I^2C, несовпадения циклического кода прочитанной из журнала записи и др. некорректных ситуаций сообщение об этом должно выводиться в терминал во время чтения журнала.

Последнее произведенное действие (в формате записи в журнале событий), сообщения о выполненных операциях (чтение журнала и очистка памяти), об ошибках в работе должны отображаться на ЖКИ контроллера SDK-1.1. После запуска системы на ЖКИ выводится последняя запись журнала событий.

В рамках задания необходимо реализовать:

- драйвер DIP-переключателей;
- драйвер таймера;
- драйвер последовательного канала;
- драйвер клавиатуры (может быть реализован без переповторов, т.е. по принципу «кнопка нажата или нет»);
- драйвер ЖКИ;
- драйвер I2C;
- драйвер EEPROM;
- функцию расчета CRC8 с использованием полинома x8 + x5 + x4 + 1.

3. Часы.

В задании используются часы реального времени. Необходимо на ЖКИ контроллера SDK-1.1 отображать текущую дату и время в формате «dd.mm.yyyy» на первой строке и «hh:mm:ss» – на второй строке. Введение новой даты и времени организовать с помощью клавиатуры стенда SDK-1.1. Во время редактирования текущее редактируемое знакоместо отображать курсором. Некорректный ввод должен быть блокирован. Для входа/выхода в/из режим(а) редактирования необходимо предусмотреть специальную клавишу (или комбинацию клавиш).

Кроме того, введение новой даты и времени нужно организовать и по последовательному каналу со стороны персонального компьютера с помощью терминальной программы. Формат даты и времени такой же, как и на ЖКИ. Дата и время вводятся вместе (в одной строке). Сообщение о некорректном вводе должно отображаться в терминале.

В рамках задания необходимо реализовать:

- драйвер последовательного канала;
- драйвер клавиатуры (может быть реализован без переповторов, т.е. по принципу «кнопка нажата или нет»);
- драйвер ЖКИ;
- драйвер I2C;
- драйвер часов реального времени (RTC).

4. Будильник.

На одной из строк ЖКИ контроллера SDK-1.1 отображается текущее время в формате «hh:mm:ss». На другой строке ЖКИ отображается время срабатывания будильника в таком же формате. Чтобы отличать одну строку от другой, можно использовать какой-нибудь специальный символ. Введение текущего времени и нового времени срабатывания будильника организовано по последовательному каналу со стороны персонального компьютера с помощью терминальной программы. Для этого нужно реализовать систему меню из соответствующих двух пунктов. Формат ввода времени такой же, как и на ЖКИ. Некорректный ввод должен быть блокирован.

При наступлении времени срабатывания будильника запускается проигрывание мелодии с помощью звукового пьезоизлучателя контроллера SDK-1.1. После нажатия кнопки клавиатуры контроллера SDK-1.1 сигнал будильника перестает звучать.

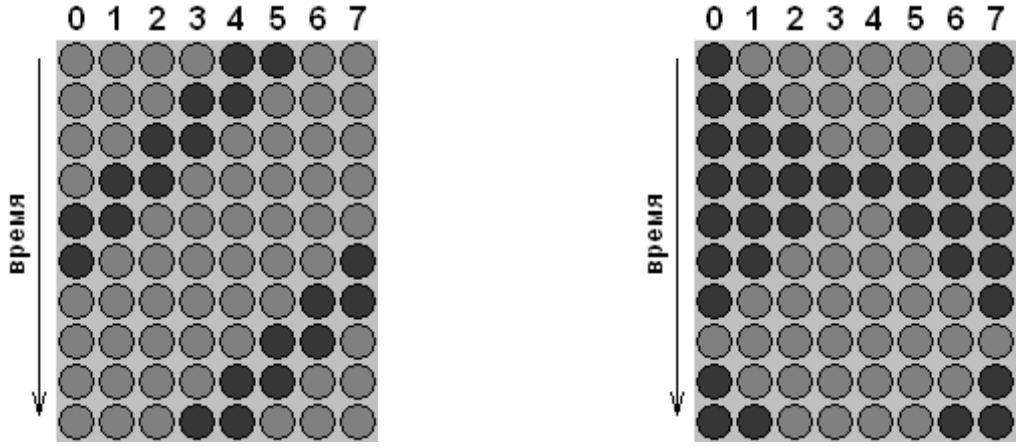
Следует уделить особое внимание тому, что функция будильника реализуется на основе часов реального времени.

В рамках задания необходимо реализовать:

- драйвер звукового излучателя на основе таймера (лабораторная работа № 2);
- драйвер последовательного канала;
- драйвер клавиатуры (может быть реализован без переповторов, т.е. по принципу «кнопка нажата или нет»);
- драйвер ЖКИ;
- драйвер I2C;
- драйвер часов реального времени (RTC).

5. Сохранение контекста.

На светодиодные индикаторы контроллера SDK-1.1 выводится либо одна, либо другая анимация.



A=1 A=2

Выбор анимации осуществляется при помощи специального параметра (например, A). Отображение первой анимации (A=1) задается с использованием трех параметров: количество зажженных светодиодов (например, N), направление движения зажженных светодиодов (например, D) и скорость движения (например, S). Отображение второй анимации (A=2) задается с использованием двух параметров: количество зажженных светодиодов (N) и скорость движения (S). Параметр S выражается в условных единицах (диапазон значений: 1 – 9), причем 1 – самая низкая скорость, 9 – самая высокая. Для формирования скорости движения анимации обязательным является использование таймера микроконтроллера ADuC812.

