ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

Русанов В.В., Шевелев М.Ю.

# МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА И СИСТЕМЫ

Учебное пособие

2012

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ**

**Кафедра Промышленной электроники**

# «МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА И СИСТЕМЫ (МПУиС)»

**Учебное пособие**
**для студентов**
**направления 210100 «Электроника и наноэлектроника»**

2012

УДК 681.325.5-0181.4(075.8)
ББК 32.97я73
Р882

Р882 **Русанов В.В., Шевелёв М.Ю.**
Микропроцессорные устройства и системы: Учебное пособие для вузов. – Томск: Томский государственный университет систем управления и радиоэлектроники, 2012. – 184 с.
ISBN 978-5-94154-128-7

В пособии излагаются вопросы организации функционирования и программирования микропроцессорных средств. Представлены микропроцессоры общего применения, процессоры обработки сигналов, а также микроконтроллеры для встроенных приложений: коммуникационные, для задач управления и др. Рассматриваются программные модели процессоров и микроконтроллеров, особенности организации периферийных устройств, средства отладки и проектирования. Приводятся примеры применения и программирования.

Учебное пособие ориентировано на студентов технических университетов, обучающихся по направлению «Цифровая и микропроцессорная техника».

УДК 681.325.5-0181.4(075.8)
ББК 32.97я73

ISBN 978-5-94154-128-7 

**Оглавление**

**Введение**

Дисциплина «Микропроцессорные устройства и системы» логически продолжает цикл «Цифровая и микропроцессорная техника», включающий дисциплины «Микроэлектроника» и «Основы микропроцессорной техники».

Целью курса является изучение принципов построения и организации микропроцессорных систем, особенностей проектирования электронных систем управления на их основе и знакомство с отладочными средствами микропроцессорных устройств.

*Микропроцессорная техника* (МПТ) – это комплекс технических и программных средств для построения различных микропроцессорных устройств и систем.

*Микропроцессорная система* (МПС) – это функционально законченное изделие, состоящее из одного или нескольких устройств, главным образом микропроцессорных.

*Микропроцессорное устройство* (МПУ) представляет собой функционально и конструктивно законченное изделие, состоящее из нескольких микросхем, в состав которых входит микропроцессор. Оно предназначено для выполнения определенного набора функций: получение, обработка, передача, преобразование информации и управление.

*Микропроцессор* (МП) – это программно-управляемое устройство, построенное, как правило, на одной БИС и осуществляющее процесс управления и цифровой обработки информации.

*Микроконтроллер* – это однокристальная ЭВМ, структурная схема которой содержит все функциональные узлы, необходимые для обеспечения автономной работы в качестве управляющего или вычислительного устройства.

Основной особенностью микроконтроллера является то, что он, кроме микропроцессора, может содержать на одном кристалле ОЗУ и ПЗУ нескольких типов, блоки ввода-вывода, управления и синхронизации, и, главным образом, набор различных периферийных блоков.

Хотя современные микропроцессоры тоже могут содержать на кристалле дополнительные блоки, принято считать, что микропроцессоры применяются для создания персональных компь-

ютеров, а микроконтроллеры – для создания различных встраиваемых систем, систем управления телекоммуникациями, портативным и технологическим оборудованием. Его назначение следует из самого названия «микроконтроллер» (control – управление).

Расширение сферы использования МК повлекло за собой развитие их архитектуры за счет размещения на кристалле устройств (модулей), отражающих своими функциональными возможностями специфику решаемых задач. Такие дополнительные устройства стали называться периферийными. По этой причине в последнее время введен еще один термин — *«интегрированный процессор»* (ИП), который определяет новый класс функционально-емких однокристальных устройств с другим составом модулей. По количеству и составу периферийных устройств интегрированные процессоры уступают микроконтроллерам и занимают промежуточное положение между микропроцессорами и микроконтроллерами. По этой же причине появились не только семейства микроконтроллеров, которые объединяют родственные (с одинаковой системой команд, разрядностью) микроконтроллеры, но и стали выделяться их различные подвиды: коммуникационные, для управления и т. д.

Под *семейством* микроконтроллеров понимают микроконтроллеры, имеющие общую архитектуру и структуру и объединенные общей системой команд.

*Архитектурой* микроконтроллера называется комплекс его аппаратных и программных средств, предоставляемых пользователю.

*Структура* микропроцессора определяет состав и взаимодействие основных устройств и блоков, размещенных на его кристалле.

# 1. КЛАССИФИКАЦИЯ МИКРОПРОЦЕССОРОВ

## 1.1. Основные варианты архитектуры и структуры

В подавляющем большинстве случаев микропроцессор является универсальным устройством для выполнения программной обработки информации, которое может использоваться в самых разнообразных сферах человеческой деятельности. Многочисленные компании-производители выпускают несколько тысяч типов микропроцессоров, имеющих разные характеристики и предназначенных для различных областей применения. Выпускаемые микропроцессоры делятся на отдельные классы в соответствии с их архитектурой, структурой и функциональным назначением. Ниже приведен обзор основных архитектурных и структурных вариантов реализации современных микропроцессоров, используемых в различных сферах применения.

*Основными направлениями развития микропроцессоров* является увеличение их производительности и расширение функциональных возможностей, что достигается как повышением уровня микроэлектронной технологии, используемой для производства микропроцессоров, так и применением новых архитектурных и структурных вариантов их реализации. Развитие микроэлектронной технологии обеспечивает непрерывное уменьшение размеров полупроводниковых компонентов, размещаемых на кристалле микропроцессора. При этом уменьшаются паразитные емкости, определяющие задержку переключения логических элементов, и увеличивается число элементов, размещаемых на кристалле. В настоящее время разрешающая способность промышленной технологии изготовления микросхем обеспечивает создание компонентов с минимальными размерами 65-90 нм. При этом обеспечивается создание микропроцессоров, работающих с тактовой частотой до 1 — 2 ГГц и содержащих на кристалле десятки миллионов транзисторов. В соответствии с эмпирическим правилом, которое сформулировал Гордон Мур, один из основателей компании Intel, степень интеграции микросхем удваивается каждые 1,5 – 2 года. Это правило выполнялось в течение 40 лет развития микроэлектроники, и можно прогнозировать, что оно будет выполняться и в близком будущем. По-

этому можно ожидать последующего быстрого прогресса технологии и связанного с ним повышения характеристик микропроцессоров.

Постоянное совершенствование технологии обеспечивает возможность создания на кристалле все большего количества активных компонентов — транзисторов, которые могут быть использованы для реализации новых архитектурных и структурных решений, обеспечивающих повышение производительности и расширение функциональных возможностей микропроцессоров. Кратко рассмотрим основные решения.

*Архитектура* процессора – это комплекс его аппаратных и программных средств, предоставляемых пользователю. В это общее понятие входит набор программно-доступных регистров и исполнительных (операционных) устройств, система основных команд и способов адресации, объем и структура адресуемой памяти, виды и способы обработки прерываний. Например, все модификации процессоров Pentium, Celeron, i486 и i386 имеют архитектуру IA-32 (Intel Architecture — 32 bit), которая характеризуется стандартным набором регистров, предоставляемых пользователю, общей системой основных команд и способов организации и адресации памяти, одинаковой реализацией защиты памяти и обслуживания прерываний.

При описании архитектуры и функционирования процессора обычно используется его представление в виде совокупности программно-доступных регистров, образующих регистровую или программную модель. В этих регистрах содержатся обрабатываемые данные (операнды) и управляющая информация. Соответственно, в регистровую модель входит группа регистров общего назначения, служащих для хранения операндов, и группа служебных регистров, обеспечивающих управление выполнением программы и режимом работы процессора, организацию обращения к памяти (защита памяти, сегментная и страничная организация и др.).

Регистры общего назначения (РОН) образуют внутреннюю регистровую память процессора. Состав и количество служебных регистров определяется архитектурой микропроцессора. Обычно в их состав входят:

– программный счетчик PC (или CS + IP в архитектуре микропроцессоров Intel);

– регистр состояния PSW (или EFLAGS);

– регистры управления режимом работы процессора CR (Control Register);

– регистры, реализующие сегментную и страничную организацию памяти;

– регистры, обеспечивающие отладку программ и тестирование процессора.

Кроме того, различные модели микропроцессоров содержат ряд других специализированных регистров.

Процесс функционирования процессора можно представить в виде реализации регистровых пересылок — процедур изменения состояния этих регистров путем чтения-записи их содержимого. В результате таких пересылок обеспечивается адресация и выбор команд и операндов, хранение и пересылка результатов, изменение последовательности команд и режимов функционирования процессора в соответствии с поступлением нового содержимого в служебные регистры, а также все другие процедуры, реализующие процесс обработки информации согласно заданным условиям.

В некоторых процессорах выделяются регистры, которые используются при выполнении прикладных программ и доступны каждому пользователю, и регистры, которые управляют режимом работы всей системы и доступны только для привилегированных программ, входящих в состав операционной системы (супервизора). Соответственно, такие процессоры представляются в виде регистровой модели пользователя, в которую входят регистры, используемые при выполнении прикладных программ, или регистровой модели супервизора, которая содержит весь набор программно-доступных регистров процессора, используемых операционной системой.

*Структура* микропроцессора определяет состав и взаимодействие основных устройств и блоков, размещенных на его кристалле. В эту структуру входят:

– центральный процессор (процессорное ядро), состоящее из устройства управления (УУ), одного или нескольких операционных устройств (ОУ);

– внутренняя память (РЗУ, кэш-память, блоки оперативной и постоянной памяти);

– интерфейсный блок, обеспечивающий выход на системную шину и обмен данными с внешними устройствами через параллельные или последовательные порты ввода/ вывода;

– периферийные устройства (таймерные модули, аналого-цифровые преобразователи, специализированные контроллеры);

– различные вспомогательные схемы (генератор тактовых импульсов, схемы для выполнения отладки и тестирования, сторожевой таймер и ряд других).

Состав устройств и блоков, входящих е структуру микропроцессора, и реализуемые механизмы их взаимодействия определяются функциональным назначением и областью применения микропроцессора.

Понятия архитектуры и структуры микропроцессора тесно взаимосвязаны. Реализация тех или иных архитектурных особенностей требует введения в структуру микропроцессора необходимых аппаратных средств (устройств и блоков) и обеспечения соответствующих механизмов их совместного функционирования.

Исторически сложились следующие варианты архитектур микропроцессоров.

*CISC* (Complex Instruction Set Computer) – архитектура реализована во многих типах микропроцессоров, выполняющих большой набор разноформатных команд с использованием многочисленных способов адресации. Эта классическая архитектура процессоров, которая начала свое развитие в 1940-х годах с появлением первых компьютеров. Типичным примером CISC-процессоров являются микропроцессоры семейства Pentium. Они выполняют более 200 команд разной степени сложности, которые имеют размер от 1 до 15 байт и обеспечивают более 10 различных способов адресации. Такое большое многообразие выполняемых команд и способов адресации позволяет программисту реализовать наиболее эффективные алгоритмы решения различных задач. Однако при этом существенно усложняется структура микропроцессора, особенно его устройства управления, что приводит к увеличению размеров и стоимости кристалла, снижению производительности. В то же время многие ко-

манды и способы адресации используются достаточно редко. Поэтому, начиная с 1980-х годов, интенсивное развитие получила архитектура процессоров с сокращенным набором команд (RISC-процессоры).

*RISC* (Reduced Instruction Set Computer) – архитектура отличается использованием ограниченного набора команд фиксированного формата. Современные RISC-процессоры обычно реализуют не более 100 команд, имеющих фиксированный формат длиной 4 байта. Также значительно сокращается число используемых способов адресации. Обычно в RISC-процессорах все команды обработки данных выполняются только с регистровой или непосредственной адресацией. При этом для сокращения количества обращений к памяти RISC-процессоры имеют увеличенный объем внутреннего РЗУ — от 32 до нескольких сотен регистров, тогда как в CISC-процессорах число регистров общего назначения обычно составляет 8 – 16.

Обращение к памяти в RISC-процессорах используется только в операциях загрузки данных в РЗУ или пересылки результатов из РЗУ в память. При этом используется небольшое число наиболее простых способов адресации: косвенно-регистровая, индексная и некоторые другие. В результате существенно упрощается структура микропроцессора, сокращаются его размеры и стоимость, значительно повышается производительность.

Эти достоинства RISC-архитектуры привели к тому, что во многих современных CISC-процессорах используется RISC-ядро, выполняющее обработку данных. При этом поступающие сложные и разноформатные команды предварительно преобразуются в последовательность простых RISC-операций, быстро выполняемых этим процессорным ядром. Таким образом, работают, например, последние модели микропроцессоров Pentium и K7, которые по внешним показателям относятся к CISC-процессорам. Использование RISC-архитектуры является характерной чертой многих современных микропроцессоров.

*VLIW* (Very Large Instruction Word) – архитектура появилась относительно недавно – в 1990-х годах. Ее особенностью является использование очень длинных команд (128 бит и более), отдельные поля которых содержат коды, обеспечивающие вы-

полнение различных операций. Таким образом, одна команда вызывает выполнение сразу нескольких операций параллельно в различных операционных устройствах, входящих в структуру микропроцессора. При трансляции программ, написанных на языке высокого уровня, соответствующий компилятор производит формирование «длинных» VLIW-команд, каждая из которых обеспечивает реализацию процессором целой процедуры или группы операций. Данная архитектура реализована в некоторых типах современных микропроцессоров (PA8500 компании «Hewlett-Packard», Itanium — совместная разработка «Intel» и « Hewlett-Packard», некоторые типы DSP — цифровых процессоров сигналов) и является весьма перспективной для создания нового поколения сверхвысокопроизводительных процессоров.

Кроме набора выполняемых команд и способов адресации важной архитектурной особенностью микропроцессоров является используемый вариант реализации памяти и организация выборки команд и данных. По этим признакам различаются процессоры с Принстонской и Гарвардской архитектурой. Эти архитектурные варианты были предложены в конце 1940-х годов специалистами соответственно Принстонского и Гарвардского университетов США для разрабатываемых ими моделей компьютеров.

*Принстонская* архитектура, которая часто называется архитектурой Фон-Неймана, характеризуется использованием общей оперативной памяти для хранения программ, данных, а также для организации стека. Для обращения к этой памяти используется общая системная шина, по которой в процессор поступают и команды, и данные. Эта архитектура имеет ряд важных достоинств. Наличие общей памяти позволяет оперативно перераспределять ее объем для хранения отдельных массивов команд, данных и реализации стека в зависимости от решаемых задач. Таким образом, обеспечивается возможность более эффективного использования имеющегося объема оперативной памяти в каждом конкретном случае применения микропроцессора. Использование общей шины для передачи команд и данных значительно упрощает отладку, тестирование и текущий контроль функционирования системы, повышает ее надежность. Поэтому

Принстонская архитектура в течение долгого времени доминировала в вычислительной технике.

Однако ей присущи и существенные недостатки. Основным из них является необходимость последовательной выборки команд и обрабатываемых данных по общей системной шине. При этом общая шина становится «узким местом» (bottleneck — «бутылочное горло»), которое ограничивает производительность цифровой системы. Постоянно возрастающие требования к производительности микропроцессорных систем вызвали в последние годы все более широкое применение Гарвардской архитектуры при создании многих типов современных микропроцессоров.

*Гарвардская* архитектура характеризуется физическим разделением памяти команд (программ) и памяти данных. В ее оригинальном варианте использовался также отдельный стек для хранения содержимого программного счетчика, который обеспечивал возможности выполнения вложенных подпрограмм. Каждая память соединяется с процессором отдельной шиной, что позволяет одновременно с чтением-записью данных при выполнении текущей команды производить выборку и декодирование следующей команды. Благодаря такому разделению потоков команд и данных и совмещению операций их выборки реализуется более высокая производительность, чем при использовании Принстонской архитектуры.

Недостатки Гарвардской архитектуры связаны с необходимостью проведения большего количества шин, а также с фиксированным объемом памяти, выделенной для команд и данных, значение которой не может оперативно перераспределяться в соответствии с требованиями решаемой задачи. Поэтому приходится использовать память большего объема, коэффициент использования которой при решении разнообразных задач оказывается более низким, чем в системах с Принстонской архитектурой. Однако развитие микроэлектронной технологии позволило в значительной степени преодолеть указанные недостатки, поэтому Гарвардская архитектура широко применяется во внутренней структуре современных высокопроизводительных микропроцессоров, где используется отдельная кэш-память для хранения команд и данных. В то же время во внешней структуре

большинства микропроцессоров реализуются принципы Принстонской архитектуры.

Гарвардская архитектура получила также широкое применение в микроконтроллерах – специализированных микропроцессорах для управления различными объектами, рабочая программа которых обычно хранится в отдельном ПЗУ.

## 1.2. Классификация современных микропроцессоров по функциональному признаку

Микропроцессор является универсальным средством для цифровой обработки информации, однако отдельные области применения требуют реализации определенных специфических вариантов их структуры и архитектуры. По этой причине по функциональному признаку выделяются два класса: микропроцессоры общего назначения и специализированные микропроцессоры (рис. 1.1). Среди специализированных микропроцессоров наиболее широкое распространение получили микроконтроллеры, предназначенные для выполнения функций управления различными объектами, и цифровые сигнальные процессоры (DSP — Digital Signal Processor), которые ориентированы на реализацию процедур, обеспечивающих преобразование аналоговых сигналов, представленных в цифровой форме (в виде последовательности числовых значений).

*Микропроцессоры общего назначения* предназначены для решения широкого круга задач обработки разнообразной информации. Их основной областью использования являются персональные компьютеры, рабочие станции, серверы и другие цифровые системы массового применения. К этому классу относятся CISC-процессоры Pentium компании «Intel», K7 — компании «Advanced MicroDevices» (AMD), 680x0 — компании «Motorola», RISC-процессоры PowerPC, выпускаемые компаниями «Motorola» и IBM, SPARC — компании «Sun Microsystems» и ряд других изделий различных производителей.

Расширение области применения таких микропроцессоров достигается главным образом путем роста производительности, благодаря чему увеличивается круг задач, который можно решать с их использованием. Поэтому повышение производитель-

ности является магистральным направлением развития этого класса микропроцессоров. Обычно это 32-разрядные микропроцессоры (некоторые микропроцессоры этого класса имеют 64-разрядную или 128-разрядную структуру), которые изготавливаются по самой современной промышленной технологии, обеспечивающей максимальную частоту функционирования.

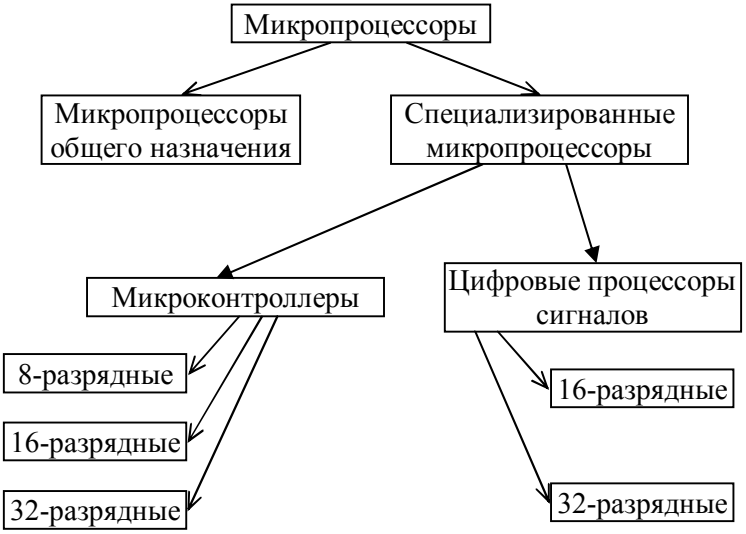


Рисунок 1.1. – Классификация микропроцессоров по функциональному признаку

Ряд наиболее популярных микропроцессоров этого класса (Pentium, AMD K7 и некоторые другие) следует отнести к CISC-процессорам, так как они выполняют большой набор разноформатных команд с использованием многочисленных способов адресации. Однако в их внутренней структуре содержится RISC-процессор, который выполняет поступившие команды после их преобразования в последовательность простых RISC-операций. Ряд других микропроцессоров этого класса непосредственно реализует RISC-архитектуру. Поэтому можно считать, что использование RISC-архитектуры характерно для большинства этих микропроцессоров. Однако в ряде последних разработок

(Itanium PA8500) некоторых ведущих производителей успешно применяются принципы VLIW-архитектуры, которая может составить конкуренцию RISC-архитектуре в соревновании за достижение наивысшей производительности.

Практически все современные микропроцессоры этого класса используют Гарвардскую внутреннюю архитектуру, где разделение потоков команд и данных реализуется с помощью отдельных блоков кэш-памяти. В большинстве случаев они имеют суперскалярную структуру с несколькими исполнительными конвейерами (до 10 в современных моделях) которые содержат до 20 ступеней.

Благодаря своей универсальности микропроцессоры общего назначения используются также в специализированных системах, где требуется высокая производительность. На их основе реализуются одноплатные компьютеры и промышленные компьютеры, которые применяются в системах управления различными объектами. Одноплатные (встраиваемые) компьютеры содержат на плате необходимые дополнительные микросхемы, обеспечивающие их специализированное применение, и предназначены для встраивания в аппаратуру различного назначения. Промышленные компьютеры размещаются в корпусах специальной конструкции, обеспечивающих их надежную работу в жестких производственных условиях. Обычно такие компьютеры работают без стандартных периферийных устройств (монитор, клавиатура, «мышь») или используют специальные варианты этих устройств, модифицированные с учетом специфических условий применения.

*Микроконтроллеры* являются специализированными микропроцессорами с встроенной периферией, которые ориентированы на реализацию устройств управления, встраиваемых в разнообразную аппаратуру. Ввиду огромного количества объектов, управление которыми обеспечивается с помощью микроконтроллеров, годовой объем их выпуска превышает 2 миллиарда экземпляров, на порядок превосходя объем выпуска микропроцессоров общего применения. Весьма широкой является также номенклатура выпускаемых микроконтроллеров, которая содержит несколько тысяч типов.

Характерной особенностью структуры микроконтроллеров является размещение на одном кристалле с центральным процессором внутренней памяти и большого набора периферийных устройств. В состав периферийных устройств обычно входят несколько 8-разрядных параллельных портов ввода-вывода данных (от 1 до 8), один или два последовательных порта, таймерный блок, аналого-цифровой преобразователь (АЦП). Кроме того, различные типы микроконтроллеров содержат дополнительные специализированные устройства — блок формирования сигналов с широтно-импульсной модуляцией (ШИМ), контроллер клавиатуры, жидкокристаллическою дисплея и ряд других. Благодаря использованию внутренней памяти и периферийных устройств реализуемые на базе микроконтроллеров системы управления содержат минимальное количество дополнительных компонентов.

В связи с широким диапазоном решаемых задач управления требования, предъявляемые к производительности процессора, объему внутренней памяти команд и данных, к набору необходимых периферийных устройств, оказываются весьма разнообразными. Для удовлетворения запросов потребителей выпускается большая номенклатура микроконтроллеров, которые принято подразделять на 8-, 16- и 32-разрядные.

*8-разрядные микроконтроллеры* представляют наиболее многочисленную группу этого класса микропроцессоров, которые имеют относительно низкую производительность, которая, однако, вполне достаточна для решения широкого круга задач управления различными объектами. Это простые и дешевые микроконтроллеры, ориентированные на использование в относительно несложных устройствах массового выпуска. Основными областями их применения являются бытовая и измерительная техника, промышленная автоматика, автомобильная электроника, теле-, видео- и аудиоаппаратура, средства связи.

Для этих микроконтроллеров характерна реализация Гарвардской архитектуры, где используется отдельная память для хранения программ и данных. Для хранения программ в различных типах микроконтроллеров применяется либо масочное программируемое ПЗУ (ROM), либо однократно-программируемое ПЗУ (PROM), либо электрически репрограммируемое ПЗУ

(EPROM, EEPROM или Flash). Внутренняя память программ обычно имеет объем от нескольких единиц до десятков Кбайт. Для хранения данных используется регистровый блок, организованный в виде нескольких регистровых банков, или внутреннее ОЗУ. Объем внутренней памяти данных составляет от нескольких десятков байт до нескольких Кбайт. Ряд микроконтроллеров этой группы позволяет в случае необходимости дополнительно подключать внешнюю память команд и данных объемом до 64-256 Кбайт.

8-разрядные микроконтроллеры этой группы обычно выполняют относительно небольшой набор команд (50—100), использующих наиболее простые способы адресации. В ряде последних моделей этих микроконтроллеров реализованы принципы RISC-архитектуры (например, семейство AVR компании Atmel, PIC-контроллеры компании Microchip), что позволяет существенно повысить их производительность. В результате такие микроконтроллеры обеспечивают выполнение большинства команд за один такт машинного времени.

*16-разрядные микроконтроллеры* во многих случаях являются усовершенствованной модификацией своих 8-разрядных прототипов. Они характеризуются не только увеличенной разрядностью обрабатываемых данных, но и расширенной системой команд и способов адресации, увеличенным набором регистров и объемом адресуемой памяти, а также рядом других дополнительных возможностей, использование которых позволяет повысить производительность и обеспечить новые области применения. Обычно эти микроконтроллеры позволяют расширить объем памяти программ и данных до нескольких Мбайт путем подключения внешних микросхем памяти. Во многих случаях реализуется их программная совместимость с младшими 8-разрядными моделями. Основная сфера применения таких микроконтроллеров — сложная промышленная автоматика, телекоммуникационная аппаратура, медицинская и измерительная техника.

*32-разрядные микроконтроллеры* содержат высокопроизводительный процессор, соответствующий по своим возможностям младшим моделям микропроцессоров общего назначения. В ряде случаев процессор, используемый в этих микроконтрол-

лерах, аналогичен CISC- или RISC-процессорам, которые выпускаются или выпускались ранее в качестве микропроцессоров общего назначения. Например, в 32-разрядных микроконтроллерах компании Intel используется процессор i386, в микроконтроллерах компании Motorola широко применяется процессор 680x0, в ряде других микроконтроллеров в качестве процессорного ядра служат RISC-процессоры типа PowerPC. На базе данных процессоров были реализованы различные модели персональных компьютеров. Введение этих процессоров в состав микроконтроллеров позволяет использовать в соответствующих системах управления огромный объем прикладного и системного программного обеспечения, созданный ранее для соответствующих персональных компьютеров. Кроме 32-разрядного процессора на кристалле микроконтроллера размещается внутренняя память команд емкостью до десятков Кбайт, память данных емкостью до нескольких Кбайт, а также сложнофункциональные периферийные устройства — таймерный процессор, коммуникационный процессор, модуль последовательного обмена и ряд других. Микроконтроллеры работают с внешней памятью объемом до 16 Мбайт и выше. Они находят широкое применение в системах управления сложными объектами промышленной автоматики (двигатели, робототехнические устройства, средства комплексной автоматизации производства), в контрольно-измерительной аппаратуре и телекоммуникационном оборудовании.

Во внутренней структуре этих микроконтроллеров реализуется Принстонская или Гарвардская архитектура. Входящие в их состав процессоры могут иметь CISC- или RISC-архитектуру, а некоторые из них содержат несколько исполнительных конвейеров, образующих суперскалярную структуру.

*Цифровые процессоры сигналов* (ЦПС) представляют класс специализированных микропроцессоров, ориентированных на цифровую обработку поступающих аналоговых сигналов. Специфической особенностью алгоритмов обработки аналоговых сигналов является необходимость последовательного выполнения ряда команд умножения-сложения с накоплением промежуточного результата в регистре-аккумуляторе. Поэтому архитектура ЦПС ориентирована на реализацию быстрого выпол-

нения операций такого рода. Набор команд этих процессоров содержит специальные команды MAC (Multiplication with Accumulation), реализующие эти операции.

Значение поступившего сигнала может быть представлено в виде числа с фиксированной или с «плавающей» точкой. В соответствии с этим ЦПС делятся на процессоры, обрабатывающие числа с фиксированной или плавающей точкой. Более простые и дешевые ЦПС с фиксированной точкой обычно обрабатывают 16-разрядные операнды, представленные в виде правильной дроби. Однако ограниченная разрядность в ряде случаев не позволяет обеспечить необходимую точность преобразования. Поэтому в ЦПС с фиксированной точкой, выпускаемых компанией «Motorola», принято 24-разрядное представление операндов. Наиболее высокая точность обработки обеспечивается в случае представления данных в формате с «плавающей» точкой. В ЦПС, обрабатывающих данные с «плавающей» точкой, обычно используется 32-разрядный формат их представления.

Для повышения производительности при выполнении специфических операций обработки сигналов в большинстве ЦПС реализуется Гарвардская архитектура с использованием нескольких шин для передачи адресов, команд и данных. В ряде ЦПС нашли применение также некоторые черты VLIW-архитектуры: совмещение в одной команде нескольких операций, обеспечивающих обработку имеющихся данных и одновременную загрузку в исполнительный конвейер новых данных для последующей обработки.

# 2. СТРУКТУРА МИКРОПРОЦЕССОРНЫХ СИСТЕМ И ПРИНЦИПЫ ИХ ФУНКЦИОНИРОВАНИЯ

## 2.1. Общие сведения о структуре микропроцессорных систем

Большинство микропроцессорных систем имеет магистрально-модульную структуру (рис. 2.1), в которой отдельные устройства (модули), входящие в состав системы, обмениваются информацией по общей системной магистрали, которая содержит три основные шины: шину управления, шину данных и шину адреса.

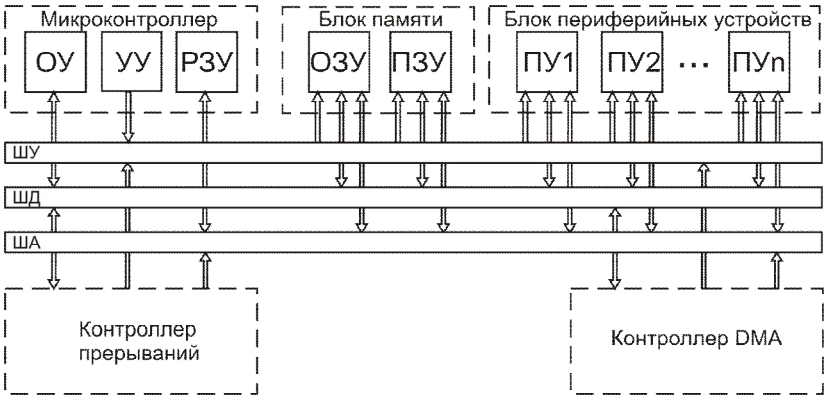


Рисунок 2.1. – Типовая структура МП-системы

Основным модулем системы является микропроцессор, который содержит устройство управления (УУ), операционное устройство (ОУ) и регистровое запоминающее устройство (РЗУ) — внутреннюю память, реализованную в виде набора регистров. Оперативное запоминающее устройство (ОЗУ) служит для хранения выполняемой программы (или ее фрагментов) и данных, подлежащих обработке. В простейших микропроцессорных системах объем ОЗУ составляет десятки и сотни байт, а современных персональных компьютерах, серверах и рабочих станциях он достигает сотен Мбайт и более. Так как обращение к ОЗУ по

системной шине требует значительных затрат времени, в большинстве современных высокопроизводительных микропроцессоров дополнительно вводится быстродействующая промежуточная память (кэш-память) ограниченного объема (от нескольких Кбайт до сотен Кбайт).

Постоянное запоминающее устройство (ПЗУ) служит для хранения констант и стандартных (неизменяемых) программ. В ПЗУ обычно записываются программы начальной инициализации (загрузки) систем, тестовые и диагностические программы и другое служебное программное обеспечение, которое не меняется в процессе эксплуатации систем. В микропроцессорных системах, управляющих определенными объектами с использованием фиксированных или редко изменяемых программ, для их хранения также обычно используется ПЗУ (память ROM — Read-Only Memory) или репрограммируемое ПЗУ (память EEPROM — Electrically Erased Programmable Read-Only Memory или Flash-память).

Остальные устройства являются внешними и подключаются к системе с помощью периферийных устройств (ПУ), реализующих определенные протоколы параллельного или последовательного обмена. Такими внешними устройствами могут быть клавиатура, монитор, внешние запоминающие устройства, использующие гибкие или жесткие магнитные диски, оптические диски (CD-ROM), магнитные ленты и другие виды носителей информации, датчики и преобразователи информации (аналого-цифровые или цифроаналоговые), разнообразные исполнительные устройства (индикаторы, принтеры, электродвигатели, реле и другие). Для реализации различных режимов работы к системе могут подключаться дополнительные устройства — контроллеры прерываний, прямого доступа к памяти (DMA) и другие, реализующие необходимые специальные функции управления.

Данная структура соответствует архитектуре Фон-Неймана, предложенной этим ученым в 1940-х годах для реализации первых моделей цифровых ЭВМ. Ниже будут рассмотрены и другие варианты процессорных архитектур.

*Системная шина* содержит несколько десятков (в сложных системах более 100) проводников, которые в соответствии с их

функциональным назначением подразделяются на отдельные шины — адреса, данных и управления.

Шина адреса служит для передачи адреса, который формируется микропроцессором и позволяет выбрать необходимую ячейку памяти ОЗУ (ПЗУ) или требуемое ИУ при обращении к внешнему устройству.

Шина данных служит для выборки команд, поступающих из ОЗУ или ПЗУ в УУ микропроцессора, и для пересылки обрабатываемых данных (операндов) между микропроцессором и ОЗУ или ИУ (внешним устройством).

Шина управления служит для передачи разнообразных управляющих сигналов, определяющих режимы работы памяти (запись или считывание), интерфейсных устройств (ввод или вывод информации) и микропроцессора (запуск, запросы внешних устройств на обслуживание, информация о текущем режиме работы и другие сигналы). Разрядность шины данных обычно соответствует разрядности операндов, обрабатываемых микропроцессором. Поэтому чаще всего шина данных содержит 8, 16 или 32 линии для передачи соответствующих разрядов данных и команд. В ряде последних моделей микропроцессоров используется шина данных с расширенной разрядностью, чтобы обеспечить одновременную передачу нескольких команд или операндов. Например, 32-разрядные микропроцессоры Pentium имеют 64-разрядную шину данных.

Разрядность шины адреса определяет максимальный объем адресуемой процессором внешней памяти. Например, 16-разрядная шина адреса обеспечивает адресацию памяти объемом до 64 Кбайт, а 32-разрядная шина – до 4 Гбайт. Процессоры Pentium II, Pentium III, Pentium IV имеют 36-разрядную шину адреса, обеспечивающую обращение к памяти объемом до 64 Гбайт. Отметим, что в ряде микропроцессоров, например в Pentium, вместо младших разрядов адреса формируются сигналы выборки соответствующих байтов (сигналы байтной выборки BEi, где i — номер байта), которые позволяют организовать хранение байтов в отдельных банках памяти.

Во многих микропроцессорных системах передача адреса и данных сопровождается посылкой контрольных битов четности, которые обеспечивают выявление возможных ошибок, возни-

кающих в процессе обмена. При этом обычно реализуется побайтный контроль четности, при котором каждый байт адреса/данных сопровождается дополнительным (9-м) контрольным битом, поступающим на отдельный вывод микропроцессора.

В некоторых системах для уменьшения числа необходимых линий связи и соответствующих выводов и контактов используется *мультиплексирование линий адреса и данных.* В таких системах для передачи адреса и данных используются одни и те же линии связи, на которые сначала выдается адрес, а затем поступают данные. Например, 16-разрядные микроконтроллеры семейства MCS-196, выпускаемые компанией Intel, имеют мультиплексированную 16-разрядную шину адреса\данных. Обмен информацией по мультиплексированной шине требует введения отдельного регистра для хранения адреса в процессе пересылки данных. При этом требуется также дополнительное время для реализации обмена, что несколько снижает производительность системы.

Разрядность шины управления определяется организацией работы системы, возможностями реализации различных режимов ее функционирования, используемыми методами контроля микропроцессора и других устройств. Поэтому набор передаваемых по шине управляющих сигналов является индивидуальным для каждой модели микропроцессора. Имеется ряд управляющих сигналов, которые используются в большинстве микропроцессорных систем. К ним относятся сигналы начального запуска (RESET), сигналы, задающие режим работы памяти (чтение — RD, запись — WR), сигналы, необходимые для реализации прерываний и ряд других. В простых системах для передачи управляющих сигналов может использоваться всего несколько линий, а в сложных системах число этих линий составляет несколько десятков.

В процессе функционирования микропроцессорной системы реализуются следующие основные режимы ее работы:

– выполнение основной программы;

– вызов подпрограммы;

– обслуживание прерываний и исключений;

– прямой доступ к памяти.

Рассмотрим основные принципы реализации этих режимов.

## 2.2. Принцип реализации выполнения программы

В режиме выполнения основной программы процессор выбирает из памяти очередную команду программы и выполняет соответствующую операцию. Команда представляет собой многоразрядное двоичное число, которое состоит из двух частей (полей) — кода операции (КОП) и кода адресации операндов (КАД). Код операции КОП задает вид операции, выполняемой данной командой, а код адресации КАД определяет выбор операндов (способ адресации), над которыми производится заданная операция. В зависимости от типа микропроцессора команда может содержать различное число разрядов (байтов). Например, команды процессоров Pentium содержат от 1 до 15 байтов, а большинство процессоров с RISC-архитектурой использует фиксированный 4-байтный формат для любых команд.

Для хранения адреса очередной команды служит специальный регистр процессора – *программный счетчик PC* (Program Counter), содержимое которого автоматически увеличивается на 1 после выборки следующего байта команды. Таким образом, обеспечивается последовательная выборка команд в процессе выполнения программы. При выборке очередной команды содержимое PC поступает на шину адреса, обеспечивая считывание из ОЗУ следующей команды выполняемой программы. При реализации безусловных или условных переходов (ветвлений) или других изменений последовательности выполнения команд выполняется загрузка в PC нового содержимого, в результате чего производится переход к другой ветви программы или подпрограмме.

В процессорах Pentium и предыдущих моделях микропроцессоров компании «Intel» (8086, 80186, 80286, 386, 486 и ряде других) реализуется сегментная организация памяти. При этом адрес выбираемой команды определяется содержимым двух регистров – сегментного регистра CS, который задает начальный (базовый) адрес этого сегмента и указателя команды EIP, указывающего положение команды (ее смещение) в сегменте программ. Регистры IP и CS выполняют функции программного счетчика PC, и различные виды передачи управления в программе реализуются путем изменения их содержимого.

Принятая из ОЗУ команда поступает в регистр команд, входящий в состав УУ процессора. Затем производится дешифрация команды, в процессе которой определяется вид выполняемой операции (расшифровка КОП) и формируется адрес необходимых операндов (расшифровка КАД). В соответствии с кодом поступившей команды УУ процессора генерирует *последовательность микрокоманд,* обеспечивающих выполнение заданной операции. Каждая микрокоманда выполняется в течение одного машинного такта — периода тактовых импульсов, задающих рабочую частоту всех внутренних узлов и блоков микропроцессора. Таким образом, тактовая частота микропроцессора определяет время выполнения отдельных микрокоманд, последовательность которых обеспечивает получение необходимого результата операции (поступившей команды).

Для выполнения каждой поступившей команды требуется определенное количество командных циклов и тактов. *Командным циклом* называется промежуток времени, требуемый для выполнения обращения к ОЗУ или внешнему устройству с помощью системной шины. Обычно реализация такого цикла занимает от 2 до 4 *системных тактов* (периодов синхросигналов шины), которые требуются для установки требуемого адреса, выдачи сигналов, определяющих вид цикла – чтение или запись, получения сигнала готовности к обмену (от памяти или внешних устройств) и собственно передачи данных или команд. При современной технологии изготовления системных плат частота синхросигналов шины обычно составляет десятки мегагерц (типичные значения 25, 33, 50, 66, 75,100, 133 МГц).

При выполнении каждой команды в первых циклах производится ее выборка из памяти по адресу, который задается содержимым программного счетчика PC. Последующая дешифрация выбранной команды определяет необходимое число циклов для ее последующего выполнения. Если для выполнения команды не требуется считывание операндов из памяти (внешних устройств) или запись в память (вывод на внешние устройства) результатов операции, то такая команда выполняется за один цикл. При считывании операндов из памяти (внешних устройств) или записи результата в память (вывод на внешние устройства) требуется выполнение дополнительных циклов чтения (ввода) или

записи (вывода). В зависимости от разрядности обрабатываемых операндов и разрядности используемой системной шины число циклов, необходимых для выполнения команд, может быть различным: от 1 (выборка команды) до 4 – 5 (зависит от команды, разрядности шин и операндов).

