Министерство образования Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

А.Е. Васильев

МИКРОКОНТРОЛЛЕРЫ
Разработка встраиваемых приложений

Учебное пособие

Санкт-Петербург
Издательство СПбГПУ
2003

Министерство образования Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

А.Е. Васильев

МИКРОКОНТРОЛЛЕРЫ
Разработка встраиваемых приложений

Учебное пособие

Санкт-Петербург
Издательство СПбГПУ
2003

УДК 681.32.

Васильев А.Е. Микроконтроллеры. Разработка встраиваемых приложений: Учеб. пособие. СПб.: Изд-во СПбГПУ, 2003. 210 с.

Пособие соответствует государственному образовательному стандарту дисциплины "Микроконтроллеры и микропроцессоры в системах управления" направления бакалаврской подготовки 552800 "Информатика и вычислительная техника".

В пособии рассмотрены архитектурные принципы построения микроконтроллеров, приемы проектирования, отладки и диагностирования систем управления на базе встраиваемых микроконтроллеров, а также приведены примеры решения типовых задач, возникающих при разработке подобных систем.

Предназначено для студентов 4-го курса факультета технической кибернетики, изучающих дисциплины "Микроконтроллеры и микропроцессоры в системах управления" и "Микропроцессорные системы" в рамках бакалаврской подготовки, а также в качестве дополнительного материала при изучении курсов, связанных с применением микроконтроллеров в системах управления.

Печатается по решению редакционно-издательского совета Санкт-Петербургского государственного политехнического университета.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Неослабевающие темпы роста сложности технологических объектов и требований к качеству управления ими влекут за собой необходимость использования в системах управления средств вычислительной техники, многофункциональность которых позволяет обеспечить множественность режимов управления, повысить способности системы к адаптации, обеспечить требуемые надежностные характеристики.

К числу задач, ставящихся перед системами управления, традиционно относятся следующие:

- формирование сложных импульсных последовательностей;
- определение длительности временных интервалов;
- аналого-цифровое и цифро-аналоговое преобразование;
- ввод-преобразование-вывод цифровых кодов и другие.

С 70-х годов разработчики систем управления стали использовать вычислительные системы на базе микропроцессоров, выпуск которых был освоен рядом производителей (Intel, Motorola и др.). Применение этой технологии разработки систем управления позволяло повысить скорость и эффективность проектирования новых систем на базе старых, снизить затраты на обнаружение и устранение неисправностей, а также удешевить производство.

Однако в силу своих архитектурных ограничений микропроцессоры не обладали возможностью непосредственно решать задачи управления, и для достижения поставленных целей разработчики вынуждены были снабжать их набором дополнительных устройств: памятью программ и данных, а также набором периферийных элементов: таймерами, счетчиками, аналого-цифровыми и цифро-аналоговыми преобразователями, программируемыми контроллерами ввода-вывода и т.п.

Характерный пример структуры подобной системы показан на рисунке В-1.

Здесь применен микропроцессор МП с шинами данных ШД, адреса ША и управления ШУ. Программа функционирования МП занесена в энергонезависимую память ПЗУ, необходимые переменные и выборки измерений хранятся в оперативной памяти ОЗУ. Объект управления ОУ имеет аналоговый исполнительный механизм ИМ и снабжен системой аналоговых датчиков $Д_1$ - $Д_N$. Для возможности изменения оператором режимов работы системы применяется пульт управления ПУ.

Взаимодействие МП с ОУ и ПУ напрямую невозможно, и для их сопряжения применен ряд периферийных устройств.

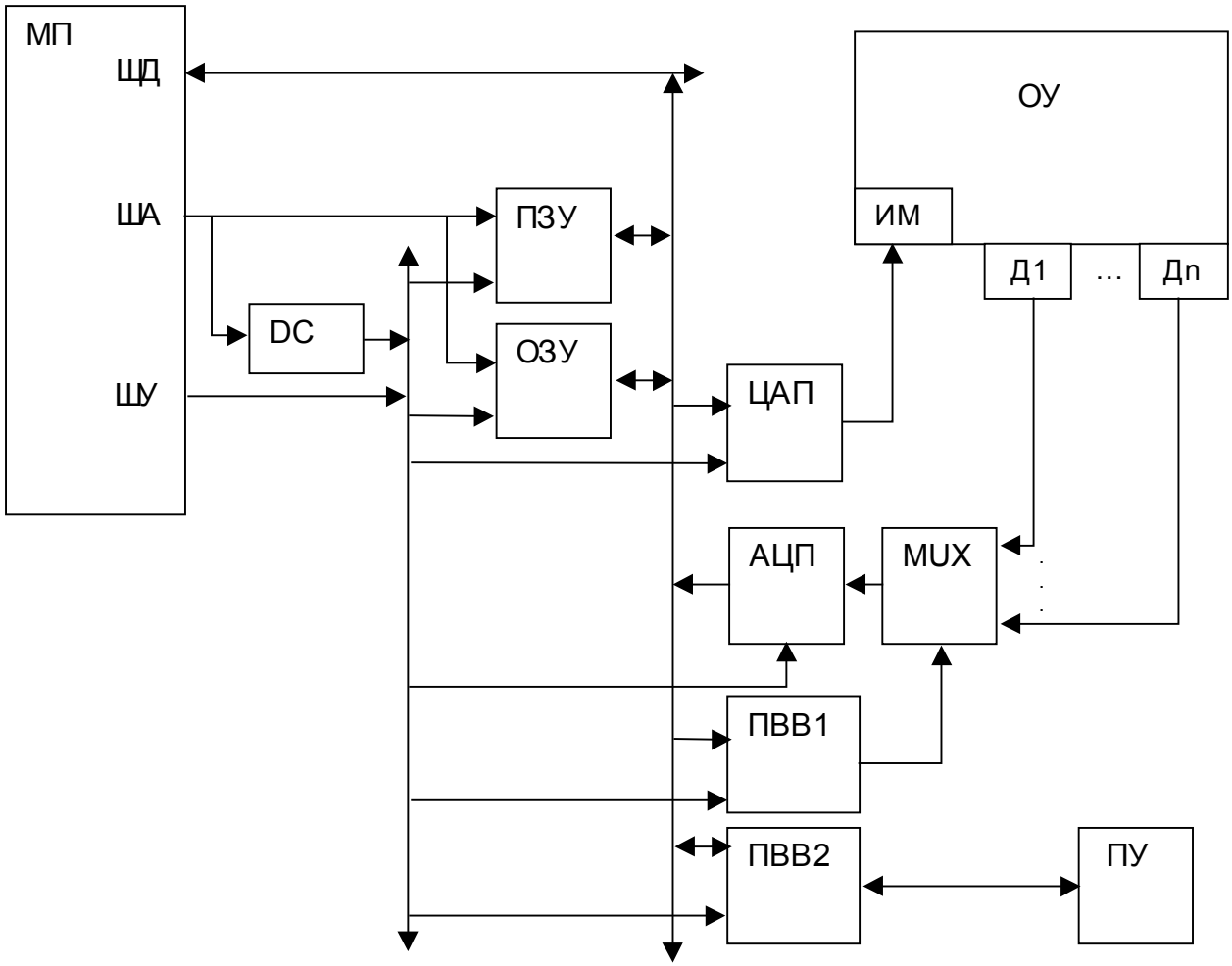


Рис. В-1. Структура системы управления на базе микропроцессора.

Дешифратор DC дополняет ШУ сигналами активизации устройств по их адресу. Процессор ввода-вывода ПВВ1 на основе информации с ШД под воздействием сигналов с ШУ управляет мультиплексором MUX для подачи на вход аналого-цифрового преобразователя АЦП аналогового сигнала с одного из датчиков. АЦП преобразует сигнал в цифровую форму и выдает его на ШД под воздействием сигналов с ШУ. Процессор ввода-вывода ПВВ2 передает в систему информацию о воздействиях оператора, взаимодействуя с ПУ. Цифро-аналоговый преобразователь ЦАП по коду с ШД и управляющим сигналам с ШУ выдает на ИМ ОУ управляющее воздействие.

Описанная структура с непринципиальными изменениями употреблялась в разработках достаточно часто, в связи с чем возникла идея интеграции наиболее часто применяемых элементов систем управления на одном кристалле (подобная идея интеграции применялась и ранее, что дало разработчикам возможность скомпоновать набор транзисторов в виде интегральной микросхемы).

Воплощение идеи произошло в 1978 году с выпуском фирмой Intel устройства под кодовым обозначением 8048, позднее получившего название микроконтроллер (МК) и ставшего основой систем управления, встраиваемых в робототехнические комплексы, бытовую электронику и др.

Под микроконтроллером здесь и далее подразумевается программируемое вычислительное устройство, обладающее набором периферийных устройств и применяемое для решения задач управления в технических системах.

Появившись на рынке, МК наращивали свою популярность, и в настоящее время применяются чрезвычайно широко - объем выпуска микроконтроллеров продолжает увеличиваться и составляет в настоящее время около двух миллиардов штук в год [3].

По области применения, структурной организации, разрядности, набору периферийных устройств, системе команд и прочим признакам МК сгруппированы в семейства, число которых достаточно велико. К наиболее ярким представителям различных семейств следует отнести 32-разрядные микроконтроллеры фирмы Motorola, 16-разрядные микроконтроллеры MCS-96 Intel, RISC-микроконтроллеры AVR фирмы Atmel, миниатюризированные PIC-контроллеры фирмы MicroChip, микроконтроллеры общего назначения SAB фирмы Siemens.

Рассмотрению архитектурных особенностей микроконтроллеров, методов и средств разработки систем управления на их основе посвящено настоящее учебное пособие.

РАЗДЕЛ 1. ОСНОВЫ ОРГАНИЗАЦИИ МИКРОКОНТРОЛЛЕРОВ.

1.1. Структура микроконтроллера.

Микроконтроллер представляет собой вычислительную систему, реализованную в виде одной интегральной схемы, и включает следующие основные блоки: ядро, память программ и память данных, периферийные устройства (рис 1-1).

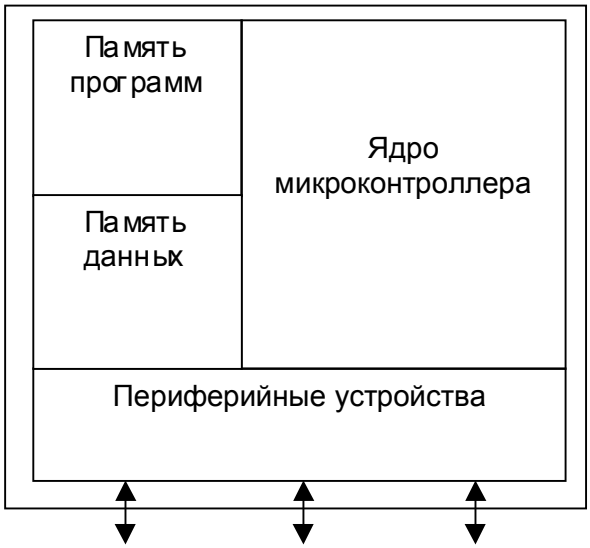


Рис. 1-1. Обобщенная структурная схема микроконтроллера.

Ядро микроконтроллера реализует процесс управления, задаваемый программой. На базе микроконтроллерного ядра фирмами-производителями интегральных схем разрабатываются изделия, различные по номенклатуре модулей памяти и периферийных устройств, но совместимые между собой по системе команд и циклам обмена данными. Множество совместимых по этому признаку МК носит название семейства микроконтроллеров.

Память программ предназначена для хранения управляющих программ. Необходимые для процесса управления данные располагаются в памяти данных.

Периферийные устройства предназначены для обеспечения сопряжения МК с внешними объектами и аппаратной реализации ряда управляющих функций.

Микроконтроллеры, как и вычислительные машины других классов, реализуются на основе гарвардской или принстонской архитектур (рис 1-2). В микроконтроллерах, выполненных на основе гарвардской архитектуры, программы и данные располагаются в логически независимых блоках памяти с различными методами доступа. В микроконтроллерах, выполненных на основе принстонской архитектуры, программы и данные могут располагаться в общем блоке памяти; для обращения используется единый метод доступа.

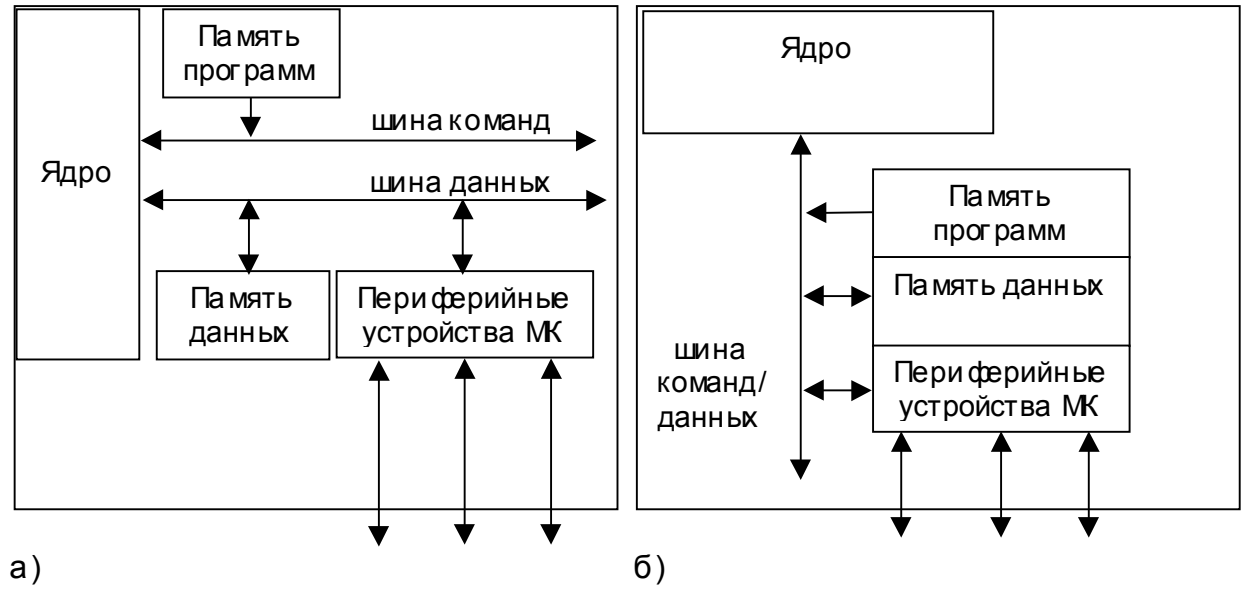


Рис. 1-2. Гарвардская (а) и принстонсткая (б) архитектуры МК.

К числу типовых, наиболее часто интегрируемых на кристалл МК периферийных устройств, относятся следующие блоки:

- параллельные цифровые порты ввода-вывода, осуществляющие обмен данными, представленными в виде логических сигналов;
- таймеры-счетчики, осуществляющие формирование временных интервалов и выполняющие подсчет логических событий;
- узлы аппаратной обработки событий с привязкой по времени;
- цифро-аналоговые и аналого-цифровые преобразователи, осуществляющие вывод и ввод непрерывных сигналов;
- последовательные порты ввода-вывода, осуществляющие обмен данными в распределенных системах;
- блоки обслуживания прерывающих событий;
- средства повышения надежности функционирования.

Каждый периферийный узел МК обладает возможностью настройки посредством записи управляющих кодов в программно доступные конфигурационные регистры узла, называемые регистрами специальных функций. Настройка позволяет производить выбор режима работы устройства (например, требуемой разрядности таймера, направления передачи данных на разрядах параллельного порта и т.п.).

Состав размещаемых на МК периферийных блоков зависит от целевого назначения устройства и определяется производителем на основе типовых задач, реализуемых на микроконтроллерах данного семейства.

1.2. Ядро микроконтроллера.

В состав ядра МК входят процессор, тактовый генератор и контроллер шины (рис. 1-3). Процессор непосредственно осуществляет процесс переработки информации, представленной в виде двоичных кодов, и управление этим процессом в соответствии с программой, представляющей собой последовательность команд. Тактовый генератор осуществляет формирование последовательности опорных сигналов, синхронизирующих протекание процессов в узлах МК, на основе внешней последовательности опорных импульсов. Контроллер шины осуществляет формирование распространяемой по внутренней шине многофазной импульсной последовательности, тактирующей различные стадии выполнения команд в МК, и необходимой для организации обмена данными с периферийными устройствами МК.

Команды располагаются по заданным адресам (номерам ячеек) в памяти команд и представляют собой управляющие коды, описывающие выполняемую операцию и задающие операнды (данные, над которыми выполняется операция).

Каждый МК обладает определенной системой команд, характеризуемой списком команд и их форматом. Список команд представляет собой набор операций, выполнение которых предусмотрено на процессоре данного МК. В списке команд любого МК можно выделить четыре группы операций:

- операции передачи данных (между ячейками памяти МК, а также другими программно доступными элементами МК);
- арифметические операции (сложение, вычитание, умножение, деление);
- логические операции ("И", "ИЛИ", инверсия, исключающее "ИЛИ", различные сдвиги);
- операции передачи управления (безусловный переход по заданному адресу, переход по условию равенства или неравенства операндов, переход на подпрограмму и возврат из нее и т.п.).

Формат команды позволяет определить тип выполняемой на очередном шаге программы операции, входные и выходные операнды, а также адрес команды, подлежащей выполнению на следующем шаге программы.

Тип выполняемой команды задается кодом операции (КОП).

Для задания операндов применяются следующие методы указания их локализации (способы адресации):

- неявная: операнд не указывается в связи с однозначностью доступа к нему (например, в связи с единственно возможным его размещением);

- непосредственная: входной операнд помещается в тело команды (например, с целью задания констант);

- прямая: в команде указывается адрес в памяти данных, по которому расположен операнд;

- косвенная: в команде указывается адрес ячейки в памяти данных, содержащей адрес ячейки в памяти данных, по которому расположен операнд (например, при организации доступа к последовательно расположенным данным при неоднократном повторении участка программы удобно изменять значение операнда команды, тем самым меняя адрес искомого данного);

- относительная: в команде указывается адрес ячейки в памяти данных, содержимое которой, будучи сложенное с некоторой величиной (например, задаваемой неявно), даст адрес ячейки в памяти данных, по которой расположен искомый операнд (например, при обращении к элементу таблицы данных, удобно определять искомый операнд по смещению относительно начала таблицы).

Адрес следующей исполняемой команды задается неявно как адрес памяти программ, следующий за адресом выполняемой в данный момент команды, что объясняется преобладанием в большинстве программ линейных участков последовательностей команд. Для его явного задания при организации циклов, подпрограмм, ветвлений по условиям и т.п., применяют команды, КОП которых кодирует определенную операцию передачи управления.

В состав систем команд большинства МК включены (по количеству адресуемых в одной команде операндов) одно-, двух-, трех- и безадресные команды.

Процедура выполнения команд в МК сводится к следующему.

По окончании действия импульса сброса проводится инициализация регистров ядра МК. В указатель команды заносится адрес начального пуска.

По адресу, содержащемуся в указателе команды, из области памяти программ под воздействием управляющих сигналов, формируемых контроллером шины, в регистр команд загружается очередная команда исполняемой контроллером программы.

Выполнение любой команды представляет собой последовательность элементарных действий (микроопераций): определение количества требуемых для операции операндов, определение локализации необходимых операндов, их извлечение, формирование кода действия для исполнительного блока, ожидание окончания исполнения операции, определение локализации результатов, занесение ре-

зультатов, определение адреса следующей команды и ряд других. Конкретный перечень микроопераций, реализуемый при выполнении очередной команды, определяется ее КОП.

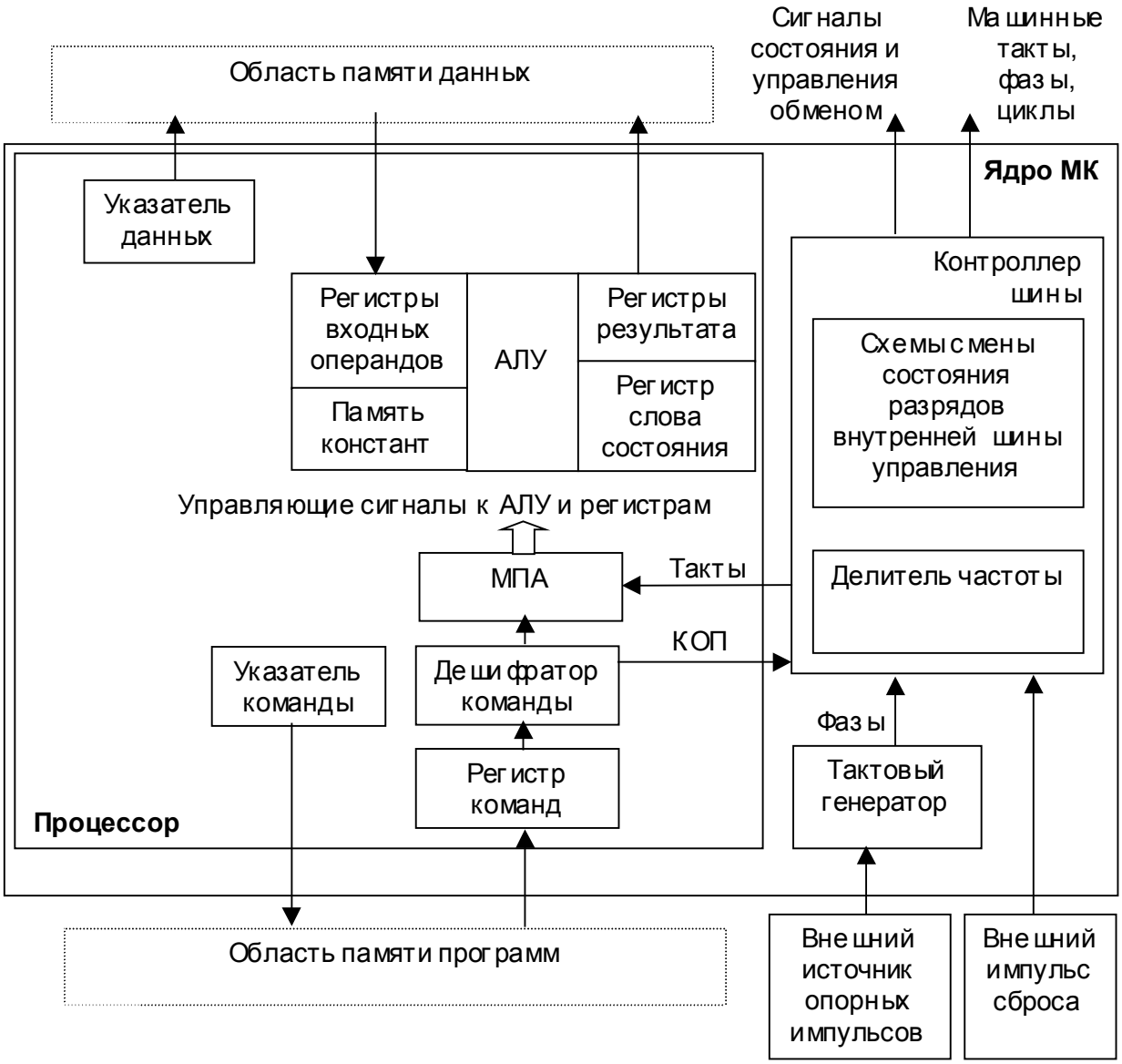


Рис. 1-3. Структурная схема ядра микроконтроллера.

Для настройки схем процессора на требуемую микрооперацию используются последовательности управляющих сигналов. КОП считанной из памяти программ команды дешифрируется и поступает на микропрограммный автомат (МПА), вырабатывающий с каждым очередным тактом синхронизации, поступающим от контроллера шины, необходимый на данной стадии отработки команды набор управляющих сигналов.

Выполнение в процессоре арифметических и логических операций, сдвигов, обнуления и т.п. обеспечивается арифметико-логическим устройством (АЛУ). Память констант обеспечивает выработку корректирующего кода при двоично-

десятичном представлении данных, кода маски при выполнении в АЛУ операций над битами, а также выдачу кодов констант. Для временного хранения данных, над которыми в АЛУ выполняется операция, предназначены регистры входных операндов, информация в которые заносится из области памяти данных с применением соответствующего указателя.

По окончании выполнения операции в АЛУ ее результаты заносятся в регистры результатов, а также формируются признаки результатов операции (переполнение, сдвиг, знак и т.п.), заносимые в регистр слова состояния процессора и доступные для считывания и анализа программой (например, для организации перехода на иную ветвь программы в связи с наличием арифметического переполнения). Затем в общем случае в указатель данных последовательно заносятся адреса ячеек памяти данных, в которые необходимо поместить результаты команды (эти адреса извлекаются из полей адресации выходных операндов), и из регистров результата операнды заносятся в ячейки памяти данных, адресуемые указателем данных. (Следует отметить, что, как правило, в процессоре МК используется дополнительный указатель на память данных, снабженный механизмами автоувеличения при занесении операндов в память и автоуменьшения при извлечении операндов из памяти. Такой метод доступа к памяти называется стековым, а выделяемая в памяти данных область для этих манипуляций называется стеком. Стек используется и при организации подпрограмм, в частности, подпрограмм обработки прерываний).

После размещения выходных результатов происходит автоувеличение указателя команд, либо, в случае выполнения ветвления в исполняемой программе, в него заносится содержимое заданного поля операнда в исполненной команде. В обоих случаях, в указателе команд оказывается адрес ячейки памяти, содержащей очередную подлежащую выполнению команду, и описанный процесс повторяется.

Например, выполнение гипотетической команды PLUS opX, opY, opZ (opZ=opX+opY), вызовет следующие действия: загрузка адреса opX в указатель данных, подача сигнала "чтение памяти данных", ожидание готовности памяти данных, загрузка с шины данных памяти данных в регистр №1 входного операнда АЛУ, выполнение аналогичных действий для opY (с записью в регистр №2), подача на АЛУ кода "сложить", ожидание готовности АЛУ, загрузка адреса opZ в указатель данных, подача данных из регистра №1 выходного операнда АЛУ на шины данных памяти данных, подача сигнала "запись памяти данных", ожидание готовности памяти данных.

1.3. Память микроконтроллера.

На кристалл микроконтроллера интегрированы два блока памяти: память программ и память данных. В связи с ориентацией МК на функционирование в автономном режиме память программ должна сохранять содержимое в отсутствие напряжения питания (т.е. являться энергонезависимой), а для упрощения внутренней архитектуры МК и возможности работы в широком диапазоне частот тактового генератора память данных должна обладать статической архитектурой (т.е. не требовать регенерации).

Обобщенная структура модуля памяти показана на рис. 1-4. Модуль памяти состоит из матрицы запоминающих элементов, организованной в виде N m-разрядных строк, дешифратора адреса ячейки и буферного каскада.

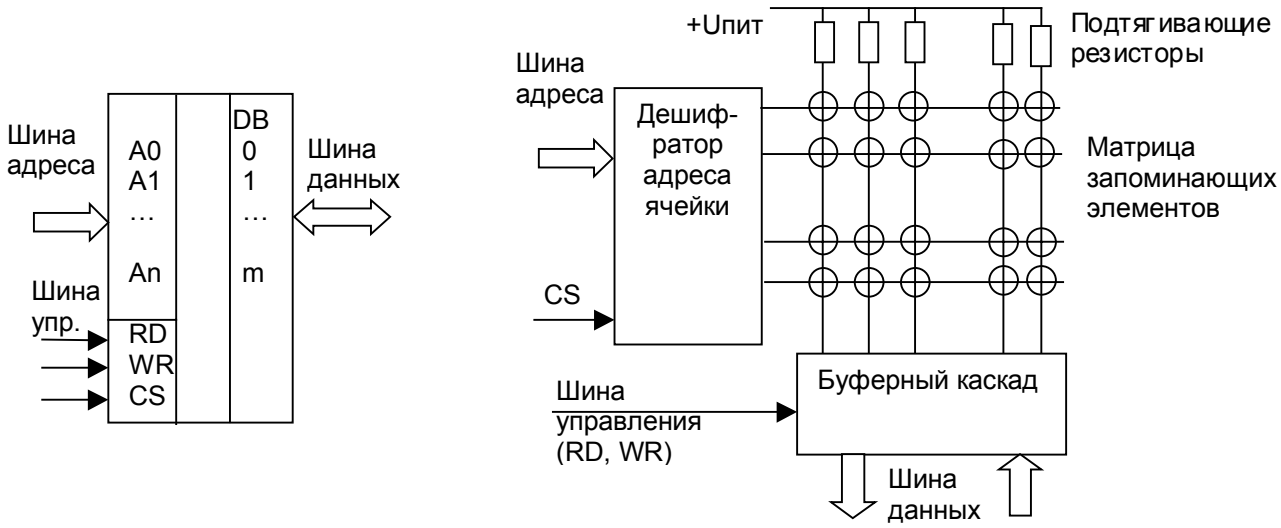


Рис. 1-4. Обобщенная структура модуля памяти.

Разрядность шины адреса такого модуля памяти составляет $n=\log_2 N$, а разрядность шины данных – m. Информация о номере подлежащей выборке ячейки в виде кода адреса поступает на дешифратор, активизирующий одну из строк матрицы запоминающих элементов генерацией высокого логического уровня на одном из своих выходов. При этом (в зависимости от поступающих сигналов управления) логические уровни всех запоминающих элементов выбранной строки поступают через буферный усилительный каскад на шину данных (ситуация чтения состояния ячейки), либо передаются с шины данных через буферный усилительный каскад на запоминающие элементы выбранной строки (ситуация записи состояния ячейки). Ло-

гические состояния запоминающих элементов прочих строк не изменяются и не оказывают влияния на выходные логические уровни.

Энергонезависимая память программ является постоянным запоминающим устройством (ПЗУ). Каждый запоминающий элемент ПЗУ находится в том логическом состоянии, в которое он был переведен при занесении информации в ПЗУ (программировании).

В зависимости от количества допустимых циклов записи управляющей программы в ПЗУ различаются однократно и многократно программируемые модули.

В однократно программируемых ПЗУ каждый запоминающий элемент матрицы допускает только одну смену состояния. Запись программы в ПЗУ может производиться либо в условиях промышленного производства при изготовлении кристалла микроконтроллера ("по маске"), либо пользователем с помощью программатора. ПЗУ такого типа наиболее дешевы, так как каждый элемент матрицы предельно прост (рис. 1-5).

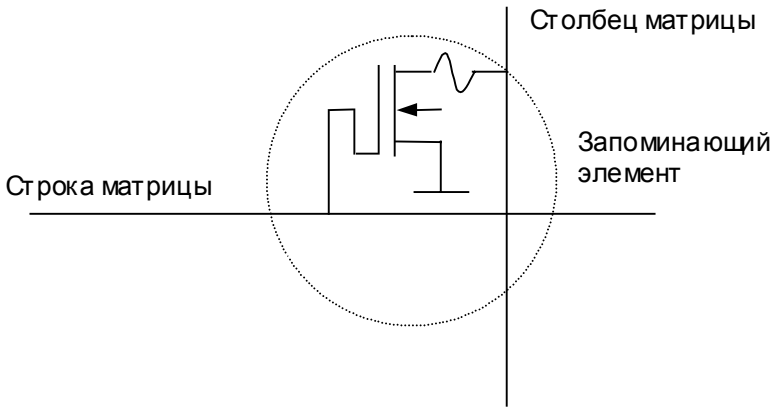


Рис. 1-5. Организация запоминающего элемента однократно программируемых ПЗУ программ.

При программировании ячейки на разряды шины данных прикладывается повышенное напряжение к тем линиям, которые должны быть переведены в состояние логической единицы. Так как в это же время на строку матрицы, задающую программируемую ячейку, поступает высокий уровень от дешифратора адреса, через плавкие перемычки протекает сильный ток, необратимо разрушающий их. Элементы, программируемые в состояние логического нуля, не подвергаются воздействию повышенных напряжения и тока, в связи с чем остаются в исходном состоянии.

При чтении на строку матрицы, задающую считываемую ячейку, поступает высокий уровень от дешифратора адреса. Если при программировании переход столбца в данной ячейке был разрушен, на столбце будет присутствовать высокий

логический уровень, формируемый при помощи подтягивающего резистора. Если при программировании переход столбца в данной ячейке был сохранен, на столбце будет присутствовать низкий логический уровень, величина напряжения которого определяется соотношением сопротивлений подтягивающего резистора столбца и открытого перехода активного элемента.

В многократно программируемых ПЗУ каждый запоминающий элемент матрицы допускает несколько смен состояний. В связи с этим усложнен как запоминающий элемент, представляющий собой транзистор с плавающим затвором (рис. 1-6), так и последовательность программирования, состоящая из нескольких этапов.

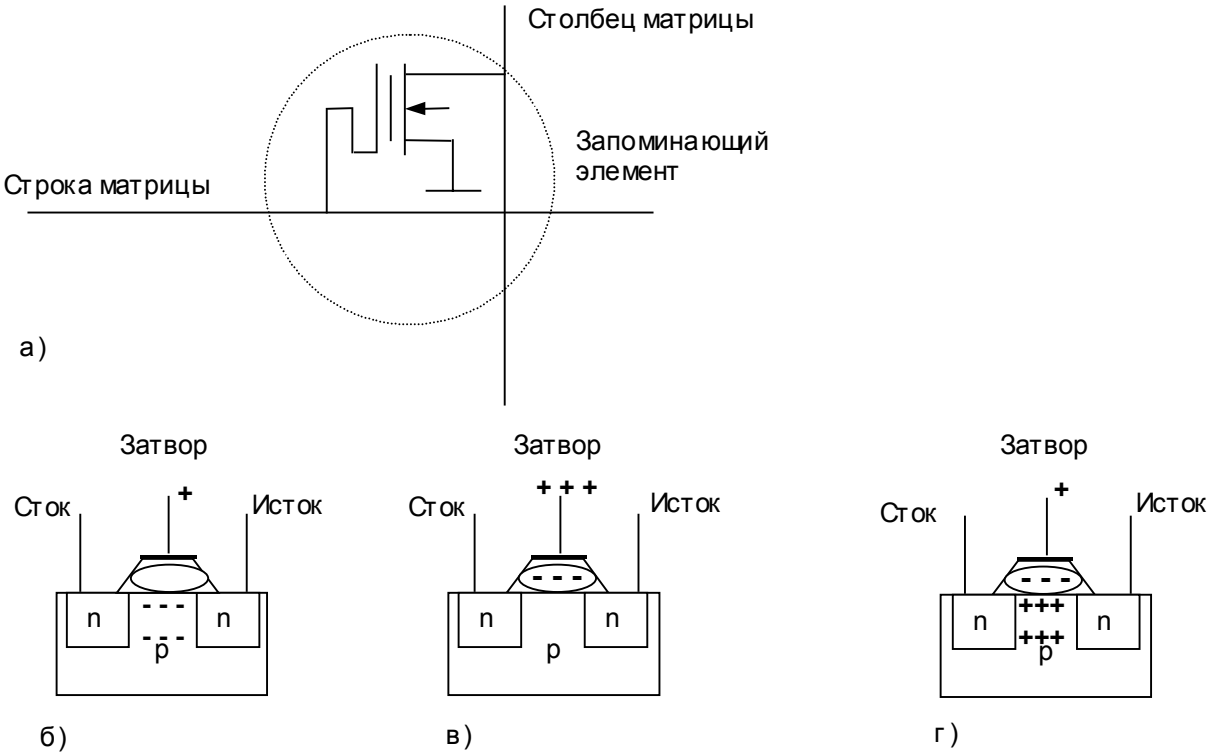


Рис. 1-6. Организация запоминающего элемента многократно программируемых ПЗУ программ.(а – архитектура, б – незапрограммированный элемент: наведение канала возможно, в – программирование элемента, г – запрограммированный элемент: наведение канала невозможно).

Технология изготовления запоминающего элемента допускает принудительное введение ("инжекцию") в околозатворную область транзистора при программировании дополнительного количества отрицательного заряда, препятствующего наведению канала в транзисторе при подаче на затвор напряжения высокого уровня. В этом случае запоминающий элемент матрицы постоянно хранит состояние логической единицы. В случае если инжекция заряда не проводилась, при подаче на

затвор напряжения высоко уровня открывается канал "сток-исток", и выходное напряжение в столбце снижается до нулевого логического уровня.

Для обеспечения возможности проведения повторной записи в ПЗУ необходим предварительный перевод всех запоминающих элементов матрицы в исходное (незапрограммированное) состояние, заключающийся в выведении введенного при предыдущем программировании избыточного заряда из околозатворного пространства транзистора (т.е. в стирании ячеек памяти). Стирание может производиться с помощью электрических импульсов или ультрафиолетового излучения (в последнем случае на поверхности микросхемы МК располагается окно для пропускания УФ-излучения). Число циклов программирования ПЗУ с УФ-стиранием составляет несколько сотен, с электрическим стиранием – несколько тысяч, что объясняется более щадящим режимом стирания во втором случае, дольше сохраняющем нормативные характеристики функционирования транзисторов запоминающих элементов.

Память данных МК является оперативным запоминающим устройством (ОЗУ) со статической организацией запоминающего элемента (рис. 1-7). Такая организация позволяет хранить записанную информацию неограниченно долго без необходимости регенерации, что позволяет МК функционировать в широком диапазоне частот от 0 Гц до F_{MAX}.

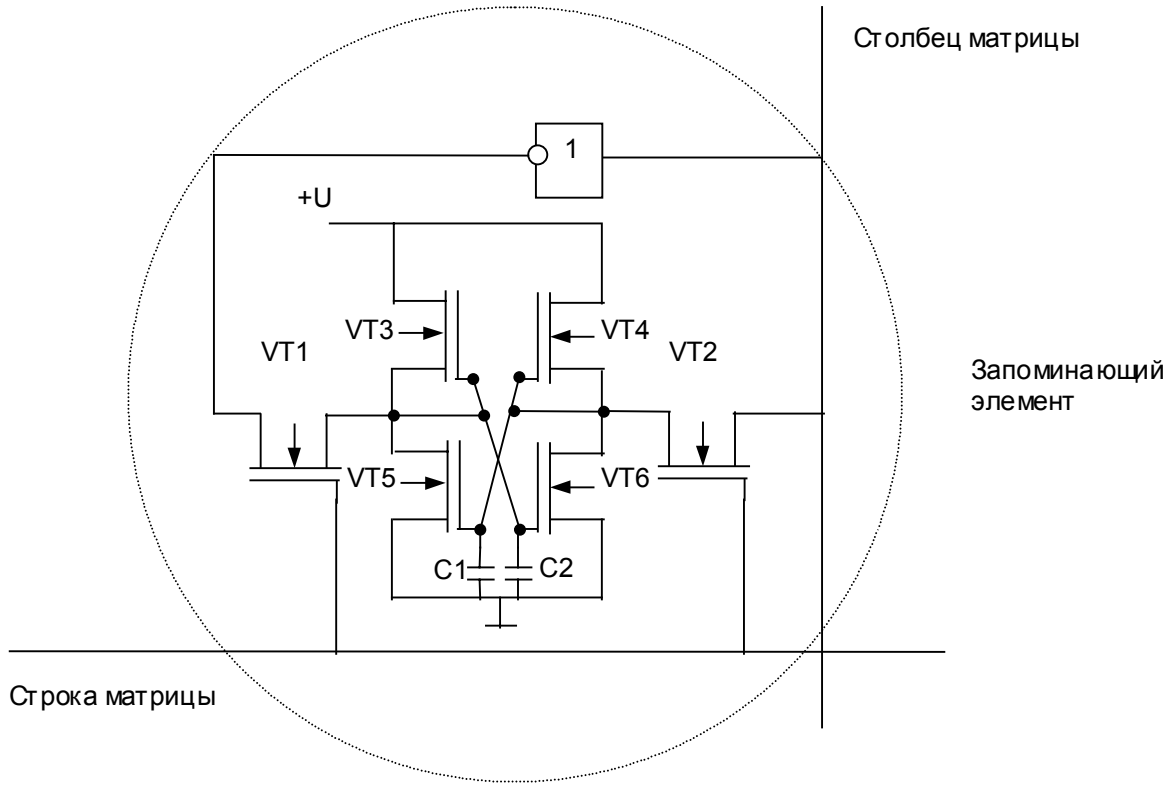


Рис. 1-7. Организация запоминающего элемента ОЗУ данных.

В качестве элемента памяти применяют статический триггер с парафазным входом/выходом, коммутируемый посредством транзисторов VT1 и VT2 к линии столбца матрицы при активизации данной строки дешифратором адреса ячейки памяти.

При записи в запоминающий элемент кода логической единицы на столбце присутствует высокий логический уровень, передаваемый через открытый транзистор VT2 на правое плечо триггера. Одновременно на левое плечо триггера через открытый транзистор VT1 поступает низкий логический уровень, полученный инвертированием значения логического уровня столбца матрицы. Конденсатор C1 заряжается практически до уровня логической единицы, конденсатор C2 разряжается практически до уровня логического нуля. В этом случае транзисторы VT4 и VT5 открыты, а транзисторы VT3 и VT6 – закрыты. Состояние запоминающего элемента будет сохраняться и после снятия напряжения высокого логического уровня со строки матрицы, активизирующего данную ячейку, так как является устойчивым состоянием триггера.

При записи в запоминающий элемент кода логического нуля на столбце присутствует низкий логический уровень, передаваемый через открытый транзистор VT2 на правое плечо триггера. Одновременно на левое плечо триггера через открытый транзистор VT1 поступает высокий логический уровень, полученный инвертированием значения логического уровня столбца матрицы. Конденсатор C1 разряжается практически до уровня логического нуля, конденсатор C2 заряжается практически до уровня логической единицы. В этом случае транзисторы VT3 и VT6 открыты, а транзисторы VT4 и VT5 – закрыты. Состояние запоминающего элемента будет сохраняться и после снятия напряжения высокого логического уровня со строки матрицы, активизирующего данную ячейку, поскольку, как и предыдущее, является устойчивым состоянием триггера.

При чтении буферный каскад модуля памяти (см. рис. 1-4) настраивается на прием данных, и, после активизации необходимого запоминающего элемента выбором строки матрицы, напряжение, соответствующее логическому уровню хранимого в элементе бита данных, посредством транзистора VT2 будет передано на столбец матрицы памяти.

Конфигурационные регистры периферийных устройств нередко располагают в адресном пространстве внутренней памяти, что позволяет организовать взаимодействие с ними через команды пересылок, т.е. упростить процедуру обмена.

1.4. Параллельные порты ввода-вывода.

Параллельные порты ввода-вывода предназначаются для обмена микроконтроллера и внешнего объекта данными, представленными в виде логических сигналов, передаваемыми через линии ввода-вывода микросхемы МК. В общем случае с каждым портом связаны регистр данных (для хранения выводимой из МК на объект информации или для хранения информации, введенной в МК с объекта), система управления (для задания режимов работы порта) и выходной каскад, решающий задачи усиления и сопряжения сигналов. Структура порта показана на рис. 1-8.

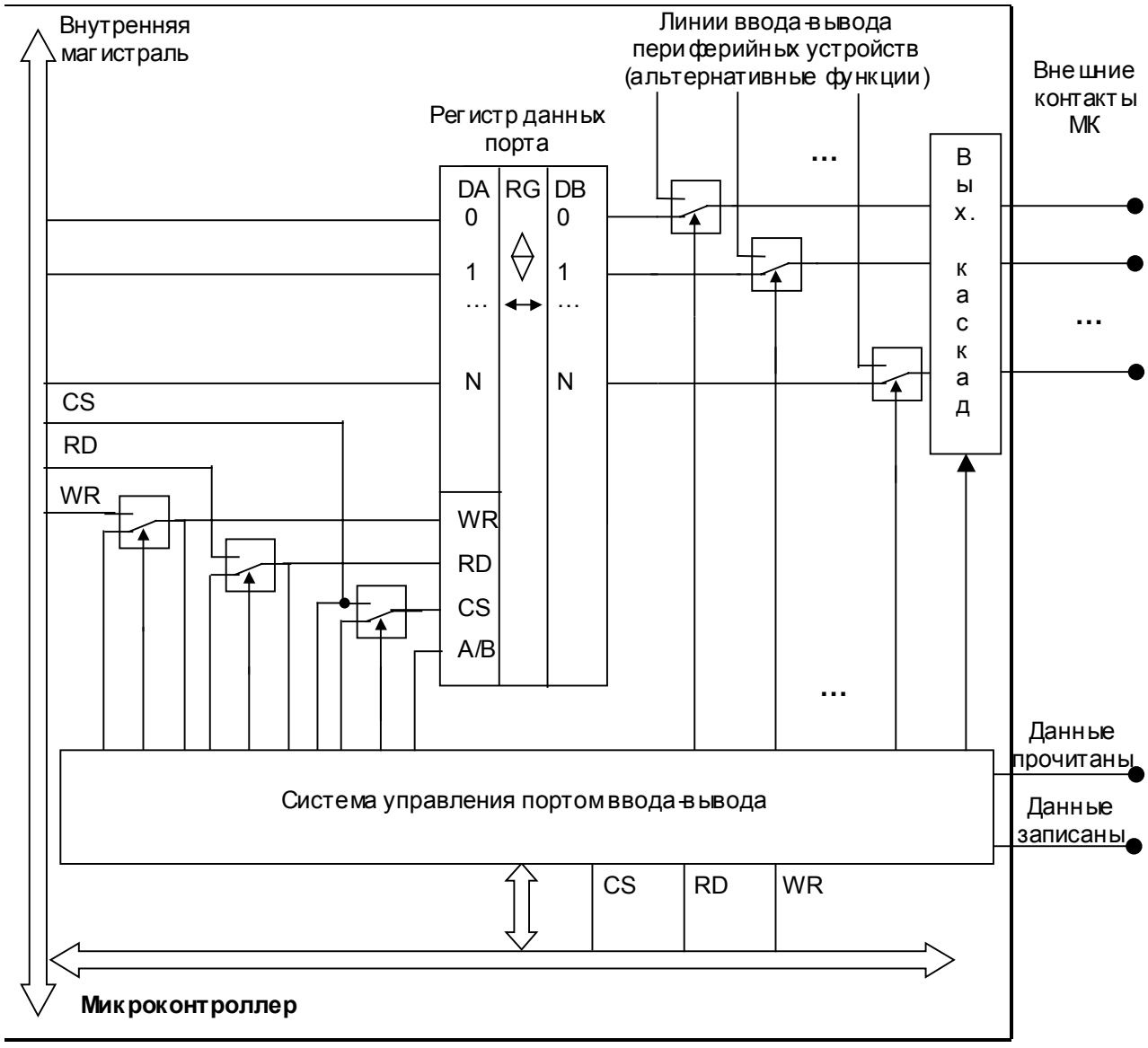


Рис. 1-8. Структура параллельного порта ввода-вывода.

Порты ввода-вывода реализованы во всех моделях МК.

Регистр данных представляет собой двунаправленный N-разрядный регистр ввода-вывода с линиями управления записью и чтением, позволяющими заносить в регистр информацию с внутренней шины МК, или выдавать на внутреннюю шину состояние регистра. Момент обмена информацией между процессором ядра МК и регистром порта через внутреннюю шину определяет контроллер шины ядра. В зависимости от типа команды обмена (пересылка из процессора в порт или из порта в процессор) формируются сигналы WR или RD, после чего обмен стробируется сигналом управления CS.

В общем случае, для возможности решения более сложных задач обмена, в структуру порта ввода-вывода вводится система управления, представляющая собой комбинационную схему обнаружения и изменения состояний сигналов, а также программно доступный для записи и чтения регистр, сохраняющий эти состояния. Таких задач две: это задача мультиплексирования линий ввода-вывода и задача поддержки расширенных протоколов обмена. Рассмотрим их более подробно.

Помимо регистра данных, одни и те же контакты МК могут потребоваться некоторому периферийному устройству (например, таймеру) для обмена информацией с объектом (это так называемые альтернативные функции ввода-вывода). При этом необходимо определить, за каким устройством (портом или другим периферийным модулем) будет закреплен каждый конкретный контакт микросхемы МК. Такая настройка проводится записью в регистр системы управления специального кода, определяющего положение коммутатора сигналов (см. рис. 1-8). В зависимости от положения контакта коммутатора, каждый внешний контакт микросхемы МК будет физически подсоединен либо к линиям ввода-вывода регистра порта, либо к линиям ввода-вывода иного периферийного устройства.

Расширенные протоколы обмена требуются для организации надежной передачи данных между МК и объектом. В простейшем случае, при выполнении команды пересылки из процессора МК в регистр данных порта, после установки на внутренней шине информации, подлежащей записи, контроллер шины формирует сигналы WR и CS, после чего данные в виде совокупности двоичных сигналов будут занесены в регистр, выведены на контакты МК и восприняты объектом. Факт записи в порт, а, следовательно, и изменения выходных данных не индицируется на внешних контактах, в связи с чем объекту неизвестен момент времени обновления информации. Так как вновь записываемые данные заменяют собой ранее присутствовавшие на контактах МК неодновременно по всем разрядам (в связи с разным вре-

менем протекания переходных процессов, обусловленным различной токовой нагрузкой контактов, гонками в комбинационных схемах и др.), может возникнуть ситуация чтения объектом ложной информации: частично измененных данных.

В связи с тем, что факт прочтения объектом передаваемой ему информации недоступен для МК, возможна ситуация, при которой частота обновления информации МК превышает частоту восприятия этой информации объектом, что приводит к потерям в посылках.

При чтении информации из объекта возможны ситуации недостоверного чтения и пропуска посылок, аналогичные описанным выше.

Для устранения подобных коллизий в структуру порта вводятся две линии внешних служебных сигналов обмена: момент завершения обновления новых данных источником (позволяющая исключить чтение ложной информации приемником в момент переключения выводимых данных) и факт прочтения данных приемником (позволяющая сбалансировать частоту ввода информации приемником и частоту вывода информации источником). Обобщенный протокол обмена показан на рис.1-9.

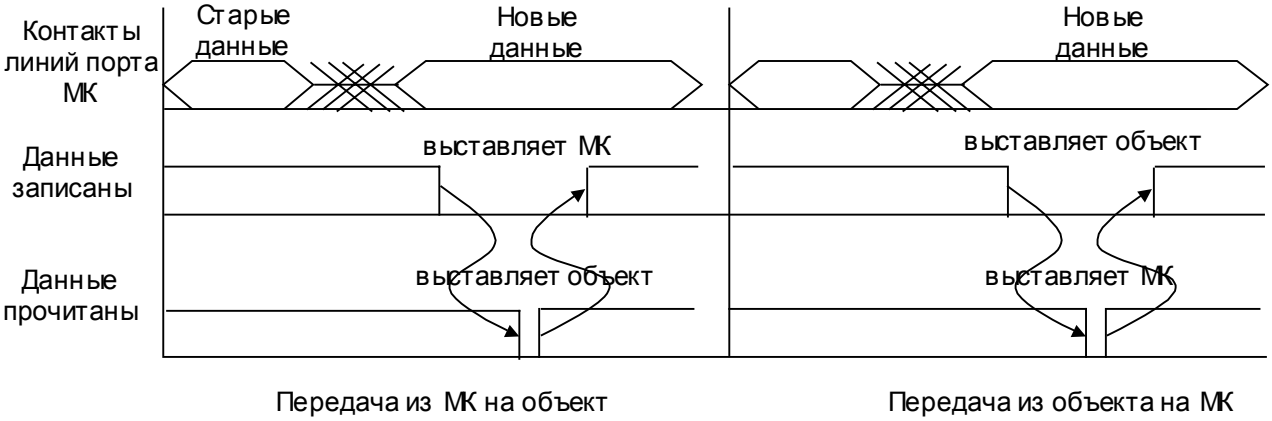


Рис. 1-9. Временные диаграммы сигналов при обмене по расширенному протоколу.

Коммутация (с необходимой выдержкой времени) сигналов WR, RD и CS между внутренней шиной МК и внешними контактами в этом случае также осуществляется системой управления.

При передаче прикладной программой данных из МК на объект, сигнал WR, зафиксировавший данные в регистре ввода-вывода, после выдержки паузы, необходимой для завершения типовых по длительности переходных процессов на контактах МК (указывается в паспортных данных на микросхему МК; там же указывается значение максимальной емкости нагрузки, превышение которой может затянуть пе-

реходные процессы и нарушить обмен по рассматриваемому протоколу), коммутируется на линию "Данные записаны", после чего объект воспринимает корректные данные и формирует сигнал подтверждения чтения "Данные прочитаны". Сигнал от объекта "Данные прочитаны" запоминается в регистре системы управления, и может быть проанализирован прикладной программой для определения возможности передачи следующей порции N-разрядных данных на объект.

При передаче данных из объекта на МК сигнал "Данные записаны", формируемый объектом, коммутируется системой управления на вход CS регистра данных; системой управления формируется и сигнал WR. Кроме того, сигнал "Данные записаны" запоминается в регистре системы управления. Прикладная программа перед чтением регистра данных порта опрашивает регистр системы управления на предмет наличия новых данных от объекта, и после обнаружения этого факта осуществляет чтение регистра данных, в ходе которого контроллер шины формирует сигналы RD и CS. Система управления коммутирует сигнал RD на линию "Данные прочитаны", сообщая объекту о готовности МК к приему следующей порции данных.

Расширенные протоколы обмена применяются, в частности, в МК Scenix.

Обобщенная структура разряда выходного каскада показана на рис. 1-10.