Все перечисленные параметры (A, N, D, S) являются контекстными, т.е. сохраняются в EEPROM и защищаются CRC8. Таким образом, после перезапуска системы эти параметры считываются и задают режим отображения анимаций. Если при чтении параметров CRC8 не сошелся, то они считаются некорректными и принимаются значения по умолчанию (табл. 19). Необходимо заметить, что параметры N и S имеют различные значения для одной и другой анимации (для каждой анимации свой контекст).

Таблица 19. Допустимые значения параметров анимаций

	A = 1			**A = 2**	
	N	D	S	N	S
Диапазон значений	1 – 7	Налево, направо	1 – 9	1 – 4 (относится к половинкам линейки светодиодов)	1 – 9
Значение по умолчанию (см. рисунки)	2	Налево	1	4	1

Кроме того, все перечисленные параметры отображаются на ЖКИ стенда SDK-1.1 и могут редактироваться при помощи клавиатуры SDK-1.1. На первой строке ЖКИ – выбранный тип анимации, на второй строке – параметры отображаемой анимации. Пример вывода на ЖКИ:

```
     A=1
N=3  D=←  S=5
```

Изменение параметра контекста должно тут же сохраняться в EEPROM и оказывать влияние на выводимую анимацию.

Необходимо реализовать функцию очистки EEPROM по нажатию кнопки клавиатуры SDK-1.1 (например, «#»).

Перечисленные функции по изменению параметров анимаций и очистке памяти EEPROM выполняются и при помощи системы меню, организованной в терминальной программе персонального компьютера (обмен данными по последовательному каналу).

В рамках задания необходимо реализовать:

- драйвер светодиодных индикаторов;
- драйвер таймера;
- драйвер последовательного канала;
- драйвер клавиатуры (может быть реализован без переповторов, т.е. по принципу «кнопка нажата или нет»);
- драйвер ЖКИ;
- драйвер I2C;
- драйвер EEPROM;
- функцию расчета CRC8 с использованием полинома x8 + x5 + x4 + 1.

6. Текстовый редактор с памятью.

Написать программу, реализующую функции простого текстового редактора на ЖКИ стенда SDK-1.1. Ввод текста должен осуществляться с клавиатуры контроллера SDK-1.1 и персонального компьютера (при помощи терминальной программы). Стенд SDK-1.1 подключен к ПК через коммуникационный кабель RS-232. На клавиатуре SDK-1.1 для ввода текста используются только кнопки «0 – 9», таким образом выводимая на ЖКИ информация является исключительно цифровой. Объем выводимого текста не должен превышать 8 строк по 16 символов, при этом ЖКИ отображает только две соседние строки. Номер текущей строки должен отображаться на светодиодном индикаторе (первая строка – первый светодиод, вторая строка – второй светодиод и т.д.).

На клавиатуре SDK-1.1 перемещение по тексту осуществляется с помощью кнопок «A» (влево), «B» (вправо), «C» (вверх) и «D» (вниз), текущая позиция выделяется курсором. Ввод текста производится в позицию, указываемую курсором, при этом все символы находящиеся правее введенного символа сдвигаются, курсор перемещается (как в обычном текстовом редакторе). В качестве клавиши перевода строки выступает кнопка «*», удаление символов должно производиться кнопкой «#».

На клавиатуре ПК реализуются аналогичные функции редактирования: перемещение вправо-влево и вверх-вниз (клавиши навигации), в начало и в конец текста (Home, End), удаление символа (Delete, Backspace).

Вводимый текст сохраняется в EEPROM и защищается CRC8. Таким образом, после перезапуска системы этот текст считывается и отображается на ЖКИ, готовый для редактирования. Если при чтении сохраненного текста из EEPROM CRC8 не сошелся, то он считается некорректным и принимается значение по умолчанию (на усмотрение студента).

В рамках задания необходимо реализовать:

- драйвер светодиодных индикаторов;
- драйвер последовательного канала;
- драйвер клавиатуры (может быть реализован без переповторов, т.е. по принципу «кнопка нажата или нет»);
- драйвер ЖКИ;
- драйвер I2C;
- драйвер EEPROM;
- функцию расчета CRC8 с использованием полинома x8 + x5 + x4 + 1.

Приложение В. Проектирование и разработка программы

Одним из основных компонентов отчета в лабораторных работах №№5-6 является модель написанной программы. Что под этим понимается?

Сложность задачи, которую необходимо решить в этих лабораторных работах, требует ознакомления с понятием процесса проектирования программного обеспечения. В упрощенном виде к основным этапам традиционного потока проектирования ПО можно отнести:

1. Разработку технического задания и спецификаций (в данном случае они уже представлены и предложены в виде варианта задания).
2. Архитектурное проектирование (по-другому, системное или концептуальное проектирование).
3. Разработку программного обеспечения (непосредственная реализация, т.е. кодирование).
4. Отладку и тестирование.
5. Создание рабочей документации (в данном случае это отчет).
6. Ввод в эксплуатацию, сопровождение (в данном случае это демонстрация преподавателю выполненной работы и ее защита).

Вопросам технологий проектирования программного обеспечения в частности и вычислительных систем в целом, проблемам в различных подходах к проектированию и разработке систем, так называемому «кризису сложности» в вычислительной технике посвящено множество статей и книг [28, 30, 40, 41, 42, 43, 49, 64, 69, 14, 15, 20, 25]. Однако недостаточное качество, многократно заваленные сроки выполнения, высокая стоимость до сих пор являются характерными чертами систем такого рода.