*Машинным (процессорным) тактом* в микропроцессорных системах является длительность периода тактовых сигналов Tt, которая задается тактовой частотой микропроцессора Ft = 1/Tt. При выполнении операций, не требующих обращений к системной шине, эта частота определяет производительность микропроцессора. Для современных микропроцессоров частота Ft достигает 1 ГГц и более (последние модели микропроцессоров Pentium, AMD K7, Alpha и другие). Таким образом, обработка информации внутри процессора (без обращения к системной шине) производится значительно быстрее, чем обмен по шине. Если тактовая частота микропроцессора отличается от частоты обмена по системной шине, то вывод данных на шину реализуется с помощью промежуточной буферной памяти, в которой хранятся данные, посылаемые микропроцессором на системную шину. Данные выбираются из буферной памяти и поступают на системную шину с частотой, соответствующей скорости обмена по этой шине.

Текущее состояние процессора при выполнении программы определяется содержимым *регистра состояния SR* (State Register). В микропроцессорах Pentium данный регистр называется EFLAGS, в микроконтроллерах семейства MSC-51 – это регистр PSW, в AVR — STATUS. Этот регистр содержит биты управления, задающие режим работы процессора, и биты признаков (флаги), указывающие характеристики результата выполненной операции:

N – признак знака (старший бит результата), N = 0 – при положительном результате, N=1 – при отрицательном результате;

C – признак переноса, C = 1, если при выполнении операции образовался перенос из старшего разряда результата;

V – признак переполнения, V = 1, если при выполнении операций над числами со знаком произошло переполнение разрядной сетки процессора;

Z – признак нуля, Z=1, если результат операции равен нулю.

Некоторые микропроцессоры фиксируют также другие виды признаков: признак четности результата, признак переноса между тетрадами младшего байта (например, для МК51 это флаг AC). Специальные виды признаков устанавливаются по результатам операций над числами, представленными в формате с «плавающей точкой».

### 2.3. Вызов подпрограммы.

Обращение к подпрограмме реализуется при поступлении в микропроцессор специальной команды CALL (в некоторых процессорах эта команда имеет мнемоническое обозначение JSR — Jump-to-SubRoutine), которая указывает адрес первой команды вызываемой подпрограммы. Этот адрес загружается в PC, обеспечивая в следующем командном цикле выборку первой команды подпрограммы. Предварительно выполняется процедура сохранения в специальном регистре или ячейке памяти (в стеке) текущего содержимого PC, где хранится адрес следующей команды основной программы, чтобы обеспечить возвращение к ней после выполнения подпрограммы. Возврат к основной программе реализуется при поступлении команды RETURN (мнемоническое обозначение RET), завершающей подпрограмму. По этой команде сохранявшееся содержимое PC снова загружается в программный счетчик, обеспечивая выполнение команды, которая в исходной программе следовала за командой CALL.

Особенность этой процедуры состоит в том, что большинство МП обеспечивают возможности *вложения под-программ,* т. е. реализуют при выполнении подпрограммы вызов новой подпрограммы с последующим возвращением к предыдущей подпрограмме. При вложении нескольких подпрограмм требуется сохранение нескольких промежуточных значений содержимого PC и последовательная загрузка этих значений в PC при возврате к предыдущим подпрограммам и к основной программе.

Для реализации этой процедуры используется стек – специальная память магазинного типа, работающая по принципу «последний пришел — первый ушел» (стек типа LIFO—«Last In-First Out»). Существуют различные варианты реализации стека.

Регистровый стек реализуется с помощью реверсивных сдвиговых регистров (их число строго ограничено). Каждая команда CALL вызывает ввод в стек очередного содержимого PC. По команде RETURN направление сдвига изменяется и производится извлечение из стека последнего поступившего содержимого PC. Таким образом обеспечивается выполнение вложенных подпрограмм. Возможное число вложенных подпрограмм определяется глубиной стека, т. е. разрядностью используемых регистров сдвига. Если число вложений превышает глубину стека, первые из введенных в стек значений PC теряются, т. е. возврат к основной программе не будет обеспечен. Поэтому при использовании регистрового стека необходим строгий контроль за числом вложений. Такая реализация стека применяется в системах, решающих задачи с ограниченным числом вложенных подпрограмм (обычно не более 10 – 20).

Значительно более широкие возможности вложения подпрограмм обеспечивает реализация стека в ОЗУ. В этом случае часть ОЗУ выделяется для работы в качестве стека. Адресация к ячейкам стека производится с помощью специального регистра-указателя стека SP (Stack Pointer), который вводится в состав УУ процессора. Регистр SP содержит адрес верхней заполненной ячейки стека, в которой хранится значение PC, записанное при выполнении команды CALL. При поступлении новой команды CALL содержимое SP автоматически уменьшается на 1, адресуя следующую, еще незаполненную ячейку стека. Полученный адрес SP–1 выдается на шину адреса, а на шину данных поступает содержимое PC, которое должно сохраняться в стеке. Таким образом, производится последовательное заполнение ячеек стека «снизу-вверх», при этом SP всегда адресует вершину стека. По команде RETURN текущее содержимое SP выдается на шину адреса, и по шине данных производится считывание с вершины стека последнего записанного значения PC. После этого содержимое SP увеличивается на 1, адресуя предыдущее значение PC, хранящееся в стеке. Так как ОЗУ обычно имеет значительный объем, то для размещения стека можно выделить достаточно большое количество ячеек памяти, обеспечивая необходимый уровень вложения подпрограмм.

## 2.4. Обслуживание прерываний и исключений.

При работе микропроцессорной системы часто возникают ситуации, когда требуется прервать выполнение текущей программы и перейти к подпрограмме, обеспечивающей необходимую реакцию системы на создавшиеся обстоятельства. Такие ситуации называются *прерываниями* или *исключениями* в зависимости от причин, вызывающих их возникновение.

*Прерываниями* (interuption) являются ситуации, возникающие при поступлении соответствующих команд (программные прерывания) или сигналов от внешних устройств (аппаратные прерывания).

*Исключениями* (exception) являются нештатные ситуации (ошибки), возникающие при работе процессора. При выявлении таких ошибок соответствующие блоки, контролирующие работу процессора, вырабатывают внутренние сигналы запроса, обеспечивающие вызов необходимой подпрограммы обслуживания. Классификация прерываний и исключений приведена на рис. 2.2.

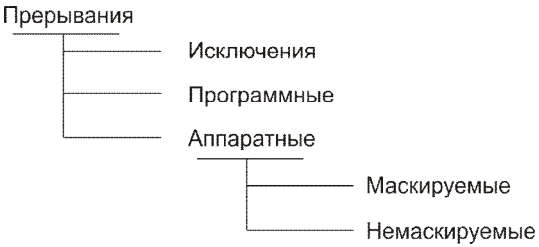


Рисунок 2.2. – Классификация прерываний и исключений

Во всех этих ситуациях микропроцессор завершает выполнение очередной команды и заносит в стек текущее содержимое программного счетчика PC, которое является адресом возврата к прерванной программе после реализации подпрограммы обслуживания, и содержимое регистра состояний SR. Если запрос прерывания поступает от внешнего устройства, то процессор формирует сигнал подтверждения прерывания, который информирует это устройство, что начато обслуживание данного запроса. Затем в PC загружается из памяти вектор прерывания – начальный адрес соответствующей подпрограммы обслуживания.

Эти вектора являются входами в подпрограммы обслуживания и хранятся в таблице векторов прерываний, которая обычно записывается в ОЗУ. Размер таблицы зависит от числа типов обслуживаемых прерываний и исключений. В простейших микропроцессорах это число составляет несколько единиц, а для микропроцессоров семейства Pentium или MC68000 обеспечивается возможность реализации до 256 различных подпрограмм обслуживания.

Завершается подпрограмма обслуживания специальной командой возврата из прерывания IRET, которая выбирает из стека хранившееся содержимое PC и SR и загружает его обратно в эти регистры, обеспечивая возвращение к выполнению прерванной программы.

Программные прерывания реализуются при поступлении специальных команд (INTn для микропроцессоров Pentium, TRAPn для микропроцессоров семейства MC68000 и другие). Эти команды вызывают переход к выполнению стандартных подпрограмм обслуживания, для размещения которых выделяются определенные позиции в ОЗУ. Таким образом, при вызове подпрограмм обслуживания реализуется обращение к фиксированным адресам.

Причинами аппаратных прерываний являются запросы от различных внешних (периферийных) устройств системы. Эти запросы поступают на внешние выводы микропроцессора или формируются периферийными устройствами, размещенными на одном кристалле с процессором. Аппаратные прерывания могут быть *маскируемые* или *немаскируемые.*

Запросы маскируемых прерываний обслуживаются только в том случае, если соответствующий бит управления в регистре состояния, который называется *маской прерываний,* имеет значение 1. (Например, в микроконтроллерах семейства МК51 – это бит EA в регистре IE). С помощью специальных команд значение этого бита может быть установлено в 1 или сброшено в 0. Таким образом, можно разрешить или запретить обслуживание поступивших аппаратных прерываний при выполнении определенных программ или их фрагментов. При одновременном поступлении нескольких запросов обслуживание реализуется в соответствии с их приоритетом. В ряде микропроцессорных сис-

тем для обеспечения приоритетного обслуживания запросов от многих внешних устройств включаются специальные микросхемы – *контроллеры прерываний.* Некоторые типы микропроцессоров имеют внутренние контроллеры для организации приоритетных прерываний.

Немаскируемые запросы прерывания обслуживаются в первоочередном порядке и не могут быть маскированы. Обычно микропроцессор имеет один вход для подачи немаскируемых запросов, которые формируются при возникновении каких-либо аварийных ситуаций. Чаще всего этот вход используется для контроля напряжения питания. Если напряжение питания выходит за допустимые пределы, то специальный датчик вырабатывает немаскируемый запрос прерывания, поступающий в микропроцессор. При этом источник питания должен некоторое время (порядка 10 мс) сохранять необходимый уровень напряжения питания, в течение которого микропроцессор выполняет подпрограмму перезаписи на магнитный диск информации, достаточной для продолжения прерванной программы после восстановления нормального режима питания.

Причинами исключений могут быть различные ошибки и нештатные ситуации, возникающие при работе системы. Различные типы микропроцессоров контролируют разные варианты такого рода ситуаций. Типичными причинами исключений являются, например, использование нулевого делителя при выполнении команды деления (деление на 0); выборка неправильного кода команды; выход за границы разрешенного фрагмента памяти; поступление команд, выполнение которых запрещено при данном режиме функционирования микропроцессора и ряд других. Соответствующие причины исключений будут рассмотрены при описании конкретных типов микропроцессоров.

### 2.5. Прямой доступ к памяти.

Прямой доступ к памяти DMA (Direct Memory Access) используется, если необходимо произвести пересылку значительного массива информации между ОЗУ и каким-либо внешним устройством, которое подает в систему соответствующий запрос. Реализация такой пересылки с помощью соответствующей

программы обмена требует выполнения отдельной команды пересылки для передачи каждого байта или слова. При этом необходим определенный объем памяти для хранения программы и требуется значительное время для ее выполнения.

В большинстве современных микропроцессорных систем пересылка массивов информации обеспечивается с помощью специальных устройств — контроллеров DMA, которые реализуют режим прямого доступа к памяти. При поступлении запроса от внешнего устройства контроллер выдает соответствующий сигнал микропроцессору. Получив этот сигнал, микропроцессор завершает очередной цикл обмена по системной шине и отключается от нее, то есть переводит свои выводы, подключенные к шинам данных и адреса и линиям управления ОЗУ и внешними устройствами, в отключенное (высокоимпедансное) состояние. При этом микропроцессор выдает контроллеру DMA сигнал разрешения на реализацию прямого доступа. Получив этот сигнал, контроллер принимает на себя управление системой. Он выдает на адресную шину адреса ячеек ОЗУ, с которыми выполняется текущий цикл обмена, формирует необходимые сигналы, определяющие режим работы ОЗУ (запись или считывание) и интерфейсного устройства, через которое производится пересылка информации (ввод или вывод).

Передача сигналов запроса и подтверждения прямого доступа к памяти между микропроцессором и контроллером DMA производится по соответствующим линиям шины управления.

Предварительно контроллер DMA программируется для выполнения указанных функций. В него вводятся начальные адреса массивов памяти в ОЗУ, с которых начинается процесс обмена, и размеры массивов, подлежащего пересылке. Обычно контроллер DMA обслуживает запросы от нескольких внешних устройств, поэтому он программируется на реализацию определенного приоритета их обслуживания в случае одновременного поступления нескольких запросов. Программирование контроллера производится путем посылки ему необходимых управляющих сообщений. Эти сообщения обычно предварительно вводятся в контроллер от микропроцессора, когда он выполняет специальную программу инициализации контроллера DMA.

# 3. 8-РАЗРЯДНЫЕ МИКРОКОНТРОЛЛЕРЫ

## 3.1. Типовая модульная структура микроконтроллера

В настоящее время выпускается большое количество разнообразных по структуре и функциям микроконтроллеров для применения в различных системах. Эта номенклатура постоянно расширяется, чтобы обеспечить решение специфических задач в различных прикладных задачах. Возможность разработки и производства новых моделей в сжатые сроки обеспечивает *модульный принцип структурной организации.* При модульном принципе построения все процессоры одного семейства содержат в себе одинаковый базовый функциональный блок — процессорное ядро, и изменяемый функциональный блок, который отличает МК разных моделей в пределах одного семейств (рис 3.1).

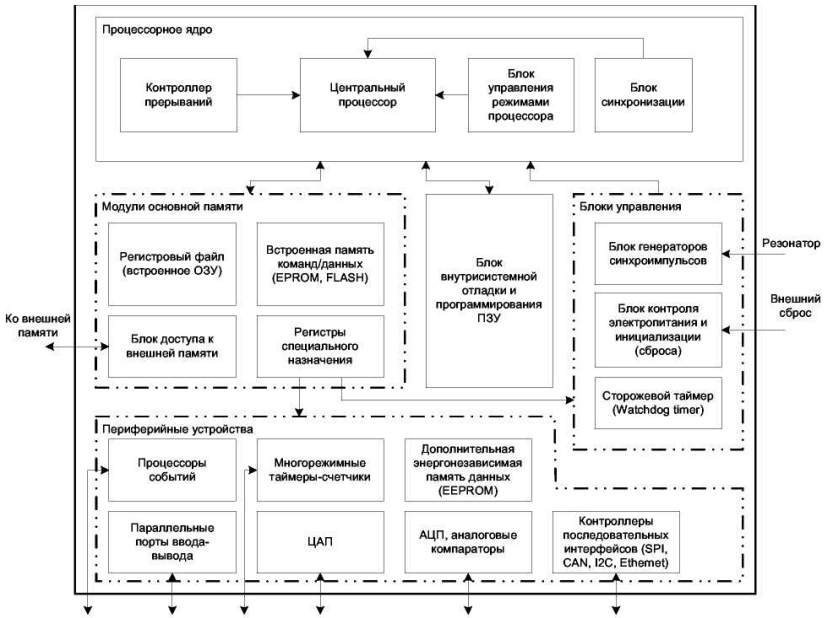


Рисунок 3.1. – Типовая модульная структура микроконтроллера

Базовый блок (процессорное ядро) включает:

– центральный процессор;

– внутренние магистрали адреса, данных и управления;

– блок формирования множества сигналов с различными фазами и частотами для синхронизации центрального процессора и внутренних магистралей;

– блок управления режимами работы процессора, который может настраивать процессор на следующие режимы: активный режим, режим обработки прерываний, несколько режимов пониженного энергопотребления, режим рестарта;

Процессорное ядро является основным отличительным признаком архитектуры определенного семейства процессоров, поэтому его (ядро) называют по названию семейства. Например, ядро MCS-51, ATMega, ATTiny семейства AVR (Atmel), PIC12, PIC16 семейства PIC (Microchip), семейство HC08 (Motorola).

Изменяемый функциональный блок включает:

– модули памяти различных типов: оперативную память данных типа SRAM; постоянную память команд (программ) типов ROM, EPROM или FLASH; энергонезависимую память данных типа EEPROM;

– модули периферийных устройств;

– модули управления и синхронизации.

В различных микросхемах семейства может иметься различный набор модулей изменяемого функционального блока. Общую совокупность модулей, реализованных в микросхемах одного семейства, называют *библиотекой периферийных модулей* данного семейства. В данную библиотеку входят, как говорилось, не только периферийные, но и модули памяти, встроенные генераторы синхронизации, блок контроля электропитания и формирования сигналов рестарта системы в случае сбоев или «внешнего сброса», модули внутрисхемной отладки и программирования. В последнее время активно развивается направление «System-On-Chip», когда конечный пользователь сам может формировать структуру специализированного процессора из предоставленной библиотеки периферийных модулей, а также самостоятельно разрабатывать новые модули.

Библиотека каждого современного семейства микроконтроллеров включает модули пяти функциональных групп:

1) модули памяти;

2) модули периферийных устройств;

3) модули встроенных генераторов синхронизации;

4) модули контроля за напряжением питания и ходом выполнения программы;

5) модули внутрисхемной отладки и программирования.

Термин «модуль памяти» в применении к микроконтроллеру стал использоваться на этапе перехода к новым технологиям резидентной памяти программ и данных. Энергонезависимая память типа FLASH и EEPROM имеет не только режимы хранения и чтения информации, которая была в нее записана до начала эксплуатации изделия на этапе программирования, но и режимы стирания и программирования под управлением прикладной программы. Вследствие этого энергонезависимая память типа FLASH и EEPROM требует управления режимами работы, для чего снабжена дополнительными схемами управления. Массив ячеек памяти, доступных для чтения, стирания и записи информации, дополнительные аналоговые и цифровые схемы управления, а также регистры специальных функций для задания режимов работы объединены в функциональный блок, который и носит название модуля памяти. В настоящее время термин «модуль памяти» используется в равной мере для всех типов резидентной памяти: ОЗУ и ПЗУ.

В направлениях развития 8-разрядной элементной базы МК отчетливо прослеживается тенденция к закрытой архитектуре, при которой линии внутренних магистралей адреса и данных отсутствуют на выводах корпуса МК. И, как следствие, не представляется возможность использования внешних по отношению к МК БИС запоминающих устройств. В этом случае разработчик изделия на МК при выборе элементной базы должен позаботиться о том, чтобы предполагаемый алгоритм управления, реализованный в виде прикладной программы, сумел разместиться в резидентной (внутренней) памяти МК. В противном случае придется сменить элементную базу и перейти к МК с большим объемом внутреннего ПЗУ. Для подобных случаев разработчики элементной базы МК обычно предлагают ряд модификаций МК

с одним и тем же набором периферийных модулей и различающимся объемом резидентной памяти программ и данных (например, ATMega16 и ATMega32 – 16 и 32 кБайта ПЗУ соответственно).

Группа модулей периферийных устройств включает следующие основные типы:

– параллельные порты ввода/вывода;

– таймеры-счетчики, таймеры периодических прерываний, процессоры событий;

– контроллеры последовательного интерфейса связи нескольких типов (UART, SPI, 1-Wire, $I^2C$, CAN, USB);

– аналого-цифровые преобразователи (АЦП);

– цифроаналоговые преобразователи (ЦАП);

– ШИМ-контроллеры;

– контроллеры клавиатуры, ЖК-индикаторов и светодиодной матрицы.

Возможны также некоторые другие типы модулей, например, модуль прямого доступа к памяти, модуль управления ключами силовых инверторов напряжения, модуль генератора DTMF для тонального набора номера в телефонии и т. п.

Существенное изменение претерпели в настоящее время генераторы синхронизации 8-разрядных МК. Произошло функциональное разделение собственно генератора синхронизации, который выделился в отдельный модуль, и схемы формирования многофазной последовательности импульсов для тактирования центрального процессора и межмодульных магистралей, которая является неотъемлемой частью процессорного ядра. Появилась возможность выбора внешнего времязадающего элемента: кварцевый или керамический резонатор, RC-цепь. Поскольку схемотехника выполнения усилителей с положительной обратной связью определяется типом времязадающего элемента, то для одного и того же МК появились разные модификации модулей встроенного генератора синхронизации. Повышение производительности процессорного ядра МК связано с повышением частоты тактирования центрального процессора и межмодульных магистралей. Однако применение высокочастотных кварцевых резонаторов в качестве времязадающего элемента повышает уровень электромагнитного излучения, т. е. возрастает интенсив-

ность генерации помех. Поэтому все чаще генераторы синхронизации имеют в своем составе умножитель частоты с программно настраиваемым коэффициентом. Умножитель частоты часто выполняется по схеме синтезатора с контуром фазовой автоподстройки (англоязычная аббревиатура PLL — Phase Loop Lock). Цепи синтезатора частоты и регистры специальных функций для управления режимами его работы объединены в один из модулей генератора синхронизации.

Относительно новыми для 8-разрядных МК являются модули контроля за напряжением питания и ходом выполнения программы. Они осуществляют диагностику некоторых подсистем МК и позволяют восстановить работоспособность устройства на основе МК при нарушениях программного характера, сбоях в системе синхронизации, снижении напряжения питания.

Очень удобным нововведением стали модули внутрисхемной отладки и программирования. Они являются аппаратной основой режимов отладки и программирования в системе, которые позволяют отлаживать прикладную программу и заносить коды программы в энергонезависимую память МК прямо на плате конечного изделия, без использования дополнительных аппаратных средств отладки и программирования (ISP — In System Programming).

### 3.2. Процессорное ядро микроконтроллера

Техническое решение процессорного ядра определяют следующие параметры:

*Архитектурные* — набор регистров, организация памяти, способы адресации операндов в памяти, система команд для обработки этих данных.

*Схемотехнические решения* — схемы регистров, АЛУ, схемы управления магистралями и т.п. Схемотехника определяет также внутреннюю диаграмму функционирования – последовательность перемещения данных по магистралям между регистрами, памятью, АЛУ.

*Технология производства* – определяет допустимую сложность схемы, максимальную частоту переключений, энергопотребление.

Ядро современных 8-разрядных МК реализуют как на основе CISC-архитектуры — это МК семейств HC05, HC11, HC08 фирмы «Motorola», семейства MCS-51 фирм «Intel», «Atmel», «Philips», МК семейства C500 фирмы «Infineon», — так и на основе RISC-архитектуры — семейства PIC16, PIC17, PIC18 фирмы «Microchip», семейство AVR фирмы «Atmel», семейство SX фирмы «Scenix».

В приложении к 8-разрядным МК микропроцессор с CISC-архитектурой имеет однобайтовый, двухбайтовый и трехбайтовый (иногда четырехбайтовый) формат команд. Выборка команды из памяти осуществляется побайтно в течение нескольких машинных циклов. Время выполнения каждой команды с учетом времени выборки в большинстве случаев составляет от 1 до 10 циклов. Длительность машинного цикла равна периоду частоты тактирования внутренних магистралей микроконтроллера $f_{BUS}$. Максимально допустимое значение частоты $f_{BUS}$ является одной из важнейших характеристик процессорного ядра, так как чем больше $f_{BUS}$, тем выше его производительность. Следует особо обратить внимание, что для МК с CISC-архитектурой частота тактирования внутренних магистралей МК $f_{BUS}$ всегда в несколько раз меньше предельно допустимой частоты кварцевого резонатора, который используется в качестве времязадающего элемента встроенного генератора.

Микроконтроллер с RISC-архитектурой имеет формат команды фиксированной длины: например, 12, 14 или даже 16 бит для МК с 8-разрядным форматом обрабатываемого слова. Выборка из памяти и исполнение подавляющего большинства команд осуществляются за один машинный цикл МК, т. е. один период $f_{BUS}$ — одна команда. Однако и для МК с RISC архитектурой частота $f_{BUS}$ не всегда совпадает с частотой подключаемого кварцевого резонатора.

Производительность микропроцессоров и МК в том числе принято оценивать числом элементарных операций, которые могут быть выполнены в течение одной секунды. Единица измерения производительности — миллион операций в секунду (MIPS – Million Instructions Per Second). Для расчета численного значения производительности в MIPS принято использовать время выполнения команды пересылки «регистр-регистр». Эта

команда присутствует в перечне инструкций Ассемблера любого микропроцессора и имеет минимальное время выполнения.

*Производительность (MIPS) = 1/$t_{команды}$(мкс).*

На практике в качестве косвенного параметра для оценки производительности МК используют предельную частоту тактирования, т. е. частоту времязадающего элемента генератора синхронизации $f_{XCLK}$. Именно эта частота обычно указана в справочных данных 8-разрядного МК. Однако использовать ее для прямого расчета производительности в большинстве случаев нельзя. Дело в том, что длительность машинного цикла центрального процессора определяется частотой обмена по внутренним магистралям адреса и данных $f_{BUS}$. Соотношение $f_{XCLK}$ и $f_{BUS}$ индивидуально для каждого процессорного ядра МК. Так для «Intel» MSC-51 $f_{XCLK}/f_{BUS}$ = 12, для «Microchip» PIC16 $f_{XCLK}/f_{BUS}$ = 4, для AVR «Amtel» $f_{XCLK}/f_{BUS}$ = 1. В МК «Motorola» HC08 тактирование осуществляется с использованием умножителя частоты и $f_{BUS} > f_{XCLK}$ Поэтому при сравнении производительности различных МК следует сопоставлять максимальную частоту тактирования межмодульных магистралей $f_{BUS}$, а не приведенную в паспортных данных $f_{XCLK}$. Численные значения $f_{BUS}$ для популярных семейств 8-разрядных МК приведены в табл. 3.1.

По определению микроконтроллеры с RISC-архитектурой должны иметь более высокую производительность по сравнению с CSIC микроконтроллерами при одной и той же частоте внутренней магистрали $f_{BUS}$, так как первые выполняют каждую команду за один машинный цикл, а последние – за несколько. Для МК с RISC-архитектурой время выполнения любой операции составляет $1/f_{BUS}$, следовательно, их производительность (в MIPS) равна $f_{BUS}$(в МГц). Например, производительность МК PIC16 составляет 5 MIPS, МК AVR — 20 MIPS. В МК с CISC-архитектурой число циклов выполнения операции «регистр-регистр» составляет от 1 до 3, что снижает производительность.

Таблица 3.1.

Частота тактирования межмодульных магистралей 8-разрядных МК

| Семейство МК | Архитектура ядра | $f_{maxBUS}$ (МГц) | $f_{XCLK}$ /$f_{BUS}$ | $f_{maxXCLK}$ (МГц) |
|---|---|---|---|---|
| *МК с ядром MSC-51* | | | | |
| Intel MSC-51 | CISC | 2 | 12 | 24 |
| Atmel 89C55 | CISC | 2,75 | 12 | 33 |
| Infineon C500 | CISC | 3,3 | 12 | 40 |
| Pfilips 89C51/52 | CISC | 2,75 | 12 | 33 |
| Pfilips 8051XA | CISC | 20* | — | 30 |
| Dallas High Speed | CISC | 8,25 | 4 | 33 |
| *МК с другими типами процессорного ядра* | | | | |
| Motorola HC05 | CISC | 2** | 2 | 4 |
| Motorola HC08 | CISC | 8 | 4 или < 1*** | 32 |
| Motorola HC11 | CISC | 4 | 4 | 16 |
| Microchip PIC16 | RISC | 5 | 4 | 20 |
| Microchip PIC17 | RISC | 8,25 | 4 | 33 |
| Microchip PIC18 | RISC | 10 | 4 | 40 |
| Atmel AVR | RISC | 20 | 1 | 20 |
| Scenix | RISC | 50 | 1 | 50 |

* Указана эквивалентная частота, повышение быстродействия связано с коренной переработкой ядра MSC-51.
** Для версий с повышенным быстродействием — 4 МГц.
*** В МК HC08 реализованы два режима тактирования: от генератора кварцевого резонатора, тогда $f_{XCLK}/f_{BUS} = 4$, или от умножителя частоты, тогда $f_{BUS} > f_{XCLK}$

Однако такая оценка производительности является общей. Она не учитывает особенности алгоритмов управления, используемых в каждой конкретной области применения. Так, при реализации быстродействующих регуляторов основное внимание следует уделять времени выполнения операций умножения и деления, которые требуются при реализации уравнений различных передаточных функций. А при реализации кнопочной станции кабины лифта следует оценивать время выполнения только

логических функции, которые используются при опросе клавиатуры и при генерации протокола последовательного интерфейса связи с контроллером управления движения, который оптимизирует перемещение между этажами сразу нескольких кабин лифта. В задачах оптимального управления по таблицам, которые характерны для устройств силовой электроники, на первый план выходит возможность быстрого перебора больших таблиц данных. Поэтому в критических ситуациях, связанных с требованиями высокого быстродействия, следует оценивать производительность на основе тех операций, которые преимущественно используются в алгоритме управления и имеют ограничение по времени выполнения.

В задачах управления объектом в реальном времени существует еще один очень важный фактор производительности, который никак не отображается числом операций в секунду. Это время перехода на подпрограмму прерывания по запросу внешнего устройства или периферийного модуля. В процессе перехода на подпрограмму прерывания каждый МК должен:

– распознать запрос на прерывание;

– дождаться завершения выполнения текущей команды;

– сохранить программный счетчик PC и некоторые регистры центрального процессора в стеке, загрузить вектор прерывания;

– выполнить некоторые вспомогательные команды;

– приступить к выполнению алгоритма обслуживания устройства, которое вызвало это прерывание.

Суммарное время перехода на подпрограмму прерывания определяется архитектурой процессорного ядра МК и частотой его тактирования.

### 3.3. Модули резидентной памяти микроконтроллера

Термин «модуль памяти» подразумевает объединение собственно массивов ячеек памяти со специальными аналоговыми и цифровыми схемами управления режимами записи-стирания, со схемами (и иногда источниками) электропитания и с регистрами управления режимами.

Закрытая архитектура современных 8-разрядных МК стала реализуемой лишь при условии интеграции на кристалл МК мо-

дулей памяти двух типов: энергонезависимого запоминающего устройства для хранения кодов прикладных программ (ПЗУ) и оперативного запоминающего устройства для хранения промежуточных результатов вычислений (ОЗУ). С момента появления МК технология энергонезависимых запоминающих устройств претерпела множество изменений, которые позволили не только повысить информационную емкость, быстродействие, надежность хранения информации, но и привели к появлению принципиально новых технологий программирования резидентной памяти МК.

С точки зрения пользователей МК следует различать пять типов энергонезависимой резидентной памяти.

### 3.3.1. ПЗУ масочного типа

Информация в ячейки ПЗУ масочного типа (Mask-ROM) записывается на заводе-изготовителе МК с помощью масок и не может быть заменена или «допрограммирована» в области ранее не использованного сегмента памяти. Поэтому МК с таким типом памяти программ следует использовать в изделии только после достаточно длительной опытной эксплуатации этого изделия.

Первые образцы масочных ПЗУ появились в начале 1960-х гг., но даже сегодня ПЗУ масочного типа — самое дешевое и эффективное решение при больших объемах выпускаемой аппаратуры.

Использование МК с масочным ПЗУ экономически становится рентабельным при партии в несколько десятков тысяч штук. Кроме благоприятных экономических аспектов решения с ПЗУ масочного типа имеют и другое преимущество. Они обеспечивают высокую надежность хранения информации по причине программирования в заводских условиях с последующим контролем качества. Недостатки ПЗУ масочного типа очевидны: любое изменение прикладной программы потребует новой серии ИС МК, что может оказаться весьма дорогостоящим и времяемким решением.

### 3.3.2. Однократно программируемые ПЗУ

В незапрограммированном состоянии каждая ячейка памяти модуля однократно программируемого ПЗУ (OTPROM – One-Time Programmable ROM)) при считывании возвращает код $FF. Программированию подлежат только те разряды, которые после программирования должны содержать «0». Если в процессе программирования некоторые разряды какой-либо ячейки памяти были установлены в «0», то восстановить в этих разрядах единичное значение уже невозможно. Поэтому рассматриваемый тип памяти и носит название «однократно программируемые ПЗУ». Однако те разряды, которые в процессе предшествующего сеанса программирования не изменялись, т. е. имеют единичные значения, могут быть подвергнуты программированию в последующем и «доустановлены» в «0». Число возможных сеансов программирования модуля однократно программируемого ПЗУ в составе МК не имеет ограничений. Технология программирования состоит в многократном приложении импульсов повышенного напряжения к элементарным ячейкам адресуемого байта памяти (т. е. к битам), подлежащим программированию. Уровень напряжения программирования, число импульсов и их временные параметры должны в точности соответствовать техническим условиям. В противном случае ячейки памяти могут восстановить единичное значение по прошествии некоторого времени (иногда нескольких лет) или при изменении условий работы. МК с однократно программируемым ПЗУ рекомендуется применять в изделиях, выпускаемых небольшими партиями.

### 3.3.3. ПЗУ, программируемые пользователем с ультрафиолетовым стиранием

ПЗУ данного типа (в англоязычной интерпретации – EPROM (Eraseble Programmable ROM) допускают многократное программирование. Технология программирования близка к технологии однократно программируемых ПЗУ. Перед каждым сеансом программирования для восстановления единичного значения ранее запрограммированных ячеек памяти весь модуль ПЗУ должен быть подвергнут операции стирания при помощи ульт-

рафиолетового облучения. Для этого корпус МК выполнен со специальным стеклянным окном, внутри которого расположена пластина ИС МК. Но если некоторые разряды ячеек памяти должны быть «допрограммированы» с «1» на «0» при неизменном состоянии ранее запрограммированных разрядов, то операция стирания может быть пропущена. Число сеансов стирания /программирования ПЗУ данного типа ограничено и составляет 25–100 раз при условии соблюдения технологии программирования (напряжение, число и длительность импульсов программирования) и технологии стирания (волновой диапазон источника УФ-излучения). МК с ПЗУ данного типа имеют высокую стоимость, поэтому их рекомендуется использовать только в опытных образцах изделий. Например, МК PIC16C505 – однократно программируемый (OTPROM). Для от-ладки программы этого микроконтроллера производителем выпущен специальный микроконтроллер PIC16C505JW. Он содержит память EPROM вместо OTPROM и число циклов перезаписи составляет около 100. В отличие от однократно программируемого варианта МК, который стоит меньше 1 доллара США, стоимость репрограммируемого МК составляет более 15 долларов.

### 3.3.4. ПЗУ, программируемые пользователем с электрическим стиранием

Электрически программируемые и электрически стираемые ПЗУ (EEPROM или $E^2$PROM (Electrically Erasable Programmable ROM) совместили в себе положительные качества рассмотренных выше типов памяти. Во-первых, ПЗУ типа EEPROM программируются пользователем, во-вторых, эти ПЗУ могут быть многократно подвергнуты операции стирания, и, следовательно, многократно программируются пользователем, в-третьих, эти ПЗУ дешевле ПЗУ с ультрафиолетовым стиранием. Максимальное число циклов стирания/программирования ПЗУ типа EEPROM в составе МК обычно равно 10000. Для сравнения тот же тип памяти в автономном корпусе допускает $10^6$ циклов стирания/программирования. Технология программирования памяти типа EEPROM позволяет реализовать *побайтное стирание* и *побайтное программирование,* для чего к выбранной ячейке па-

мяти должно быть приложено относительно высокое напряжение 10 —20 В. Однако допускается также одновременное стирание некоторого количества ячеек памяти с последовательными адресами, т. е. стирание блока памяти.

Несмотря на очевидные преимущества, редкие модели современных МК используют ПЗУ типа EEPROM для хранения программ. Виной тому два обстоятельства. Во-первых, ПЗУ типа EEPROM имеют ограниченную емкость и могут использоваться в качестве резидентной памяти программ только в маловыводных МК с небольшим объемом памяти. Во-вторых, почти одновременно с EEPROM ПЗУ появились ПЗУ типа FLASH, которые обеспечивают близкие пользовательские характеристики, но при этом имеют более низкую стоимость.

### 3.3.5. ПЗУ с электрическим стиранием типа FLASH

Электрически программируемые и электрически стираемые ПЗУ типа FLASH (FLASH ROM) были предназначены для заполнения «ниши» между дешевыми однократно программируемыми ПЗУ большой емкости и дорогими EEPROM ПЗУ малой емкости. ПЗУ типа FLASH сохранили преимущества, присущие EEPROM: возможность многократного стирания и программирования посредством приложения повышенного напряжения. Однако для увеличения объема памяти транзистор адресации каждой элементарной ячейки был удален, что не дает возможности программировать каждый бит памяти отдельно. *Память типа FLASH стирается и программируется страницами или блоками.* Страница, как правило, составляет 8, 16 или 32 байта памяти, блоки могут объединять некоторое число страниц, вплоть до полного объема резидентного ПЗУ МК (до 60 Кбайт). Упрощение декодирующих схем, произошедшее из-за уменьшения числа транзисторов, и, как следствие, снижение стоимости и размеров привело к тому, что МК с FLASH памятью программ в настоящее время становятся конкурентоспособными не только по отношению к МК с однократно программируемым ПЗУ, но и с масочным ПЗУ также.

Первый тип памяти (MASK ROM) предполагает программирование МК только в заводских условиях. Второй и третий типы

памяти (OTPROM и EPROM) могут программироваться непосредственно пользователем, но для программирования требуют подключения источника повышенного напряжения к одному из выводов МК. Для их программирования используются специальные программаторы, в которых требуемая последовательность импульсов программирования с амплитудой 10–25 В создается внешними по отношению к МК средствами. Технология программирования памяти первых трех типов не предполагает изменения содержимого некоторых ячеек энергонезависимой памяти в процессе работы устройства под управлением прикладной программы. Память типа EEPROM и FLASH также требует в процессе стирания/программирования приложения повышенного напряжения. В ранних образцах МК (например, Microchip P1C16C5xx) это напряжение должно было быть подано на один из выводов МК в режиме программирования. В новейших версиях МК (Motorola HC08, Microchip PIC16, Atmel AVR) модули FLASH и EEPROM ПЗУ содержат встроенные схемы усиления, которые называют генераторами накачки. Допускается включение и отключение генератора накачки под управлением программы посредством установки битов в регистрах специальных функций модулей памяти. Т.е. появилась принципиальная возможность осуществить программирование или стирание ячеек памяти FLASH и EEPROM ПЗУ в процессе управления объектом, без останова выполнения прикладной программы и перевода МК в режим программирования. Вспомним разницу между EEPROM и FLASH ПЗУ в составе МК. EEPROM ПЗУ практически никогда не используется для хранения программ, но оно имеет режим побайтного программирования. Предоставленная техническая возможность программирования под управлением прикладной программы становится реализуемой, так как носителем программы в МК является другой модуль памяти. Следовательно, в процессе программирования повышенное напряжение не прикладывается к носителю программы алгоритма программирования, и эта программа может быть выполнена в обычном режиме. Это обстоятельство сделало EEPROM память идеальным энергонезависимым запоминающим устройством для хранения изменяемых в процессе эксплуатации изделия настроек пользователя. В качестве примера достаточно вспомнить со-

временный телевизор или музыкальный центр: настройки каналов сохраняются при отключении питания. Одной из тенденций совершенствования резидентной памяти 8-разрядных МК стала интеграция на кристалл МК сразу двух модулей энергонезависимой памяти: OTP или FLASH — для хранения программ и EEPROM — для хранения перепрограммируемых констант.

Сложнее обстоит дело с возможностью программирования FLASH ПЗУ под управлением прикладной программы. Даже если модуль FLASH ПЗУ содержит встроенный генератор накачки, то попытка перевода модуля в режим программирования установкой битов режима приведет к невозможности дальнейшего считывания программы, которая в это FLASH ПЗУ записана, и МК «зависнет». Поэтому та часть программы, которая реализует программирование FLASH ПЗУ, должна быть обязательно расположена в памяти другого типа. Наиболее часто в качестве такой памяти выбирают ОЗУ МК. Поскольку, если в МК имеется EEPROM ПЗУ, то бессмысленно «допрограммировать» FLASH в процессе работы изделия, в противном случае другой памяти, кроме ОЗУ, в МК просто нет. Если МК допускает возможность выполнения программы, расположенной в ОЗУ, и имеет встроенный генератор накачки модуля FLASH ПЗУ, то такой МК становится «программируемым в системе» (англоязычный термин «In system programmable» — ISP (не путать с SPI)). Для того чтобы возможность программирования в системе стала реализуемой, необходимо предусмотреть пути, по которым в ОЗУ МК будет передана программа алгоритма программирования FLASH ПЗУ, а затем порциями будут передаваться коды прикладной программы, которая должна быть занесена во FLASH ПЗУ. Не следует забывать, что объем сегмента программирования значительно превышает объем резидентного ОЗУ МК. В качестве такого «пути» разработчики МК назначают один из последовательных портов МК. Обслуживание порта реализует специальная программа монитора связи, которая расположена в резидентном масочном ПЗУ МК. Эта программа активизируется установкой определенных линий ввода/вывода МК в указанное в спецификации состояние при сбросе МК или простым обращением к ней. Способ активизации указан в техническом описании МК. По последовательному интерфейсу связи персональ-

ный компьютер загружает в ОЗУ МК сначала коды программы «программирования», а затем порциями коды прикладной программы для программирования. Возможны также решения, при которых программа «программирования» сразу записана в память масочного типа и не требует загрузки в ОЗУ МК.