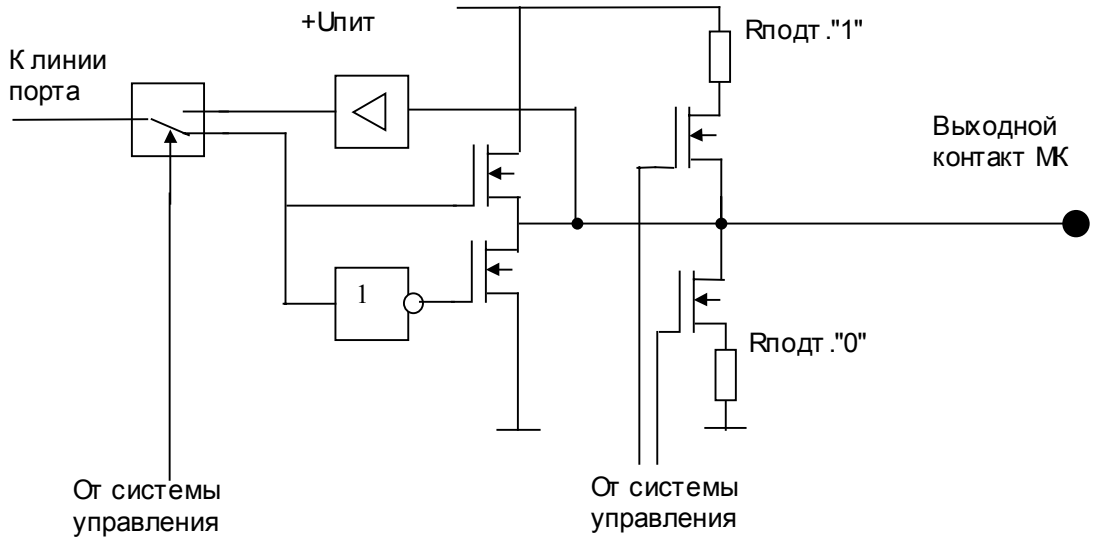


Рис. 1-10. Схемотехника выходного каскада порта.

Парафазный информационный сигнал управляет выходными транзисторами, обеспечивающими необходимую мощность сигнала. Для работы с выходами объекта типа "открытый коллектор" или "открытый эмиттер" по сигналам программно настроенной системы управления коммутируются подтягивающие резисторы R"1" или R"0". Направление обмена (ввод или вывод) по линии также задается программно путем настройки системы управления портом.

1.5. Таймеры-счетчики.

Таймеры-счетчики предназначаются для формирования временных интервалов и подсчета событий, что позволяет (при использовании соответствующего программного обеспечения) реализовывать на их основе любые функции времени, в том числе управление в реальном времени (т.е., во временном масштабе объекта).

Обобщенная структура таймера-счетчика показана на рис. 1-11.

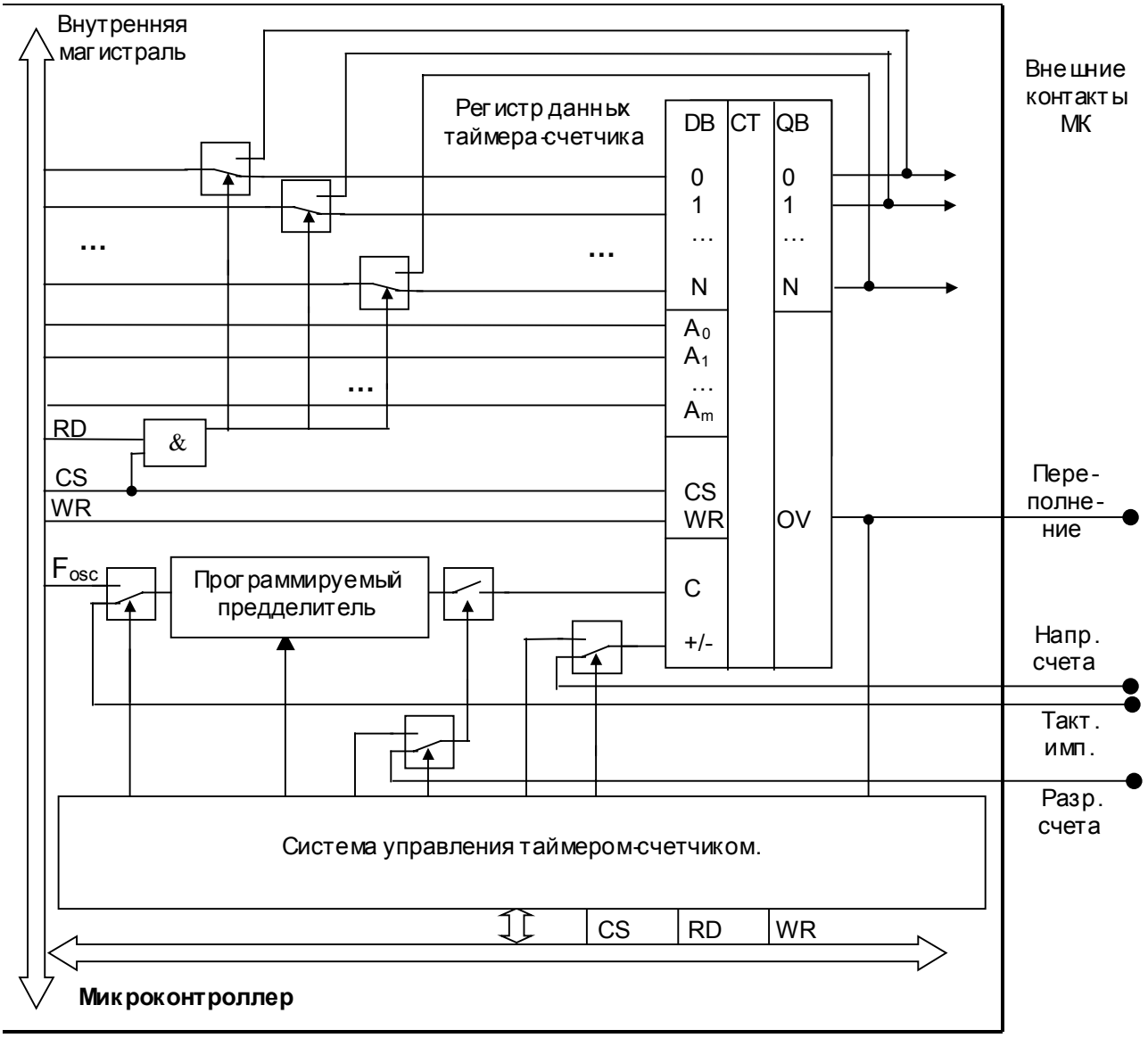


Рис. 1-11. Структура таймера-счетчика.

Таймер-счетчик базируется на синхронном двоичном реверсивном счетчике с возможностью параллельной загрузки и чтения информации. Так как разрядность счетчика МК, как правило, превышает разрядность процессора, используется адресный обмен информацией между ядром МК и частью регистра данных счетчика.

В указанной структуре при одновременной активизации сигналов RD и CS на внутреннюю шину МК коммутируется выходная шина таймера-счетчика, в противном случае внутренняя шина МК коммутируется к входам параллельной загрузки. Собственно загрузка данных в таймер-счетчик произойдет при одновременной активизации сигналов WR и CS.

Система управления позволяет определить источник задания направления счета (программно доступный бит регистра системы управления или внешний сигнал), а также источник тактирования (опорная частота ядра или внешние импульсы) и коэффициент деления тактовых импульсов. Полный диапазон счета – от 0 до $2^{(N+1)*(m+1)}$. Для задания временных интервалов, отличных от номинального (являющегося максимальным), используется предварительная программная загрузка счетчика (величиной N=T/Tкванта при счете на убывание, или Nmax-N при счете на возрастание).

При достижении кода 0 (при счете на убывание) или Nmax (при счете на возрастание), генерируется сигнал переполнения OV, доступный как для программных средств (при чтении регистра состояния системы управления таймером-счетчиком), так и для внешней аппаратуры, в частности, для объекта.

Следует отметить, что решение задачи выдержки временных интервалов возможно применением чисто программных средств: зная время t выполнения некоторой инструкции, несложно организовать ее N-кратное выполнение в цикле, тем самым обеспечив задержку длительностью T=Nt, однако, в этом случае невозможно выполнение иных программных действий (например, сбора информации, обмена данными с оператором и пр.); кроме того, неудовлетворительно велик может оказаться квант t.

Таймеры-счетчики реализованы практически во всех моделях МК.

1.6. Блоки обработки событий.

Для расширения возможностей реализации функций времени, достигаемых с применением таймеров-счетчиков, в состав периферийных устройств некоторых моделей МК введены блоки аппаратной обработки временных событий, связанных с таймерами-счетчиками. Название таких блоков в МК различных фирм может отличаться (часто применяемые названия: EPA (Event Processors Array – массив процессоров событий), HSIO (High Speed Input/Output unit – блок быстрого ввода-

вывода), PCA (Programmable Counters Array – массив программируемых счетчиков) и др.).

Основными задачами таких блоков являются:

- определение момента наступления события заданного вида (захват события). Принципиальной трудностью программной реализации захвата является невозможность точного чтения показаний регистра данных таймера без его останова (т.к. разрядность таймера превышает разрядность внутренней шины, и процедура последовательного чтения фрагментов регистра неизбежно будет проведена с искажениями);

- генерация события заданного вида в заданный момент времени (привязка события ко времени). Принципиальной трудностью программной реализации привязки является существование временной задержки между моментом обнаружения заданного момента времени (например, по переполнению регистра данных таймера) и моментом исполнения команды генерации заданного логического события;

- поддержка многоканальности указанных видов на общей временной базе. Принципиальной трудностью программной реализации многоканальности является сложность и неэффективность наблюдения за показаниями таймера со стороны нескольких программных модулей.

Особенно существенными отмеченные трудности являются при малых интервалах времени между событиями, программная обработка которых вносит значительные погрешности в измеряемые и генерируемые интервалы.

Структура канала блока обработки событий показана на рис. 1-12.

В качестве общей временной базы блока обработки выбран один из таймеров-счетчиков МК, шина QB которого, отражающая показания времени, поступает на все каналы модуля. Источник/приемник обрабатываемых/генерируемых событий является альтернативной функцией порта ввода-вывода МК.

Путем записи в регистр системы управления для каждого канала (как правило, независимо от остальных) выбирается режим его функционирования.

В случае реализации захвата путем записи в регистр системы управления определяется тип обнаруживаемого входного события – фронт, срез или перепад сигнала, что приводит к коммутации входного сигнала на один из входов синхронизации триггера детектора событий. Кроме того, в режиме захвата система управления модулем скоммутирует на разряды входной шины регистра канала одноименные разряды выходной шины таймера. При обнаружении события заданного типа систе-

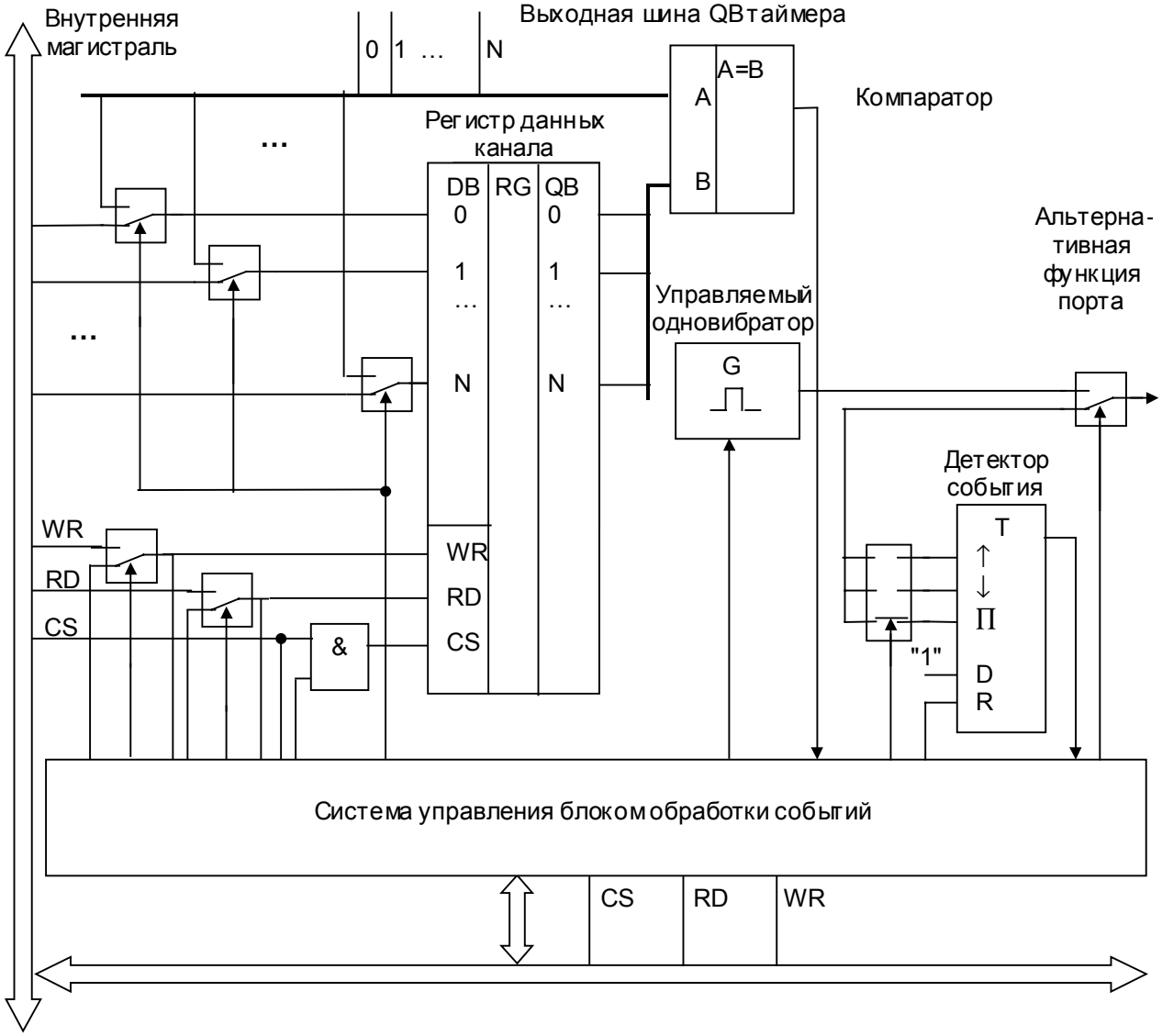


Рис. 1-12. Структура канала блока обработки событий.

ма управления сформирует сигналы CS и WR регистра данных канала, что приведет к аппаратной фиксации показаний таймера в регистре данных канала. Таким образом, в регистр данных канала аппаратно (т.е. с минимальными временными задержками) будут занесены и доступны прикладной программе показания таймера базы в момент появления заданного сигнала на входе микросхемы МК.

В случае реализации привязки путем записи в регистр системы управления определяется тип выходного события, что приводит к соответствующей настройке управляемого одновибратора. Кроме того, в режиме захвата система управления модулем скоммутирует на разряды входной шины регистра канала одноименные разряды внутренней шины МК. После записи со стороны прикладной программы значений в регистр данных канала, начнется ожидание требуемого момента времени. При совпадении показаний таймера базы и содержимого регистра данных канала

компаратор сформирует сигнал, сигнализирующий системе управления о необходимости перезапуска одновибратора. Таким образом, при достижении заданного момента времени аппаратно (т.е. с минимальными временными задержками) на выходе микросхемы МК будет сгенерирован сигнал заданного вида.

Следует отметить, что средствами блока обработки событий могут быть реализованы и дополнительные режимы: например, перезагрузка регистра данных таймера базы по достижении им заданного кода, чтение показаний регистра данных таймера базы без останова таймера (т.е. захват не внешнего, а программного события), а также генерация широтно-импульсно модулированных (ШИМ) сигналов.

Рассмотрим более подробно последний режим.

ШИМ-сигнал представляет собой периодический двоичный сигнал с фиксированным периодом T и с управляемой длительностью импульса. Канал, настроенный в данный режим, при совпадении показаний регистра данных таймера базы и регистра данных канала (хранящего значение константы ШИМ), формирует начало импульса, а по достижении счетчиком максимального значения завершает импульс и начинает новый период ШИМ (рис. 1-13).

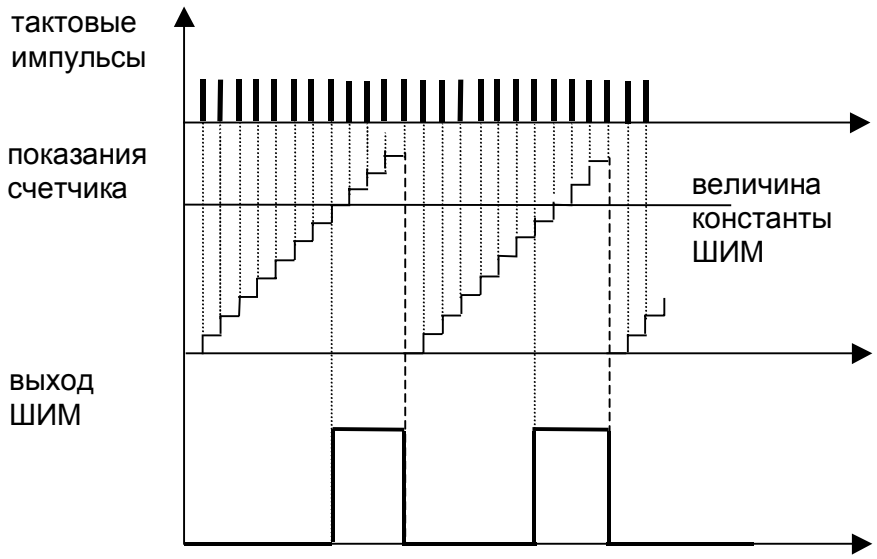


Рис. 1-13. Генерация ШИМ-сигналов.

Так как инерционность объекта существенно превышает инерционность системы управления объектом, генератор высокочастотного ШИМ воспринимается объектом как генератор, выдающий мощность, пропорциональную длительности импульсов ШИМ.

Блоки обработки событий реализованы в МК Infineon, Intel, Motorola и др.

1.7. Цифро-аналоговые преобразователи.

Цифро-аналоговые преобразователи (ЦАП) служат для перевода кодового представления выводимой из цифровой вычислительной системы информации в эквивалентный этому коду аналоговый сигнал (выходное напряжение).

Принцип цифро-аналогового преобразования иллюстрируется выражением: $U_{ВЫХ}=k*(U_{ОП.\ МАКС}-U_{ОП.МИН.})*(a_{N-1}*2^{N-1}+ a_{N-2}*2^{N-2} +\ldots+ a_0*2^0)$. Здесь a_{N-1} a_{N-2} … a_0 – двоичный код, поступающий на ЦАП, $U_{ОП.\ МАКС}$ и $U_{ОП.МИН.}$ – опорные напряжения ЦАП (максимальное и минимальное соответственно), k – коэффициент пропорциональности, $U_{ВЫХ}$ –выходное напряжение ЦАП. Так, при k=1 и входном коде ЦАП, равном половине максимального, $U_{ВЫХ}=(U_{ОП.\ МАКС}-U_{ОП.МИН.})/2$.

При построении ЦАП могут быть использованы различные методы. В МК подавляющее большинство ЦАП строится на основе резистивной сетки "R-2R". Структура ЦАП такого типа показана на рис. 1-14.

Конфигурируя блок ЦАП посредством записи настроечных команд в регистр системы управления, пользователь может выбрать источник опорных напряжений – внешний (являющийся альтернативной функцией линии некоторого порта) или внутренний. Источник внутреннего опорного напряжения допускает конфигурирование посредством выбора значения стабилизированного напряжения: от U_1 до U_K для $U_{ОП.МАКС.}$, от U_M до U_S для $U_{ОП.МИН.}$. При этом любое из значений $U_{ОП.МАКС.}$ превышает любое из значений $U_{ОП.МИН.}$ Имеется возможность управлять коммутацией сформированного значения $U_{ВЫХ}$ на выходную линию порта. На выходе ЦАП имеется усилитель-повторитель, решающий вопросы согласования с объектом управления. Выходные разряды регистра данных ЦАП управляют ключами, осуществляющими выбор слагаемого для данного разряда: $U_{ОП.\ МАКС}$ при единичном значении разряда, $U_{ОП.МИН.}$ – при нулевом.

Рассмотрим работу ЦАП на следующих примерах:

Пусть код a_{N-1} a_{N-2} … a_0 равен 100…0. В этом случае старшим разрядом будет скоммутирована величина $U_{ОП.\ МАКС}$, а остальными – величина $U_{ОП.МИН.}$. Упрощая конфигурацию резистивной матрицы для разрядов N-2…0, имеем: 2R//(R+2R//(R…R+2R//2R)…), что эквивалентно величине R. Таким образом, $U_{ВЫХ}=(U_{ОП.\ МАКС}-U_{ОП.МИН.})*(R+R)/(2R+R+R)$, т.е. $U_{ВЫХ}=(U_{ОП.\ МАКС}-U_{ОП.МИН.})/2$.

Пусть код a_{N-1} a_{N-2} … a_0 равен 0100…0. В этом случае разрядом N-2 будет скоммутирована величина $U_{ОП.\ МАКС}$, а остальными – $U_{ОП.МИН.}$. Упрощая конфигурацию резистивной матрицы для разрядов N-1,N-3…0, имеем:

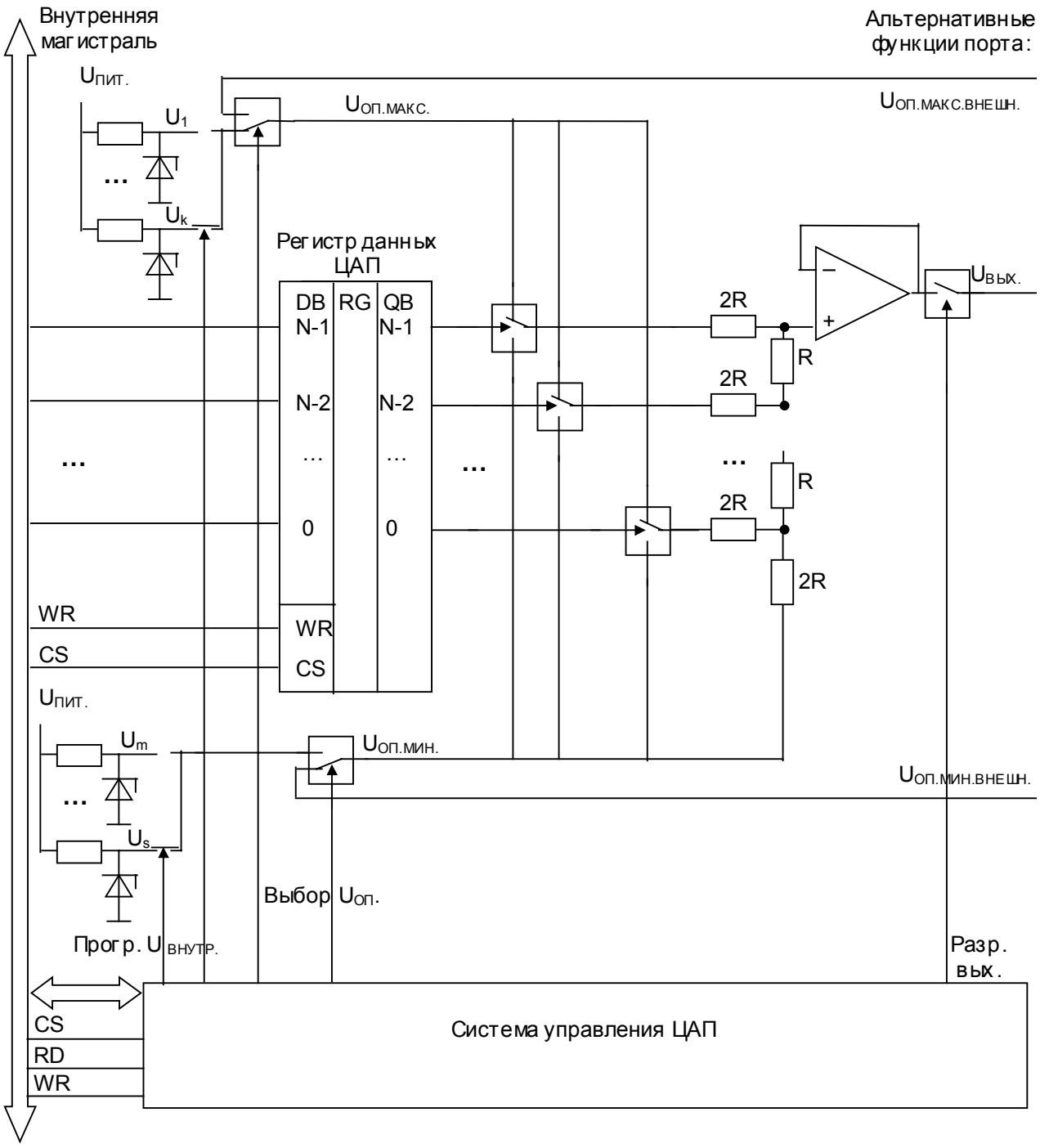


Рис. 1-14. Структура ЦАП.

(2R+R)//(R+2R//(R+2R//(R…R+2R//2R)…)), что эквивалентно величине 6R/5. Т.е., $U_{ВЫХ}=((U_{ОП.МАКС}-U_{ОП.МИН.})*(6R/5)/(2R+6R/5))*2R/3R$, т.е. $U_{ВЫХ}=(U_{ОП.\ МАКС}-U_{ОП.МИН.})/4$.

Аналогичным образом иллюстрируется результат преобразования любого кода "1 из N"; по принципу суперпозиции показывается работа ЦАП с любым двоичным кодом на входе.

Блоки ЦАП используются в контроллерах фирм Fujitsu, Cygnal и некоторых других.

1.8. Аналоговые компараторы.

Аналоговые компараторы (АК) служат для сравнения значений двух входных напряжений U_1 и U_2. Результатом сравнения является битовый результат $y=(U_1>U_2)$, истинный при $U_1 > U_2$ и ложный в противном случае.

Принцип работы АК основан на использовании дифференциального усилителя без обратной связи в ключевом режиме. Структура АК показана на рис. 1-15.

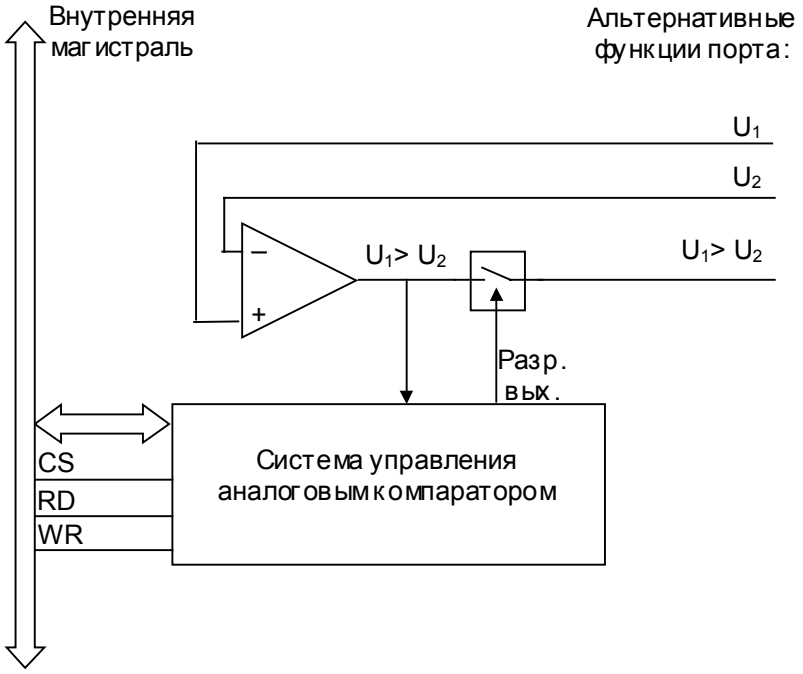


Рис. 1-15. Структура аналогового компаратора.

Данное периферийное устройство использует внешние линии МК для ввода/вывода данных, как правило, задействуя для этого разряды некоторого порта в режиме альтернативных функций.

Результат сравнения доступен в виде флага признака в регистре данных системы управления и, кроме того, может быть скоммутирован на выходную линию МК для непосредственной передачи на объект результата сравнения сигналов.

Помимо собственно задачи сравнения двух аналоговых величин, АК можно использовать в качестве простейших АЦП с ограниченными возможностями, если в качестве одного из U_{BX} использовать генератор с известной характеристикой (например, генератор линейно нарастающего напряжения, реализованный на внешнем счетчике, тактированием которого управляет МК, или RC-цепь с известной постоянной времени, заряжаемой/разряжаемой до уровня лог. "1".).

Блоки АК используются в МК фирм Atmel, Scenix и др.

1.9. Аналого-цифровые преобразователи.

Аналого-цифровые преобразователи (АЦП) служат для ввода информации, представленной аналоговым сигналом, в цифровую вычислительную систему.

Принцип аналого-цифрового преобразования заключается в разбиении полного диапазона значений аналогового сигнала на N поддиапазонов и сопоставлении входному сигналу кода уровня, покрывающего собой этот сигнал (см. рис. 1-16).

Рис. 1-16. Принцип аналого-цифрового преобразования.

При построении АЦП могут быть использованы различные методы. В МК подавляющее большинство АЦП строится на основе метода последовательного приближения с применением ЦАП и взвешивающего регистра сдвигов (рис. 1-17).

Функционирование такого АЦП осуществляется следующим образом: запуск предваряется инициализацией (путем подачи импульса "Reset"), по окончании которой во всех Т-триггерах и взвешивающем сдвиговом регистре находятся нули. Сразу по окончании инициализации подается сигнал "Пуск", заносящий уровень логической "1" в триггер старшего разряда. С приходом импульса опорной последовательности Fтакт в старший разряд сдвигового регистра также поступит уровень логической "1", после чего сигнал "Пуск" снимается. На ЦАП поступает цифровой код Nmax/2, приводящий к установке Uоп=(Uоп.макс.- Uоп.мин.)/2. Полученный уровень сравнивается аналоговым компаратором с входной измеряемой величиной Uизм; в случае Uоп>Uизм на выходе компаратора устанавливается уровень логической "1". В этом случае вентиль "И" будет открыт, что приведет к сбросу триггера старшего

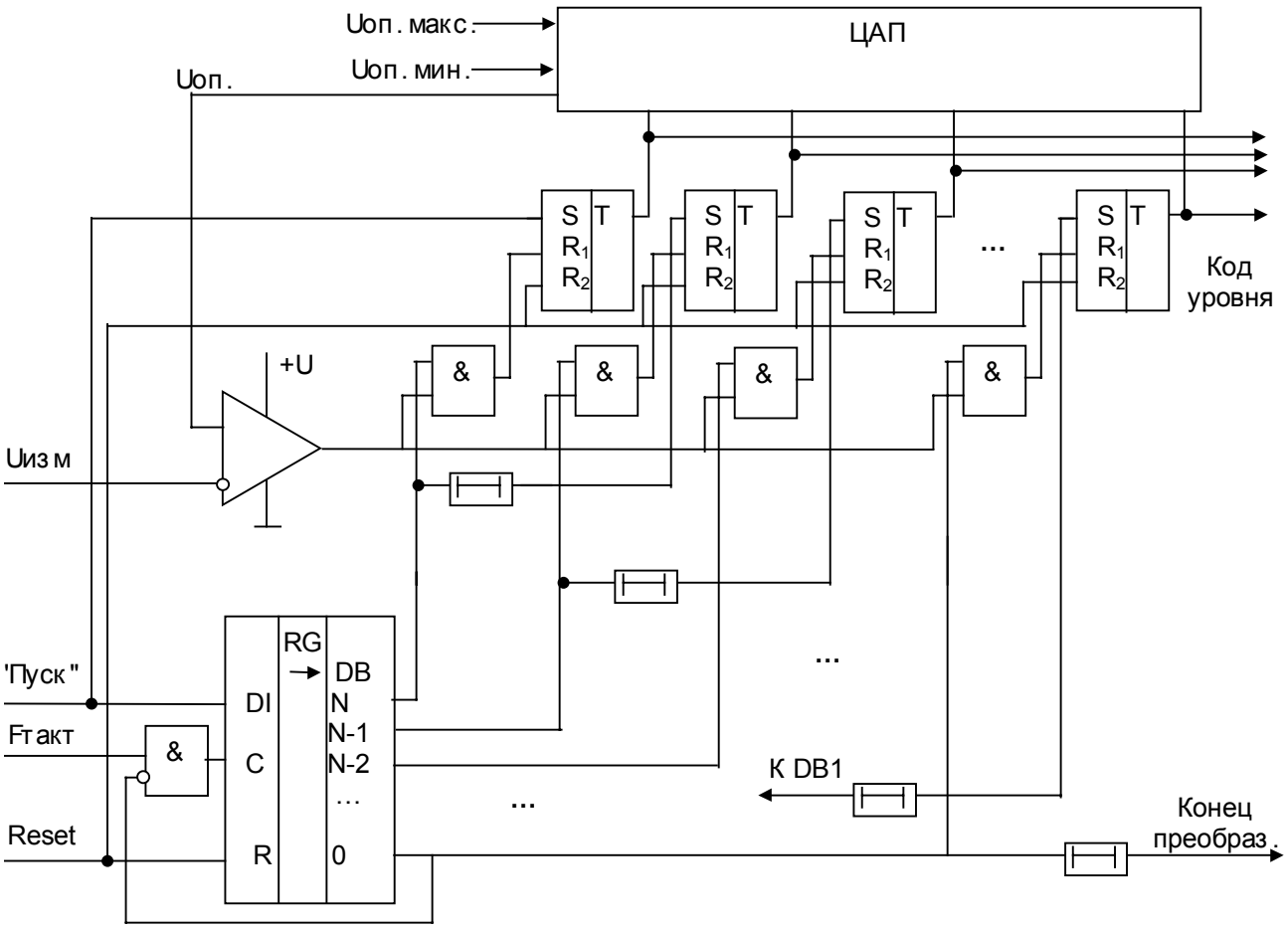


Рис. 1-17. Структура блока АЦП последовательного приближения.

разряда (т.е. отказу от включения в выходной код старшего разряда, равного "1", определяющего наличие в Uизм слагаемого (Uоп.макс.- Uоп.мин.)/2). Далее будет установлен следующий триггер и произведена проверка наличия слагаемого (Uоп.макс.- Uоп.мин.)/4. Если Uизм<(Uоп.макс.- Uоп.мин.)/4, то на выходе аналогового компаратора устанавливается уровень логического "0", соответствующий вентиль "И" после сдвига в регистре не откроется, и указанный разряд сохранится в выходном коде. Процесс повторяется до момента перехода бегущей единицы в младший разряд сдвигового регистра, после чего генерируется сигнал "Конец преобразования", сдвиги во взвешивающем регистре прекращаются, а на выходах "Код уровня" находится соответствующий величине Uизм цифровой двоичный код $a_{N...}a_0$, определяющий эталонные весовые напряжения в соответствии с формулой $Uизм=\Sigma a_i * Uоп_i$, где i=N..0. Время преобразования АЦП постоянно и определяется числом разрядов и частотой сдвигов; погрешность определятся погрешностью ЦАП и чувствительностью компаратора.

Структура блока АЦП МК показана на рис. 1-18.

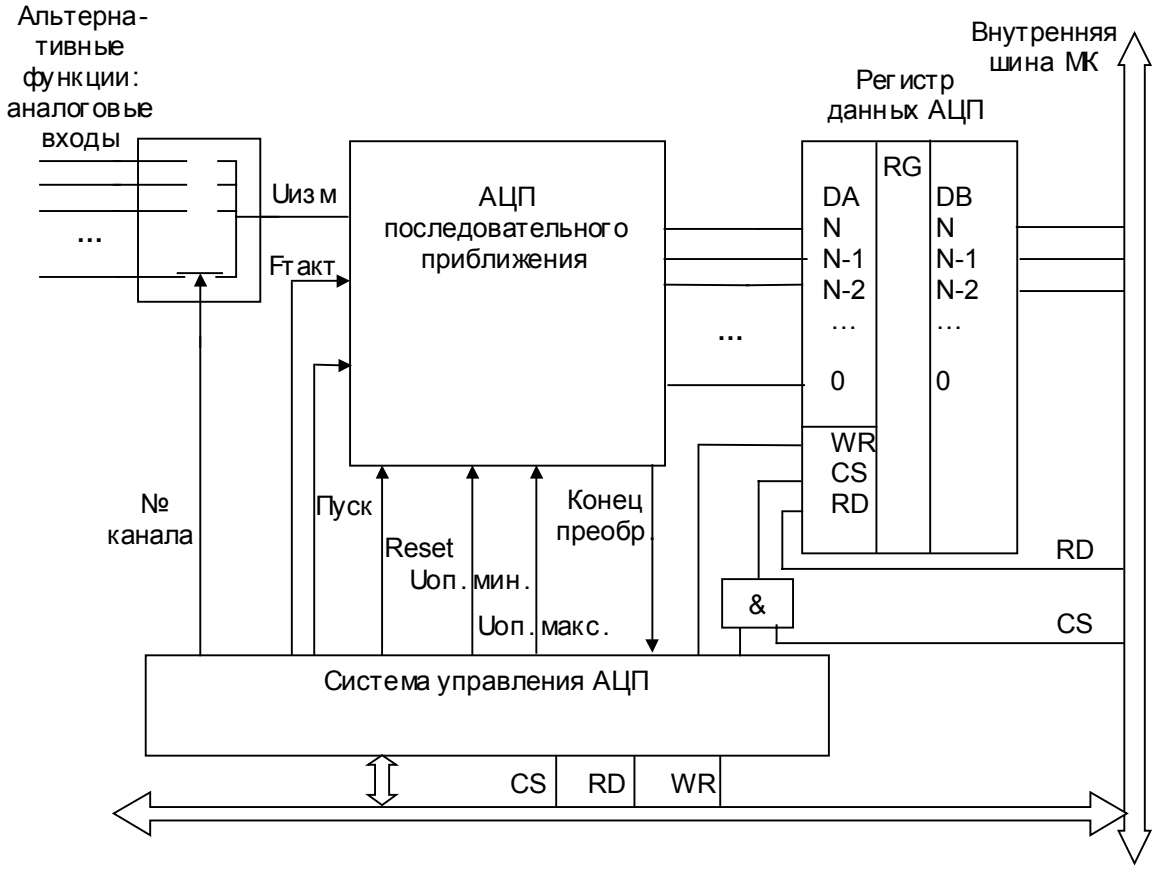


Рис. 1-18. Структура блока АЦП МК.

Настройки системы управления блоком АЦП позволяют осуществить выбор канала аналогового входа (являющегося альтернативной функцией входа некоторого порта), задать диапазон измеряемых величин и определить момент начала преобразования. После активизации флага готовности прикладная программа может осуществить чтение регистра данных АЦП.

Вес единицы младшего разряда определяется по формуле:

$\delta=(U_{оп.макс.}-U_{оп.мин})/(2^{N+1})$, в связи с чем для повышения точности (при неизменной разрядности) следует уменьшить предполагаемый диапазон измеряемых сигналов. Для этого в прикладной программе предусматривают два цикла измерений: в полном диапазоне входных сигналов для определения величин опорных напряжений, а затем – в локализованном – для получения точного результата.

Блоки АЦП присутствуют в МК фирм Infineon, Intel, Motorola, Cygnal и др.

1.10. Средства поддержки межпроцессорного обмена.

Средства поддержки межпроцессорного обмена применяются для построения распределенных систем. В связи с тем, что практически всегда для построения таких систем используется последовательная передача данных, средства поддержки межпроцессорного обмена часто именуются последовательными портами ввода-вывода. К причинам выбора последовательного метода передачи данных относятся следующие: необходимость обеспечения надежной передачи данных при существенной территориальной протяженности канала передачи, необходимость снижения затрат на кабельное хозяйство с учетом относительно невысокой интенсивности обмена, а также необходимость обеспечения передачи данных по различным протоколам на основе единой аппаратной реализации.

К основным причинам использования распределенных систем относятся:

- территориальная распределенность объекта управления (в этом случае эффективным является решения построения управляющей вычислительной сети из нескольких МК, решающих задачи локального управления, и обменивающихся информацией о характере этого управления для сохранения единства целей);

- иерархичность системы управления (в случае использования человеко-машинных систем управления необходимо осуществлять передачу от операторской станции к МК (или к сети МК) информации об установках оператора, а в обратном направлении – о показаниях датчиков, параметрах управления в МК и др.);

- потребность в отладке системы на базе МК с применением инструментальных средств (в основном, инструментальных ЭВМ) в ходе реализации целевой функции МК (в этом случае необходимо реализовать возможность доступа к ресурсам МК, не блокируя штатные средства ввода-вывода МК, занятые для реализации процесса взаимодействия с объектом).

Номенклатура средств межпроцессорного обмена достаточно широка. К универсальным средствам относятся интерфейсы UART (Universal Asynchronous Receiver and Transmitter – универсальный асинхронный приемопередатчик), SSI (Synchronous Serial Interface – синхронный последовательный интерфейс) и его разновидность SPI (Serial Peripherial Interface –последовательный периферийный интерфейс). К средствам поддержки обмена в управляющих вычислительных сетях относятся I^2C (Inter-Integrated Circuit – межсхемный интерфейс) и CAN (Contollers Area Network – сеть контроллеров). К средствам поддержки отладки относятся интерфей-

сы TAP (Test Access Port – порт для тестового доступа) и OnCE (On Crystal Emulator – внутрикристальный эмулятор).

Структура блока межпроцессорного обмена показана на рис. 1-19.

Блок содержит сдвиговые регистры передатчика и приемника, позволяющие преобразовывать данные из последовательного представления в параллельное. Оба регистра имеют входы сброса R и разрешения работы E, а регистр передатчика – вход разрешения параллельной загрузки. Входы синхронизации C сдвиговых регистров являются независимыми и могут принимать такты сдвига как от внешнего синхросигнала, так и от внутреннего источника, использующего счетчик/делитель частоты Fosc с программируемым коэффициентом деления (в ряде случаев в качестве такого счетчика может использоваться таймер общего назначе-

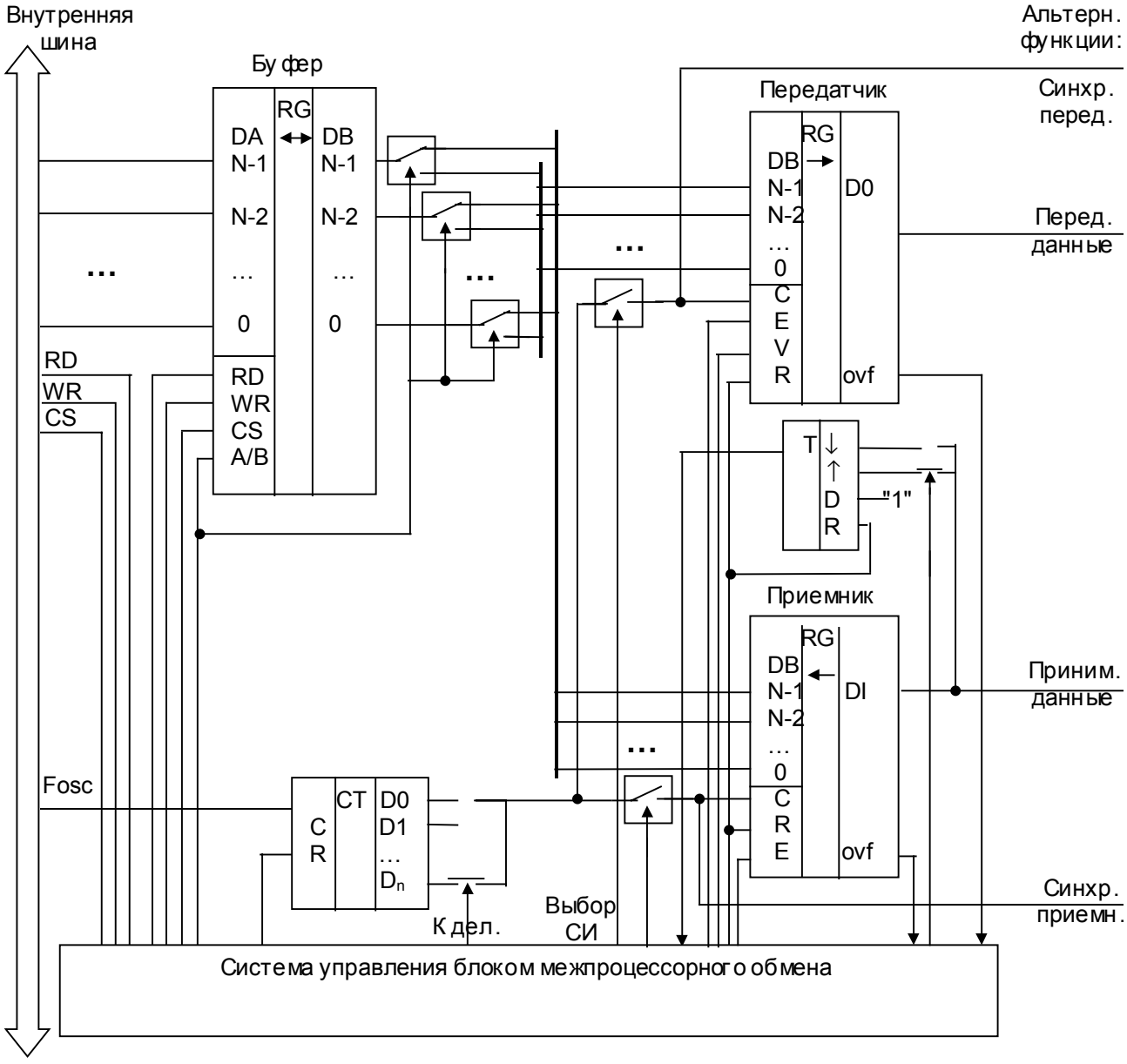


Рис. 1-19. Структура блока межпроцессорного обмена.

ния). Синхросигналы в общем случае поступают на внешнее устройство.

И передатчик, и приемник связаны с буфером, доступным для записи и чтения программой МК. При записи в буфер его содержимое дополнительно копируется в передатчик, после чего побитно выводится через выходную линию. Окончание накопления (флаг ovf) регистрируется в системе управления и может быть обнаружено программно. При попытке чтения буфера в него копируется содержимое приемника, после чего становится доступным на внутренней шине. (Следует отметить, что для достоверного чтения посылки прежде необходимо проконтролировать наличие флага ovf от приемника). Начало посылки обнаруживается триггером.

Рассмотрим более подробно конкретные реализации блока.

В интерфейсе UART используются две линии: вход приемника RxD и выход передатчика TxD, что позволяет организовать полнодуплексную (одновременную двустороннюю) асинхронную связь. Обмен между абонентами ведется на частоте, задаваемой только внутренним источником, и устанавливаемой заранее. Для повышения надежности передачи посылка включает старт-бит, 7 или 8 информационных разрядов, бит контрольной суммы и стоп-бит. Старт-бит (переход из "1" в "0") активизирует бит "E" приемника, после чего он фиксирует посылку. Реализуемая топология в многоабонентных системах (управляющих сетях) – "звезда".

В SSI используются линии передаваемых и принимаемых данных, каждая стробируемая своим синхросигналом. Как разновидность SSI наиболее часто применяется SPI с тремя линиями: выход передатчика MOSI, вход приемника MISO и общая линия синхронизации SCK, что позволяет организовать полнодуплексную связь с частотой, задаваемой ведущим абонентом. Ведущим является устройство, синхросигнал от которого подается на SCK всех остальных. Каждое ведомое устройство выбирается с помощью отдельной дополнительной линии. Топология - "звезда".

В I^2C используется две линии: SDA (линия данных) и SCL (линия синхросигналов), что позволяет организовать полудуплексную (разновременную двустороннюю) синхронную связь. Данные передаются от передатчика к приемнику по SDA, при этом ведущее устройство управляет линией SCL. Смена ролей устройств (передатчика и приемника; ведущего и ведомого) осуществляется программно на основе анализа информации, передаваемой в посылке. Топология – "шина".

В интерфейсе CAN также используются две линии: CANH и CANL, что позволяет организовать полудуплексную асинхронную связь. Все устройства CAN-шины заранее пронумерованы. Ведущим в сеансе обмена считается устройство, чей

номер является наибольшим (процедура CSMACD-AMP – Carrier Sense Multiple Access with Collision Detection and Arbitration on Message Priority – множественный доступ с контролем несущей и обнаружением столкновений по приоритету сообщений). Применяемая топология сети – "шина".

В интерфейсе TAP используется пять линий: TMS, TRST, TCK, TDI и TDO, три последние из которых относятся к модулю последовательного обмена и являются соответственно линией синхросигналов, линией ввода данных и линией вывода данных. Через эти выводы происходит синхронизированная передача отладочных команд в МК и получение из МК данных о ходе его функционирования и его состоянии (первые две – это входной сигнал активизации модуля и входной сигнал его инициализации). В интерфейсе OnCE используется четыре линии: DR, DSCK, DSI и DSO, аналогичные по назначению линиям TMS, TCK, TDI и TDO в TAP. Оба интерфейса реализуют синхронную полудуплексную связь и используются только для отладочных целей; управляющие сети на их основе не реализуются.

1.11. Блоки обслуживания прерывающих событий.

Функционирование любой вычислительной системы (в том числе, на базе МК) как правило, предполагает обслуживание некоторого ряда событий. Место и причина их возникновения могут быть достаточно разнообразными, однако эти события обладают одним общим свойством – момент их возникновения заранее неизвестен для программы. В связи с этим, в вычислительной системе необходимо предусмотреть средства обнаружения этих событий с тем, чтобы по факту их обнаружения предпринять некоторые действия по их обслуживанию. (Так как при обслуживании события следует отложить ("прервать") выполнявшуюся программу, и выполнить обслуживающую программу, такие события называются прерывающими событиями).

Наиболее простым способом обнаружения прерывающих событий является периодическая проверка факта их возникновения программным способом, что имеет два существенных недостатка:

- снижение производительности вычислительной системы (некоторая часть программы будет отведена под перебор признаков возникновения событий);

- наличие задержки между моментом возникновения события и моментом его обнаружения (в ряде случаев - например, в системах реального времени, принципиально недопустимая).

Исключение указанных недостатков возможно лишь в случае аппаратного обнаружения заданных прерывающих событий, что реализуется посредством системы обслуживания прерывающих событий (кратко – "система прерываний"). Помимо обнаружения прерывающих событий, на систему прерываний возложен арбитраж событий для определения наиболее приоритетного из них.

Прерывающие события имеют и программную, и аппаратную природу:

- к программным причинам возникновения прерывающих событий относятся: попытка выполнения несуществующих команд (т.е. таких, чей КОП отсутствует в системе команд данного МК) или выполнение команд с недопустимыми данными (например, в случае деления на ноль);

- к аппаратным причинам возникновения прерывающих событий относятся: готовность периферийных модулей (например, окончание преобразования в АЦП, переполнение таймера, прием посылки по последовательному порту и т.п.) и активизация сигналов, поступающих на МК извне от объекта управления (например, двоичный сигнал от датчика срабатывания исполнительного механизма).

Количество прерывающих событий ограничено (так, количество внешних прерывающих событий не может превышать числа линий ввода-вывода МК), поэтому все они могут быть индивидуально обнаружены аппаратными средствами. Напротив, действия по их обслуживанию определяются целевой функцией МК и могут быть достаточно разнообразными, в связи с чем прерывающие события, как правило, обслуживаются программными средствами. Исключение составляют ситуации, связанные с пересылками данных (например, помещение очередного результата аналого-цифрового преобразования в заданную ячейку массива данных), выполняемые по заранее заданной схеме действий аппаратно в режиме прямого доступа к памяти (ПДП) (модули обслуживания такого типа называются PTS – Peripherial Transaction Servers – периферийные серверы транзакций).

Суть обработки прерывающего события сводится к следующим действиям:

- обнаружение факта прерывающего события;

- установление необходимости реакции на него (т.к. не все из потенциально возможных прерывающих событий необходимы для реализации данной целевой функции МК – например, может не требоваться прерывание от АЦП, работа с которым осуществляется путем выдержки паузы в Тпреобр.);

- выполнение арбитража событий (в случае возникновения нового прерывающего события в момент обслуживания предыдущего прерывающего события, в

зависимости от степени важности этих событий следует либо продолжить начатое обслуживание, либо прервать его, переключившись на обслуживание нового;

- выполнение переключения программного контекста (сохранение состояния прерываемой программы и переключение на программу обслуживания, адрес которой заранее однозначно сопоставлен типу прерывающего события (так называемый "вектор прерывания" или "точка входа в обработчик"));

- выполнение восстановления программного контекста (восстановление состояния прерванной программы и обратное переключение на нее).

Обобщенная схема системы прерываний показана на рис. 1-20.

Обработка прерывающих событий от любого источника может быть индивидуально запрещена ("замаскирована") записью лог. "0" в регистры масок (за исключением особо важных, так называемых "немаскируемых событий", для которых в регистре масок всегда занесена лог. "1"). Кроме того, каждому источнику может быть сопоставлен один из К уровней приоритетов записью соответствующего кода в регистры приоритетов, управляющих коммутаторами входов приоритетного шифратора. (Т.к. количество прерывающих событий может превышать разрядность МК, то для индивидуального маскирования и кодирования приоритетов может требоваться несколько регистров; на рисунке показано по одному регистру масок и приоритетов). Каждое прерывающее событие запоминается в фиксаторе события своего типа. В случае наличия нескольких зафиксированных событий благодаря использованию приоритетного шифратора на вход памяти таблицы векторов будет подан код самого старшего прерывающего события, а на выходах таблицы будет находиться адрес точки входа в обработчик прерывающего события данного типа.