В данной работе Вам настоятельно рекомендуется не пропускать этап архитектурного проектирования, т.е. продумать, как Вы будете решать поставленную задачу, а потом уж приступать к кодированию придуманного. Что это Вам даст? Грамотное системное проектирование

- Существенно сократит время реализации ПО в частности и срок выполнения работы в целом (примерное временное соотношение этапа №2 и этапа №3 – «80% к 20%»).
- Сделает Вашу программу прозрачной, простой для понимания, предсказуемой, значит, легко отлаживаемой.
- Повысит качество.
- Сократит ресурсоемкость.
- Добавит в Вашу программу возможность наращивания функциональности, модифицируемости и дальнейшего развития (если, конечно, критерий повторного использования учитывался на этапе проектирования).
- Сделает более доступным анализ разработанного ПО.

- и т.д.

Этап архитектурного проектирования обычно тесно связан с вопросом описания разрабатываемой системы. Недостатком вербального описания программного обеспечения является низкая степень формализации, т.е. практическая невозможность математического доказательства правильности тех или иных утверждений, выраженных таким способом. Блок-схема алгоритма, применяющаяся для описания сравнительно несложных программ, уже не дает никакого эффекта в крупных проектах. Еще в работах Дейкстры и Вирта, а далее в работах Йордона, Росса и др. было предложено описывать программные системы в виде совокупности структурных и поведенческих составляющих. Система разрабатывалась на основе декомпозиции (разделения) общих сущностей на более частные. Была предложена так называемая «абстракция», т.е. выделение существенных для проектировщика деталей проекта и сокрытие второстепенных.

Что такое структура и поведение? Структура системы – это совокупность частей (элементов и подсистем) и связи между ними. Поведение системы – это изменение структурных составляющих (подсистем и элементов), а также связей между ними во времени.

Для описания двух этих понятий, люди издавна используют графические изображения. К сожалению, пока не существует способа изображения на одной картинке всей структуры или поведения вычислительной системы. Приходится рассматривать и структуру, и поведение с разных позиций.

Структура может быть представлена следующими способами:

- Совокупность блоков системы и интерфейсов (объект A соединен с объектом B с помощью интерфейса I2C).
- Совокупность объектов и зависимостей (объект «дом» зависит от объекта «электростанция»).
- Схема наследования классов (класс X происходит от классов Y и Z).
- Схема включения классов (агрегация, класс Q включает в себя объекты S и D).

Поведение можно представить в виде:

- Конечного автомата [96].
- Временной диаграммы (процессограммы, диаграммы взаимодействий и т.п.).
- Потоковой диаграммы (DFD, CFD и т.п.).

Эти перечни не претендуют на полноту. В книгах по проектированию вычислительных систем можно найти другие способы описания структуры и поведения или, так называемых, нотаций [30].

Требуемая в отчете модель программы – это и есть ее архитектурное описание. В случае прохождения этапа архитектурного проектирования представить эту модель не составляет труда.

Чтобы проанализировать и оценить логику работы программы (решения поставленной задачи), организацию работы с периферийными устройствами, алгоритмы управления в прикладной и системной части необходимо оторваться от строчек кода (абстрагироваться) и посмотреть на программу в целом, панорамно, по уровням. Поэтому Вам нужно выделить основные сущности Вашей программы и способы (каналы) их взаимодействия – соответственно получается модель первого уровня. Потом (на втором уровне) необходимо представить модели каждой из выделенных сущностей: чаще всего это может быть модель функционирования драйвера клавиатуры, последовательного канала, звукового излучателя, шины I^2C, основного прикладного алгоритма и т.д. с выделением потока данных и/или управления (команд). Например, в лабораторной работе № 4 «Клавиатура» для представления задачи сканирования клавиатуры (это лишь часть драйвера клавиатуры) прекрасно подходят конечные автоматы (FSM) [30, 85, 96]. Также могут быть применены сети процессов Кана, DFD и другие перечисленные ранее способы описания структуры и поведения системы. Рекомендуется ознакомиться с этими нотациями, но можно использовать и свой способ описания, – главное, чтобы он удовлетворял перечисленным выше требованиям.

Приложение Г. Требования к оформлению программ на языке Си

Г.1 Соглашения по идентификаторам

Г.1.1 Подбор идентификаторов

А. Все идентификаторы должны выбираться из соображений читаемости и максимальной семантической нагрузки.

Например:

```
const float Eps = 0.0001;      // точность
unsigned short Sum;            // сумма
unsigned char Message[ 20 ];   // сообщение
```

Неудачными можно считать идентификаторы:

```
const float UU = 0.0001;       // точность
unsigned short Kk;             // сумма
unsigned char Zz[ 20 ];        // сообщение
```

Б. Идентификаторы рекомендуется подбирать из слов английского языка.

Например:

```
// выдает звуковой сигнал заданной частоты и длительности
void Beep( unsigned short Hertz, unsigned short MSec );
// выдает True (1), если файл с именем FName существует
unsigned char ExistFile( unsigned char* FName );
// признак окончания работы с программой
unsigned char Done;
// размеры изделия (ширина, высота)
unsigned short Width, Height;
```

Не очень удачными можно считать идентификаторы:

```
// выдает звуковой сигнал заданной частоты и длительности
void Zvuk(unsigned short Chast, unsigned short Dlit );
// выдает True (1), если файл с именем Im существует
unsigned char EstFile(unsigned char* Im );
// признак окончания работы с программой
unsigned char Konec;
// размеры изделия (ширина, высота)
unsigned short Shirina, Vysota;
```

Г.1.2 Написание идентификаторов

Существует два основных способа написания идентификаторов.