Рассмотренный режим «программирования в системе» в настоящее время используется для занесения прикладной программы в МК, расположенный на плате конечного изделия. Специальный программатор в этом случае не нужен. Надежность программирования гарантируется внутренними режимами МК и не зависит от схемных решений программатора. Но режим «программирования в системе» отличается от режима «допрограммирования» нескольких байтов FLASH памяти под управлением прикладной программы в процессе работы системы. Теоретически возможно решение, при котором программа «программирования», хранящаяся во FLASH ПЗУ, сначала будет перенесена в ОЗУ под управлением прикладной программы, а затем выполнена из ОЗУ. Но при таком решении на время программирования потребуется запретить все прерывания МК, поскольку их обслуживание невозможно по причине недоступности векторов прерывания. Поэтому реализация описанного режима возможна далеко не всегда. В качестве одного из путей предлагается разбить модуль FLASH ПЗУ на два — с независимыми генераторами накачки и регистрами управления. Такое решение предложено в МК HC908AZ60 фирмы «Motorola». Тогда один из модулей может быть поставлен в режим программирования, в то время как программа «программирования» будет выполняться из другого модуля. Впрочем, следует надеяться, что в недалеком будущем проблема программирования FLASH памяти программой из FLASH памяти будет решена. А пока наиболее совершенные модели МК со свойством «программирования в системе» часто имеют в своем составе четыре типа памяти: FLASH ПЗУ программ, Mask-ROM монитора связи, EEPROM ПЗУ для хранения изменяемых констант и ОЗУ промежуточных данных.

Технология резидентной FLASH памяти МК непрерывно совершенствуется. Одни из лучших показателей достигнуты в МК семейства HC08 фирмы «Motorola»:

– гарантированное число циклов стирания/программирования составляет $10^5$;

– гарантированный период хранения записанной информации равен 10 годам, т. е. составляет жизненный цикл изделия;

– модули FLASH памяти работают и программируются при напряжении питания МК от 1,8 до 2,7 В;

– эквивалентное время программирования 1 байта памяти снижено до 60 мкс, что позволяет выполнить программирование МК с 32 Кб памяти в течение 2 с.

Перспективные технологии FLASH памяти предполагают увеличение скорости программирования до 1 Мбит/с.

### 3.3.6. Статическое ОЗУ

Кроме ПЗУ в состав МК входит также и *статическое оперативное запоминающее устройство* (ОЗУ). Определение «статическое» выделено не случайно: современные 8-разрядные МК допускают снижение частоты тактирования до сколь угодно малых значений с целью снижения энергии потребления. Содержимое ячеек ОЗУ при этом сохраняется в отличие от динамической памяти. В качестве еще одной особенности следует отметить, что многие МК в техническом описании имеют параметр «напряжение хранения информации» — $U_{STANDBY}$. При снижении напряжения питания ниже минимально допустимого уровня $U_{DDmin}$, но выше напряжения хранения $U_{STANDBY}$ программа управления микроконтроллером выполняться не будет, но информация в ОЗУ сохранится. Тогда при восстановлении напряжения питания можно будет выполнить сброс МК и продолжить выполнение программы без потери данных. Уровень напряжения хранения составляет 1 В. Это позволяет в случае необходимости перевести МК на питание от автономного источника (батарейки или аккумулятора) и сохранить тем самым данные ОЗУ. Большого расхода энергии потребления в этом случае не будет, так как система тактирования МК может быть отключена. В последнее время появились МК, которые в корпусе имеют автономный источник питания, гарантирующий сохранение данных в ОЗУ на протяжении 10 лет (например, микроконтроллер DS5000 фирмы «Dallas Semiconductor»).

# 4. ОСНОВНЫЕ ПЕРИФЕРИЙНЫЕ МОДУЛИ МИКРОКОНТРОЛЛЕРОВ

## 4.1. Порты ввода-вывода

### 4.1.1. Общие сведения

Любой микроконтроллер должен иметь некоторое количество внешних линий ввода-вывода, подключенных к внешним выводам микросхемы и называемых внешними портами. Иначе с ним невозможно будет взаимодействовать. 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.

Порты обозначают либо цифрами (P0, P1, P2 и т.д. – МК 51), либо буквами латинского алфавита (PA, PB, PC и т. д. – AVR, PIC). В карте памяти МК каждый порт ввода/вывода представлен регистром данных порта, например DPTx. В режиме ввода логические уровни сигналов на линиях порта Px отображаются нулями и единицами в соответствующих разрядах регистра DPTx. В режиме вывода данные, записанные под управлением программы в регистр DPTx, передаются на выводы МК, которые отмечены в качестве линий порта Px. Обращение к регистру данных DPTx осуществляется теми же командами, что и обращение к ячейкам резидентной оперативной памяти. Кроме того, во многих МК отдельные разряды портов могут быть опрошены как отдельные биты.

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».

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

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 (не изменившегося) 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 проис-

ходит только по стробу, что позволяет защититься от приема данных во время переходного процесса входного сигнала. Пример: порт PSP (Parallel slave port) в микроконтроллере PIC.

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 обоим взаимодействующим сторонам и предотвращает потерю данных, когда одна из них перегружена. Пример порта с квитированием служит порт LPT персонального компьютера. Во встроенных модулях процессоров данный режим чаще всего реализуется программно-аппаратно.

В аналоговых портах (или перестраиваемых портах в аналоговом режиме) используются подключения внешних сигналов к ЦАП, АЦП или аналоговым компараторам, а так же к 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.

### 4.1.2. Однонаправленные дискретные порты ввода

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

Схема однонаправленного порта ввода представлена на рисунке 4.1.

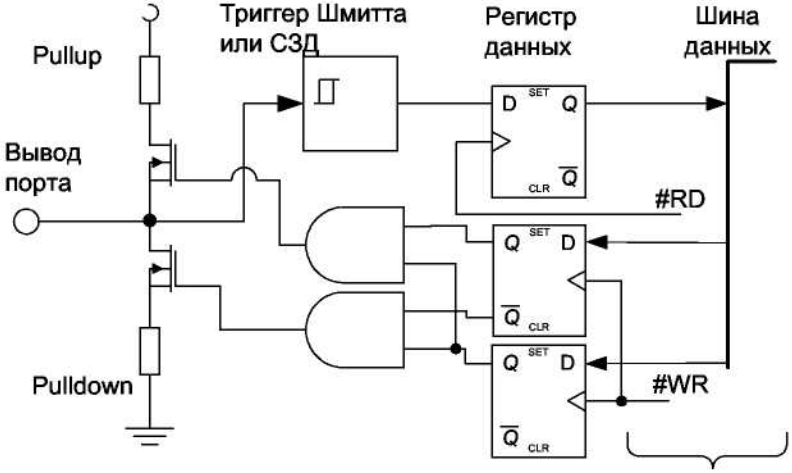


Рисунок 4.1. – Функциональная схема однонаправленного порта ввода

Внешние данные считываются через вывод порта (ножку микросхемы), проходят через триггер Шмидта (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 микроконтроллеров) 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» (Pullup) или логического «0» (Pulldown). 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 неподключенных

к внешним схемам входов («открытых входов»). Через специальные управляющие регистры «схемы поддержки» могут быть отключены полностью или включены в режим Pullup или Pull-down.

Все перечисленные блоки — триггер Шмита, 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.

### 4.1.3. Дискретные порты вывода с двухтактной выходной схемой

Данные порты являются самыми распространенными и реализованы, например, в семействах Atmel AVR, Microchip PICmicro, Motorola HC08, HC11 и многих других.

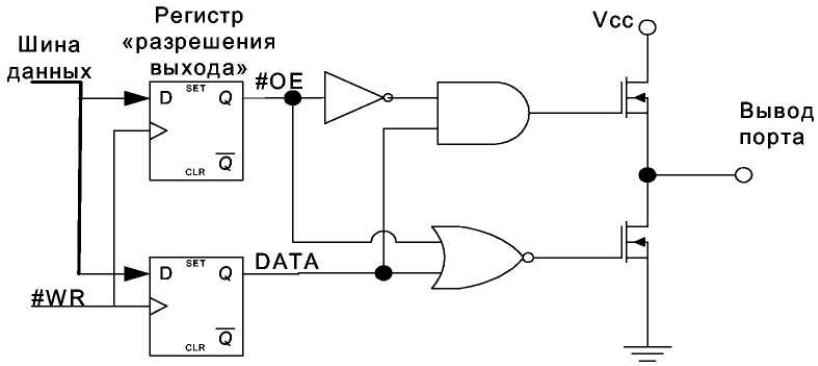


Рисунок 4.2. – Функциональная схема двухтактного порта вывода

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 выходе напряжение равно $V_{CC}$, то есть «1». Если в регистр записано значение «0», то открыт ниж-

ний по схеме транзистор, а верхний закрыт: выход соединен с минусовой шиной питания, то есть там установлен «0».

Верхний по схеме 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: микроконтроллер используется как периферийный контроллер и его выходной порт подключен к периферийной шине другого процессора (мастера), и к шине также подсоединены еще несколько периферийных контроллеров;

– в схемах двунаправленных портов (см. ниже).

Достоинства:

Значительный максимальный втекающий (в состоянии «0») и вытекающий (в состоянии «1») ток выхода: 2..6 мА для каскадов с нормальной нагрузочной способностью (например, Fujitsu MB90) и 5..30 мА для каскадов с повышенной нагрузочной способностью (например, PICmicro, AVR). Встречаются отдельные микросхемы со сверхвысокой нагрузочной способностью — до 60..90 мА (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 особо зависит от скорости переключения. Для ограничения токов в момент переключений иногда используют специальные демпфирующие схемы. Однако они снижают быстродействие портов. Наибольшее применение демпфи-

рующие 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.

### 4.1.4. Дискретные порты вывода с однотактной выходной схемой и внутренней нагрузкой

Данные порты широко применяются, например, в семействе MCS-51. Они имеют более простую внутреннюю схему, представленную на рис 4.3.

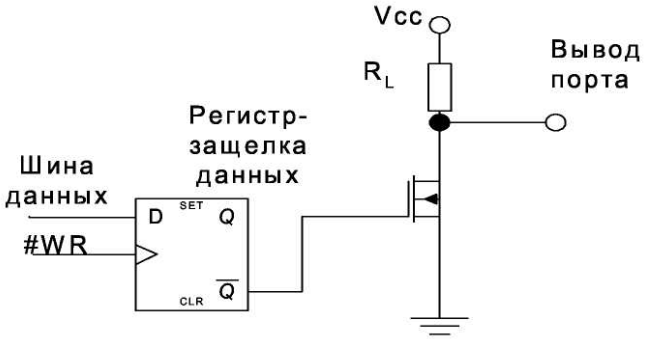


Рисунок 4.3. – Порт вывода с однотактной схемой

Когда в регистр-защелку записано значение «1», транзистор закрыт и на выходе через резистор $R_L$ устанавливается $V_{CC}$ – логическая «1». Когда же в регистр-защелку записан «0», открывается транзистор и соединяет выход с минусовой шиной питания, то есть 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..100 кОм. 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.

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).

Недостатки:

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»).

### 4.1.5. Порты вывода с открытым выходом

Порты вывода с открытым выходом (открытым коллектором или стоком). Применяются во многих семействах микроконтроллеров, например PIC. Выходной каскад построен по однотактной схеме с внешней нагрузкой. Принцип функционирования аналогичен описанному для однотактного выходного каскада.

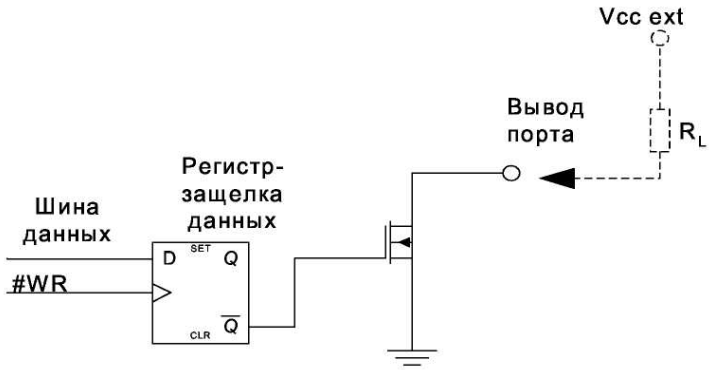


Рисунок 4.4. – Порт вывода с открытым выходом (стоком)

Достоинства:

– внешнее напряжение питания нагрузки $V_{CCext}$ может быть иным – большим или меньшим, чем питание микропроцессора. Это удобно для сопряжения схем с различными уровнями логической «1», например, 3.3 В и 5 В. Если внешнее напряжение достаточно высокое, то можно непосредственно управлять высоковольтной нагрузкой. Например, анонсирован микроконтроллер семейства PICmicro, допускающий подключение внешнего напряжения $V_{CCext}$ до 15 В при питании ядра напряжением 2..6 В.

– 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 $R_L$, например, стандарт $I^2C$ 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.

### 4.1.6. 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.

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, однако, так как сопротивление резистора велико (10..100 кОм), то даже на маломощный входной сигнал данная нагрузка не окажет заметного влияния. Схема квазидвунаправленного порта используется в семействе MCS-51.

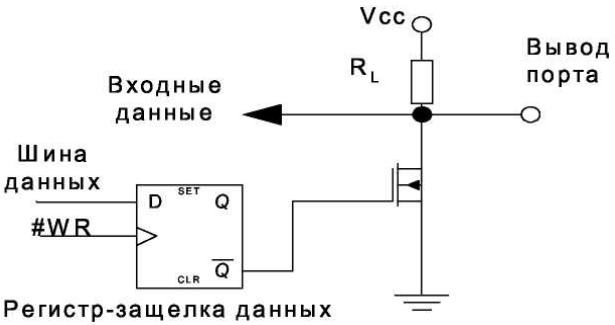


Рисунок 4.5. – Квазидвунаправленный порт

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

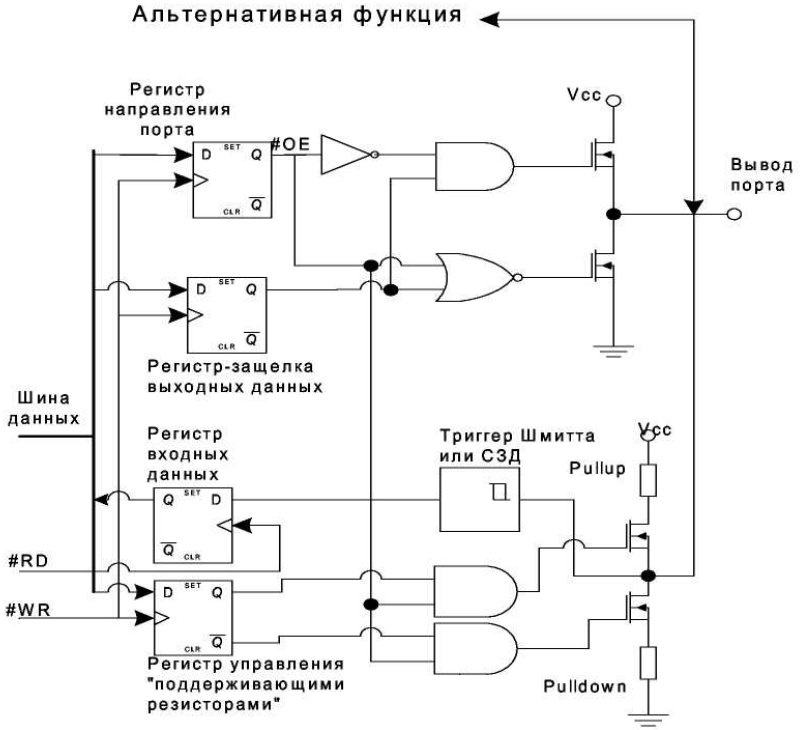


Рисунок 4.6. – Переключаемый двунаправленный порт с комплементарным выходным каскадом

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 pullup и pulldown подключаются только в режиме ввода, для чего на вход соответствующей схемы управления подключается выход регистра «вход/выход» («1» — ввод).

Кроме исполнения функции порта ввода-вывода, внешние выводы микросхемы могут быть задействованы для связи с внутренними периферийными модулями микропроцессора, а так же 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.

### 4.2. Таймеры-счетчики

Исходя из опыта построения микропроцессорных систем можно выделить типовые задачи, которые должен уметь решать МК для эффективного управления в реальном времени:

– отсчет равных интервалов времени заданной длительности, повтор алгоритма управления по истечении каждого такого временного интервала; обычно эту функцию называют формированием меток реального времени.

– контроль за изменением состояния линии ввода МК;

– измерение длительности сигнала заданного логического уровня на линии ввода МК;

– подсчет числа импульсов внешнего сигнала на заданном временном интервале;

– формирование на линии вывода МК сигнала заданного логического уровня с программируемой задержкой по отношению к изменению сигнала на линии ввода;

– формирование на линии вывода МК импульсного сигнала с программируемой частотой и программируемым коэффициентом заполнения.

Каждая из перечисленных задач в отдельности может быть выполнена только программными средствами, без использования специальных аппаратных решений. Так, для формирования интервала времени следует загрузить в регистр центрального процессора число, а затем выполнять команду декрементирования этого регистра до тех пор, пока содержимое регистра не станет равным нулю. Похожие решения можно предложить и для остальных перечисленных задач. Однако все эти решения будут обладать общим недостатком: невозможностью выполнения вычислений одновременно с отсчетом временного интервала. Поэтому для выполнения функций, связанных с управлением в реальном времени, в состав МК включают специальные аппаратные средства, которые называют *таймером.*

В процессе развития модули таймеров в составе 8-разрядных МК непрерывно совершенствовались. Рассмотрим принцип действия одного из первых модулей, который входит в состав МК AT89S51 фирмы «Atmel».

Таймер представляет собой 16-разрядный счетчик со схемой управления (рис. 4.7). В карте памяти МК счетчик отображается двумя регистрами: TH — старший байт счетчика, TL — младший байт. Регистры доступны для чтения и для записи. Направление счета счетчика — только прямое, т. е. при поступлении тактовых импульсов десятичный эквивалент двоичного кода счетчика изменяется в сторону увеличения.

В зависимости от программных настроек счетчик может использовать один из двух источников тактирования:

1) импульсную последовательность с выхода управляемого делителя частоты $f_{BUS}$;

2) внешнюю импульсную последовательность, поступающую на один из входов МК.

В первом случае говорят, что счетчик работает в *режиме таймера,* во втором – в *режиме счетчика событий.* При переполнении счетчика устанавливается триггер переполнения TF, который генерирует запрос на прерывание, если прерывания от таймера разрешены. После переполнения работа счетчика про-

должается. Последовательность изменения кодов следующая: $FF, $00, $01 и т. д. Запуск и останов счетчика могут выполняться только под управлением программы посредством установки/сброса соответствующего бита. Программа также может установить старший TH и младший TL байты счетчика в произвольное состояние или прочитать текущий код счетчика. Однако эти операции не следует выполнять в процессе счета, так как длительность операции чтения или записи может превысить длительность периода частоты тактирования счетчика. Тогда за время чтения одного из байтов второй успеет измениться. В результате будет прочитана недостоверная информация. По этой же причине может оказаться ошибочной операция записи. Например, пользователь записывает в регистры счетчика код $0005: сначала старший байт $00, а затем младший — $05. Если текущее состояние счетчика пользователю неизвестно, то может оказаться, что на момент завершения операции записи старшего байта младший равен $FF.

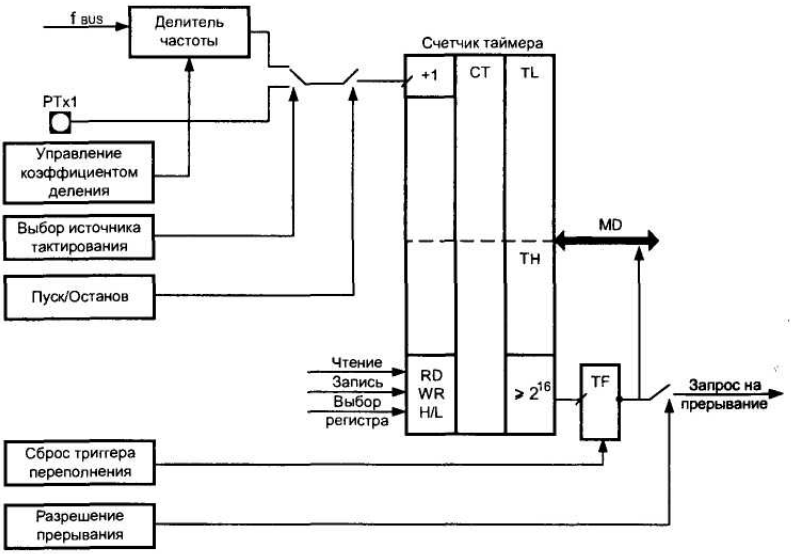


Рисунок 4.7. – Структурная схема модуля таймера-счетчика

Тогда в процессе записи младшего байта старший, ранее записанный, успеет измениться, и конечный код счетчика будет равен $0105.

Если рассмотренный таймер используется для измерения временного интервала $t_{ИЗМ}$ (рис. 4.8), то необходимо выполнить следующую последовательность действий:

1) прервать выполнение текущей программы при изменении сигнала на линии PTx.1 с «0» на «1»; в подпрограмме прерывания установить регистры счетчика таймера в $0000 и разрешить счет;

2) при изменении сигнала на линии PTx1 с «1» на «0» еще раз прервать выполнение программы МК; в подпрограмме прерывания остановить счет. Код в регистрах TH и TL будет равен длительности интервала $t_{ИЗМ}$, выраженной числом периодов частоты тактирования счетчика таймера.

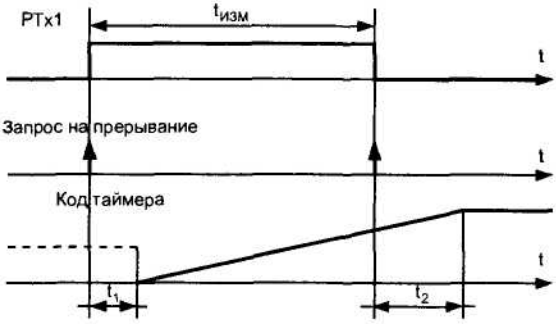


Рисунок 4.8. – Измерение временного интервала с использованием «классического» таймера

Однако такой способ измерения годится для сигналов, длительность которых составляет единицы мс и более. Моменты разрешения счета таймера $t_1$ и его остановки $t_2$ не совпадают с моментами изменения сигнала на входе PTx1, так как пуск и останов выполняются в подпрограмме прерывания. Ошибка счета равна $t_1 - t_2$. Каждое из значений $t_1$ и $t_2$ определяется временем перехода МК к выполнению подпрограммы прерывания и временем выполнения некоторого количества инструкций, предшествующих команде разрешения или останова счета таймера.

Вторая величина является систематической ошибкой и может быть учтена при выполнении дальнейших расчетов, в то время как первая — время перехода на подпрограмму прерывания — величина случайная, которая зависит от особенностей выполнения программного обеспечения МП системы. Так, если рассматриваемых каналов измерения несколько, и изменение сигналов на входах PTxi произошло одновременно, то в первом из обслуженных по прерыванию каналов ошибка будет минимальной, а в последнем — максимальной. Эта максимальная ошибка может составить несколько десятков микросекунд, поэтому рассмотренный метод не может быть использован для изменения временных интервалов микросекундного диапазона. В противном случае точность измерения будет недостаточной. Наряду с задачей измерения временного интервала может возникнуть необходимость одновременного формирования времяимпульсных сигналов по нескольким каналам. Тогда желательно, чтобы МК имел в своем составе несколько таймеров. Увеличение числа модулей таймеров, интегрированных на кристалл МК, является объективной тенденцией совершенствования структуры современных МК.

Рассмотренный «классический» модуль таймера наиболее часто используется в МК с архитектурой MSC-51. Однако даже в составе основателя семейства МК 8051 AH он претерпел некоторые усовершенствования:

– дополнительная логика счетного входа позволяет тактовым импульсам поступать на вход счетчика, если уровень сигнала на одной из линий ввода равен 1; это повышает точность измерения временного интервала, так как интервалы $t_1$ и $t_2$ теперь не являются составляющими погрешности измерения;

– реализован режим перезагрузки счетчика произвольным кодом в момент переполнения; это позволяет формировать метки реального времени с периодом, отличным от периода полного коэффициента счета, равного $2^{16}$.

Главный недостаток модуля «классического» таймера — невозможность одновременного обслуживания (измерения или формирования импульсного сигнала) сразу в нескольких каналах. Совершенствование структуры подсистемы реального времени 8-разрядных МК ведется по двум направлениям:

1) простое увеличение числа модулей таймеров; этот путь характерен для части МК компаний «Philips» и «Atmel» со структурой MSC-51, для МК компаний «Mitsubishi» и «Hitachi».

2) модификация структуры модуля таймера, при которой увеличение числа каналов достигается не увеличением числа счетчиков, а введением дополнительных аппаратных средств *входного захвата* и *выходного сравнения;* Такой путь характерен для 8-разрядных МК «Motorola», «Microchip», «Philips», «Infineon».

### 4.3. Аппаратные средства входного захвата и выходного сравнения

Типовая структура усовершенствованного модуля таймера представлена на рис. 4.9 и 4.11. Счетчик таймера дополнен аппаратными средствами входного *захвата* и *выходного сравнения.* Эти средства принято называть каналом входного захвата IC *(Input Capture)* и выходного сравнения OC *(Output Compare)*.

Принцип действия канала входного захвата поясняет рис. 4.9. Схема детектора события «следит» за уровнем напряжения на одном из входов МК. Обычно это одна из линий порта ввода/вывода. При изменении уровня логического сигнала на входе детектора с «0» на «1» или наоборот вырабатывается строб записи, и текущее состояние счетчика таймера записывается в 16-разрядный регистр данных TIC канала захвата. Описанное действие в микропроцессорной технике называют *событием захвата*. Предусмотрены три типа изменения сигнала на входе детектора, которые воспринимаются как событие захвата:

1) изменение логического уровня с 0 на 1 (нарастающий фронт сигнала);

2) изменение логического уровня с 1 на 0 (падающий фронт сигнала);

3) любое изменение логического уровня сигнала.

Выбор типа события захвата устанавливается в процессе инициализации модуля таймера и может многократно изменяться по ходу выполнения программы. Каждое событие захвата отмечается установкой в «1» триггера TFIC. Состояние триггера

может быть считано программно, а если прерывания по событию захвата разрешены, то генерируется запрос на прерывание.

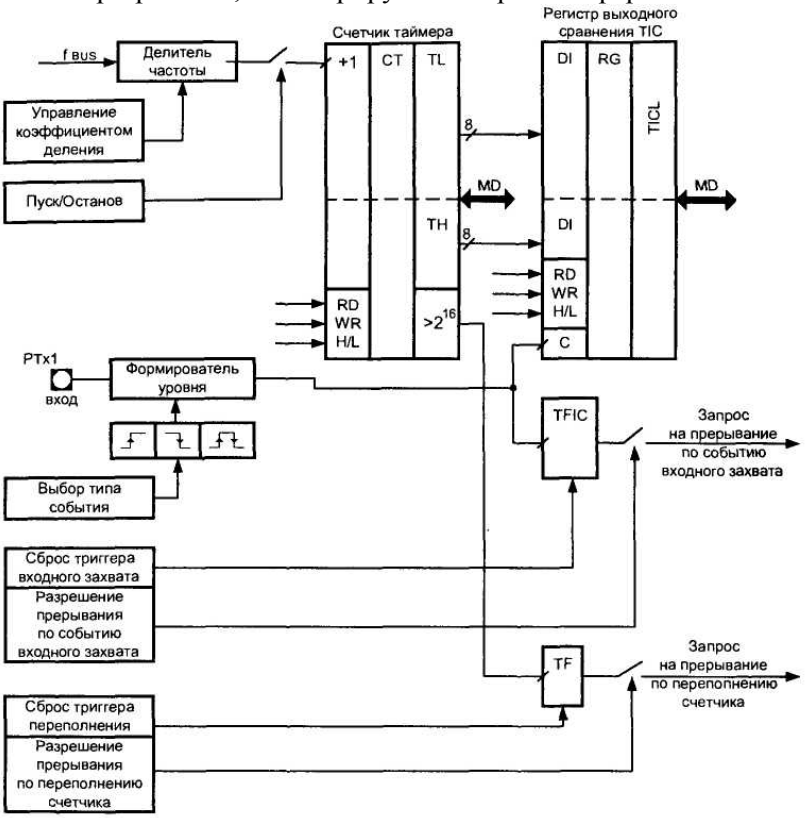


Рисунок 4.9. – Принцип действия канала входного захвата таймера

Временные диаграммы рис. 4.10 поясняют процесс изменения временного интервала с использованием режима входного захвата. Первоначально детектор события инициализируется на контроль за нарастающим фронтом сигнала на линии PTx1. При изменении уровня сигнала с «0» на «1» код счетчика К копируется в регистр канала захвата TIC. Триггер TFIC устанавливается в «1», одновременно формируется запрос на прерывание: таймер «сообщает» МК о том, что измеряемый интервал начался. С за-

держкой времени $t_1$ по отношению к моменту появления запроса на прерывание МК считывает код $К_1$ из регистра TIC, сбрасывает триггер TF1C и инициализирует детектор события на контроль за падающим фронтом сигнала PTx1. При изменении уровня сигнала с «1» на «0» детектор снова фиксирует событие захвата, и код счетчика $К_2$ копируется в регистр TIC. Снова выставляется запрос на прерывание, с задержкой $t_2$ этот код считывается в память МК. Разность кодов $К_2 – К_1$ и есть длительность измеряемого временного интервала, выраженная в числе периодов частоты тактирования счетчика таймера. Максимальная ошибка измерения равна двум периодам частоты тактирования, так как погрешность детектора событий не может превышать единицы квантования таймера. Время перехода к подпрограммам прерывания $t_1$ или $t_2$ не оказывает влияния на точность измерения, так как копирование текущего состояния счетчика осуществляется аппаратными, а не программными средствами. Однако время перехода на подпрограмму прерывания ($t_1$) накладывает ограничение на длительность измеряемого интервала $t_{ИЗМ}$, так как обсуждаемый метод реализуем при условии, что второе событие захвата произойдет позже, чем код $К_1$ будет считан в память МК.

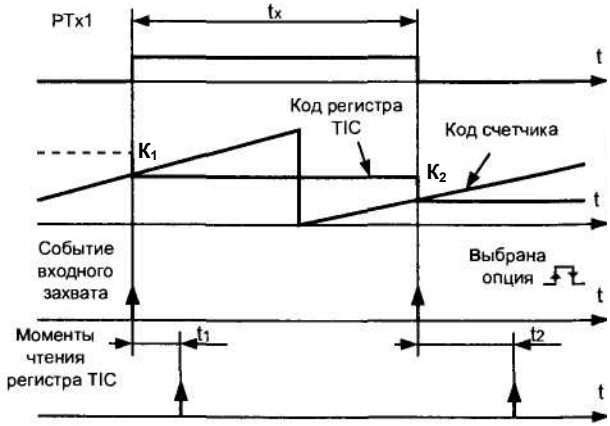


Рисунок 4.10. – Измерение временного интервала средствами канала входного захвата

Структура аппаратных средств канала выходного сравнения представлена на рис. 4.11. Многоразрядный цифровой компара-

тор непрерывно сравнивает изменяющийся во времени код счетчика\таймера с кодом, который записан в 16-разрядном регистре TOC канала сравнения. В момент равенства кодов на одном из выводов МК (PTx2 на рис. 4.11) устанавливается заданный уровень логического сигнала.

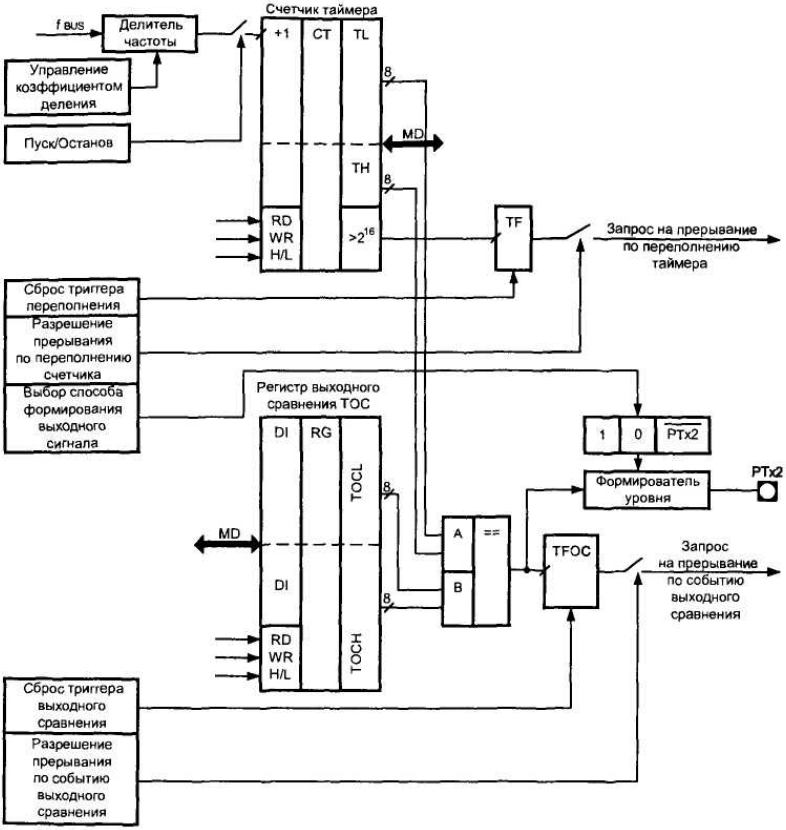


Рисунок 4.11. – Принцип действия канала выходного сравнения таймера

Рассмотренное действие называют событием выходного сравнения. Предусмотрены три типа изменения сигнала на выходе PTx2 в момент события выходного сравнения:

1) инвертирование сигнала на выходе;

2) установка низкого логического уровня;

3) установка высокого уровня.

При наступлении события захвата устанавливается в «1» триггер TFOC. Аналогично предыдущему случаю состояние триггера может быть считано программно, а если прерывания по событию выходного сравнения разрешены, то генерируется запрос на прерывание.

Временные диаграммы рис. 4.12 иллюстрируют способ формирования временного интервала предварительно рассчитанной длительности $t_x$ с использованием аппаратных средств канала выходного сравнения. Первое событие сравнения в момент $t_1$ формирует нарастающий фронт сигнала PTx2. Одновременно генерируется запрос на прерывание МК, и в подпрограмме прерывания происходит загрузка нового кода сравнения $K_2$. Время, необходимое для записи нового значения в регистр канала сравнения TOC, ограничивает минимальную длительность формируемого временного интервала. В момент $t_1$ наступает второе событие сравнения, и выход PTx2 устанавливается в «0». Таким образом, длительность сформированного временного интервала $t_x$ определяется только разностью кодов и не зависит от особенностей программного обеспечения МК.

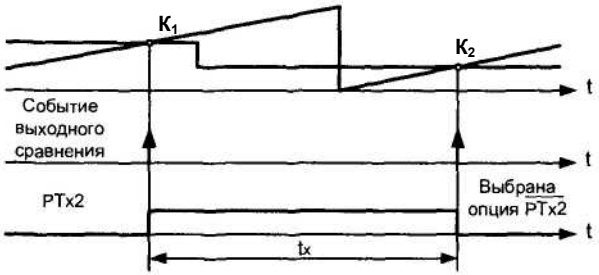


Рисунок 4.12. – Формирование импульса длительностью $t_x$ средствами канала выходного сравнения

В рассмотренных примерах каждому событию модуля таймера (изменению уровня логического сигнала на входе или выходе МК) ставится в соответствие код счетчика, т. е. счетчик используется для создания непрерывно изменяющихся меток текущего времени. В отличие от «классического» таймера, где счетчик используется непосредственно для формирования кода

измеряемого временного интервала, в усовершенствованном таймере счетчик лишь создает образ времени, подобно часам. А все действия по формированию или измерению временных интервалов производят аппаратные средства сравнения/захвата. Поэтому счетчик в составе модуля усовершенствованного таймера называют счетчиком временной базы, или просто «временной базой».

Аппаратные средства усовершенствованного таймера позволяют решить многие задачи управления в реальном времени. Однако процесс совершенствования алгоритмов управления предъявляет все новые требования к структуре МК. И, как следствие, все более отчетливо проявляются ограничения модулей усовершенствованного таймера:

– недостаточное число каналов сравнения и захвата, принадлежащих одному счетчику временной базы; в результате невозможно сформировать синхронизированные между собой многоканальные импульсные последовательности;

– однозначно определенная конфигурация каналов (или захват или сравнение) часто не удовлетворяет пользователя;

– с использованием средств выходного сравнения возможно формирование сигнала по способу широтно-импульсной модуляции (ШИМ), однако несущая частота ШИМ сигнала тем меньше, чем больше вычислений требуется выполнять при реализации алгоритма управления и чем больше число ШИМ каналов требуется реализовать.

### 4.4. Процессоры событий

Впервые модули процессоров событий были предложены фирмой «Intel» в составе МК 8xC51FA/FB/FC/GB, позже аналогичный модуль появился в МК с ядром MSC-51 фирмы «Philips». Модуль, который входит в состав перечисленных МК, носит название программируемого счетного массива PCA (Programmable Counter Array).

Структурная схема процессора событий приведена на рис. 4.13. Модуль процессора событий содержит в себе 16-разрядный счетчик временной базы и некоторое количество универсальных каналов захвата/сравнения. Счетчик может тактироваться им-

пульсной последовательностью с выхода программируемого делителя частоты стробирования межмодульных магистралей $f_{BUS}$ или внешним генератором. Счетчик имеет опции пуска/ останова и сброса в «0». В некоторых моделях процессора событий счетчик временной базы доступен для чтения «на лету». Режим чтения «на лету» предусматривает автоматическое копирование содержимого старшего и младшего байтов счетчика в специальные буферные регистры в момент выполнения операции чтения указанного в спецификации байта счетчика (старшего или младшего). Тогда при чтении второго байта счетчика возвращается код из соответствующего буферного регистра. Тем самым исключается ошибка считывания по причине изменения состояния счетчика временной базы за время чтения.

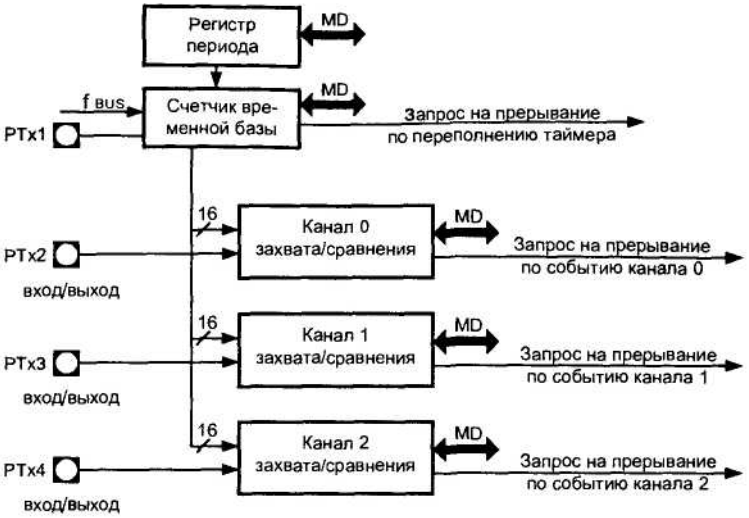


Рисунок 4.13. – Структурная схема модуля процессора событий

Наиболее совершенные модели процессора событий 8-разрядных МК допускают изменение коэффициента счета счетчика временной базы или, что то же самое, изменение периода его работы. Для этого в составе модуля имеется двухбайтовый программно доступный регистр периода и многоразрядный

цифровой компаратор (не путать с каналом захвата). При совпадении текущего кода счетчика временной базы с кодом периода триггеры счетчика временной базы автоматически сбрасываются в «0».

Универсальные каналы захвата/сравнения в процессоре событий полностью идентичны друг другу и в зависимости от программных настроек могут работать в одном из трех режимов:

1) режим входного захвата;

2) режим выходного сравнения;

3) режим широтно-импульсной модуляции (ШИМ).

Первые два режима по принципу действия ничем не отличаются от аналогичных режимов модуля усовершенствованного таймера. Программно-логическая модель каждого канала включает двухбайтовый регистр данных канала и триггер события. В зависимости от выбранного режима регистр данных канала используется аппаратными средствами для записи кода временной базы в момент наступления входного захвата или для хранения кода выходного сравнения. Триггер устанавливается при наступлении любого из этих событий. При работе канала в режиме выходного сравнения могут возникать нарушения алгоритма работы, приводящие к неправильному формированию сигнала на выходе PTxi модуля. Причиной таких сбоев является изменение под управлением программы величины кода сравнения в процессе работы канала. Наиболее совершенные модели процессора событий предусматривают для таких случаев специальный режим буферированного сравнения, при котором: 16-разрядный регистр кода сравнения дублируется; в каждый момент времени к входу компаратора оказывается подключенным один из регистров данных, а для записи оказывается доступным другой; в момент наступлении события выходного сравнения регистры автоматически меняются местами.