Перед занесением данного адреса в указатель команд процессора ядра МК, необходимо выполнить микропрограмму сохранения контекста прерываемой программы. Для этого в модуле обработки прерывающих событий имеются регистр-указатель адреса ПДП (адресует ячейку области памяти данных), регистр данных ПДП (хранит содержимое адресуемой ячейки) и ОЗУ команд ПДП (хранит последовательность действий сохранения контекста). С их помощью аппаратно считывается содержимое регистров контекста и сохраняется в специальной области памяти данных МК. Для того чтобы по окончании обработки прерывающего события вернуться к прерванному контексту, как минимум, необходимо сохранить указатель команд (остальные ресурсы могут быть сохранены программно в обработчике события). Как правило, для удобства сохранения и восстановления контекста, используется стек.

По окончании обработки события (завершающегося, как правило, специальной командой, при выполнении которой процессор ядра посылает в систему управления обслуживания прерывающих событий специальный сигнал) заявка, хранившаяся в фиксаторе и вызвавшая обработку события, снимается (сигналом на входе "R" фиксатора).

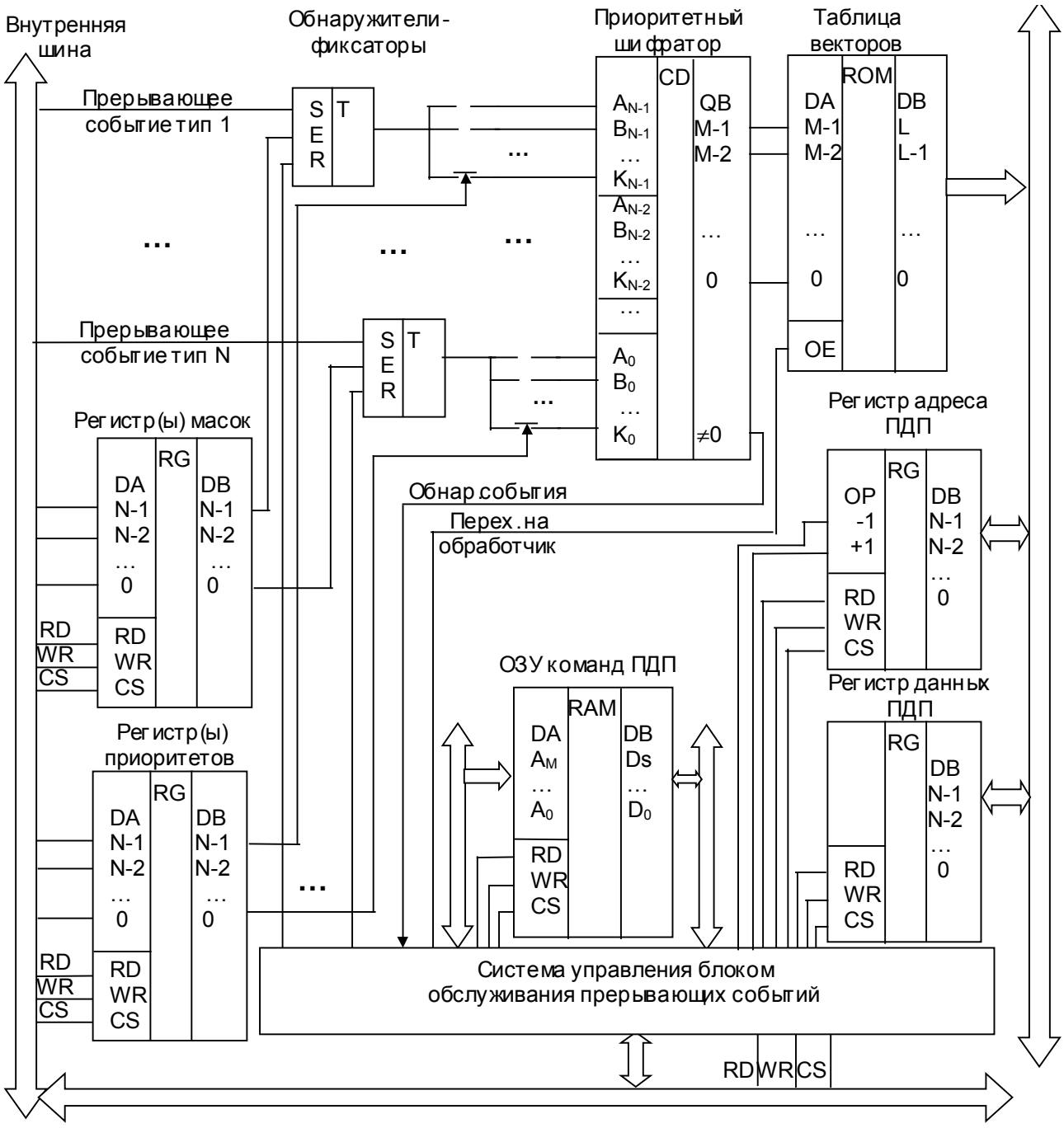


Рис. 1-20. Структура блока обслуживания прерывающих событий.

В случаях однозначной и несложной обработки события (типа пересылки данных из ячейки в ячейку), такие действия могут быть выполнены полностью аппаратно, в режиме ПДП. При подготовке этих действий и инициализации ПДП-

обслуживания прерывающих событий в ОЗУ команд ПДП основной программой записывается управляющий блок, описывающий будущую процедуру аппаратных пересылок. Приблизительный формат управляющего блока таков: "начальный адрес источника данных", "начальный адрес приемника данных", "количество повторений пересылок", "признак необходимости изменения адреса источника", "направление изменения адреса источника", "величина изменения адреса источника", "признак необходимости изменения адреса приемника ", "направление изменения адреса приемника ", "величина изменения адреса приемника". Используя эти поля, можно организовать одиночную и групповую пересылку данных по разным направлениям адресов в памяти и с различным шагом этих адресов.

Блоки обслуживания прерывающих событий есть в подавляющем большинстве МК. Блоки PTS встречаются в некоторых моделях МК фирмы Intel.

1.12. Средства повышения надежности функционирования МК.

В связи с тем, что МК-системы зачастую эксплуатируются в автономном режиме, без участия оператора, в составе МК желательно наличие средств, обнаруживающих ситуации, угрожающие нормальной работе МК.

К этим средствам относятся сторожевые таймеры и мониторы питания.

Основными причинами сбоев в работе МК-систем является помехи, вызванные нестабильностями в цепях питания: повалами, выбросами, высокочастотными помехами. Наиболее сильно влияющим на достоверность вычислительного процесса в МК из критичных к таким помехам элементов относится указатель команд: любое искажение его содержимого приведет к выполнению кодов, не соответствующих исходной программе, и, следовательно, катастрофическим последствиям для МК-системы в целом.

Характерно, что прикладная программа для МК является бесконечно повторяемым циклом, реализующим целевую функцию. Временные характеристики выполнения этого цикла (в частности, среднее и максимальное время одной итерации) легко определяемы. Если аппаратными средствами контролировать время выполнения цикла и сопоставлять его с расчетным, то факт их существенного несовпадения будет достоверно свидетельствовать о сбоях в программе. Обнаружив расхождение во временных параметрах, модуль аппаратного контроля генерирует сигнал запроса прерывания или формирует сигнал сброса МК, перезапуская програм-

му. Сопоставление временных параметров функционирования программы осуществляет устройство WDT (WatchDog Timer – сторожевой таймер). При нормальном функционировании программа периодически очищает таймер сигналом "Сброс WDT". Для надежности работы системы сигнал "Сброс WDT" в ряде МК должен формироваться в виде заданной последовательности обращений к системе управления WDT.

Монитор питания (Power Monitor) осуществляет контроль за величиной питающего напряжения и формирует сигнал "Сброс МК" в случае, если его значение стало ниже предельно допустимого порогового значения. При подаче питания сигнал "Сброс МК" с помощью таймера POR (Power-On Reset) удерживается некоторое время после установления штатного питания для надежного запуска МК.

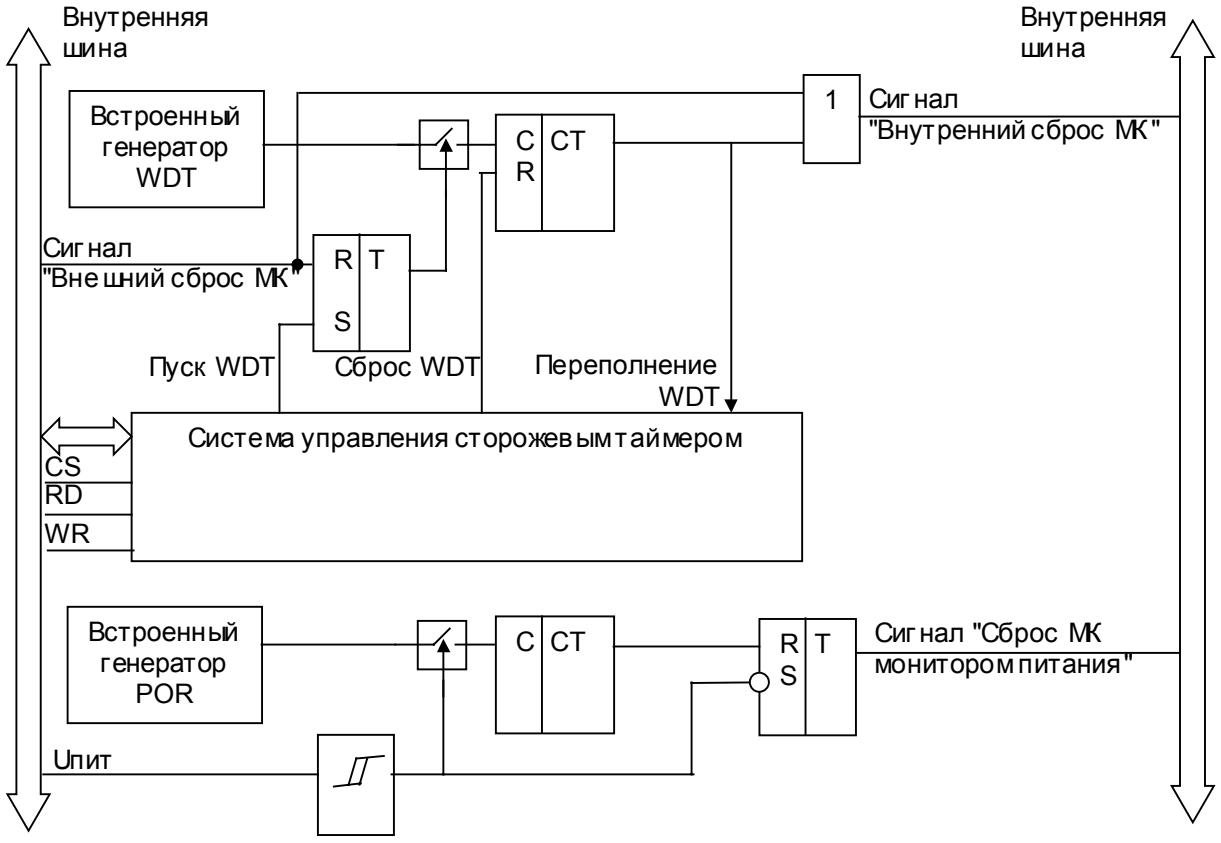


Рис. 1-21. Структура сторожевого таймера и монитора питания.

К дополнительным средствам повышения надежности относятся средства перевода МК в режимы пониженного энергопотребления, что обеспечивает продление срока автономной работы при питании от источников аккумуляторного типа. В этих режимах аппаратно отключается периферия МК, а в ряде случаев – и процессорное ядро, что приводит к значительному сокращению энергопотребления. Эти режимы эффективно применять в системах с холостым ожиданием редких событий.

РАЗДЕЛ 2. МИКРОКОНТРОЛЛЕР INFINEON 80C515.

2.1. Общее описание и цоколевка.

В данном разделе рассматривается представитель семейства микроконтроллеров MCS-51 - микроконтроллер 80C515 фирмы Infineon. Данное семейство микроконтроллеров выбрано как объект для рассмотрения в связи с многолетней значительной популярностью среди разработчиков встраиваемых приложений, что объясняется продуманностью архитектурных решений, широкой номенклатурой клонов семейства от всемирно известных производителей (Siemens, Intel, Dallas Semiconductor, Atmel и др.), а также эффективной ценовой политикой изготовителей и поставщиков.

Основными характеристиками микроконтроллера 80C515 являются:

- 8-разрядная гарвардская архитектура;
- 111 базовых команд;
- длительность машинного цикла - 1 мкс при частоте опорных импульсов 12 МГц;
- 8 КБайт внутрикристальной памяти программ с масочным программированием;
- 256 Байт внутренней памяти данных;
- возможность обращения к внешней памяти программ объемом до 64 КБайт и к внешней памяти данных объемом до 64 КБайт;
- шесть 8-разрядных портов ввода-вывода;
- 8-разрядный порт ввода, совмещенный с 8-канальным АЦП с программируемыми диапазонами измерения;
- три 16-разрядных таймера-счетчика;
- блок быстрого ввода-вывода;
- полнодуплексный последовательный порт с управляемой и фиксированной скоростями передачи;
- 12-векторный 4-уровневый внутрикристальный контроллер прерываний;
- 16-разрядный сторожевой таймер;
- режимы холостого хода и пониженного энергопотребления.

На рис. 2-1 изображена цоколевка корпуса микросхемы микроконтроллера. Совокупность контактов можно разделить на следующие группы: линии портов ввода-вывода, линии шины управления, а также линии для подключения источника опорных тактовых импульсов и цепей питания.

В таблице 2-1 рассмотрено назначение контактов микросхемы.

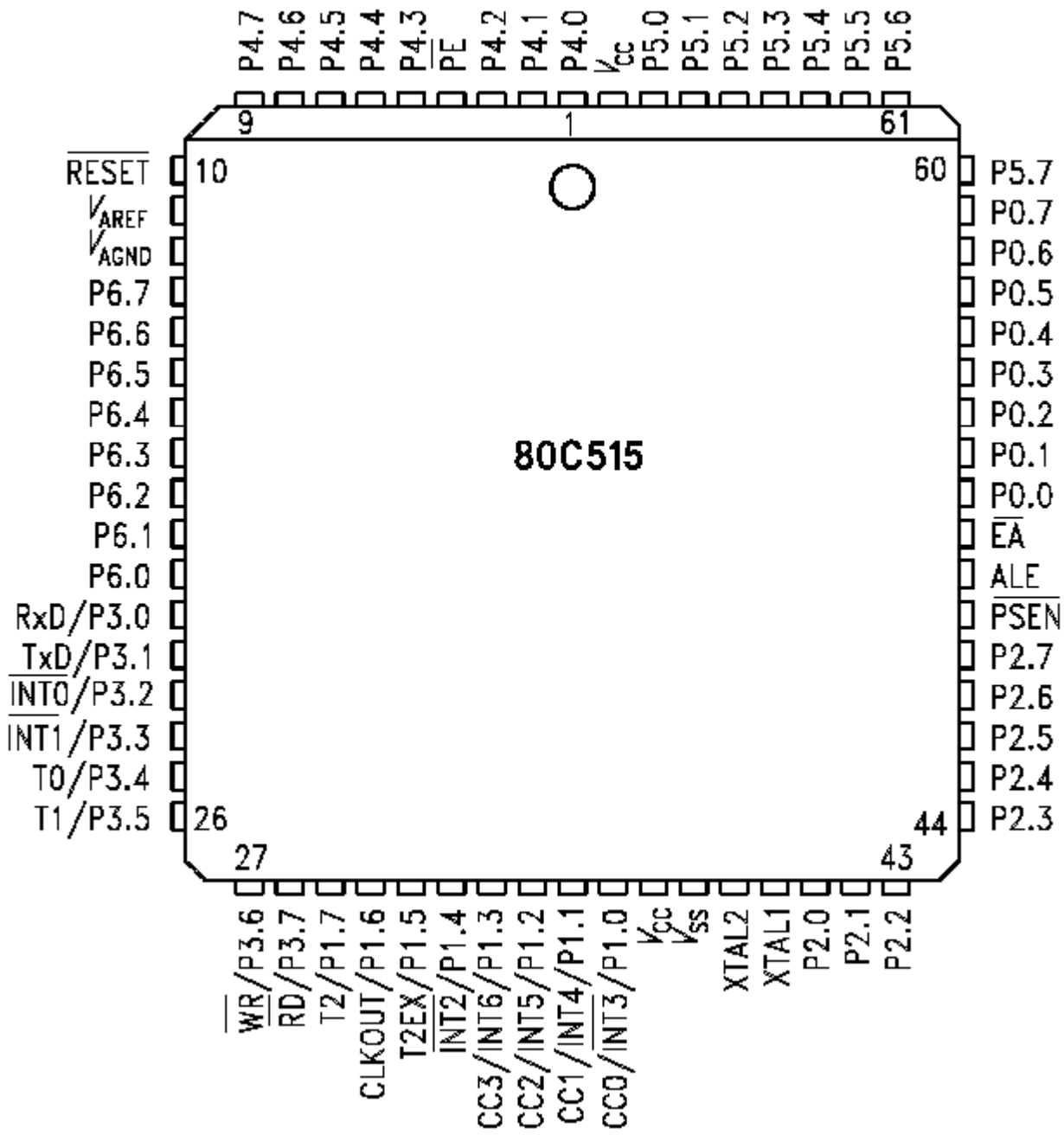


Рис. 2-1. Цоколевка корпуса микроконтроллера 80C515.

Табл. 2-1.Назначение контактов микросхемы микроконтроллера 80C515.

Обозн.	Конт.	Напр.	Назначение
P0.0-P0.7	52-59	вх/вых	8-разрядный двунаправленный параллельный порт №0 с альтернативной функцией вывода младшего байта адреса/ввода-вывода байта данных при обменах с внешней памятью.
P1.7 - P1.0	29-36	вх/вых	8-разрядный двунаправленный параллельный порт №1 с набором альтернативных функций.
P2.0-P2.7	41-48	вх/вых	8-разрядный двунаправленный параллельный порт №2 с альтернативной функцией вывода старшего байта адреса при обменах с внешней памятью.
P3.0-P3.7	21-28	вх/вых	8-разрядный двунаправленный параллельный порт №3 с набором альтернативных функций.
P4.0-P4.7	1-3, 5-9	вх/вых	8-разрядный двунаправленный параллельный порт №4.
P5.7-P5.0	60-67	вх/вых	8-разрядный двунаправленный параллельный порт №5.
P6.7-P6.0	13-20	вх	8-разрядный входной параллельный порт №6 с альтернативной функцией входов 8-канального АЦП.
Reset	10	вх	Сигнал "сброс". Низкий уровень сигнала осуществляет перезапуск микроконтроллера.
EA	51	вх	Отключение внутренней памяти. Низкий уровень переводит контроллер в режим работы с внешней памятью программ.
ALE	50	вых	Строб адреса внешней памяти. Стробирует выдачу адресной информации при работе с внешней памятью.
PSEN	49	вых	Разрешение внешней памяти программ. Стробирует чтение внешней памяти программ.
PE	4	вх	Разрешение перехода в режим пониженного энергопотребления. Низкий уровень разрешает функции программного перехода в режим пониженного энергопотребления.
XTAL1	40	–	Контакт №1 источника опорных тактовых импульсов.

Обозн.	Конт.	Напр.	Назначение
XTAL2	39	–	Контакт №2 источника опорных тактовых импульсов.
Vcc	68,37	–	Напряжение питания (5 В) - основное (контакт 68) и дополнительное (контакт 37).
Vss	38	–	Цифровая "земля" (0 В).
V_{AREF}	11	–	Опорное напряжение для АЦП.
V_{AGND}	12	–	Аналоговая "земля".

2.2. Организация памяти микроконтроллера.

Микроконтроллер может работать с внутренней или с внешней памятью, что позволяет в необходимых случаях наращивать ее объем. Переключение типа памяти осуществляется с помощью сигнала EA.

Доступ к памяти программ (и внешней, и внутренней) осуществляется посредством указателя команд PC (инициируется аппаратно при выборке команд) и указателя данных DPTR (инициируется программой при выполнении команд чтения констант из памяти программ). Внутренняя память программ занимает диапазон адресов 0000-1FFFh. Внешняя память программ занимает диапазон адресов 0000-FFFFh. При перезапуске (активизации сигнала "Reset") указателю команд присваивается значение 0000. По адресам 0003 - 0072h располагаются точки входа в обработчики прерываний (см. раздел 2-9).

Доступ к памяти данных (и внешней, и внутренней) инициируется программой при выполнении команд чтения и записи данных. Внутренняя память данных занимает диапазон адресов 0000-00FF. Внешняя память данных занимает диапазон адресов 0000-FFFFh.

Доступ к внешней памяти данных осуществляется посредством указателя данных DPTR.

Доступ к 128 младшим байтам внутренней памяти данных возможен с применением прямой и косвенной адресации; доступ к 128 старшим байтам внутренней памяти данных - с применением только косвенной адресации; доступ к 128 байтам регистров специальных функций - с применением только прямой адресации.

Номенклатура регистров специальных функций приводится в таблице 2-2.

Организация доступа к внешней памяти показана на рис. 2-2.

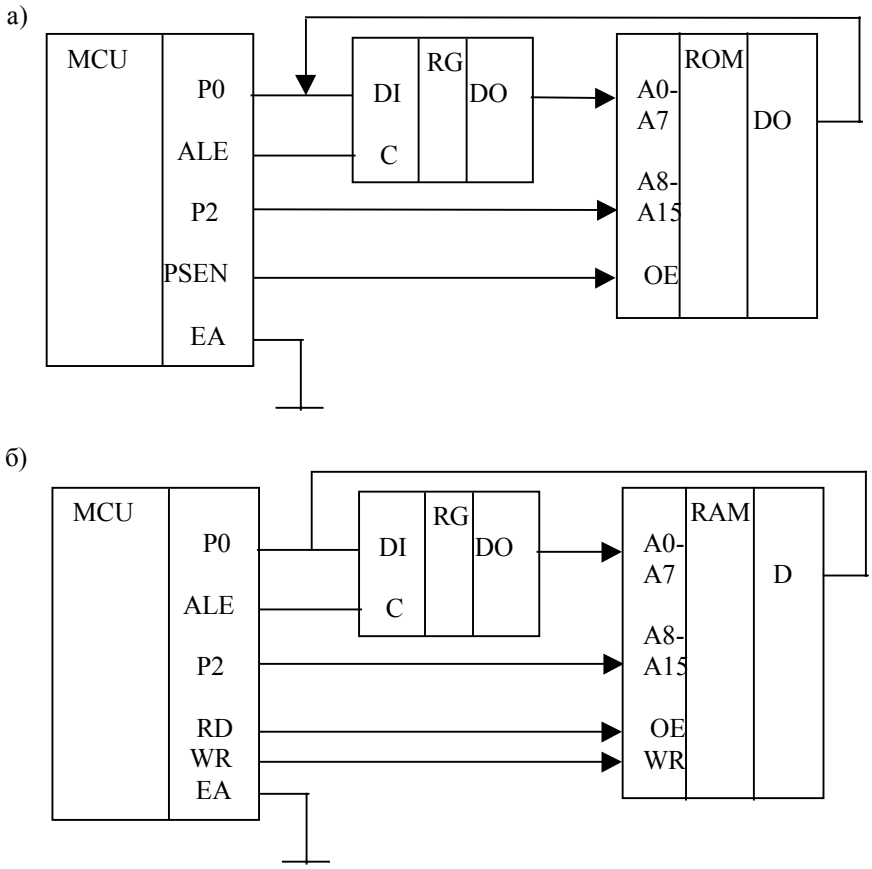


Рис. 2-2. Организация доступа к внешней памяти программ (а) и к внешней памяти данных (б).

Схемотехническая реализация, а также пример совмещения памяти программ и памяти данных в едином адресном пространстве приводятся в разделе 3.5.

Табл. 2-2. Регистры специальных функций МК 80С515.

Имя регистра	**Адрес**
Регистры АЛУ:	
ACC Аккумулятор	E0h
B Регистр B	F0h
PSW Флаги слова состояния программы	D0h
Указатели:	
SP Указатель стека	81h
DPL Указатель данных - младший байт	82h
DPH Указатель данных - старший байт	83h
Параллельные порты:	

Имя регистра	**Адрес**
P0 Порт 0	80h
P1 Порт 1	90h
P2 Порт 2	A0h
P3 Порт 3	B0h
P4 Порт 4	E8h
P5 Порт 5	F8h
P6 Порт 6	DBh
Последовательный порт:	
SCON Регистр управления	98h
SBUF Буфер	99h

Имя регистра	Адрес
Таймеры:	
TCON Регистр управления таймерами 0 и 1	88h
TMOD Регистр режимов таймеров 0 и 1	89h
T2CON Регистр управления таймером 2	C8h
TH0 Регистр данных таймера 0 старший байт	8Ch
TL0 Регистр данных таймера 0 младший байт	8Ah
TH1 Регистр данных таймера 1 старший байт	8Dh
TL1 Регистр данных таймера 1 младший байт	8Bh
TH2 Регистр данных таймера 2 старший байт	CDh
TL2 Регистр данных таймера 2 младший байт	CCh
Блок HSIO:	
CCEN Регистр разрешения захвата/привязки	C1h
CCL1 Регистр захвата/сравнения 1 младший байт	C2h
CCH1 Регистр захвата/сравнения 1 старший байт	C3h
CCL2 Регистр захвата/сравнения 2 младший байт	C4h
CCH2 Регистр захвата/сравнения 2 старший байт	C5h
CCL3 Регистр захвата/сравнения 3 младший байт	C6h
CCH3 Регистр захвата/сравнения 3 старший байт	C7h
CRCL Регистр загрузки, захвата/сравнения младший байт	CAh
CRCH Регистр загрузки, захвата/сравнения старший байт	CBh
Система прерываний:	
IEN0 Регистр масок 0	A8h
IEN1 Регистр масок 1	B8h
IP0 Регистр приоритетов 0	A9h
IP1 Регистр приоритетов 1	B9h
IRCON Регистр флагов	C0h
АЦП:	
ADCON Регистр управления	D8h
ADDAT Регистр данных	D9h
DAPR Регистр опорных напряжений	DAh
Управление потреблением:	
PCON Регистр управления энергопотреблением	87h

Формат регистра PSW:

Номер бита	Название бита	Назначение
7	C	Перенос/заем в беззнаковых арифметических операциях.
6	AC	Межтетрадный перенос/заем.
5	F0	Флаг №0, специфицируемый пользователем.
4	RS1	Двоичный код активного банка регистров: 00 - банк №0; 01 - банк №1; 10 - банк №2; 11 - банк №3.
3	RS0	

2	OV	Переполнение в знаковых арифметических операциях.
1	F1	Флаг №1, специфицируемый пользователем.
0	P	Дополнение до четного количества единиц в аккумуляторе.

Во внутренней памяти программ можно выделить следующие участки:

- в байтах 00…07, 08…15, 16…23 и 24…31 располагаются четыре банка регистров общего назначения R0…R7. Переключение между банками осуществляется настройкой битов RS1… RS0 в PSW;

- в байтах 32-47 располагаются 128 прямоадресуемых бит: биты 00…07 расположены в 32 байте (младший справа), биты 08…15 расположены в 33 байте (младший справа) и т.д.

Остальные байты памяти не несут дополнительной функциональной нагрузки и предназначены для хранения программных переменных;

Стек, растущий в область старших адресов, по умолчанию начинается с байта 08. При использовании банков регистров, отличных от нулевого, и области битовых переменных, необходимо следить за отсутствием перекрытий между этими областями и стеком, либо переопределить стек, занеся в SP необходимый адрес начала стека, уменьшенный на единицу.

2.3. Система команд.

МК 80С515 оперирует с операндами четырех типов: битами, тетрадами, байтами и 16-битными словами, при этом большинство операндов команд являются байтами.

В МК 80С515 используются следующие типы адресации: неявная, регистровая, прямая, непосредственная, косвенная регистровая и разновидность косвенной - базово-индексная адресация. Указанные способы адресации обеспечивают обращение к операндам-источникам. При обращении к операндам-приемникам непосредственная и базово-индексная адресации не используются.

Неявная адресация используется для адресации аккумулятора, регистра B в командах умножения и деления, флага C в командах битовых операций, косвенной адресации посредством DPTR и SP.

Регистровая адресация применяется для адресации регистров R0 - R7 текущего банка регистров. Так как эти регистры принадлежат пространству младших 128 байт внутреннего ОЗУ, то к ним также можно обратиться, используя прямую и косвенную адресацию через регистр Ri (i = 0, 1) текущего банка регистров.

Прямая адресация используется для задания адресов младших 128 байт внутренней памяти данных и адресов регистров блока SFR.

Косвенная адресация используется для обращения к операндам внутренней памяти данных через регистр Ri (i = 0, 1) и в командах пересылок из внешней/во внешнюю память данных через

регистр Ri или DPTR.

Для адресации бит используется только прямая адресация, т.е. адресация, когда в команде содержится 8-разрядный адрес бита.

При обращении к внешней памяти программ для считывания констант используется базово-индексная адресация, обеспечивающая формирование адреса выбираемого байта путем сложения 8-битного беззнакового содержимого аккумулятора с содержимым 16-битного базового регистра (DPTR или PC).

В МК 80C515 применяется 13 типов команд (см. рис. 2-3). Первый байт любой команды содержит код операции.

Синтаксис команд языка ассемблера состоит из мнемонического обозначения функции команды, за которым могут размещаться операнды, указывающие методы адресации и типы данных. Машинные коды команд МК представляются одним, двумя или тремя байтами. Команды выполняются за один или два машинных цикла (исключение составляют команды умножения и деления, длительность исполнения которых составляет 4 машинных цикла).

Для обозначения типов данных и способов адресации в мнемониках команд применяются следующие обозначения:

Rn - регистры R0 - R7 текущего банка регистров;

bit - прямоадресуемый бит внутренней памяти данных или регистра SFR;

#d - 8-битная константа, входящая в состав команды;

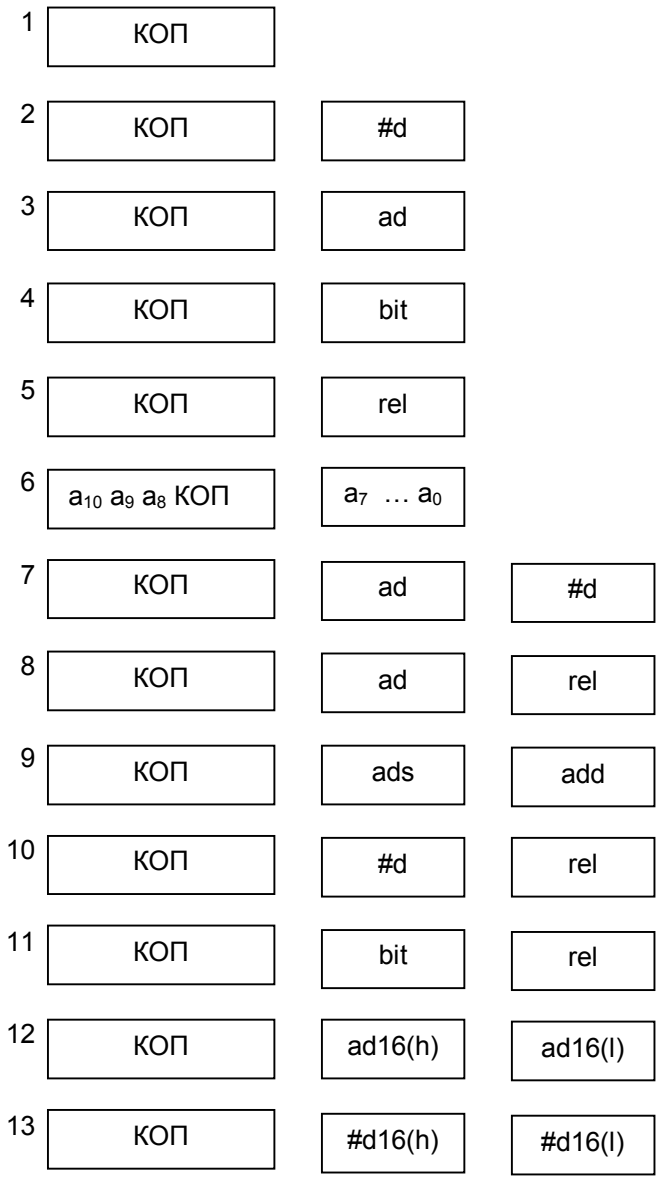


Рис. 2-3. Структура системы команд 80C515.

#d16 - 16-битная константа, входящая в состав команды;

ad - 8-битный адрес 128 младших байт внутренней памяти данных или регистра SFR;

@Ri - обозначение косвенной адресации через регистр Ri (i = 0, 1);

@DPTR - обозначение косвенной адресации через регистр DPTR;

ad16 - 16-битный адрес безусловной передачи управления;

ad11 - 11-битный адрес безусловной передачи управления в пределах 2-Кбайтного блока памяти программ;

rel - 8-битное значение смещения со знаком, которое суммируется с содержимым PC для формирования адреса перехода в командах условной передачи управления и командах короткого безусловного перехода;

/bit - в команде используется инвертированное значение бита-источника.

Состав системы команд МК 80C515 приведен в Приложении 2.

2.4. Параллельные порты ввода-вывода.

В состав периферийных устройств МК входят семь 8-разрядных портов: шесть портов ввода-вывода P0, P1, P2, P3, P4, P5 и один порт ввода P6.

Порты ввода-вывода МК 80C515 не содержат регистров управления. Для реализации альтернативных функций, а также для настройки линии порта на ввод информации необходимо записать "1" в соответствующий разряд регистра защелки.

Альтернативными функциями обладают порты P0, P1, P2 и P3.

При работе с внешней памятью (на входе EA шины управления МК низкий уровень) линии портов P0 и P2 выполняют функции внешней шины адреса и двунаправленной шины данных. С помощью порта P0 осуществляется мультиплексирование младшего байта адреса и данных на внешней шине МК. При внешних обменах на выводы порта P0 последовательно во времени поступают младший байт адреса A7 - A0, строб фиксации адреса ALE, а затем данные D7 - D0, стробируемые сигналами PSEN при обращении к памяти программ для чтения, RD - при обращении к памяти данных для чтения, WR - при обращении к памяти данных для записи.

Через порт P2 выводится старший байт адреса A15 - A8.

Альтернативные функции порта P1:

Номер бита	Название бита	Назначение
7	T2	Внешний счетный вход таймера T2
6	CLKOUT	Выходной сигнал тактовой частоты Fosc/12
5	T2EX	Сигнал внешней принудительной перезагрузки таймера T2
4	INT2	Сигнал внешнего запроса прерывания №2
3	INT6/CC3	Сигнал внешнего запроса прерывания №6/ Канал захвата/сравнения №3
2	INT5/CC2	Сигнал внешнего запроса прерывания №5/ Канал захвата/сравнения №2
1	INT4/CC1	Сигнал внешнего запроса прерывания №4/ Канал захвата/сравнения №1
0	INT3/CC0	Сигнал внешнего запроса прерывания №3/ Канал захвата/сравнения №0

Альтернативные функции порта P3:

Номер бита	Название бита	Назначение
7	RD	Сигнал чтения внешней памяти данных
6	WR	Сигнал записи внешней памяти данных
5	T1	Счетный вход таймера 1
4	T0	Счетный вход таймера 0
3	INT1	Вход внешнего прерывания 1/вход разрешения счета таймера 1
2	INT0	Вход внешнего прерывания 0/вход разрешения счета таймера 0

1	TxD	Выход асинхронного последовательного порта/выход синхроимпульсов синхронного последовательного порта
0	RxD	Вход асинхронного последовательного порта/вход/выход данных синхронного последовательного порта

2.5. Таймеры-счетчики T0 и T1.

В состав периферийных устройств МК 80C515 входят три 16-битных таймера/счетчика. Рассмотрим таймеры/счетчики общего назначения T/C0 и T/C1. При работе в качестве таймера содержимое T/C инкрементируется в каждом машинном цикле, т.е. через каждые 12 периодов резонатора. При работе в качестве счетчика содержимое T/C инкрементируется под воздействием перехода из 1 в 0 внешнего входного сигнала, подаваемого источником импульсов на вход T0 (T1).

Для управления режимами работы T/C и для организации взаимодействия таймеров с системой прерывания используются два регистра специальных функций - TMOD и TCON, а также регистры данных TH1, TL1 (старший и младший байты счетчика 1) и TH0, TL0 (старший и младший байты счетчика 0).

Формат регистра TMOD:

Номер бита	Название бита	Назначение
7-T/C1 3-T/C0	Gate	Управление блокировкой: 0 - счет разрешается битом TRx 1- счет разрешается функцией (TRx)&(уровень на входе INTx)
6-T/C1 2-T/C0	C/T	Выбор источника тактирования: 0 - Focs/12; 1 - вход "Tx".
5-T/C1 1-T/C0	M1	Режим работы: 00: 13-битный таймер/счетчик; 01: 16-битный таймер/счетчик; 10: 8-битный автоперезагружаемый таймер/счетчик. "THx" хранит значение, которое перезагружается в "TLx" при его переполнении; 11: Таймер/счетчик 1 останавливается. TL0 работает как 8-битный таймер/счетчик, его режим определяется управляющими битами таймера 0. TH0 работает как 8-битный таймер, его режим определяется управляющими битами таймера 1
4-T/C1 0-T/C0	M0	

Формат регистра TCON:

Номер бита	Название бита	Назначение
7	TF1	Флаг переполнения таймера 1. Устанавливается аппаратно. Сбрасывается при обслуживании прерывания.
6	TR1	Бит управления таймера 1. Устанавливается/сбрасывается программно для пуска/останова таймера 1.
5	TF0	Флаг переполнения таймера 0. Устанавливается аппаратно при переполнении таймера/счетчика. Сбрасывается аппаратно при обслуживании прерывания.
4	TR0	Бит управления таймера 0. Устанавливается/сбрасывается программно для пуска/останова таймера/счетчика 0.
3	IE1	Флаг фронта прерывания 1. Устанавливается по срезу сигнала INT1. Сбрасывается при обслуживании прерывания.
2	IT1	Бит управления типом прерывания 1. Устанавливается/ сбрасывается программно для спецификации запроса INT1 (срез/низкий уровень).
1	IE0	Флаг фронта прерывания 0. Устанавливается по срезу сигнала INT0. Сбрасывается при обслуживании прерывания.
0	IT0	Бит управления типом прерывания 0. Устанавливается/ сбрасывается программно для спецификации запроса INT0 (срез/низкий уровень).

Модуль использует альтернативные функции T0, T1, INT0, INT1 порта P3.

2.6. Таймер 2 и блок быстрого ввода-вывода.

В МК 80C515 встроен четырехканальный блок быстрого ввода-вывода, использующий в своей работе дополнительный таймер 2 (T2). В качестве входов/выходов блока используются линии порта P1, настроенные на режим работы с альтернативными функциями: P1.0 - канал 0; P1.1 - канал 1; P1.2 - канал 2; P1.3 - канал 3; P1.5 - вход принудительной перезагрузки T2; P1.7 - счетный вход/разрешение работы T2.

Регистры SFR, используемые блоком быстрого ввода-вывода:

Регистр SFR	Назначение регистра SFR	Адрес
TH2/TL2	Старший/младший байт регистра данных T2	0CDh/0CCh
T2CON	Регистр управления T2	0C8h
CCEN	Регистр выбора режима работы каналов	0C1h
CCH1/CCL1	Старший/младший байт регистра меток канала 1	0C3h/0C2h
CCH2/CCL2	Старший/младший байт регистра меток канала 2	0C5h/0C4h
CCH3/CCL3	Старший/младший байт регистра меток канала 3	0C7h/0C6h

Регистр SFR	Назначение регистра SFR	Адрес
CRCH/CRCL	Старший/младший байт регистра перезагрузки и меток канала 0	0CBh/0CAh

Формат T2CON:

Номер бита	Название бита	Назначение
7	T2PS	Предделитель на 2 источника счетных импульсов: 0 - счетные импульсы поступают с частотой Fosc/12; 1 - с частотой Fosc/24.
6	I3FR	Вид сигнала запроса прерывания INT3 и захвата событий каналом 0: 0 - по срезу; 1 - по фронту.
5	I2FR	Вид сигнала запроса прерывания INT2: 0 - по срезу; 1 - по фронту.
4	T2R1	Режим перезагрузки: 00 и 01 - перезагрузка запрещена; 10 - перезагрузка по переполнению; 11 - перезагрузка по срезу сигнала на P1.5.
3	T2R0	
2	T2CM	Режим сравнения.
1	T2I1	Режим работы: 00 - останов; 01 - неуправляемый таймер; 10 - счетчик импульсов на P1.7 (инкремент по срезу сигнала); 11 - управляемый таймер (счет разрешен при P1.7 = 1).
0	T2I0	

Переполнение таймера 2 во всех режимах фиксируется во флаге TF2.

Формат CCEN:

Номер бита	Название бита	Назначение
7	Режим	Для каждого канала:
6	канала CC3	00: Захват/сравнение запрещены
5	Режим	01: Захват внешнего события (быстрый ввод)
4	канала CC2	10: Сравнение (быстрый вывод или ШИМ)
3	Режим	11: Захват содержимого T2
2	канала CC1	
1	Режим	
0	канала CC0	

В режиме быстрого ввода в каналах 1-3 захват события происходит по срезу сигнала на соответствующем разряде порта 1, в канале 0 - по срезу или фронту в зависимости от значения бита I3FR регистра T2CON. Содержимое регистра данных T2, соответствующее моменту захвата, загружается в регистр меток канала CCx в цикле, следующем за идентификацией перехода. В режиме "01" при поступлении внешнего события осуществляется запись содержимого регистра данных T2 в

соответствующий регистр меток CCx. В режиме "11" захват инициируется при выполнении записи в регистр CCLx, что обеспечивает чтение содержимого регистра данных T2 без остановки счета (на лету).

Требуемый режим сравнения задается битом T2CM регистра T2CON.

В режиме 0 (формирование ШИМ-сигналов) при совпадении значений регистра данных T2 и регистра меток канала на соответствующем выводе порта устанавливается единичное значение выходного ШИМ-сигнала. Импульс переполнения таймера T2 переключает ШИМ-сигнал в значение логического нуля.

В режиме 1 (привязка генерации кода) при совпадении значений регистра данных T2 и регистра меток канала на соответствующем выводе порта устанавливается значение, предварительно записанное в него программой. Особенность этого режима - отложенное до наступления заданного времени формирование логического уровня на выходе порта.

2.7. Аналого-цифровой преобразователь.

АЦП содержит три регистра специальных функций: ADCON - регистр управления, ADDAT-регистр данных и DAPR - регистр опорных напряжений.

Регистр управления АЦП ADCON (адрес 0D8h) служит для управления входным мультиплексором, установки работы АЦП и фиксации бита готовности.

Формат ADCON:

Номер бита	Название бита	Назначение
7	BD	Управление режимом последовательного порта: 0 - настройки определяются SCON; 1 - частота обмена 9600 бит/с
6	CLK	Подключение частоты Focs/12 к выводу P1.6
5	-	Зарезервирован
4	BSY	Флаг занятости: 0 - АЦП свободен; 1 - выполняется преобразование.
3	ADM	Режим преобразования: 0 - одиночный; 1- непрерывный
2	MX2	Двоичный код выбранного канала
1	MX1	
0	MX0	

Регистр данных АЦП ADDAT (адрес 0D9h) служит для хранения байта данных результата аналого-цифрового преобразования.

Регистр специальных функций DAPR (адрес 0DAh) предназначен для программирования внутренних опорных напряжений АЦП - минимального VAGND и максимального VAREF. Четыре младших разряда задают значение VAGND, четыре старших - значение VAREF. Шаг изменения составляет 0,3125В, при этом разность VAREF-VAGND должна составлять не менее 1 В.

Преобразование начинается с обращения к регистру DAPR и занимает не менее 15 машинных циклов. Изменение любого опорного напряжения требует дополнительно семь машинных циклов. Таким образом, максимальное время преобразования при программировании обоих внутренних опорных напряжений составляет 29 машинных циклов (29 мкс при частоте Fosc=12МГц).

2.8. Последовательный порт.

Посредством универсального синхронно-асинхронного приемопередатчика (УСАПП) осуществляется прием и передача информации, представленной последовательным кодом (младшими битами вперед), в полном дуплексном режиме обмена.

Для управления режимами работы УСАПП используются регистр режимов SCON, бит SMOD (PCON.7), бит BD (ADCON.7) а также регистр буфера приемопередатчика SBUF.

Формат регистра SCON:

Номер бита	Название бита	Назначение
7	SM0	Биты управления режимом, устанавливаются программно. 00: режим 0; 01: режим 1; 10: режим 2; 11: режим 3
6	SM1	
5	SM2	Бит управления режимом. Устанавливается программно для запрета приема сообщения, в котором девятый бит равен 0.
4	REN	Бит разрешения приема. Устанавливается/сбрасывается программно для разрешения/запрета приема последовательных данных.
3	TB8	Передаваемый бит 8. Устанавливается/сбрасывается программно для задания значения девятого передаваемого бита в режиме 9-битного приемопередатчика.
2	RB8	Принятый бит 8. Устанавливается/сбрасывается аппаратно для фиксации девятого принятого бита в режиме 9-битного приемопередатчика.
1	TI	Флаг прерывания передатчика. Устанавливается аппаратно при окончании передачи. Сбрасывается программно после обслуживания прерывания.
0	RI	Флаг прерывания приемника. Устанавливается аппаратно при окончании приема. Сбрасывается программно после обслуживания прерывания.

В режиме 0 ("синхронный 8-битный приемопередатчик") 8-битная посылка передаётся и принимается через внешний вывод входа приемника (RXD). Через внешний вывод выхода передатчика (TXD) выдаются импульсы сдвига, которые сопровождают каждый бит. Частота передачи бита информации равна 1/12 частоты резонатора.

В режиме 1 ("асинхронный 8-битный приемопередатчик с изменяемой таймером скоростью передачи") 10-битная посылка (старт-бит ("0"), 8 бит данных и стоп-бит ("1")) передается через TXD, принимается через RXD. Частота передачи равна ((2**SMOD)/32)*FovT1, где FovT1 - частота пе-

реполнений таймера T1.

В режиме 2 ("асинхронный 9-битный приемопередатчик с фиксированной скоростью передачи") 11-битная посылка (старт-бит ("0"), 8 бит данных, программируемый девятый бит и стоп-бит ("1")) передается через TXD, принимается через RXD. Частота передачи равна ((2**SMOD)/64)*Fosc.

В режиме 3 ("асинхронный 9-битный приемопередатчик с изменяемой таймером скоростью передачи") 11-битная посылка (старт-бит ("0"), 8 бит данных, программируемый девятый бит и стоп-бит ("1")) передается через TXD, принимается через RXD. Частота передачи равна ((2**SMOD)/32)*FovT1, где FovT1 - частота переполнений таймера T1.

Примерные константы перезагрузки T1 при использовании его для задания некоторых частот передачи в 8-битном автоперезагружаемом режиме указаны ниже.

9,6 кГц	0FDh
4,8 кГц	0F9h
2,4 кГц	0F3h
1,2 кГц	0E6h

Отметим, что для организации обмена с частотой, отличной от 9600 бит/с, в соответствующих режимах необходимо обнулить бит BD ADCON.

Передача инициируется любой командой, в которой SBUF указан как приемник. Прием в режиме 0 осуществляется при условии, что RI = 0 и REN = 1. В остальных режимах прием начинается с приходом старт-бита, если REN = 1.

Устройство использует альтернативные функции RxD и TxD порта P3.

2.9. Система прерываний.

Система прерываний контроллера 80C515 имеет двенадцать источников прерываний и четыре уровня приоритетов.

Для управления системой прерываний используются следующие регистры SFR: регистры масок IE0 и IE1, регистры приоритетов IP0 и IP1, регистры настроек сигналов запроса прерываний IRCON, T2CON(частично) и TCON (частично).

При обработке запроса прерывания контроллер прерываний осуществляет переключение на обработчик путем перехода на точку входа, жестко сопоставленную источнику прерываний.

Точки входа в обработчики прерываний:

Источник	Адрес входа	Наименование прерывания
IE0	0003H	Внешнее прерывание 0
TF0	000BH	Прерывание от таймера 0
IE1	0013H	Внешнее прерывание 1
TF1	001BH	Прерывание от таймера 1
RI+TI	0023H	Прерывание от последовательного порта
TF2+EXF2	002BH	Прерывание от таймера 2

Источник	Адрес входа	Наименование прерывания
IADC	0043H	Прерывание от АЦП
IEX2	004BH	Внешнее прерывание 2
IEX3	0053H	Внешнее прерывание 3
IEX4	005BH	Внешнее прерывание 4
IEX5	0063H	Внешнее прерывание 5
IEX6	006BH	Внешнее прерывание 6

Все флаги запросов прерываний, кроме IE0 и IE1, могут быть установлены/сброшены программным способом, и таким образом выполнять функции программных прерываний.

Все источники прерываний могут быть индивидуально разрешены/запрещены установкой/сбросом соответствующего источнику бита из регистров IEN0 и IEN1 (либо полностью запрещены сбросом бита EA).

Формат регистра IE0 (0A8h):

Номер бита	Название бита	Назначение
7	EA	Бит разрешения работы контроллера прерываний.
6	WDT	Управление сторожевым таймером (см. раздел 2-10).
5	ET2	Бит разрешения прерывания по переполнению таймера T2
4	ES	Бит разрешения прерывания по окончанию обмена через последовательный порт
3	ET1	Бит разрешения прерывания по переполнению таймера T1
2	EX1	Бит разрешения прерывания по внешнему сигналу INT1
1	ET0	Бит разрешения прерывания по переполнению таймера T0
0	EX0	Бит разрешения прерывания по внешнему сигналу INT0

Формат регистра IE1 (0B8h):

Номер бита	Название бита	Назначение
7	EXEN2	Бит разрешения прерывания по перезагрузке таймера внешним сигналом
6	SWDT	Управление сторожевым таймером (см. раздел 2-10).
5	EX6	Бит разрешения прерывания по внешнему сигналу INT6
4	EX5	Бит разрешения прерывания по внешнему сигналу INT5
3	EX4	Бит разрешения прерывания по внешнему сигналу INT4

Номер бита	Название бита	Назначение
2	EX3	Бит разрешения прерывания по внешнему сигналу INT3
1	EX2	Бит разрешения прерывания по внешнему сигналу INT2
0	EADC	Бит разрешения прерывания по завершению преобразования в АЦП

Для присвоения источнику прерываний определенного приоритета используются регистры IP0 и IP1. Приоритеты назначаются одновременно паре источников прерываний двумя битами IP0.x и IP1.x. Код приоритета с меньшим номером соответствует более низкому уровню приоритета.

Биты	Пара источников прерываний
IP1.0 IP0.0	IE0/IADC
IP1.1 IP0.1	TF0/IEX2
IP1.2 IP0.2	IE1/IEX3
IP1.3 IP0.3	TF1/IEX4
IP1.4 IP0.4	RI+TI/EX5
IP1.5 IP0.5	TF2+EXF2/IEX6

Формат регистра запросов прерываний IRCON(0C0H):

Номер бита	Название бита	Назначение
7	EXF2	Флаг запроса прерывания по перезагрузке таймера внешним сигналом
6	TF2	Флаг запроса прерывания по перезагрузке таймера T2
5	IEX6	Флаг запроса прерывания по внешнему сигналу INT6
4	IEX5	Флаг запроса прерывания по внешнему сигналу INT5
3	IEX4	Флаг запроса прерывания по внешнему сигналу INT4
2	IEX3	Флаг запроса прерывания по внешнему сигналу INT3
1	IEX2	Флаг запроса прерывания по внешнему сигналу INT2
0	IADC	Флаг запроса прерывания по завершению преобразования в АЦП

Внешние прерывания INT2 (INT3) могут быть активизированы фронтом/срезом в зависимости от значения бита I2FR (I3FR) в регистре T2CON. Прерывания устанавливают флаги IEX2 (IEX3) в регистре IRCON, которые сбрасываются автоматически при вызове программы обслуживания.

Внешние прерывания INT4, INT5 и INT6 активизируются фронтом. Они устанавливают флаги IEX4, IEX5 и IEX6 в регистре IRCON, которые сбрасывается автоматически.

Переполнение таймера 2 активизирует флаг TF2. Сбрасывается программно.

Прерывание от P1.5 (внешнее прерывание/прерывание перезагрузки T2), активизирует флаг EXF2. Сбрасывается программно.

Внешние прерывания INT0 и INT1 могут активизироваться срезом/низким уровнем в зависимости от значения бит IT0 и IT1 в регистре управления таймерами T0 и T1 TCON. Эти прерывания устанавливают флаги IE0 и IE1 в TCON.

Переполнение таймеров T0 и T1 активизирует флаги запросов прерывания TF0 и TF1. Флаги сбрасываются автоматически при вызове обработчика.

Прерывания от последовательного порта могут генерироваться приемником (флаг окончания приема - RI) или передатчиком (флаг окончания передачи - TI). Обработчик прерывания должен опросить эти флаги для определения источника прерывания, а затем сбросить установленный флаг.

2.10. Сторожевой таймер и особые режимы работы МК 80C515.

Сторожевой таймер МК управляется посредством следующих битов:

- бит запуска/перезапуска SWDT (IEN1.6);
- бит статуса WDTS (IP0.6)
- бит перезапуска WDT (IEN0.7).

После запуска таймера установкой SWDT автоматически устанавливается бит статуса, и таймер осуществляет выдержку интервала времени в 65532 мкс. Если за это время не было перезапуска таймера, он сгенерирует сигнал внутреннего сброса для инициализации МК. Прикладная программа, анализируя бит WDTS, может определить причину сброса МК. Для программного перезапуска таймера должны быть последовательно выполнены две команды: установка бита WDT и установка бита SWDT, которые будут сброшены автоматически.

Различают три особых режима работы 80C515: режим сброса, режим холостого хода и режим пониженного энергопотребления.