А. В любых идентификаторах каждое слово, входящее в идентификатор, писать, начиная с большой буквы, остальные буквы – маленькие.

Например:

```
float NextX, LastX;      // следующая и предыдущая итерация
char BeepOnError;        // подавать ли звуковой сигнал при
                         // неправильном вводе пользователя?
unsigned char FileName[ 20 ];

// стандартная функция модуля Graph; выдает описание
```

```
// ошибки использования графики по ее коду
unsigned char* GraphErrorMsg( short ErrCode );
```

Б. В любых идентификаторах каждое слово, входящее в идентификатор, разделять символом “_”, при этом все буквы – маленькие.

Например:

```
float next_x, last_x;    // следующая и предыдущая итерация
char beep_on_error;      // подавать ли звуковой сигнал при
                         // неправильном вводе пользователя?
unsigned char file_name[ 20 ];

// стандартная функция модуля Graph; выдает описание
// ошибки использования графики по ее коду
unsigned char* graph_error_msg( short err_code );
```

Г.2 Соглашения по самодокументируемости программ

Г.2.1 Комментарии

А. Комментарии в теле программы следует писать на русском языке и по существу так, чтобы программист, не участвовавший в разработке программы (но имеющий опыт работы на языке Си), мог без особого труда разобраться в логике программы, и, при необходимости, сопровождать данный программный продукт.

Б. Рекомендуется комментарии к программе писать после символов //, а /* и */ использовать при отладке программы как "заглушки" участков программного кода.

Г.2.2 Спецификация функций

Для каждой пользовательской функции должна быть описана в виде комментария спецификация, содержащая следующую информацию [95]:

- Назначение функции;
- Описание семантики параметров-значений (параметров, передаваемых по значению).
- Описание семантики параметров-переменных (параметров, передаваемых по ссылке).
- Описание семантики возвращаемого значения.

Например:

```
////////////////////////Gauss//////////////////////////////
// Решение системы линейных алгебраических уравнений
// методом Гаусса.
// Вход:
//           A   - матрица коэффициентов системы;
//           B   - столбец свободных членов системы;
//           Eps - точность вычислений.
// Выход:
//           X                - вектор решения;
//           HasSolution      - флаг, устанавливаемый в True, если
```

```
//                                решение системы существует, и в False
//                                во всех остальных случаях;
//           NumOfRoots         - число корней в решении системы, может
//                                принимать значения:
//                                0         - если решение системы не
//                                            существует,
//                                MaxN      - если решение системы
//                                            существует и единственно,
//                                MaxInt    - если существует бесконечное
//                                            множество решений;
//           Det                - значение определителя матрицы A;
//           AForReverse        - нижняя треугольная матрица,
//                                полученная из A в результате
//                                выполнения прямого хода алгоритма
//                                Гаусса;
//           BForReverse        - столбец свободных членов, полученный
//                                из B в результате выполнения
//                                прямого хода алгоритма Гаусса.
// Результат: 0 - успешно,1 – ошибка.
////////////////////////////////////////////////////////////

unsigned char Gauss( Matrix A, Vector B, float Eps, Vector* X,
 char* HasSolution, short* NumOfRoots,
  float* Det, Matrix* AForReverse,
  Vector* BForReverse )
{
…
return 0;
…
return 1;
}
```

Замечание:

Если функция реализует какой-либо вычислительный метод (например: нахождение площади фигуры методом трапеций, поиск минимума функции методом Ньютона и т.п.), рекомендуется в теле функции поместить комментарий с кратким описанием метода, либо ссылку на источник, где описан метод [95].

Г.2.3 Спецификация программного файла или модуля

Программный файл или модуль должен начинаться со спецификации в виде комментария, содержащего следующую информацию [95]:

- Идентификация проекта, к которому принадлежит файл.
- Назначение (название) и имя файла.
- Версия файла.
- Фамилия автора.
- Описание модуля.
- История изменений модуля.

Например:

```
/*-------------------------------------------------------------
Проект:     ABC-1.0
Название:   Математические расчеты
Файл:       primes.c
Версия:     1.0.2
Автор:      Иванов И.И.
Описание:   Подсчет количества простых чисел.
Изменения:
-------------------------------------------------------------
NДата  Версия      Автор       Описание
-------------------------------------------------------------
101.03.07    1.0.1 Иванов И.И. Расчет в промежутке [1..200].
205.07.07    1.0.2 Иванов И.И. Расчет в заданном пользователем промежутке.
-------------------------------------------------------------*/
```

Замечание:

После спецификации программного файла рекомендуется поместить комментарий с указаниями по запуску программы и работе с ней (указаниями по использованию модуля другими программистами) или ссылку на источник, который использован при составлении программы (модуля).