### 4.5. Работа процессора событий в режиме широтно-импульсной модуляции

В режиме широтно-импульсной модуляции (рис. 4.14) на выводе PTxi МК формируется последовательность импульсов с периодом, равным периоду работы счетчика временной базы.

Длительность импульса (в некоторых моделях длительность паузы) прямо пропорциональна коду в регистре данных канала. Режим ШИМ чрезвычайно удобен с точки зрения программного обслуживания. Если изменение коэффициента заполнения γ не требуется, то достаточно один раз занести код γ в регистр данных и проинициализировать режим ШИМ, и импульсная последовательность будет воспроизводиться с требуемыми параметрами без дальнейшего вмешательства со стороны программы.

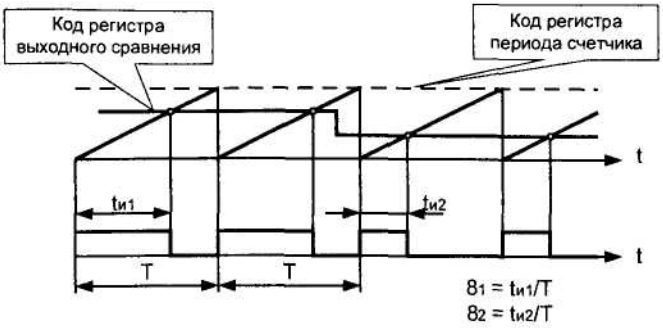


Рисунок 4.14. – Временные диаграммы работы канала в режиме ШИМ

Режим ШИМ в различных моделях процессоров событий имеет существенные отличия. В модулях программируемого счетного массива PCA код коэффициента заполнения γ имеет однобайтовый формат, следовательно, дискретность регулирования коэффициента заполнения составляет 1/256 периода ШИМ-сигнала. Причем, 16-разрядный регистр данных универсального канала PCA в режиме ШИМ «распадается» на два однобайтовых регистра. Доступным для записи является только один из регистров. В начале каждого периода ШИМ-сигнала содержимое этого регистра копируется во второй регистр, который используется аппаратными средствами для формирования длительности импульса в текущем периоде. Такое буферирование позволяет избежать нарушений при формировании импульсной последовательности в случаях, когда до наступления момента равенства кодов регистра и счетчика происходит изменение кода коэффициента γ, и новое значение γ меньше текущего кода счет-

чика. Тогда сравнение кодов в текущем периоде ШИМ сигнала не наступит, и импульс будет пропущен. Кроме недостатка по низкой дискретности регулирования коэффициента заполнения модуль PCA имеет ограниченный набор несущих частот сигнала ШИМ. Это происходит потому, что коэффициент счетчика временной базы не может быть изменен, а регулирование частоты достигается только изменением коэффициента деления предварительного делителя частоты.

### 4.6. Модуль аналого-цифрового преобразователя

Отличительная особенность многих современных 8-разрядных МК (например, ATMega компании Atmel) — интегрированный на кристалл МК модуль многоканального аналого-цифрового преобразователя (АЦП). Модуль АЦП предназначен для ввода в МК аналоговых сигналов с датчиков физических величин и преобразования этих сигналов в двоичный код с целью последующей программной обработки. Структурная схема типового модуля АЦП представлена на рис. 4.15.

Многоканальный аналоговый коммутатор служит для подключения одного из источников аналоговых сигналов (PTx0 – PTx7) к входу АЦП. Выбор источника сигнала для измерения осуществляется посредством записи номера канала коммутатора в соответствующие разряды регистра управления АЦП. Заметим, что в модулях АЦП 8-разрядных МК предусмотрена только программная установка номера канала, режим автоматического последовательного сканирования каналов с записью результата измерения каждого канала в индивидуальную ячейку памяти не реализуется.

Диапазон измеряемых значений напряжения аналоговых входов определяется напряжением опоры $U_{ОП}$. Могут быть использованы опорные источники следующего типа:

1) внешний, подключаемый через специальные выводы микросхемы;

2) внутренний фиксированный или программируемый (с помощью встроенного ЦАП).

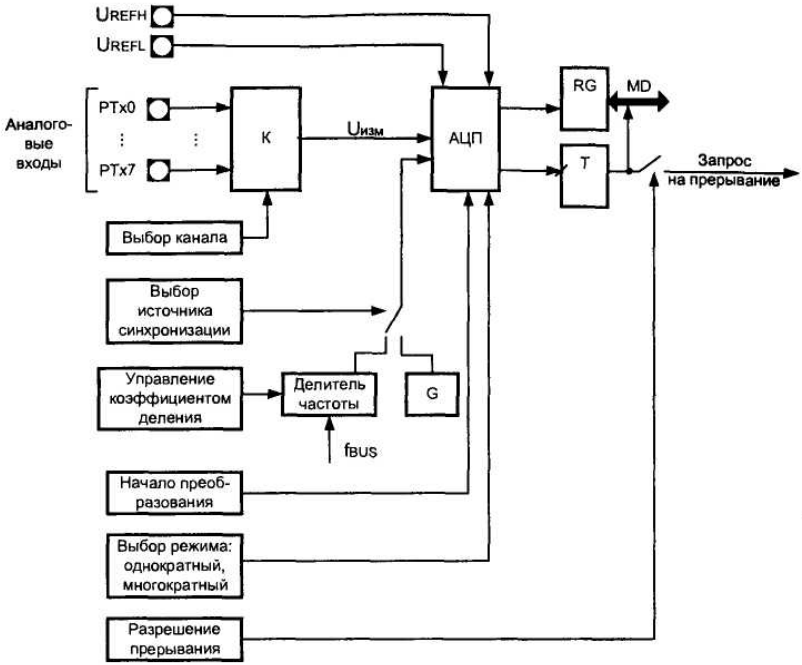


Рисунок 4.15. – Типовая структура модуля АЦП

Разрешающая способность АЦП составляет $U_{ОП}/2^n$, где n— число двоичных разрядов в слове результата. Максимальное значение опорного напряжения, как правило, равно напряжению питания МК. Два вывода модуля АЦП используются для задания опорного напряжения: $U_{REFH}$ – верхний предел, $U_{REFL}$ – нижний предел. Разность потенциалов на входах $U_{REFH}$ и $U_{\mathrm{REFL}}$ и составляет $U_{ОП}$. Если измеряемое напряжение $U_{ИЗМ} > U_{REFH}$, то результат преобразования будет равен $FF, код $00 соответствует напряжениям $U_{ИЗМ} < U_{REFL}$. Для достижения максимальной точности измерения следует выбирать диапазон входного напряжения АЦП максимально приближенным к диапазону опорных источников.

Собственно аналого-цифровой преобразователь выполнен по способу последовательного приближения. Принцип действия

АЦП иллюстрируют функциональная схема и временные диаграммы (рис. 4.16 и 4.17).

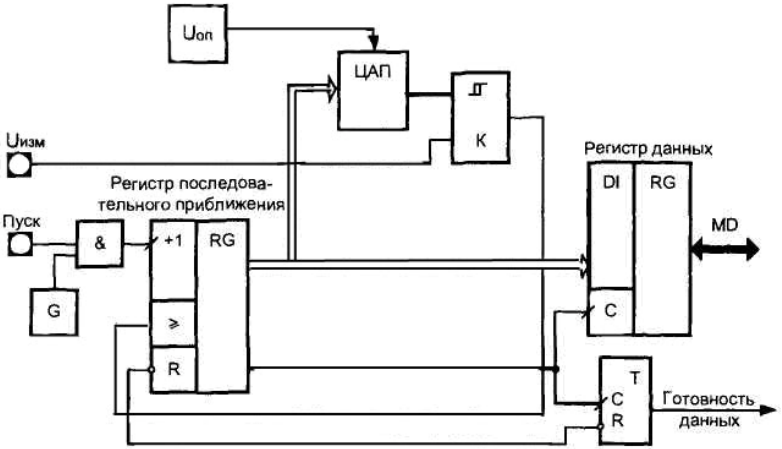


Рисунок 4.16. – Структура АЦП последовательного приближения

АЦП включает регистр последовательного приближения, цифроаналоговый преобразователь, компаратор, генератор импульсов синхронизации, схему управления, регистр результата и триггер готовности данных. Начало преобразования задает сигнал «Пуск», который устанавливает регистр последовательного приближения в состояние 10000000. При этом на выходе ЦАП формируется напряжение, равное половине опорного. Компаратор сравнивает измеряемое напряжение с напряжением ЦАП. Если $U_{ИЗМ} > U_{ЦАП}$, то в регистре последовательного приближения формируется следующий код сравнения, равный 11000000. Если $U_{ИЗМ} < U_{ЦАП}$ , то старший разряд регистра последовательного приближения устанавливается в «0», следующий код сравнения равен 01000000. Таким образом, на первом такте измеряемое напряжение сравнивается с эталонным значением *Uоп/2*. Аналогичные действия выполняются в каждом из тактов преобразования, однако значение напряжения сравнения зависит от результатов сравнения в предыдущих тактах. В примере рис. 4.17 на втором такте напряжение сравнения равно $U_{ОП}/2 + U_{ОП}/4$, на третьем такте — $U_{ОП}/2 + U_{ОП}/8$, поскольку на втором такте

было установлено, что $U_{ИЗМ} < U_{ОП}/2 + U_{ОП}/4$. Интервал преобразования состоит из n тактов: один такт для получения каждого двоичного разряда слова результата.

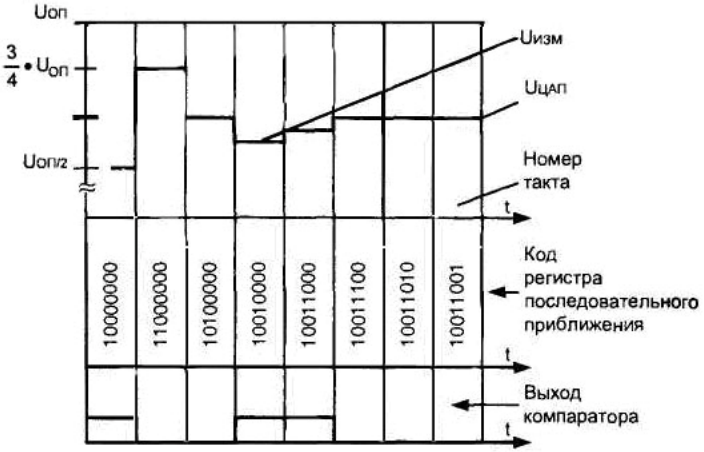


Рисунок 4.17. – Принцип действия АЦП последовательного приближения

Точность аналого-цифрового преобразования определяется разрешающей способностью блока ЦАП. В разных моделях МК он может быть выполнен на основе матрицы из 256 резисторов равных номиналов, матрицы весовых резисторов R/2R, достаточно часто встречается решение на основе емкостного делителя. Однако во всех моделях МК разрешающая способность ЦАП составляет 8 двоичных разрядов. Соответственно формат представления результата измерения АЦП – однобайтовый. Исключение составляют лишь модули АЦП микроконтроллеров для управления преобразователями частоты электропривода, разрешающая способность которых равна 10 двоичным разрядам. Два младших разряда результата получают с помощью дополнительного емкостного делителя, не связанного с регистром последовательного приближения. Длительность такта преобразования задает генератор синхронизации: один цикл равен двум периодам частоты генератора $t_{ADC}$. Время преобразования можно рассчитать по формуле $t_{ИЗМ} = (2n + 1)t_{ADC}$, последний цикл необходим для переноса результата в регистр данных АЦП. Частота

генератора синхронизации, как правило, не может быть выбрана произвольно. Ее ограничение определяется двумя обстоятельствами:

1) конденсаторы емкостного делителя неидеальны и имеют определенное сопротивление утечки; поэтому длительность такта измерения должна быть, с одной стороны, достаточна для заряда конденсаторов до уровня напряжения, определяемого разрядами регистра последовательного приближения, а с другой стороны, не слишком большой, чтобы погрешность утечки не была значительной;

2) в некоторых моделях источник питания аналоговой части модуля АЦП выполнен на основе импульсного преобразователя; для его работы требуется тактирование, частота тактирования определена с достаточно узким допуском ±10 %.

Источником синхронизации модуля АЦП может служить встроенный RC-генератор или импульсная последовательность тактирования межмодульных магистралей МК. В первом случае частота синхронизации АЦП обязательно окажется оптимальной, той, которая рекомендуется в техническом описании. Во втором случае выбранная в результате других соображений $f_{BUS}$ может оказаться неподходящей для модуля АЦП. На этот случай в составе некоторых модулей предусмотрен программируемый делитель частоты $f_{BUS}$.

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 $f_{\text{ЗАП.ВН.}}$ от основного тактового генератора или встроенного таймера. 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.

2. Внешнего запуска. Запуск осуществляется внешним сигналом $f_{\text{ЗАП.ВНЕШ.}}$, что позволяет четко определить момент считывания значения аналогового напряжения со входа.

3. Программно управляемого запуска, по установке специального бита $F_{\text{ЗАП.ПРГ}}$.

Большинство моделей АЦП имеет только режим программного запуска: установка одного из битов регистра режима запускает очередное измерение. Наиболее универсальные модули АЦП имеют также режим автоматического запуска, при котором после завершения одного цикла преобразования немедленно начинается следующий. Однако данные измерения каждого цикла должны быть считаны программным способом.

### 4.7. Модуль цифроаналогового преобразования

Цифроаналоговые преобразователи в составе МК являются большой редкостью. Модули параллельных ЦАП можно встретить лишь в МК «Mitsubishi» и «Hitachi».

ЦАП предназначен для генерации аналогового сигнала с уровнем напряжения, соответствующим заданному цифровому коду.

Блок-схема ЦАП приведена ниже.

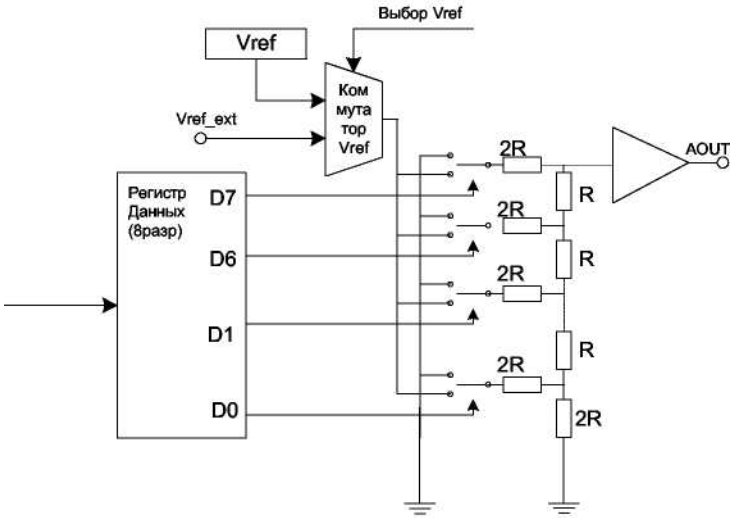


Рисунок 4.18. – Модуль цифро-аналогового преобразователя (ЦАП)

Регистр данных — в него записывается цифровой код. Регистр данных определяет разрядность ЦАП.

Матрица R-2R — самый распространенный метод цифроаналогового 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 $V_{REF}$ или 0 В. Ключи управляются разрядами регистра данных: «1» — на матрицу подается $V_{ref}$, «0» — подается 0 В.

Коммутатор опорного напряжения $V_{ref}$ позволяет выбрать внешний или встроенный источник опорного напряжения.

В микроконтроллерах функция цифроаналогового преобразователя часто реализуется средствами модуля программируемого таймера. На одном из выводов МК формируется высокочастотная импульсная последовательность с регулируемой длительностью импульса. Полученный сигнал сглаживается фильтром нижних частот на операционном усилителе (рис. 4.19). Разрешающая способность такого ЦАП определяется дискретностью регулирования коэффициента заполнения в режиме ШИМ. Как было отмечено выше, немногие модели 8-разрядных МК способны реализовать коэффициент заполнения с дискретностью более 8 разрядов. Этим вызвана необходимость применения внешних интегральных схем ЦАП. Кроме того, частота ШИМ сигнала определяет время преобразования ЦАП, этим объясняется особая привлекательность МК с частотой ШИМ в десятки килогерц.

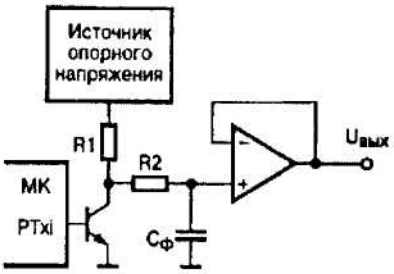


Рисунок 4.19. – Цифроаналоговый преобразователь на основе ШИМ-генератора

# 5. МОДУЛИ ПОСЛЕДОВАТЕЛЬНОГО ОБМЕНА В МИКРОКОНТРОЛЛЕРАХ

Модули последовательных интерфейсов ориентированы на решение следующих задач:

– связь встраиваемой микропроцессорной системы с системой управления верхнего уровня: промышленным или офисным компьютером, программируемым контроллером. Наиболее часто для этих целей используют интерфейсы RS232C, RS422, USB, IrDA;

– связь с внешними по отношению к микропроцессору периферийными микросхемами (памяти EEPROM, часов реального времени (RTC) и т.д.), а также с различными датчиками с последовательным цифровым выходом. Для этих целей наиболее часто применяются интерфейсы SPI, $I^2C$, Micro Wire, uLAN и другие;

– интерфейс связи с локальной сетью в распределенных информационно-управляющих системах. В этой сфере находят применение интерфейсы RS232C, RS485, C, uLAN, CAN, Ethernet;

– внутрисистемное программирование резидентной памяти программ (OTPROM, EPROM, FLASH) или данных (EEPROM) у процессоров для встраиваемых применений. Обычно для этого используется интерфейс RS232C (ADuC (Analog Devices), MB90Fxxx (Fujitsu), MSP430 (Texas Instruments)) или SPI (AVR(Atmel)).

В настоящее время встроенные контроллеры последовательных интерфейсов имеются почти у всех микроконтроллеров, исключая простейшие 8-16 выводные микросхемы. У большинства имеются несколько таких модулей одного или различных типов.

С точки зрения инженера-схемотехника, упомянутые типы интерфейсов последовательной связи отличаются: режимом передачи данных (синхронный или асинхронный), форматом кадра (число бит в посылке при передаче байта полезной информации) и временными диаграммами сигналов на линиях (уровни сигналов и положение фронтов при переключениях). Число линий, по которым происходит передача в последовательном коде, обычно

равно двум ($I^2C$, RS-232C, RS-485, CAN) или трем (SPI, некоторые нестандартные синхронные протоколы). Последнее позволяет спроектировать модули контроллеров последовательного обмена таким образом, чтобы с их помощью на аппаратном уровне можно было бы реализовать несколько типов последовательных интерфейсов. При этом режим передачи (синхронный или асинхронный) и формат кадра поддерживаются на уровне логических сигналов, а реальные физические уровни сигналов, характерные для каждого типа интерфейса, получают с помощью специальных ИС, которые носят название приемопередатчиков, конверторов, трансиверов.

В состав 8-разрядных МК различных фирм производителей входят следующие модули контроллеров последовательных интерфейсов:

– модуль универсального последовательного интерфейса USI (Universal Serial Interface). Он входит в состав МК семейства AVR фирмы «Atmel»; может поддерживать протоколы асинхронного обмена для интерфейсов RS-232, RS-422 и RS-485, а также синхронные протоколы интерфейсов SPI и $I^2C$;

– модуль универсального асинхронного интерфейса UART (Universal Asynchronous Receiver and Transmitter). Он поддерживает протоколы асинхронного обмена интерфейсов RS-232, RS-422 и RS-485;

– модуль универсального асинхронного интерфейса SCI (Serial Communication Interface). Он характерен для МК фирмы «Motorola»; входит в состав 8-разрядных МК семейств HC05, HC11 и HC08; является функциональным аналогом модулей типа UART, т. е. поддерживает протоколы асинхронного обмена для интерфейсов RS-232, RS-422 и RS-485;

– модуль синхронного последовательного интерфейса SPI (Serial Peripheral Interface). Он поддерживает протокол синхронного обмена в стандарте SPI; интерфейс SPI был предложен фирмой «Motorola», поэтому контроллер SPI входит в состав большого числа моделей МК семейств HC05, HC11 и HC08. В МК других производителей протокол SPI обычно реализуется в качестве альтернативного одним из модулей контроллеров последовательных интерфейсов;

– модуль синхронного последовательного интерфейса $I^2C$ (Inter Integrated Circuit). Изначально разработан компанией Philips. Он входит в состав 8-разрядных МК фирмы «Philips» и «Microchip»; следует заметить, что для МК «Microchip» характерна реализация аппаратными средствами одного и того же модуля протоколов SPI и $I^2C$;

– модуль контроллера CAN (Control Aria Network); присутствует в 8-разрядных МК семейства HC08 фирмы «Motorola», МК семейства C500 фирмы «Infineon», семейства 89 фирмы «Philips». Он поддерживает стандартные протоколы обмена CAN сетей;

– модуль контроллера USB (Universal Serial Bus); поддерживает новый стандарт периферийного интерфейса вычислительной техники USB.

Протоколы интерфейсов локальных сетей на основе МК — $I^2C$ и CAN — отличает более сложная логика работы. То же можно сказать и о новом стандарте периферийного интерфейса USB. Поэтому контроллеры CAN и USB интерфейса всегда выполняются в виде самостоятельного модуля, аппаратные средства которого ориентированы на поддержку соответствующих протоколов обмена. Интерфейс $I^2C$ с возможностью работы как в ведущем, так и ведомом режиме, также обычно поддерживается специальным модулем (модуль последовательного порта в МК 89C52 фирмы «Philips»). Но если реализуется только ведомый режим $I^2C$, то в МК PIC16 «Microchip» он успешно сочетается с SPI: настройка одного и того же модуля на один из протоколов осуществляется путем инициализации.

Следует заметить, что одноименные модули контроллеров последовательных интерфейсов даже одной фирмы-производителя имеют отличия в реализации для разных семейств МК. Так, аппаратные средства контроллера SCI в составе МК семейства HC08 диагностируют большее количество ошибок на линии, чем одноименные контроллеры в составе семейства HC05. И естественно, отличаются одноименные модули в МК различных фирм. Однако эти отличия преимущественно сводятся к различию регистров специальных функций, которые обслуживают модуль. Меньше затрагивают алгоритмы функционирования одноименных модулей. И, по определению, все

аналогичные модули обязательно реализуют на аппаратном уровне логику протокола обмена выбранного интерфейса. Поэтому при рассмотрении данной темы целесообразно остановиться именно на протоколах обмена. По режиму обмена информацией интерфейсы подразделяют на симплексные, полудуплексные, дуплексные, мультиплексные. В интерфейсах с симплексным режимом обмена информацией возможна лишь однонаправленная передача информации от одного абонента к другому. Соответственно и буферы приемника и передатчика информации выполнены однонаправленными. В интерфейсах с полудуплексным режимом обмена в произвольный момент времени может производиться либо только прием, либо только передача данных между двумя абонентами, буферы приемо-передатчика каждого из абонентов связи выполнены двунаправленными. В интерфейсах с дуплексным режимом обмена в любой произвольный момент времени может производиться одновременный прием и передача данных между двумя абонентами. Линии приема и передачи информации физически разделены, соответственно контроллер обмена каждого абонента имеет два вывода (приемника и передатчика), и буферы этих выводов однонаправленные. В интерфейсах с мультиплексным режимом обмена в каждый момент времени может осуществляться прием или передача данных между парой любых абонентов сети.

## 5.1. Модуль универсального синхронно-асинхронного приемо-передатчика USART

Среди контроллеров последовательного обмена стандартом «де-факто» стал модуль универсального синхронно-асинхронного приемо-передатчика (USART (Universal Synchronous/Asynchronous Receiver and Transmitter)). В названии часто опускают слово «синхронный» и модуль не совсем корректно именуется UART (чисто асинхронные приемо-передатчики сейчас встречаются достаточно редко). Модули UART в асинхронном режиме поддерживают протокол обмена для интерфейсов RS232C (8N1 или 9N1); в синхронном режиме — нестандартные синхронные протоколы, в некоторых случаях — протокол SPI.

Упрощенная структура приемопередатчика типа UART представлена на рисунке.

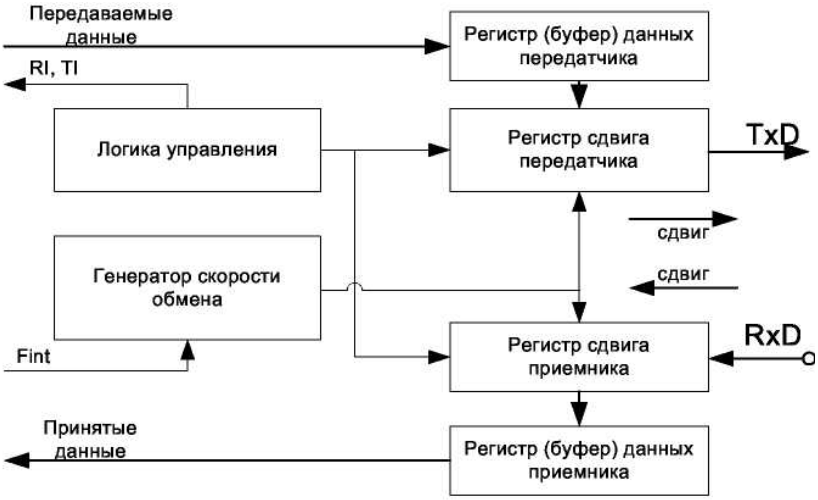


Рисунок 5.1. – Упрощенная структура UART

Генератор скорости обмена представляет собой делитель внутренней тактовой частоты процессора Fint с плавно или пошагово (дискретно) программируемым коэффициентом деления. При «плавном» программировании можно настраивать требуемую скорость вне зависимости от частоты Fint (в определенных пределах). Для этого используется стандартный или специально выделенный таймер-счетчик в режиме автоперезагрузки. В случае «фиксированных» коэффициентов деления для поддержания стандартного ряда скоростей необходимо выбирать определенную частоту тактирования процессора. С выхода генератора скорости сигнал синхронизации поступает на вход тактирования приемного и передающего сдвиговых регистров, которые осуществляют последовательную выдачу/прием бит данных с заданной скоростью. Полностью принятый байт попадает в регистр — буфер данных приемника. Байт для передачи помещается в сдвиговый регистр из буфера передатчика.

Процессы приема и передачи в асинхронном режиме UART происходят независимо. Таким образом, поддерживается дуп-

лексный режим обмена. Однако требуется, чтобы приемник и передатчик были настроены на одну скорость.

Более простым является функционирование в синхронном режиме. Здесь каждый принимаемый/передаваемый бит стробируется специальным сигналом и нет необходимости точно согласовывать скорость приемника и передатчика.

Еще более упрощается функционирование в режиме SPI: приемник и передатчик работают синхронно: приему одного бита соответствует передача одного бита, начало передачи байта совпадает с началом приема, за сеанс обмена происходит прием одного байта и передача одного байта.

В большинстве случаев приемопередатчики работают с входными и выходными сигналами уровней TTL. Формирование физических сигналов с уровнями напряжения и тока, соответствующими реализуемому интерфейсу выполняется с помощью специальных микросхем — трансиверов или адаптеров физического интерфейса. Например: MAX232 (MAXIM) — RS232C, MAX485 (MAXIM) — RS422/485, PCA82C251 (Philips) — CAN.

## 5.2. Модуль последовательной шины I²C

### 5.2.1. Общие сведения о шине I²C

Разработанная фирмой Philips шина I²C («Inter-Integrated Circuit») — это двунаправленная синхронная шина с последовательной передачей данных и возможностью адресации до 128 устройств. Физически шина I²C содержит две сигнальные линии, одна из которых (SCL — Serial CLock) предназначена для передачи тактового сигнала, вторая (SDA — Serial Data and Adress) для обмена данными. Для управления линиями применяются выходные каскады с открытым коллектором, поэтому линии шины должны быть подтянуты к источнику питания +5В через резисторы сопротивлением в диапазоне 1...10 кОм, в зависимости от физической длины линий и скорости передачи данных. Длина соединительных линий в стандартном режиме может достигать 2-х метров, скорость передачи — до 100 кбит/с. Суммарная ёмкость линий должна быть не больше 400 пФ, входная ёмкость на каждую ИС должна быть в пределах 5... 10 пФ.

В конце 90-х годов был расширен диапазон стандартных скоростей.

| Напряжение питания устройства $I^2C$, В | Макс. частота на SCL, кГц |
|---|---|
| 1,8 | 100 |
| 2,7 | 400 |
| 5 | 1000 |

Это позволило значительно расширить сферу применения данного класса устройств.

Все абоненты шины делятся на два класса — «Master» (главный, ведущий) и «Slave» (подчиненный, ведомый). Устройство «Master» генерирует тактовый сигнал (SCL) и, как следствие, является ведущим. Оно может самостоятельно выходить на шину и адресовать любое «Slave»-устройство с целью передачи или приёма информации. Все «Slave»-устройства «слушают» шину на предмет обнаружения собственного адреса и, распознав его, выполняют предписываемую операцию. Кроме того, возможен так называемый «Multi Master»-режим, когда на шине установлено несколько «Master-абонентов, которые либо совместно разделяют общие «Slave»-устройства, либо попеременно являются то «Master»-устройствами, когда сами инициируют обмен информацией, то «Slave», когда находятся в режиме ожидания обращения от другого «Master»-устройства. Режим «Multi Master» требует арбитража и распознавания конфликтов. Естественно, он сложнее в реализации (имеется ввиду программная реализация) и, как следствие, реже используется в реальных изделиях.

В зависимости от направления передачи возможны два типа обмена данными для $I^2C$-шины.

1. Передача данных от главного передатчика к подчиненному приемнику. Первый байт, передаваемый передатчиком, является адресом подчиненного приемника. Затем следует несколько байтов данных. Подчиненный приемник возвращает бит подтверждения после каждого принятого байта.

2. Передача данных от подчиненного передатчика к главному приемнику. Первый байт (адрес подчиненного передатчика) передается главным устройством. Затем подчиненный передат-

чик возвращает бит подтверждения. Следующие несколько байтов данных передаются подчиненным устройством главному. Главное устройство возвращает бит подтверждения после каждого принятого байта, кроме последнего. В конце последнего принятого байта возвращается «нет подтверждения».

Когда нет передачи данных, реализуется режим ожидания: линии тактирования SCL и данных SDA приведены подтягивающими резисторами к высокому уровню логического сигнала.

Модуль контроллера интерфейса $I^2C$, который удовлетворяет спецификации $I^2C$-шины, может работать в следующих четырех режимах.

1. *Режим главного передатчика.*

Последовательный вывод данных через выход SDA передатчика, в то время как на выходе SCL передатчика формируются последовательные синхроимпульсы. Первый переданный байт содержит адрес подчиненного приемного устройства (7 бит) и бит направления данных R/W = 0. В этом случае говорят, что передается «W». Таким образом, первый переданный байт представляет собой адрес подчиненного приемника плюс «W». Последовательные данные передаются по 8 бит. После отправки каждого байта главный передатчик ожидает от подчиненного устройства бит подтверждения ACK (ACKnowledge). Условия START и STOP формируются ведущим (главным) устройством для указания начала и конца сеанса последовательного обмена посылкой, состоящей в общем случае из нескольких байтов.

2. *Режим главного приемника.*

Первый переданный приемником байт содержит адрес подчиненного передающего устройства (7 бит) и бит направления данных R/W = 1. В этом случае говорят, что передается «R». Таким образом, первый переданный приемником байт представляет собой адрес подчиненного передатчика плюс «R». Последовательные данные передаются по линии SDA от ведомого (подчиненного) устройства к ведущему (главному), в то время как импульсы синхронизации на линии SCL формирует ведущий. Последовательные данные передаются по 8 бит. После того, как ведущий (главный) принял очередной байт, он выставляет на линию сигнал подтверждения приема ACK. Сигналы START и STOP формируются ведущим.

3. *Режим подчиненного приемника.*

Последовательные данные и синхроимпульсы передаются по линиям SDA и SCL на одноименные входы подчиненного приемника. После того, как принят каждый байт, приемник (с приемом очередного синхроимпульса от главного передатчика по шине SCL) выставляет на линию SDA бит подтверждения ACK, который анализируется главным передатчиком. Условия START и STOP формируются передатчиком. Распознавание адреса выполняется аппаратными средствами модуля приемника после приема адреса подчиненного устройства и бита направления.

4. *Режим подчиненного передатчика.*

Первый байт принимается и обрабатывается подчиненным передатчиком также, как и в режиме подчиненного приемника. Однако бит направления в принятом байте будет указывать, что направление обмена должно быть изменено на обратное. Далее последовательные данные передаются по линии SDA с одноименного выхода подчиненного (ведомого) передатчика, в то время как синхроимпульсы принимаются им по входу SCL от главного приемника. После передачи каждого байта подчиненный передатчик анализирует наличие на линии бита подтверждения ACK от главного приемника. Условия START и STOP формирует главный приемник.

В подчиненном режиме аппаратные средства контроллера $I^2C$-интерфейса осуществляют поиск своего собственного подчиненного адреса или адреса общего вызова. Если детектируется один из этих адресов, запрашивается прерывание. Когда микроконтроллер желает, чтобы шина стала главной, аппаратные средства ожидают, пока шина освободится. Возможное функционирование в качестве подчиненного при этом не прерывается. Если арбитраж шины потерян в главном режиме, то соответствующий контроллер $I^2C$ переключается в подчиненный режим немедленно и может детектировать свой собственный подчиненный адрес.

### 5.2.2. Алгоритм работы шины $I^2C$

В начальный момент времени — в режиме ожидания — обе линии SCL и SDA находятся в состоянии логической единицы (транзистор выходного каскада с ОК закрыт). В режиме передачи (рис. 5.2) бит данных SDA стробируется положительным импульсом SCL. Смена информации на линии SDA производится при нулевом состоянии линии SCL. «Slave»-устройство может «придерживать» линию SCL в нулевом состоянии, например, на время обработки очередного принятого байта, при этом «Master»-устройство обязано дождаться освобождения линии SCL, прежде чем продолжить передачу информации.

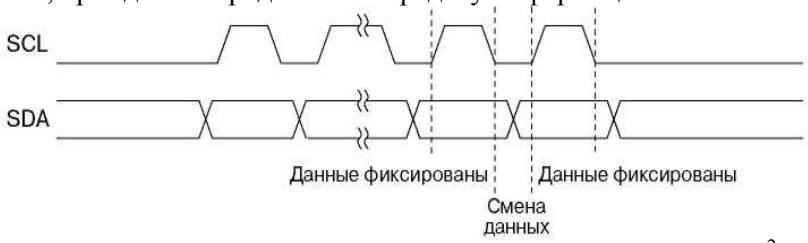


Рисунок 5.2. – Диаграмма процесса передачи данных по $I^2C$

Для синхронизации пакетов шины I2C различают два условия — «Start» и «Stop», ограничивающие начало и конец информационного пакета (рис. 5.3). Для кодирования этих условий используется изменение состояния линии SDA при единичном состоянии линии SCL, что недопустимо при передаче данных. «Start»-условие образуется при отрицательном перепаде линии SDA, когда линия SCL находится в единичном состоянии. «Stop»-условие образуется при положительном перепаде линии SDA при единичном состоянии линии SCL.

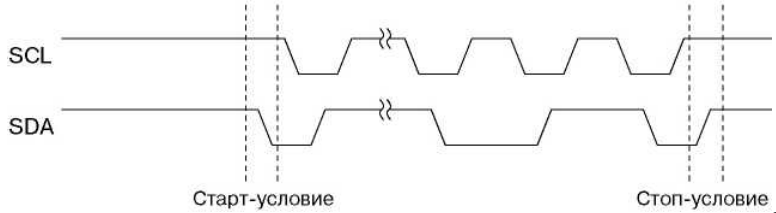


Рисунок 5.3. – Диаграмма «Старт»/«Стоп» условия шины $I^2C$

Передача данных начинается по первому положительному импульсу на линии SCL (рис. 5.4), которым стробируется старший бит первого информационного байта. Каждый информационный байт (8 битов) содержит 9 тактовых периодов линии SCL. В девятом такте устройство-получатель выдаёт подтверждение (ACK) — отрицательный импульс, свидетельствующий о «взаимопонимании» передатчика и получателя. Сразу отметим, что любой абонент шины, как «Master», так и «Slave», может в разные моменты времени быть как передатчиком, так и получателем и в соответствии с режимом обязан либо принимать, либо выдавать сигнал ACK, отсутствие которого интерпретируется как ошибка.

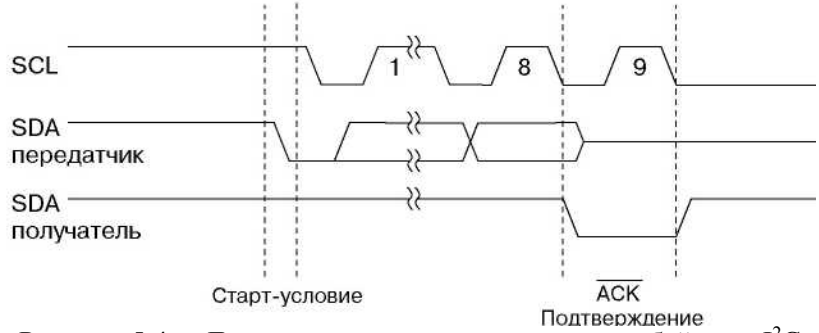


Рисунок 5.4. – Диаграмма подтверждения приёма байта по $I^2C$

Временная диаграмма сигналов SCL и SDA шины I2C приведена на рис. 5.5. Здесь S обозначает «Start»-условие, P — «Stop»-условие. Значения временных характеристик приведены в таблице5.1.

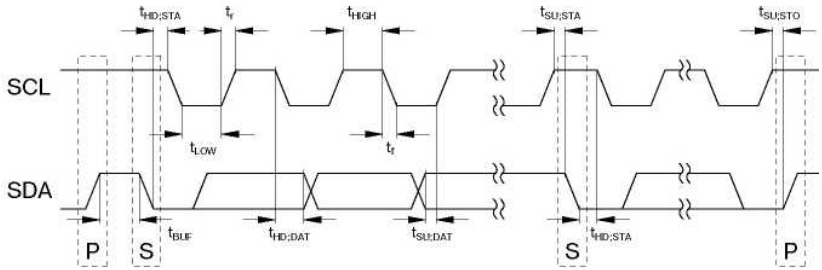


Рисунок 5.5. – Временная диаграмма работы шины $I^2C$

Таблица 5.1.

Значения временных характеристик шины $I^2C$

| Параметр | Обозн. | Мин. | Макс. | Един. |
|---|---|---|---|---|
| Частота сигнала SCL | $f_{SCL}$ | 0 | 100 | кГц |
| Свободная шина | $t_{BUF}$ | 4.7 | – | мкс |
| Фиксация «Start»-условия | $t_{HD;STA}$ | 4.0 | – | мкс |
| Длительность «LOW» полупериода SCL | $t_{LOW}$ | 4.7 | – | мкс |
| Длительность «HIGH» полупериода SCL | $t_{HIGH}$ | 4.0 | – | мкс |
| Готовность повторного «Start»-условия | $t_{SU;STA}$ | 4.7 | – | мкс |
| Удержание данных | $t_{HD;DAT}$ | 0 | – | мкс |
| Готовность данных | $t_{SU;DAT}$ | 250 | – | нс |
| Фронт сигналов SCL и SDA | $t_r$ | – | 1000 | нс |
| Спад сигналов SCL и SDA | $t_r$ | – | 300 | нс |
| Готовность «Stop»-условия | $t_{SU;STO}$ | 4.0 | | мкс |

Чтобы начать операцию обмена, устройство «Master» выдаёт на шину «Start»-условие, за которым следует байт с адресом «Slave»-устройства (рис. 5.6). Этот байт состоит из семибитового адреса устройства (биты 1...7) и однобитового флага операции — «R/W» (бит 0), определяющего направление обмена. Если значение этого бита равно «0» — это означает передачу от «Master» к «Slave» (рис. 5.6, а), а «1» — чтение из «Slave» (рис. 5.6, б). Все биты передаются по шине $I^2C$ в порядке старший-младший, то есть первым передаётся 7-ой бит, последним 0-ой. За адресом могут следовать один или более информационных байтов (в направлении, определённом флагом R/W), биты которых стробируются сигналом SCL из «Master»-устройства.

При совершении операции чтения «Master»-абонент должен сопровождать прочитанный байт сигналом ACK, если необходимо прочитать следующий байт, и не выдавать сигнала ACK, если собирается закончить чтение пакета (рис. 5.6, б).