Режим сброса предназначен для инициализации МК и активизируется подачей низкого логического уровня на вход "Reset" МК. При сбросе обнуляются следующие SFR: A, B, PSW, PC, DPTR, SCON, TMOD, TCON, T2CON, TL0, TH0, TL1, TH1, TL2, TH2, CCL3, CCH3, CCL2, CCH2, CCL1, CCH1, CRCL, CRCH, CCEN, DAPR, ADDAT, IEN0, IEN1, IRCON, Watchdog. Обнуляются используемые биты следующих SFR: ADCON (00X00000B), PCON (000X0000B), IP0 (X0000000B), IP1 XX000000B. В регистры портов P0…P5 заносятся коды FFh. Состояние SBUF неопределенное. В SP заносится код 07. Состояние прочих ячеек памяти не изменяется (таким образом, после перезапуска программа может продолжить работу с неизмененными значениями своих переменных).

Режимы холостого хода и пониженного энергопотребления предназначены для уменьшения потребляемой мощности. Для их программной активации необходимо наличие низкого логического уровня на входе PE. Управление режимами проводится с использованием SFR PCON (87h).

Формат регистра управления потребляемой мощностью PCON:

Номер бита	Название бита	Назначение
7	SMOD	Режим удвоенной скорости передачи последовательного порта (к управлению мощностью не имеет отношения).
6	PDS	Переход в режим пониженного энергопотребления.
5	IDLS	Переход в режим холостого хода.
4	-	Не используется
3	GF1	Флаг общего назначения №1 (к управлению мощностью не имеет отношения).
2	GF0	Флаг общего назначения №0 (к управлению мощностью не имеет отношения).
1	PDE	Разрешение перехода в режим пониженного энергопотребления.
0	IDLE	Разрешение перехода в режим холостого хода.

Переход в режим холостого хода осуществляется установкой битов IDLE, а затем IDLS в двух последовательно выполняемых командах (для снижения вероятности случайного перехода в режим). При этом останавливается выполнение программы, но продолжает функционировать контроллер прерываний, активизирующий возобновление штатного функционирования МК при возникновении любого разрешенного прерывания (кроме того, возможен перезапуск системы по сигналу "сброс"). Ток потребления составляет около 15 мА (в штатном режиме 35 мА).

Переход в режим пониженного энергопотребления осуществляется установкой битов PDE, а затем PDS в двух последовательно выполняемых командах. Возобновление функционирования МК возможно лишь путем перезапуска системы по сигналу "сброс". Режим предназначен для сохранения содержимого внутренней памяти в критичных к мощности источника питания приложениях. Ток потребления составляет около 50 мкА.

РАЗДЕЛ 3. ПРОЕКТИРОВАНИЕ МИКРОКОНТРОЛЛЕРНЫХ СИСТЕМ.

3.1. Общие положения.

Проектирование - комплекс мероприятий, обеспечивающих поиск технических решений, удовлетворяющих заданным требованиям, их оптимизацию и реализацию в виде комплекта конструкторских документов и опытного образца (образцов), подвергаемого циклу испытаний на соответствие требованиям технического задания.

Процесс разработки любого изделия состоит из трех основных этапов: этапа научно-исследовательской работы (НИР), этапа опытно-конструкторской работы (ОКР) и этапа производства, которые входят в жизненный цикл изделия (рис. 3-1).

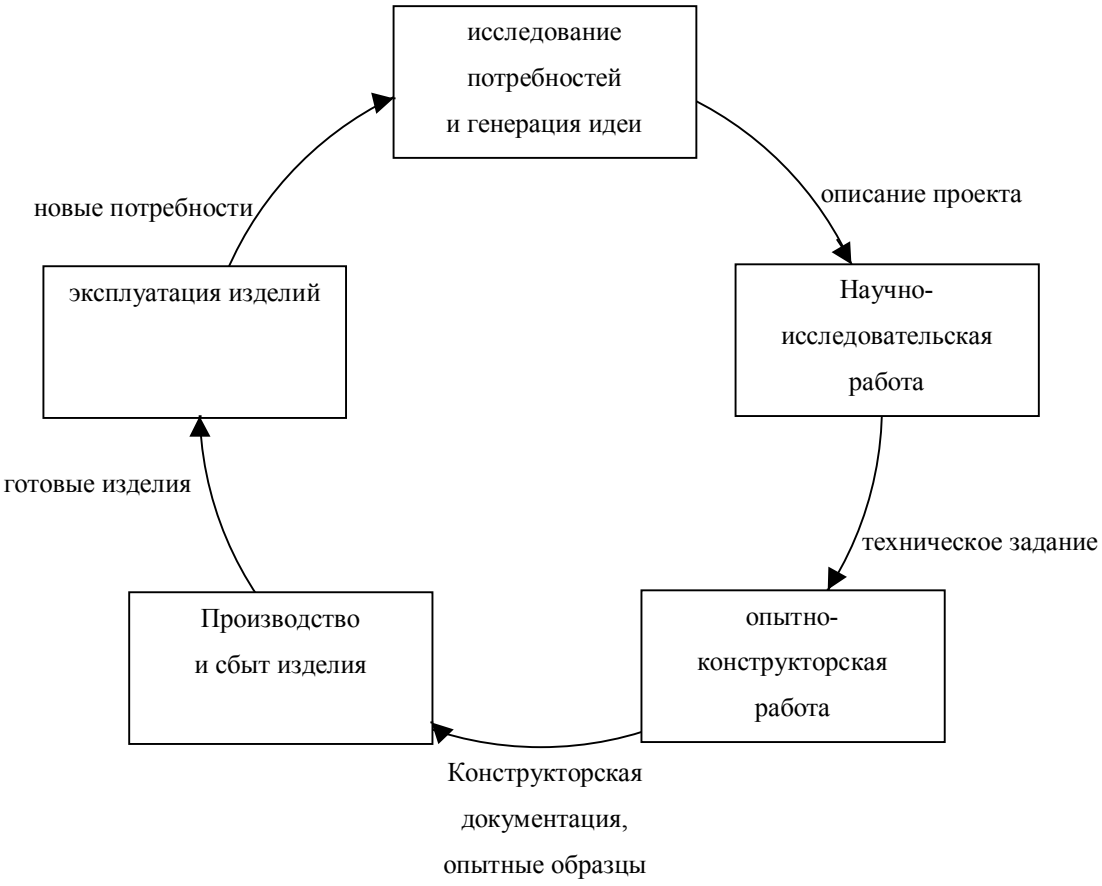


Рис. 3-1. Жизненный цикл изделия.

Этап НИР предназначен для определения принципиальной возможности и целесообразности разработки системы; по окончании этапа либо осуществляется постановка технического задания (ТЗ) на ОКР, либо дается мотивированное обоснование ее невозможности или нецелесообразности. Основными критериями оценки

являются: величина ожидаемой прибыли, степень риска производителя, длительность жизненного цикла продукции, доступность необходимых трудовых и материальных ресурсов, возможность производства по конкурентоспособным ценам и др.

Процесс выполнения НИР и ее основные стадии показаны на рис. 3-2.

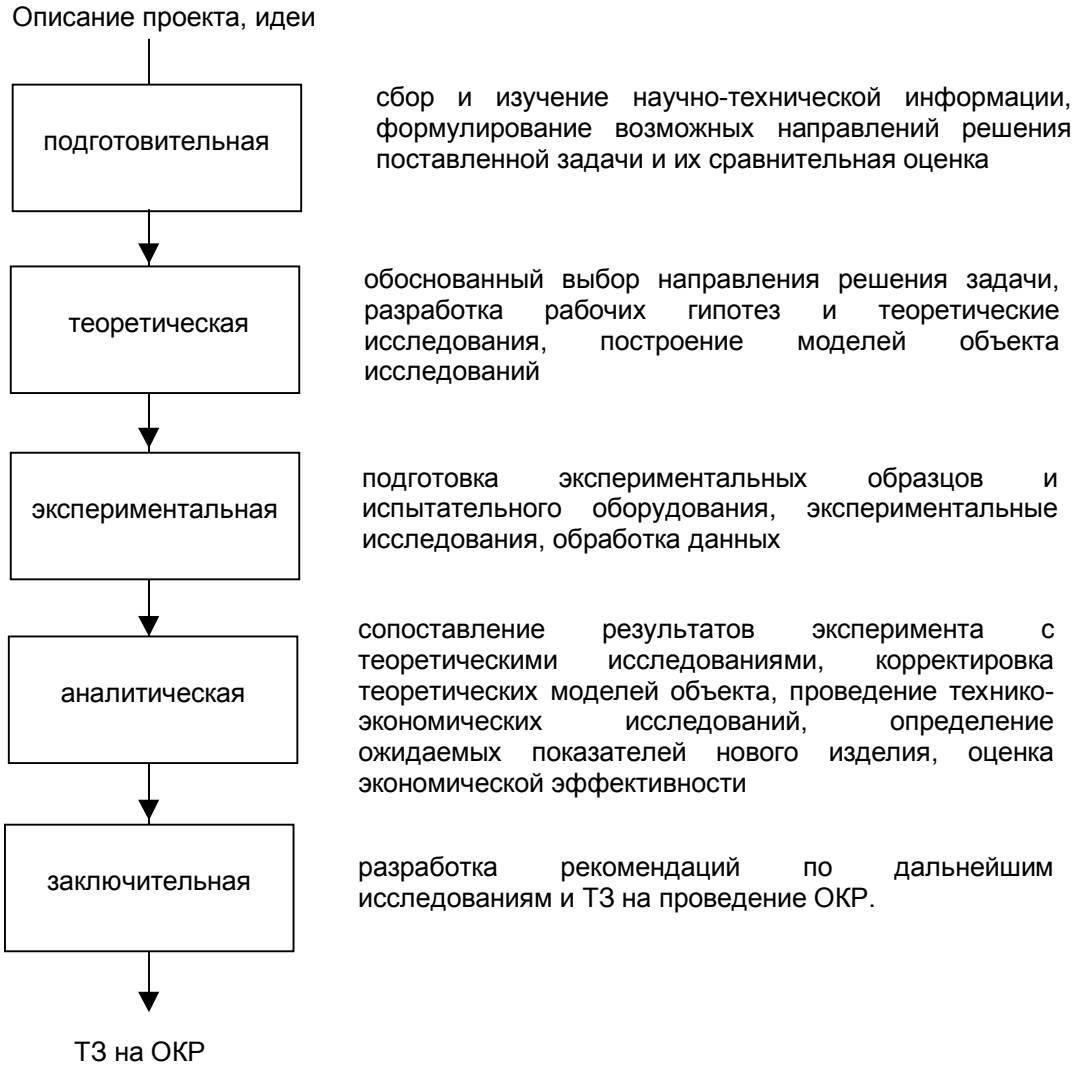


Рис. 3-2. Стадии НИР.

ТЗ описывает основное назначение, технические характеристики, показатели качества и технико-экономические требования, предъявляемые к системе.

После завершения НИР при условии положительных результатов анализа предложения на разработку, проводятся опытно-конструкторские работы (ОКР). Основная задача ОКР – создание комплекта конструкторской документации для серийного производства изделия.

Процесс выполнения ОКР и ее основные стадии показаны на рис. 3-3.

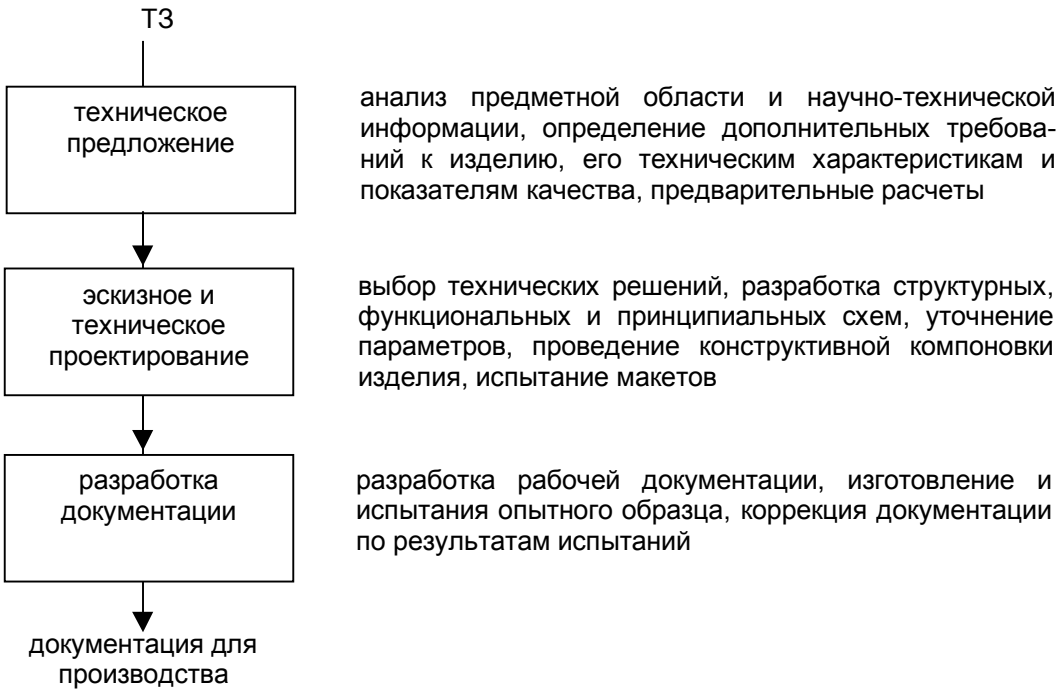


Рис. 3-3. Стадии ОКР.

Обеспечение производства включает в себя три составляющие:

- конструкторское обеспечение (адаптация конструкторской документации ОКР к условиям конкретного серийного производства предприятия-изготовителя. Производится в соответствии с правилами "Единой системы конструкторской документации" (ЕСКД));

- технологическое обеспечение (обеспечение технологической готовности предприятия к производству; разработка технологических маршрутов и процессов; сопровождение изготовления. Производится в соответствии с правилами "Единой системы технологической подготовки производства" (ЕСТПП));

- организационное обеспечение (расчеты хода производства, загрузки оборудования, движения материальных потоков; обеспечение кадрами, оборудованием, материалами, полуфабрикатами, финансовыми средствами; проектирование участков и цехов, планировка расположения оборудования).

Вопросы сбыта продукции находятся в компетенции отделов рекламы и сбыта предприятия, изучаются в разделе "менеджмент" экономических дисциплин и в настоящем пособии не рассматриваются.

3.2. Специфика проектирования встраиваемых приложений.

Обобщенная встраиваемая система управления на базе микроконтроллера, структура которой показана на рис. 3-4, включает в себя:

- объект управления, содержащий исполнительные устройства, собственно объект или процесс, и систему датчиков, предоставляющих информацию об объекте управления;

- микроконтроллерную систему управления, реализующую процесс управления объектом, заключающийся в сообщении объекту воздействий в соответствии с алгоритмом решаемой задачи и информацией, поступающей с датчиков контролируемых величин и с пульта управления системой;

- блок сопряжения с объектом управления, заключающий в себе унифицированный набор элементов, реализующих функции согласования сигналов (как управляющих, так и информационных) стандартов микроконтроллерной системы и объекта управления;

- пульт оператора системы, предоставляющий пользователю возможность контролировать параметры процесса управления и вносить в него коррективы по своему усмотрению.

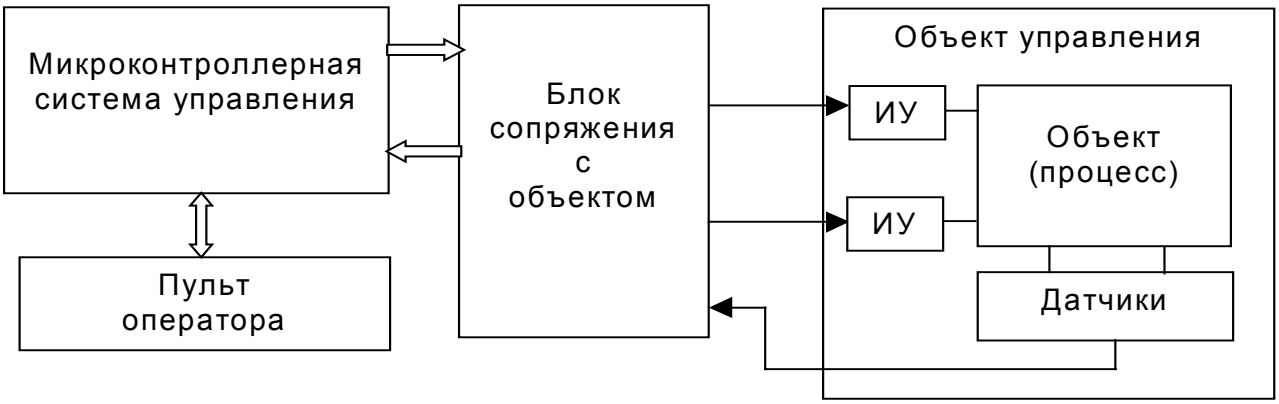


Рис 3-4. Структура системы управления на базе микроконтроллера.

Разработка встраиваемых приложений на МК представляет собой совокупность ряда этапов (рис. 3-5).

Далее в данном разделе рассмотрим вопросы, касающиеся непосредственно проектирования; вопросы отладки ввиду их особой важности будут рассмотрены в следующем разделе.

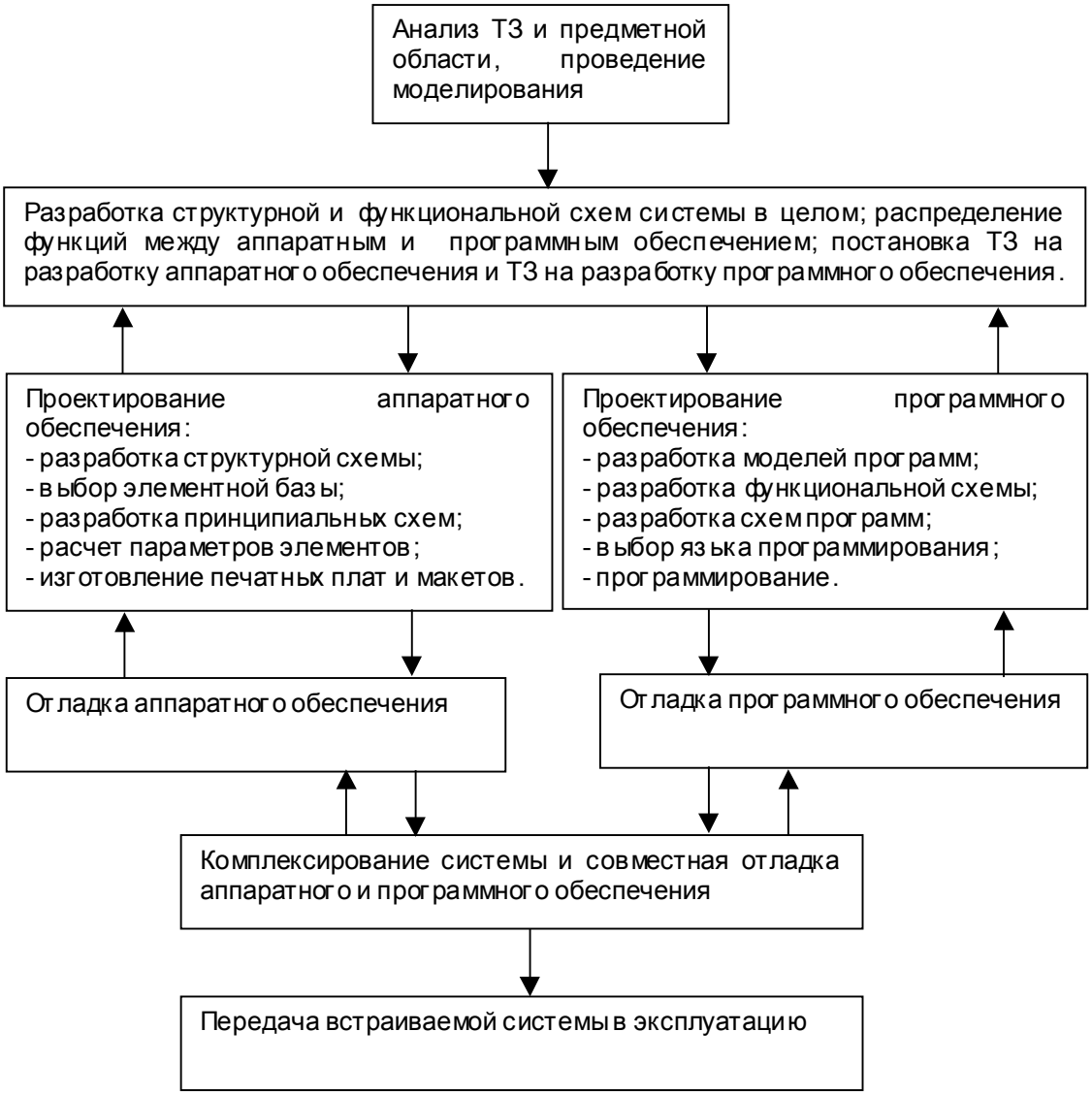


Рис. 3-5. Стадии проектирования встраиваемой системы на базе МК.

3.3. Анализ предметной области и моделирование.

Анализ ТЗ и предметной области - необходимый этап разработки любого изделия. Он проводится на начальной стадии проектирования при разработке технического предложения.

Целью анализа является генерация предложений по разработке структуры системы. Решение таких творческих задач основано на проводимых совместно системно-структурном анализе и синтезе. Полнота синтеза определяется глубиной анализа, которая зависит от степени разработки ТЗ. Данный этап позволяет выявить особенности объекта управления и требования к системе управления и проводится общими для любых областей технического проектирования методами системного анализа и моделирования.

Рассмотрим пример проведения моделирования.

Системе управления технологическим процессом необходимо поддерживать заданный уровень концентрации реагента в растворе, который в отсутствие регулирования экспоненциально уменьшается. Для определения текущего значения концентрации реагента в системе имеется фотоэлектрический датчик, применение которого основывается на законе Бугера-Ламберта-Бера, описывающего зависимость уровня интенсивности световой волны, проходящей через раствор реагента, от концентрации этого реагента в растворе. В случае снижения концентрации реагента относительно требуемой система производит его добавление в раствор, в случае превышения - для снижения концентрации используется естественное разложение реагента.

Структура системы показана на рис. 3-6, а ее описание в терминах среды моделирования MathCad приведено на рис. 3-7 и 3-8.

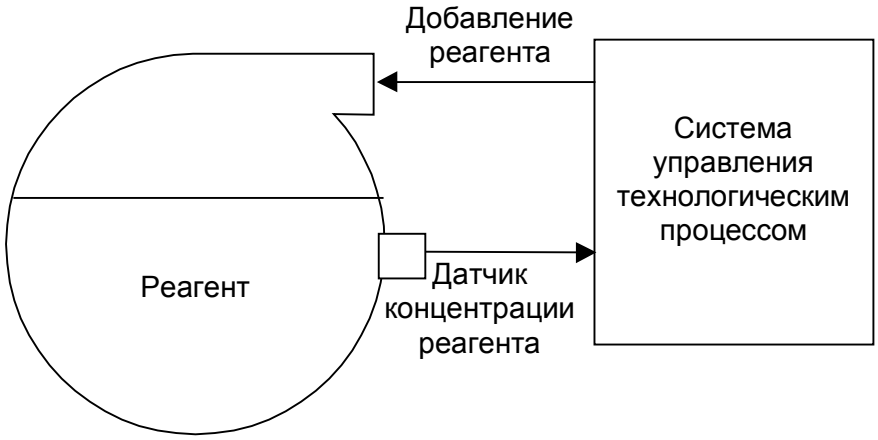


Рис. 3-6. Структура системы поддержания заданной концентрации реагента.

В построенной модели системы приведены описания и графики нерегулируемого процесса снижения концентрации раствора, желаемого уровня концентрации и реальных процессов регулирования при различных значениях коэффициента регулирования. Описание процесса регулирования дано в рекуррентной форме и с учетом невозможности системы управлять снижением концентрации реагента.

Примечания. Используемый закон, отражающий зависимость интенсивности выходной световой волны от концентрации раствора имеет силу лишь при слабых концентрациях растворенного реагента; используемый параметр А абстрактен.

Модель системы управления концентрацией реагента в растворе.

Описание необходимых переменных:

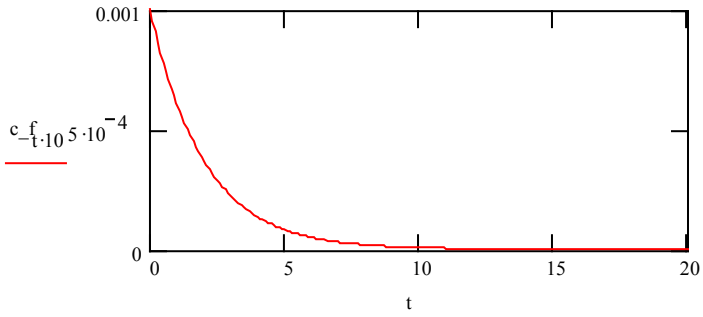
$t := 0, 0.1 .. 20$ Время наблюдения процессов

$c\_max := 0.001$ Максимальная концентрация раствора

Описание нерегулируемого процесса с разложением реагента:

$c\_f\_e := 0.5$ Множитель показателя степени разложения реагента

$c\_f_{t \cdot 10} := c\_max \cdot e^{-c\_f\_e t}$ Закон разложения реагента



Описание желаемого процесса изменения концентрации реагента:

$w\_c\_n := 0.6$ Частота изменения концентрации реагента

$c\_n_{t \cdot 10} := c\_max \cdot \left(0.5 + \frac{\sin(w\_c\_n \cdot t)}{2}\right)$ Требуемый закон изменения концентрации

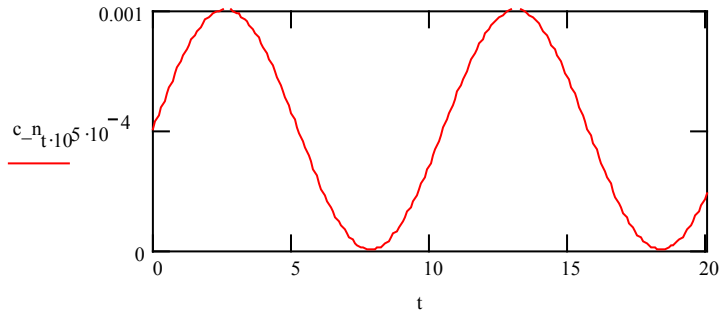


Рис. 3-7. Построение модели системы поддержания заданной концентрации реагента в растворе. Начало.

Описание регулируемого процесса поддержания заданной концентрации реагента:

$A := 1 \qquad d := 0.05$

Закон Бугера-Ламберта-Бера изменения относительной интенсивности световой волны, проходящей через раствор, в зависимости от концентрации реагента:
A - постоянная характеристика реагента,
d - расстояние между излучателем и приемником.

$U(c) := e^{-A \cdot c \cdot d}$

$K\_reg := (0 \quad -2 \quad -15)^T$

Три анализируемых коэффициента П-регулятора

$m := 0 .. 2$

Описание процесса регулирования:
очередное значение концентрации определяется на основе предыдущего (с учетом разложения реагента) значения и добавки реагента по П-закону.

$c\_res_{m,0} := c\_max$

$minus(a) := if(a < 0, a, 0)$

Система может лишь увеличивать концентрацию

$$c\_res_{m,t \cdot 10+1} := \left(c\_res_{m,t \cdot 10} \cdot e^{-c\_f\_e0.1} + K\_reg_m \cdot minus\left(U\left(c\_n_{t \cdot 10}\right) - U\left(c\_res_{m,t \cdot 10}\right)\right)\right)$$

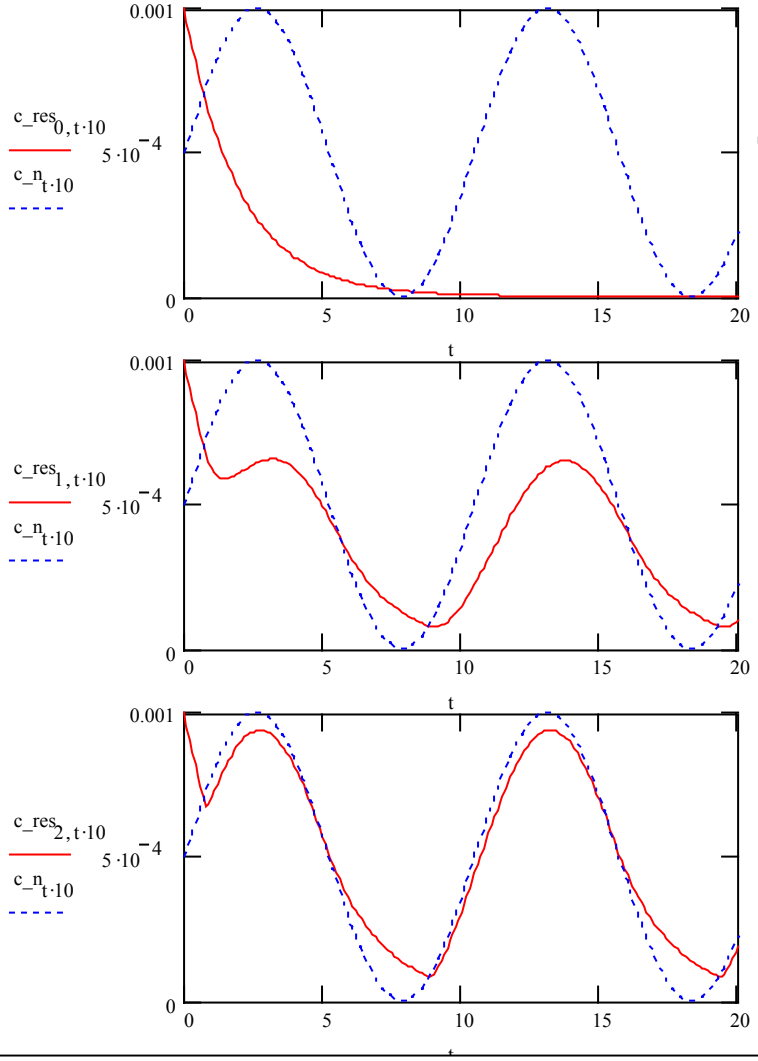


Рис. 3-8. Построение модели системы поддержания заданной концентрации реагента в растворе. Окончание.

3.4. Определение функций аппаратного и программного обеспечения. Разработка структурной схемы.

Процесс разработки структурной схемы системы проводится путем последовательного уточнения ее компонентов. На исходном этапе из ТЗ известны входы, выходы и управляющие воздействия на систему, поэтому проектируемую систему можно представить в виде совокупности более простых подсистем, связанных между собой входами, выходами и управляющими воздействиями (рис. 3-9). При последовательном уточнении структуры увеличивается число составляющих исходную систему подсистем, но сложность каждой из них уменьшается. При этом производится декомпозиция требований ТЗ, применяемых к конкретной подсистеме, начиная с подсистем, выходы которых являются выходами всей системы и заканчивая входными подсистемами. Процесс повторяется до достижения элементарного уровня детализации, представляющего систему состоящей из известных элементов, для которых проведение дальнейшей декомпозиции нецелесообразно.

На элементарном уровне для каждого вида применяемых элементов выполняется следующая процедура:

- определяется множество M известных разновидностей элемента;
- составляется иерархия требований ТЗ к элементам данного вида по критерию принципиальности для разработки;
- требованию a ставится в соответствие подмножество $A \subset M$;
- требованию b ставится в соответствие подмножество $B \subset A$;
- требованию c ставится в соответствие подмножество $C \subset B$;
- процесс повторяется до удовлетворения всех требований.

Если мощность полученного множества превышает 1, то некоторые из требований ТЗ могут быть превзойдены. Если пересечение является пустым множеством, значит, среди известных элементов нет удовлетворяющего всем требованиям ТЗ, и необходимо проведение дополнительной конкретизации данной подсистемы.

Методы синтеза делятся на эвристические (творческие) и формальные (математические). Как правило, эвристически предлагается общая структура системы, а формальными методами определяются конкретные технические решения. Соотношение формальных и эвристических методов зависит от сложности системы и ее класса (наиболее формализованы методы синтеза комбинационных цифровых и

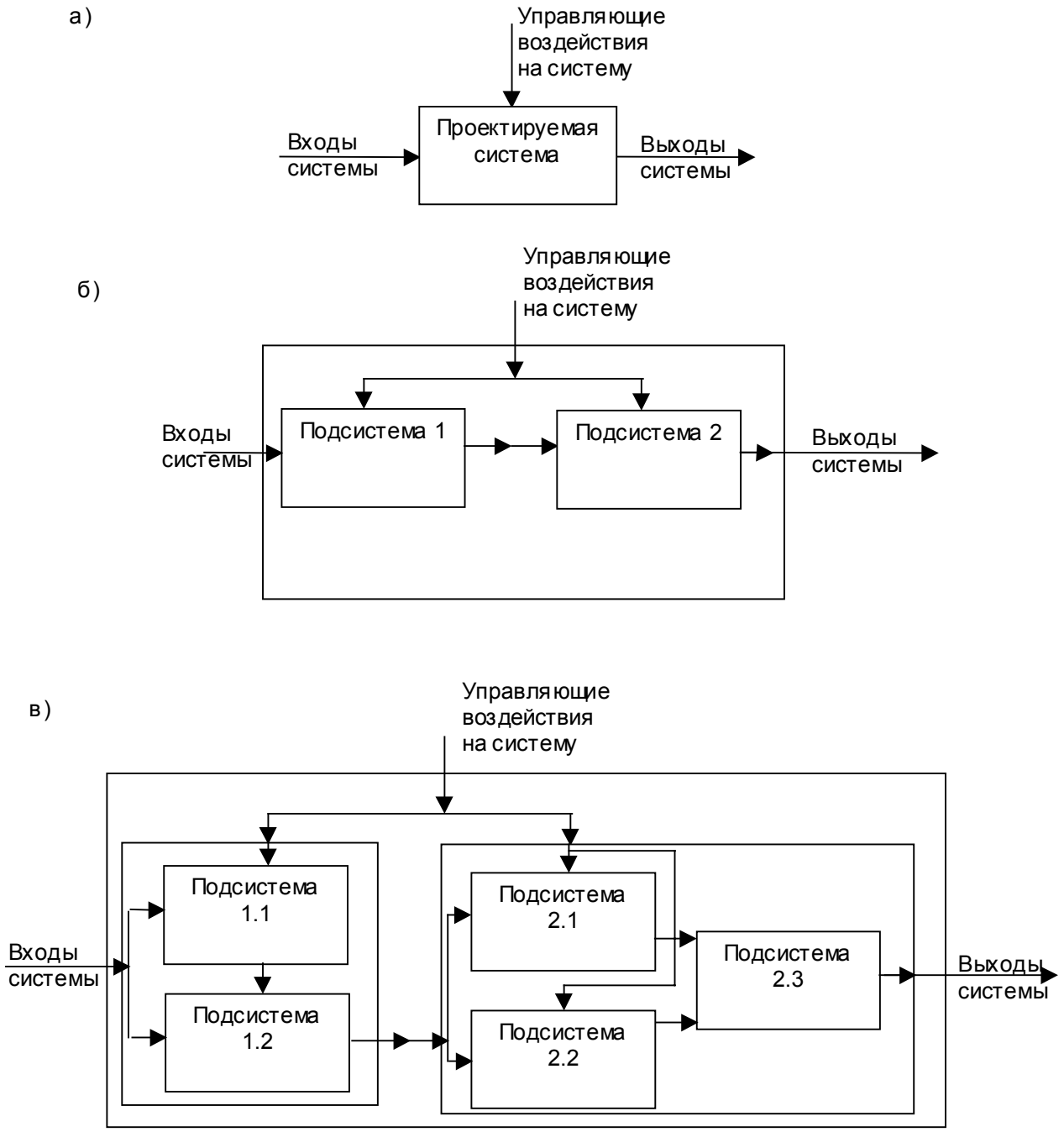


Рис. 3-9. Пример последовательного уточнения структуры системы.

линейных аналоговых схем, наименее - методы синтеза нелинейных и импульсных устройств).

Выбор конкретных технических решений математически представляет собой задачу оптимизации, для решения которой могут использоваться известные методы теории операций (вариационное исчисление, численные методы поиска, линейное и нелинейное программирование и др.).

При распределении функций между аппаратным и программным обеспечением следует исходить из того, что использование специализированных дополнительных БИС упрощает разработку и обеспечивает высокое быстродействие системы в целом, но сопряжено с увеличением стоимости, объема и потребляемой мощности. Больший удельный вес программного обеспечения позволяет сократить число компонентов системы и стоимость ее аппаратных средств, но это приводит к снижению быстродействия и увеличению затрат и сроков разработки и отладки прикладных программ. При этом время жизни изделия, в котором большая часть функций реализована в программном обеспечении, многократно возрастает за счет того, что срок морального старения изделия может быть существенно отодвинут. Программная реализация основных элементов алгоритма работы контроллера допускает его модификацию путем перепрограммирования, в то время как возможность изменения уже существующей фиксации элементов алгоритма в аппаратуре контроллера практически отсутствует [1].

3.5. Проектирование аппаратного обеспечения.

3.5.1. Выбор элементной базы.

Одна из наиболее важных задач выбора элементной базы относится к выбору непосредственно микроконтроллера. Принятие решения о выборе, как правило, проводится на основании методики комплексного сравнения параметров микроконтроллеров по совокупности основных технико-экономических характеристик.

Приведем одну из таких методик, базирующуюся на определении мультипликативного оценочного функционала [4].

Пусть, исходя из предполагаемой области будущего применения, заданы следующие требуемые параметры микроконтроллерной системы: производительность W, коэффициент запаса производительности K, количество линий ввода-вывода N, коэффициент запаса линий ввода-вывода L, потребляемая мощность P, массогабаритные показатели S, стоимость C, диапазон рабочих температур T и требования заказчика U.

Необходимо из M имеющихся на рынке микроконтроллерных систем выбрать оптимальную с точки зрения удовлетворения указанным требованиям.

Решение задачи заключается в вычислении функционалов

$$E_i = w_i \cdot n_i \cdot p_i \cdot s_i \cdot c_i \cdot t_i \cdot u_i \cdot \frac{W_i \cdot N_i}{P_i \cdot S_i \cdot C_i}$$

для всех i=1..M и нахождении j такого, что $E_j=\max\{E_i\}$. Оптимальной системой для данной области применения с точки зрения предъявленных требований будет система с порядковым номером j.

Здесь w_i, n_i, p_i, s_i, c_i, t_i, u_i - пороговые коэффициенты, определяющие применимость данного типа микроконтроллера в данной задаче, такие, что

$w_i=0$ при $W_i<K*W$, $w_i=1$ при $W_i=>K*W$;

$n_i=0$ при $N_i<L*N$, $n_i=1$ при $N_i=>L*N$;

$p_i=0$ при $P_i>P$, $p_i=1$ при $P_i<=P$;

$s_i=0$ при $S_i>S$, $s_i=1$ при $S_i<=S$;

$c_i=0$ при $C_i>C$, $c_i=1$ при $C_i<=C$;

$t_i=0$ при $T_i<>T$, $t_i=1$ при $T_i=T$;

$u_i=0$ при $U_i<>U$, $u_i=1$ при $U_i=U$.

Примечания к методике.

Примечание 1. Оценка уровня производительности как величины числа выполняемых команд в единицу времени является необъективной, так как не учитывает наиболее часто встречающуюся ситуацию с несовпадением систем команд сравниваемых микроконтроллерных систем. Например, в системе команд микроконтролера$_1$ из арифметических команд присутствуют только команды сложения и вычитания, а в системе команд микроконтролера$_2$ в состав арифметических команд дополнительно включены команды умножения и деления. При решении задачи Y=A*X+B микроконтроллер$_2$ выполнит две команды - умножения и сложения, а микроконтролер$_1$ вычислит произведение путем реализации нескольких сложений. При равной тактовой частоте окажется, что в единицу времени микроконтролер$_1$ выполнил большее количество команд, чем микроконтроллер$_2$ (так как время выполнения команды сложения меньше, чем время выполнения команды умножения), поэтому производительность микроконтролера$_1$ будет ошибочно считаться выше.

В связи с отмеченным, производительность микроконтроллеров определяется на тестовых задачах, содержащих смесь различных команд. Как правило, для удобства команды разделяют на короткие и длинные (например, к коротким относят команды сложения, а к длинным - умножения). В этом случае

$$W_i = \frac{1}{t_{к} \cdot \delta_{к} + t_{дл} \cdot \delta_{дл}} \cdot 100\,\%, \quad \text{при этом} \quad \delta_{к} + \delta_{дл} = 1$$

В смеси присутствуют короткие и длинные команды, имеющие определенную долю от общего количества команд смеси и время выполнения. В зависимости от доли команд каждого типа в данной смеси производительность контроллера может изменяться (см. рис. 3-10).

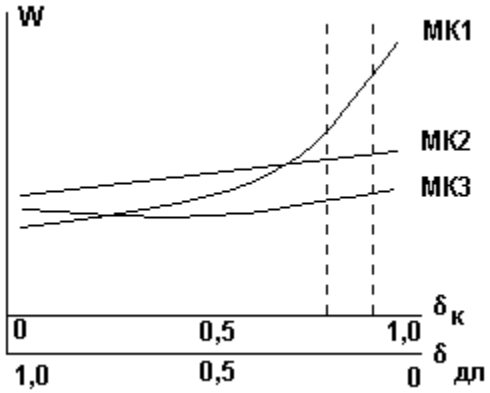


Рис. 3-10. Сравнение производительности микроконтроллеров.

В случае если предполагаемая область применения микроконтроллера требует относительно большой доли коротких команд (как показано на рисунке пунктирными вертикальными линиями), лидером по производительности оказывается микроконтроллер МК1.

Примечание 2. В случае, когда в состав микроконтроллерной системы входят несколько блоков, то значения энергетических, массогабаритных и стоимостных показателей микроконтроллерной системы определяются как сумма значений показателей всех блоков, входящих в состав системы.

Примечание 3. В Приложении 1 приведены некоторые представители МК и их основные характеристики.

Микроконтроллер представляет собой логический автомат с высокой степенью детерминированности, в связи с чем число вариантов его системного включения невелико. Поэтому типовой состав аппаратных средств ядра любой МК-системы (МК, ПЗУ, ОЗУ, интерфейсные БИС, схемы синхронизации и системного управления) оформляется конструктивно в виде одноплатных универсальных программируемых контроллеров, которые предназначены для встраивания в контур управления объектом или процессом. На некоторых моделях таких плат имеется так называемое монтажное поле пользователя, на котором он имеет возможность смонтировать свои специфические схемы, такие как оптронные развязки, реле и т.п. Кроме того, на пла-

те МК-системы может быть размещен источник электропитания. Пример реализации одноплатного МК показан на рис. 3-11 и 3-12.

При использовании в системе управления такого одноплатного контроллера, задача разработчика сводится к проектированию лишь специализированных схем сопряжения с объектом.

Рис. 3-11. Внешний вид одноплатного контроллера.

При выборе элементной базы для устройств сопряжения наиболее предпочтительным является использование интегральных схем, а дискретные компоненты выполняют вспомогательные функции. Элементы сравнивают по значению комплексного показателя качества $Q=\sum\varphi_i\alpha_i$, определяемого в виде совокупности нормированных значений частных показателей α_i, взвешенной коэффициентами значимости φ_i. Так, например, для диода к частным показателям относятся значения прямого и обратного токов, обратного напряжения, масса, рабочая температура. Для интегральных схем данным методом определяется серия.

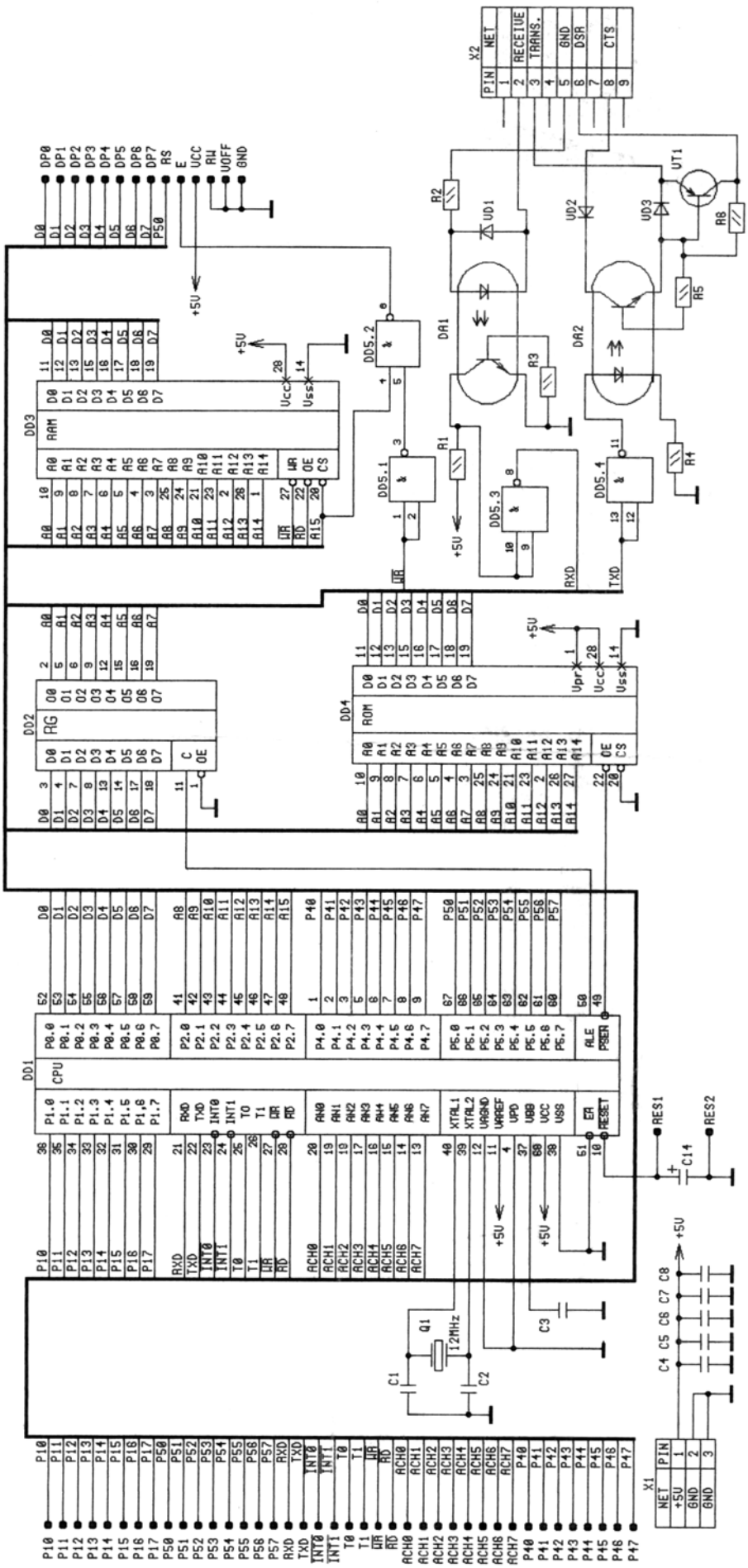


Рис. 3-12. Принципиальная схема одноплатного контроллера.

3.5.2. Разработка принципиальных схем.

Принципиальная схема аппаратной части системы разрабатывается по структурной схеме на основе требований ТЗ и требований, предъявляемых разработчиком к каждому функциональному элементу. Эти требования определяются нормативно-техническими документами, результатами экспериментов и испытаний, общими конструкторскими нормами и правилами с целью ограничения, типизации и унификации изделия. Разработка принципиальной схемы функционального элемента заключается в предложении схемы, удовлетворяющей совокупности технико-экономических требований при максимальной простоте и надежности.

При разработке МК-системы в состав аппаратных средств, как правило, вводят элементы настройки, управления, контроля и защиты.

К элементам настройки относятся элементы, параметры которых могут быть изменены в процессе производства с целью установления необходимого значения одного или нескольких параметров устройства. Элементы настройки используются в случаях, когда к допускам на выходные параметры функциональных элементов предъявляются жесткие требования, при дополнительных требованиях снижения стоимости системы и ее упрощения.

К элементам управления относятся элементы, параметры которых могут быть изменены в процессе эксплуатации с целью установления необходимого значения одного или нескольких параметров устройства, для изменения структуры системы (в многофункциональных системах), для включения резерва и т.п. В их число входят устройства регулировки, тумблеры, кнопки, сенсоры и др.

К элементам контроля относятся элементы, предназначенные для передачи оператору информации о функционировании системы. В их число входят измерительные приборы, табло, панели, индикаторы и т.п.

К элементам защиты относятся элементы, предназначенные для предотвращения выхода из строя или гибели системы при отклонении значений внешних параметров от допустимых, а также при возникновении аварийных режимов в самой системе по причине отказа ее частей.

При производстве, эксплуатации и ремонте системы осуществляют контроль и настройку режимов работы ее цепей. Для этих целей на схеме определяют некоторую совокупность контрольных точек для подключения контрольно-измерительной и тестовой аппаратуры.

3.5.3. Расчет параметров элементов.

Основной задачей расчета является определение значений электрических параметров элементов принципиальной схемы, обеспечивающих ее эффективную оптимизацию в дальнейшем. Задачу расчета полагают решенной, если определены номинальные значения всех пассивных компонентов, значения параметров схем замещения активных элементов, определены типы элементов при значениях выходных параметров, гарантирующих работоспособность системы в случайных условиях ее производства и эксплуатации.

При проектировании аппаратуры расчет наиболее часто выполняется в следующем порядке:

- ориентировочный расчет выходных параметров функциональных элементов;

- расчеты, позволяющие выбрать типы активных элементов (транзисторы, диоды, микросхемы);

- расчеты рабочих режимов активных элементов;

- расчет значений пассивных элементов, обеспечивающих выбранные режимы активных элементов;

- определение номинальных значений параметров пассивных элементов и выбор их типов;

- расчет выходных параметров системы и проверка их соответствия ТЗ.

Для любой аппаратной схемы существует некоторое множество подмножеств значений параметров компонентов, удовлетворяющее предъявляемым к ней техническим требованиям, так как вариации значений параметров могут компенсироваться и оказывать слабое влияние на выходные параметры.

Определение оптимального варианта производится путем вариации параметров: аналитически (формальными методами анализа), имитационно (с помощью моделей на ЭВМ) или натурно (на макете).

Расчет аппаратуры носит итерационный характер. После выполнения ряда расчетных операций может возникнуть необходимость повторить предыдущие операции для улучшения режимов всей аппаратуры или ее части. Например, расчет может показать необходимость введения дополнительных обратных связей, что потребует новых расчетов.

Следует отметить, что расчетные значения элементов необходимо заменять номинальными величинами, соответствующими стандартным шкалам, так как вероятность совпадения расчетного и номинального значений мала.

Выбор элементов производится на основании анализа справочной документации, при этом элементы должны соответствовать назначению, условиям эксплуатации и т.п. Элемент считается выбранным правильно, если номинальные значения его параметров находятся в допустимых отношениях с расчетными, а будущие условия эксплуатации элемента (температурный режим, значения влажности, уровень вибраций и др.) соответствуют его заводским ТУ [11].

3.5.4. Разработка печатных плат и макетирование.

Высокая стоимость аппаратуры и ее относительно низкая надежность делают актуальными такие конструктивные решения, при которых замена некоторой части изделия была бы легко осуществима. В связи с этим практически все аппаратные решения для встраиваемых систем выполняются на печатных платах.

Печатная плата представляет собой пластину из электроизоляционного материала, применяемую в качестве основания для установки и механического закрепления элементов, а также для их электрического соединения посредством печатного монтажа. Наиболее часто печатные платы изготавливаются из стеклотекстолита, на поверхность которого наклеена металлическая фольга. В процессе обработки платы на ее поверхности создают монтажные и крепежные отверстия и проводящий рисунок.

Применение печатных плат на производстве обеспечивает:

- идентичность параметров монтажа;
- высокую плотность размещения элементов;
- возможность автоматизации монтажных, сборочных и контрольно-регулировочных процессов.

Существует несколько технологий изготовления печатных плат, которые сводятся либо к применению только механических (как разновидность - электромеханических: электроискровых) способов обработки, либо комбинированных - фотохимических и механических.

При механическом способе изготовление происходит на одном станке и включает в себя следующие операции:

- подготовка управляющего файла для станка с ЧПУ (топология платы преобразуется в команды фрезерования и сверления);

- автоматическая сверловка заготовки;

- фрезерование изолирующих каналов (для ускорения процесса проводится не удаление всех пробельных участков, а оконтуривание проводников, выделение их из слоя фольги);

- металлизация переходных отверстий (путем вставок заклепок, втулок, впрыскивания электропроводящей пасты).

При химическом способе изготовление происходит на нескольких станках и включает в себя следующие операции:

- изготовление фотошаблонов. Рисунок чертежа проводников с помощью фотоплоттера переносится на прозрачную пленку - фотошаблон.

- сверление монтажных и переходных отверстий с помощью сверлильного станка с ЧПУ на основе информации об описании платы.

- экспонирование. Заготовка платы покрывается фоточувствительным материалом - фоторезистом (меняющим свою растворимость в зависимости от дозы поглощенного излучения), после чего она покрывается фотошаблоном и производится ее засветка мощным источником ультрафиолетового излучения. После экспонирования засвеченный фоторезист удаляется растворителем. (Как вариант может применяться негативное экспонирование с последующей металлизацией лишь токопроводящих участков платы);

- травление. Плата помещается в травильный раствор, который растворяет участки металлизации, не покрытые слоем фоторезиста. После травления на плате формируется готовый рисунок печатного монтажа.