Г.3 Соглашения по читаемости программ

Г.3.1 Лесенка

"Лесенка" должна отражать структурную вложенность языковых конструкций. Рекомендуется отступ не менее 2-х и не более 8-и пробелов. Принятого отступа нужно придерживаться во всем тексте программы. Правила написания некоторых конструкций [95]:

```
if ( <условие> )
{
    <операторы>
{

if ( <условие> )
    <оператор>;

if ( <условие> )
{
    <операторы>
}
else
{
    <операторы>
}

while ( <условие> )
{
    <операторы>
}

while ( <условие> )
    <оператор>;

for ( <иниц. счетчика>; <условие>; <изменение счетчика> )
{
```

```
    <операторы>
}

for ( <иниц. счетчика>; <условие>; <изменение счетчика> )
    <оператор>;

switch ( <выражение> )
{
    case <выражение>:
        <операторы>;
        break;
    ........
    default:
        <операторы>;
}

short Sign( float X )
{
    // выдает знак числа X
    if ( X > 0 ) return 1;
    else
        if ( X < 0 ) return -1;
        else
            return 0;
}

void Equation( float A, float B, float C,
               float* X1, float* X2, char* Num )
{
    // нахождение действительных корней квадратного уравнения;
    // A, B, C -- коэффициенты
    // X1, X2  -- корни (если действительного решения нет, то
    //             полагаются равными 0);
    // Num     -- число корней (0, 1, или 2)

    float D;

    D = sqr( B ) -4 * A * C;
    if ( D < 0 )
    {
        *Num = 0;
        *X1 = 0;
        *X2 = 0;
    }
    else
    {
        *X1 = ( -B + sqrt( D ) ) / ( 2 * A );
        *X2 = ( -B - sqrt( D ) ) / ( 2 * A );
        if ( *X1 == *X2 ) *Num = 1;
        else *Num = 2;
    }
}
```

Г.3.2 Длина строк программного текста

Длина строк программы не должна превышать ширины экрана (80 символов).

Г.3.3 Прочие рекомендации

А. Рекомендуется операнды бинарных операций (+, = и т.п.) отделять от знака операции одним пробелом " ".

Например:

```
Sum = A + B;
```

Б. Рекомендуется при перечислении идентификаторов после запятой "," ставить один пробел " ".

Например:

```
printf( “Сумма: %d; Разность: %d.”, A + B, A – B );
unsigned short Day, Month, Year;
unsigned char i, j, k, l, m, n;
```

В. Рекомендуется всегда писать символ-разделитель операторов ";" (непосредственно после оператора).

Например:

```
switch ( Num )
{
    case 1: printf( “один...” ); break;
    case 2: printf( “два...” ); break;
    case 3: printf( “три...” ); break;
    default: printf( “много!” ); break;
}
```

Г. Рекомендуется 16-ричные числа писать большими буквами.

Например:

```
#define BadDate 0xFFFF
#define kbEnter 0x0D // код клавиши <Enter>
```

Литература

1. 8237 DMA controller
   URL: http://zet.aluzina.org/index.php/8237_DMA_controller
2. 8237A High Performance Programmable DMA Controller Datasheet. – September 1993, Intel Corporation, Order number: 231466-005.
3. ADuC812: MicroConverter, Multichannel 12-Bit ADC with Embedded Flash MCU Data Sheet (Rev E, 04/2003) // Norwood: Analog Devices Inc. 2003. URL: http://www.analog.com/static/imported-files/data_sheets/ADUC812.pdf.
4. ADuC812: Quick Reference Guide – СПб.: СПбГУ ИТМО, 2009. – URL: http://embedded.ifmo.ru/sdk/sdk11/components/mcu_aduc812_eng/ADuC812_QuickRefGuideRev0.pdf
5. AMBA Design Tools for CoreLink System IP.
   URL: http://www.arm.com/products/system-ip/amba-design-tools
6. AMBA Specification (Rev 2.0) ARM Limited 1999
7. Application Note #1 (uC001): MicroConverter I2C Compatible Interface (Rev. 2.1, 2002/11) // Norwood: Analog Devices Inc. 2003.
   URL: http://www.analog.com/static/imported-files/application_notes/uC001_-_MicroConverter_I2C_Compatible_Interface.pdf.
8. Ayala K. The 8051 Microcontoller, 3rd ed. Delmar Cengage Learning, 2004. 448 p.
9. Ayala K. The 8051 microcontroller: architecture, programming and applications; 2nd ed. NY: Thomson Delmar Leaning, 1996. 367 p.
10. Calcutt D., Cowan F., Parchizadeh H. 8051 Microcontrollers: An Applications Based Introduction. Oxford: Newnes, 2004.
11. GNU make manual // The GNU Operating System. 1996.
    URL: http://www.gnu.org/software/make/manual/html_node/index.html.
12. I2C Bus. URL: http://www.i2c-bus.org
13. I²C-bus specification and user manual (Rev. 03, 06/2007) // NXP Semiconductors. 2006.
    URL: http://www.nxp.com/documents/user_manual/UM10204.pdf.
14. Lee A. Edward. Cyber-Physical Systems – Are Computing Foundations Adequate? // Position Paper for NSF Workshop On Cyber-Physical Systems: Research Motivation, Techniques and Roadmap. October 16-17, 2006, Austin, TX. 9 p.
15. Lee E.A. Model-Driven Development – From Object-Oriented Design to Actor-Oriented Design // In Workshop on Software Engineering for Embedded Systems: From Requirements to Implementation (a.k.a. The Monterey Workshop). Chicago. September 24, 2003. 7 p.
16. MacKenzie L.S., Phan R. The 8051 Microcontroller, 4th ed. NJ: Prentice Hall, 2005.
17. Mazidi M.A., McKinlay R. 8051 Microcontroller and Embedded Systems. NJ: Prentice Hall, 2005. 640 p.