Сигнал ACK служит подтверждением приема данных приемником и выдается приемником на шину SDA в момент передачи «Master»-устройством 9-го синхроимпульса на шину SCL.


Рисунок 5.6. – Формат операций чтения/записи

Рассмотрим небольшой пример, поясняющий сущность и необходимость ACK. Например, если «Master»-устройство выдало «Slave»-устройству сначала «Start»-условие, а затем 8 бит адреса «Slave»-устройства, сопровождая их восемью синхроимпульсами на шине SCL, то при выдаче 9-го синхроимпульса на шину SCL «Master»-устройство переводит линию SDA в режим чтения и считывает ее состояние. Если напряжение на линии SDA соответствует уровню логического «0», то это значит, что «Slave»-устройство приняло все 8 бит данных и готово к приему следующих данных. В противном случае «Master»-устройство должно выдать на шину $I^2C$ «Stop»-условие и повторить операцию взаимодействия с «Slave»-устройством с самого начала.

Допускается многократное возобновление «Slave»-адреса в одном цикле передачи, то есть передача повторного «Start»-условия без предварительного «Stop»-условия. Такой принцип широко применяется в управлении $I^2C$ абонентами, когда выдача нового «Start»-условия служит для синхронизации начала нового пакета данных, сопровождаемого, например, новым управляющим словом, уточняющим адресацию пакета. Логическая реализация протоколов на шине $I^2C$ не нормируется документами фирмы Philips, содержащими формальные описания шины, и может быть произвольной для каждой конкретной ИС.

### 5.2.3. Практические рекомендации при работе с шиной $I^2C$

Удобства применения шины $I^2C$ очевидны — малое количество соединительных линий и высокая скорость обмена, простота аппаратной реализации линии связи. Наиболее широко поддерживает шину $I^2C$, конечно же, фирма Philips, производящая множество ИС различной сложности с управлением по $I^2C$ В первую очередь, можно выделить микросхемы энергонезависимой памяти (EEPROM) серии 24Cxx в 8-ми выводных корпусах, фактически ставшие промышленным стандартом. Из широко распространённых ИС можно выделить: микросхемы часов PCF8583, параллельный порт PCF8574, 4-х канальный 8-ми разрядный АЦП PCF8591. Существует множество модификаций этих ИС и более специализированные контроллеры, в частности, фирма КТЦ-МК (Москва) выпускает специализированные контроллеры с управлением по $I^2C$: контроллер цифрового светодиодного дисплея и клавиатуры (CE210, CE220), контроллер ЖКИ-модуля с параллельной шиной, совмещённый с контроллером клавиатуры (CE110), контроллер сети на основе шины RS-485 (CE302).

$I^2C$-абоненты жёстко разделяются по классам: «Master»- и «S1ave»-устройство. Тот факт, что сигнал SCL всегда генерируется «Master»-устройством, означает, что «Master»-абонент может быть достаточно легко реализован чисто программными средствами, так как все изменения на шине будут происходить только по сигналу SCL. И наоборот, реализация «Slave»-устройства требует аппаратной поддержки, кроме случая очень низких скоростей обмена. Существуют однокристальные микроЭВМ (ОМЭВМ) поддерживающие «Slave»-операции шины $I^2C$. Это прежде всего Philips PCF80C552 (652), Microchip PIC16C62 (64, 73 и др.), Motorola MC68HC705CJ4 (BD3, E5). Кроме того, ОМЭВМ фирмы Philips аппаратно поддерживают «Master»-протокол, хотя практически аппаратные «Master»-контроллеры применяются редко.

В типичной микропроцессорной системе с применением $I^2C$-устройств обычно существует центральный МК, две линии портов которого отведены для управления линиями шины $I^2C$ SCL и

SDA. Линии шины $I^2C$ должны быть выполнены с открытым коллектором (ОК), и подтянуты резисторами к линии питания +5 В, что позволяет абонентам шины производить операции типа «Монтажное ИЛИ» для торможения процесса обмена в случае, если они не успевают отрабатывать операции со скоростью «Master-абонента.

Типичная ошибка при реализации программ «Master-абонента — управление значением порта ОМЭВМ для установки нулевого и единичного состояний линий SCL и SDA. Если для МК семейства MCS-51 – это нормальный режим работы, так как единичное состояние порта у них реализуется встроенным подтягивающим резистором 50 кОм, то для МК с симметричными портами (Motorola 68HCxxx, Microchip PIC, Atmel AVR) это будет порождать электрические конфликты. Например, в руководстве «Microchip. Embedded Control Handbook 1994/1995» приведены практические программы для связи PIC с ИС EEPROM, содержащие подобные грубые ошибки. Положение усугубляется тем, что в случае микросхем EEPROM такой вариант может сработать, так как они являются 100% аппаратными схемами и не вносят задержек в связной протокол, а паузу ожидания окончания цикла программирования производят переходом в пассивное состояние. Использование таких подпрограмм с ИС, производящими захват линии SCL (практически любой «Slave»-абонент, реализованный с применением микроконтроллера), приведёт к невозможности связи, а возможно, и к выходу ИС из строя.

Реализовать настоящую имитацию ОК (назовем этот режим имитацией ОК, так как он не позволяет устанавливать на линии напряжение выше напряжения питания, что было бы нормально для настоящего ОК, но так как по спецификации $I^2C$ напряжение на линиях SCL и SDA не должно превышать напряжение питания, его вполне законно можно считать выходом с ОК) на порте с симметричным выходом можно, если установить значение порта постоянно в ноль, а управлять состоянием линии через манипуляции с регистром направления данных. Для микроконтроллеров семейства PIC это будет регистр «TRISx», переводящий порт либо в третье состояние, либо подключающий линии в соответствии с состоянием регистра «PORTx». Практически так

же это реализуется в AVR, где «DDRx» коммутирует порт «PORTx», с той лишь разницей, что у них другая полярность управляющего сигнала — у PIC ноль в «TRISx» соответствует нулю на выходе, а у AVR единица в «DDRx» соответствует нулю на выходе.

Ещё одна тонкость, связанная с PIC-контроллером, не имеющая собственно к $I^2C$ никакого отношения заключается в том, что у него совмещены регистр чтения состояния линий порта и регистр записи значения порта, в результате чего, если часть линий порта запрограммирована на вывод, а другая находится в третьем состоянии, то при выполнении PIC'ом операции модификации в регистр порта перепишутся значения непосредственно с выводов PIC, установленных в третье состояние, что может нарушить (и скорее всего нарушит) работу программы обмена (потеряются нули, обеспечивающие имитацию выходов с ОК). Поэтому лучше производить принудительную установку в ноль битов в «PORTx», например, перед каждой операцией обмена, скажем такой последовательностью:

```
movlw   b,'11100111'    ; Маска линий SCL и SDA в порту
                        ; C  (RC3 и RC4)
andwf   portc,f         ; Сброс  значений линий SCL и
                        ; SDA в порту C в ноль
```

Другая важная сторона вопроса — необходимость тщательного соблюдения параметров временной диаграммы процесса обмена. Несмотря на то, что шина $I^2C$ синхронная и позволяет затягивать передачу бита (байта) на сколь угодно длительное время (это свойство позволяет реализовывать программы $I^2C$-обмена на самом низком уровне приоритета, прерывая процесс передачи в любое время), требования к минимальным значениям длительностей импульсов очень жёсткие. Ситуация усугубляется тем, что положительные перепады состояния линии имеют склонность затягиваться, так как несимметричные управляющие выходы не могут создать крутые положительные фронты.

При написании программ очень важно контролировать время между операциями на шине, реализуемыми различными подпрограммами, например выдача «Start» и «Stop»-условий, передача бита, передача байта. При состыковке этих подпрограмм не должны быть нарушены минимальные значения времени, что

очень легко происходит при использовании высокоскоростных процессоров (AVR, PIC). Кроме того, необходимо следить, чтобы время между изменением на линии SDA и стробированием положительным импульсом на линии SCL было не меньше половины минимальной длительности полупериода SCL (4.7 мкс/2 = 2.4 мкс). Помимо этого, некоторые «Slave»-устройства могут ужесточить требования к максимальной частоте обмена (например, CE110 — 48 кГц), в этом случае необходимо пропорционально снижению частоты обмена увеличивать значения минимумов временных допусков.

Ещё одна распространённая ошибка — игнорирование требования слежения за захватом линии SCL «Slave»-абонентом. Грамотно реализованные программы операций «Master»-абонента должны контролировать возврат линии SCL после того, как переводят её в единичное состояние, и только дождавшись реальной установки линии SCL в единичное состояние продолжать операции приёмопередачи.

### 5.2.4. Минимально необходимый набор операций для реализации «Master»-абонента

Прежде всего, это подпрограммы выдачи «Start»- и «Stop»-условия, подпрограмма передачи байта, подпрограмма приёма байта, выдающая сигнал ACK и подпрограмма приёма байта, не выдающая сигнал ACK. Три последние используют подпрограммы передачи бита и приёма бита. Во временных промежутках между подпрограммами линия SCL имеет нулевое значение. Как уже говорилось ранее, важно правильно согласовать все временные характеристики подпрограмм. Если, к примеру, вы вызываете повторный «Start» сразу за передачей (приёмом) байта, то необходимо, чтобы полностью завершился нулевой полупериод (> 4.7 мкс), предшествующий «Start»-условию, а подпрограмма, реализующая «Start»-условие, переведя линии SDA и SCL в единичное состояние, дождавшись их реального возврата, должна удерживать их в этом состоянии не менее установленного времени (> 4.7 мкс), и так во всех случаях.

Оперируя этими пятью подпрограммами можно легко наладить обмен по шине $I^2C$. Можно выдать на шину «Start»-

условие, за ним «Slave Adress», далее данные, в соответствии с протоколом конкретной $I^2C$-ИС. Можно выдать повторное «Start»-условие, новый «Slave Adress» и так далее, формируя процесс обмена как из элементов конструктора. Обычно протоколы обмена «Slave»-устройств довольно логично соответствуют программированию в таком стиле.

### 5.3. Модуль SPI

Стандарт SPI (Serial Peripheral Interface) предложен фирмой «Motorola». И, конечно, он реализован в большинстве моделей микроконтроллеров «Motorola». Но он также содержится и во многих микроконтроллерах других производителей. Например, в микроконтроллерах AVR и некоторых моделях семейства MSC-51 фирмы «Atmel». Стандарт SPI предназначен для связи МК с периферийными устройствами МП системы. Наиболее часто эти устройства расположены на одной плате с МК, реже — это вынесенные пульты управления, индикаторные панели и т. п. В качестве периферийных устройств могут использоваться как простейшие сдвиговые регистры, так и сложные периферийные ИС со встроенными контроллерами управления, такие как ЦАП, сигма-дельта АЦП с цифровой фильтрацией, последовательные запоминающие устройства типа FLASH или EEPROM, энергонезависимые ОЗУ и т. д. В редких случаях интерфейс SPI используется для обмена данными между несколькими МК системы.

На рис. 5.7 представлена структурная схема сопряжения МК и двух периферийных ИС с использованием интерфейса SPI. В рассматриваемом примере МК является ведущим устройством, он инициирует обмен при передаче информации между МК и одной из периферийных ИС. Каждая из периферийных ИС является устройством ведомым. SPI-шина представлена тремя общими линиями связи (MISO, MOSI, SCK) и двумя линиями выбора ведомого устройства (SS1, SS2), которые индивидуальны для каждой периферийной ИС:

MOSI — линия передачи данных от ведущего к ведомому (Master Output Slave Input);

MISO — линия передачи данных от ведомого к ведущему (Master Input Slave Output);

SCK — линия сигнала стробирования данных;
SS1 и SS2 — линии сигналов выбора ведомого устройства.

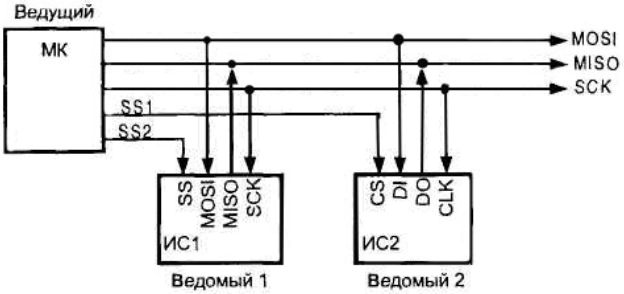


Рисунок 5.7. – Сопряжение микроконтроллера с периферийными устройствами по интерфейсу SPI

Как видно из рис. 5.7, образованная на основе интерфейса SPI мини-сеть относится к классу магистрально-радиальных. Линии передачи данных и линия синхронизации являются примером шинной организации, а линии выбора ведомого устройства — элемент системы радиального типа. Перед началом обмена (рис. 5.8) ведущее устройство отмечает одно ведомое устройство, с которым будет производиться обмен. Для этого на линии выбора устройства SSi устанавливается низкий активный уровень сигнала. Затем ведущее устройство последовательно выставляет на линию MOSI восемь бит информации, сопровождая каждый бит сигналом синхронизации SCK. Ведомое устройство дешифрирует переданный байт информации и определяет, в каком направлении будет производиться дальнейший обмен. Если ведомое устройство должно принимать информацию, то ведущее устройство, не снимая сигнала выбора ведомого SSi, продолжит передачу по линии MOSI. Если ведомое устройство должно передавать информацию, то оно активизирует линию MISO и в ответ на каждый импульс синхронизации от ведущего будет выставлять один бит информации. Длина посылки обмена в общем случае не ограничена, но для правильной работы модуля SPI должна составлять целое число байтов. Завершение обмена также инициируется ведущим посредством установки в неактивное состояние сигнала выбора ведомого SSi.

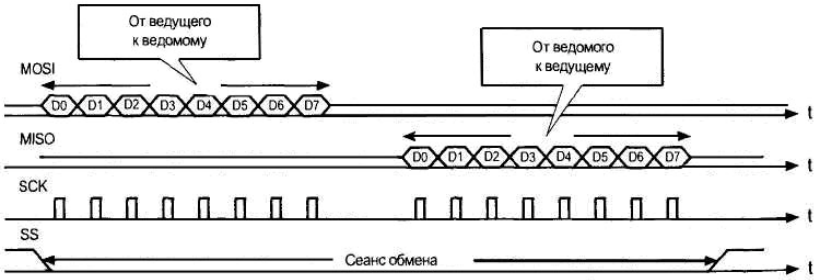


Рисунок 5.8. – Временные диаграммы обмена по протоколу SPI

Для подключения к SPI-шине встроенный контроллер SPI имеет четыре вывода: MOSI, MISO, SCK, SS. Модули контроллеров SPI фирмы «Motorola» могут работать как в ведущем, так и в ведомом режимах. Скорость приема и передачи определяется частотой тактирования межмодульных магистралей МК $f_{BUS}$: в ведущем режиме скорость обмена не может превышать $f_{BUS}/2$, в ведомом режиме максимальная скорость обмена равна $f_{BUS}$. Поэтому для МК семейства HC05 максимальная скорость обмена в ведущем режиме составляет 1 Мбит/с, в ведомом – 2 Мбит/с. Аналогичные показатели для МК семейства HC08 соответственно 4 и 8 Мбит/с.

При работе встроенного контроллера в ведущем режиме к выводу MOSI подключается выходная линия данных, а к MISO — входная. При работе в ведомом режиме выводы меняются ролями. Вывод SCK является выходом, если контроллер SPI работает в ведущем режиме, и входом, если – в ведомом. В системах с несколькими ведущими устройствами все выводы SCK соединяются вместе. То же делается с выводами MOSI и MISO. На время отсутствия связи буферы выводов встроенного контроллера SPI переводятся в высокоимпедансное состояние. Последнее позволяет избежать конфликтов на шине SPI. В противном случае несколько выводов MISO ведомых устройств одновременно были бы активными, что не позволило бы ведущему устройству произвести прием достоверной информации.

Вывод SS встроенного контроллера SPI используется в зависимости от того, в каком режиме работает данное устройство.

При работе в ведомом режиме при подаче высокого уровня сигнала на вход SS устройство игнорирует сигналы SCK и удерживает вывод MISO в высокоимпедансном состоянии. Если же в ведомом режиме работы на входе SS установлен низкий логический уровень, то буферы линий MOSI и SCK разворачиваются на ввод, линия MISO — на вывод. При работе в ведущем режиме вывод SS может быть использован как обычная линия вывода. В системах со сложной логикой работы этот вывод может использоваться как вход сигнала обнаружения ошибки для индикации состояния шины в случаях, если более чем одно устройство пытается стать ведущим.

Схема управления контроллера SPI интерфейса позволяет выбрать один из двух протоколов обмена и полярность импульсов синхронизации SCK. При работе в ведущем режиме возможно также программно выбрать частоту импульсов синхронизации.

Два бита регистра управления любого контроллера SPI интерфейса определяют временную диаграмму обмена по шине SPI:

1) бит CPHA назначает протокол обмена;

2) бит CPOL определяет полярность сигнала синхронизации SCK.

В соответствии с комбинацией битов CPHA:CPOL принято различать четыре режима работы интерфейса SPI. Комбинации CPHA:CPOL = 00 соответствует режим 0, комбинации CPHA:CPOL = 11 – режим 3. Встроенный контроллер SPI МК позволяет программно настраивать режим SPI в процессе инициализации, в то время как периферийные ИС реализуют один или два режима SPI, которые определяются их техническим описанием. Наиболее часто это режимы 0 и 3.

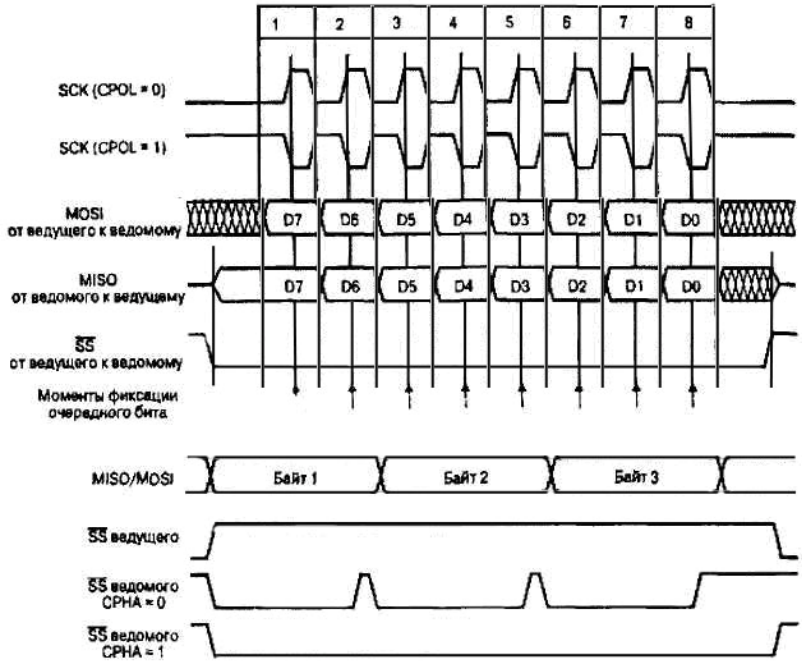


Рисунок 5.9. – Временные диаграммы обмена SPI интерфейса в режимах 0 и 1

На рисунок 5.9 представлены временные диаграммы сигналов для протокола передачи CPHA = 0. Для сигнала SCK приводятся две диаграммы, различающиеся полярностью сигнала.

Первая соответствует режиму 0, вторая — режиму 1. Диаграммы относятся как к ведущему, так и к ведомому устройству, поскольку выводы MISO и MOSI ведущего соединены с аналогичными выводами ведомого. Сигнал SS подается только на ведомое устройство. Поэтому вывод SS у ведущего остается незадействованным и его диаграмма не представлена, но имеется в виду, что она соответствует неактивному состоянию. Встроенные контроллеры SPI выполнены таким образом, что длина посылки составляет один байт, что и отражено на временных диаграммах.

Начало обмена рассматриваемого протокола определяется установкой сигнала выбора ведомого SS в активное состояние

SS = 0. При направлении передачи от ведущего к ведомому первый перепад сигнала синхронизации SCK используется ведомым устройством для запоминания очередного бита во внутреннем сдвиговом регистре контроллера SPI. Ведущий выставляет очередной бит посылки на линии MOSI по каждому четному фронту сигнала SCK. При передаче данных от ведомого к ведущему старший бит передаваемого байта должен быть выставлен ведомым на линию MISO сразу после изменения уровня сигнала SS — 0. По первому фронту SCK уровень сигнала на линии MISO будет запомнен в младшем разряде сдвигового регистра ведущего устройства. По этой причине сигнал на линии выбора ведущего должен быть возвращен в неактивное состояние SS = 1 после передачи каждого байта в любом направлении. Тогда передача каждого нового байта будет сопровождаться предварительной установкой SS в «0».

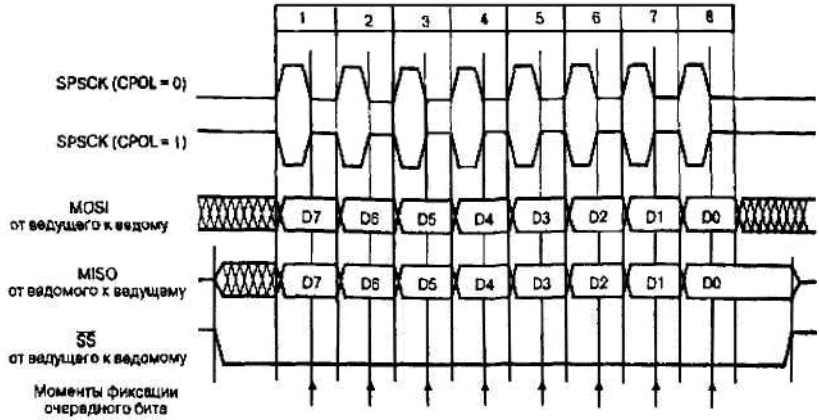


Рисунок 5.10. – Временные диаграммы обмена SPI интерфейса в режимах 2 и 3

Начало обмена для протокола опции CPHA = 1 (рис. 5.10) определяет первое изменение уровня сигнала на линии SCK после установки сигнала выбора ведомого SS в активное состояние SS = 0. При передаче данных от ведущего к ведомому и в обратном направлении все нечетные перепады SCK вызывают выдвижение очередного бита посылки из сдвигового регистра передатчика на линию. Каждый четный перепад используется для

записи этого бита в сдвиговый регистр приемника. Сигнал выбора ведомого может оставаться в активном состоянии SS = 0 в течение передачи нескольких байт информации. Это несколько упрощает логику программного драйвера SPI.

Рассмотренные протоколы обмена не имеют различий по скорости и надежности передачи информации. Выбор протокола диктуется периферийным устройством. В некоторых случаях полярность и фаза сигнала SCK изменяются между передачами для того, чтобы обеспечить связь с устройствами, имеющими различный протокол.

### 5.4. Модуль CAN

Разработанный в середине 1980-х фирмой «Bosch» для систем управления узлами автомобиля, протокол CAN (Controller Area Network — сеть контроллеров) является последовательным протоколом высокоскоростной и высоконадежной передачи данных в широковещательном (broadcast) режиме в мультимастерной среде. Удачное сочетание низкой стоимости подключения, простоты и надежности с доступностью элементной базы и инструментальных средств разработки — это основные достоинства CAN-технологии. Положения стандарта, закрепленные в используемой на сегодня спецификации 2.0A/B фирмы «Bosch» и международном стандарте ISO 11898, соответствуют двум начальным уровням (физическому и канальному) 7-уровневой модели взаимодействия открытых систем ISO/OSI. Ряд оригинальных технических решений, реализованных при разработке протокола, наилучшим образом позволили сориентировать его на решение задач контроля и управления.

Структура CAN-сети представлена на рис. 5.11. Шинная топология, являющаяся основой CAN, требует наличия механизма адресации узлов, однако в CAN нет адресов как таковых: сообщение принимается всеми узлами. Любое передаваемое сообщение имеет определяющий его содержание уникальный идентификатор (ID), на основании которого каждый узел фильтрует «свои» сообщения и «решает», реагировать или нет на сообщение, транслируемое в данный момент. Неоспоримыми преиму-

ществами отсутствия адресации являются теоретически неограниченное число узлов и простота их добавления и отключения.

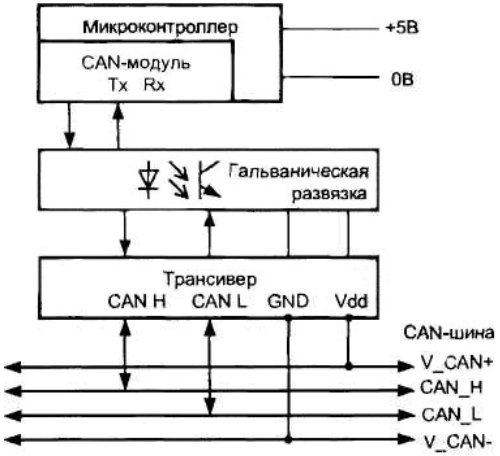


Рисунок 5.11. – Структура CAN-сети

Физическая среда передачи данных в CAN может быть самой разной – витая пара, плоский кабель, оптоволокно, а так же радио- и ИК-каналы и даже линии электропередач. Основным ограничением протяженности шины является лишь предельно допустимая суммарная задержка распространения сигнала для заданной скорости передачи (в кабеле, трансиверах, входных цепях контроллеров и т. д.). В соответствии с рекомендациями ISO 11898 при использовании стандартных трансиверов и быстродействующих оптопар (для гальванической развязки) максимальная протяженность сети при скорости передачи 1 Мбит/с ограничена девятью метрами. Предельная рекомендуемая протяженность сети в соответствии с тем же стандартом достигается при снижении скорости передачи до 50 кбит/с. А в документах промышленной CAN-группы CiA (CAN in Automation) приведены следующие полученные практическим путем соотношения «скорость — протяженность» для проводной сети без гальванической развязки: 1 Мбит/с — 30 м; 500 кбит/с — 100 м; 125 кбит/с — 500 м; 20 кбит/с — 2500 м; 10 кбит/с — 5000 м.

Сообщения, передаваемые по CAN-шине, именуются фреймами. Форматы фреймов передаваемых данных приведены на

рис. 5.12. В зависимости от инициатора передачи и ее цели существуют четыре типа фреймов:

1) Data Frame — фрейм данных;
2) Remote Frame — фрейм запроса данных;
3) Error Frame — фрейм ошибки;
4) Overload Frame — фрейм перегрузки.

Собственно для передачи данных используется Data Frame, в поле данных которого (Data Field) могут находиться от 0 до 8 байтов данных.

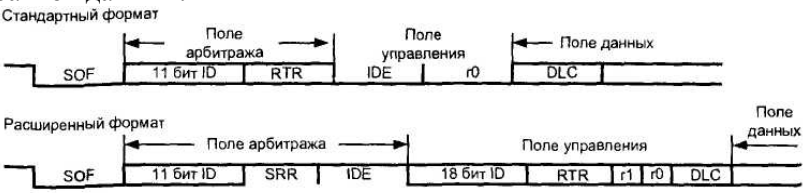


Рисунок 5.12. – Структура сообщения CAN-протокола

Поле арбитража (Arbitration Field) фрейма включает в себя идентификатор (ID), однозначно определяющий содержание и приоритет сообщения. Стандартным форматом сообщений (CAN Specification 2.0A) предусмотрен 11-битный идентификатор, позволяющий различать до 20348 типов сообщений (на практике обычно до 2032), а расширенный (CAN Specification 2.0B) — 29-битный (стандартный 11-битный с 18-битным расширением) с теоретически возможным числом типов сообщений более 536 млн. Фреймы, соответствующие стандартному и расширенному форматам сообщений, приведены на рис. 5.12.

Бит RTR (Remote Transmission Request — запрос передачи данных) для фрейма должен иметь доминантный уровень. В расширенном формате фрейма бит SRR (Substitute Remote Request) с рецессивным уровнем заменяет следующий (в стандартном формате) за 11-разрядным идентификатором бит RTR. Бит распознавания формата фрейма IDE (ID Extension) имеет доминантный уровень для стандартного формата фрейма и рецессивный — для расширенного. Биты r0 и r1 — резервные.

В поле управления (Control Field) содержится 4-разрядный код, задающий длину поля данных (0-8 байт) — DLC — Data Lenght Code. Поле контрольной суммы CRC Field включает в

себя контрольную сумму сообщения (15 бит) и бит-разделитель. В поле подтверждения ACK (Acknowledgement) передающий узел всегда выставляет рецессивный уровень. В случае, если передача прошла успешно, приемный узел сигнализирует об этом установкой в этом поле доминантного уровня.

Начинается фрейм доминантным битом SOF (Start of Frame), служащим также для синхронизации битового потока, а заканчивается семью битами рецессивного уровня поля EOF (End of Frame) и 3-битным того же уровня промежутком между фреймами. Для исключения потери синхронизации при передаче длинной последовательности одинаковых битов в пределах полей начала фрейма, арбитража, управления, данных и контрольной суммы используется битстаффинг — вставка дополнительного бита противоположного значения после подряд идущих пяти одинаковых. При приеме производится обратная (дебитстаффинг) операция.

Для запроса данных от удаленного узла служит фрейм запроса данных Remote Frame, также имеющий стандартный и расширенный форматы. Отличия фрейма запроса данных от фрейма данных — в отсутствии поля данных и рецессивном уровне бита RTR. При получении фрейма запроса данных запрашиваемый узел отвечает передачей фрейма данных.

Сигнализация об ошибках происходит посредством передачи фрейма Error Frame. Он инициируется любым узлом (в CAN правильность передачи контролируется каждым узлом), обнаружившим ошибку.

Шесть доминантных бит флага ошибки (активный флаг ошибки) перекрывают остаток ошибочно переданного фрейма и создают глобальную ошибку в сети — ошибку битстаффинга, которая воспринимается остальными узлами, если им не удалось обнаружить первоначальную (локальную) ошибку. Далее они выставляют свои флаги ошибки. Ввиду этого обстоятельства последовательность доминантных бит (суперпозиция флагов ошибки) может иметь длину от 6 до 12 бит. Ненадежным или частично поврежденным узлам (см. ниже) при обнаружении ошибки разрешено посылать лишь пассивный флаг ошибки — последовательность шести рецессивных бит.

Для задержки передачи данных или посылки фрейма запроса данных (при неготовности приемника или наличии доминантных бит в промежутке между фреймами) служит фрейм перегрузки Overload Frame. В отличие от фрейма ошибки он не влияет на счетчик ошибок и не вызывает повторную передачу сообщения.

Несколько необычно решается проблема коллизий (столкновений в сети), присущая шинной топологии. В этом случае снова используется идентификатор сообщения в сочетании со схемой подключения к шине типа «монтажное ИЛИ», где узел, выставляющий на шину «0» — доминантный уровень, подавляет «1» — рецессивный уровень, выставленный другим узлом. Победителем в арбитраже является узел, имеющий идентификатор с наименьшим численным значением и, как следствие, наивысший приоритет. Только победивший узел продолжает передачу данных, остальные пытаются сделать это позже.

Подобный режим доступа к шине известен как CSMA/CD+AMP (Carrier Sense Multiple Access with Collision Detection and Arbitration on Message Priority) — множественный доступ с контролем несущей, обнаружением коллизий и арбитражем на основе приоритета сообщений. Этот режим не позволяет поспорившим узлам устраивать столкновение на шине, а сразу выявляет победителя. CAN-протокол, изначально разработанный специально для систем управления жизненно важными узлами автомобилей, критичных к уровню безопасности и степени достоверности передаваемых данных, обладает эффективными средствами обнаружения ошибок.

В отличие от других сетевых протоколов, в CAN не используются подтверждающие сообщения, а при обнаружении одной или более ошибок хотя бы одним узлом (в CAN все узлы принимают все сообщения и участвуют в проверке сообщения на наличие ошибок – вычисляют контрольную сумму и т. п.), текущая передача прерывается (при условии, что ошибку обнаружил как минимум один узел со статусом Error Active) генерацией фрейма ошибки с флагом ошибки. Передатчик, сообщение которого было прервано, повторяет передачу.

## 5.5. Шина USB

### 5.5.1. Общие сведения о шине USB

Шина USB (Universal Serial Bus) является промышленным расширением архитектуры компьютеров PC. USB — быстрый, двунаправленный, дешевый, динамически подключаемый последовательный интерфейс, который совместим с основными требованиями различных платформ PC.

Основные отличительные особенности архитектуры шины USB:

– легкость в использовании для расширения числа периферийных устройств PC до 127;

– простота работы для конечного пользователя;

– дешевизна контроллеров, кабелей и оборудования;

– широкие возможности по подключению различных устройств со скоростями работы в пределах от нескольких Кбит/с до нескольких Мбит/с; поддержка скоростей передачи 12 Мбит/с и 1,5 Мбит/с;

– полная поддержка для передачи в реальном масштабе времени голоса, звука, и сжатого видео; при изохронных передачах обеспечивается гарантируемое требование по быстро действию и малое время отклика;

– поддерживаются как изохронные, так и асинхронные типы передачи данных по одним и тем же проводам.

– совместимость с различными конфигурациями PC и с существующими интерфейсами операционных систем;

– возможность динамически присоединять, идентифицировать и реконфигурировать периферийные устройства;

– высокая степень загрузки шины;

– широкий диапазон размеров пакета и встроенное в протокол управление потоком данных при буферной обработке;

– согласование скоростей передачи данных, размеров буферизируемого пакета и времени отклика;

– встроенный в протокол механизм восстановления при ошибках и обработки неисправностей. Поддержка обнаружения и отключения отказавших устройств.

Топология USB-шины. Шина USB соединяет USB-устройства с USB-хостом (host). В любой USB-системе может быть только один хост-контроллер (Host). На физическом уровне топология USB представляется в виде многоуровневой звезды (рис. 5.13). Устройства USB могут подключаться непосредственно к хосту, но так как число устройств может быть велико, предусмотрено подключение через специальные концентраторы (hubs), которые расположены в центре каждой звезды. Корневой концентратор (root hub) обычно интегрирован внутрь хост-системы, чтобы обеспечивать одну или большее число точек подключения. Каждый сегмент провода — двухточечное соединение между хостом и концентратором или функцией, или концентратором, соединенным с другим концентратором или функцией.

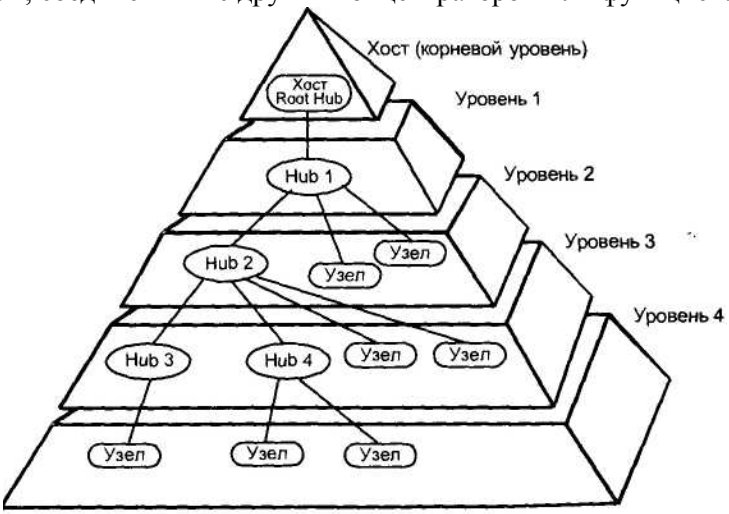


Рисунок. 5.13. – Топология USB-шины

USB-хост *взаимодействует с USB-устройствами через хост-контроллер и отвечает за:*

– обнаружение подключения и удаления USB-устройств;

– управление управляющим (Control) потоком между хостом и USB-устройствами;

– управление перенумерацией и конфигурирование подключенных USB-устройств;

– управление потоком данных между хостом и USB-устройствами;

– сбор статистики о состоянии и активности USB-устройств;

– обеспечение подачи питания ограниченной мощности на подключенные USB-устройства.

Существует два главных класса USB-устройств: устройства-концентраторы и устройства-функции. *Устройства-концентраторы* (hubs) обеспечивают дополнительное присоединение USB-узлов, а *устройства-функции* (functions) — подключение функциональных устройств. В одном USB-устройстве могут объединяться возможности устройств-функций и устройств-концентраторов, для подключения других функций (рис. 5.14).

Устройство-функция — устройство USB, которое способно передать или получить данные или управляющую информацию по шине. Функция обычно выполняется как отдельное периферийное устройство с кабелем, который подключается в порт концентратора (например, мышь, клавиатура). Каждая функция содержит информацию о конфигурации, которая описывает ее параметры и требования к ресурсам. Прежде чем устройство-функция будет использовано, оно должно быть сконфигурировано хостом. Такая конфигурация включает в себя распределение пропускной способности USB-шины и выбор специфических настроек конфигурации функции.

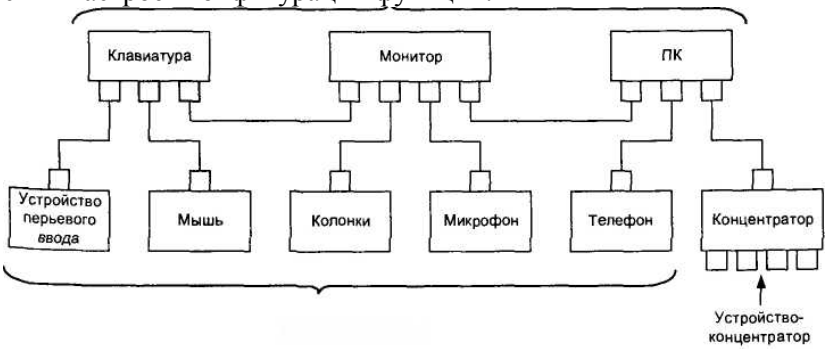


Рисунок 5.14. – Пример объединения концентраторов и функций

### 5.5.2. Конечные точки

Каждое логическое устройство USB состоит из набора независимо функционирующих конечных точек — endpoints (EP). Конечная точка — уникально идентифицируемая часть устройства USB, которая является конечным пунктом назначения потока связи между программным обеспечением хоста и устройством USB. Каждая конечная точка создается во время разработки и имеет свой уникальный идентификатор или номер конечной точки. Конечные точки находятся в неопределенном состоянии, и к ним нельзя обратиться, пока они не будут сконфигурированы (за исключением конечной точки «О»). Комбинация уникального адреса устройства, который присваивается USB-устройству при подключении его к шине, и номера конечной точки позволяет однозначно обращаться к каждой конечной точке внутри USB-устройства.

Каждая конечная точка имеет характеристики, и их необходимо знать ПО клиента для определения типа соединения:

– требования к частоте доступа и времени отклика на USB-шине;

– требования по пропускной способности канала связи с этой точкой;

– уникальный номер конечной точки;

– особенности реакции при обнаружении ошибок;

– максимальный размер пакета, с которым работает конечная точка;

– тип передачи для данной конечной точки;

– направление передачи данных — для блочных (bulk) и изохронных передач.

Все USB-устройства должны иметь конечную точку с номером «О» (Endpoint 0), через которую хост инициализирует, конфигурирует и управляет устройством USB. Конечная точка «О» обеспечивает доступ к информации о конфигурации USB-устройства, предоставляет возможность настраивать его режимы работы и всегда конфигурируется автоматически при подключении устройства к шине USB.

### 5.5.3. Пропускная способность USB-шины

Вся пропускная способность USB-шины может быть распределена среди множества различных потоков данных. Это позволяет широкому диапазону разноскоростных устройств присоединиться к USB-шине. USB-хост резервирует некоторую пропускную способность для конкретного канала только после его установления. Для USB-устройств, требующих большой пропускной способности, следует продумать вопросы буферизации, т. е. выделить большие по размеру буферы и обеспечить, чтобы аппаратная задержка буферизации не превышала нескольких миллисекунд.

Если при распределении дополнительного канала произойдет нарушение существующей пропускной способности или изменение времен отклика, USB блокирует распределение пропускной способности, и дальнейшее распределение каналов отклоняется или блокируется. Когда канал закрыт, выделенная ему пропускная способность освобождается и может быть перераспределена на другой канал.

### 5.5.4. Основные режимы работы

Как определено в спецификациях USB 1.0 и 1.1, имеются два режима передачи сигналов, которые могут использоваться на одной шине благодаря динамическому переключению скоростей. Полноскоростной режим передачи информации по USB-шине со скоростью 12 Мбит/с и низкоскоростной режим передачи сигналов в 1,5 Мбит/с, который имеет ряд функциональных ограничений и позволяет работать при меньшем уровне защиты от электромагнитных помех (EMI) и который определен, чтобы поддерживать ограниченное число низкоскоростных устройств (типа мыши), так как включение большого числа низкоскоростных устройств значительно снижает пропускную способность шины.