- промывка и очистка.

- прессование многослойных плат, обрезка.

После изготовления платы в ряде случаев (при использовании компонентов для поверхностного монтажа) возможен автоматический монтаж элементов на плате с помощью робота-манипулятора (элементы фиксируются вязкой паяльной пастой на контактных площадках платы, после чего плата подвергается термической обработке, под воздействием которой элементы припаиваются к площадкам).

Параметры разрабатываемой аппаратной системы оптимизируют на ее физической модели (макете), номинальные параметры элементов которой соответствуют расчетным. Задача макетирования состоит в том, чтобы получить требуемые

значения выходных параметров, установить необходимые режимы работы, исследовать влияние внешних факторов на функционирование аппаратуры.

Экспериментальная отладка производится путем последовательного подбора параметров для каждого из функциональных элементов. Для сокращения сроков отладки необходимо применять методы планирования эксперимента, а также использовать оптимизацию на численных моделях с применением ЭВМ.

3.5.5. Средства автоматизированного проектирования аппаратуры.

В условиях возрастающей сложности аппаратуры ручное проектирование МК-систем практически не применяется; для проектирования используются системы автоматизированного проектирования (САПР).

Современные САПР решают задачи размещения элементов на плате, трассировки печатных проводников, формируют программы управления для станков ЧПУ, осуществляющих сверление отверстий, изготовление фотошаблонов, травление и др. операции. САПР существенно снижает затраты труда и обеспечивает высокое и стабильное качество проектирования.

В состав подавляющего большинства средств САПР входят (рис. 3-13):

- редактор принципиальных схем, позволяющий разработать чертеж схемы устройства;
- библиотека параметризируемых элементов, содержащая описание работы цифровых (вентили, регистры, счетчики, мультиплексоры, устройства памяти и др.), аналоговых (резисторы, конденсаторы, дроссели, трансформаторы и др.) и смешанных (АЦП, ЦАП и др.) устройств;
- редактор элементов, позволяющий сформировать описание отсутствующего в библиотеке компонента;
- редактор входных воздействий, позволяющий построить план проведения моделирования (в том числе, задать входные воздействия на систему);
- подсистема моделирования, выполняющая моделирование работы спроектированного устройства на основе его принципиальной схемы, библиотечных данных о примененных в схеме компонентах и построенного плана моделирования с выводом отчета о результатах моделирования;
- подсистема разработки печатных плат, позволяющая определить оптимальные размеры печатной платы, расположение на ней элементов и расположение соединительных проводников.

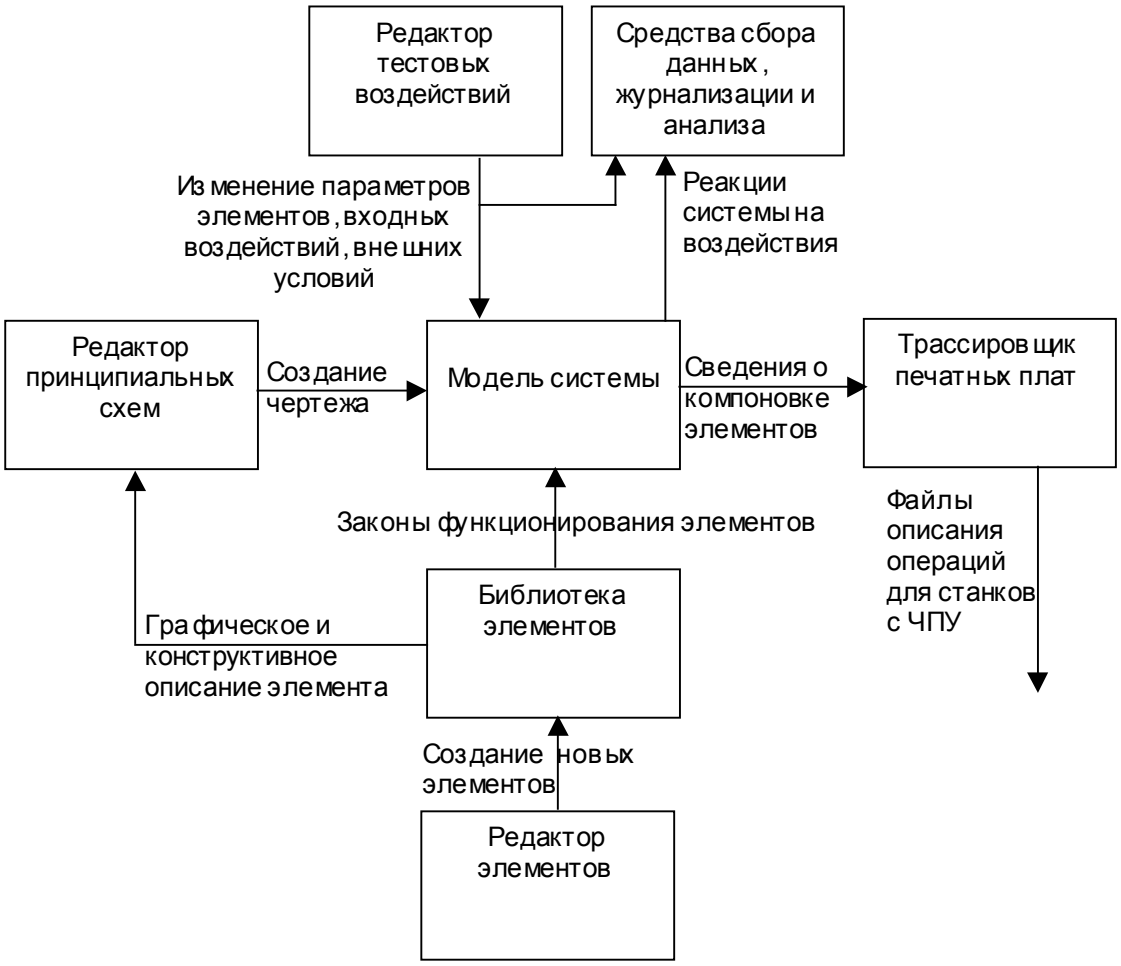


Рис. 3-13. Структура САПР электронных устройств.

К числу таких систем относятся пакеты P-CAD, DesignLab, MicroCap и др.

Рассмотрим пример их применения для проектирования аппаратуры.

В разрабатываемой микроконтроллерной системе необходимо реализовать вывод информации на регистратор-самописец, угол отклонения пера которого от положения покоя пропорционален уровню интенсивности подаваемого на него аналогового сигнала. Применить цифроаналоговый преобразователь в системе невозможно по причине нехватки выходных линий микроконтроллера, в связи с чем для передачи требуемой интенсивности сигнала используется метод широтно-импульсной модуляции, однако подавать ШИМ-сигнал непосредственно на вход самописца нельзя в связи с недостаточной инерционностью объекта.

Для разрешения указанной проблемы используется внешний пассивный интегратор, принципиальная схема которого, разработанная в системе проектирования DesignLab, показана на рис. 3-14.

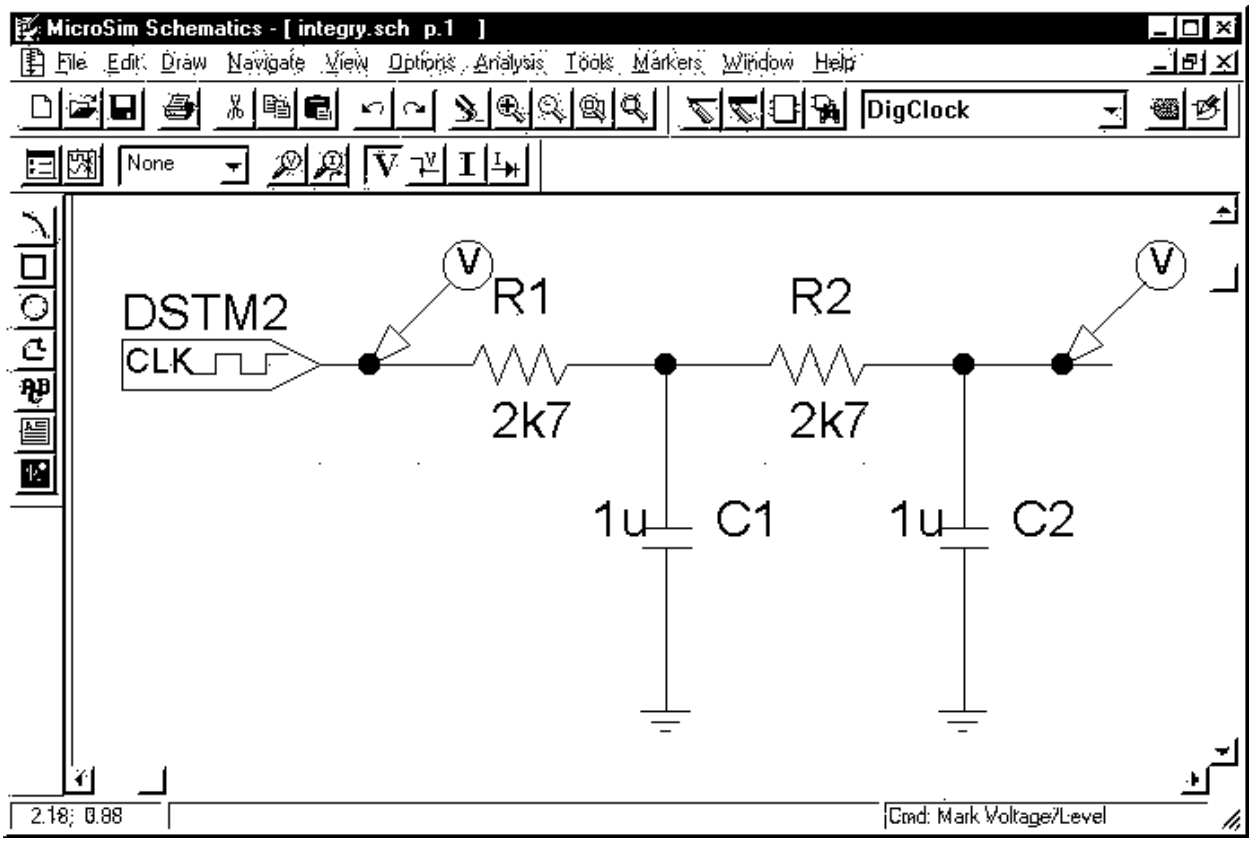


Рис. 3-14. Схема пассивного интегратора второго порядка.

Помимо собственно интегратора, образованного резисторами R1 и R2 и конденсаторами C1 и C2, для проведения моделирования в схему включен источник цифровых сигналов DSTM2 с изменяемыми параметрами длительности импульса и длительности паузы, позволяющий моделировать выходные сигналы микроконтроллера с различными значениями коэффициентов ШИМ, а также контрольные приборы V, дающие возможность регистрировать уровни напряжений в указанных точках.

На рис. 3-15 показан переходный процесс в системе при выводе сигнала с отношением длительности импульса к длительности периода, равным 0.9. На рис. 3-16 показано установившееся значение аналогового сигнала, равное 0.9 от максимально возможного. На рис. 3-17 показано установившееся значение аналогового сигнала для коэффициентов ШИМ, равного 0.5.

Примечание. Методика программного формирования ШИМ-сигналов на микроконтроллере будет рассмотрена в разделе 6.

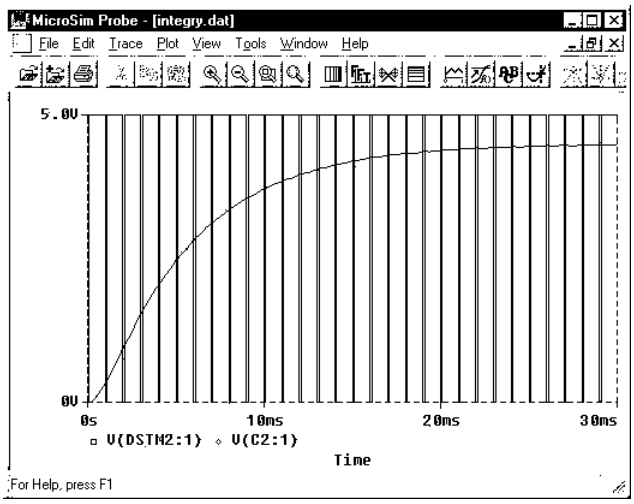


Рис. 3-15. Переходный процесс в системе при коэффициенте ШИМ 0.9.

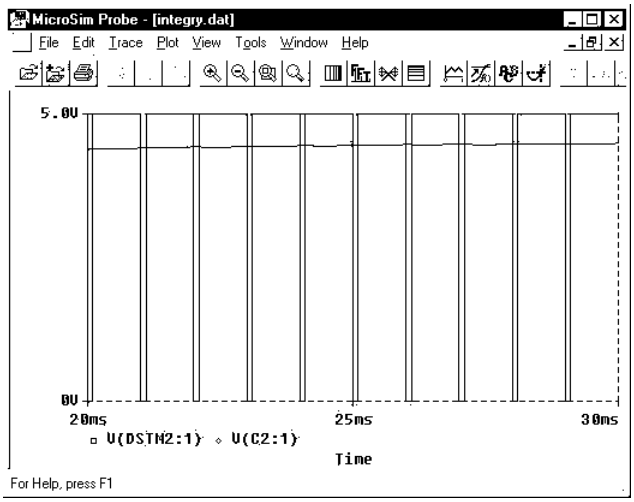


Рис. 3-16. Установившийся процесс при коэффициенте ШИМ 0.9.

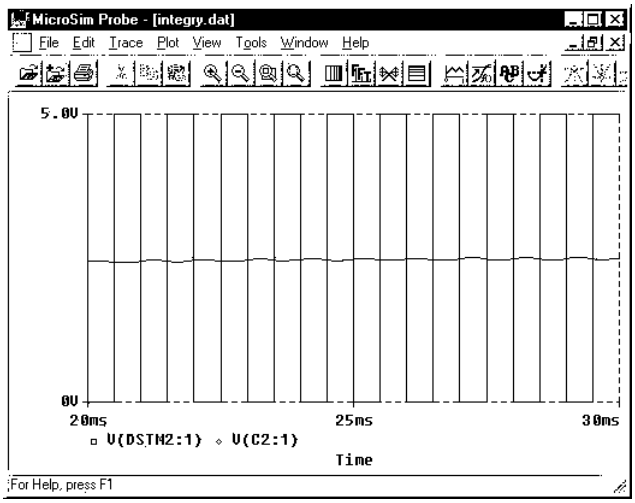


Рис. 3-17. Установившийся процесс при коэффициенте ШИМ 0.5.

3.6. Разработка программного обеспечения.

Под программным обеспечением микроконтроллерных систем подразумевается совокупность программ, используемых в процессе подготовки и решения задач на микроконтроллерных системах.

Структура программного обеспечения микроконтроллерных систем показана на рис. 3-18.

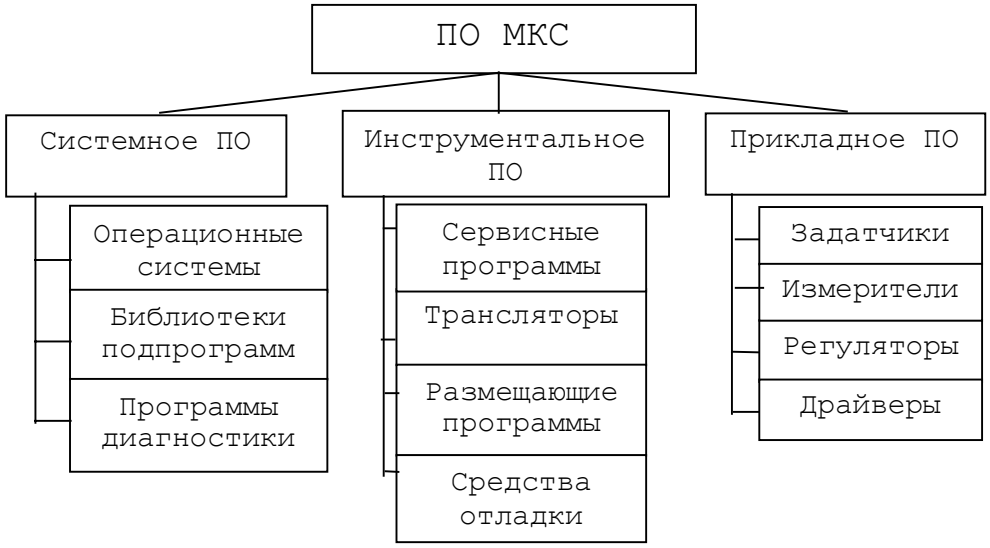


Рис 3-18. Структура ПО микроконтроллерных систем.

В соответствии с выполняемыми функциями программное обеспечение микроконтроллерных систем подразделяется на системное, инструментальное и прикладное.

К системному ПО относятся программы, предназначенные для организации вычислительного процесса в микроконтроллерной системе и контроля ее функционирования.

К инструментальному ПО относятся программы, применяемые для разработки программного обеспечения микроконтроллерных систем.

К прикладному ПО относятся программы, реализующие целевую задачу управления микроконтроллерной системы.

Программное обеспечение, предназначенное для выполнения на данной микроконтроллерной системе, называется резидентным программным обеспечением. К резидентному ПО относятся системное и прикладное ПО, а также ряд программ инструментального ПО.

Программное обеспечение, предназначенное для работы с данной микроконтроллерной системой, но выполняющееся на системах, программно несовместимых с данной, называется кроссовым ПО. К кроссовому ПО принадлежат ряд программ инструментального ПО.

3.6.1. Структура и функции системного ПО.

При необходимости предоставления пользователю возможностей манипулирования ходом вычислительного процесса на микроконтроллерной системе (загрузка новых модулей из инструментальной системы, изменение содержимого памяти данных и оперативной памяти программ, запуск модулей на выполнение, управление обменом информацией в мультимикроконтроллерных системах и др.) в состав программного обеспечения микроконтроллерной системы вводятся компоненты, обладающие способностью управления процессом выполнения задач на основе команд оператора. Такие программные системы, называемые операционными системами, содержат в своем составе: ядро, реализующее общую логику функционирования операционной системы (реакция на прерывания, исполнение команд и др.), модули диалога с пользователем (проведение которого предполагает наличие в составе микроконтроллерной системы устройств ввода-вывода информации), модули управления задачами (загрузкой, выгрузкой, переключением между задачами в мультизадачных системах, определением достижения условий останова и др.), а также модули связи с внешними вычислительными системами (в мультимикроконтроллерных системах, в системах с инструментальными ЭВМ и др.) (см. рис. 3-19).

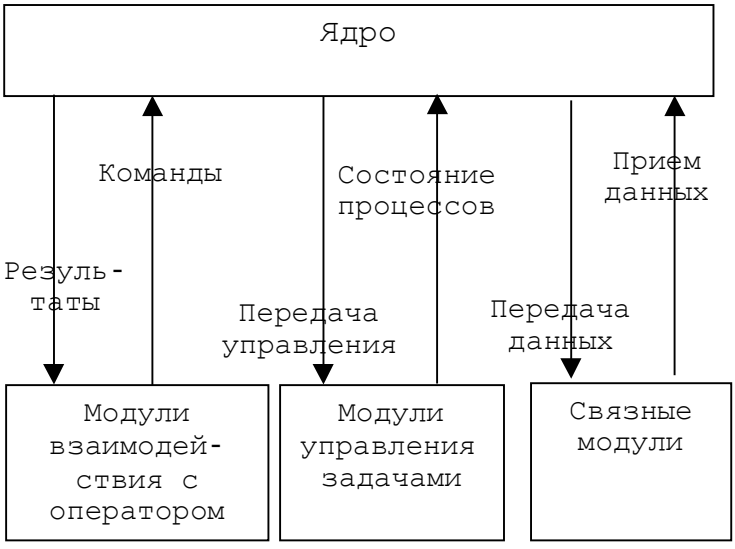


Рис. 3-19. Структура операционной системы МК.

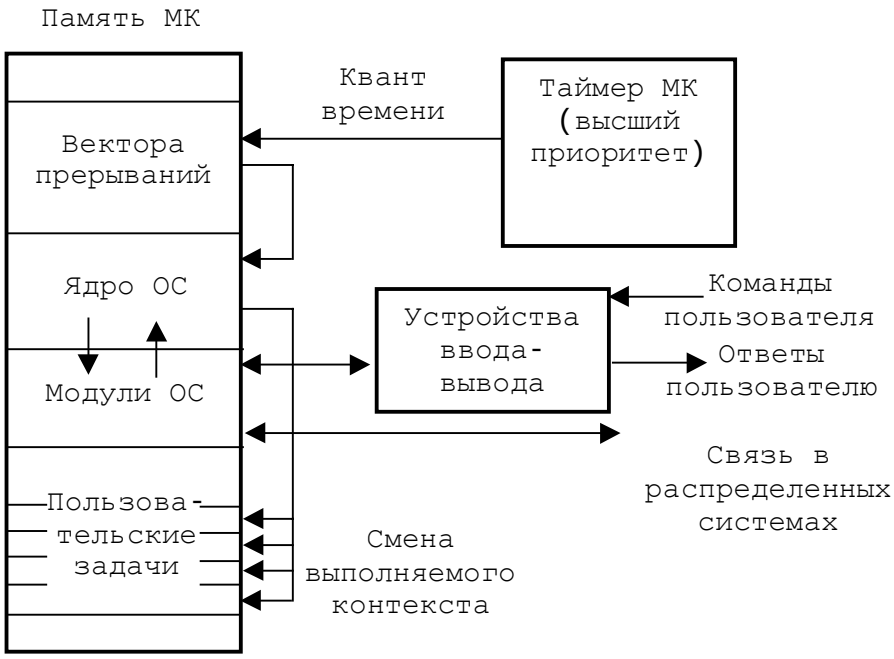


Рис. 3-20. Управление в микроконтроллерных системах с ОС.

Для осуществления своих функций ядро операционной системы, передача управления на которое осуществляется в соответствии с определенными временными метками, должно использовать высшие приоритеты системы обработки прерываний (рис. 3-20).

В отличие от операционных систем, библиотеки подпрограмм непосредственно не решают задачи управления вычислительным процессом, однако содержат набор функционально законченных модулей, выполняющих типовые действия по обмену информацией с устройствами ввода-вывода, математические функции и т.п.

Программы диагностики выполняют проверку исправности микроконтроллерной системы и ее составных частей. Проведение диагностических процедур описано в разделе 3.

3.6.2. Структура и функции инструментального ПО.

В состав инструментального ПО входят сервисные программы, трансляторы, размещающие программы и средства отладки.

К сервисным программам относятся редакторы, библиотекари, информационно-обучающие программы и другие.

Редакторы предназначены для подготовки исходных текстов программ микроконтроллерной системы. Редакторы принимают исходную программу с некото-

рого устройства ввода-вывода (клавиатура, дисковая память и пр.) и заносят в оперативную или дисковую память инструментальной ЭВМ. Для подготовки программ в инструментальных системах используются общецелевые редактирующие программы, которые могут использоваться для создания исходных программ на любом языке. Редактор оперирует с исходной программой как с текстом, не учитывая те синтаксические правила, которым должна удовлетворять программа.

Библиотекари предназначены для создания и ведения библиотек подпрограмм, реализующих типовые функции, часто применяемые в разработках. Библиотекари, как правило, имеют команды организации библиотеки, включения и удаления программных модулей, получения информации о содержащихся в библиотеке программных модулях.

Информационно-обучающие программы предназначены для начинающих пользователей и позволяют приобрести некоторые навыки применения инструментальных систем.

Трансляторы преобразуют исходную программу, написанную на входном языке (в качестве которого, как правило, применяются ассемблер и ряд языков высокого уровня, в частности, язык Си) в результирующую – так называемую объектную программу на языке кодов команд микроконтроллера. Кроме объектной программы трансляторы выдают листинг программы, содержащий распечатку исходной и объектной программ, таблицы идентификаторов, сообщения об ошибках и другие виды диагностической информации. Как правило, трансляторы обладают возможностью формировать объектную программу в перемещаемых адресах и обрабатывать внешние связи между программными модулями.

Размещающие программы преобразуют объектные программы к виду, непосредственно готовому к выполнению на микропроцессорной системе и заносят полученные программы в память МКС. Процесс перевода объектных программ к виду, пригодному для исполнения, заключается в преобразовании перемещаемого варианта объектной программы в вариант программы в абсолютных адресах. При необходимости установления связей между отдельными объектными модулями производится дополнительное редактирование внешних ссылок. Отметим, что сам процесс занесения программы в память микропроцессорной системы должен быть поддержан средствами как инструментальной вычислительной системы, так и самой МКС.

В качестве примера подобных систем можно привести пакет фирмы 2500 A.D. v4.02, включающий в себя макроассемблер X8051 для МКС, совместимых с однокристальными микроЭВМ семейства МК-51, библиотекарь и редактор связей, формирующий результирующие программы в ряде широко распространенных форматов.

Примечания.

Примечание 1. Директивы ассемблера X8051 приведены в Приложении 3.

Примечание 2. Средства отладки ПО, а также аппаратные средства размещения программ в памяти микроконтроллерной системы будут подробно рассмотрены в разделе 4-2.

3.6.3. Структура и функции прикладного ПО.

Прикладное ПО, реализующее процесс управления объектом, как любая сложная программная система строится по модульному принципу, в соответствии с которым каждая функционально законченная программная единица оформляется в виде подпрограммы, обращение к которой возможно из других программных модулей (см. рис. 3-21).

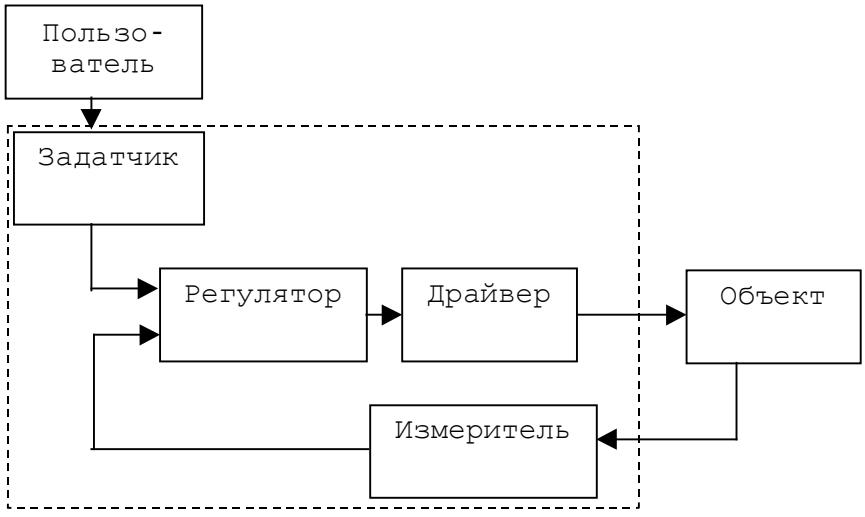


Рис. 3-21. Организация взаимодействия прикладного ПО.

Задающие модули (задатчики) реализуют функцию, описывающую желаемое (заданное) поведение объекта управления в виде набора ряда программно доступных переменных управления. Информацию, необходимую для формирования за-

дающей функции (например, выбор типа функции) задатчик может получать от оператора либо генерировать самостоятельно на основе информации об объекте управления. Функция, задающая желаемое поведение объекта, может быть реализована путем математических вычислений, либо путем обращения к участку памяти, хранящему набор значений данной функции. Критериями выбора одного из методов служат необходимая точность и требуемая скорость вычислений (например, sin(x)=x-x^3/3!+x^5/5!-… либо sin[x]=[0, 0.01, 0.03, 0.07, …1]).

Измерительные модули выполняют взаимодействие с аппаратурой датчиков объекта управления, предоставляя программам информацию о текущем состоянии объекта управления в виде значения ряда программно доступных переменных.

Регулирующие модули (регуляторы) предназначены для построения выдаваемого на объект воздействия на основе информации о требуемом поведении объекта управления и его реальном поведении. В настоящее время в рамках теории автоматического управления разработано достаточно большое количество способов регулирования. В качестве примера рассмотрим пропорционально-интегрально-дифференциальный регулятор (ПИД-регулятор).

ПИД-регулятор имеет два входа (задатчика и измерителя) и один выход (на драйвер). Уравнение, связывающее выход регулятора u(t) и величину рассогласования его входов $y=x_{зад}-x_{изм}$, следующее:

$$u(t) = K_{\text{П}} \cdot y(t) + \frac{1}{T_{И}} \int_0^t y(t)dt + T_{\text{Д}} \cdot \frac{dy(t)}{dt},$$

где $К_п$ - коэффициент передачи пропорциональной части регулятора, $Т_и$ - постоянная времени интегрирования, $Т_д$ - постоянная времени дифференцирования.

При квантовании с малой длительностью такта Т подобные уравнения можно преобразовать в разностные, заменив производные разностями соответствующих порядков, а интегралы - интерполяционными формулами. При использовании интерполяционной формулы по методу прямоугольников имеем:

$$u[n] = K_{\text{П}} \cdot y[n] + \frac{T}{T_{\text{И}}} \cdot \sum_{i=1}^{n} y[i] + \frac{T_{\text{Д}}}{\text{Т}} \cdot (y[n] - y[n-1]).$$

Драйверы объекта предназначены для преобразования информации об управляющем воздействии в управляющее воздействие того типа, которое соответствует объекту управления.

Так, например, для реализации управления частотой вращения шагового двигателя с тахогенератором в роли датчика частоты, задатчик формирует требуемое значение частоты вращения (в оборотах за секунду), измеритель определяет текущее значение частоты в тех же единицах, что и задатчик, осуществляя пересчет показаний аналого-цифрового преобразователя в количество оборотов в секунду, регулятор определяет уровень воздействия на объект (например, путем вычислений функции пропорционального регулятора), а драйвер преобразует полученный уровень воздействия в сигналы коммутации фазовых обмоток шагового двигателя, следующие с частотой, определяющейся информацией, полученной с регулятора.

Драйвер и измеритель принадлежат к объектно-зависимым компонентам прикладного ПО, а задатчик и регулятор - к объектно-независимым.

При смене объекта управления драйвер объекта также подлежит смене (в описанной системе замена шагового двигателя на двигатель постоянного тока потребует от драйвера объекта реализации управления, например, методом широтно-импульсной модуляции). При смене датчиков объекта замене подлежит измеритель.

3.6.4. Стадии разработки программного обеспечения.

При проектировании программного обеспечения микроконтроллерных систем управления техническим объектом разработчику необходимо осуществить продвижение проекта через ряд стадий [7]:

1. постановка задачи;
2. анализ проблемы и построение математической модели;
3. построение алгоритмов решения задачи;
4. проектирование программы, решающей поставленную задачу;
5. программирование задачи;
6. проведение автономной отладки;
7. проведение комплексной отладки;
8. передача в эксплуатацию;
9. сопровождение.

Первый этап предполагает формирование описания (на естественном языке либо на языке специальных символов) условий задачи и желаемого результата. Как правило, постановка задачи представляется в виде технического задания.

На втором этапе осуществляется построение математической модели того физического процесса, который описывается в постановке задачи.

На третьем этапе математическая модель представляется в виде, удобном для числовой оценки, и осуществляется выбор метода решения задачи. На двух первых этапах описывалась сущность, которую необходимо получить в процессе проектирования, начиная с третьего этапа формируется способ реализации этой сущности - алгоритм получения результата на основе исходных данных.

На четвертом этапе осуществляется проектирование программного комплекса, завершающееся разработкой спецификаций требований к составным частям программного комплекса.

На пятом этапе выполняется собственно программирование, т.е. кодирование алгоритма с помощью выбранного языка программирования.

На шестом этапе выполняется автономная отладка каждой программы, входящей в состав программного комплекса, т.е. достижение правильности реализации функции, возложенной на данную программу.

На седьмом этапе проверяется соответствие техническому заданию всего программного комплекса.

На восьмом этапе разработанный программный комплекс передается в эксплуатацию с изготовлением необходимой документации.

Процесс сопровождения заключается в устранении обнаруженных ошибок, в модификации и улучшении применяемых подпрограмм.

Следует отметить, что в микроконтроллерных системах заключительная передача в эксплуатацию разработанного программного комплекса требует проведения комплексной отладки аппаратного и программного обеспечения (см. рис. 3-5).

Примечание.

Главными факторами, определяющими выбор между языком программирования высокого уровня или языком ассемблера, являются технические (объем программ и время их выполнения) и экономические (стоимость конечного изделия) факторы. При использовании трансляторов с языков высокого уровня объем и время выполняемой программы больше, чем при использовании языка ассемблера, однако затраты на разработку программного комплекса и время его изготовления будут меньше. Стоимость микроконтроллерной системы может быть представлена суммой

стоимости программного и аппаратного обеспечения. При изготовлении партии систем стоимость разработки программного обеспечения остается постоянной, а стоимость аппаратуры растет пропорционально числу изготавливаемых систем (см. рис. 3-22).

Так как стоимость разработки программного обеспечения ниже при использовании языка высокого уровня, но каждое аппаратное изделие при этом дороже (программы больше по объему, ряд функций в связи с недостаточной скоростью выполнения программ необходимо реализовать аппаратно и т.п.), то, начиная с некоторого количества изделий в партии экономически выгоднее использовать язык ассемблера.

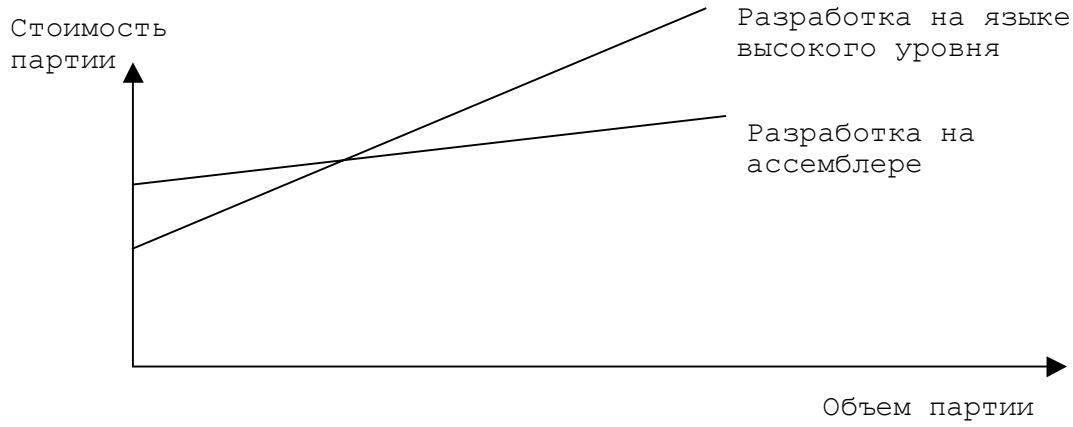


Рис. 3-22. Стоимость партии в зависимости от языка разработки программного обеспечения.

РАЗДЕЛ 4. ОТЛАДКА МИКРОКОНТРОЛЛЕРНЫХ СИСТЕМ.

Под отладкой понимается процесс поиска, обнаружения и исправления ошибок в разрабатываемой системе.

Отладка микроконтроллерных систем включает в себя стадии отладки аппаратных средств, программных средств, и комплексную совместную отладку аппаратуры и программного обеспечения.

4.1. Средства и методы отладки аппаратуры МК-систем.

Для проведения отладки аппаратных средств микроконтроллерных систем применяются аппаратные и программные средства отладки (см. рис. 4-1).

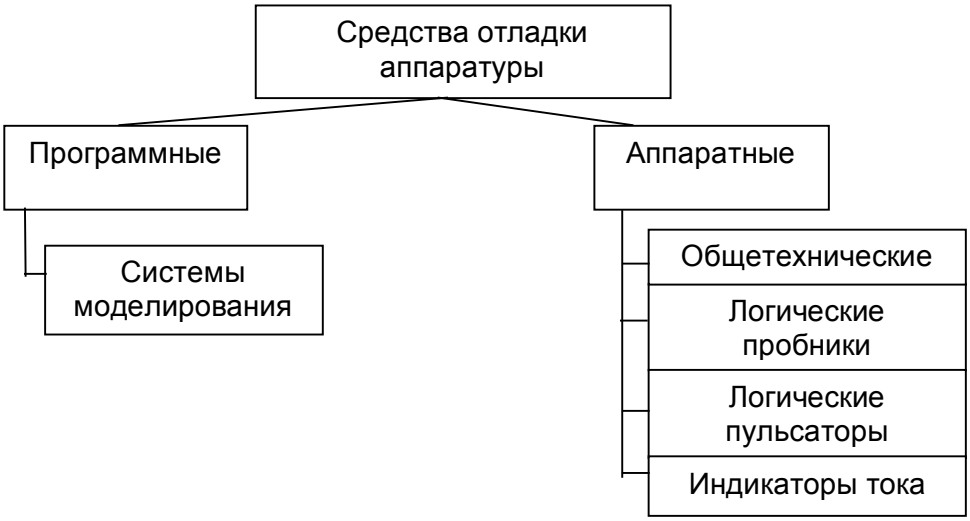


Рис. 4-1. Средства отладки аппаратуры.

Программные системы моделирования аппаратных средств позволяют исследовать работу аппаратной части микроконтроллерной системы и выявить ошибки в функционировании аппаратуры без проведения трудоемкого макетирования.

Как отмечалось в разделе 3.5, одной из компонент современных систем проектирования аппаратных средств является подсистема моделирования, выполняющая моделирование работы спроектированного устройства на основе его принципиальной схемы, библиотечных данных о примененных в схеме компонентах и построенного плана моделирования.

При разработке пользовательских устройств (как правило, это устройства сопряжения с объектом) проведение отладки путем моделирования и сопоставления

его результатов с требуемым поведением системы позволяет существенно сократить трудозатраты и временные потери.

По окончании этапов моделирования и разработки принципиальной схемы аппаратного обеспечения и формирования топологии платы проводится построение макета системы и проводится автономная отладка аппаратуры.

4.1.1. Общетехнические средства отладки аппаратуры.

В состав общетехнических средств отладки входят контрольно-измерительные приборы, предназначенные для использования в любых электрических схемах. К их числу относятся мультиметры, частотомеры и осциллографы.

Мультиметр предназначен для измерения постоянных и переменных периодических напряжений и токов, а также значений сопротивлений, емкостей и индуктивностей. С помощью мультиметра выявляются ошибки в проектировании блоков электропитания (в частности, ситуации несоответствия уровня питающего напряжения номинальному диапазону питающих напряжений микросхем), а также факты ошибок монтажа - обрывы и короткие замыкания проводников.

Частотомер предназначен для определения значений частоты следования периодических сигналов. С помощью частотомера выявляются ошибки проектирования блока системной синхронизации.

Осциллограф предназначен для регистрации электрических сигналов в виде графика функции зависимости уровня сигнала от времени. С помощью осциллографа выявляются сбои в работе устройств синхронизации, фиксируются факты гонок (непредусмотренных задержек одного сигнала по отношению к другому в связи с различным распространением сигналов в схеме) и т.п.

Особенности микроконтроллерных систем управления (параллельное представление информации, высокая скорость смены состояний и т.п.) ограничивают область применения общетехнических средств отладки и число разрешаемых с их помощью задач, в связи с чем в микроконтроллерных системах широко применяются специализированные средства.

4.1.2. Логические пробники.

В современных цифровых системах, как правило, применяется двоичная система счисления, в связи с чем для задания цифровых сигналов применяется два уровня напряжения - высокий и низкий, разделенные зоной неопределенности. Уро-

вень напряжения цифрового сигнала номинально соответствует одному из разрешенных уровней. Его переход в зону неопределенности свидетельствует о ненормальной работе участвующих в его формировании элементов, и воспринимается как заранее непредсказуемое цифровое значение. Так, для ТТЛ-входа область логического нуля лежит в диапазоне 0..0,8В, а область логической единицы - в диапазоне 2..5В.

Устройство логического пробника показано на рис. 4-2, а его внешний вид - на рис. 4-3.

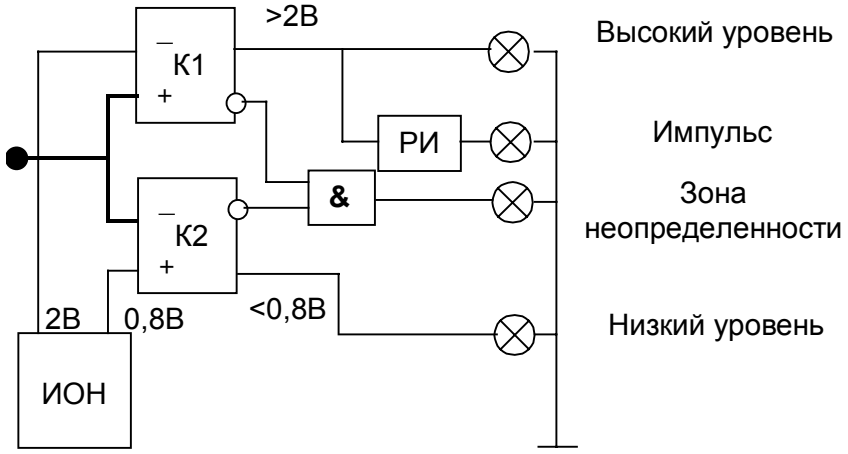


Рис. 4-2. Устройство логического пробника.

Рис. 4-3. Внешний вид логического пробника.

С помощью источника опорного напряжения ИОН и компараторов К1 и К2 выявляются следующие ситуации:

- уровень напряжения входного сигнала более 2 В;
- уровень напряжения входного сигнала менее 0,8 В;
- уровень напряжения входного сигнала лежит в диапазоне от 0,8 до 2В.

Указанные события активизируют соответствующие индикаторы. Схема расширителя импульсов РИ позволяет отобразить ситуацию прохождения высокочастотных импульсов.

В микроконтроллерной системе логический пробник применяется для проверки питания микросхем, определения логических уровней сигналов в различных участках схемы, установления наличия импульсов в цепях, определения целостности проводников.

4.1.3. Логические пульсаторы.

Для анализа работы цифровой схемы в ряде случаев необходимо определить поведение системы в ответ на некоторое воздействие, формируемое внешними по отношению к системе элементами и подаваемое на вход интересующего элемента. Логические пульсаторы используются для введения в узел мощных (током до 1А) коротких (длительностью до 100 нс) импульсов, переводящих его из одного логического состояния в другое, а затем возвращающие в первоначальное состояние. Параметры вводимого импульса таковы, что не повреждают цифровую схему, в связи с чем логические пульсаторы можно использовать для анализа без демонтажа элементов.

Устройство логического пульсатора показано на рис. 4-4.

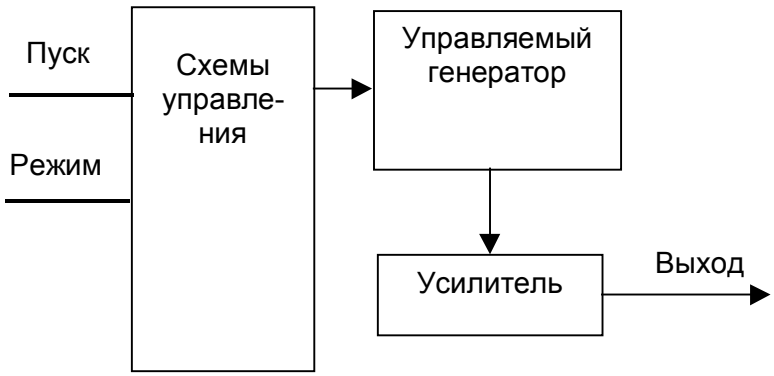


Рис. 4-4. Устройство логического пульсатора.

Пользователь может задать несколько режимов работы пульсатора. Как правило, в их число входят:

- формирование одиночного импульса;
- формирование непрерывной последовательности импульсов с заданной частотой;
- формирование пакета заданного количества импульсов заданной частоты.

С помощью логического пульсатора легко установить счетчики в требуемое состояние, чтобы затем проследить работу системы с данного момента.

В совокупности с логическим пробником логический пульсатор позволяет контролировать целостность проводников и работу отдельных узлов системы так,

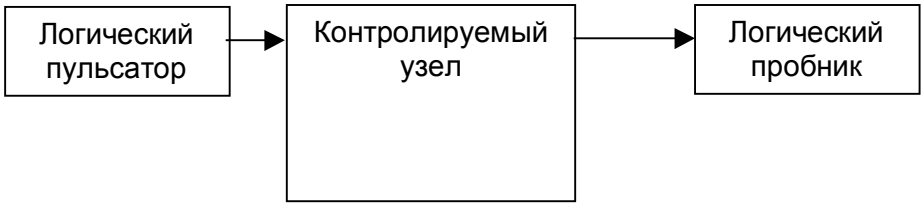


Рис. 4-5. Совместное использование логического пробника и логического пульсатора.

как показано на рис. 4-5.

4.1.4. Индикаторы тока.

В ряде случаев для анализа работы цифровой схемы необходимо определить наличие протекания тока в проводнике. Для этих целей применяется индикатор тока, работа которого основана на явлении электромагнитной индукции.

Устройство индикатора тока показано на рис. 4-6.

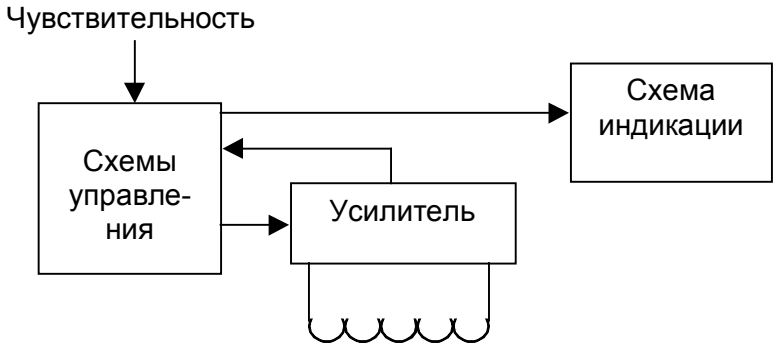


Рис. 4-6. Устройство индикатора тока.

Интенсивность свечения индикатора устройства пропорциональна величине скорости изменения тока в проводнике, над которым располагается катушка индуктивности индикатора тока. Задание чувствительности индикатора тока позволяет определить относительные величины протекающих в отлаживаемом устройстве токов.

В совокупности с логическим пульсатором индикатор тока позволяет контролировать целостность проводников и работу отдельных узлов системы так, как

показано на рис. 4-7. Принцип отладки заключается в том, что при наличии замыканий проводников или неисправных элементов с пробоем выходных каскадов путь основной части тока отличается от предполагаемого.

Следует отметить, что с помощью описанных приборов провести детальный анализ системы с шинной структурой, такой, как микроконтроллерная система, в

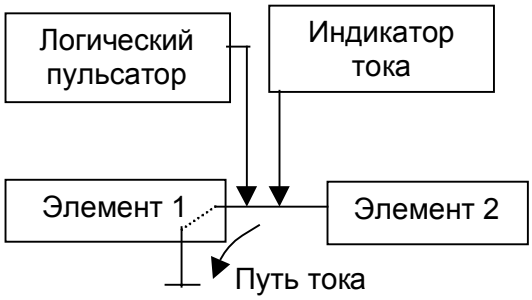


Рис. 4-7. Совместное использование логического пульсатора и индикатора тока.

ряде случаев не удается, в связи с чем необходимо использовать более сложные технические средства.

4.2. Средства и методы отладки программного обеспечения.

По завершению стадии программирования микропроцессорной системы процесс разработки программного обеспечения переходит на следующую стадию – стадию отладки, во время которой выявляются семантически неверные с точки зрения управления объектом конструкции программного обеспечения.

На данном этапе могут применяться программные и программно-

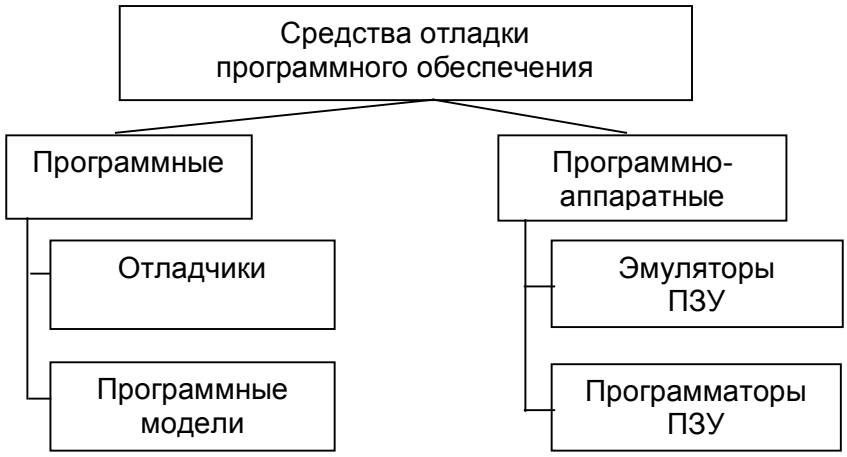


Рис. 4-8. Средства отладки программного обеспечения.

аппаратные средства (см. рис. 4-8).

4.2.1. Отладчики.

Отладчик представляет собой резидентную программу, выполняющую следующие действия:

- запуск программы на выполнение с указанного адреса;
- приостановка выполнения программы по достижении одной из заданных команд или при выполнении заданного условия;
- вывод на пульт оператора содержимого требуемой области памяти и/или регистров процессора;
- изменение с пульта оператора содержимого требуемой области памяти и/или регистров процессора.

Более сложные отладчики позволяют получать текущую информацию о ходе вычислительного процесса (адреса контрольных точек, содержимое стековой памяти) и контролировать действия выполняемой программы на допустимость.

Отметим, что для использования отладчика в МК-системе необходимо наличие средств взаимодействия с оператором (табло, клавиатура и т.п.).

4.2.2. Программные модели.

Программная модель (симулятор) представляет собой комплекс программ, размещаемый на инструментальной ЭВМ, позволяющий моделировать работу микроконтроллерной системы, для которой производится разработка программного обеспечения, непосредственно на ЭВМ. Для этих целей каждая команда для процессора микроконтроллера эквивалентируется набором команд инструментальной ЭВМ, выполняемых над областью данных, эквивалентирующих пространство памяти микроконтроллерной системы (см. рис. 4-9).

Программа, пройдя стадию трансляции, обработанная редактором связей и превращенная в двоичный или шестнадцатеричный формат, поступает на вход программной модели. В процессе отладки оператор имеет возможность запускать и приостанавливать выполнение программы, наблюдать за содержимым симулируемой памяти, контролировать и задавать состояние периферийных узлов симулируемой микроконтроллерной системы.

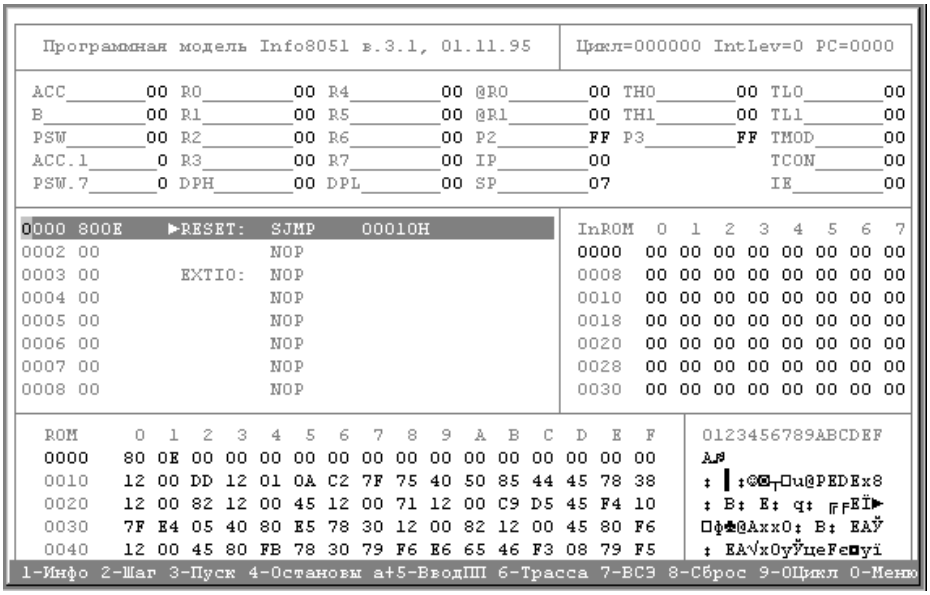


Рис. 4-9. Вид экрана программной модели.

4.2.3. Эмуляторы ПЗУ.

Аппаратура эмулятора ПЗУ - наиболее простого средства отладки программного обеспечения (см. рис. 4-10) - имеет в своем составе блок оперативной памяти, подсоединяемый на место внешней памяти программ отлаживаемой микроконтроллерной системы, устройство управления, а также средства связи с инструментальной ЭВМ. Такой комплекс пригоден для отладки микроконтроллеров с внешней памятью программ.

Программное обеспечение инструментальной ЭВМ позволяет заносить в память эмулятора пользовательские программы.

Взаимодействуя с программным обеспечением эмулятора ПЗУ, пользователь обладает возможностью указать файл, содержащий исполнимый код программы, размещаемой в эмулируемом ПЗУ.

Эмулятор принимает от инструментального ПО ЭВМ адреса, по которым в ОЗУ следует разместить данные, представляющие собой исполнимый код программы, а также сами данные. Установив необходимые сигналы на входах ОЗУ, эмулятор заносит информацию в требуемую ячейку.

По окончании размещения программы в ОЗУ эмулятор подключает к нему линии микроконтроллерной системы, позволяя микроконтроллеру работать с хранимой в ОЗУ информацией.

При обнаружении ошибок в исполняемой программе производится возврат на предыдущие этапы разработки (анализ предметной области, алгоритмизацию или

программирование), после чего программа с внесенными в нее изменениями вновь заносится в оперативную память эмулятора ПЗУ для исполнения. Подобная тактика проведения отладки не требует выполнения циклов стирания предыдущей версии программы из ПЗУ, что сокращает временные затраты и сохраняет ограниченные по числу циклов стирания-записи ресурсы ПЗУ микроконтроллерной системы.