18. Non-return-to-zero URL: http://en.wikipedia.org/wiki/Non-return-to-zero
19. PCF8583 – Clock/calendar with 240 x 8-bit RAM // NXP Semiconductors. 2006. URL: http://www.nxp.com/documents/data_sheet/PCF8583.pdf.
20. Sangiovanni-Vincentelli A. Quo Vadis SLD: Reasoning About the Trends and Challenges of System Level Design // Proceedings of the IEEE. 95(3), 2007. P. 467-506.
21. Schultz T.W. C and the 8051. Otsego: PageFree Publishing Inc., 2004. 3rd ed. 412 p. ISBN 1-58961-237-X
22. SDCC Compiler User Guide // SDCC – Small Device C Compiler. 2009. URL: http://sdcc.sourceforge.net/doc/sdccman.html/.
23. The I²C-bus specification (version 2.1, 2000) // NXP Semiconductors. 2006. URL: http://www.nxp.com/acrobat_download2/literature/9398/39340011.pdf.
24. Two-wire Serial EEPROM AT24C02A/AT24C04A // Atmel Corporation. 1998.
URL: http://www.atmel.com/dyn/resources/prod_documents/doc5083.pdf.
25. Wolf W.H. Computers as Components: Principles of Embedded Computing Systems Design. San Francisco: Morgan Kaufmann, 2005. 656 p. – ISBN 978-0-12-369459-1
26. Александров Е.К. Микропроцессорные системы: Учебное пособие для вузов. М.: Политехника, 2002. – 936 с. - ISBN 5-7325-0516-4
27. Бень Е.А. RS-485 для чайников. 2003.
URL: http://masters.donntu.edu.ua/2004/fema/kovalenko/library/art7.html
28. Болл Стюарт Р. Аналоговые интерфейсы микроконтроллеров. М.: Издательский дом «Додэка-XX1», 2007. 360 с: ил. – ISBN 978-5-94120-142-6
29. Брукс Ф. Мифический человеко-месяц или как создаются программные системы. СПб.: Символ-Плюс, 2010. 304 с. ISBN 5-93286-005-7, ISBN 0-201-83595-9
30. Буч Г. Объектно-ориентированный анализ и проектирование с примерами на языке C++. СПб.: Бином, Невский диалект, 1998. 560 с. ISBN 0-8053-5340-2, ISBN 5-7989-0067-3
31. Вербин В.С. Классификация электромагнитных помех
URL: http://www.problemaemc.narod.ru/emp_klass.html
32. Волович Г.И. Схемотехника аналоговых и аналого-цифровых электронных устройств. М.: Издательский дом "Додэка-XXI", 2005
33. Гарновский Н. Н. Теоретические основы электропроводной связи. Ч.2. М. 1959.
34. Говард В. Д. Высокоскоростная передача цифровых данных: высший курс черной магии.
35. Гоноровский И. С. Радиотехнические цепи и сигналы. Ч.2. М. 1967.
36. Горелик Г. С. Колебания и волны. М. – Л.. 1950.
37. Гук М.Ю. Аппаратные интерфейсы ПК. Энциклопедия. СПб.: Питер, 2002. 528 с.: ил. ISBN 5-94723-180-8

38. Гук М.Ю. Аппаратные средства IBM PC. Энциклопедия. 3-е изд. СПб.: Питер, 2006. 1072 с.: ил. ISBN 5-469-01182-8
39. Давыдов А.В. Лекции по проектированию радиоэлектронной геофизической аппаратуры. Электрические соединения в радиоэлектронной аппаратуре.
URL: http://prodav.narod.ru/design/index.html
40. Дейкстра Э.В. Два взгляда на программирование // Клуб программистов «Весельчак У»
URL: http://club.shelek.ru/viewart.php?id=211.
41. Дейкстра Э.В. Конец информатики? // Communications of the ACM, Ноябрь, 19, 2000. 44(3). с. 92
42. Дейкстра Э.В. Почему программное обеспечение такое дорогое? Пояснение для разработчиков аппаратуры // Springer-Verlag, 1982. с. 338-348
43. Дейкстра Э.В. Программирование как вид человеческой деятельности // Клуб программистов «Весельчак У»
URL: http://club.shelek.ru/viewart.php?id=137.
44. Ершова Н.Ю., Ивашенков О.Н., Курсков С.Ю. Микропроцессоры: Пособие к курсам «Микропроцессорные средства» и «Автоматизированные системы для научных исследований».
URL: http://dfe.karelia.ru/koi/posob/microcpu/index.html
45. Игнатов В. Эффективное использование GNU Make // Центр Информационных Технологий. 1997.
URL: http://www.citforum.ru/operating_systems/gnumake/index.shtml.
46. Интерфейсная шина IIC (I2C) // Easy Elecronics – Электроника для всех. 2008. URL: http://easyelectronics.ru/interface-bus-iic-i2c.html.
47. Использование сбалансированных схем
URL: http://www.elart.narod.ru/articles/article26/article26.htm
48. Керниган Б., Ритчи Д. Язык программирования С. 5-е изд. М.: Вильямс, 2009. 304 с. ISBN 978-5-8459-0891-9, ISBN 5-8459-0891-4, ISBN 0-13-110362-8
49. Ключев А.О., Кустарев П.В., Ковязина Д.Р., Петров Е.В. Программное обеспечение встроенных вычислительных систем. СПб.: СПбГУ ИТМО, 2009. 212 с.
50. Ключев, А.О., Ковязина, Д.Р., Кустарев, П.В., Платунов, А.Е. Аппаратные и программные средства встраиваемых систем. Учебное пособие. СПб.: СПбГУ ИТМО, 2010. 287 с.: ил.
51. Комплекс лабораторных работ для учебного лабораторного стенда SDK-1.1 // Интернет-портал «Встроенные вычислительные системы и системы на кристалле». 2009.
URL: http://embedded.ifmo.ru/sdk/sdk11/labs/2003.
52. Компьютерные сети. Цифровое кодирование
URL: http://sesia5.ru/lokseti/s222.htm
53. Краткое описание интерфейса ИРПС 20мА «токовая петля»
URL: http://www.sector-t.ru/info/sector/proizvod/cl_reference.php