Определение скоростных характеристик устройства и самого факта включения его на шину производится благодаря имеющимся в устройстве pull-up-резисторам, подключенным к линиям D+ или D–. Подключение резистора к линии D+ сигнализи-

рует подключение полноскоростного устройства, к линии D– — низкоскоростного.

Для подключения внешних устройств USB позволяет иметь кабельный сегмент длиной до 5 м. USB-кабель состоит из 4 проводов:

1) Vbus — линия для передачи питания +5 вольт при максимальном токе в 500 мА;

2) D– — первый провод витой пары для передачи данных;

3) D+ — второй провод витой пары для передачи данных;

4) GND — цифровая «земля».

Проводники Vbus и GND служат для подводки питания (+5 В) к устройствам, которые запитываются от шины. Любое подключенное устройство может или использовать свое питание (self-powered), или получать питание от хоста или хаба (bus-powered). Отметим, что системное программное обеспечение USB-хоста управляет энергосбережением в сети, посылая устройствам команды войти или выйти из режима энергосбережения.

### 5.5.5. Основные принципы передачи данных

У USB передача данных и сигналов управления происходит между программным обеспечением хоста и особой конечной точкой в USB-устройстве. Хост USB обрабатывает связь с любой конечной точкой USB-устройства независимо от любой другой конечной точки. Такие соединения между программным обеспечением хоста и конечной точкой устройства USB называются каналами. Например, USB-устройство может иметь одну конечную точку, которая будет поддерживать канал для передачи данных в USB-устройство, и другую конечную точку, которая поддерживает канал для передачи данных из USB-устройства.

Стандарт USB определяет четыре типа передачи: Control, Interrupt, Bulk, Isochronous. Каждый тип передачи определяет различные характеристики потока связи:

– свой формат кадров данных для обмена по USB;

. направление передачи;

– ограничения на размер пакета;

– ограничения на доступ к шине;

– требуемую последовательность пакетов данных.

*Передача типа Управление (Control)* — пакетная, непериодическая передача управляющих сигналов. Программное обеспечение хоста использует этот тип передачи в режиме запрос-ответ для инициализации, настройки конфигурации USB-устройства или получения информации о статусе USB-устройства. Control-данные доставляются без потерь, так как хост резервирует часть каждого USB-кадра для передачи control-информации.

*Передача типа Bulk* — непериодическая, применяется для обмена большими массивами информации для данных, которые могут использовать любую доступную пропускную способность, не используемую другими типами передач в данный момент, и могут быть задержаны, пока не будет доступна нужная пропускная способность. Надежный обмен данными обеспечивается на аппаратном уровне, с использованием обнаружения ошибок на аппаратном уровне и автоматической повторной перепосылки поврежденных данных, но только ограниченное число раз.

*Передача типа Прерывание (Interrupt)* — передача по прерыванию — небольшая спонтанная непериодическая, низкочастотная передача небольших данных от USB-устройства, которая может быть произведена в любое время и будет передана по USB-шине со скоростью не меньшей, чем определено устройством. Данные прерывания обычно состоят из сообщений о произошедшем событии, символов или, например, координат из устройства управления, которые представляют собой один или несколько байт. Этот тип передачи похож на блочную (bulk), но передача происходит только для IN-каналов. Хотя большая скорость синхронизации ответа не требует, интерактивные данные могут иметь ограниченное время отклика, который должна поддерживать USB-шина.

*Изохронные (Isochronous)* или *потоковые (Streaming) передачи* данных в реальном времени, которые занимают заранее оговоренную пропускную способность USB-шины с заранее оговоренным временем отклика. Таким образом, это периодическая, непрерывная связь между хостом и устройством, которая обычно используется для передачи потоковой, критичной ко времени информации, такой как аудио или видеоинформация.

Таким образом, изохронные данные могут быть чувствительны к скорости и задержкам доставки, так как их поток непрерывен и требует обработки и передачи в реальном масштабе времени (например, речевая информация). Чтобы поддержать необходимую синхронизацию, изохронные данные должны передаваться по шине со скоростью их поступления. Если скорость доставки потоков этих данных не поддерживается на определенном уровне, то в потоке произойдут сбои из-за переполнения или обнуления буферов.

USB разработана так, чтобы минимизировать задержки изохронных передач данных; для этого изохронные потоки данных в USB занимают выделенную часть пропускной способности USB-шины, а это гарантирует, что данные могут доставляться с нужной скоростью. При изохронной передаче любая ошибка на физическом уровне не исправляется аппаратно путем повторений. Но эта проблема решается за счет того, что средняя частота передачи ошибочных битов в USB-шине достаточно мала.

# 6. АЛФАВИТНО-ЦИФРОВЫЕ ИНДИЦИРУЮЩИЕ ЖК-МОДУЛИ НА ОСНОВЕ КОНТРОЛЛЕРА HD44780

## 6.1. Введение

Контроллер HD44780 фирмы Hitachi фактически является промышленным стандартом и широко применяется при производстве алфавитно-цифровых ЖКИ-модулей. Аналоги этого контроллера или совместимые с ним по интерфейсу и командному языку микросхемы выпускают множество фирм, среди которых: Epson, Toshiba, Sanyo, Samsung, Philips. Еще большее число фирм производят ЖКИ-модули на базе данных контроллеров. Эти модули можно встретить в самых разнообразных устройствах: измерительных приборах, медицинском оборудовании, промышленном и технологическом оборудовании, офисной технике — принтерах, телефонах, факсимильных и копировальных аппаратах.

Алфавитно-цифровые ЖКИ-модули представляют собой недорогое и удобное решение, позволяющее сэкономить время и ресурсы при разработке новых изделий, при этом обеспечивают отображение большого объема информации при хорошей различимости и низком энергопотреблении. Возможность оснащения ЖКИ-модулей задней подсветкой позволяет эксплуатировать их в условиях с пониженной или нулевой освещенностью, а исполнение с расширенным диапазоном температур (–20°С...+70°С) в сложных эксплуатационных условиях, в том числе в переносной, полевой и даже иногда в бортовой аппаратуре.

Контроллер HD44780 потенциально может управлять 2-мя строками по 40 символов в каждой (для модулей с 4-мя строками по 40 символов используются два однотипных контроллера), при матрице символа 5x7 точек. Контроллер также поддерживает символы с матрицей 5x10 точек, но в последние годы ЖКИ-модули с такой матрицей практически не встречаются.

Существует несколько различных более-менее стандартных форматов ЖКИ-модулей (символов x строк): 8x2, 16x1, 16x2, 16x4, 20x1, 20x2, 20x4, 24x2, 40x2, 40x4. Встречаются и менее распространенные форматы: 8x1, 12x2, 32x2 и др., — принципиальных ограничений на комбинации и количество отображаемых

символов контроллер не накладывает — модуль может иметь любое количество символов от 1 до 80, хотя в некоторых комбинациях программная адресация символов может оказаться не очень удобной.

В рамках одного формата могут производиться ЖКИ-модули нескольких конструктивов, отличающихся как габаритами ЖКИ (и, как следствие, размерами символов), так и размерами платы и посадки. Например, фирма Powertip предлагает алфавитно-цифровые ЖКИ-модули 11-ти форматов (от 8x2 до 40x4) в 37-ми различных конструктивах, 16x1 в 6-ти, а модули формата 16x2 в 11-ти.

Изучая каталоги различных фирм-производителей ЖКИ-модулей, можно убедиться, что одни форматы и конструктивы являются собственными разработками и не обнаруживают аналогов в номенклатуре остальных фирм, другие являются фактическими стандартами и производятся большинством изготовителей. В качестве примера можно назвать ЖКИ-модуль формата 24x2, именуемый PC2402-A у Powertip, ED24200 у EDT, DMC-24227 у Optrex, SC2402A у Bolymin, MDLS-24265 у Varitronix, PVC240202 у Picvue и др., все эти модули имеют одинаковые конструктивные размеры и являются взаимозаменяемыми.

В рамках одного конструктива ЖКИ-модуль может иметь еще ряд модификаций. В частности, могут применяться несколько типов ЖКИ, отличающихся цветом фона и цветом символов, а также по применяемым ЖК-материалам и структуре: TN, STN и FSTN типа. ЖКИ STN и FSTN типа имеют более высокую стоимость, но одновременно обладают повышенной контрастностью и вдвое большим максимальным углом обзора, причем ЖКИ FSTN типа имеют лучшие характеристики, чем STN.

ЖКИ-модули могут оснащаться задней подсветкой, размещаемой между ЖКИ и печатной платой, для чего ЖКИ производятся с полупрозрачным или прозрачным задним слоем (в последнем случае считывание информации возможно только при наличии подсветки). Собственно подсветка может быть реализована несколькими способами: с помощью электролюминесцентной панели, представляющей собой тонкую пленку, излучающую свет при прикладывании переменного напряжения порядка

100...150 В; люминесцентной лампой с холодным катодом (также работающей при повышенном напряжении), излучение которой равномерно распределяется по всей площади ЖКИ с помощью отражателя или плоского световода; третий вариант — подсветка на основе светодиодной матрицы.

Первые два способа подсветки обеспечивают высокую яркость и могут иметь белый тон свечения при относительно низком потреблении, но требуют наличия источника повышенного напряжения, что создает некоторые трудности при создании аппаратуры с автономным питанием. Напротив, светодиодная подсветка не требует высоковольтного источника (прямое падение напряжения составляет 4,2 В) и при использовании несложного источника тока позволит производить питание от источника с напряжением 5 В. Кроме того, светодиодная подсветка имеет значительно большее (в десятки раз) время наработки, а также только она допустима к эксплуатации в расширенном диапазоне температур (–20°С...+70°С). С другой стороны, светодиодная подсветка потребляет ток, в десять раз превышающий ток потребления самого контроллера HD44780. Например, ЖКИ МТС S16208 фирмы Microtips (2x16) потребляет ток 5 мА, а светодиодная подсветка – 50 мА. Поэтому в каждом конкретном случае принимается решение, какой тип подсветки использовать и использовать ли ее вообще.

### 6.2. Подключение

Для соединения ЖКИ-модуля с управляющей системой используется параллельная синхронная шина, насчитывающая 8 или 4 (выбирается программно) линий данных DB0...DB7, линию выбора операции R/W, линию выбора регистра RS и линию стробирования/синхронизации E. Кроме линий управляющей шины имеются две линии для подачи напряжения питания 5 В — GND и $V_{CC}$, и линия для подачи напряжения питания драйвера ЖКИ – $V_0$.

Указанные выше названия линий шины являются стандартными, но существует множество различных вариантов расположения контактов у каждого конкретного конструктива ЖКИ-модуля. На самом деле, единственным реально стандартным ва-

риантом расположения контактов является одно- или двухрядное 14-ти контактное поле, расположенное на краю печатной платы модуля, или 16-ти контактное поле, содержащее дополнительную пару контактов с подключенными к ней выводами питания подсветки. В любом случае, для получения достоверной информации необходимо воспользоваться соответствующей справочной литературой изготовителя модуля.

Схема включения модуля, рассчитанного на стандартный диапазон температур, показана на рис.6.1.

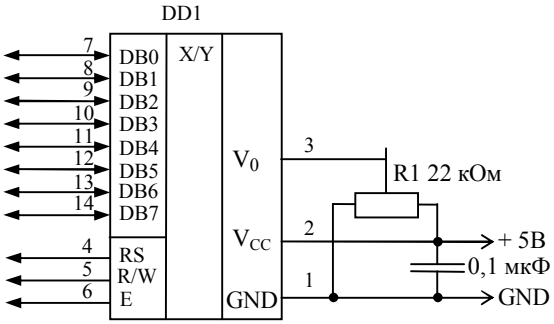


Рисунок 6.1. – Стандартная схема включения модуля ЖКИ

Подстроечный резистор R1 позволяет плавно менять напряжение питания драйвера ЖКИ, что приводит к изменению угла поворота жидких кристаллов. Этим резистором можно отрегулировать фактическую контрастность при некотором преимущественном угле наблюдения (снизу-вверх или сверху-вниз). Включение в данную схему ЖКИ-модуля, рассчитанного на расширенный диапазон температур, не приведет к успеху, так как из-за особенностей применяемых в них ЖК-материалов, эти ЖКИ требуют повышенного напряжения питания и при питании напряжением 5 В изображение либо будет отсутствовать совсем, либо будет слабоконтрастным. Для преодоления ситуации необходимо подать на вывод $V_0$ отрицательное напряжение (напряжение на ЖКИ определяется разностью $V_{CC}$ и $V_0$), составляющее в предельном случае — 5 В. Если в схеме отсутствует источник отрицательного напряжения, то не составляет труда собрать простейший преобразователь, например, по схеме на рис. 6.2.

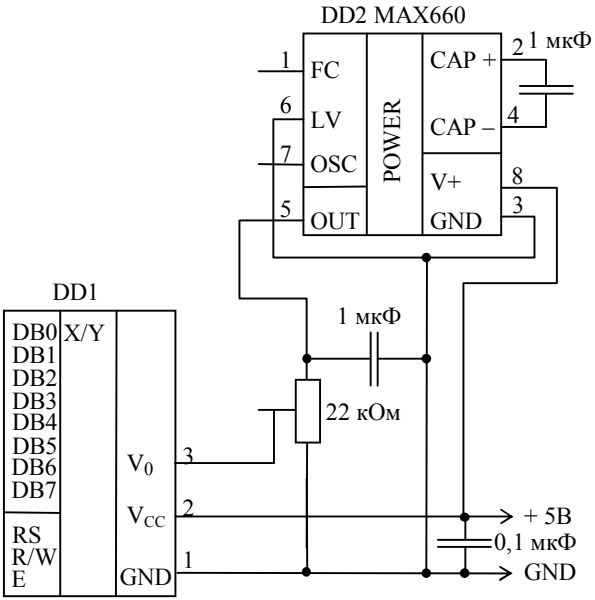


Рисунок 6.2. – Подача питания на ЖКИ-модуль

При включении питания после окончания цикла внутренней инициализации ЖК модуль включается в режим развертки одной верхней строки. При изменении напряжения на выводе $V_0$ сегменты этой строки должны менять свое состояние от прозрачного до непрозрачного, что является свидетельством правильного подключения питания модуля и работоспособности контроллера и драйверов ЖКИ.

Для соединения модуля с управляющей системой можно выбрать один из двух вариантов: по 8-ми или 4-х разрядной шине. В первом случае потребуется 11 сигнальных линий, во втором — только 7.

Рассмотрим варианты подключения ЖКИ к микроконтроллеру.

В 90-х годах традиционным способом являлось подключение ЖКИ-модуля к системной шине микропроцессора (если таковая имеется) и выполнение обмена в синхронном режиме с максимальной скоростью. Этому широко распространенному

способу присущ ряд недостатков. Во-первых, большинство современных устройств выполняется с применением однокристальных микро-ЭВМ без использования дополнительной внешней памяти и, как следствие, системная шина у этих устройств просто отсутствует. Во-вторых, в современных системах повышенной сложности и производительности, у которых присутствует дополнительная память и, естественно, системная шина, скорость операций на шине находится за пределами возможностей контроллера HD44780 (2 МГц при питании 5 В и 1 МГц при 3 В). Это может потребовать введения дополнительных схем для замедления скорости работы шины при выполнении операций обмена с ЖКИ-модулем. В-третьих, подключение к системной шине в большинстве случаев потребует вводить схемы дешифрации и формирования сигналов E и R/W, что опять приведет к дополнительным затратам. Все сказанное выше не означает, что вариант с подключением к шине принципиально неэффективен. В какой-то конкретной системе этот способ, наоборот, может быть самым оптимальным. Кроме того, некоторые современные процессоры имеют встроенные средства для формирования сигналов выборки (CS), с возможностью программно определить скорость обмена с каждым конкретным устройством.

Другой очень простой вариант — обмен с ЖКИ-модулем выполняется программными средствами через порты ввода-вывода управляющего микроконтроллера. В дальнейшем мы сконцентрируем внимание именно на этом варианте, так как он позволяет рассмотреть общий случай, абстрагируясь от конкретной системы. Вариант соединения с системной шиной, напротив, требует рассмотрения конкретных устройств, поэтому в случае необходимости можно рекомендовать изучить временные диаграммы операций чтения и записи, приведенные на рис. 6.5 и 6.6, а также значения временных параметров, приведенные в табл. 6.1, 6.2 и сконструировать соответствующие управляющие схемы.

На рис. 6.3 приведена схема подключения ЖКИ-модуля с 8-ми разрядной шиной к некоторому абстрактному микроконтроллеру. Этот микроконтроллер содержит два порта: 8-ми разрядный двунаправленный PX0...PX7, к которому подключена шина

DB0...DB7 ЖКИ-модуля, и 3-х разрядный PY0...PY2, к которому подключены линии управляющих сигналов: E, RS и R/W.

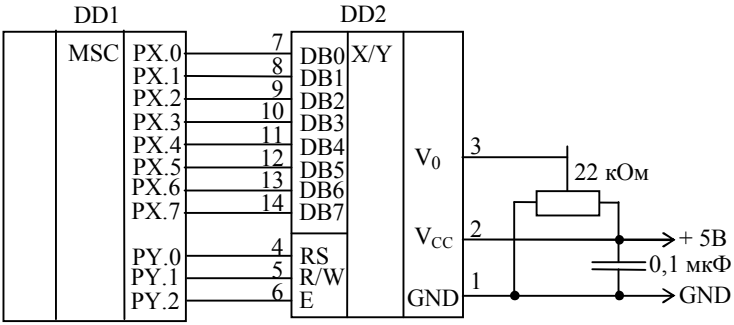


Рисунок 6.3. – Подключение к управляющей системе в 8-битном режиме

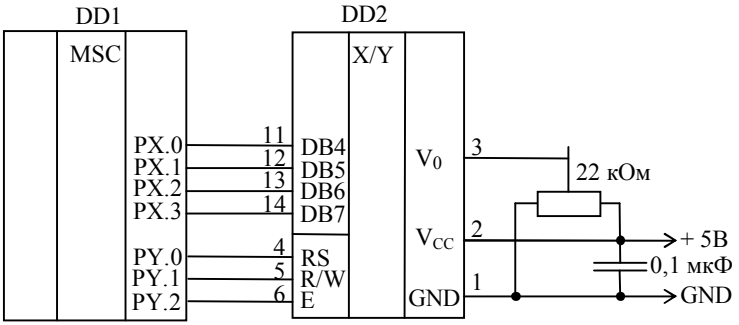


Рисунок 6.4. – Подключение к управляющей системе в 4-битном режиме

На рис. 6.4 представлен вариант схемы подключения ЖКИ-модуля к этому же микроконтроллеру в 4-х разрядном режиме. Обратите внимание, что для обмена в 4-х разрядном режиме используется старшая тетрада шины данных — DB4...DB7.

В соответствии с временной диаграммой записи (рис. 6.5) и диаграммой чтения (рис. 6.6) в исходном состоянии E = 0, сигнал R/W = 0, значение сигнала RS – произвольное, шина данных DB0...DB7 – в состоянии высокого импеданса (HI). Такое состояние управляющих сигналов (E и R/W) должно поддерживаться все время в промежутках между операциями обмена с ЖКИ-

модулем. Шина данных в эти моменты свободна, и может использоваться в мультиплексном режиме для каких-либо других целей, например, для сканирования матрицы клавиатуры. И, конечно, нужно позаботиться об исключении конфликтов на шине данных в момент совершения операций обмена с ЖКИ-модулем.

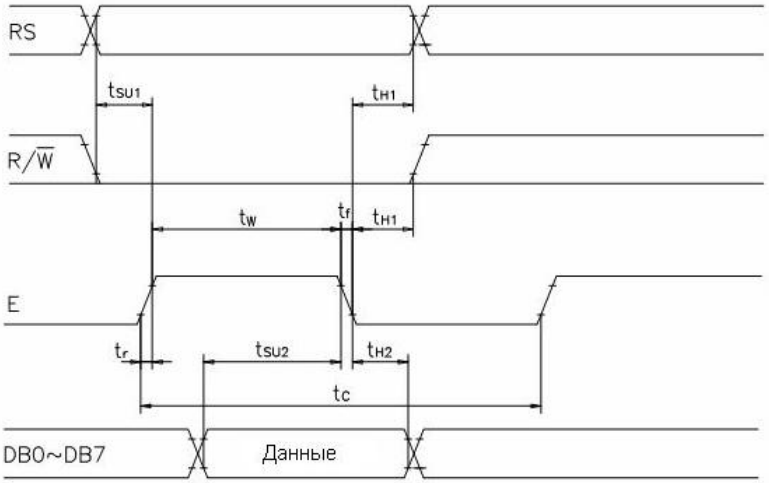


Рисунок 6.5. – Временная диаграмма операции записи

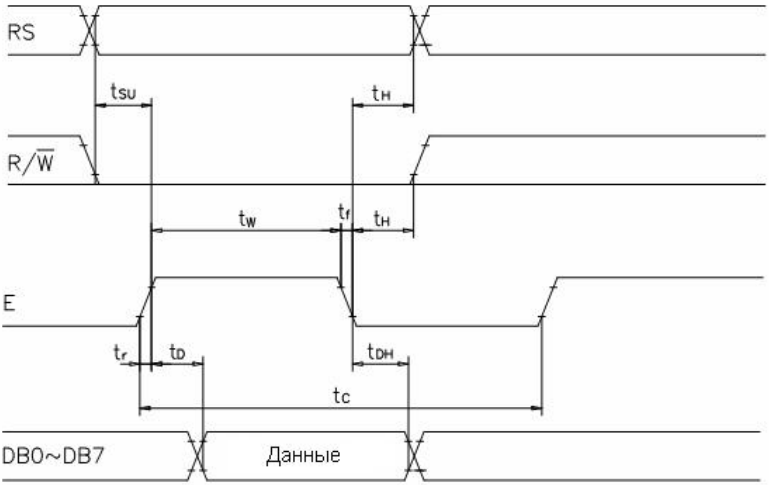


Рисунок 6.6. – Временная диаграмма операции чтения

Таблица 6.1.

Значения временных характеристик. Операция записи

| Параметр | Обозн. | Мин. | Макс. | Един. |
|---|---|---|---|---|
| Период сигнала E | $t_C$ | 500 | – | нс |
| Положит. полупериод сигнала E | $t_W$ | 230 | – | нс |
| Фронт/спад сигнала E | $t_r$, $t_f$ | – | 20 | нс |
| Установление адреса | $t_{SU1}$ | 40 | – | нс |
| Удержание адреса | $t_{H1}$ | 10 | – | нс |
| Установление данных | $t_{SU2}$ | 80 | – | нс |
| Удержание данных | $t_{H2}$ | 10 | – | нс |

Таблица 6.2.

Значения временных характеристик. Чтение.

| Параметр | Обозн. | Мин. | Макс. | Ед. |
|---|---|---|---|---|
| Период сигнала E | $t_C$ | 500 | – | нс |
| Положит. полупериод сигнала E | $t_W$ | 230 | – | нс |
| Фронт/спад сигнала E | $t_r$, $t_f$ | – | 20 | нс |
| Установление адреса | $t_{SU}$ | 40 | – | нс |
| Удержание адреса | $t_H$ | 10 | – | нс |
| Установление данных | $t_D$ | – | 160 | нс |
| Удержание данных | $t_{DH}$ | 5 | – | нс |

Последовательности действий, необходимые микроконтроллеру при совершении операций записи и чтения для 8-ми и 4-х разрядной шины, приведены соответственно в табл. 6.3...6.6. Время выполнения каждого шага составляет не менее 250 нс. При использовании быстродействующих микроконтроллеров это условие может быть нарушено, поэтому необходимо контролировать минимальные значения временных интервалов, чтобы они находились в области допустимых значений, указанных в табл. 6.1, 6.2 и при необходимости вводить задержки.

Таблица 6.3.

Операция записи для 8-ми разрядной шины

| |
|---|
| 1. Установить значение линии RS (0 или 1) |
| 2. Вывести значение данных на линии шины DB0...DB7 |
| 3. Установить линию E = 1 |
| 4. Установить линию E = 0 |
| 5. Установить линии шины DB0...DB7 = HI |

Таблица 6.4.

Операция чтения для 8-ми разрядной шины

| |
|---|
| 1. Установить значение линии RS (0 или 1) |
| 2. Установить линию R/W = 1 |
| 3. Установить линию E = 1 |
| 4. Считать значение данных с линий шины DB0...DB7 |
| 5. Установить линию E = 0 |
| 6. Установить линию R/W = 0 |

Таблица 6.5.

Операция записи для 4-х разрядной шины

| |
|---|
| 1. Установить значение линии RS (0 или 1) |
| 2. Вывести значение старшей тетрады байта данных на линии шины DB4...DB7 |
| 3. Установить линию E = 1 |
| 4. Установить линию E = 0 |
| 5. Вывести значение младшей тетрады байта данных на линии шины DB4...DB7 |
| 6. Установить линию E = 1 |
| 7. Установить линию E = 0 |
| 8. Установить линии шины DB4...DB7 = HI |

Таблица 6.6.

Операция чтения для 4-х разрядной шины

| |
|---|
| 1. Установить значение линии RS (0 или 1) |
| 2. Установить линию R/W = 1 |
| 3. Установить линию E = 1 |
| 4. Считать значение старшей тетрады байта данных с линий шины DB4...DB7 |
| 5. Установить линию E = 0 |
| 6. Установить линию E = 1 |
| 7. Считать значение младшей тетрады байта данных с линий шины DB4.. .DB7 |
| 8. Установить линию E = 0 |
| 9. Установить линию R/W = 0 |

Описанные выше операции записи/чтения байта являются базовыми для осуществления обмена с ЖКИ-модулем. Реализация этих двух операций — единственное, что отличает процесс обмена по 8-ми разрядной шине от обмена по 4-х разрядной шине. На основе этих двух операций, реализованных программно (когда модуль подключен к портам микро-ЭВМ), или аппаратно (когда модуль подключен к системной шине), строятся все виды операций программирования и управления.

### 6.3. Программирование и управление

Перед началом рассмотрения принципов управления ЖКИ-модулем обратимся к внутренней структуре контроллера HD44780, чтобы понять принципы построения ЖКИ-модулей на его основе. Эта информация позволит понять способы организации модулей различных форматов с точки зрения программной модели, а также мотивации конструкторов ЖКИ-модулей.

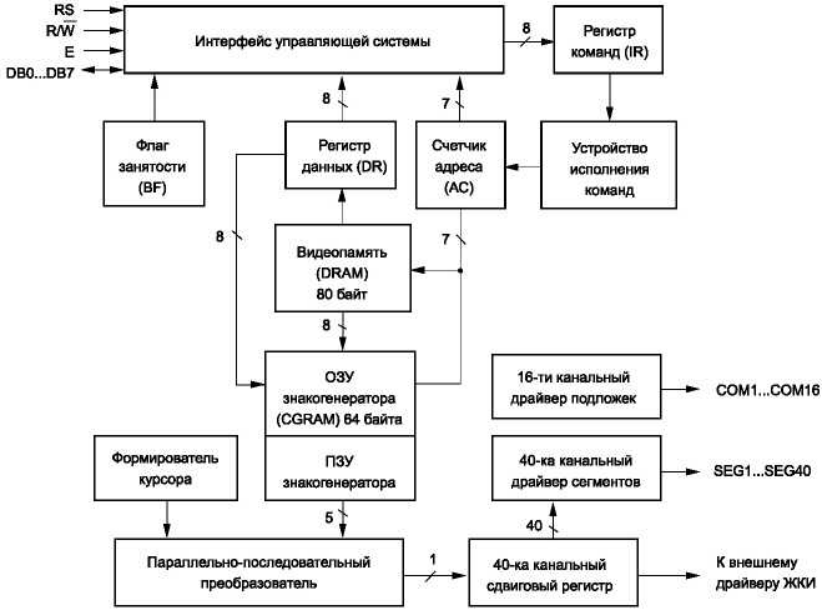


Рисунок 6.7. – Упрощенная структурная схема контроллера HD44780

Упрощенная структурная схема контроллера приведена на рис. 6.7. Можно сразу выделить основные элементы, с которыми приходится взаимодействовать при программном управлении: регистр данных (DR), регистр команд (IR), видеопамять (DDRAM), ОЗУ знакогенератора (CGRAM), счетчик адреса памяти (AC), флаг занятости контроллера.

Другие элементы не являются объектом прямого взаимодействия с управляющей программой — они участвуют в процессе регенерации изображения на ЖКИ: знакогенератор, формирователь курсора, сдвиговые регистры и драйверы (напоминаем, что приведенная схема — упрощенная, и многие не важные для получения общей картины промежуточные элементы на ней опущены).

Управление контроллером ведется посредством интерфейса управляющей системы. Основными объектами взаимодействия являются регистры DR и IR. Выбор адресуемого регистра производится линией RS. Если RS = 0 — адресуется регистр команд (IR), если RS = 1 — регистр данных (DR).

Данные через регистр DR, в зависимости от текущего режима, могут помещаться (или прочитываться) в видеопамять (DDRAM) или в ОЗУ знакогенератора (CGRAM) по текущему адресу, указываемому счетчиком адреса (AC). Информация, поступающая в регистр IR, интерпретируется устройством выполнения команд как управляющая последовательность. Прочтение регистра IR возвращает в 7-ми младших разрядах текущее значение счетчика AC, а в старшем разряде флаг занятости (BF).

Видеопамять, имеющая общий объем 80 байт, предназначена для хранения кодов символов, отображаемых на ЖКИ. Видеопамять организована в две строки по 40 символов в каждой. Эта привязка является жесткой и не подлежит изменению. Другими словами, независимо от того, сколько реальных строк будет иметь каждый конкретный ЖКИ-модуль, скажем, 80x1 или 20x4, адресация видеопамяти всегда производится как к двум строкам по 40 символов.

Будучи устройством с динамической индикацией, контроллер циклически производит обновление информации на ЖКИ. Сам ЖКИ организован как матрица, состоящая в зависимости от режима работы из 8-ми (одна строка символов 5x7 точек), 11-ти

(одна строка символов 5x10 точек) или 16-ти (две строки символов 5x7 точек) строк по 200 сегментов (когда строка насчитывает 40 символов) в каждой. Собственный драйвер контроллера HD44780 имеет только 40 выходов (SEG1...SEG40) и самостоятельно может поддерживать только 8-ми символьные ЖКИ. Это означает, что ЖКИ-модули форматов до 8x2 реализованы на одной единственной микросхеме HD44780, модули, имеющие большее количество символов, содержат дополнительные микросхемы драйверов, например, HD44100, каждая из которых дополнительно предоставляет управление еще 40-ка сегментами.

Особняком стоят ЖКИ-модули формата 16x1. Они также реализованы с помощью одной единственной микросхемы HD44780, но одна 16-ти символьная строка в них фактически составлена из двух 8-ми символьных. И хотя это усложняет программное управление, ведь строка оказывается логически разорванной посередине, тем не менее, экономически это оправдано, ибо позволило создать ЖКИ-модуль, содержащий всего одну микросхему. Другой вариант пространственной адресации встречается в 4-х строчных модулях. Из-за проблем разводки токоведущих дорожек первая и вторая строки этих модулей являются таковыми как обычно, третья же является продолжением первой строки, а четвертая — второй.

У контроллера HD44780 существует набор внутренних флагов, определяющих режимы работы различных элементов контроллера (таблица 6.7).

В таблице 6.8 приведены значения управляющих флагов непосредственно после подачи на ЖКИ-модуль напряжения питания. Переопределение значений флагов производится специальными командами, записываемыми в регистр IR, при этом комбинации старших битов определяют группу флагов или команду, а младшие содержат собственно флаги.

Таблица 6.7.

Флаги, управляющие работой контроллера HD44780

| | |
|---|---|
| I/D | режим смещения счетчика адреса AC:<br>0 – уменьшение, 1 – увеличение. |
| S | Флаг режима сдвига содержимого экрана. 0 — сдвиг экрана не производится, 1 – после записи в DDRAM очередного кода экран сдвигается в направлении, определяемым флагом I/D: 0 – вправо, 1 – влево. При сдвиге не производится изменение содержимого DDRAM. изменяются только внутренние указатели расположения видимого начала строки в DDRAM. |
| S/C | Флаг-команда, производящая вместе с флагом R/L операцию сдвига содержимого экрана (так же, как и в предыдущем случае, без изменений в DDRAM) или курсора. Определяет объект смещения: 0 – сдвигается курсор, 1 – сдвигается экран. |
| R/L | Флаг-команда, производящая вместе с флагом S/C операцию сдвига экрана или курсора. Уточняет направление сдвига: 0 – влево, 1 – вправо. |
| D/L | Флаг, определяющий ширину шины данных:<br>0 – 4 разряда, 1 – 8 разрядов. |
| N | Режим развертки изображения: 0 – 1 строка, 1 – 2 строки |
| F | Размер матрицы символов: 0 – 5x8 точек, 1 – 5x10 точек. |
| D | Наличие изображения: 0 — выключено, 1 – включено |
| C | Курсор в виде подчерка: 0 — выключен, 1 – включен |
| B | Курсор в виде мерцающего знакоместа: 0 – выкл., 1 – вкл. |

Таблица 6.8.

Значения управляющих флагов после подачи питания

| | |
|---|---|
| I/D = 1: | режим увеличения счетчика на 1 |
| S = 0: | без сдвига изображения |
| D/L = 1: | 8-ми разрядная шина данных |
| N = 0: | режим развертки одной строки |
| F = 0: | символы с матрицей 5x8 точек |
| D = 0: | отображение выключено |
| C = 0: | курсор в виде подчерка выключен |
| B = 0: | курсор в виде мерцающего знакоместа выключен |

Так как на момент включения ЖКИ-модуль ничего не отображает (флаг D = 0), то для того, чтобы вывести какой-либо текст необходимо, как минимум, включить отображение, уста-

новив флаг D = 1. Пример широко распространенной последовательности для инициализации ЖКИ-модуля: $38, $0C, $06. $38 устанавливает режим отображения 2-х строк с матрицей 5x8 точек и работу с 8-ми разрядной шиной данных; $0C включает отображение на экране ЖКИ-модуля, без отображения курсоров; $06 устанавливает режим автоматического перемещения курсора слева-направо после вывода каждого символа.

Список управляющих комбинаций битов регистра IR и выполняемые ими команды приведены в таблице 6.9.

Таблица 6.9.

Управляющие комбинации битов регистра IR

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Назначение |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Очистка экрана, AC = 0, адресация AC на DDRAM |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | – | AC = 0, адресация на DDRAM, сброшены сдвиги, начало строки адресуется в начале DDRAM |
| 0 | 0 | 0 | 0 | 0 | 1 | I/D | S | Выбирается направление сдвига курсора или экрана |
| 0 | 0 | 0 | 0 | 1 | D | C | B | Выбирается режим отображения |
| 0 | 0 | 0 | 1 | S/C | R/L | – | – | Команда сдвига курсора/экрана |
| 0 | 0 | 1 | DL | N | F | – | – | Определение параметров развертки и ширины шины данных |
| 0 | 1 | AG | AG | AG | AG | AG | AG | Присвоение счетчику AC адреса в области CGRAM |
| 1 | AD | AD | AD | AD | AD | AD | AD | Присвоение счетчику AC адреса в области DDRAM |

Контроллер HD44780 поддерживает как операции записи, так и операции чтения. Чтение регистра DR приводит к загрузке содержимого DDRAM или CGRAM, в зависимости от текущего режима, при этом курсор смещается на одну позицию, как и при записи. Чтение регистра IR возвращает 8 значащих разрядов, причем в 7-ми младших содержится текущее значение счетчика AC (7 разрядов, если адресуется DDRAM, и 6 – если CGRAM), а в старшем – флаг занятости BF. Этот флаг имеет значение 1 когда контроллер занят и 0 — когда свободен. Нужно учитывать, что большинство операций, выполняемых контроллером, зани-

мают значительное время, около 40 мкс, а время выполнения некоторых доходит до единиц миллисекунд, поэтому цикл ожидания снятия флага BF должен обязательно присутствовать в программах драйвера ЖКИ-модуля и предшествовать совершению любой операции (кроме операции проверки флага BF).

Необходимо обратить внимание на один важный момент. После совершения операции записи или чтения DDRAM и появления после нее признака готовности (BF = 0), прочитанное в этом же цикле (вместе с флагом BF) значение AC скорее всего не будет достоверным. Дело в том, что между появлением признака готовности и вычислением контроллером нового значения AC существует некоторый временной интервал, составляющий около 4 мкс при тактовой частоте контроллера 270 кГц. Поэтому, если необходимо получить истинное значение AC, нужно совершить повторную операцию прочтения IR спустя не менее чем 4 мкс (если контроллер работает на частоте 270 кГц, время ожидания необходимо пропорционально увеличить).

Вывод на экран символа производится записью его кода в регистр DR. При этом символ размещается в DDRAM по текущему адресу, указываемому AC, а значение AC увеличивается или уменьшается на 1. Чтобы произвести переустановку курсора на нужную позицию, необходимо присвоить AC соответствующее значение (см. табл. 6.9).

Когда производится последовательная запись символов и в результате заполняется вся строка, курсор автоматически переходит на вторую строку, но если необходимо принудительно установить курсор, скажем, на начало второй строки, то будет неверным присвоить AC, казалось бы, логичное значение $28 (40). Правильным является значение $40 (64). Значения адресов DDRAM в диапазоне $28...$3F (как и в диапазоне $68...$7F) являются неопределенными и результаты работы с ними могут быть непредсказуемыми.

Нужно учитывать, что контроллеры, устанавливаемые на ЖКИ-модули, могут иметь разные наборы символов, причем это может зависеть как от производителя контроллера, так и от модификации конкретной модели. Например, фирма Powertip выпускает ЖКИ-модули с четырьмя базовыми модификациями наборов символов: японской, европейской, французской и русской.

Таблица 6.10.

Таблица знакогенератора.

Более того, существует как минимум два варианта русского набора символов: контроллер фирмы Hitachi (H2 по маркировке фирмы Powertip) и контроллер фирмы Epson (EH по маркировке Powertip). Контроллер Hitachi обладает существенным недостатком — у него весьма ограниченный набор русских символов, фактически у него имеются только прописные русские буквы, и даже среди них отсутствует символ «Ф». Напротив, контроллер фирмы Epson содержит полный набор русских символов в прописном и строчном вариантах, поэтому он весьма удобен для отечественных применений. Это свойство контроллеров фирмы Epson обеспечило им популярность на российском рынке, поэтому в последнее время основная масса импортируемых в нашу страну ЖКИ-модулей оснащены именно этим контроллером.

Из допустимых для размещения в DDRAM кодов символы с кодами $00...$07 (и их дубликат с кодами $08...$0F) имеют специальное назначение — это переопределяемые символы, графи-

ческое изображение которых может назначить сам потребитель, разместив соответствующую информацию в области CGRAM. Для программирования доступны 8 переопределяемых символов в режиме с матрицей 5x7 точек и 4 с матрицей 5 x 10 (в режиме 5 x 10 переопределяемые символы адресуются кодами DDRAM через один: $00, $02, $04, $06). Для кодирования матрицы используются горизонтально «уложенные» байты, пять младших битов которых несут информацию о рисунке (причем 1 означает, что сегмент будет включен), 4-й разряд каждого из 8-ми (или 11-ти в режиме 5 x 10) байтов матрицы определяет левую колонку символа, а 0-й — правую. Старшие три бита не используются, как и старшие пять байтов, составляющих полную область матрицы символа (16 байтов) в режиме 5x10 (обратите внимание, что матрица программируемых символов допускает использование полной высоты строки (8 строк для режима 5x7 и 11 строк для режима 5x10), то есть можно размещать точки в области подчеркивающего курсора). Чтобы определить собственный символ, необходимо установить счетчик AC на адрес начала матрицы требуемого символа в CGRAM — $00, $08, $10 и т.д. ($00, $10, $20 для режима 5x10 точек) — произвести перезапись всех байтов матрицы, начиная с верхней строки. После этого, записав в DDRAM код запрограммированного символа: $00, $01, $02 ($00, $02, $04 для режима 5x10 точек), на экране в соответствующем месте будет отображаться переопределенный символ.

Производители контроллера рекомендуют выполнять следующую последовательность действий для инициализации.

1. Выдержать паузу не менее 15 мс между установлением рабочего напряжения питания (>4,5 В) и выполнением каких-либо операций с контроллером. Первой операцией выполнить команду, выбирающую разрядность шины (это должна быть команда $30 независимо от того, какой разрядности интерфейс вы собираетесь использовать в дальнейшем), причем перед выполнением этой операции не проверять значение флага BF.

2. Опять выдержать паузу не менее 4,1 мс и повторить команду выбора разрядности шины, причем перед подачей команды вновь не производить проверку флага BF.

3. Вновь выдержать паузу 100 мкс, и в третий раз повторить команду установки разрядности шины, вновь без проверки BF.

Эти три операции являются инициализирующими и призваны вывести контроллер в исходный режим работы (то есть перевести в режим работы с 8-ми разрядной шиной) из любого состояния. Следом за ними нормальным порядком (без выдерживания пауз, но с проверкой флага BF) выполняется инициализация режимов работы с выдачей инициализирующей последовательности, аналогичной указанной в таблице 7 (содержащей, в том числе, команду выбора необходимой разрядности шины).