Примечания.

Примечание 1. Следует отметить, что подобная тактика отладки программного обеспечения имеет ограничения на степень локализации ошибки, не позволяя, в частности, определить конкретный оператор, приводящий к неверной работе программы.

Примечание 2. Эмулятор ПЗУ позволяет выполнять элементы комплексной отладки аппаратуры и программного обеспечения, поскольку отлаживаемая программа может быть размещена в проектируемой микроконтроллерной системе, при этом выполнение отлаживаемой программы (ввод-вывод через порты, работа с внешними источниками прерываний и пр.) и работа отлаживаемой аппаратуры оказывают взаимное влияние.

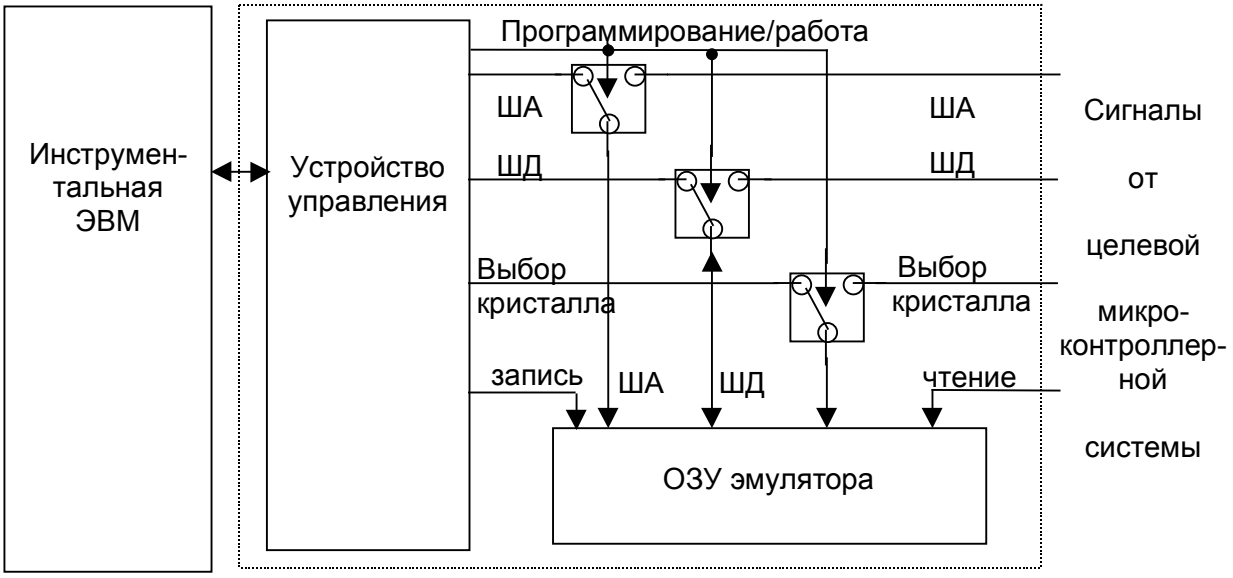


Рис. 4-10. Аппаратура эмулятора ПЗУ.

4.2.4. Программаторы.

Как уже отмечалось ранее, завершающим этапом проектирования программного обеспечения, позволяющим осуществить переход к комплексной отладке

аппаратуры и программного обеспечения, является размещение разработанной программы в постоянной памяти микроконтроллерной системы, как и предполагается для конечного изделия.

Приведем классификацию устройств постоянной памяти:

- по отношению к микросхеме микроконтроллера: внешняя и внутренняя;

- по количеству допустимых циклов "стирание - программирование": однократно программируемая и многократно программируемая;

- по способу доступа: последовательная и параллельная.

Программаторы позволяют выполнять размещение пользовательских программ в различных по приведенной классификации типах постоянной памяти.

Таким образом, программатор позволяет, с одной стороны, выполнить подготовку финальной версии ПО, а с другой стороны - провести заключительный этап отладки ПО в реальном аппаратном окружении.

Структура и внешний вид программатора показаны на рис. 4-11 и 4-12.

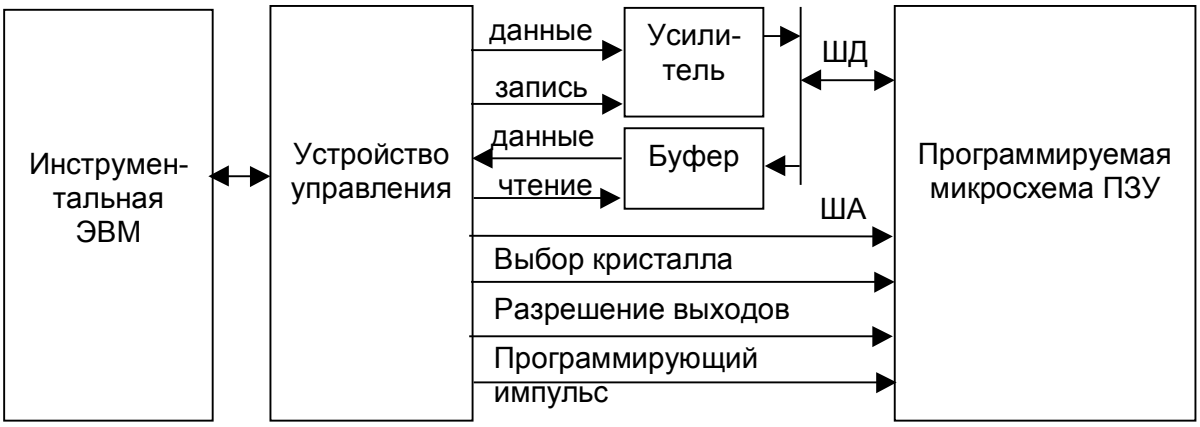


Рис. 4-11. Устройство программатора.

Взаимодействуя с программным обеспечением программатора, пользователь обладает возможностью указать:

- режим работы программатора: чтение ПЗУ, программирование, верификация;

- файл, содержащий исполнимый код программы, размещаемой в программируемом ПЗУ (для режима программирования и верификации) либо файл, в который будет помещаться информация из ПЗУ (для режима чтения);

- алгоритм программирования ПЗУ (щадящий, нормальный и пр.).

Программатор принимает от инструментального ПО ЭВМ команды (указания пользователя) и информацию, предназначенную для размещения в памяти ПЗУ.

Рис. 4-12. Внешний вид программатора.

Например, в режиме записи программатор выполняет следующую последовательность шагов:

1. Принять от ЭВМ адрес А;

2. Принять от ЭВМ данные Д;

3. Считать ячейку А ПЗУ;

4. Если ячейка не пуста - выдать сообщение "микросхема не стерта" и прекратить текущий сеанс программирования;

5. Если ячейка пуста, установить на запись данные Д;

6. Выдать серию программирующих импульсов;

7. Считать байт К из ячейки А ПЗУ;

8. Если К=Д, перейти на шаг 1;

9. Если не исчерпано количество попыток программирования ячейки, определяемое алгоритмом записи, вернуться на шаг 6;

10. Если количество попыток исчерпано - выдать сообщение "микросхема неисправна" и прекратить текущий сеанс программирования.

Занесенная в память микроконтроллерной системы программа исполняется в реальном аппаратном окружении в реальных временных режимах. Несмотря на то, что, как и в случае эмулятора ПЗУ, возможности отладки программного обеспечения ограничены, возможности комплексной отладки аппаратуры и программного обеспечения с программатором шире, чем эмулятором ПЗУ, так как в данном случае

временные характеристики доступа к внешней памяти соответствуют характеристикам конечного изделия.

4.2.5. Методика отладки программного обеспечения.

Как следует из статистических данных, на стадиях проектирования программного обеспечения допускается порядка 60% всех ошибок проекта, приходящихся на программное обеспечение, и около 40% ошибок допускается на стадии реализации. При этом в процессе отладки обнаруживаются лишь 40% допущенных ошибок, а 60% допущенных ошибок выявляются в процессе эксплуатации программного комплекса. В соответствии со статистикой, вероятность правильного исправления ошибок со временем снижается, а стоимость данного исправления возрастает [7] (см. рис. 4-13).

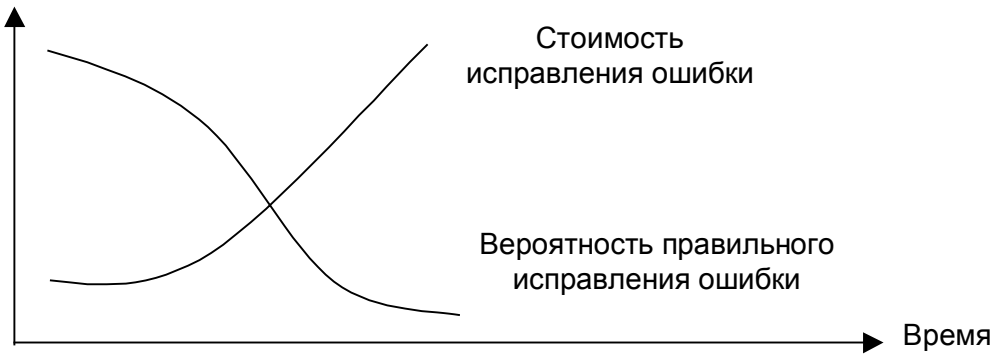


Рис. 4-13. Основания для раннего обнаружения ошибок.

В отличие от аппаратного обеспечения, надежность достоверного функционирования которого со временем снижается, ненадежность программного обеспечения является следствием внесения в него на этапе разработки ошибок. По мере их обнаружения и исправления надежность программного обеспечения повышается, однако, как правило, частота отказов в программном обеспечении не снижается до нуля, а колеблется вблизи некоторого установившегося значения, что объясняется теоретической возможностью внесения в программное обеспечение новых ошибок при попытке исправления старых (см. рис. 4-14).

В зависимости от объекта отладки различают синтаксическую и семантическую отладку. Синтаксическая отладка предназначена для выявления в программе конструкций, не соответствующих синтаксису применяемого языка программирования. Вопросы синтаксической отладки изучаются в теории формальных грамматик и

теории языков и в настоящее время реализованы в транслирующих системах инструментального программного обеспечения.

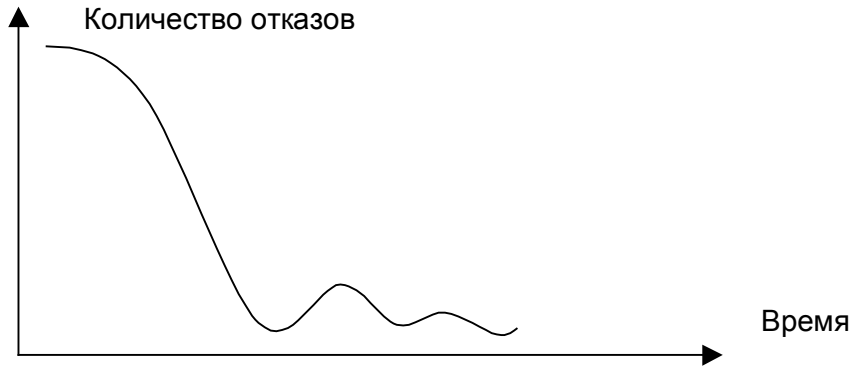


Рис.4-14. Зависимость надежности отлаживаемого программного обеспечения от времени.

Целью семантической отладки является выявление и корректировка в программе ошибок, не позволяющих программе выполнять заданную функцию преобразования входных данных в выходные. Этап семантической отладки проводится по завершению этапа синтаксической отладки.

Существует несколько способов классификации методов семантической отладки программного обеспечения.

Приведем два способа классификации:

- по способу получения результатов выполнения программы: статические методы (не требующие выполнения программы в процессе ее отладки) и динамические методы (требующие выполнения программы в процессе отладки);

- по характеру воздействия на исходный текст программы: разрушающие (предполагающие модификацию исходного текста с целью введения отладочных операторов, процедур, макрокоманд и др.) и неразрушающие (не предполагающие модификации исходного текста программы).

Эффективность применяемых методов отладки программного обеспечения оценивается по степени отлаженности программы и стоимостью обнаружения ошибки.

Степень отлаженности программы в зависимости от специфики задачи, реализуемой отлаживаемой программой и требований к ней может выражаться в следующем виде:

- количество выявленных ошибок в программе;
- количество оставшихся ошибок в программе;

- среднее количество оставшихся ошибок в программе, приходящихся на сто операторов исходного текста;
- количество пройденных при тестировании ветвей или операторов программы;
- гарантированное время наработки программы на отказ;
- среднее время наработки программы на отказ;
- число ошибок, обнаруживаемых за постоянное время в режиме эксплуатации программы;
- прочие оценки.

Стоимость обнаружения ошибки оценивается на основе затрачиваемых на обнаружение временных, стоимостных, материально-технических ресурсов.

Для обнаружения ошибки в программном обеспечении на основе текста программы и представления об алгоритме решения задачи, программисту следует выполнить анализ некоторой совокупности объектов программы (переменных, операторов, управляющих и информационных связей между ними), объем которой может быть достаточно велик. Особенности психологического восприятия информации человеком накладывают ограничения на количество одновременно воспринимаемых объектов на уровне 5..9 объектов.

Используемые методы отладки служат для автоматизированного выделения необходимой для обнаружения ошибки информации и представления ее в виде, удобном для восприятия человеком. При этом процесс отладки является итерационным, на каждом этапе которого программист воспринимает ограниченный объем информации о программе, позволяющий сделать некоторые частичные обоснованные выводы об ошибке и определить направление сокращения информации на последующих итерациях процесса отладки.

Таким образом, задача отладки сводится к построению процедур сокращения объема информации о программе до уровня, на основе которого программист в состоянии сделать обоснованные выводы относительно наличия или отсутствия ошибки в программе, а в случае наличия - о характере и местоположении. При этом процедуры должны обеспечивать наибольшую степень отлаженности программы и наименьшую стоимость обнаружения ошибки.

Исходя из сути семантических ошибок, процесс отладки программ любым методом сводится к следующей схеме:

- формирование гипотезы о наличии в программе ошибки;

- формирование на базе внешнего по отношению к программе источника информации описания функций программы;
- формирование на основе самой программы описания выполняемых ею функций;
- выполнение сравнения двух полученных описаний;
- построение на основе проведенного сравнения заключения об отсутствии или наличии ошибки, ее характере и местонахождении.

Таким образом, для поиска семантических ошибок в программе необходимо внешнее по отношению к самой программе описание решаемой задачи. Такими описаниями являются:

1. Техническое задание на программу, содержащее требования к функционированию программы (ограничения, описания функций, входные и выходные данные и т.п.);

2. Описание математической модели решаемой задачи (в данном формализованном описании задачи выделены все исследуемые параметры, входные и выходные данные, процессы преобразования информации, критерии оценки результатов и т.п.);

3. Алгоритм решения задачи (описывает метод решения задачи и фиксирует все процессы преобразования входных данных в выходные так, как это должно быть реализовано в программе);

4. Проект программы (представляет программу в виде блок-схем или другим способом, уточняя описание алгоритма в терминах, близких к конструкциям используемого языка программирования);

5. Данные заказчика (дополнительные уточняющие сведения о решаемой задаче, деталях технического задания и т.п.).

Типовыми методами семантической отладки программного обеспечения являются метод тестирования, метод верификации и метод инспекций исходного текста.

В общем случае задача эффективного поиска семантических ошибок в программах является алгоритмически неразрешимой, в связи с чем каждый из методов решает практическую задачу - поиск некоторого подмножества семантических ошибок, определяемого критерием отладки данного метода.

В таблице приведено сравнение указанных методов отладки.

	Методы отладки		
	Тестирование	Верификация	Инспекции текста
Критерии отладки	Прохождение некоторого множества путей на графе программы.	Установление факта совпадения результата преобразования входных данных с требуемой выходной функцией.	Поиск в программе ошибок, включенных в каталог.
Характеристики обнаруживаемых ошибок	Позволяет обнаружить ошибки, находящиеся на исследуемых структурах программы.	Позволяет обнаружить ошибки, находящиеся на верифицируемых участках программы.	Позволяет обнаружить ошибки, включенные в каталог.
Зависимость эффективности от выделяемых ресурсов	Увеличение выделяемых на отладку ресурсов позволяет увеличить сложность алгоритмов тестирования (в частности, определяющих выбор входных данных для прохождения различных путей графа).	Увеличение выделяемых на отладку ресурсов позволяет увеличить число верифицируемых участков программы.	Увеличение выделяемых на отладку ресурсов позволяет увеличить число реализуемых алгоритмов поиска ошибок, внесенных в каталог.
Причины ограничения метода	Невозможно построить алгоритм, позволяющий для любой программы найти входные данные, определяющие прохождение всех реализуемых путей графа.	Невозможно полностью снять ограничения, налагаемые на отлаживаемые верификацией программы.	Невозможно составить каталог, содержащий все ошибки.
Стратегия отладки	Последовательность обнаружения ошибок должна соответствовать вероятности прохождения ветвей графа программы.	Последовательность обнаружения ошибок должна соответствовать простоте проведения верификации.	Последовательность обнаружения ошибок должна соответствовать их вероятности появления в программе.

На рис. 4-15 показаны подмножества ошибок, обнаруживаемых каждым из методов. Как следует из рисунка, в общем случае в программе останутся необнаруженные ни одним из методов ошибки. Использование нескольких методов совместно повышает отлаженность программы.

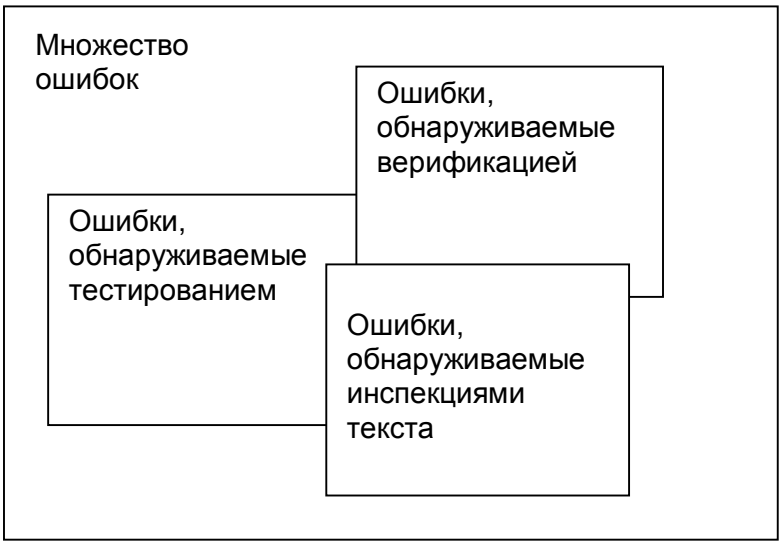


Рис. 4-15. Ошибки, обнаруживаемые различными методами.

На рис. 4-16 показана стоимость обнаружения ошибок данными методами.

В зависимости от количества ошибок в программе, эффективным в плане стоимости является один из используемых методов. Кривая, выделенная жирной линией, показывает эффективную стратегию отладки.

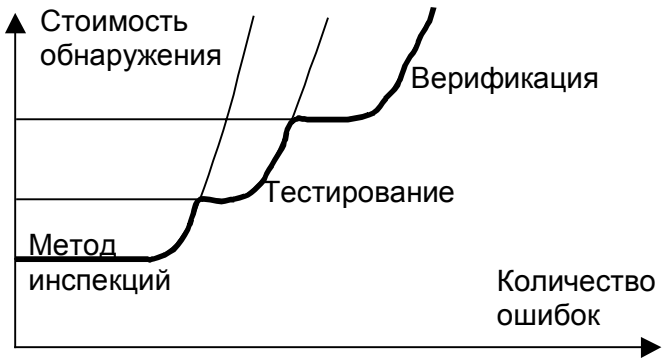


Рис.4-16. Изменение стоимости обнаружения ошибок различными методами в зависимости от их количества.

Одним из важных условий качественного проектирования и эффективной отладки программного обеспечения является знание разработчиком типовых семантических ошибок, допускаемых при программировании данного класса задач в вычислительной системе данного типа.

4.2.6. Каталог семантических ошибок программирования.

Анализ семантических ошибок программирования и статистики их появления показывает, что, несмотря на многочисленность факторов, определяющих структуру и свойства программ, существует ограниченное число устойчивых ошибок программирования, присутствующих в большинстве реальных программ.

Исходя из сложности поиска и устранения, ошибки в программах следует разделить на три группы (см. рис. 4-17).

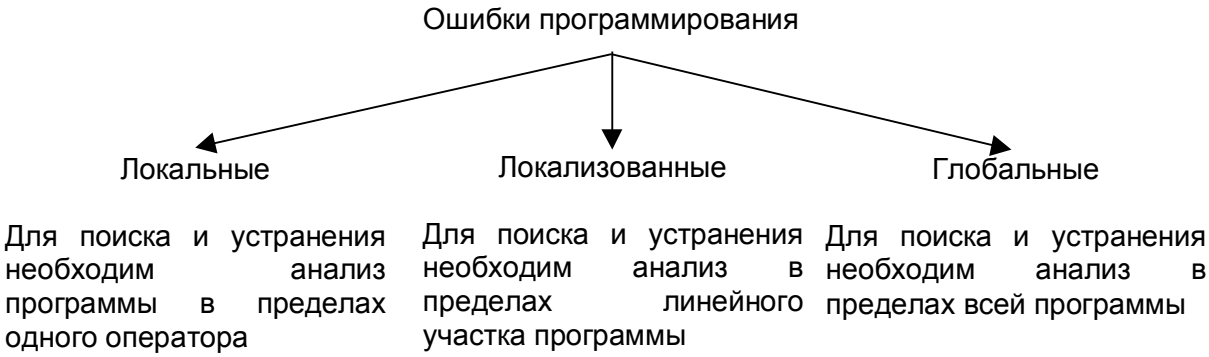


Рис. 4-17. Типы ошибок программирования.

Рассмотрим далее наиболее распространенные семантические ошибки на примерах программирования микроконтроллерных систем семейства МК-51, приводя обобщенные ситуации, наиболее ярко иллюстрирующие возможные ошибки.

Локальные ошибки.

1. Неверный тип или значение константы.

Такая ошибка связана с величинами программы, которые не меняют своего значения при ее выполнении. Ошибочным может быть тип константы или неверно задано ее значение, отличающееся от обеспечивающего правильную работу.

Ошибочно	Комментарий	Правильно
add a, #101	Программист предполагал добавить к аккумулятору значение 5_{10}, записав его в двоичном виде, но не указал постфикс b двоичного операнда, что привело к суммированию с аккумулятором величины 101_{10}, т.к. по умолчанию при записи операндов предполагается употребление десятичной системы счисления.	add a, #101b
mov a,#12h	Программист ошибся в величине константы, которую следовало занести в аккумулятор.	mov a,#11h

2. Неправильный сдвиг регистра.

Ошибка возникает при выполнении операций сдвига содержимого регистра, в частности, с целью выделения определенных разрядов регистра.

Ошибочно	Комментарий	Правильно
rl a	Программист ошибся в задании направления сдвига аккумулятора.	rr a

3. Неверное формирование и употребление логических выражений.

Ошибка появляется в операторах, содержащих логические выражения, и в операторах, непосредственно связанных с этими выражениями.

Ошибочно	Комментарий	Правильно
anl a,r0	Программист ошибся в выборе типа логической операции - наложил маску по "И", в то время как по условиям задачи следовало использовать наложение маски по "ИЛИ".	orl a,r0
xrl a,b	Программист ошибся в выборе операнда, взяв его из регистра-расширителя, в то время как по условиям задачи следовало использовать операнд из порта 1.	xrl a,p1

4. Неверный результат арифметических выражений.

Ошибка представляет собой появление в результате выполнения последовательности арифметических операций результата, отличного от предполагаемого.

Ошибочно	Комментарий	Правильно
add a,#100	Программист корректирует значение аккумулятора. По условию задачи при коррекции следовало учесть перенос в старший разряд, который мог возникнуть перед выполнением данной команды.	addc a,#100

5. Наличие знаменателей, способных обратиться в нуль.

Такая ситуация не является непосредственно ошибкой, но может приводить к неправильному функционированию программы, если результат (а при делении на ноль он не определен) используется без надлежащей проверки.

Ошибочно	Комментарий	Правильно

div ab	Программист не учитывает возможности деления на ноль, рискуя получить неожиданное поведение программы.
	Следует сначала проверить делитель (в регистре-расширителе) на равенство нулю и, если это так, перейти к обработке данной ситуации, а в противном случае осуществить деление.
		mov a,b
		jz err_div
		mov a,op1
		div ab

6. Применение одних синтаксических конструкций вместо других.

Возникает при замене одних конструкций языка другими без нарушения правил синтаксического построения операторов и программы в целом.

Ошибочно	Комментарий	Правильно
s1: equ 30h sl: equ 31h mov p1,s1	Программист передает в первый порт значение переменной s1 вместо sl. Так как оба этих символических имени определены, транслятор не объявит примененную конструкцию недопустимой.	s1: equ 30h sl: equ 31h mov p1,sl

7. Неверная размерность данных.

Ошибка возникает при использовании операндов с размерностью, отличной от предписываемой в данной команде.

Ошибочно	Комментарий	Правильно
push dptr	Программист предполагает сохранить в стеке значение указателя данных (имеющего размерность 16 бит), чтобы использовать dptr для	push dph push dpl

Ошибочно	Комментарий	Правильно
	обращения к новой области данных, а затем восстановить указатель на прежнюю область. Так как в качестве операнда команды push используется 8-битный адрес внутренней памяти данных, то в стек занесется содержимое байта, эквивалентного байту начала области, отведенной под dptr, а именно - содержимого 82h ячейки внутренней памяти данных (т.е. dpl). Старший байт dptr сохранен в стеке не будет, и ошибка проявится в случае изменения старшего байта dptr. Следует сохранять многобайтные конструкции в стеке побайтно.	

Локализованные ошибки.

1. Неверная передача управления оператором перехода.

Возникает в операторах условной и/или безусловной передачи управления и в связанных с ними метках. Ошибка приводит к нарушению порядка выполнения операторов. Причиной ошибки могут быть неверно расставленные метки и/или неверно сформированные условия перехода.

Ошибочно	Комментарий	Правильно
inc lo jnc m1 inc hi m1:	Программист предполагал увеличить на 1 двухбайтную переменную (hi lo), не учтя тот факт, что условие для выполнения инкрементирования байта hi никогда не будет сформировано, т.к. команда inc не влияет на флаг C. Следует анализировать перенос в старший байт сравнением младшего байта с нулем.	inc lo mov a,lo jnz m1 inc hi m1:

2. Отсутствие анализа кодов ответа.

Если при возврате управления из вызванной подпрограммы вызвавшая подпрограмма не анализирует ее код ответа (например, информацию об успешном за-

вершении вычислений в подпрограмме), то при ненормальном завершении вызванной подпрограммы в вызывающей может возникнуть ошибка, так как она будет выполняться так же, как и в случае успешного завершения вызванной подпрограммы.

Ошибочно	Комментарий	Правильно
lcall subr	Программист не анализирует результат работы подпрограммы subr, помещаемый ею в аккумулятор, что может привести к ошибке. Перед продолжением вычислений следует проверить, не равно ли нулю значение аккумулятора, и, если равно, то перейти к обработке ошибки.	lcall subr jz err

3. Неверный адрес в дескрипторе.

Такая ошибка возникает в операторах, формирующих адреса, указатели, ссылки для косвенного обращения к объектам.

Ошибочно	Комментарий	Правильно
mov dptr,#tbl jmp @a+dptr	Программист предполагал перейти к началу таблицы tbl, не учтя тот факт, что данная команда использует смещение, хранимое в аккумуляторе, которое к моменту выполнения команды может быть любым. Перед выполнением команды следует обнулить аккумулятор.	mov dptr,#tbl clr a jmp @a+dptr

4. Ошибки при работе со стековой памятью.

Ошибки возникают при выполнении операторов, использующих стековую память в качестве одного из операндов. К ошибкам данного типа относятся: выход за пределы области стековой памяти (при использовании большего количества обращений на запись в стек, чем это предусмотрено отведенной под него памятью, не занятой программными переменными), а также нарушение структуры стековой памя-

ти (при чтении из стека большего количества информации, чем было в него помещено, при выполнении лишних операторов возврата из подпрограммы и пр.).

Ошибочно	Комментарий	Правильно

mov a, op1
mov b,op2
lcall divider
…
divider:
push a
mov a,b
jz ex
 	
ok: pop a
div ab
 	
ex: ret
	Программист использует подпрограмму divider для вычисления частного. В подпрограмме избегается ситуация деления на ноль, для анализа которой используется стек как средство временного хранения операнда A. Однако в случае нулевого делителя обратное чтение из стека, восстанавливающее его баланс, произведено не будет, в результате чего команда ret будет использовать для возврата неверный адрес: вместо (HI addr, Lo addr) наверху стека будет (a HI addr).
	Для того чтобы исправить ошибку, необходимо усложнить логику переходов к оператору возврата в нормальном случае и в случае нулевого делителя.
		mov a, op1
		mov b,op2
		lcall divider
		…
		divider:
		push a
		mov a,b
		jz ex
		
		ok: pop a
		div ab
		jmp quit
		ex: pop a
		quit:ret

5. Сдвиг на единицу при индексировании.

Сущность ошибки заключается в том, что при использовании индексирующих переменных значение индекса оказывается на единицу больше или меньше граничного значения индекса.

Ошибочно	Комментарий	Правильно

mov r0,#10
mov dptr,#tbl
 	
m1: mov a,r0
 	
movx @dptr,a
inc dptr
djnz r0,m1
	Программист предполагал заполнить область tbl значениями 10, 9, 8, …, 2, 1, 0. Однако переход на метку m1 для занесения последнего нуля выполнен не будет.
	Следует либо помещать в аккумулятор на единицу меньшее значение, либо использовать другой вариант построения цикла (например: cjne r0, #0FFh, m1).
		mov r0,#11
		mov dptr,#tbl
		
		m1: mov a,r0
		dec a
		movx @dptr,a
		inc dptr
		djnz r0,m1

6. Бесконечный повтор последовательности операторов.

Ошибка ведет к прекращению выполнения задачи вследствие невыполнения условий входа из циклически повторяющегося участка.

Ошибочно	Комментарий	Правильно

mov a,#255
mov r0,#0
 	
m1: dec a
inc r0
cjne a,0,m1
	Программист предполагал выполнить численное решение уравнения 255-X=X методом перебора возможных значений X. В аккумуляторе значения функции (255-X) изменяются от 255 до нуля, в регистре r0 значения функции X изменяются от нуля до 255. Цикл будет выполняться бесконечно, т.к. значения в a и r0 никогда не совпадут (точное решение уравнения X=127.5). В данной программе значения будут изменяться так:
	a: 255, 254, …, 129, 128, 127, 126, …, 0, 255..
	r0: 0, 1, …, 126, 127, 128, 129, …, 255, 0..
	Следует не выявлять равенство значений функций, а выявлять ситуацию, когда изначально большая по значениям функция стала меньше, чем изначально меньшая (иными словами, произошел переход через точку пересечения графиков).
		mov a,#255
		mov r0,#0
		
		m1: dec a
		inc r0
		cjne a,0,m2
		m2: jnc m1

7. Ошибки при организации циклов.

Возникают при неправильной работе с переменными цикла, неправильным формированием условий выхода из цикла, ошибочным включением операторов в тело цикла и т.п.

Ошибочно	Комментарий	Правильно
n: equ 30h res: equ 31h mov r1,n	Программист предполагал осуществить накопление в переменной res суммы (по модулю 256) ряда неотрицательных чисел n, n-3, n-6, и т.д.	n: equ 30h res: equ 31h mov r1,n

Ошибочно	Комментарий	Правильно

mov res,#0
 	
m1: mov a,res
add a,r1
mov res,a
mov a,r1
clr c
subb a, #3
mov r1,a
jnz m1
	Однако не было учтено то, что равенство нулю завершающего слагаемого (по факту которого выполняется выход из цикла) выполняется не всегда (это ряд …,8,5,2,-1 и ряд …,7,4,1,-2) - в таких случаях цикл будет повторен (два или три раза), выполняя дополнительное суммирование (ряда 255, 252, …, 0, либо ряда 254, 251,…, 2, 255, 252, …, 0).
	Следует отметить, что в данной программе бесконечный повтор участка команд, описанный в п. 6, не наблюдается.
	В данном случае следует выявлять отрицательный результат, возникающий при подготовке очередного слагаемого, и заканчивать на этом цикл суммирования.
		mov res,#0
		
		m1: mov a,res
		add a,r1
		mov res,a
		mov a,r1
		clr c
		subb a, #3
		mov r1,a
		jnc m1

Глобальные ошибки.

1. Ошибки инициализации.

Ошибка такого типа означает, что перед первым использованием регистров, переменных, областей памяти и других программно доступных элементов микроконтроллера не выполнялось либо было неверно выполнено занесение начальных значений.

Ошибочно	Комментарий	Правильно
cnt: equ 31h ptr: equ 32h lcall init m1: movx a,@dptr jnz m2 inc cnt m2: inc dptr inc ptr mov a,ptr	Программист предполагал выполнить подсчет количества нулевых значений среди первых пяти элементов таблицы tbl, размещаемой во внешней памяти данных. Настройка указателя данных и счетчика количества нулевых элементов производится в подпрограмме init. Однако программистом не был учтен тот факт, что для пятикратного выполнения цикла необходимо, чтобы инкрементирующаяся перемен-	cnt: equ 31h ptr: equ 32h lcall init m1: movx a,@dptr jnz m2 inc cnt m2: inc dptr inc ptr mov a,ptr

Ошибочно	Комментарий	Правильно

cjne a,#5,m1
ret
init: mov cnt,#0
mov dptr,#tbl
ret
	ная ptr при первом выполнении цикла равнялась нулю (в противном случае цикл может выполняться от одного до 255 раз в зависимости от значения переменной ptr).
	В подпрограмме init следует присвоить счетчику проходов нулевое значение.
		cjne a,#5,m1
		ret
		init: mov cnt,#0
		mov dptr,#tbl
		mov ptr,#0
		ret

2. Наложение переменных и участков программ.

Ошибка проявляется в ситуации, при которой обращение к некоторой переменной или метке по ее адресу фактически приводит к обращению в участок памяти, отведенный для размещения другой переменной или метки.

Ошибочно	Комментарий	Правильно
main.asm org 8100h - - - lcall divider - - - include mod1.asm include mod2.asm end mod1.asm org 8200h divider: - - - ret mod2.asm org 8200h multipl: - - -	Программист использует в программе main.asm два модуля - mod1.asm и mod2.asm. Из модуля main вызывается подпрограмма divider, описанная в модуле mod1. Фактически при вызове подпрограммы divider обращение произойдет к подпрограмме multipl, расположенной в модуле mod2. Причина заключается в указанных адресах расположения модулей в памяти, задаваемых директивой org ассемблера, а также в порядке трансляции модулей, определяемом порядком подключения модулей директивами include. Так как и у модуля mod1.asm, и у модуля mod2.asm одинаковые адреса расположения - 8200h, то фактически с адреса 8200h будет расположен оттранслированный код модуля, подключаемого последним, то есть код модуля mod2. Таким образом, при любых вызовах по общим адресам обращение будет производиться к участку кода последнего модуля. При этом	main.asm org 8100h - - - lcall divider - - - include mod1.asm include mod2.asm end mod1.asm divider: - - - ret mod2.asm multipl: - - -

Ошибочно	Комментарий	Правильно
ret	возможно выполнение кодов, соответствующих данным и прочие ошибки. Следует избегать задания абсолютных адресов в подключаемых модулях.	ret

3. Неправильное использование переменных многими модулями.

Рассматриваемая ошибка связана с взаимодействием подпрограмм и обнаруживается, как правило, на стадиях комплексной отладки. Примером ошибки может являться ситуация, при которой одна подпрограмма искажает значения переменных, используемых другой подпрограммой.

Ошибочно	Комментарий	Правильно

num: equ 30h
ar1: equ 40h
ar2: equ 50h
res1: equ 61h
res2: equ 62h
 	
- - -
sub1: mov r1,#ar1
 	
clr a
cc1: add a,@r1
inc r1
djnz num,cc1
mov res1,a
ret
- - -
sub2: mov r1,#ar2
 	
clr a
cc2: add a,@r1
	Программистом реализованы две подпрограммы sub1 и sub2, выполняющие суммирование num элементов из областей данных ar1 и ar2 соответственно с сохранением результатов вычислений в переменных res1 и res2 соответственно.
	Программой main осуществляется определение числа num подлежащих суммированию элементов, а затем производится вызов описанных подпрограмм.
	Ошибка заключается в том, что подпрограмма sub1 завершит свое выполнение, обнулив переменную-счетчик num, в результате чего подпрограмма sub2 произведет суммирование 256 элементов внутренней памяти данных, расположенных с начала области ar2 (байта 50h) и выше (Так как внутренняя память данных микроконтроллера имеет объем 256 байт, то адреса, большие 255, будут вычислены по модулю 256).
	Следует оставить переменную num, указывающую количество обрабатываемых данных,
		num: equ 30h
		ar1: equ 40h
		ar2: equ 50h
		res1: equ 61h
		res2: equ 62h
		cnt: equ 63h
		- - -
		sub1: mov r1,#ar1
		mov cnt,num
		clr a
		cc1: add a,@r1
		inc r1
		djnz cnt,cc1
		mov res1,a
		ret
		- - -
		sub2: mov r1,#ar2
		mov cnt,num
		clr a
		cc2: add a,@r1

Ошибочно	Комментарий	Правильно
inc r1 djnz num,cc2 mov res2,a ret - - - main: <num> lcall sub1 lcall sub2 - - -	без изменений по ходу выполнения программы, а цикл суммирования организовать на другом счетчике - cnt, в который каждый раз перед выполнением цикла заносится информация о количестве его повторений.	inc r1 djnz cnt,cc2 mov res2,a ret - - - main: <num> lcall sub1 lcall sub2 - - -

4. Некорректное внесение изменений в программу.

Ошибка появляется при попытке внесения изменений в отлаживаемую программу без учета возможных влияний этих изменений на оставшуюся нетронутой часть программы.

Первоначально	Ошибочно	Правильно	Комментарий
x1: equ 30h x2: equ 31h x3: equ 32h x4: equ 33h - - - sum: mov r0,#30h mov r1,#4 clr a	y1: equ 30h y2: equ 31h y3: equ 32h y4: equ 33h x1: equ 40h x2: equ 41h x3: equ 42h x4: equ 43h - - - sum: mov r0,#30h mov r1,#4 clr a	y1: equ 30h y2: equ 31h y3: equ 32h y4: equ 33h x1: equ 40h x2: equ 41h x3: equ 42h x4: equ 43h - - - sum: mov r0,#x1 mov r1,#4 clr a	Программист в подпрограмме sum предполагал осуществить суммирование четырех переменных x1 - x4, расположенных в смежных ячейках памяти, начиная с адреса 30h. Развитие программы потребовало размещения переменных y1 - y4 с адреса, первоначально отведенного для переменных x1 - x4, так что сами они стали располагаться, начиная с адреса 40h. Этот факт не был учтен в подпрограмме

Первоначально	Ошибочно	Правильно	Комментарий
cyc: add a,@r0 inc r0 djnz r1,cyc ret	cyc: add a,@r0 inc r0 djnz r1,cyc ret	cyc: add a,@r0 inc r0 djnz r1,cyc ret	учтен в подпрограмме sum, в результате чего она стала осуществлять суммирование переменных y1 - y4. Следует использовать скорректированные значения адреса. В данном случае - через символическую адресацию переменных.

5. Функциональное изменение последовательности выполняемых команд.

Программа может выполнять действия, не предусмотренные алгоритмом, или не выполнять предусмотренных действий, либо выполнять только предусмотренные, но в неверной последовательности.

Ошибочно	Комментарий	Правильно
lcall subr1 lcall subr2 lcall subr3 lcall subr5	Программист предполагает выполнять 1-ю, 2-ю, 3-ю и 5-ю подпрограммы. Однако правильным для реализации задачи является последовательность вызовов 3-й, 4-й, 1-й и 2-й подпрограмм (был нарушен порядок вызовов и, кроме того, вызывалась лишняя - 5-я - и не вызывалась нужная - 4-я - подпрограмма).	lcall subr3 lcall subr4 lcall subr1 lcall subr2

6. Ошибки при обработке прерываний.

Ошибки данного класса могут состоять как в задании неправильных действий по обработке прерываний, так и в неверном определении места обработки прерываний (в частности, того множества операторов, на котором разрешены прерывания).

Ошибочно	Комментарий	Правильно
org 8003h - - - ret	Программист завершает обработчик прерывания от внешнего источника 0 оператором ret. Данный оператор не восстанавливает внутренний флаг окончания обработки прерывания в системе управления прерываниями, в связи с чем обработка прерываний данного типа и прерываний с уровнем приоритета, равным и меньшим, чем у данного, будет автоматически заблокирована до момента сброса микроконтроллера. Завершать обработку прерываний следует командой reti.	org 8003h - - - reti

- - -
add a,lo
mov lo,a
mov a,hi
addc a,#0
mov hi,a
- - -
int_sub:
push a
push b
- - -
mov a,in1
mov b,in2
mul ab
mov res,a
pop b
pop a
reti
	Программист в фоновой программе выполняет сложение с участием двухбайтной переменной (hi lo) с использованием для коррекции старшего байта команды addc. Обработчик прерывания int_sub в числе прочих действий выполняет умножение переменных in1 и in2.
	В случае переключения на обработчик прерывания на участке между командами фоновой программы add a,lo и addc a,#0 и наличии переноса из младшего байта в старший, результат сложения будет неверным, так как команда умножения обнуляет бит C слова состояния программы, в результате чего команда addc не сможет учесть предшествующее значение бита C.
	Следует сохранять (как правило, в стековой памяти) содержимое слова состояния программы фонового процесса при входе в обработчик прерывания, способного своими вычислениями влиять на значение слова состояния программы, а при выходе из обработчика - восстанавливать значение слова состояния, существовавшее на момент переключения на обработчик прерывания.
		- - -
		add a,lo
		mov lo,a
		mov a,hi
		addc a,#0
		mov hi,a
		- - -
		int_sub:
		push a
		push b
		push psw
		- - -
		mov a,in1
		mov b,in2
		mul ab
		mov res,a
		pop psw
		pop b
		pop a
		reti

Ошибочно	Комментарий	Правильно

- - -
 	
mov a,flags
clr acc.3
mov flags,a
 	
- - -
int_sub:
- - -
mov a,flags
clr acc.4
mov flags,a
reti
	Программист использует переменную flags, в которой расположены биты-признаки ряда программных ситуаций.
	Ошибка в программе заключается в том, что, если прерывание, обрабатываемое подпрограммой int_sub возникнет в момент после выполнения команды clr acc.3, но перед выполнением команды mov flags, a, и при этом в байте flags третий бит был установлен в "1", то его обнуление в байте flags выполнено не будет, так как выполняющийся обработчик прочтет в аккумулятор необновленное значение байта flags.
	Сохранение аккумулятора в стеке (как это сделано в предыдущем случае) в данной ситуации не позволит подпрограмме int_sub обнулить установленный четвертый бит (так как в восстановленном из стека аккумуляторе будет старое значение четвертого бита).
	Следует в подобных ситуациях запрещать на время прохождения критического участка прерывания, конфликтующие по ресурсам с фоновым заданием.
		- - -
		clr ea
		mov a,flags
		clr acc.3
		mov flags,a
		setb ea
		- - -
		int_sub:
		- - -
		mov a,flags
		clr acc.4
		mov flags,a
		reti

7. Непредусмотренные исключительные ситуации.

Ошибки данного вида возникают, когда характеристики обрабатываемой программой входной информации выходят за допустимые алгоритмом пределы, и обработка подобных ситуаций в программе не была предусмотрена.

Ошибочно	Комментарий	Правильно

receive:
jnb ri,receive
mov a,sbuf
clr ri
ret
 	
- - -
mov r1,#3
m1: lcall receive
add a,res
mov res,a
djnz r1,m1
	Программист при возникновении некоторых событий (например, приход подтверждающего сигнала) предполагает осуществить прием трех байт информации через последовательный порт микроконтроллера, реализуя обмен путем опроса флага готовности приемника.
	При аварии источника данных, произошедшей перед окончанием приема трех байт, программа будет находиться в состоянии ожидания готовности приемопередатчика, не производя действий по решению основной задачи.
	Следует предусмотреть возможность принудительного выхода из цикла приема в случае превышения некоторого времени ожидания.
	В данном примере с помощью таймера 0 реализуется подсчет временных меток с инкрементированием счетчика t_0. При каждом входе в подпрограмму receive значение счетчика обнуляется, и счет времени начинается сначала. В процессе ожидания данных, поступающих в приемник микроконтроллера от источника, подпрограмма receive анализирует истечение интервала тайм-аута путем сравнения накопленного в переменной t_0 значения с максимально возможным интервалом ожидания time. При превышении допустимого времени происходит принудительный выход из подпрограммы с фиксацией ошибки в переменной err для реализации дальнейших действий по обработке аварийной ситуации.
	Примечание 1. Таймер 0 и система прерываний микроконтроллера должны быть соответствующим образом настроены.
	Примечание 2. В ряде моделей микроконтроллеров подобные функции выполняет специальное устройство - сторожевой таймер.
		org 800bh
		inc t_0
		reti
		- - -
		receive:
		mov t_0,#0
		mov err,#0
		wt: mov a,t_0
		cjne a,#time,l1
		l1: jc ok
		mov err,#1
		jmp ex
		ok: jnb ri,wt
		mov a,sbuf
		clr ri
		ex: ret
		- - -
		mov r1,#3
		m1: lcall receive
		add a,res
		mov res,a
		mov a,err
		jnz to_error
		djnz r1,m1

8. Ошибки времени.

Выделяются две группы таких ошибок. К первой группе относятся ошибки, обусловленные неправильным квантованием времени, проявляющиеся в том, что для выполнения некоторого участка программы отводится недостаточно длительный отрезок. Ко второй группе относятся ошибки, обусловленные неправильным расчетом времени выполнения отдельных процессов (связанные, например, с неучетом частоты вызовов процессов и времени их выполнения).

Ошибочно	Комментарий	Правильно

chan: equ 30h
res: equ 31h
- - -
mov 0DAh, #0
mov a,0D8h
anl a,#0F8h
orl a,chan
mov 0D8h,a
 	
mov res,0D9h
- - -
	Программист осуществляет чтение значения сигнала со встроенного аналого-цифрового преобразователя МК SAB80C515 в переменную res, предварительно настраивая диапазон входного сигнала и осуществляя переключение аналогового канала.
	Ошибка заключается в том, что после изменения параметров работы АЦП и перед чтением новых значений необходимо выдержать паузу, во время которой АЦП аппаратно осуществляет перепрограммирование встроенного ЦАП, переключение каналов и собственно аналого-цифровое преобразование. В противном случае будут считаны неверные показания.
	При полной перенастройке АЦП время преобразования достигает 29 мкс (в случае использования микроконтроллера с тактовой частотой в 12МГц).
	Следует выдержать паузу указанной длительности либо осуществлять работу по флагу готовности.
		chan: equ 30h
		res: equ 31h
		- - -
		mov 0DAh, #0
		mov a,0D8h
		anl a,#0F8h
		orl a,chan
		mov 0D8h,a
		lcall pause
		mov res,0D9h
		- - -
		pause:
		mov r0,#10
		wait: nop
		djnz r0,wait
		ret

mov a,#1
mov r0,#8
m1: mov p1,a
rl a
 	
djnz r0,m1
	Программист предполагал осуществить выдачу на разряды первого порта последовательности "бегущая единица", предназначенной для внешнего исполнительного устройства, работающего на частоте около 1 кГц.
	Ошибка заключается в неучете скорости выполнения операций микроконтроллером. Ко-
		mov a,#1
		mov r0,#8
		m1: mov p1,a
		rl a
		lcall pause
		djnz r0,m1

Ошибочно	Комментарий	Правильно
- - -	полнения операций микроконтроллером. Команды mov и rl выполняются за 1 машинный цикл, команда djnz - за 2 цикла. При тактовой частоте микроконтроллера в 12МГц длительность удержания разряда порта составит 4 мкс, что недопустимо для внешнего устройства. Следует ввести программную задержку между переключениями разрядов порта в пределах 1 - 1,2 мс.	- - - pause: mov cnt1,#5 pa1:mov cnt2,#100 pa2:djnz cnt2,pa2 djnz cnt1,pa1 ret

4.3. Средства и методы комплексной отладки микроконтроллерных систем.

К средствам отладки, позволяющим проводить совместную отладку аппаратуры и программного обеспечения, относятся логические анализаторы, внутрисхемные эмуляторы и интегрированные среды отладки. Все эти средства являются сложными программно-аппаратными комплексами, решающими достаточно широкий круг задач отладки.

4.3.1. Логические анализаторы.

Во многих случаях для анализа работы вычислительной системы необходимо зафиксировать изменяющийся с высокой частотой информационный поток нескольких линий на протяжении нескольких циклов работы вычислительной системы. Для этих целей применяются логические анализаторы. Типовая структура данного средства отладки показана на рис. 4-18, а его внешний вид - на рис. 4-19.

При использовании логического анализатора его входы подключают к контролируемым точкам вычислительной системы и проводят его программирование. При программировании логического анализатора задаются:

- метод формирования тактовых импульсов регистрации событий во внутренней памяти анализатора (от внутреннего генератора, от внешней цепи, коэффициент деления частоты тактовых импульсов);

- признаки событий, формирование которых активизирует процесс фикса-

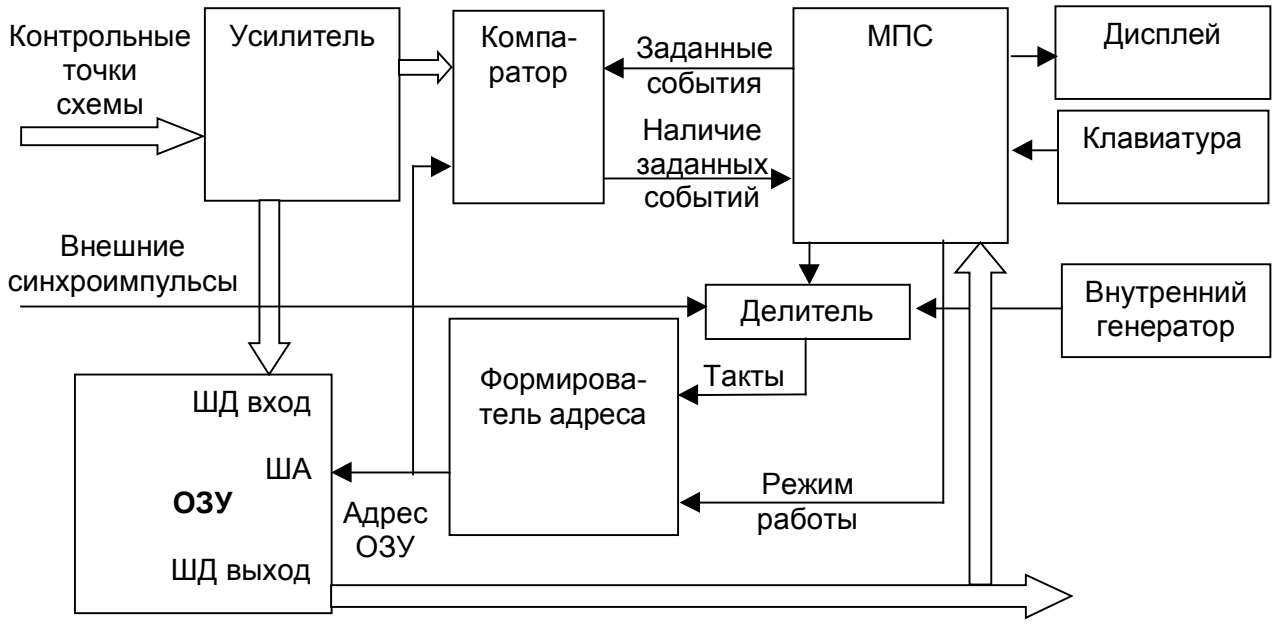


Рис. 4-18. Устройство логического анализатора.

ции состояний отлаживаемой вычислительной системы в памяти анализатора (значения или диапазон значений контролируемых шин, тип события - запускающее процесс регистрации или останавливающее его);

- количество предзапусков логического анализатора (указывает, какое количество регистрационных отсчетов анализатора, предшествующих появлению запускающих событий, следует сохранить в памяти);

- количество постзапусков логического анализатора (указывает, какое количество регистрационных отсчетов анализатора, следующих за появлением останавливающих событий, следует сохранить в памяти).

Рис. 4-19. Внешний вид логического анализатора.

Накопленные измерения могут быть представлены различным способом:

- в виде графа переходов в пространстве состояний;
- в виде временных диаграмм;
- в виде цифровых кодов;
- в виде дизассемблированных мнемоник команд шины данных.

Логические анализаторы позволяют подсчитывать количество событий заданного типа, произошедших за интервал измерения, определять временные параметры сигналов, фиксировать и отображать помехи (импульсы с длительностью менее минимально допустимой, заданной пользователем) в контролируемых цепях.

Используемые в настоящее время логические анализаторы позволяют контролировать одновременно до 150 точек схемы, обладают емкостью памяти до 32 КБайт на канал и позволяют фиксировать состояние контрольных точек с периодом от 0,5 с до 5 нс. Популярны логические анализаторы фирмы Hewlett-Packard.