54. Кузьминов А.Ю. Интерфейс RS232: Связь между компьютером и микроконтроллером: От DOS к WINDOWS98/XP. М.: Издательский дом «ДМКпресс», 2006. 320 с. ISBN 5-9706-0029-6
55. Кустарев П.В. Схемотехника ЭВМ. Конспект лекций.СПб.: СПбГУ ИТМО, 2008. 39 с.
56. Лапин А.А. Интерфейсы. Выбор и реализация. М.: Техносфера. 2005. 168 с. – ISBN 5-94836-058-X
57. Логическая схема расширителя портов ввода-вывода стенда SDK-1.1 (Rev. 3) // Интернет-портал «Встроенные вычислительные системы и системы на кристалле». 2009.
URL: http://embedded.ifmo.ru/sdk/sdk11/sch/sdk11r3_pld_ext.pdf.
58. Матричная клавиатура // Easy Elecronics – Электроника для всех. 2008.
URL: http://easyelectronics.ru/matrichnaya-klaviatura.html.
59. Матричная клавиатура // SKF development. 2009.
URL: http://www.microcontrollerov.net/index.php/ru/microcontrollers/articles/23-matrixkeyboard.
60. Микроконтроллеры семейства MCS-51. СПб.: СПбГУ ИТМО, 2009.
URL: http://embedded.ifmo.ru/sdk/sdk11/components/mcu_aduc812_rus/mcs51.pdf.
61. Микросхемы приемопередатчиков
URL: http://www.compitech.ru/html.cgi/arhiv/02_05/stat_68.htm
62. Могнонов П.Б. Организация микропроцессорных систем: Учебное пособие. Улан-Удэ: Изд-во ВСГТУ. 2003. 355 с.
63. Мячев А.А., Степанов В.Н., Щербо В.К. Интерфейсы систем обработки данных: Справочник; Под ред. А.А. Мячева. М.: Радио и связь. 1989. 416 с.: ил. – ISBN 5-256-00315-1
64. Непейвода Н.Н., Скопин И.Н. Основания программирования. М.-Ижевск: Институт компьютерных исследований, 2003. 868 с. ISBN 5-93972-299-7
65. Новиков Ю.В., Кондратенко С.В. Основы локальных сетей
URL: http://www.intuit.ru/department/network/baslocnet/3/
66. Новиков Ю.В., Скоробогатов П.К. Основы микропроцессорной техники. URL: http://www.intuit.ru/department/hardware/mpbasics/
67. Обзор стандарта RS-232
URL: http://www.gaw.ru/html.cgi/txt/interface/rs232/start.htm
68. Передача сигнала по кабелю витой пары
URL: http://www.kramer.ru/academy/courses/1268
69. Платунов А.Е., Постников Н.П. Перспективы формализации методов проектирования встроенных систем // «Электронные компоненты», 2005, №1. с.1-6.
70. Полосковая линия.
URL: http://www.cultinfo.ru/fulltext/1/001/008/091/089.htm
71. Последовательный интерфейс SPI.
URL: http://www.gaw.ru/html.cgi/txt/interface/spi/index.htm

72. Правильная разводка сетей RS-485.
URL: http://www.gaw.ru/html.cgi/txt/interface/rs485/app.htm
73. Предко М. Руководство по микроконтроллерам. В двух томах. М.: Постмаркет. 2004. 904 с. ISBN 5-901095-07-3
74. Применение устройств гальванической развязки цифрового сигнала в интерфейсах
URL: http://www.compitech.ru/html.cgi/arhiv/06_01/stat_ad.htm
75. Принципиальная электрическая схема стенда SDK-1.1 (Rev. 4) // Интернет-портал «Встроенные вычислительные системы и системы на кристалле». 2009.
URL: http://embedded.ifmo.ru/sdk/sdk11/sch/sdk1_1_sch_rev4.pdf.
76. Природа и основные параметры влияния между цепями
URL: http://www.tehnauk.ru/1/14?start=3
77. Ратхор Т.С. Цифровые измерения. АЦП / ЦАП. М.: Мир электроники, 2006.
78. Система команд микроконтроллеров семейства MCS-51
URL: http://www.gaw.ru/html.cgi/txt/doc/micros/mcs51/asm/start.htm
79. Согласование, экранирование и гальваническая развязка линий связи
URL: http://www.intuit.ru/department/network/baslocnet/3/
80. Спецификация ЖКИ WH1602B-YGK-CP (Winstar Display Co.) // Интернет-портал «Встроенные вычислительные системы и системы на кристалле». 2009.
URL: http://embedded.ifmo.ru/sdk/sdk11/components/lcd/WH1602B-YGK-CP.pdf.
81. Спецификация интерфейса I²C // KAZUS.RU «Электронный портал». 2003. URL: http://kazus.ru/articles/343.html.
82. Спецификация контроллера ЖКИ HD44780U (HITACHI) // Интернет-портал «Встроенные вычислительные системы и системы на кристалле». 2009.
URL: http://embedded.ifmo.ru/sdk/sdk11/components/lcd/hd44780.pdf.
83. Таненбаум Э. Архитектура компьютера. 5-е изд. СПб.: Питер, 2007. 844 с.: ил. ISBN 5-469-01274-3
84. Таненбаум Э. Современные операционные системы. 2-е изд. СПб.: Питер, 2002. 1040 с.: ил. ISBN 5-318-00299-4
85. Татарчевский В. Применение SWITCH-технологии при разработке прикладного программного обеспечения для микроконтроллеров. Часть 8 // «Компоненты и технологии», 2007, №8. с. 170-172
86. Угрюмов Е.П. Цифровая схемотехника. СПб.: БХВ-Петербург, 2004. 528 с.: ил. ISBN 5-8206-0100-9.
87. Уилльямс Т. ЭМС для разработчиков продукции Технологии. 2003. 540с.
88. Учебный стенд SDK-1.1. Руководство пользователя (Версия 1.0.11) // Интернет-портал «Встроенные вычислительные системы и системы на кристалле». 2009.
URL: http://embedded.ifmo.ru/sdk/sdk11/doc/sdk11_userm_v1_0_11.pdf.