Необходимо помнить, что когда объявляется режим работы с 4-х разрядной шиной, то есть выдается команда $20, то делается это обычно из 8-ми разрядного режима, который устанавливается автоматически после подачи напряжения питания, а, значит, невозможно адекватно объявить необходимое значение флагов N и F, располагающихся в младшей тетраде команды установки разрядности шины. Поэтому команду необходимо повторить в уже установившемся 4-х разрядном режиме путем последовательной передачи двух тетрад, то есть способом для 4-х разрядного режима.

В заключение хотелось бы поделиться соображениями из практического опыта применения ЖКИ-модулей в процессе разработки и отладки реальных устройств. Не смотря на наличие «экономичного» 4-х разрядного режима, 7 необходимых для связи линий могут оказаться чрезмерным требованием для приборов, которые в современных условиях нередко строятся с применением микро-ЭВМ в 16- или 28-ми выводных корпусах и имеющих ограниченный ресурс свободных портов. Оказалось, что часто, если конечно речь не идет о жесткой экономии каждой копейки, наиболее удобным способом использования ЖКИ-модуля становится создание отдельного контроллера на базе конкретной микро-ЭВМ с каким-либо последовательным интерфейсом, осуществляющего посредничество между управляющей системой и ЖКИ-модулем. Для тех, кто ценит свое время и не имеет желания разрабатывать такой контроллер самостоятельно, есть готовое решение — микросхема CE110, созданная на базе микро-ЭВМ в 28-ми выводном корпусе SDIP или SOIC. Микросхема CE110 выполняет роль контроллера ЖКИ-модуля и клавиатуры (до 64-х клавиш) и предлагает в качестве интерфейса связи с управляющей системой двухпроводную шину I2C.

# 7. ПРОГРАММИРУЕМАЯ ЛОГИКА

## 7.1. Классификация интегральных схем программируемой логики

Микросхемы программируемой логики классифицируются по нескольким признакам. По *уровню интеграции* их можно разделить на простые, сложные и схемы типа «системы на кристалле» (SOC, System On Chip), как показано на рис. 7.1. Можно сказать также, что простые ИС программируемой логики относятся к первому их поколению, тогда как сложные и SOC принадлежат к следующим.

Простые ИС с программируемой логикой (ИС ПЛ), обозначаемые в совокупности как PLD (Programmable Logic Devices), делятся на микросхемы программируемой матричной логики ПМЛ (PAL, Programmable Array Logic) и микросхемы программируемых логических матриц ПЛМ (PLA, Programmable Logic Array). Усложненные варианты PAL некоторые производители называют схемами GAL (Generic Array Logic). Простые ИС ПЛ рассчитаны на реализацию систем переключательных функций и использовались для замены нескольких корпусов или даже десятков корпусов стандартных ИС на один корпус PLD. По мере их усложнения решаемые ими задачи также усложнялись, в частности, появилась ориентация на реализацию конечных автоматов (в схемы PLD стали вводить элементы памяти).

Продолжением линии развития ПМЛ стали сложные БИС/СБИС типа CPLD (Complex PLD), в которых, как и в PLD, используются схемы непосредственной реализации дизъюнктивных нормальных форм переключательных функций (функций типа SOP, Sum Of Products), но в одной CPLD имеется несколько ПМЛ (PAL, GAL), объединенных системой коммутации.

Сложные ИС ПЛ типа FPGA (Field Programmable Gate Arrays) содержат матрицу логических блоков того или иного типа, расположенных по строкам и столбцам, между которыми размещены средства коммутации, позволяющие с помощью программирования получать необходимые взаимные соединения логических блоков. Сами блоки могут быть выполнены различ-

ным образом. Стремление объединить достоинства, присущие CPLD и FPGA, привело к созданию БИС/СБИС комбинированной архитектуры, для которых еще не выработано общепринятое название.

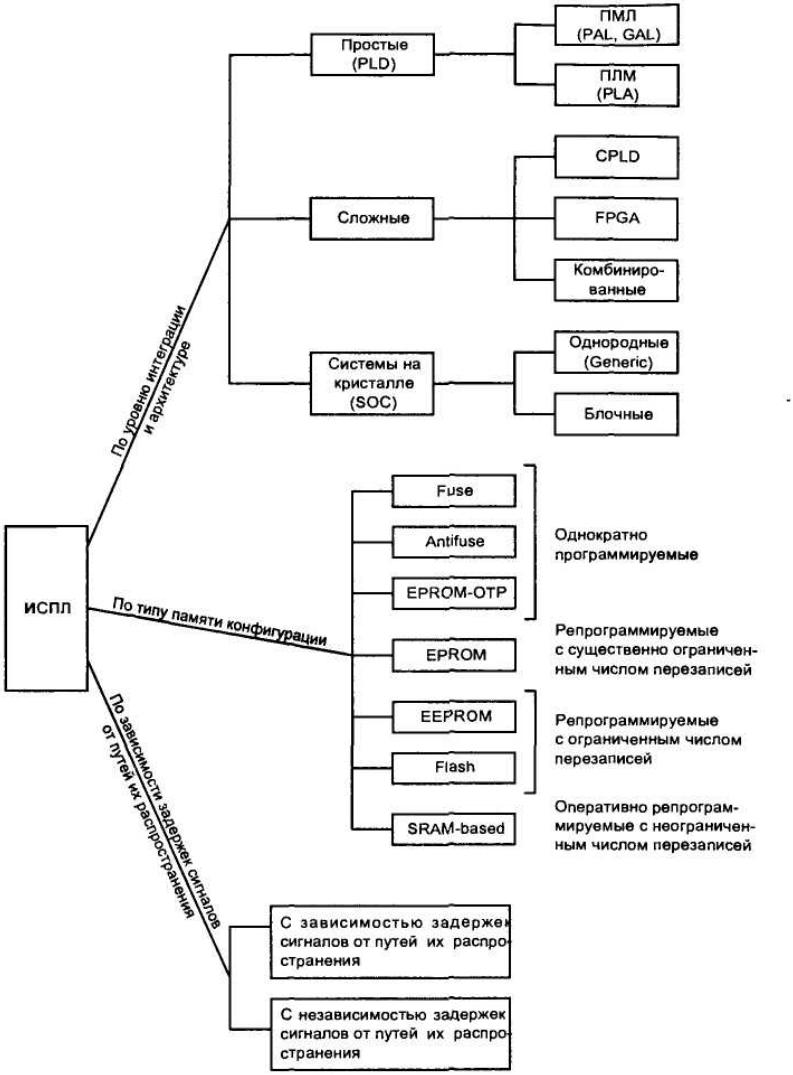


Рисунок 7.1. – Классификация ИС программируемой логики

Рост уровня интеграции дал возможность размещать на кристалле схемы, сложность которых соответствует целым системам (мегавентильные схемы). Такие схемы именуются SOC и могут быть разделены на два типа — однородные схемы, в которых функциональное назначение отдельных областей кристалла обеспечивается программированием одних и тех же по типу ресурсов (схемы типа generic), и блочные структуры, в которых отдельные области кристалла специализированы уже при их изготовлении. Про такие кристаллы говорят, что они содержат специализированные аппаратные ядра (Hardcores).

Важным классификационным признаком ИС ПЛ является *тип памяти конфигурации,* т. е. тип программируемых элементов, задаваемое состояние которых как раз и создает требуемое устройство как конкретный вариант межсоединений имеющихся на кристалле схемотехнических ресурсов. Программируемые элементы представляют собой двухполюсники, играющие роль ключей, которым при программировании задаются состояния «замкнуто» или «разомкнуто». Число программируемых элементов в ИС ПЛ зависит от их сложности и в схемах наибольшего уровня интеграции измеряется миллионами.

В ИС ПЛ используются или использовались ранее следующие типы программируемых элементов:

– плавкие перемычки Fuse (в схемах первых образцов);

– пробиваемые диэлектрические перемычки Antifuse (краткий русский термин отсутствует);

– однократно заряжаемые «плавающие затворы» МОП-транзисторов (EPROM-OTP);

– перезаряжаемые «плавающие затворы» с введением заряда электрическими воздействиями на транзистор и его стиранием с помощью облучения кристалла ультрафиолетовыми лучами (EPROM);

– перезаряжаемые «плавающие затворы» с электрическими записью и стиранием зарядов (EEPROM, Flash);

– ключевые МОП-транзисторы, управляемые триггерами памяти конфигурации (SRAM-based).

Репрограммируемые элементы EPROM, EEPROM, Flash, SRAM-based различаются по свойствам. Элементы EPROM с ультрафиолетовым стиранием допускают ограниченное число

перезаписей заряда, так как процесс облучения постепенно изменяет свойства кристалла.

Элементы с электрическим стиранием имеют существенно большее число допустимых перезаписей заряда (приблизительно в тысячу раз), а элементы с триггерной памятью конфигурации могут репрограммироваться неограниченно.

Одним из признаков классификации служит *наличие или отсутствие связи между задержками распространения сигналов и конкретными путями их передачи* по межсоединениям кристалла. Этот фактор важен, так как независимость задержки от конкретного пути передачи сигнала означает предсказуемость задержек, что существенно облегчает построение на кристалле рабочих схем, особенно схем высокого быстродействия.

## 7.2. Конструктивно-технологические типы современных программируемых элементов

Программируемые перемычки типа «antifuse» (рис. 7.2, 7.3) в исходном состоянии (до программирования) имеют чрезвычайно большие сопротивления (токи утечки порядка фемтоампер). Программирующий импульс напряжения пробивает трехслойный диэлектрик с чередованием слоев «оксид-нитрид-оксид» и создает проводящий поликремниевый канал между поликремниевым электродом и диффузионной областью n+. В зависимости от тока через перемычку в режиме ее программирования можно получить проводящий участок с сопротивлениями от 100 Ом и выше при очень малой паразитной емкости.

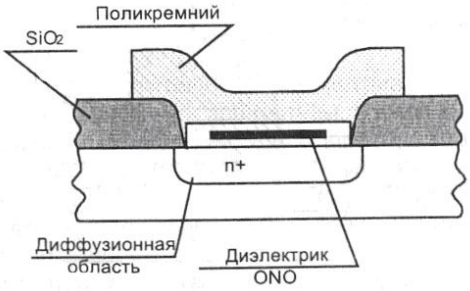


Рисунок 7.2. – Перемычка типа «antifuse» до программирования

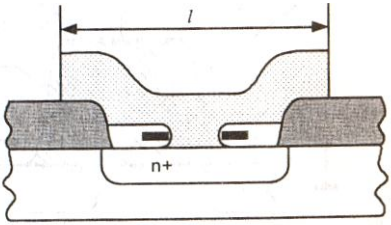


Рисунок 7.3. – Перемычка типа «antifuse» после программирования

Элементы EEPROM и Flash реализуются на ЛИЗМОП-транзисторах (название транзистора отражает процесс лавинной инжекции заряда в плавающий затвор). На рис. 7.4 показан ЛИЗМОП-транзистор с двумя затворами — плавающим и обычным.

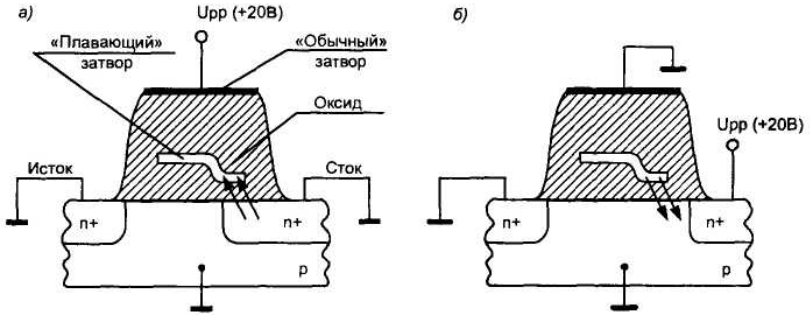


Рисунок 7.4. – Схематическое представление ЛИЗМОП-транзистора с двойным затвором

Плавающий затвор не имеет внешнего вывода и как бы погружен в диэлектрик (оксид, т. е. двуокись кремния). В этом затворе может создаваться или ликвидироваться заряд электронов. При подаче на обычный затвор повышенного значения программирующего напряжения через тонкий слой оксида электроны туннелируют в плавающий затвор, в котором создается заряд отрицательного знака. После снятия программирующего напряжения и возврата напряжения на затворе к уровню рабочих напряжений электроны оказываются в ловушке, где могут сохра-

няться в течение десятков лет. При этом транзистор будет заперт, так как отрицательный заряд плавающего затвора создает электрическое поле, противодействующее полю положительно заряженного затвора. При отсутствии заряда в плавающем затворе рабочее положительное напряжение на внешнем затворе обеспечивает отпирание транзистора (создает между стоком и истоком проводящий канал). На рис. 7.4, *а* показан режим программирования ЛИЗМОП-транзистора, а на рис. 7.4, *б* — режим стирания заряда. Стрелками показаны пути туннелирования электронов через тонкий слой оксида.

Память конфигурации типа EEPROM на основе ЛИЗМОП для обновления содержимого не требует извлечения микросхемы из устройства и допускает большое число циклов стирания данных (от десятков тысяч до миллиона). Стирание старой информации и запись новой занимают время порядка миллисекунд.

Программирование заряда в плавающем затворе используется и в технике EPROM, причем в этом случае возможно применение ЛИЗМОП-транзисторов с одним плавающим затвором при стирании заряда путем облучения кристалла ультрафиолетовыми лучами через специальное окошко в корпусе микросхемы. Стирание информации в памяти конфигурации типа EPROM является длительным процессом, занимающим десятки минут, и производится на специальном программаторе. Число циклов стирания существенно ограничено (сотни, тысячи). В последнее время схемотехника EEPROM быстро совершенствуется и все больше вытесняет схемотехнику EPROM, широко распространенную в предыдущих программируемых схемах.

Вариантом схемотехники EEPROM является так называемая Flash-память. Принцип работы элементов этой памяти не отличается от принципа работы описанных ЛИЗМОП-транзисторов, но новый технологический уровень их реализации и полученные вследствие этого улучшенные технико-экономические характеристики, как и блочное стирание данных, выделили Flash-память в отдельный класс, который считается вершиной достижений в области памяти с электрическим стиранием данных, хранимых в виде зарядов плавающих затворов.

В схемах со статической памятью конфигурации роль программируемого соединения играет транзисторный ключ. Такой

ключ, управляемый триггером памяти конфигурации, показан на рис. 7.5. Ключевой транзистор Т2 замыкает или размыкает участок ab в зависимости от состояния триггера, подключенного к затвору транзистора. При программировании сигналом с линии выборки включается транзистор Т1 и с линии записи/чтения подается сигнал установки или сброса триггера. В рабочем режиме транзистор Т1 заперт, а триггер сохраняет заданное ему состояние. Соответственно характеру памяти конфигурации (статическая триггерная) схемы такого типа называют SRAM-based.

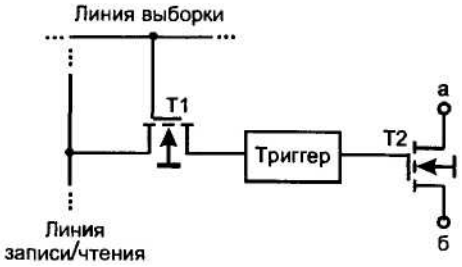


Рисунок 7.5. – Ключевой транзистор, управляемый триггером памяти конфигурации

Загрузка тех или иных данных в память конфигурации программирует микросхему. Процесс программирования может производиться неограниченное число раз и с высокой скоростью. При выключении питания конфигурация разрушается, поэтому после каждого включения питания требуется новая загрузка данных в память конфигурации. Такая загрузка может производиться из какой-либо энергонезависимой памяти за время порядка миллисекунд (в зависимости от объема файла конфигурации за единицы, десятки, сотни миллисекунд или даже больше). Триггеры памяти конфигурации распределены по всей площади кристалла и размещаются вблизи тех схем, которые они конфигурируют.

В современных микросхемах программируемой логики триггерная память конфигурации занимает важнейшее место.

### 7.3. Области применения микросхем с программируемой логикой

Возможности применения микросхем с программируемой логикой в системах обработки информации самых разных видов настолько обширны, что их рассмотрение в рамках одного подпараграфа не представляется возможным. Поэтому ограничимся лишь несколькими примерами.

1. Построение реконфигурируемых систем.

В аппаратуре различного назначения нередко встречаются ситуации, в которых те или иные блоки работают поочередно. Например, в системе передач сообщений с помощью помехоустойчивых кодов средства кодирования используются в процессе выдачи данных в канал связи, а средства декодирования — при их приеме. Поэтому не обязательно иметь два устройства (кодер и декодер), а можно иметь одну ИС ПЛ с двумя различными конфигурациями, хранимыми в энергонезависимой памяти и используемыми поочередно. Таким образом, одна *и та же аппаратура может решать несколько задач после соответствующей перестройки.* Технико-экономические выгоды такого варианта очевидны.

В настоящее время развивается концепция систем с динамической реконфигурацией (Run-Time Reconfiguration), применимая в системах с выполнением действий по шагам, последовательным во времени, когда в данное время требуется только одна определенная конфигурация ИС ПЛ. Этот режим работы сходен с рассмотренным выше, но в устройствах с динамическим реконфигурированием может потребоваться быстрая смена настроек. Обычное реконфигурирование с введением в микросхему последовательного потока битов или байтов занимает довольно большое время. Для динамически реконфигурируемых систем задача решается иначе. В системе уже хранится набор заранее загруженных настроек, быстро сменяющих друг друга соответственно требованиям реализуемого алгоритма.

Проблема построения систем на основе СБИС ПЛ с динамической реконфигурацией активно исследуется.

2. Использование БИС/СБИС ПЛ как окончательной продукции при изготовлении малых партий изделий. В этом случае

проявляются такие их достоинства как простота проектирования, скорость разработки и получение технических параметров, соответствующих возможностям микросхем высокого уровня интеграции. При этом продукция остается достаточно дешевой.

3. Использование микросхем программируемой логики для отладки прототипов при проектировании устройств и систем.

Системные компании широко используют репрограммируемые БИС/СБИС на стадиях отработки проектов. В проекты, реализованные на таких БИС/СБИС, легко вносить изменения, так как новый вариант получается путем простой коррекции кода конфигурации. Отладка репрограммируемого прототипа обеспечивает устранение ошибок в проекте, после отладки можно пользоваться ее результатами независимо от способа изготовления конечной продукции. При большом объеме выпуска реализацию проекта можно перенести на БИС/СБИС типа БМК (базовый матричный кристалл), а при очень большом — и на схемы заказного изготовления.

Следует отметить, что отладка прототипов проекта на репрограммируемых микросхемах не отменяет, а дополняет традиционные методы отладки устройств и систем. При такой отладке издавна пользовались изготовлением макетного прототипа и программными моделями. Изготовление макетного прототипа — сложная и дорогостоящая задача, но зато с его помощью можно вести тестирование и отладку с реальными сигналами и на высоких скоростях, наблюдая фактические возможности устройства или системы. Программное моделирование лишено этих достоинств, но проще и дешевле. Модели легко изменяются, и в них просто обеспечивается хорошая наблюдаемость процессов в объекте исследования.

Применение репрограммируемых микросхем в задачах логической эмуляции дает сочетание достоинств обоих классических подходов. Система из таких микросхем легко создается и изменяется, но в то же время может работать с реальными сигналами и частотами их изменения. Однако по затратам труда и времени создание системы из микросхем репрограммируемой логики все же сложнее, чем создание программной модели. Поэтому программные модели не исключаются с появлением репрограммируемых микросхем. С другой стороны, следует также

учитывать, что полные свойства окончательно изготовленной продукции эмуляция на репрограммируемых микросхемах при переходе в последующем на иные методы производства отобразить не может, так как временные характеристики зависят от конкретной трассировки схемы, которой еще нет на этапе отладки с помощью репрограммируемых микросхем. Таким образом, эмуляция проектов на репрограммируемых микросхемах не отменяет прежние методы разработки и тестирования схем, а лишь эффективно их дополняет.

### 7.4. Свойства интегральных схем программируемой логики, важные для их применения в составе систем

Ряд свойств микросхем программируемой логики не связан непосредственно с их логическим функционированием, но имеет важное значение при их использовании в системе. К таким (системным) свойствам относятся следующие.

1. Уровни питающих напряжений.

Имеются веские причины для перехода ко все меньшим напряжениям как для питания микросхем в рабочих режимах, так и для программирования таких элементов, как ЛИЗМОП-транзисторы в схемах с памятью конфигурации типов EPROM и EEPROM. Если сравнительно недавно типовым значением питающего напряжения было 5 В, то сейчас все больше используются схемы с напряжениями питания 3,3; 2,5; 1,8 и даже 1,6 В. Кроме того, исключаются требования повышенных напряжений для программирования, и все процессы в схемах обеспечиваются с помощью внутренних средств от единого источника внешнего питания.

2. Наличие режимов различного энергопотребления. При работе того или иного блока для поддержания его быстродействия требуются определенные затраты энергии (для быстрых переключений требуется быстрый перезаряд неизбежно существующих паразитных емкостей, т. е. нужны большие токи, обеспечивающие требуемые длительности переходных процессов). При отсутствии переключений в той или иной схеме можно резко снизить ее энергопотребление, так как для сохранения неизменного логического состояния достаточны микротоки. Кроме

двух указанных режимов можно создавать и ряд промежуточных, что широко используется в микросхемах программируемой логики. Активные режимы могут программироваться в разных вариантах (максимального быстродействия и номинального быстродействия с меньшей потребляемой мощностью). Пассивные режимы также часто имеют несколько подрежимов (покоя, когда схема не переключается, но готова к быстрому вхождению в рабочий режим, глубокого понижения мощности, когда потребление энергии чрезвычайно мало, но переход в активный режим занимает относительно большое время, и т. д.).

Интересно отметить, что с помощью несложных логических элементов схемы могут автоматически переходить из одного режима в другой, выявляя факт изменения информационных сигналов и реагируя на него увеличением рабочих токов.

В число программируемых величин может входить так называемый Turbo bit, задающий один из двух режимов, различающихся по соотношению скорость/мощность.

3. Наличие или отсутствие в микросхеме средств поддержки интерфейса JTAG. Первоначальные версии этого интерфейса обеспечивали тестирование систем методом периферийного (граничного) сканирования, более поздние расширенные версии, ориентированные на микросхемы программируемой логики, позволяют через интерфейс JTAG конфигурировать схемы с триггерной памятью, для чего требуется лишь загрузка в память конфигурации загрузочного файла.

4. Свойство программируемости в системе. Схемы, обладающие этим свойством, называются In-System Programmable (ISP). Свойство ISP могут иметь микросхемы с триггерной памятью конфигурации (заметим, что термин «программируемость» появился в этом абзаце в связи с его наличием в английском термине ISP, по существу же он означает то же самое, что и термин «конфигурируемость»). Поскольку триггерная память теряет свое содержимое при выключении питания, схемы типа SRAM-based программируются при каждом включении питания. Свойство ISP означает, что реконфигурацию можно производить без изъятия микросхемы из системы, перезагружая файл конфигурации, что возможно и во время функционирования системы. Такая возможность позволяет строить системы с многофункцио-

нальным использованием одних и тех же программируемых микросхем в качестве разных блоков системы, если эти блоки не используются одновременно, а также и другие варианты адаптивных систем.

5. Наличие средств защиты от считывания данных конфигурации. Такие средства имеют разную степень защиты. В простейшем случае предусматривается бит защиты или несколько таких битов.

6. Программирование крутизны фронтов. Известно, что одной из острых проблем при реализации цифровых устройств и систем является борьба с помехами, в том числе создаваемыми самими схемами этих устройств и систем. С целью снижения уровня возникающих импульсных помех для мощных буферов, прежде всего тех, которые формируют выходные сигналы кристалла, вводится программирование крутизны фронтов. Бит SLC (Slow Rate Control) задает режимы крутых или пологих фронтов. Рекомендуется везде, где допустимо, устанавливать режим пологих фронтов, создающий гораздо меньшие помехи. В критичных для быстродействия цепях с целью повышения производительности системы используются режимы крутых фронтов (т. е. быстрых переключений).

Завершая обсуждение общих вопросов, связанных с особенностями микросхем программируемой логики, следует подчеркнуть, что они выпускаются промышленностью как полностью готовые. При использовании таких схем потребитель не обращается к изготовителю для проведения каких-либо завершающих разработку операций и выполняет программирование микросхем самостоятельно. Это дает основание отнести ИС ПЛ к стандартной продукции электронной промышленности, что ведет к известным положительным последствиям — массовости производства и снижению стоимости микросхем.

# 8. МЕТОДИКА И СРЕДСТВА ПРОЕКТИРОВАНИЯ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ

## 8.1. Общее описание процесса проектирования

*Процесс проектирования* — разработка технической документации, позволяющей изготовить устройство с заданным функционированием, с заданными свойствами и в заданных условиях.

Основа *стратегии проектирования* – функциональная декомпозиция. Для системы в целом и ее блоков используется концепция «черного ящика». Для «черного ящика» разрабатывается функциональная спецификация, включающая внешнее описание блока (входы и выходы) и внутреннее описание – функцию или алгоритм работы:

$F = \Phi(X, t)$, где $X$ – вектор входных величин, $F$ – вектор выходных величин, $t$ – время. При декомпозиции функция $\Phi$ разбивается на более простые функции $\Phi_1 - \Phi_K$, между которыми должны быть установлены определенные связи, соответствующие принятому алгоритму реализации функции $\Phi$. Переход от функции к структуре называется синтезом.

Синтез, как правило, неоднозначен. Выбор наилучшего варианта осуществляется по результатам анализа, когда проверяется правильность работы и некоторые показатели, характеризующие устройство.

Декомпозиция функций блоков выполняется до тех пор, пока не получатся типовые функции, каждая из которых может быть реализована элементами выбранного уровня иерархии.

*Процесс проектирования* — многоуровневый, многошаговый и итерационный, с возвратами назад и пересмотром ранее принятых решений.

Последовательная декомпозиция проекта на отдельные фрагменты (с определением функций каждого фрагмента и его интерфейса) не зависит от иерархического уровня проектирования и характерна для разработки широкого класса цифровых устройств, начиная от устройства целиком и кончая проектированием отдельных БИС/СБИС. Такая методология проектирования отображает процесс проектирования «сверху-вниз»: от

технического задания до электрических схем, файлов прошивки ПЗУ и конфигурации программируемых приборов, а также конструкции устройства в целом.

Другая последовательность, соответствующая методологии «снизу-вверх», предусматривает объединение простейших модулей в более сложную структуру до тех пор, пока, в конце концов, не будет создан конечный проект. Исходные модули — это решения, созданные проектировщиком на более ранних этапах работы или в ходе работ над другими проектами или доступные проектировщику и входящие в состав имеющихся библиотек САПР.

Современным условиям проектирования, когда создаются сложные проекты с привлечением большого числа разработчиков, больше соответствует применение стратегии «сверху-вниз».

Необходимо отметить, что приведенное выше наглядное описание процесса проектирования относится к каждому уровню проектирования. При этом декомпозиция заканчивается при получении типовых функций, соответствующих выбранному уровню иерархии. Так, на верхнем уровне (при многоплатной реализации) декомпозиция заканчивается при представлении проекта в виде отдельных плат, на следующем уровне — в виде отдельной платы (типового элемента замены), еще ниже декомпозиция осуществляется до реализации функций при помощи той или иной микросхемы. А при ориентации на программируемые (разрабатываемые) пользователем микросхемы процедура декомпозиции осуществляется уже для этой микросхемы в соответствии с составом функциональных библиотек программируемых БИС/СБИС.

С учетом возможностей современных систем автоматизации проектирования (САПР) проектирование может считаться законченным после верификации проекта в целом, когда завершена отладка готового изделия.

Различие теоретической базы и понятийного аппарата, используемых на разных стадиях проектирования, приводит к тому, что традиционным является разбиение процесса проектирования, как цифровых устройств, так и БИС/СБИС на следующие этапы:

1. Системное проектирование. На этом этапе определяется архитектура будущей системы, состав компонентов и основные характеристики системы при таком её построении;

2. Структурно-алгоритмическое проектирование. Здесь определяются алгоритмы функционирования аппаратных и программных компонентов системы;

3. Функционально-логическое проектирование. На этом этапе разрабатываются функциональные и принципиальные электрические схемы, программы, подготавливаются тестовые и контрольные данные;

4. Конструкторско-технологическое проектирование. На этом этапе производится привязка элементов проекта к конструктивным элементам.

Широкое использование САПР на всех этапах проектирования приводит к тому, что современные подходы к разбиению процесса проектирования связывают с различием как технических средств (инструментария), привлекаемых для создания проекта, так и технических средств, используемых в качестве компонентов проекта и технологических особенностей реализации конечного продукта. Хотя общая методология процесса проектирования не зависит от варианта разбиения процесса проектирования на отдельные уровни, содержание, а также методы и средства проектирования для различных уровней оказываются очень специфичными и существенно зависят как от типа применяемой элементной базы, так и от способа реализации (изготовления) конечного продукта.

## 8.2. Классификация методик проектирования электронных схем

Существует множество методик проектирования. Из всего многообразия можно выделить следующие факторы, влияющие на специфику проектирования:

1. *Тип обрабатываемой информации.* С ним связаны *методы и способы* ее *обработки.* Проект или его отдельные фрагменты могут включать аналоговые, аналого-цифровые и/или цифро-аналоговые элементы, строиться на основе дискретных (цифровых) компонентов или опираться на встроенные микропроцес-

сорные средства. Отсюда следует многообразие вариантов проектирования, эти варианты в современных технологиях часто называют потоком проектирования (Design Flow). Проектирование при этом может быть цифровым, аналого-цифровым, смешанным цифровым и программным, а также проектированием с ориентацией на синтезируемые цифровые или аналоговые схемы. Возможны и другие комбинации этих вариантов.

2. *Выбор технической базы* для реализации проекта, а также *технологического способа реализации* самого проекта. Как правило, одно и то же электронное изделие может быть реализовано различными способами. Здесь должен быть дан ответ на вопрос — будет ли проект построен на стандартных микросхемах или будут использоваться те или иные специализированные ИС и/или комбинация различных типов ИС.

На рис. 8.1 приведена классификация ИС по признаку способа изготовления. Приведем краткие пояснения к ней.

К *стандартным микросхемам* отнесены схемы малой и средней степени интеграции (МИС и СИС). Эти микросхемы производятся массовыми тиражами и реализуют стандартные элементы и узлы, функционирование которых никак не определяется конкретными потребителями. К стандартным схемам высокого уровня интеграции (БИС и СБИС) относятся цифровые схемы: микропроцессоры, микроконтроллеры и запоминающие устройства (ЗУ), разнообразные периферийные схемы для МП и МК, включая и аналого-цифровые схемы: аналого-цифровые преобразователи (АЦП), цифроаналоговые преобразователи (ЦАП). Общее свойство этих схем то, что они остаются неизменными после изготовления независимо от устройств и систем, в которых они используются.

К *специализированным* ИС (СпИС) относятся все, структура которых в отличие от структур стандартных ИС массового производства каким-либо способом приспосабливается к конкретным требованиям того или иного проекта. В английской терминологии СпИС именуются ASICs (Application Specific Integrated Circuits). Среди СпИС различают классы *полузаказных* и *заказных*. Разновидностями заказных микросхем являются полностью заказные и спроектированные методом «на стандартных ячейках».

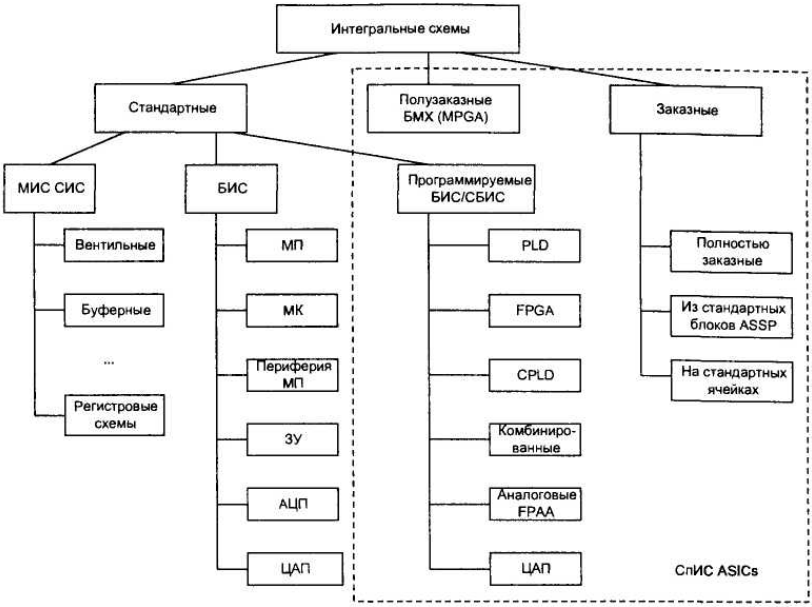


Рисунок 8.1. – Классификация ИС по признаку способа изготовления

*Полностью заказные схемы* целиком проектируются по требованиям конкретного заказчика. Проектировщик имеет полную свободу действий, определяя схему по своему усмотрению вплоть до уровня схемных компонентов (отдельных транзисторов и т. п.). Для изготовления схемы требуется разработка всего комплекта фотошаблонов, верификация и отладка всех схемных фрагментов. Такие схемы очень дороги и имеют длительные циклы проектирования.

*Схемы на стандартных ячейках* отличаются от полностью заказных тем, что их фрагменты берутся из заранее разработанной библиотеки схемных решений. Такие фрагменты уже хорошо отработаны, стоимость и длительность проектирования при этом снижаются. Для производства схем тоже требуется изготовление полного комплекта фотошаблонов, но разработка их облегчена. Потери по сравнению с полностью заказными ИС состоят в том, что проектировщик имеет меньше свободы в по-

строении схемы, т. е. результаты ее оптимизации по таким критериям как площадь кристалла, быстродействие и т. д. менее эффективны. Наивысших технических параметров добиваются от полностью заказных схем, однако метод стандартных ячеек популярен, так как при небольших потерях в технических характеристиках с его помощью можно заметно упростить проектирование схемы. Полностью заказные схемы разрабатываются за время, превышающее время разработки методом стандартных ячеек приблизительно в два раза.

На промежуточном месте находятся схемы, проектируемые по технологии «стандартная продукция для фиксированных приложений» (Application Specific Standard Products – ASSP). Основой технологии является использование в качестве строительных блоков заранее разработанной библиотеки схемных решений системного уровня, таких как микроконтроллеры, память, блоки стандартных интерфейсов и блоки специфической системной логики. Для экономической целесообразности реализации подобных проектов для фирмы «Atmel», например, требуется тиражность продукции не менее 100 тыс. кристаллов в заказе.

К *полузаказным* схемам относятся базовые матричные кристаллы БМК (в английской терминологии MPGA- Mask Programmable Gate Arrays). В этом случае имеется стандартный полуфабрикат, который доводится до готового изделия с помощью индивидуальных межсоединений. Реализация требует изготовления лишь малого числа фотошаблонов. Стоимость и длительность проектирования в данном случае по сравнению с полностью заказными схемами сокращаются в 3–4 раза. Однако результат проектирования еще дальше от оптимального, поскольку в матричных БИС (МАБИС) менее рационально используется площадь кристалла (на кристалле остаются неиспользованные элементы и т. п.), не минимальны длины связей и не максимально быстродействие.

Сходство методов проектирования на БМК и стандартных ячейках состоит в использовании библиотек функциональных элементов. Различие в том, что для схем, проектируемых по методу стандартных ячеек, библиотечный набор элементов имеет более выраженную топологическую свободу. Например, стандартизируется только высота ячеек, а их длины могут быть раз-

личными. При проектировании сначала из набора библиотечных элементов подбираются необходимые функциональные блоки, а затем решаются задачи их размещения и трассировки.

Методика, а соответственно и САПР для проектирования по методу стандартных ячеек более сложны, чем для проектирования на основе БМК, которому свойственны более жесткие топологические ограничения. Ограничения вводятся и для метода стандартных ячеек (постоянство высоты ячеек, предопределенность геометрических размеров и положения шин питания, тактирования и др.), но по мере применения более мощных САПР ограничения ослабляются.

Длительность изготовления БИС/СБИС методом стандартных ячеек превышает этот же показатель для МАБИС на основе БМК в 1,3–1,8 раз.

Особое место в классификации имеют БИС/СБИС с программируемой структурой. С одной стороны, они относятся к СпИС, так как в итоге счете приспосабливаются к требованиям конкретного проекта. В то же время этот процесс (конфигурация схемы) не затрагивает изготовителя, для которого схемы являются стандартным продуктом со всеми вытекающими из этого выгодами.

### 8.3. Области применения специализированных интегральных схем

Все типы СпИС имеют свои области применения. Каждому типу свойственно определенное соотношение таких параметров как сложность (достижимый уровень интеграции), быстродействие, стоимость. На выбор типа СпИС для реализации проекта влияет совокупность свойств. Это можно пояснить с позиций экономики, обратившись к формуле стоимости $С_{ис}$ ИС, которая производится с использованием освоенного техпроцесса:

$$C_{ис} = C_{изг} + C_{пр}/N,$$

где $C_{из\,г}$ — стоимость изготовления ИС (стоимость кристалла и других материалов, стоимость технологических операций по изготовлению ИС, контрольных испытаний); затраты на изготовление относятся к каждой ИС, т. е. повторяются столько раз, сколько ИС будет произведено; $C_{пр}$ — стоимость проектирова-

ния ИС, т. е. однократные затраты для данного типа ИС; $N$ — объем производства, т. е. число ИС, которое будет произведено.

Стоимость проектирования БИС/СБИС велика и может достигать сотен миллионов долларов. Для дорогостоящих вариантов проектирования БИС/СБИС производство становится рентабельным только при большом объеме их продаж.

Затраты $C_{пр}$ и $C_{изг}$ находятся во взаимосвязи. Рост затрат на проектирование, как правило, ведет к снижению $C_{изг}$, поскольку, чем совершеннее проект, тем рациональнее используется площадь кристалла и другие его ресурсы. Отсюда видно, что выигрыш по экономичности могут получать те или иные типы СпИС в зависимости от тиражности их производства N и сложности.

Применительно к микросхемам программируемой логики справедливы следующие положения. Простые устройства со сложностью в сотни эквивалентных вентилей целесообразно реализовывать на PLD (PAL, GAL, PLA). При росте сложности проекта естественен переход к FPGA и CPLD, если тиражность ИС сравнительно невелика. Рост тиражности (приблизительно свыше десятков тысяч) ведет к преимуществам реализаций на БМК, так как стоимость изготовления небольшого числа шаблонов для создания межсоединений разложится на большое число микросхем, а стоимость изготовления каждой ИС уменьшится благодаря исключению из схемы схем программируемых связей и средств их программирования.

При еще большей тиражности выгодным оказывается метод стандартных ячеек, позволяющий дополнительно улучшить параметры схемы, плотнее разместить ее элементы на кристалле, т. е. уменьшить $C_{изг}$ и улучшить быстродействие. При этом слагаемое $C_{пр}/N$ в формуле стоимости ИС не окажется слишком большим благодаря большой величине $N$, хотя необходимость проектировать весь комплект шаблонов для технологических процессов приводит к большим затратам $C_{пр}$.

Полностью заказное проектирование для СпИС не характерно. Оно стоит настолько дорого, что применяется практически только для создания стандартных БИС/СБИС массового производства. Например, проектирование первого 32-разрядного микропроцессора обошлось в свое время в 140 млн. долларов, а ЗУ емкостью в 1 Мбит — в 395 млн. долларов.

# 9. КОМПЛЕКСНОЕ ПРОЕКТИРОВАНИЕ ТИПОВОЙ КОНФИГУРАЦИИ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ

## 9.1. Типовые конфигурации микропроцессорной системы

В зависимости от типа МПС, существует несколько методик проектирования/отладки микропроцессорных систем. В соответствии с названием микроконтроллерные системы ориентированы на выполнение задач управления определенными устройствами или их комплексами. Микропроцессорные системы можно условно разделить на два основных класса: универсальные, которые используются для решения широкого круга задач обработки информации, и управляющие, которые специализируются на решении задач управления процессами и объектами. Типичными примерами универсальных микропроцессорных систем являются персональные компьютеры и рабочие станции, которые применяются в самых различных сферах деятельности.

Микропроцессорные системы управления имеют много общего с МК. Они также содержат различные устройства, расширяющие возможности процессора для реализации сложных алгоритмов управления. При этом периферийные устройства, многие из которых располагаются на кристалле микроконтроллера, в микропроцессорных системах реализуются с помощью дополнительных микросхем, что повышает их стоимость и снижает надежность. Разработка интегрированных микропроцессоров, имеющих в своем составе ряд периферийных устройств, и сложнофункциональных микроконтроллеров, содержащих высокопроизводительное 32-разрядное процессорное ядро, приводит к размыванию границы применения управляющих микропроцессорных и микроконтроллерных систем, постепенному стиранию функциональных и структурных различий между ними.