Рассмотрим методику использования логического анализатора при отладке на реальной аппаратуре программы, схема работы которой показана на рис. 4-20.

Допустим, исследователя интересует решение задачи в момент передачи управления от команды A5 к команде A6 после двух повторений внешнего цикла A1 - A9 и восьми повторений внутреннего цикла A2 - A7. Для запуска в этой области программы необходимо сформировать следующее условие запуска:

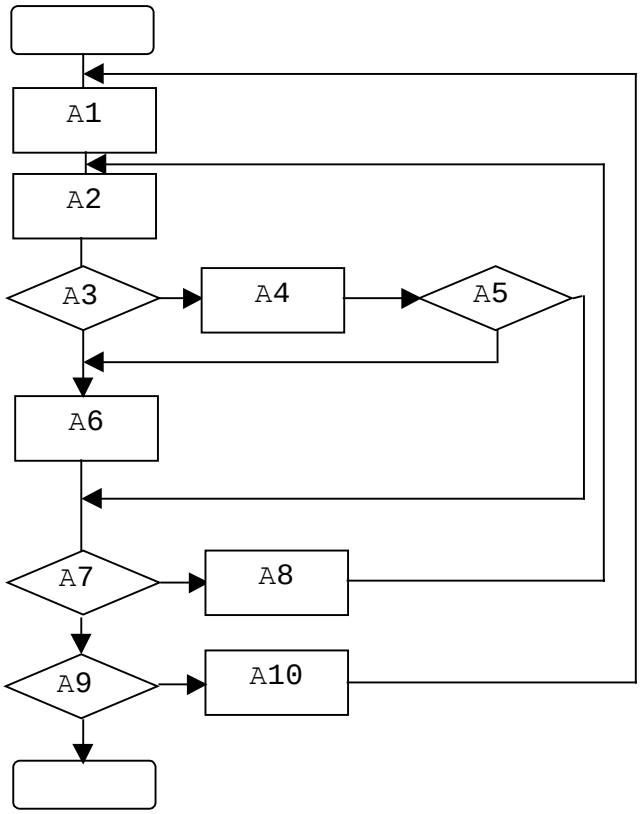


Рис. 4-20. Пример отлаживаемой программы.

- обнаружить A1 2 раза;
- обнаружить A2 8 раз;
- обнаружить последовательность адресов A5 и A6, идущих подряд.

4.3.2. Внутрисхемные эмуляторы.

Одним из главных обстоятельств, затрудняющих отладку микроконтроллерных систем, является недоступность внутренних элементов микросхемы микроконтроллера для прямого контроля и воздействий во время работы программы. Контроль и воздействия можно осуществлять лишь косвенно, путем передачи информации от внутренних элементов схемы во внешнюю среду (при контроле) и извне во внутренние элементы (при управлении) с помощью специальных подпрограмм по факту появления некоторых событий.

Вследствие сложности микроконтроллера, а также в связи с необходимостью такой реализации подпрограмм, при которой не нарушается ход основного вычислительного процесса, для реализации указанных действий требуются значительные временные и программно-аппаратные ресурсы системы.

Для преодоления указанных трудностей применяются внутрисхемные эмуляторы.

При использовании внутрисхемного эмулятора микроконтроллер извлекается из панели своего разъема и на его место с помощью кабеля подсоединяется аппаратура внутрисхемного эмулятора. Внутрисхемный эмулятор полностью заменяет микроконтроллер отлаживаемой системы, сохраняя при этом доступ пользователя ко всей информации о внутреннем состоянии контроллера и его памяти посредством инструментальной ПЭВМ (см. рис. 4-21).

Рис. 4-21. Устройство внутрисхемного эмулятора.

Внешний вид платы МК-системы с извлеченным МК и подсоединенным через его разъем внутрисхемным эмулятором показан на рис 4-22. Существуют варианты исполнений эмулятора в виде платы расширения инструментальной ЭВМ.

Рис. 4-22. Подсоединение внутрисхемного эмулятора к микроконтроллерной системе.

Внутрисхемный эмулятор выполняет следующие функции:

- управление ходом вычислительного процесса (выполнение инициализации программно доступных элементов микроконтроллера, исполнение программы по шагам, по условиям и т.п.);

- сбор информации о ходе вычислительного процесса и ее передача в инструментальную ЭВМ для последующего анализа.

Внутрисхемный эмулятор включает в себя следующие блоки:

- функциональный аналог замещаемого МК с дополнительными портами для служебных обращений к внутренним ресурсам МК;

- устройства, повторяющие отдельные внутренние узлы МК, что делает их доступными для управления и контроля со стороны инструментальной ЭВМ;

- схемы распознавания событий в системе;

- память логических последовательностей состояний в системе;

- средства связи с инструментальной ЭВМ.

Внутрисхемный эмулятор может работать в следующих режимах:

- опроса;

- пошагового исполнения;

- эмуляции исполнения программы в реальном времени.

В режиме опроса отлаживаемая программа остановлена, и оператору системы предоставляется возможность, применяя инструментальную ЭВМ, считать со-

стояние программно доступных элементов микроконтроллера и занести в них новую информацию.

В режиме пошагового выполнения исполняется одна команда отлаживаемой программы, после чего осуществляется переход в режим опроса, при этом происходит передача в инструментальную ЭВМ информации о состоянии вычислительного процесса. Режим пошагового исполнения позволяет проводить отладку исполняемой программы с учетом ее текущего состояния и отлаживать аппаратуру разрабатываемой системы. Недостатком режима является то, что отлаживаемый комплекс не функционирует в реальном времени, так как обмен внутрисхемного эмулятора с инструментальной ЭВМ требует дополнительных временных затрат, в связи с чем ошибки, связанные с нарушением временных соотношений между электрическими сигналами (входными, выходными, входными и выходными) могут остаться невыявленными.

Для исследования разрабатываемого комплекса в реальном времени применяется третий режим работы внутрисхемного эмулятора, в котором он выполняет программу, не выполняя обмена данными с инструментальной ЭВМ. Выход из режима производится при достижении заданных событий, аналогичных управляющим событиям логического анализатора (выполнение заданного числа шагов, появление заданной комбинации на шинах системы, появление заданной последовательности таких комбинаций). Так как процесс обнаружения заданных событий выполняется аппаратными средствами внутрисхемного эмулятора, скорость этого процесса достаточна для сохранения режима реального времени. Однако сервисные возможности режима невелики.

Интерфейс программного обеспечения инструментальной ЭВМ, предназначенного для работы с внутрисхемным эмулятором, близок к интерфейсу программных моделей микроконтроллерных систем (см. рис. 4-9).

Внутрисхемные эмуляторы имеют два типа погрешностей: электрофизического замещения (связана с наличием буферов между имитирующим контроллером и магистралью отлаживаемой системы) и управления (связана с необходимостью при возврате в режим опроса выполнения служебных процедур, во время исполнения которых возможны появления внешних прерываний, изменяющих ход выполнения служебных процедур и т.п.)

Внутрисхемные эмуляторы выпускаются рядом фирм, в частности, Hitex, Nohau и др.

4.3.3. Интегрированные системы разработки.

Интегрированные системы разработки - программно-аппаратные системы, предоставляющие возможность проведения программного проекта по всем стадиям разработки - от создания исходного текста программы до отладки программно-аппаратного комплекса.

В состав программного обеспечения интегрированных сред (см. рис. 4-23) входят:

- редактор исходных текстов;
- транслятор;
- программная модель;
- размещающая программа;
- средства отладки программного обеспечения в реальных условиях эксплуатации;
- справочная подсистема.

С применением интегрированных сред возможно проведение интерактивной отладки программы, функционирующей на микроконтроллерной системе. При этом на инструментальной ЭВМ формируется управляющее воздействие, выдаваемое в микроконтроллер по ходу выполнения пользовательской программы. Использование внешних воздействий позволяет исследовать поведение программно-аппаратного комплекса без внесения каких-либо изменений в текст отлаживаемой программы (задание новых значений при инициализации, запуск управляющих подпрограмм в иной последовательности и т.п.).

В состав аппаратного обеспечения интегрированных систем разработки входят так называемые отладочные платы, представляющие собой аппаратные платформы типовой конфигурации (см. раздел 3), в ряде случаев дополненные набором периферийных элементов (клавиатура, индикаторы и т.п.) и допускающие возможность установки пользовательских подсистем (устройств сопряжения и т.п.).

Интегрированная система разработки позволяет провести полный цикл проектирования системы, в результате которого формируется целевая микроконтроллерная система, способная функционировать в автономном режиме.

Интегрированные системы разработки некоторых фирм (в частности, система Visual Micro Lab) позволяют проводить моделирование типовых элементов систем управления (логических элементов, индикаторов, пассивных компонентов), под-

ключенных к микроконтроллеру, тем самым повышая надежность разрабатываемого программного обеспечения.

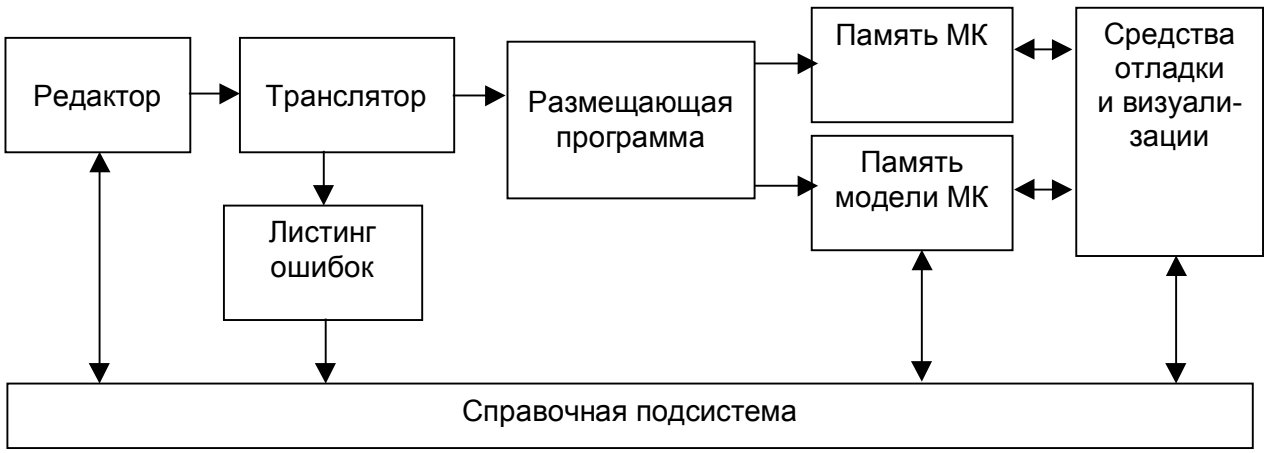


Рис. 4-23. Структура интегрированных сред отладки.

В некоторых интегрированных системах (например, DavE - Digital Application Engineer) справочная подсистема обладает возможностью по командам пользователя, специфицирующим типовые режимы работы блоков микроконтроллера, автоматически генерировать фрагменты текста разрабатываемого программного обеспечения.

Следует отметить, что дополнительные сервисные режимы, несмотря на несомненное удобство и привлекательность, носят вспомогательный характер и предназначены скорее для начинающих разработчиков. Профессиональная разработка МК-систем предполагает наличие глубоких знаний и практических навыков, получить которые можно лишь при кропотливом самостоятельном освоении предметной области.

Интегрированные среды разработки выпускаются фирмами Keil, Tasking Software и рядом других.

Примечание. В Приложении 4 приведено описание применяемой в лабораторном практикуме интегрированной среды отладки Shell51.

РАЗДЕЛ 5. ДИАГНОСТИРОВАНИЕ МИКРОКОНТРОЛЛЕРНЫХ СИСТЕМ.

5.1. Основные понятия и определения.

Под диагностированием понимается процесс поиска и обнаружения в разработанной и отлаженной системе ошибок, вызванных неисправностями в аппаратном обеспечении.

Как уже отмечалось ранее, ненадежность программного обеспечения микроконтроллерных систем объясняется наличием в нем ошибок, внесенных на стадии проектирования. При проведении отладки надежность программного обеспечения повышается.

Аппаратура микроконтроллерных систем имеет иную зависимость частоты отказов от времени (см. рис. 5-1).

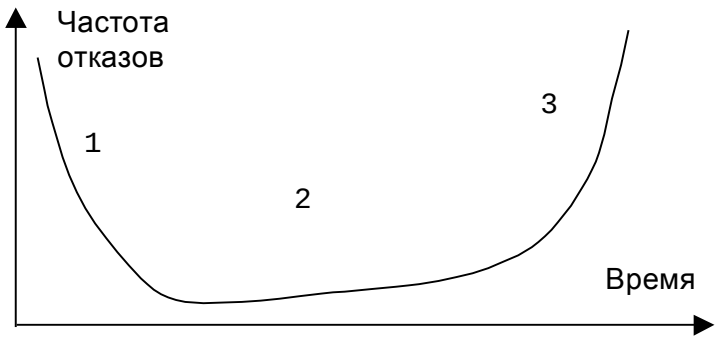


Рис. 5-1. Зависимость надежности аппаратуры от времени.

Участок 1 графика характеризуется достаточно высокой частотой отказов, вызванных дефектами производства (некачественное выполнение герметизации корпусов интегральных схем, ненадежность внешних соединений и т.п.). Период приработки длительностью около 100 часов, как правило, вызывает отказы в критичных элементах, которые успешно прошли стадию выбраковки на производстве.

Участок 2 характеризуется относительно малой частотой отказов, вызываемых естественным износом элементов. Протяженность участка определяется средним сроком службы изделия.

Участок 3 графика характеризуется ростом частоты отказов. Электрические нагрузки на электронные элементы в течение срока службы увеличивают степень износа, а вместе с этим и вероятность отказа изделия. Электрические перегрузки ускоряют темпы износа элемента. Подверженность элемента вибрациям в течение срока службы увеличивает вероятность появления отказов в соединениях.

На этапе эксплуатации микроконтроллерных систем для обеспечения возможности реализации ими целевой функции необходимо выявлять факты ошибок в их работе, определять место возникновения ошибки и ее причину и производить ликвидацию ошибки и, возможно, ее последствий. Часто повторяющиеся ошибки, вызванные недостаточно качественным проектированием системы целесообразно устранять путем перепроектирования системы или ее отдельных узлов.

Различают функциональное и тестовое диагностирование систем.

Функциональное диагностирование осуществляется в процессе использования системы по ее целевому назначению, при этом на входы системы подаются лишь рабочие воздействия.

Тестовое диагностирование проводится путем перевода системы в особый режим, при этом на входы системы подаются специальные тестовые воздействия.

На рис. 5-2 и 5-3 показаны схемы подготовки и проведения функционального и тестового диагностирования.

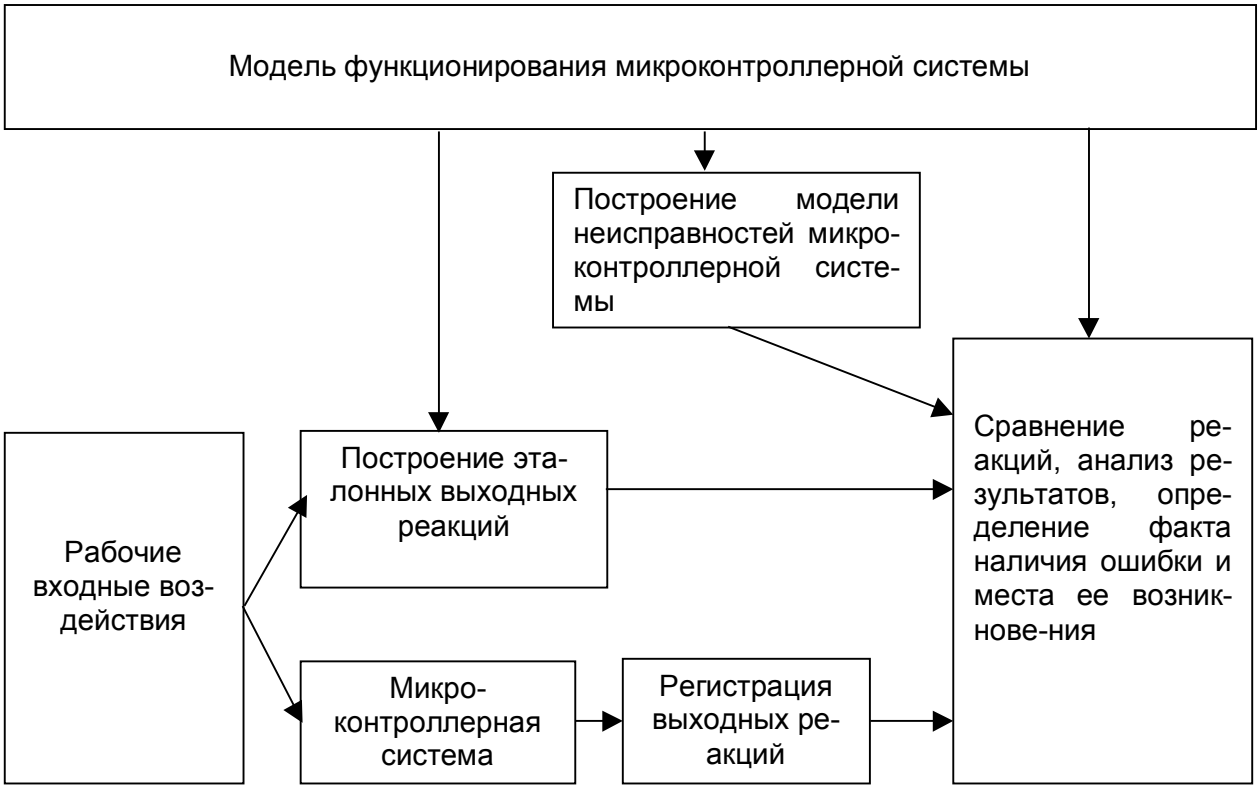


Рис. 5-2. Схема подготовки и проведения функционального диагностирования.

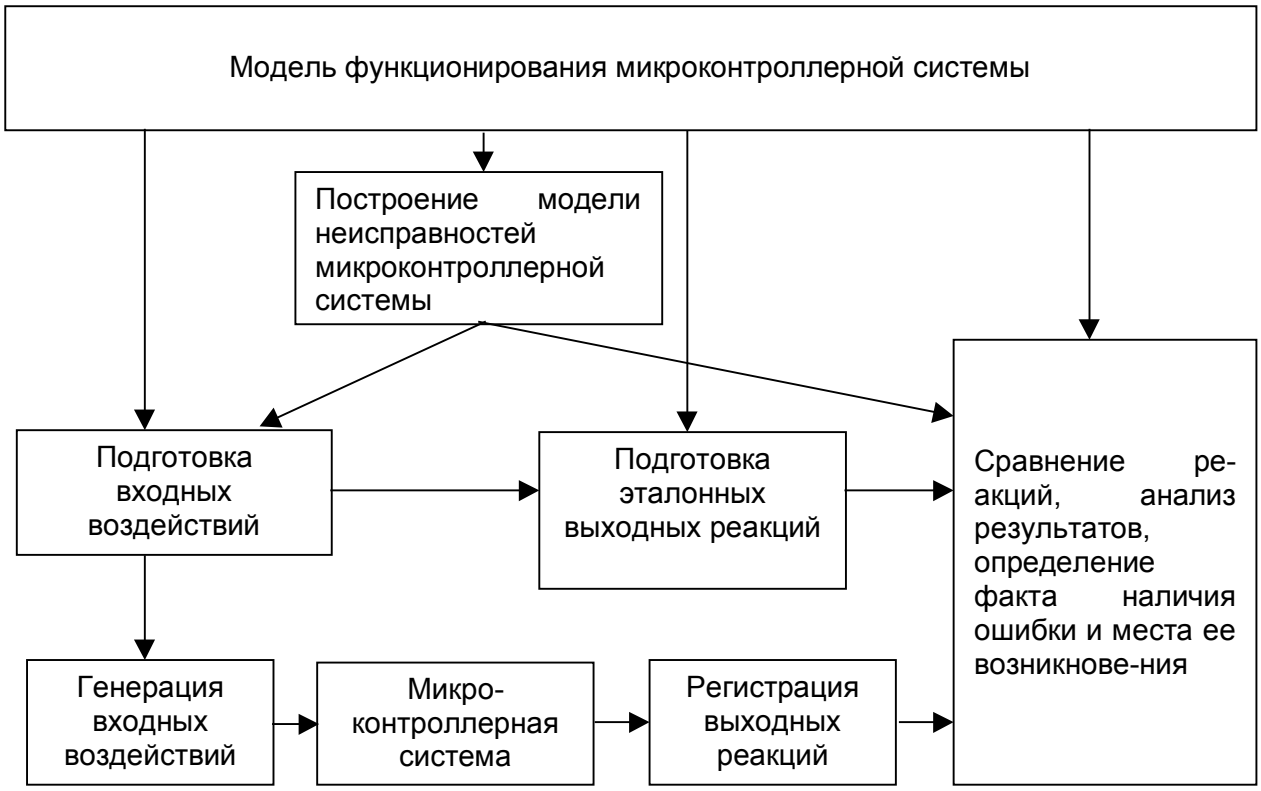


Рис. 5-3. Схема подготовки и проведения тестового диагностирования.

Модель функционирования микроконтроллерной системы, сформированная на этапе проектирования на основе технического задания, математической модели решаемой задачи, алгоритма решения задачи, принципиальных схем аппаратного обеспечения, исходных текстов программ и собственных представлений разработчика о системе, служит основой для подготовки и проведения диагностирования.

На базе модели функционирования строится модель возможных неисправностей микроконтроллерной системы, отражающая причины неверного функционирования системы и способы их проявления.

Для поступающих на систему входных воздействий (рабочих в случае функционального диагностирования и синтетических, построенных на базе указанных моделей в случае тестового диагностирования) строятся эталонные реакции системы, сравниваемые с регистрируемыми выходными реакциями.

По результатам сравнения и последующего анализа, проводимого на базе модели функционирования микроконтроллерной системы и модели неисправностей определяется факт отсутствия или наличия ошибок в работе системы и место их возникновения.

5.2. Средства диагностирования микроконтроллерных систем.

Для проведения диагностирования применяются программные, аппаратные и программно-аппаратные средства (см. рис. 5-4).

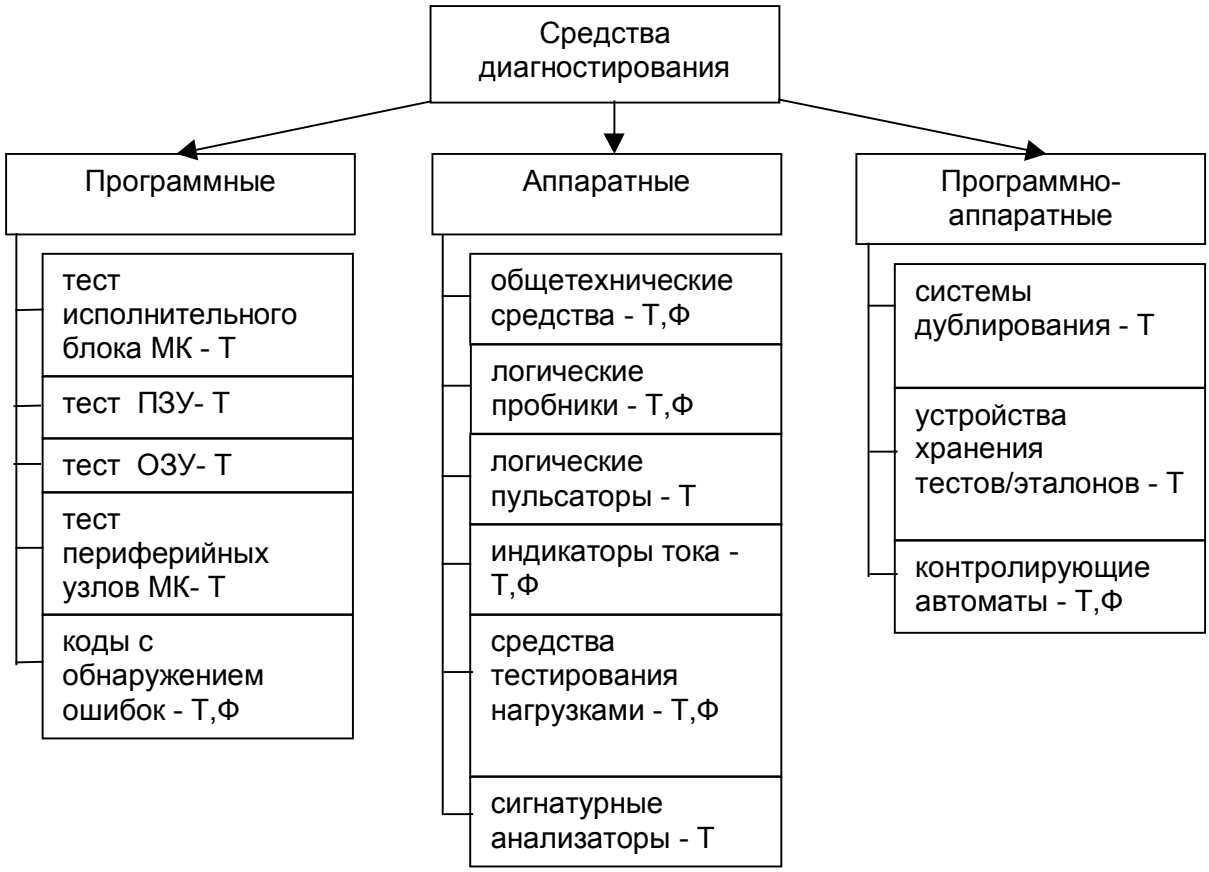


Рис. 5-4. Средства диагностирования микроконтроллерных систем.

Средства, применяемые для проведения функционального диагностирования, помечены на рисунке индексом "Ф", для тестового - индексом "Т".

Ниже рассмотрим указанные средства более подробно.

5.2.1. Программные средства диагностирования.

Программные средства диагностирования, являющиеся частью системного ПО (см. раздел 3.6), представляют собой резидентные программы. При проведении диагностирования программными средствами следует иметь в виду, что достоверность диагностирования зависит от исправности некоторой части микроконтроллерной системы, достаточной для проведения диагностирования.

Тест исполнительного блока микроконтроллера предназначен для выявления неисправностей механизмов управления и обработки данных блока выполне-

ния команд. Тест реализуется путем исполнения команд с заранее определенным результатом их выполнения, и последующего сравнения вычисленных результатов с эталонными. В качестве результатов в зависимости от типа проверяемой команды могут выступать содержимое ячейки памяти, флаги слова состояния программы, адрес перехода и т. п. Такие тесты могут состоять из детерминированной последовательности команд либо представлять собой псевдослучайные генерируемые последовательности. Одной из проблем при организации тестирования данного блока является проблема выявления чувствительности к определенным последовательностям команд.

Тесты памяти (ПЗУ и ОЗУ) предназначены для выявления факта искажения хранимой в памяти информации по причине неисправностей, связанных с дефектами физической структуры элементов памяти (пробои на шины питания, взаимное влияние ячеек и др.).

Тест ПЗУ реализуется с помощью определения контрольной суммы его содержимого, и ее последующего сравнения с эталонной контрольной суммой, также хранящейся в ПЗУ, полученной перед занесением финальной версии программного обеспечения в память программ. В простейшем случае контрольная сумма может представлять собой сумму по модулю 256 всех байтов памяти программ, расположенных в области памяти, подлежащей контролю.

Тест оперативной памяти (ОЗУ) реализуется с помощью записи в каждую из подлежащих контролю ячеек тестового слова, последующего чтения содержимого каждой ячейки и сравнения ее фактического содержимого и значения занесенного в нее тестового слова. В качестве тестирующих комбинаций могут выступать: произвольные данные, шахматный код (чередование в записываемом слове нулей и единиц, например, для ячеек емкостью один байт это коды 55_{16} и AA_{16}), коды "бегущий нуль" и "бегущая единица", при проведении которых для ячейки шириной m бит производится m циклов "запись-чтение" (например, для ячейки емкостью один байт тест "бегущий нуль" представляет собой восемь кодов: 11111110, 11111101, 11111011, …, 10111111, 01111111). Для тестирования ОЗУ применяются и другие, более сложно сформированные последовательности.

Тесты периферийных узлов предназначены для выявления неисправностей в специализированных модулях микроконтроллера: портах, аналого-цифровых преобразователях и др. Такие тесты, как правило, требуют использования дополнительных аппаратных средств для фиксации генерируемых тестом выходных сигна-

лов микроконтроллера либо для подачи на него эталонных воздействий. При тестировании портов ввода-вывода используются коммутаторы, позволяющие подать на разряды портов, использующиеся для ввода информации, сигналы с разрядов портов, использующихся для вывода информации. При тестировании аналого-цифровых преобразователей применяются источники опорных напряжений.

Коды с обнаружением ошибок предназначены для повышения надежности функционирования как микроконтроллерной системы в целом, так и ее отдельных частей, например, последовательных каналов связи и оперативной памяти. Основная идея использования таких кодов состоит во введении информационной избыточности, заключающейся в применении дополнительных служебных разрядов, содержание которых зависит от информационных. При записи кода с обнаружением ошибок служебные разряды формируются по определенным правилам как функция от информационных разрядов. При чтении кода происходит повторное формирование служебных разрядов и их сравнение со сформированными ранее. При использовании достаточной информационной избыточности возможно не только обнаружение факта ошибки, но и определение ее местоположения и исправления.

В качестве примера применения кода с обнаружением ошибок в микроконтроллерных системах приведем второй и третий режимы работы последовательного порта микроконтроллера МК-51, при использовании которых в дополнение к восьми информационным разрядам передается девятый бит, которому может быть программно поставлено в соответствие значение бита четности количества единиц в передаваемой посылке.

5.2.2. Аппаратные средства диагностирования.

Аппаратные средства диагностирования представляют собой технические устройства, позволяющие проводить поиск отказов в вычислительной системе.

Общетехнические средства, логические пробники, логические пульсаторы, и индикаторы тока, а также способы их применения были рассмотрены в разделах 2.1.1 - 2.1.4. В дополнение необходимо отметить их существенное значение при невозможности применения более удобных в применении программных средств диагностирования (например, по причине неисправностей ядра микроконтроллерной системы). Указанные аппаратные средства диагностирования позволяют определить параметры питающего напряжения, наличие импульсов системной синхронизации, выдачу сигналов на шины адреса, данных и управления микроконтроллера и

диапазон изменения значений этих сигналов, что позволяет сделать некоторые предварительные заключения о причинах неработоспособности ядра системы (неисправности блока питания, блока системной синхронизации, отказ микроконтроллера, искажение программы в памяти и переход в непредусмотренную область памяти и т.п.).

Средства тестирования нагрузками предназначены для преднамеренного изменения параметров окружающей среды с целью анализа влияния этого изменения на работу микроконтроллерной системы. Применяются три разновидности такого тестирования - механические, температурные и электрические нагрузки.

Использование механических нагрузок заключается в оказании на аппаратуру системы воздействий вибрации и деформации (как правило, деформации кручения). Нагрузки такого типа позволяют определить неисправности соединительных контактов и проводников (в частности, тонкий разрыв печатного проводника при скручивании платы исчезает, восстанавливая утраченное соединение).

Использование температурных нагрузок заключается в оказании на всю аппаратуру системы либо на ее некоторую часть воздействий повышенных или пониженных температур относительно рабочей температуры эксплуатации изделия. Методика основана на том, что при выработке ресурса у ряда электронных элементов системы сужается диапазон температур, в котором их функционирование является достоверным. Штатное незначительное повышение температуры окружающей среды вследствие прогрева оборудования приводит к отказу таких элементов. Для выявления подобных ситуаций используется искусственное повышение и/или понижение температуры печатной платы, ее отдельных участков, а затем и элементов либо бесконтактным способом (посредством воздухонагнетающих устройств), либо контактным способом (посредством нагреваемых или охлаждаемых пластин).

Использование электрических нагрузок заключается в изменении значений параметров подаваемых на систему электрических сигналов. Данный способ позволяет определить рабочий диапазон значений параметров сигналов, а также выявить элементы, работающие на границе допустимого диапазона. Метод, в частности, позволяет определить элементы, функционирующие неверно в связи с недостаточно высоким уровнем напряжения питания.

В заключение рассмотрим проведение диагностирования посредством сигнатурных анализаторов.

Сигнатурный анализ основан на преобразовании последовательности двоичных сигналов в число (сигнатуру), существенно зависящее от исходной последовательности. Существует несколько способов получения сигнатур: подсчет количества фронтов и срезов в сигнале, подсчет количества единиц, вычисление контрольной суммы, использование сдвиговых регистров с обратными связями и т.п.

Рассмотрим последний способ как наиболее эффективный.

Пример сдвигового регистра с обратными связями показан на рис. 5-5.

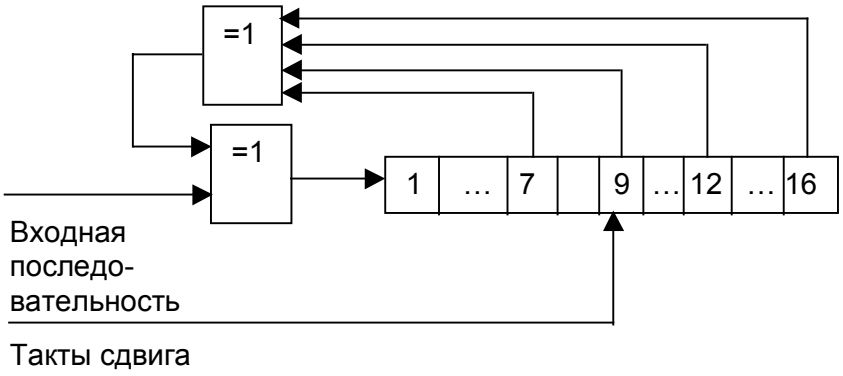


Рис. 5-5. Устройство регистра с обратной связью.

Входная последовательность двоичных сигналов объединяется с помощью ячейки "исключающее ИЛИ" с определенными разрядами сдвигового регистра, и поступает на его вход в следующем такте, сдвигая накопленное значение на один разряд в сторону старших разрядов. Таким образом, входной двоичный поток делится на порождающий полином с формированием в регистре остатка. Для достаточно длинной входной последовательности остаток будет являться псевдослучайной величиной, неизменной для данной последовательности и данного порождающего полинома. Благодаря запоминающим элементам регистра и обратной связи схема учитывает предыдущие состояния; значительная разрядность регистра и правильно подобранный характеристический полином обеспечивают уникальность сигнатур и, следовательно, высокую достоверность обнаружения искажений входной последовательности.

Устройство сигнатурного анализатора, использующего сдвиговый регистр с обратной связью, показано на рис. 5-6. В состав системы входят также: устройство управления, формирующее на основе сигналов "пуск" и "стоп" временное окно накопления сигнатуры, а также устройство индикации для визуального представления сигнатуры. Внешний вид сигнатурного анализатора приведен на рис. 5-7.

В качестве основы для тестовой программы может выступать как специально разработанное программное обеспечение, так и рабочая программа, в доста-

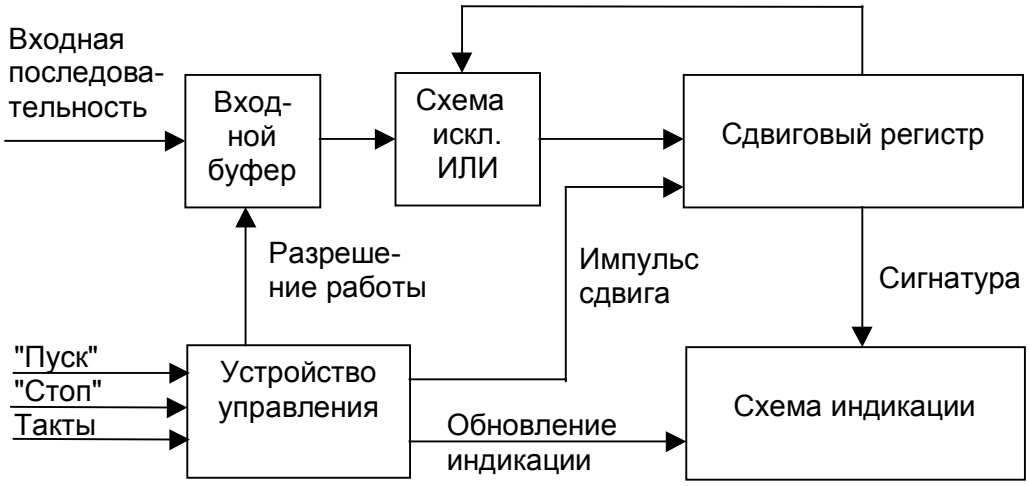


Рис. 5-6. Структура сигнатурного анализатора.

точной мере использующая устройства микроконтроллерной системы.

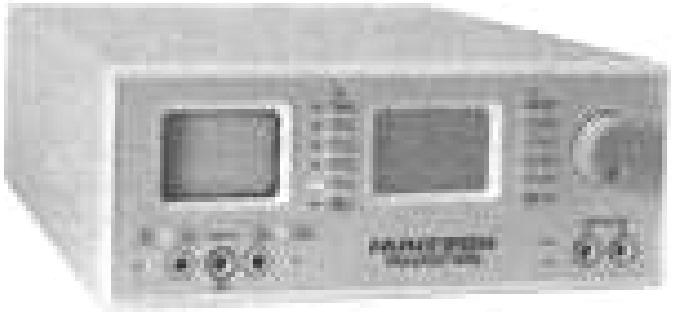


Рис. 5-7. Внешний вид сигнатурного анализатора.

Сигналы "пуск" и "стоп" формируются либо средствами тестовой программы при начале/окончании теста, либо аппаратными средствами при наличии на шине адреса соответствующих комбинаций сигналов.

На заведомо исправной микроконтроллерной системе в выбранных контрольных точках производится фиксация эталонных сигнатур.

При диагностировании системы используется та же тестовая последовательность и то же временное окно, что и в процессе формирования эталонных сигнатур. Неисправность заключается в участке, для которого все входные сигнатуры совпадают с эталонными, а хотя бы одна выходная сигнатура не совпадает.

5.2.3. Программно-аппаратные средства диагностирования.

К программно-аппаратным средства диагностирования относятся дублирующие системы, устройства хранения тестов-эталонов, контролирующие автоматы.

Обобщенная структура систем дублирования показана на рис. 5-8.

Входные сигналы X поступают на эталонную ($МКС_{ЭТ}$) и диагностируемую ($МКС_{Д}$) системы. Выходные сигналы $У_{ЭТ}$ и $У_{Д}$ поступают на вход анализатора А, определяющего соответствие выходных сигналов диагностируемой и эталонной систем. Поскольку входные сигналы обеих систем одинаковы, а система $МКС_{ЭТ}$ считается заведомо исправной, факт несовпадения $У_{Д}$ с $У_{ЭТ}$ свидетельствует о неисправности диагностируемой системы.

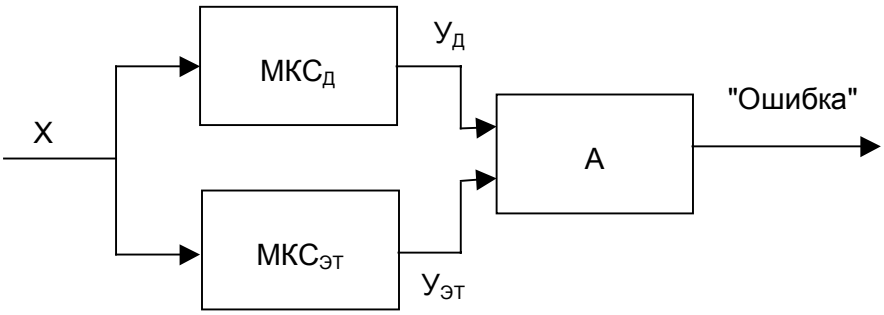


Рис. 5-8. Структура системы дублирования.

В связи с тем, что эксплуатация резервированных микроконтроллерных систем в большинстве случаев неоправданна (в связи с ухудшением массогабаритных, стоимостных и других характеристик), системы дублирования применяются лишь при проведении тестового диагностирования.

Одним из вариантов построения системы дублирования является применение специального запоминающего устройства для хранения неизбыточных тестовых последовательностей. Структура подобных систем изображена на рис. 5-9.

В данном варианте порядок следования тестовых сигналов определяется устройством управления исходя из выбранного пользователем режима тестирования и результатов сравнения реакций эталонной и диагностируемой систем на предыдущих шагах теста (например, проведение расширенного тестирования для более точного определения характера неисправности при несовпадении результатов).

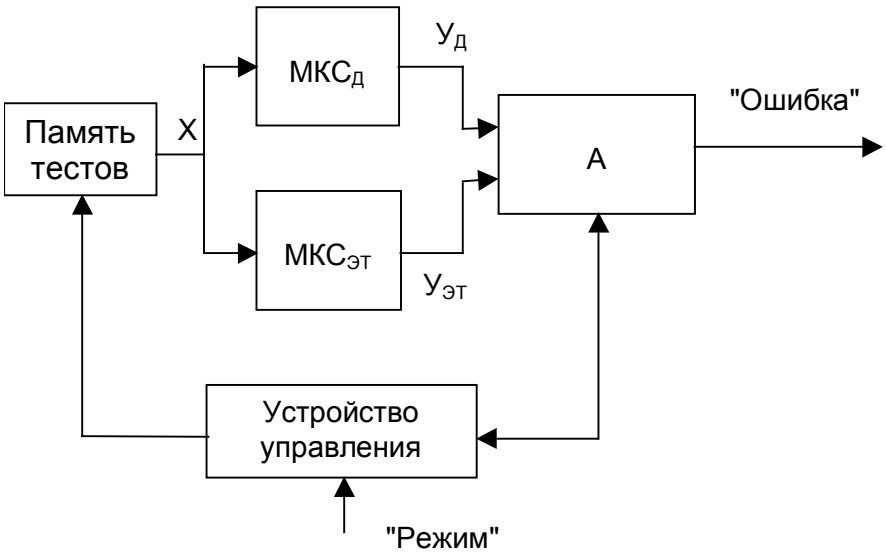


Рис. 5-9. Вариант структуры системы дублирования.

Второй класс программно-аппаратных средств диагностирования - системы хранения тестов-эталонов (см. рис. 5-10), являющиеся логическим развитием описанной выше структуры.

В данном случае эталонные выходные сигналы генерируются не эталонной системой, а хранятся в памяти наряду со входными тестовыми сигналами. Такие системы по сравнению с системами дублирования обладают лучшими стоимостными и массогабаритными характеристиками, а также снимают проблему возможных появлений неисправностей в эталонной системе.

К третьему классу программно-аппаратных средств диагностирования от-

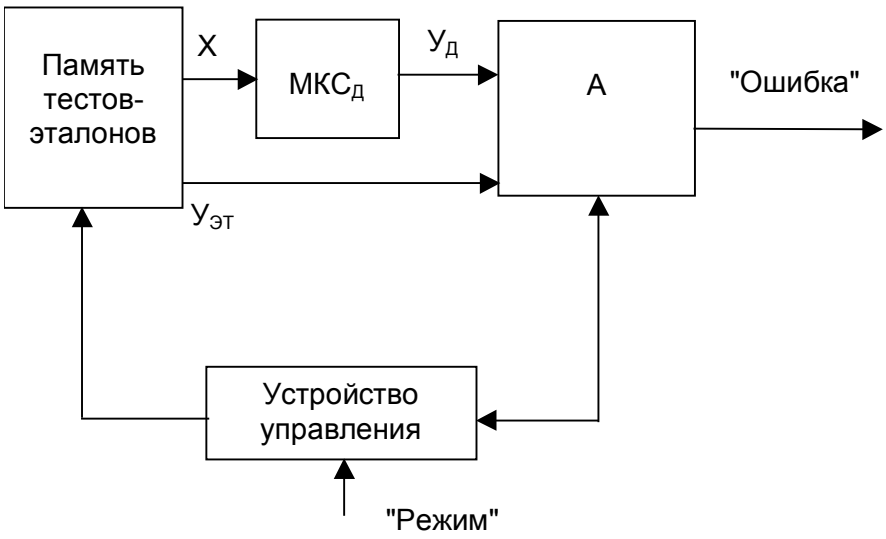


Рис. 5-10. Структура системы "память тестов-эталонов".

носятся контролирующие автоматы. Контролирующие автоматы строятся для контролируемого автомата А и известного класса ошибок в нем, подлежащих обнаружению в виде автомата A_E (см. рис. 5-11).

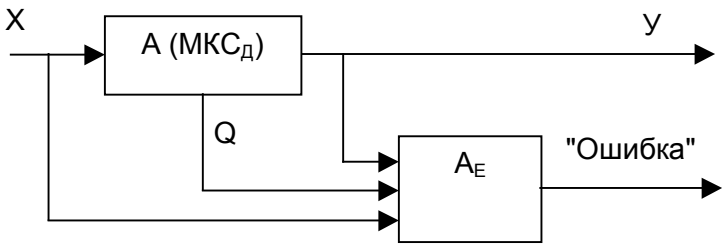


Рис. 5-11. Структура контролирующего автомата.

Ошибкой в автомате А является искажение его выходных сигналов У, а также его внутреннего состояния Q (следует отметить, что искажение Q может не влиять на У в течение некоторого времени, т.е. для внешнего наблюдателя диагностируемая система может продолжать функционировать достоверно). Задачей контролирующего автомата является обнаружение любой ошибки в контролируемом автомате в момент ее возникновения, в связи с чем на вход автомата A_E подаются не только входы X и выходы У автомата А, но и его внутренние состояния Q.

5.3. Процедура проведения диагностирования.

В режим диагностирования система переводится либо в случае аварийной ситуации (обнаружении на ее выходах неверных сигналов), либо в случае проведения плановой проверки.

При применении тестового диагностирования входы системы переводятся в режим приема воздействий системы тестирования; при применении функционального диагностирования на входы системы продолжают поступать рабочие воздействия.

Выходные реакции системы сравниваются с эталонными реакциями, построенными на основе модели функционирования системы (см. рис. 5-2 и 5-3). В случае их несовпадения производится диагностика модулей.

Модули системы диагностируются последовательно в соответствии с планом диагностирования. Выявленная отказовая ситуация не всегда позволяет точно определить неисправный модуль, в связи с чем при восстановлении системы проводится последовательная замена всех подозреваемых модулей.

РАЗДЕЛ 6. ПРИМЕРЫ РАЗРАБОТОК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МИКРОКОНТРОЛЛЕРНЫХ СИСТЕМ.

6.1. Арифметико-логические вычисления на микроконтроллере.

В качестве иллюстрирующего примера рассмотрим подпрограмму, реализующую упорядочение содержимого массива данных (например, результатов серии

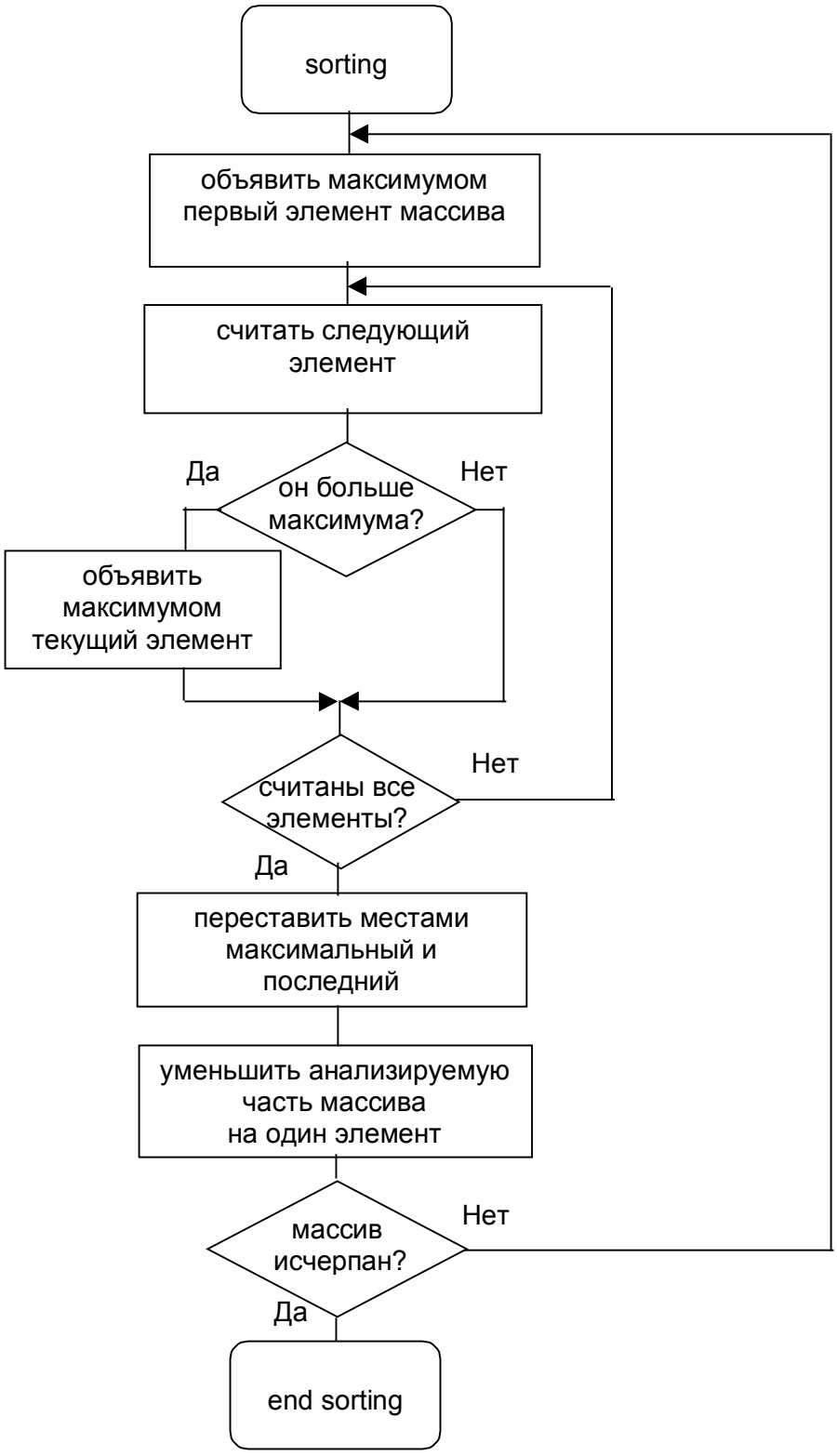


Рис. 6-1. Схема программы сортировки массива.

измерений) по возрастанию значений. Схема программы приведена на рис. 6-1.

Программа производит определение максимального элемента и помещает его в последнюю позицию, меняя местами с элементом, стоявшим последним. После этого производится сдвиг границы массива так, чтобы в анализируемой части остались еще не отсортированные элементы, и процедура повторяется до исчерпания массива.

Текст программы на языке ассемблера для микроконтроллера МК-51 приведен на рис. 6-2 и 6-3.

```
                org 8100h
start_adr:      equ 8200h           ;адрес начала массива
num_data:       equ 30h             ;количество элементов в нем
tmp_max:        equ 31h             ;локальный максимум
adr_hi:         equ 32h             ;его адрес: старший байт
adr_lo:         equ 33h             ;           младший байт

main:           mov num_data,#15    ;выполнение сортировки
                lcall sorting       ;пятнадцати элементов
                ret

sorting:        push psw            ;сохранение используемых
                push a              ;ресурсов МК
                push 0
                push 1
                push dph
                push dpl
                mov r0,num_data     ;изначально неотсортирован весь массив

$step:          cjne r0,#2,$chck_len ;в массиве должно быть
$chck_len:      jnc $ok_len         ;больше одного элемента
                ljmp end_sort

$ok_len:        mov r1,0            ;определение количества
                                    ;несортированных элементов

                mov dptr,#start_adr ;чтение первого из них
                movx a,@dptr
                mov tmp_max,a       ;объявление первого
                mov adr_hi,dph      ;элемента максимальным
                mov adr_lo,dpl
                dec r1

$search:        inc dptr            ;поиск максимального
                movx a,@dptr        ;среди оставшихся
                cjne a,tmp_max,$cmp
$cmp:           jc $no_max
$max:           mov tmp_max,a       ;обновление максимума
                mov adr_hi,dph
                mov adr_lo,dpl
$no_max:        djnz r1,$search
```

Рис. 6-2. Текст программы сортировки массива. Начало.

```
                movx a,@dptr        ;перестановка последнего
                xch a,tmp_max       ;элемента в анализируемом
                movx @dptr,a        ;участке массива
                mov dph,adr_hi      ;и максимального элемента
                mov dpl,adr_lo      ;в этом участке
                mov a,tmp_max
                movx @dptr,a

                djnz r0,$step       ;сдвиг границы
                                    ;неотсортированной части
                                    ;массива

end_sort:       pop dpl             ;восстановление
                pop dph             ;используемых ресурсов МК
                pop 1
                pop 0
                pop a
                pop psw
                ret

                org 8200h           ;данные, подлежащие сортировке
data:           db 12h,0,33h,41h,42h,43h,0FFh,0,0,1,2,0ABh,5,0C5h,87h
```

Рис. 6-3. Текст программы сортировки массива. Окончание.

Из основной программы main производится вызов подпрограммы sorting, в качестве параметров для которой сформированы начальный адрес массива и количество элементов в нем. Массив определен в области внешней памяти.

Примечание.

Следует отметить, что данный способ выявления окончания очередного прохода сортировки имеет ограничение на объем массива, определяемое разрядностью переменной num_data, а также способом организации циклов на основе команды djnz - в массиве может быть не более 255 элементов.

Снять данное ограничение возможно, например, введением двухбайтных счетчиков элементов массива и использованием команд cjne при выявлении ситуации выхода за пределы анализируемой области массива.