89. Хаммел Р.Л. Последовательная передача данных: Руководство для программиста: Пер. с англ. М.: Мир, 1996. 752 с. ISBN 5-03-003140-5
90. Хмелевский И.В., Битюцкий В.П. Организация ЭВМ и систем. Однопроцессорные ЭВМ. Ч.1, 2: Конспект лекций. 2-е изд. Екатеринбург: ГОУ ВПО УГТУ-УПИ, 2005. 87 с.
91. Хоровиц П., Хилл У. Искусство схемотехники. 7-е изд. М.: Мир, 2009. 704 с. ISBN 0-521-37095-7
92. Цикл лекций «Устройства приема и обработки радиосигналов в системах подвижной радиосвязи» URL: http://digital.sibsutis.ru/UPriOS.htm
93. Цикл лекций по микроконтроллерам семейства MCS-51 // Цифровая техника в радиосвязи. 2009. URL: http://digital.sibsutis.ru/content.htm.
94. Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем: Учебник для вузов. СПб.: Питер, 2007. 672 с. ISBN 5-94723-759-8
95. Цымблер М.Л. Требования к оформлению программ на языке Turbo Pascal // М.Л. Цымблер. 2010. URL: http://www.mzym.susu.ru/papers/coderule.html.
96. Чернышев Ю.А. Компьютерные коммуникации и периферия. Часть II: Компьютерные коммуникации и интерфейсы. Электронное учебное пособие. М.: МИФИ (ТУ), 2000. 68 с.
97. Шалыто А.А. Switch-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998. 628 с.
98. Шины микропроцессорной системы. URL: http://www.intuit.ru/department/hardware/mpbasics/2/#image.2.1
99. Электронный замок на ключах – "таблетках" iButton. URL: http://microsin.ru/content/view/508/44/
100. Юкио Сато. Обработка сигналов. Первое знакомство. М.: Издательский дом «Додэка-XXI», 2009. 175 с.
101. Яшкардин В.Л. RS-232. Рекомендованный стандарт 232. Интерфейс между терминалом данных и передающим оборудованием линии связи, применяющий последовательный обмен двоичными данными. URL: http://www.softelectro.ru/rs232.html

В 2009 году Университет стал победителем многоэтапного конкурса, в результате которого определены 12 ведущих университетов России с присвоением категории «Национальный исследовательский университет». Министерством образования и науки Российской Федерации была утверждена Программа развития государственного образовательного учреждения высшего профессионального образования «Санкт-Петербургский государственный университет информационных технологий, механики и оптики» на 2009–2018 годы.

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Кафедра ВТ СПбГУ ИТМО создана в 1937 году и является одной из старейших и авторитетнейших научно-педагогических школ России.

Традиционно основной упор в подготовке специалистов на кафедре делается на фундаментальную базовую подготовку в рамках общепрофессиональных и специальных дисциплин, охватывающих наиболее важные разделы вычислительной техники.

Кафедра является одной из крупнейших в университете. Учебными курсами и научно-исследовательскими работами руководят 8 профессоров и 16 доцентов. На кафедре обучаются более 500 студентов и 30 аспирантов.

Кафедра имеет собственные компьютерные классы и специализированные исследовательские лаборатории, оснащенные современной вычислительной и оргтехникой, уникальным инструментальным и технологическим оборудованием, измерительными приборами и программным обеспечением.

В 2007-2008 гг. коллективом кафедры была успешно реализована инновационная образовательная программа СПбГУ ИТМО по научно-образовательному направлению «Встроенные вычислительные системы».

Начиная с 2009 года кафедра вычислительной техники является активным участником реализации программы развития национального исследовательского университета СПбГУ ИТМО, вошла в состав крупнейшего в СПбГУ ИТМО научно-исследовательского центра «Интеллектуальные системы управления и обработки информации».

Аркадий Олегович Ключев
Динара Раисовна Ковязина
Евгений Владимирович Петров
Алексей Евгеньевич Платунов

ИНТЕРФЕЙСЫ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ

Учебное пособие

В авторской редакции
Дизайн Е.В.Петров
Верстка Е.В.Петров
Редакционно-издательский отдел Санкт-Петербургского государственного университета информационных технологий, механики и оптики
Зав. РИО Н.Ф. Гусарова
Лицензия ИД №
Подписано к печати
Заказ №
Тираж 100 экз.
Отпечатано на ризографе

Ключев А.О., Ковязина Д.Р., Петров Е.В., Платунов А.Е.

ИНТЕРФЕЙСЫ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ

УЧЕБНОЕ ПОСОБИЕ

Санкт-Петербург
2010

Редакционно-издательский отдел
Санкт-Петербургского государственного
университета информационных технологий,
механики и оптики
197101, Санкт-Петербург, Кронверкский пр., 49