Основной особенностью микроконтроллеров является наличие в их составе ПЗУ (ППЗУ, РППЗУ, ЭСППЗУ, флэш-памяти), в которое записывается резидентная рабочая программа системы. Разработка, отладка и запись в ПЗУ этой программы является важнейшей стадией проектирования микроконтроллерных систем. Записанная в ПЗУ рабочая программа становится составной частью системы, последующее изменение или коррек-

ция которой обычно нежелательны или невозможны. При использовании внутреннего ПЗУ возможности внешнего контроля работы микроконтроллера в процессе отладки очень ограничены. Поэтому комплексная отладка программного и аппаратного обеспечения МК систем является достаточно сложной процедурой, требующей использования специализированных методов и средств контроля. Данный этап проектирования является также наиболее ответственным, так как невыявленная ошибка может привести к весьма дорогостоящим последствиям. Особенностью МПС для ряда областей применения является необходимость строгого соблюдения определенных норм времени на выполнение программы или ее отдельных модулей.

В МПС выполняемые модули рабочей программы загружаются в ОЗУ. Благодаря этому имеется возможность оперативной коррекции рабочей программы в случае необходимости. В процессе отладки проектировщик имеет доступ к общей шине, что облегчает текущий контроль за работой системы. Однако наличие в большинстве современных микропроцессоров внутренней кэш-памяти ограничивает возможности внешнего контроля за ходом выполнения программы. Особенно возрастают сложности отладки при использовании микропроцессоров с суперскалярной структурой, в которых несколько команд выполняются одновременно и естественная очередность их выполнения может не соблюдаться. Хотя при проектировании микропроцессорных систем выполняются практически те же этапы, что и для микроконтроллерных систем, однако используемая процедура разработки и средства отладки во многих случаях существенно различаются. Рассмотрим основные этапы проектирования/ отладки этих систем и особенности их реализации.

### 9.2. Основные этапы процедуры проектирования

Процесс проектирования микропроцессорной системы – это комплекс мероприятий, состоящий из большого числа этапов. Хотя в каждом конкретном случае проектирование осуществляется по своей траектории, можно сформулировать основные этапы процедуры проектирования-отладки микропроцессорных и микроконтроллерных систем (рис. 9.1).

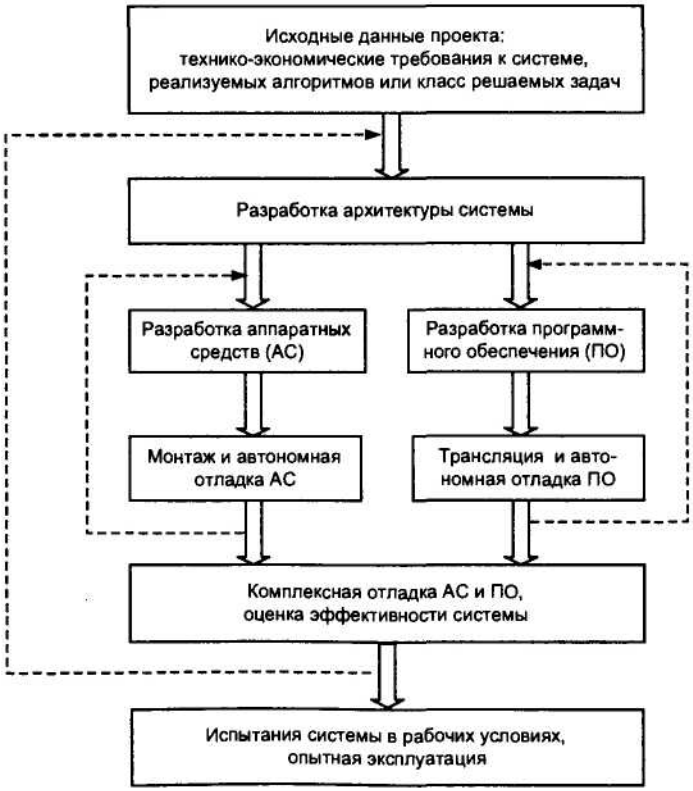


Рисунок 9.1. – Типовая процедура проектирования микропроцессорной системы

Рассмотрим все блоки и раскроем их суть.

Исходные данные для проектирования содержат требования к основным технико-экономическим показателям: производительности, энергопотреблению, стоимости, надежности, конструктивным и другим параметрам. Кроме того, для управляющих систем должны быть определены реализуемые алгоритмы управления, для универсальных систем — классы выполняемых задач.

Разработка архитектуры системы подразумевает определение оптимального состава аппаратных и программных средств для решения поставленных задач. При этом разработчик решает, какие функции системы будут реализованы аппаратными сред-

ствами (АС), а какие — программным обеспечением (ПО). Определяется номенклатура АС: выбираются тип микропроцессора или микроконтроллера, объем и тип памяти, номенклатура периферийных устройств, протоколы обмена информацией и состав требуемых сигналов управления системой. Определяется также состав ПО: наличие операционной системы, ее тип и характеристики, номенклатура необходимых программных модулей, характер их взаимодействия, используемый язык программирования. Результатом выполнения этого этапа являются частные технические задания на проектирование АС и ПО.

Этап разработки АС может быть выполнен традиционными методами, с помощью которых проектируется и моделируется электрическая схема, разрабатывается печатная плата или комплект плат, после чего выполняются монтаж и отладка системы. Однако во многих случаях можно обеспечить сокращение сроков и повышение качества разработки АС путем использования «полуфабрикатов» или готовых изделий, выпускаемых рядом производителей.

Существует достаточно большая номенклатура таких изделий, которые носят названия оценочных или целевых плат (evaluation board, target board), оценочных наборов или систем (evaluation kit, evaluation system), одноплатных компьютеров или контроллеров (SBC – single-board computer, single-board controller). В их состав входит базовый микропроцессор или микроконтроллер, память (ОЗУ, флэш-память, служебное ПЗУ), ряд периферийных и вспомогательных схем. Обычно такие платы имеют разъем для подключения к персональному компьютеру, с помощью которого производится комплексная отладка системы.

Если состав средств, имеющихся на плате развития, достаточен для реализации проектируемой системы, то ее разработка сводится к созданию ПО и выполнению комплексной отладки системы. Если имеющихся средств недостаточно, то они проектируются и размещаются на дополнительной плате, подключаемой к разъему на плате развития непосредственно или с помощью кабеля. Так реализуется прототип проектируемой системы, на котором можно выполнить комплексную отладку программных и аппаратных средств, а в ряде случаев и провести проверку их функционирования в рабочих условиях. После этого нетруд-

но разработать рабочий вариант системы, объединив на одной плате используемые модули прототипной системы. Прототипная система может использоваться в качестве рабочей (целевой), если ее параметры и конструктивное оформление удовлетворяют требованиям технического задания. В этом случае достигается сокращение сроков и стоимости проектирования системы.

Особенно следует отметить перспективность использования при разработке АС мезонинной технологии, которая унифицирует размеры и интерфейс базовой платы-носителя и размещаемых над ней небольших плат — мезонинов (типичный размер 45x99 мм). Одна плата-носитель несет от 2 до 12 мезонинов. Каждый мезонин соединяется с носителем двумя разъемами.

На этапе автономной отладки АС основными орудиями разработчика являются традиционные измерительные приборы – осциллографы, мультиметры, пробники и другие, а также логические анализаторы, которые обладают широкими возможностями контроля состояния различных узлов системы в заданные моменты времени. Весьма эффективным является использование на этом этапе средств тестирования по стандарту JTAG, которые имеются в составе многих современных моделей микропроцессоров и микроконтроллеров. С помощью размещенного на кристалле тест-порта TAP и специальных выводов TDI, TDO, TCK, TMS, TRST# обеспечивается возможность подачи необходимых входных воздействий и считывания выходной реакции, запуск-останов процессора, изменение режима его работы. Вводом специальной команды можно установить выводы микропроцессора или микроконтроллера в отключенное состояние, чтобы отдельно протестировать другие устройства системы.

### 9.3. Средства проектирования и методы автономной отладки аппаратных средств микропроцессорной системы

Мировая промышленность выпускает широкую номенклатуру микропроцессоров и микроконтроллеров, что позволяет удовлетворить запросы подавляющего большинства потребителей. Однако выбор типа микропроцессора или микроконтроллера является только первым шагом на пути создания системы, соответствующей требованиям заказчика. Реализация такой сис-

темы является сложным и трудоемким процессом, выполнение которого на современном уровне невозможно без использования комплекса специализированных программных и аппаратных средств, помогающих разработчику на различных этапах проектирования, программирования и отладки. Поэтому при оценке и выборе типа микропроцессора или микроконтроллера для конкретного применения необходимо учитывать не только его технико-экономические характеристики, но и уровень развития программно-аппаратных средств, предлагаемых для использования в процессе проектирования-отладки систем на его основе.

Если микропроцессорная система строится на основе использования типовых и стандартных элементов, то моделирование работы аппаратуры для проверки правильности её работы (ввиду невозможности влияния на содержимое этих элементов), как правило, не выполняется. Однако использование в учебных целях для уточнения представлений разработчика о функционировании таких элементов в отдельности или в каких-либо сочетаниях, естественно, допустимо, хотя обычно это совмещается с разработкой программного обеспечения, опирающегося на возможности проектируемых аппаратных средств. С другой стороны, обучение может производиться на достаточно дешевых аппаратных средствах типа Starter Kit.

На этапе автономной отладки АС основными орудиями разработчика являются традиционные измерительные приборы — осциллографы, мультиметры, пробники и другие, а также логические анализаторы, которые обладают широкими возможностями контроля состояния различных узлов системы в заданные моменты времени. Весьма эффективным является использование на этом этапе средств тестирования по стандарту JTAG, которые имеются в составе многих современных моделей микропроцессоров и микроконтроллеров. С помощью размещенного на кристалле тест-порта TAP и специальных выводов TDI, TDO, TCK, TMS, TRST# обеспечивается возможность подачи необходимых входных воздействий и считывания выходной реакции, запуск/останов процессора, изменение режима его работы. Вводом специальной команды можно установить выводы микропроцессора или микроконтроллера в отключенное состояние, чтобы отдельно протестировать другие устройства системы.

## 9.4. Средства разработки и отладки программного обеспечения

### 9.4.1. Обзор средств разработки и отладки программного обеспечения

В настоящее время для разработки программного обеспечения универсальных микропроцессорных систем существует достаточно большой набор языков программирования высокого уровня, для которых имеются соответствующие компиляторы. Чаще всего используются языки C, C++, FORTRAN, Pascal. Для решения ряда задач применяются языки поддержки искусственного интеллекта Ada, Modula-2 и некоторые другие. При программировании управляющих систем чаще всего используются машинно-ориентированный язык Ассемблера или языки C/C++. Язык Ассемблера применяется в случаях, когда имеются жесткие ограничения на объем требуемой памяти или на время выполнения программных модулей. Такие случаи являются достаточно типичными при решении задач управления, поэтому Ассемблеры являются одним из основных средств создания программного обеспечения для микроконтроллерных систем. В тех случаях, когда указанные ограничения не очень жесткие, для создания программного обеспечения используются языки высокого уровня (обычно C/C++).

Автономная отладка программного обеспечения выполняется с помощью симулятора — программной модели используемого микропроцессора или микроконтроллера. На этом этапе разработчики используют широкий набор средств программирования — компиляторы, ассемблеры, дисассемблеры, отладчики, редакторы связей и другие, без которых практически невозможно создание работоспособного программного обеспечения в течение ограниченных сроков выполнения проекта.

Как отмечалось выше, комплексная отладка аппаратных средств и программного обеспечения является наиболее сложным и ответственным этапом создания системы. На этом этапе разработчик использует весь набор программных и аппаратных средств, применяющихся для автономной отладки аппаратных средств и программного обеспечения, а также ряд специальных

средств комплексной отладки. К числу таких средств относятся схемные эмуляторы – специализированные устройства, включаемые вместо микропроцессора или микроконтроллера прототипной системы и обеспечивающие возможность контроля ее работы с помощью персонального компьютера, связанного со схемным эмулятором. Схемные эмуляторы являются наиболее эффективным средством комплексной отладки систем.

Одним из эффективных средств комплексной отладки МК систем являются эмуляторы ПЗУ. Оно включается вместо ПЗУ прототипной системы и работает под управлением подключенного к нему персонального компьютера. Так обеспечивается текущий контроль за выполнением программы и ее оперативная коррекция, что значительно упрощает процесс отладки.

Для микроконтроллерных систем заключительной процедурой комплексной отладки является запись в ПЗУ загрузочных модулей отлаженной программы и завершающее испытание ее работоспособности. Запись программы в ПЗУ осуществляется с помощью специальных программаторов.

Для универсальных микропроцессорных систем после комплексной отладки производится оценка их производительности путем прогона специального набора тестовых программ.

После выполнения указанных этапов отлаженная прототипная система может быть испытана в рабочих условиях с подключением полного набора реальных периферийных устройств и объектов управления. В процессе опытной эксплуатации выявляются ошибки, не обнаруженные на этапе отладки, определяется реакция системы на возможные непредвиденные ситуации.

Как показывает данное описание процесса разработки, при создании современных микропроцессорных и микроконтроллерных систем используется комплекс программно-аппаратных средств, которые помогают качественно и в ограниченные сроки выполнить их проектирование и отладку.

### 9.4.2. Отладчики и симуляторы

Как уже было сказано выше, *симуляторы – это* программно-логические модели микропроцессоров и микроконтроллеров, использующиеся при отладке программ. Они редко поставляют-

ся в виде отдельных средств поддержки программирования, обычно входят в состав отладчиков.

*Отладчики* являются основным инструментом разработчика программного обеспечения, без которого практически невозможно получить работоспособные объектные модули рабочей программы. Отладчик реализует различные режимы выполнения транслированной программы — пошаговый или с остановами в контрольных точках, позволяет производить просмотр и коррекцию содержимого регистров и ячеек памяти, обеспечивает в точке останова контроль выполнения предыдущих шагов программы (просмотр трассы), дисассемблирование команд. Отладчик воспринимает программу на уровне исходного кода или в символическом виде, с использованием введенных разработчиком имен и меток. Символические отладчики являются наиболее удобным средством отладки, так как они представляют и воспринимают информацию в наиболее наглядной и удобной для программиста форме.

Помимо симулятора, отладчики содержат обычно компоновщик-загрузчик объектного кода, библиотеки стандартных функций (вычисление специальных и тригонометрических функций, обработка чисел с плавающей точкой и другие). Для визуализации состояния системы на экране монитора современные отладчики используют многооконный графический интерфейс. Многие отладчики могут работать не только с симуляторами, но и реализуют интерфейс со схемными эмуляторами, т. е. с реальными микропроцессорами или микроконтроллерами в процессе комплексной отладки системы.

### 9.4.3. Прототипные платы

Прототипные платы – это многочисленный класс средств проектирования микропроцессорных и микроконтроллерных систем. Условно их можно разделить на следующие типы:

– системные комплекты (evaluation kit) – набор размещенных на плате аппаратных средств, достаточных для реализации несложных систем;

– отладочные платы и системы (evaluation board, system) – размещенные на плате программно-аппаратные комплексы,

обеспечивающие моделирование и отладку систем различного назначения на базе определенных моделей микропроцессоров или микроконтроллеров;

– целевые платы (target board) — программно-аппаратные комплексы, ориентированные на использование после отладки в качестве прототипной системы;

– одноплатные компьютеры и контроллеры (single-board computer, controller) — конструктивные комплексы, предназначенные для использования в качестве базовых модулей при реализации целевых систем промышленного применения.

Эти средства могут использоваться для следующих целей:

– изучение функционирования определенных моделей микропроцессоров и микроконтроллеров, получение навыков их практического применения;

– тестирование и отладка программного обеспечения систем на реальных образцах микропроцессоров (микроконтроллеров);

– комплексная отладка макета системы, используемого затем в качестве образца для реализации прототипной системы;

– сборка и отладка прототипной или целевой системы, в состав которой входят платы развития в качестве базовых модулей.

Практически все типы плат развития содержат в своем составе порты для подключения управляющего персонального компьютера. Чаще всего для этой цели используется последовательный обмен по стандарту RS-232. Ряд типов отладочных и целевых плат имеют также отдельное поле для макетирования пользователем дополнительных устройств с помощью проводного монтажа.

Ввиду большого разнообразия областей и способов применения номенклатура выпускаемых плат развития очень широка и четкие границы между их типами отсутствуют. Во многих случаях отладочные платы могут использоваться в качестве целевых, а одноплатные компьютеры часто служат средствами отладки прототипных систем. В данном разделе приводится краткое описание отдельных типичных представителей этого класса средств проектирования/отладки.

Рассмотрим подробно одного представителя этого класса.

Отладочные платы, или как принято их называть в зарубежной литературе — оценочные платы (Evaluation Boards), являются своеобразными конструкторами для макетирования прикладных систем. В последнее время, при выпуске новой модели кристалла микроконтроллера, фирма-производитель обязательно выпускает и соответствующую плату развития. Обычно это печатная плата с установленным на ней микроконтроллером, плюс вся необходимая ему стандартная обвязка. На этой плате также устанавливают схемы связи с внешним компьютером. Как правило, там же имеется свободное поле для монтажа прикладных схем пользователя. Иногда имеется уже готовая разводка для установки дополнительных устройств, рекомендуемых фирмой. Например, ПЗУ, ОЗУ, ЖКИ-дисплей, клавиатура, АЦП и др. Кроме учебных или макетных целей, такие доработанные пользователем платы стало выгодно (экономия времени) использовать в качестве одноплатных контроллеров, встраиваемых в мало серийную продукцию (5..20 шт.).

Для большего удобства, платы развития комплектуются еще и простейшим средством отладки на базе монитора отладки. Однако здесь проявились два разных подхода: один используется для микроконтроллеров, имеющих внешнюю шину, а второй — для микроконтроллеров, не имеющих внешней шины.

В первом случае отладочный монитор поставляется фирмой в виде микросхемы ПЗУ, которая вставляется в специальную розетку на плате развития. Плата также имеет ОЗУ для программ пользователя и канал связи с внешним компьютером или терминалом. Примером здесь может служить плата развития фирмы Intel для микроконтроллера I8051.

Во втором случае, плата развития имеет встроенные схемы программирования внутреннего ПЗУ микроконтроллера, которые управляются от внешнего компьютера. В этом случае, программа монитора просто заносится в ПЗУ микроконтроллера совместно с прикладными кодами пользователя. Прикладная программа при этом специально должна быть подготовлена: в нужные ее места вставляют вызовы отладочных подпрограмм монитора. Затем осуществляется пробный прогон. Чтобы внести в программу исправления, пользователю надо стереть ПЗУ и произвести повторную запись. Готовую прикладную программу

получают из отлаженной путем удаления всех вызовов мониторных функций и самого монитора отладки.

Важно отметить, что, плюс к монитору, иногда платы развития комплектуются и программами отладки, которые запускаются на внешнем компьютере в связке с монитором. Эти программы в последнее время заметно усложнились и зачастую имеют высокопрофессиональный набор отладочных функций, например, отладчик-симулятор или различные элементы, присущие в чистом виде интегрированным средам разработки. В состав поставляемых комплектов могут входить и программы прикладного характера, наиболее часто встречающиеся на практике.

Возможности по отладке, предоставляемые комплектом «плата развития + монитор», безусловно, не столь универсальны, как возможности внутрисхемного эмулятора, да и некоторая часть ресурсов микропроцессора в процессе отладки отбирается для работы монитора. Тем не менее, наличие законченного набора готовых программно-аппаратных средств, позволяющих без потери времени приступить к монтажу и отладке прикладной системы, во многих случаях является решающим фактором. Особенно если учесть, что стоимость такого комплекта несколько меньше, чем стоимость более универсального эмулятора.

### 9.4.4. Отладочные мониторы

Отладочный монитор – это специальная программа, загружаемая в память отлаживаемой системы. Она вынуждает процессор пользователя производить, кроме прикладной задачи, еще и отладочные функции:

– загрузку прикладных кодов пользователя в свободную от монитора память;

– установку точек останова;

– пуск и останов программы в реальном времени;

– проход программы пользователя по шагам (часть функций трассировщика);

– просмотр, редактирование содержимого памяти и управляющих регистров.

Программа монитора обязательно должна работать в связке с внешним компьютером или пассивным терминалом, на кото-

рых и происходит визуализация и управление процессом отладки. Повторим, что отладочные мониторы используют тот процессор, который уже стоит на плате пользователя.

Достоинством этого подхода являются очень малые затраты при сохранении возможности вести отладку в реальном времени.

Главным недостатком является отвлечение ресурсов микроконтроллера на отладочные и связные процедуры, например: монитор занимает некоторый объем памяти, прерывания, последовательный канал. Объем отвлекаемых ресурсов зависит от искусства разработчика монитора. В последнее время появились изделия, которые практически не занимают аппаратных ресурсов процессора, о них рассказано в разделе «Эмуляторы ПЗУ».

Как правило, каждая фирма-разработчик семейства микроконтроллеров выпускает и вариант отладочного монитора. Он обычно поставляется вместе с отладочными платами.

### 9.4.5. Мезонинная технология

Особенно следует отметить перспективность использования при разработке АС мезонинной технологии, которая унифицирует размеры и интерфейс базовой платы-носителя и размещаемых над ней небольших плат — мезонинов (типичный размер 45x99 мм). Одна плата-носитель несет от 2 до 12 мезонинов. Каждый мезонин соединяется с носителем двумя разъемами, которые выполняют также функции механических держателей. Один из разъемов подключается к локальной шине платы-носителя, функциональное назначение контактов второго разъема определяется типом мезонина, который может содержать многоканальную систему ввода/вывода, сетевые адаптеры и др. устройства.

Используя серийно выпускаемые рядом производителей платы-носители и набор мезонинов, разработчик может быстро реализовать сложнофункциональные целевые системы для разнообразных применений.

Лидерами в этой области являются фирмы «GreenSpring Computers» (США) и «PEP Modular Computer» (Германия), которые выпускают большую номенклатуру плат-носителей и мезонинов. Интеллектуальные платы-носители представляют собой одноплатные компьютеры или контроллеры, реализованные на

базе высокопроизводительных микропроцессоров (MC68030, MC68040 и др.) или микроконтроллеров (MC68332, MC68360 и др.), которые имеют связь с персональным компьютером. Такие носители могут выполнять функции плат развития и использоваться в составе прототипных и целевых систем.

Серийно выпускаемые мезонины (их около 300 типов) выполняют функции дополнительной памяти и различных периферийных устройств: параллельных и последовательных портов, таймеров-счетчиков, АЦП и ЦАП, сетевых и шинных контроллеров и др. При необходимости разработчик может самостоятельно спроектировать мезонин, выполняющий функции, которые необходимы для прототипной или целевой системы.

Таким образом, мезонинная технология является наиболее эффективным средством разработки АС современных электронных систем различного назначения, позволяя конфигурировать их из стандартных плат при минимальных затратах времени и средств на разработку дополнительных АС.

### 9.4.6. Схемные эмуляторы

Одним из наиболее удобных средств отладки программного обеспечения является схемный эмулятор (СЭ), представляющий собой программно-аппаратный комплекс, который в процессе отладки замещает в реализуемой системе микропроцессор или микроконтроллер. В результате такой замены функционирование отлаживаемой системы становится наблюдаемым и контролируемым. Разработчик получает возможность визуального контроля за работой системы на экране дисплея и управления ее работой путем установки определенных управляющих сигналов и модификации содержимого регистров и памяти.

Благодаря наличию таких возможностей СЭ является наиболее универсальным и эффективным отладочным средством, используемым на этапе комплексной отладки системы.

Наиболее широкое применение получили СЭ, подключаемые к базовому управляющему компьютеру типа IBM PC или рабочей станции. Обычно такие СЭ конструктивно оформлены в виде прибора, размещенного в отдельном корпусе с автономным источником питания и соединенного с последовательным COM-

портом базового компьютера. Некоторые типы эмуляторов для ускорения обмена связываются с компьютером через параллельный порт. С помощью плоского кабеля к СЭ подключается эмуляторная головка, которая имеет вилку для включения в систему вместо эмулируемого микропроцессора или микроконтроллера. В головке размещается эмулирующий микропроцессор (микроконтроллер), который выполняет те же функции, что и эмулируемый, но работает под управлением компьютера. Большинство СЭ предназначено для работы с определенным семейством микропроцессоров или МК, причем для эмуляции каждой модели семейства используется соответствующая головка.

В структуру СЭ входят следующие блоки:

– эмулятор микропроцессора или микроконтроллера (размещается в эмуляторной головке);

– память трассы, которая хранит значения сигналов, устанавливаемых на выводах микропроцессора (микроконтроллера) в процессе выполнения программы;

– блок контрольных прерываний, который реализует остановы в контрольных точках, заданных пользователем с клавиатуры компьютера;

– эмуляционная память ( ОЗУ), которая заменяет в процессе отладки внутреннее ПЗУ МК или другие разделы памяти, внешний доступ к которым в процессе отладки ограничен;

– таймер, используемый для контроля времени выполнения отлаживаемых фрагментов программы.

СЭ позволяет вводить в систему тестовую или рабочую программу и контролировать ее выполнение, обеспечивая прерывания в контрольных точках. Условиями прерывания могут быть различные комбинации значений адреса, данных и управляющих сигналов, поступающих на выводы эмулирующего МП или МК. Эти комбинации задаются пользователем с клавиатуры управляющего компьютера. После останова пользователь может получить на экране полную информацию о текущем состоянии регистров и ячеек памяти системы. С помощью памяти трассы можно просмотреть состояния системной шины для определенного числа предыдущих циклов выполнения программы. Дисассемблер дает возможность анализировать выполнение программы в соответствии с ее исходным текстом на языке Ассемблера.

Память трассы работает почти аналогично памяти логического анализатора (о нем будет сказано ниже), поэтому СЭ может выполнять также его функции. Число устанавливаемых контрольных точек обычно составляет несколько десятков, хотя некоторые модели современных СЭ обеспечивают существенно большие возможности. Объем памяти трассы в различных СЭ позволяет контролировать от 4К до 512К программных циклов. Таймер служит для определения времени выполнения фрагментов программы с учетом реальной тактовой частоты системы.

Программное обеспечение СЭ состоит из монитора – служебной программы, обеспечивающей работу всех блоков под управлением базового компьютера, компилятора или Ассемблера, позволяющих программировать работу системы на языке высокого уровня или Ассемблера, и отладчика. Данные программные средства обычно функционируют в составе интегрированной среды проектирования/отладки. Большинство современных СЭ используют символьные отладчики и дисассемблеры, применение которых делает процесс отладки более простым и наглядным. Программное обеспечение СЭ реализует в процессе отладки выдачу данных на экран монитора в удобном для пользователя многооконном формате.

Многие типы СЭ содержат эмуляционное ОЗУ, которое заменяет ПЗУ отлаживаемой системы. Благодаря такой замене можно в процессе отладки производить оперативное изменение содержимого этой памяти. После отладки содержимое эмуляционного ОЗУ переносится в рабочее ПЗУ системы.

Кроме описанных сложно функциональных и дорогих моделей СЭ рядом производителей выпускаются их упрощенные варианты, реализованные на одной печатной плате. Такие СЭ обладают ограниченными возможностями: имеют существенно меньший объем памяти трассы, не реализуют функции ЛА, не обеспечивают символьной отладки и т. д. Однако они позволяют выполнять отладку систем малой и средней сложности, имеют на порядок более низкую стоимость, поэтому находят достаточно широкое практическое применение. Некоторые типы плат развития также выполняют часть функций СЭ.

Некоторые модели СЭ предоставляют возможности анализа эффективности выполняемой программы, обеспечивая инфор-

мацию о частоте обращения к определенным ее фрагментам, и позволяют производить отладку мультипроцессорных систем с помощью организации многоэмуляторных комплексов. Такие комплексы, реализующие набор вышеперечисленных функций, называют отладочными комплексами, или системами развития (development system).

Достоинства и недостатки внутрисхемных эмуляторов.

К достоинствам внутрисхемных эмуляторов следует отнести

– широкий набор функциональных возможностей, что делает внутрисхемные эмуляторы наиболее мощным и универсальным средством отладки;

– работу внутрисхемного эмулятора в реальной схеме электронного блока, в котором предполагается работа МК;

– большую гибкость моделирования временных и электрических характеристик микроконтроллера, что связано с преимущественно программным методом их моделирования

Однако внутрисхемные эмуляторы имеют и недостатки. Основным из них является трудность программного моделирования электрических сигналов на выводах МК в реальном масштабе времени. Для адекватного моделирования быстродействие моделирующего процессора или компьютера должно быть существенно выше, чем эмулируемого микроконтроллера, что возможно далеко не всегда, особенно в случае эмуляции современных высокопроизводительных микроконтроллеров.

Кроме того, даже в случае работы в замедленном масштабе времени, различные модели внутрисхемных эмуляторов могут иметь разного рода ограничения по контролю и управлению функционированием отлаживаемых устройств, что связано с трудностью их моделирования. Например, это может быть некорректное обрабатывание прерываний в пошаговом режиме, или запрет на использование последовательного порта и т.п.

### 9.4.7. Эмуляторы ПЗУ

Этот вид отладочных средств используется при отладке систем, рабочая программа которых размещается в ПЗУ. Эмулятор ПЗУ содержит ОЗУ, подключаемое к системе вместо управляющего ПЗУ, и работает под управлением подключенного к эмуля-

тору базового компьютера. В простейшем случае эмулятор ПЗУ позволяет в процессе отладки выполнять многократное оперативное изменение рабочей программы. Окончательный вариант рабочей программы заносится в ПЗУ системы после отладки.

Более сложные «интеллектуальные» эмуляторы ПЗУ имеют более широкие функциональные возможности. Используя один из входов прерывания системы, они позволяют останавливать ее работу в заданных контрольных точках аналогично схемному эмулятору. При этом на дисплее базового компьютера может быть представлено содержимое эмулирующей памяти. В случае использования в эмуляторе памяти трассы можно обеспечить просмотр предыдущих шагов обращения к ПЗУ, т. е. проверить последовательность выбиравшихся команд. Во многих случаях такая информация является достаточной для выполнения отладки микроконтроллерных систем.

Таким образом, эмуляторы ПЗУ могут выполнить значительную часть функций схемных эмуляторов. При этом их реализация оказывается проще и дешевле, так как они не эмулируют функции микроконтроллера, который в процессе отладки продолжает работать в составе системы. Поэтому эмуляторы ПЗУ являются универсальными средствами, которые могут использоваться для отладки систем с различными моделями МК.

### 9.4.8. Интегрированные среды разработки

Интегрированная среда разработки — это совокупность программных средств, поддерживающая все этапы разработки программного обеспечения от написания исходного текста программы до ее компиляции и отладки, и обеспечивающая простое и быстрое взаимодействие с другими инструментальными средствами (программным отладчиком-симулятором, внутрисхемным эмулятором, эмулятором ПЗУ и программатором).

Строго говоря, интегрированные среды разработки не относятся к числу средств отладки, тем не менее обойти вниманием данный класс программных средств, существенно облегчающий и ускоряющий процесс разработки и отладки микропроцессорных систем было бы неправильно.

При традиционном подходе, начальный этап написания программы строится следующим образом:

1. Исходный текст набирается при помощи какого-либо текстового редактора. По завершении набора, работа с текстовым редактором прекращается и запускается кросс компилятор. Как правило, вновь написанная программа содержит синтаксические ошибки, и компилятор сообщает о них на консоль оператора.

2. Вновь запускается текстовый редактор, и оператор должен найти и устранить выявленные ошибки, при этом сообщения о характере ошибок выведенные компилятором уже не видны, так как экран занят текстовым редактором.

И этот цикл может повторяться не один раз. Если программа имеет большой объем, собирается из различных частей, и подвергается длительному редактированию или модернизации, то даже этот начальный этап может потребовать много сил и времени. После этого наступает этап отладки программы и к редактору с компилятором добавляется эмулятор или симулятор, за работой которого хотелось бы следить прямо по тексту программы в текстовом редакторе.

Избежать большого объема однообразных действий и тем самым существенно повысить эффективность процесса разработки и отладки позволяют т.н. интегрированные среды (оболочки) разработки (Integrated Development Environment, IDE).

Работа в интегрированной среде дает программисту:

– возможность использования встроенного многофайлового текстового редактора, специально ориентированного на работу с исходными текстами программ;

– возможность диагностировать выявленные при компиляции ошибки, и редактировать исходный текст программы;

– возможность организации и ведения параллельной работы над несколькими проектами. Менеджер проектов позволяет использовать любой проект в качестве шаблона для вновь создаваемого проекта;

– перекомпиляции подвергаются только редактировавшиеся модули;

– возможность загрузки отлаживаемой программы в имеющиеся средства отладки, и работы с ними из оболочки;

– возможность подключения к оболочке практически любых программных средств.

В последнее время функции интегрированных сред разработки становятся стандартной принадлежностью программных интерфейсов эмуляторов и отладчиков-симуляторов.

Подобные функциональные возможности, в сочетании с дружественным интерфейсом, в состоянии существенно увеличить скорость разработки программ для микроконтроллеров и процессоров цифровой обработки сигналов.

## 9.5. Средства и методы комплексной отладки МП систем

Под комплексной отладкой микропроцессорной системы будем понимать комплекс мероприятий, включающий одновременное (комплексное) тестирование всех частей системы, как программной, так и аппаратной. Для этого применяются такие средства, как программаторы и логические анализаторы.

### 9.5.1. Программаторы

Эти устройства необходимы на заключительном этапе разработки систем, когда требуется записать отлаженную программу в ПЗУ, которое входит в состав МК или реализуется в виде отдельного модуля. Выпускается два вида программаторов:

1) специализированные программаторные платы;

2) универсальные программаторы.

*Программаторные платы* предназначены для программирования одного типа микроконтроллеров или микросхем ПЗУ, которые включаются в имеющуюся на плате панельку. Платы подключаются к последовательному порту управляющего персонального компьютера, с помощью которого выполняется программирование. Данные вводятся с клавиатуры компьютера, отображаются на его экране и после редактирования загружаются в буферную память. Затем содержимое этой памяти переписывается в программируемое ПЗУ с помощью размещенных на плате формирователей сигналов требуемой мощности и длительности. После программирования выполняется верификация путем считывания и сравнения содержимого ПЗУ и буферной

памяти. При выявлении несовпадений производится повторное программирование соответствующих ячеек.

Некоторые типы программаторных плат работают без управляющего компьютера. Они используют размещенный на плате резидентный контроллер со служебным ПЗУ, где содержится управляющая программа. Вместо буферного ОЗУ на плате располагается панелька для включения микросхемы памяти (ОЗУ или ЭСППЗУ), в которую предварительно записывается требуемое содержимое ПЗУ. Под управлением контроллера производится перезапись этого содержимого в программируемое ПЗУ, верификация результата и, при необходимости, повторное программирование.

Недостатком этих плат является их специализация на программирование одной или нескольких однотипных моделей микроконтроллеров или микросхем памяти. При использовании большой номенклатуры таких изделий целесообразно применять универсальные программаторы, которые выпускаются рядом зарубежных и российских производителей.

*Универсальные программаторы* работают под управлением компьютера и имеют три варианта конструктивной реализации:

1) в виде платы расширения, размещаемой внутри управляющего компьютера, с которой соединяется внешний коммутационный блок, имеющий панельки для включения программируемых изделий (микроконтроллеров или микросхем ПЗУ);

2) в виде отдельного устройства, имеющего на корпусе панель для включения программируемых изделий, подключаемое к последовательному/параллельному порту компьютера.

3) для программирования «в системе» (ISP – In-System Programmable – не путать с SPI – Serial Peripheral Interface, который имеет те же обозначения выводов). Это так называемые внутрисхемные программаторы. Это собственно, даже не отдельные устройства, а блоки программирования, которые интегрированы в микроконтроллер. Эти блоки имеют встроенный генератор «накачки» для создания повышенного напряжения для программирования, служебные цепи для осуществления программирования и специальный последовательный интерфейс для загрузки прикладной программы в ПЗУ микроконтроллера. Название «программирование в системе» очень точно отражает суть этого

типа программирования, так как процесс программирования и перепрограммирования микроконтроллера происходит на печатной плате готовой системы. Данный тип программирования в последние годы завоевывает все большую популярность, так как позволяет сколько угодно перепрограммировать микроконтроллер уже готового устройства, не вынимая и, тем более, не выпаивая его из печатной платы готового устройства.

Для реализации данного вида программирования, как правило, не требуется никакого специального внешнего блока, а микроконтроллер (точнее, выводы программирования) подключается напрямую к какому-либо порту компьютера. Алгоритм программирования осуществляет специальная компьютерная программа-программатор.

### 9.5.2. Логические анализаторы

По настоящему универсальными приборами для анализа функционирования цифровых систем являются логические анализаторы (ЛА). Они позволяют контролировать логическое состояние нескольких десятков точек системы в течение заданного промежутка времени и выдать информацию о состоянии в визуальном (на экране монитора) или печатном виде. Форма представления может быть символьная или графическая (временные диаграммы сигналов). Входные каналы ЛА подключаются к точкам контроля с помощью зондов-клипсов или разъемов. Число каналов в современных ЛА обычно составляет от 16 до 150. Запуск анализатора производится автоматически при поступлении на определенные каналы заданного кода (адреса, данных или комбинации управляющих сигналов) или последовательности кодов. После запуска в память ЛА записывается последовательность значений логических сигналов в точках контроля. Объем этой памяти определяет число контролируемых точек на временной оси (глубину контроля), которое для большинства ЛА составляет от 2 К до 32 К. На экран выводятся несколько десятков точек для каждого канала с возможностью просмотра всей записанной в памяти последовательности состояний. Максимальная частота дискретизации временных интервалов для различных моделей ЛА имеет значение от 20 до 200 МГц.

ЛА реализуются в виде автономных измерительных приборов или плат расширения, подключаемых к базовому (host) персональному компьютеру. Эти приборы часто включают ряд дополнительных устройств, например, программируемый генератор тестовых последовательностей. ЛА, реализованные в виде автономных приборов, выпускаются рядом ведущих производителей электронно-измерительной аппаратуры: Tektronix, Hewlett-Packard, John Fluke и др. Наиболее широко при отладке систем используются ЛА типа 16500B (Hewlett-Packard), 3001GPX и 3002GPX (Tektronix), PM3580 (Fluke), CLAS 4000 (Embedded Performance/ Biomation). Их стоимость составляет несколько тысяч долларов.

Для обеспечения разработчиков недорогими средствами контроля состояния системы ряд производителей выпускает анализаторные платы, подключаемые к базовому персональному компьютеру, который программируется на выполнение значительной части функций ЛА. При этом для хранения последовательности состояний используется память базового компьютера. Визуализация временных диаграмм в символьной или графической форме выполняется на дисплее его монитора, можно выполнить распечатку результатов измерений на принтере. Базовый компьютер управляет процессом измерения и производит обработку результатов. Благодаря этому анализаторная плата оказывается достаточно простой и на порядок более дешевой, чем автономный ЛА.

### 9.5.3. Встроенные в микропроцессоры средства отладки

Следует отметить, что многие модели микропроцессоров и микроконтроллеров, выпускаемых фирмой «Motorola», имеют специальный режим отладки BDM, при котором реализуется ввод команд, ввод/вывод данных, управление режимом работы процессора с помощью специального последовательного порта. При его использовании микропроцессор или микроконтроллер может работать в режиме эмуляции под управлением подключаемого к этому порту компьютера. Режим BDM позволяет существенно облегчить процедуру комплексной отладки и использовать при этом более простые и дешевые средства.

**Литература**

1. LCD Controller/Driver LSI. Data Book. – 1994 Hitachi America Ltd. – [Электронный ресурс]. – Режим доступа: http://www.hitachi.com/catalog/datasheets/lcddriver/lsi.htm.

2. Алфавитно-цифровые индицирующие ЖК-модули фирмы Powertip: Каталог. 1-е издание. – КТЦ-МК, 1998.

3. [Электронный ресурс]. – Режим доступа: http://www.powertip.com.

4. The $I^2C$-bus and how to use it. – Philips, 1997. – [Электронный ресурс]. – Режим доступа: http://www.philips.com/ver10/databook/i2c/i2c.htm

5. Микропроцессорные системы: Учебное пособие для вузов / Е.К. Александров, Р.И. Грушвицкий, М.С. Куприянов, О.Е. Мартынов, Д.И. Панфилов, Т.В. Ремизевич, Ю.С, Татаринов, Е.П. Угрюмов, И.И. Ша Под общ. ред. Д.В. Пузанкова. – СПб.: Политехника, 200. с.: ил.