6.2. Ввод информации в микроконтроллерную систему.

6.2.1. Опрос пользовательского пульта.

В большинстве встраиваемых систем управления необходимо организовывать связь с оператором для указания параметров работы системы.

Рассмотрим задачу определения нажатой клавиши на шестнадцатиклавишном пульте, схема подключения которого к микроконтроллеру 80C515 показана на рис. 6-4.

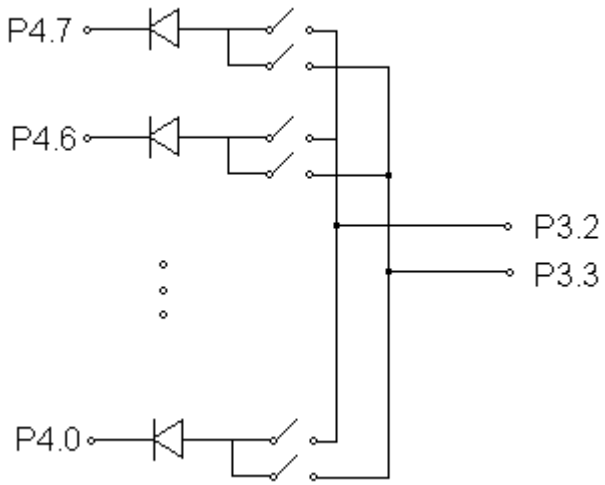


Рис. 6-4. Схема подключения пользовательского пульта.

Клавиши объединены попарно в матрицу размерности 8x2. Столбцы матрицы клавиатуры соединены с линиями порта P4. В реализованной схеме низкий уровень на одной из выходных линий порта P4 обеспечивает опрос соответствующего столбца матрицы. Строки матрицы подключены к выводам P3.3 и P3.2 порта P3. Низкий уровень на этих линиях идентифицирует включенные клавиши, опрашиваемые в текущий момент времени.

Для определения состояния клавиатуры необходимо сформировать входные сигналы опроса клавиатуры и считать ее состояние. Опрос клавиатуры удобно выполнять, организовав выдачу последовательности “бегущий нуль” по столбцам клавиатуры, считывая при этом значение строк клавиатуры в каждый такт формирования нового входного воздействия. Совокупность значений строк матрицы клавиатуры назовем картой состояния.

Схема программы приведена на рис. 6-5, а ее текст - на рис. 6-6.

Номер нажатой клавиши определяется как (X-start)*2+1 для клавиш, отображаемых на разряд P3.3 и как (X-start)*2+2 для клавиш, отображаемых на разряд P3.2, где X - номер нулевого элемента карты состояния, start - адрес начала карты.

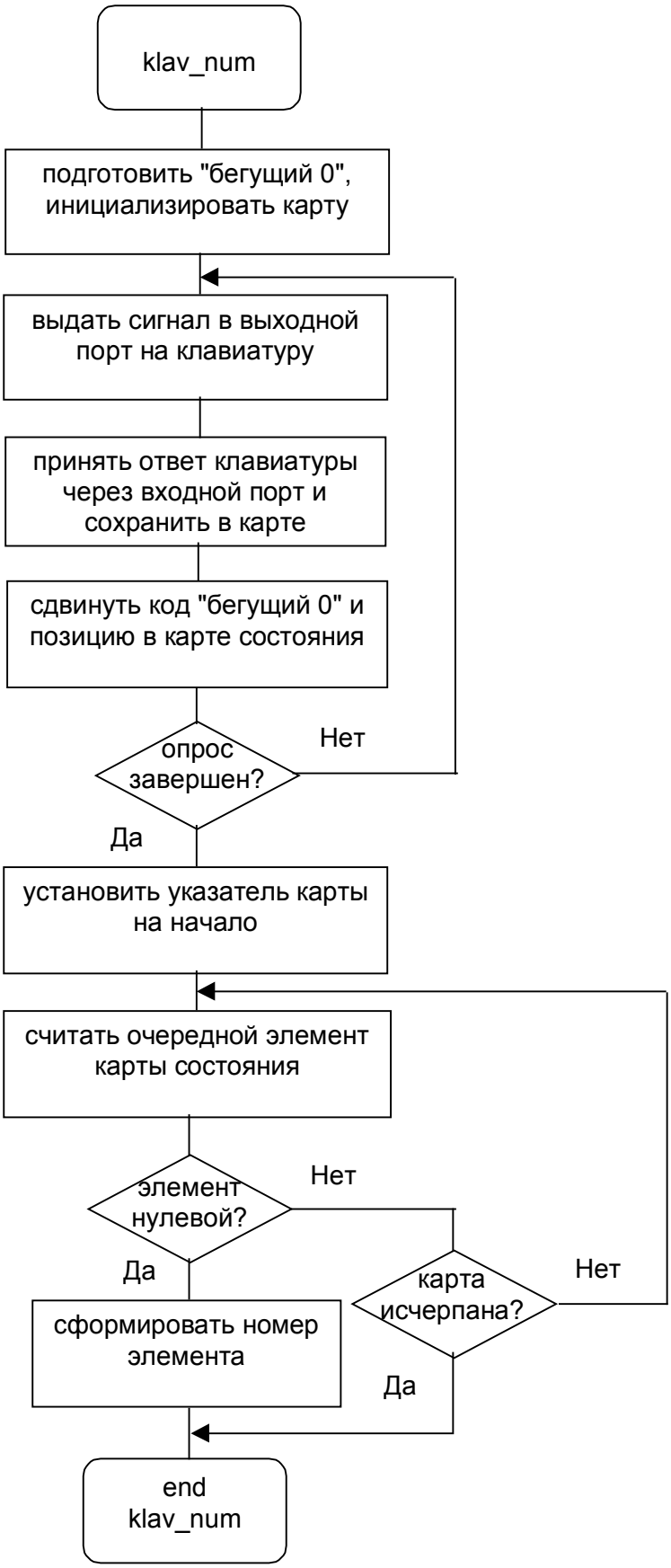


Рис. 6-5. Схема программы опроса пульта.

```
num:        equ 40h                 ;номер нажатой клавиши
map_start:  equ 30h                 ;начало области хранения карты клавиатуры

klav:       push psw                ;сохранение используемых
            push 0                  ;ресурсов МК
            push a
                                    ;чтение карты в память МК
            mov a,#07Fh             ;подготовка "бегущего нуля"
            mov r0,#map_start       ;адрес начала карты
                                    ;состояния клавиатуры

$scan:      mov 0E8h,a              ;выдача сигнала в порт 4
            setb p3.3
            setb p3.2
            mov @R0,P3              ;чтение ответа из порта 3
            inc r0                  ;в очередную позицию карты
            setb c
            rrc a                   ;подготовка нового опроса
            cjne a,#11111111b,$scan
                                    ;дешифрация карты
            mov r0,#map_start
$check:     mov a,@r0
            jnb acc.3,$odd          ;выявление нажатой клавиши
                                    ;с нечетным номером
            jnb acc.2,$even         ;выявление нажатой клавиши
                                    ;с четным номером
            inc r0                  ;просмотр карты далее
            cjne r0,#(map_start+8),$check

            mov a,#0                ;нажатых клавиш нет
            ljmp klav_end

$odd:       mov a,r0                ;вычисление номера
            clr c                   ;нажатой клавиши
            subb a,#map_start       ;для нечетных позиций
            mov b,#2
            mul ab
            add a,#1
            ljmp klav_end

$even:      mov a,r0                ;вычисление номера
            clr c                   ;нажатой клавиши
            subb a,#map_start       ;для четных позиций
            mov b,#2
            mul ab
            add a,#2

klav_end:   mov num,a               ;запись в результирующую ячейку
            pop a                   ;восстановление
            pop 0                   ;используемых ресурсов МК
            pop psw
            ret
```

Рис. 6-6. Текст программы опроса пульта.

6.2.2. Опрос д оговых величин.

В системах управления одной из типовых задач является задача определения показаний д вляющих информацию в виде аналогового сигнала - уров

Перед разработчиком в данном случае возникают две задачи - масштабирования сигналов и бразования.

Первая з нием схемотехнических решений (внешних по отношению к микроконтроллеру узлов масштабирования, выполненных, например, на базе о . Вторая задача решается либо схемотехнически (в случ го в микроконтроллер АЦП), либо программно (при наличии данного периферийного узла).

Рассмотрим подс змерения уровня освещенности, использующей в качестве датчика фотодио ципиальная схема устройства сопряжения с МК 80С515 показана на рис. 6-7.

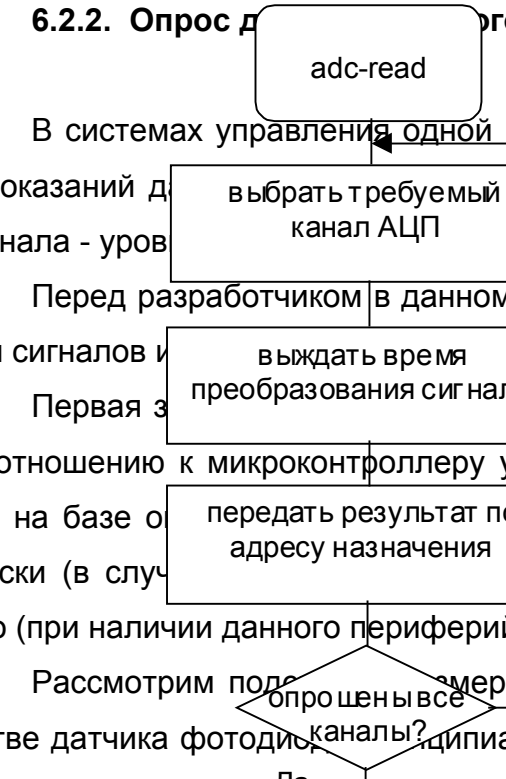


Рис. 6-8. Схема программы опроса датчиков аналоговых сигналов.

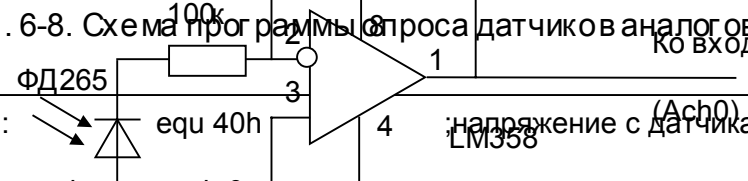


Рис. 6-7. Сопряжение датчика с МК.

```
U1:         equ 40h         4   ;напряжение с датчика

U1_read:    push 0              ;процедура опроса датчиков
            mov r0,#0           ;выбор нулевого
            lcall adc_switch    ;канала
            mov 0DAh,#0h        ;задание диапазона
            lcall adc_delay     ;задержка
            mov U1,0D9h         ;сохранение в U1
            pop 0
            ret

adc_switch: mov a,0D8h          ;настройка АЦП:
            anl a,#11111000b    ;непрерывное
            orl  a,#00001000b   ;преобразование
            orl a,r0            ;выбор канала
            mov 0D8h,a
            ret

adc_delay:  push 0              ;выдержка паузы
            mov r0,#20          ;для преобразования
            djnz r0,$           ;в АЦП
            pop 0
            ret
```

Рис. 6-9. Текст программы опроса датчиков аналоговых сигналов.

ЦП. Диа-

пазон мас

мого для

выполнени 9 мкс при

смене обо

и приме-

нение пре

6.2.3. Определение длительности временных интервалов.

Важной задачей при построении управляющих систем является задача определения времени продолжительности события, представленного двоичным сигналом или совокупностью двоичных сигналов на входах микроконтроллера.

Рассмотрим задачу определения времени продолжительности события, начало которого характеризуется переходом из состояния логической единицы в состояние логического нуля двоичного сигнала У1, а конец - переходом из состояния логической единицы в состояние логического нуля двоичного сигнала У2.

Для удобства решения задачи подадим сигнал У1 на вход первого запроса прерывания МК, а сигнал У2 - на вход нулевого запроса прерывания (см. рис. 6-10).

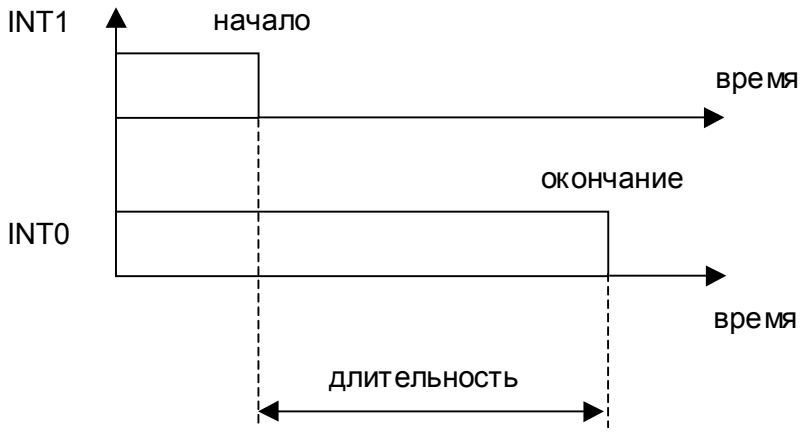
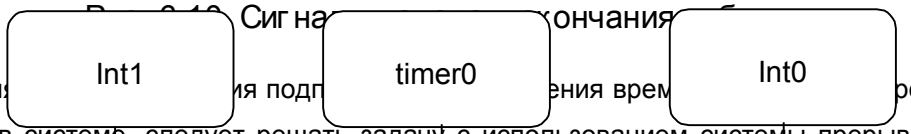


Рис. 6-10. Сигналы начала и окончания события



Для ... ия подп... ...ения врем... ...рочие процессы, реализуемые в системе, следует решать задачу с использованием системы прерываний. По срезу си... ...о-щ... ...е, об... ...е-м... ...с-то... ...а вр... ...в пр...

```
cnt_hi:     equ 30h          ;счетчик текущего
cnt_lo:     equ 31h          ;времени
last_tim_h: equ 40h          ;длительность
last_tim_l: equ 41h          ;предыдущего цикла

main:       lcall init       ;основная программа
loop:       ……..
            sjmp loop
init:                        ;инициализация переменных и устройств:
            mov cnt_hi,#0    ;обнуление
            mov cnt_lo,#0    ;счетчиков
            mov last_tim_h,#0
            mov last_tim_l,#0
            mov a,TMOD       ;таймер Т0 в режиме
            anl a,#11110000b ;перезагрузки
            orl  a,#00000010b ;будет отмерять
            mov TMOD,a       ;интервалы времени
            mov TH0,#155     ;в 100 мкс
            mov TL0,#0
            setb et0         ;разрешение работы обработчиков
            setb ex1         ;необходимых
            setb ea          ;типов прерываний
            ret
;---------------------------------------------------------------
```

Рис. 6-12. Текст программы определения длительности события. Начало.

```
;---------------------------------------------------------------
int1:                                   ;обработка события
                                        ;старт-импульса:
                mov cnt_hi,#0           ;начало нового цикла
                mov cnt_lo,#0           ;измерений
                mov TL0,#0
                setb tr0                ;запуск таймера
                setb ex0                ;подготовка
                                        ;к распознаванию
                                        ;стоп-импульса
                clr ex1                 ;блокировка прерывания
                                        ;данного типа
                reti
;---------------------------------------------------------------
int0:                                   ;обработка события
                                        ;стоп-импульса:
                clr tr0                 ;запрет дальнейшего счета
                clr ex0                 ;блокировка прерывания
                                        ;данного типа
                mov last_tim_h,cnt_hi   ;сохранение измеренного
                mov last_tim_l,cnt_lo   ;времени
                setb ex1                ;подготовка
                                        ;к распознаванию
                                        ;очередного старт-импульса
int0_ex:        reti
;---------------------------------------------------------------
tim0:           push psw
                push a
                inc cnt_lo              ;увеличение значения
                mov a,cnt_lo            ;длительности импульса
                jnz tim0_ex
                inc cnt_hi
tim0_ex:        pop a
                pop psw
                reti
;---------------------------------------------------------------

                                        ;переходы к обработчикам
                                        ;необходимых типов прерываний
                org 0003h
                ljmp int0
                org 000bh
                ljmp tim0
                org 0013h
                ljmp int1
```

Рис. 6-13. Текст программы определения длительности события. Окончание.

В переменных last_tim_h и last_tim_l хранятся соответственно старший и младший байты длительности последнего завершившегося события в квантах времени по 100 микросекунд.

6.3. Вывод информации из микроконтроллерной системы.

6.3.1. Вывод цифровых кодовых последовательностей.

В большинстве встраиваемых систем управления необходимо организовывать связь с оператором для вывода информации о параметрах работы системы.

Рассмотрим задачу вывода алфавитно-цифровой информации на жидкокристаллический алфавитно-цифровой индикатор (ЖКИ) модели DM2021 фирмы Sanyo. Данная модель ЖКИ имеет поле вывода информации размером две строки по двадцать символов в каждой. ЖКИ содержит видеопамять, в которой хранятся отображаемые символы, а также обладает собственной системой управления жидкокристаллической панелью. Наличие указанных узлов существенно упрощает работу с ЖКИ, так как собственно управление отображением точек - элементов изображения - производится автоматически в соответствии с поданной на ЖКИ командой.

ЖКИ содержит восьмиразрядную шину команд-данных и шину управления, в состав которой входят одноразрядные линии разрешения программирования (E), выбора типа посылки "команда-данные" (RS) и выбора направления передачи данных "чтение-запись" (RW). Подключение ЖКИ к МК 80C515 показано на рис. 6-14. Порт P4 предназначен для организации шины команд-данных, а старшие три бита порта P1 предназначены для организации шины управления. Переменный резистор необходим для регулировки контрастности изображения.

Процедура записи в ЖКИ выполняется в три этапа: на шине DB устанавливается информация, отражающая подаваемые в ЖКИ команды-данные, затем устанавливаются необходимые значения RW и RS и, наконец, на входе E формируется переход от высокого логического уровня к низко-

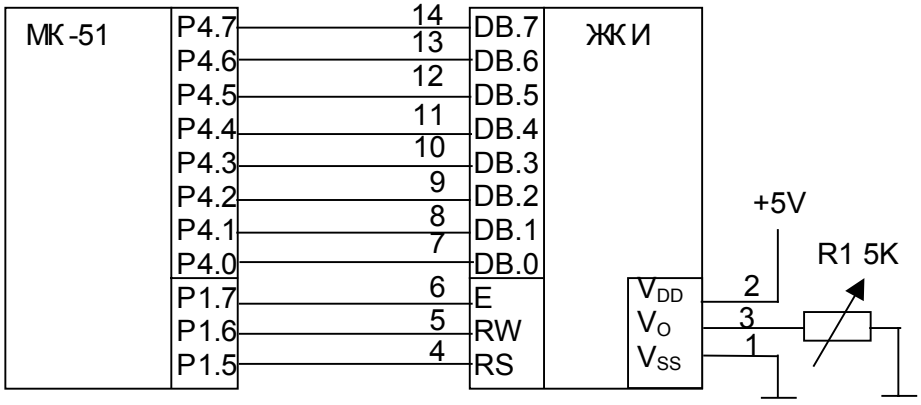


Рис. 6-14. Сопряжение ЖКИ с МК-51.

му. Для возврата системы управления ЖКИ в исходное состояние следует перевести вход E в состояние логической единицы.

Не приводя всей системы команд ЖКИ, остановимся на следующих из них: 38h - установка восьмибитного режима обмена с ЖКИ, использование для вывода обеих строк с размером символа 5x7 точек; 0Ch - активизация всех знакомест ЖКИ в режиме погашенного курсора; 80h - установка адреса, начиная с которого записываемые в ЖКИ данные будут последовательно располагаться в видеопамяти.

Схема и текст программы, осуществляющей вывод на ЖКИ сорока символов, коды которых расположены во внешней памяти данных микроконтроллера, начиная с адреса 0FFD0h, приведены соответственно на рис. 6-15 и рис. 6-16.

Примечание. Требуемое производителем время выдержки сигналов RW и RS относительно E составляет доли микросекунды, в связи с чем задержка в процедуре записи в ЖКИ необязательна. Однако время отработки каждой из используемых в программе команд в ЖКИ составляет сорок микросекунд, в связи с чем между подаваемыми командами должна выдерживаться соответствующая пауза. В данной программе в процедуру выдачи команд indic_wr включены задержки, обеспечивающие суммарное время выполнения процедуры тридцать шесть микросекунд, а с учетом времени подготовки данных и времени вызова - требуемые сорок микросекунд.

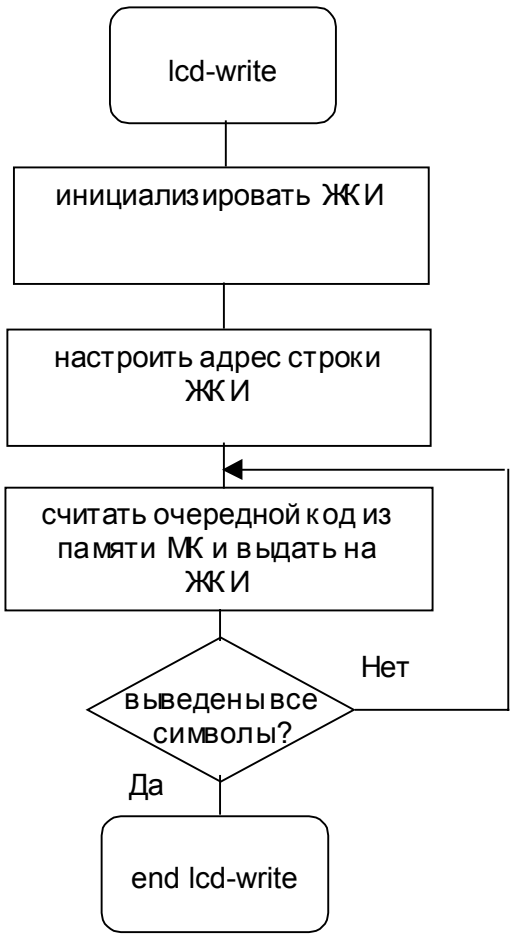


Рис. 6-15. Схема программы вывода информации на ЖКИ.

```
indic:
indic_init:         mov indic_w1,#0          ;инициализация ЖКИ
                    mov indic_w0,#38h
                    lcall indic_wr
                    mov  indic_w0,#0Ch
                    lcall indic_wr
                    mov  indic_w0,#80h       ;установка адреса первого символа
                    lcall indic_wr           ;первой строки
                    mov  indic_w1,#1
                    mov dptr,#0FFD0h
indic_data_wr_1:    movx a,@dptr             ;вывод символов первой строки
                    mov  indic_w0,a
                    lcall indic_wr
                    inc dptr
                    mov a, dpl
                    cjne a,#0E4h, indic_data_wr_1
                    mov  indic_w1,#0
                    mov  indic_w0,#0C0h      ;установка адреса первого символа
                    lcall indic_wr           ;второй строки
                    mov  indic_w1,#1
indic_data_wr_2:    movx a,@dptr             ;вывод символов второй строки
                    mov indic_w0,a
                    lcall indic_wr
                    inc dptr
                    mov a, dpl
                    cjne a,#0F8h, indic_data_wr_2
indic_exit:         ret
indic_wr:           mov 0e8h, indic_w0       ;процедура записи в ЖКИ
                    setb p1.7
                    clr p1.6
                    mov a,indic_w1
                    mov c,acc.0
                    mov p1.5,c
                    lcall indic_delay
                    clr p1.7
                    lcall indic_delay
                    setb p1.7
                    ret
indic_delay:        nop                      ;процедура программной задержки
                    nop
                    nop
                    nop
                    nop
                    nop
                    nop
                    nop
                    ret
                    org 0FFD0h
data:   db          'It is test examlpe. '   ;пример организации выводимых
        db          '0123456789ABCDEF!@#$'   ;данных
```

Рис. 6-16. Текст программы вывода данных на ЖКИ.

6.3.2. Вывод ШИМ-сигналов.

В ряде задач управления возникает необходимость выдачи на объект управляющих сигналов определенной интенсивности (либо заданной мощности за некоторый период). Как правило, для объектов с аналоговым управлением применяется формирование управляющих сигналов с изменяемой амплитудой.

Один из вариантов реализации такого управления заключается в использовании широтно-импульсной модуляции (ШИМ). В данном варианте на одном из разрядов выходного порта микроконтроллера формируются периодические импульсные сигналы с постоянной частотой следования и изменяемым отношением времени длительности импульса ко времени длительности паузы, которое определяет интенсивность управляющего сигнала.

Рассмотрим реализацию ШИМ на примере управления частотой вращения двигателя постоянного тока. Схема устройства сопряжения МК 80С515 с двигателем показана на рис. 6-17. Схема программы формирования ШИМ-сигналов приведена на рис. 6-18, а ее текст - на рис. 6-19.

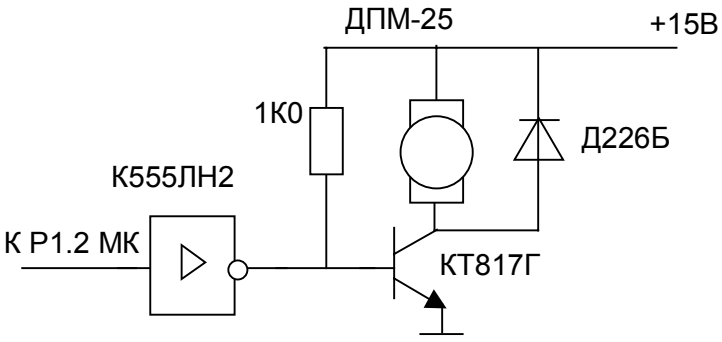


Рис. 6-17. Схема устройства сопряжения.

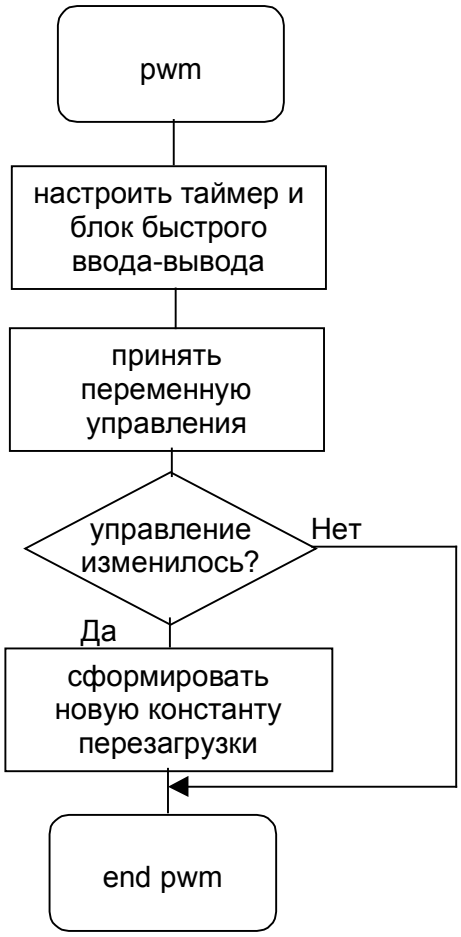


Рис. 6-18. Схема программы ШИМ-управления.

```
main:       lcall init_pwm              ;инициализация ШИМ
loop:       …
            lcall pwm_contr             ;задание ШИМ
            sjmp loop

init_pwm:   mov old_pwm,#255            ;предыдущая константа перезагрузки
            orl   p1,#00001000b         ;вывод через третий регистр меток
                                        ; - через разряд P1.3
            mov cbh,#0e0h               ;константа перезагрузки соответствует
            mov cah,#080h               ;частоте ШИМ в 124 Гц

            mov c8h,#00010001b          ;таймер 2 в режиме неуправляемого счета,
                                        ;с выводом ШИМ и автоперезагрузкой
            mov c1h,#10000000b          ;разрешен вывод через третий регистр меток
            mov c6h,#80h                ;младший байт третьего регистра меток
            ret

pwm_contr:  mov a,pwm_var               ;прием управления
            cpl a                       ;расчет константы времени
            mov b, #8
            div ab
            cjne a,old_pwm,pwm_ch
            ljmp pwm_exit
pwm_ch:     mov old_pwm,a               ;в случае изменившегося управления
            add a,#224
            mov c7h,a                   ;вывод в старший байт регистра
pwm_exit:   ret
```

Рис. 6-19. Текст программы ШИМ-управления.

Примечания.

Примечание 1. В данной программе реализуется частота ШИМ-сигналов в 124 Гц. Для этого используется режим автоперезагрузки таймера, позволяющий начинать счет с наперед заданного в регистре перезагрузки значения. В регистр следует занести константу, определяемую как $N=65536-10^6 / f_{ШИМ}$, где $f_{ШИМ}$ - желаемая частота ШИМ-сигналов в герцах. Начав с константы N и считая тактовые импульсы с частотой 1 МГц, таймер сформирует очередной период ШИМ за $10^6 / f_{ШИМ}$ микросекунд. Для данного примера N=E080h.

Примечание 2. Для корректной реализации ШИМ с усеченным периодом необходимо масштабировать помещаемую в регистр меток константу ШИМ так, чтобы возрастающий с величины N код таймера обладал возможностью ее превысить (т.е. константа ШИМ должна быть не меньше N). В данном случае диапазон кода управления pwm_var 0…255 масштабируется в диапазон 224..255.

Примечание 3. Использование схем внешних интеграторов позволяет получить сигнал с амплитудой, пропорциональной константе ШИМ (см. раздел 3).

6.3.3. Вывод сигналов с временным сдвигом.

В заключение рассмотрим задачу формирования временного интервала с управляемой длительностью, начало которого определяется внешним старт-сигналом (задача синхронизации).

Требуемая временная диаграмма показана на рис. 6-20.

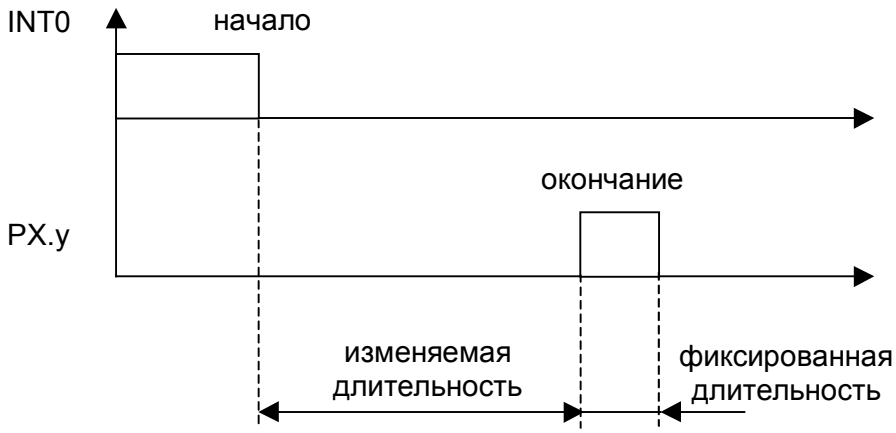


Рис. 6-20. Формируемый временной интервал.

Для возможности гибкого использования программного комплекса, реализующего данный вариант управления, целесообразно организовать его с использованием возможностей, предоставляемых системой прерываний. При возникновении запускающего события обработчик прерывания INT0 запускает таймер 0, отсчитывающий заданный основной программой временной интервал, по истечении которого обработчик прерывания таймера 0 на определенном разряде выходного порта формирует высокий логический уровень. После этого обработчик перезапускает таймер с такими настройками, чтобы выдержать фиксированный интервал длительности импульса. Выполняясь вторично, обработчик прерывания таймера 0 на заданной выходной линии формирует низкий логический уровень.

Схемы управляющих программ и их текст приведены на рис. 6-21, рис. 6-22 и рис. 6-23 соответственно.

Программный комплекс формирует импульс длительностью 240 мкс, выдаваемый спустя 2…10 мс относительно начала запускающего перепада на входе INT0 в зависимости от величины кода kupr.

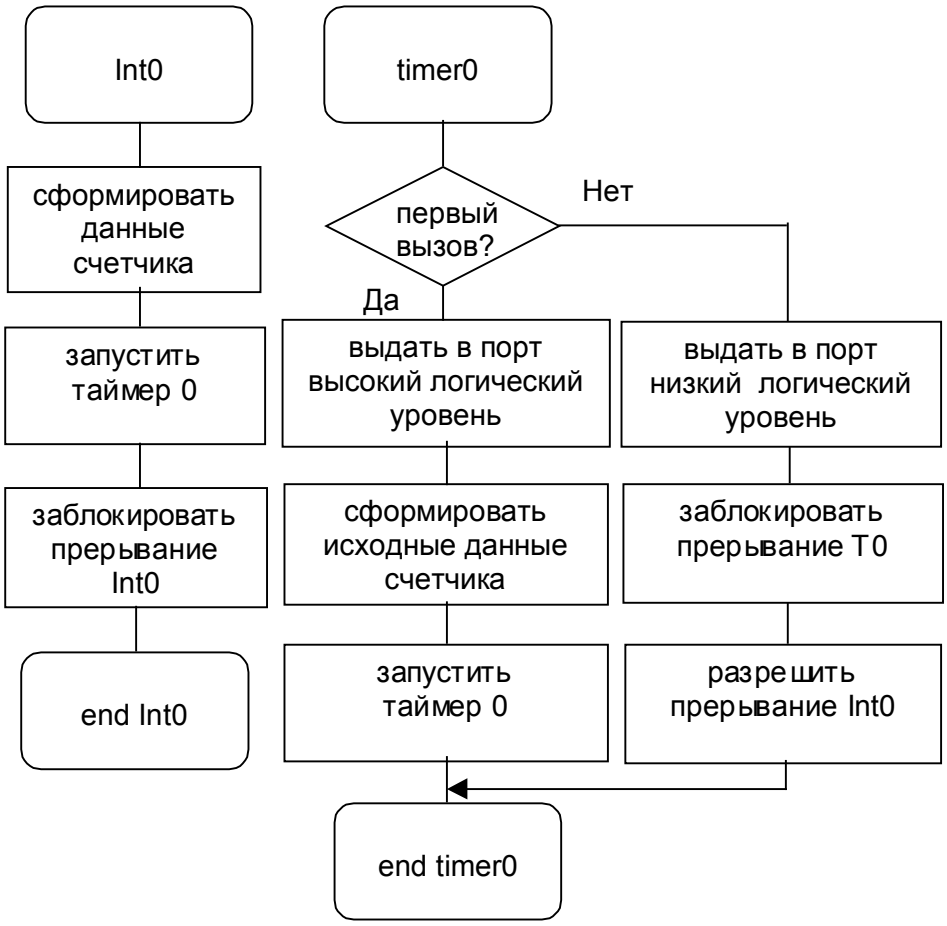


Рис. 6-21. Схема программы выдержки временного интервала.

```
kupr:       equ 30h

main:       lcall init
loop:       …
            sjmp loop

init:       mov a,tmod              ;таймер 0 - 16-битный счетчик
            anl a,#11110000b
            orl a, #00000001b
            mov tmod,a
            clr P1.2                ;обнуление выходной линии
            clr F0                  ;номер прохода: 0 -первый, 1 - второй
            setb it0                ;настройка системы прерываний
            setb ex0
            clr et0
            clr tr0
            ret
```

Рис. 6-22. Текст программы выдержки временного интервала. Начало.

```
                                ;подпрограмма обработки прерывания INT0
                                ;синхронизация по старт-импульсу
sync_impulse:       push 0
                    push acc
                    push b
                    push psw

                    mov a,kupr      ;определение временной выдержки
                    mov b,#8        ;максимальное время - 10 мс
                    div ab
                    add a, #7       ;минимальное время  -   2 мс
                    mov r0,a
                    mov a,#255
                    clr c
                    subb a,r0
                    mov th0,a       ;установка начальных показаний
                    mov tl0,#0d0h ;таймера
                    setb et0        ;настройка системы прерываний
                    setb tr0
                    clr ex0

                    pop psw
                    pop b
                    pop acc
                    pop 0
                    reti

                                ;подпрограмма обработки прерывания T0
switch:             jb F0, off      ;анализ номера вхождения в обработчик

on:                             ;выдать импульс на 240 мкс
                    setb F0
                    setb P1.2
                    mov th0,#255
                    mov tl0,#15
                    sjmp switch_exit

off:                            ;завершить импульс
                    clr F0
                    clr P1.2
                    setb ex0        ;подготовка к новому циклу
                    clr et0
                    clr tr0

switch_exit:        reti

                                ;переход к обработчикам прерываний
                    org 0003h
                    ljmp sync_impulse
                    org 000bh
                    ljmp switch
```

Рис. 6-23. Текст программы выдержки временного интервала. Окончание.

РАЗДЕЛ 7. ЛАБОРАТОРНЫЙ ПРАКТИКУМ.

7.1. Организация учебного исследовательского комплекса.

Основной задачей лабораторного практикума является формирование навыков разработки встраиваемых вычислительных систем на базе МК.

В рамках данного лабораторного практикума рассматриваются следующие вопросы:

- разработка аппаратуры: аспекты сопряжения и организации физического взаимодействия МК и объекта управления;

- разработка программного обеспечения: организация вычислений, программная поддержка ввода-вывода, решение задач реального времени и мультипроцессное программирование, организация межпроцессорного обмена и др.

Для этих целей используется автоматизированное рабочее место (см. рис. 7-1), включающее:

- инструментальную ЭВМ для разработки программного обеспечения микроконтроллера;

- набор контрольно-измерительных приборов (осциллограф, вольтметр) для регистрации физических сигналов, отражающих функционирование МК;

- лабораторный стенд изучения встраиваемых систем.

В состав лабораторного стенда входят:

- плата микроконтроллера (см. рис. 7-2), включающая микроконтроллер 80C515, внешнюю память программ ROM объемом 32Кбайт, внешнюю память программ/данных RAM объемом 32Кбайт, а также средства связи с инструментальной ЭВМ;

- совокупность объектов управления/датчиков: клавиатура и ЖКИ для изучения дискретного ввода-вывода, потенциометр и интегрирующее RC-звено для изучения аналогового ввода-вывода, микрофон и динамик (опционально);

- совокупность генераторов тестовых воздействий: источник гармонического сигнала "С" с регулируемыми пользователем частотой и амплитудой, а также источник импульсных сигналов "П" с регулируемыми пользователем частотой и скважностью;

- коммутационное поле, позволяющее задавать структуру аппаратного взаимодействия элементов стенда между собой и портами ввода-вывода МК.

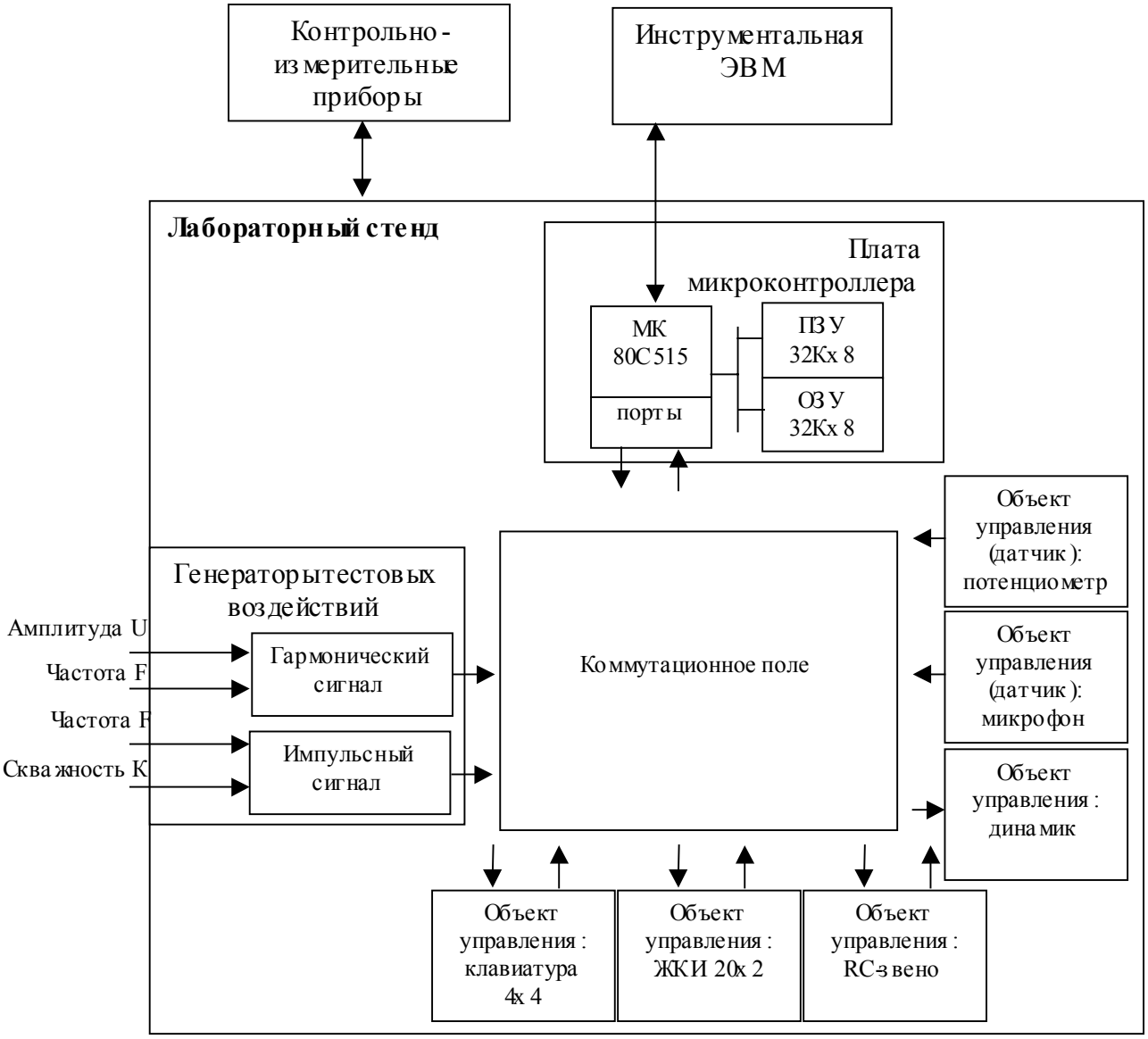


Рис. 7-1. Структура рабочего места.

Внешний вид стенда приведен на рис. 7-3.

Взаимодействие с инструментальной ЭВМ осуществляется через последовательный интерфейс RS-232.

При выполнении лабораторных заданий студент, пользуясь изложенными в разделах 3 и 4 данного пособия приемами и сведениями разделов 1 и 2, разрабатывает и документирует схему соединений портов МК с линиями ввода-вывода объектов стенда, разрабатывает и документирует программное обеспечение, решающее поставленную задачу, документирует и анализирует полученные результаты.

В качестве инструментальной среды разработки применяется система Shell51, описанная в Приложении 4.

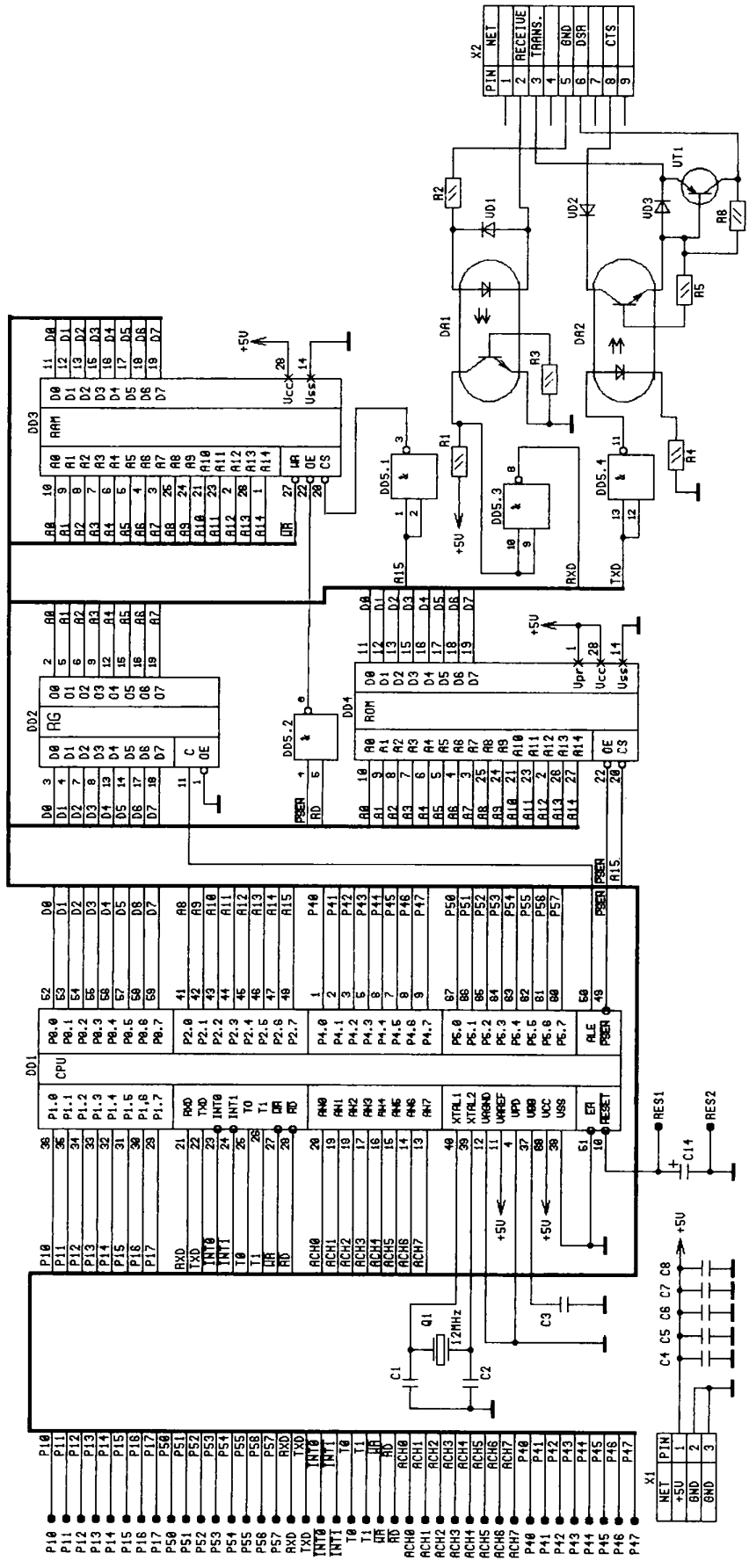


Рис. 7-2. Принципиальная схема платы контроллера.

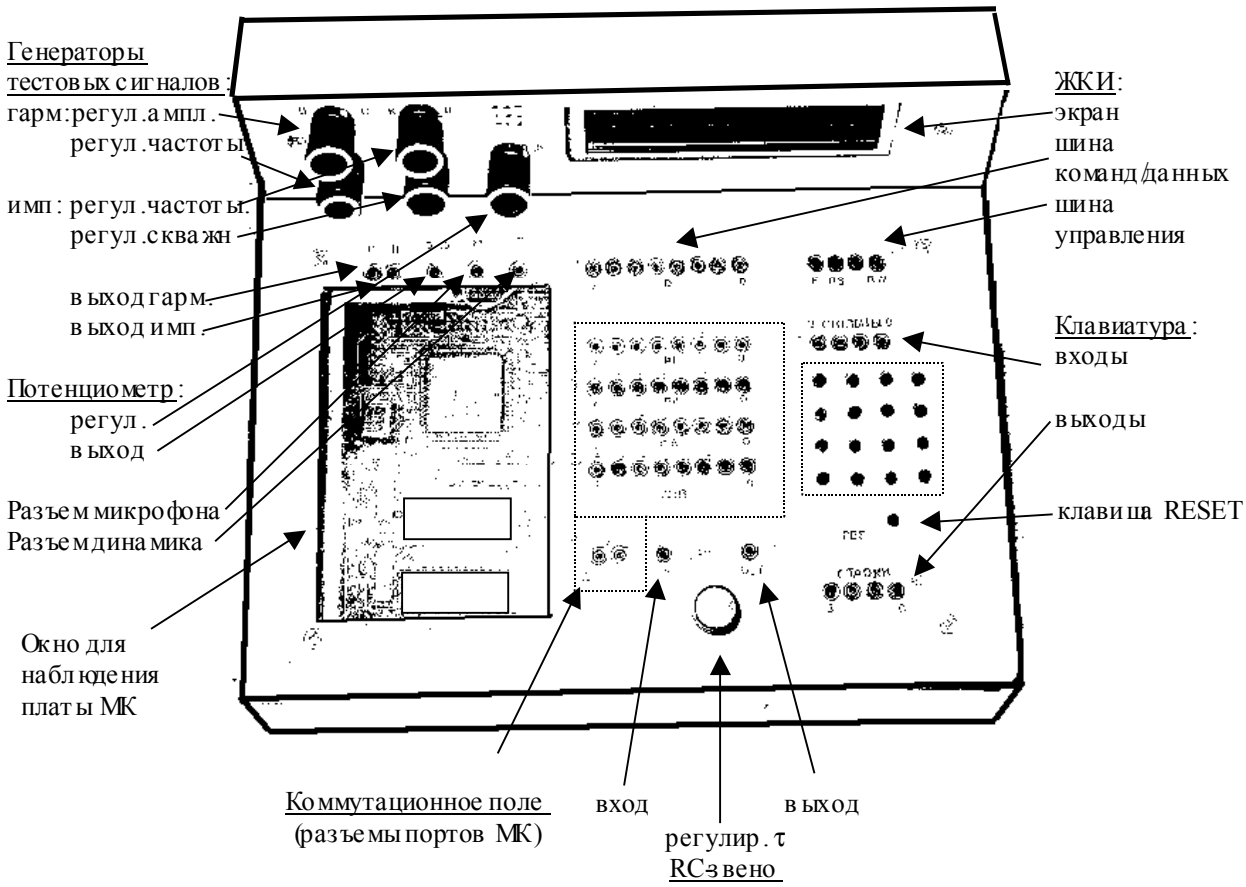


Рис. 7-3. Внешний вид стенда.

Применяемый в стенде ЖКИ Sanyo 20x2 описан в разделе 6.3.1, схема используемой клавиатуры показана на рис. 7-4.

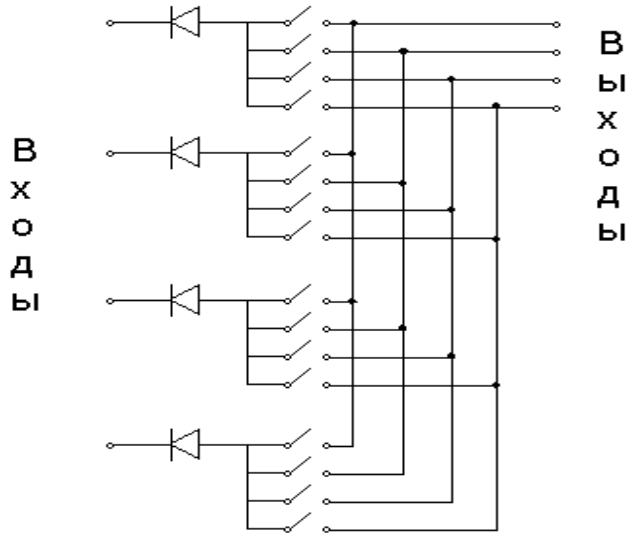


Рис. 7-4. Схема организации клавиатуры лабораторного стенда.

7.2. Программа первого посещения: "Изучение вычислительных возможностей МК".

1. Ознакомиться с программно-аппаратным комплексом поддержки разработки микроконтроллерных систем на базе МК 80С515. Изучить состав и назначение основных устройств и блоков лабораторного стенда.

Пункты 2 - 3 выполнить в программной модели МК (симуляторе) и на реальном МК. Сравнить результаты.

При загрузке, трансляции и чтении результатов выполнения программы использовать правила-рекомендации, приведенные в информационном окне рабочего поля экрана инструментальной ЭВМ.

2. Выполнить полный цикл создания прикладного программного обеспечения на примере тестовой программы proba.asm, которая осуществляет заполнение ячеек заданной области внутренней памяти данных МК.

Выполнить модификацию программы proba.asm, обеспечив изменение диапазона адресации ячеек внутренней памяти при заполнении их требуемыми значениями, например, линейно возрастающими или линейно уменьшающимися значениями.

3. Ознакомиться с системой команд МК семейства MCS51 на примере разработки простейших программ:

- разработать программу вычисления арифметического выражения Y=F(X1,X2,X3) заданного вида (номер варианта задания совпадает с номером рабочего места). Исходные данные - байтовые переменные;

- разработать и выполнить программу вычисления логического выражения Y=F(X1,X2,X3, X4) заданного вида с использованием команд булевого процессора (номер варианта задания совпадает с номером рабочего места). Исходные данные - битовые переменные;

- разработать и выполнить программу формирования кодовой последовательности заданного вида (номер варианта задания совпадает с номером рабочего места). Результаты выполнения разместить в ячейках внешней памяти данных с применением косвенной адресации ячеек памяти через регистр DPTR.

Арифметические выражения.

1. F = X1 * [0,5 * (X2 + X3) - 2*X2 / X3] где X2 <=127, 0,5 * (X2 + X3) > 2X2 / X3
2. F = 1/2 (X1 + X2) * (X3 - X1/X2) где X3 > X1/X2
3. F = 1/4 (X1 + 2*X2) * (X2 - X2/X1) где X2<=127
4. F = (X1*X2 - X2)/X1 * (2*X1 + X2) где X1 <> 0
5. F = 2*X1 * [(X2 - X3) + X2/2*X3] где X1 <=127; X2 > X3
6. F = X1+X2, где X1 и X2 - 4-разрядные 2/10 числа. Команду "DA A" не применять.
7. F = [(X1 - X2)/ (X3 + 1) + 12h] * X3 где X1 > X2; X3 + 1 <= 127
8. F = X1 + [(X2 + X3) + X1*X3]

9. F = X1 * [(X2 / X3) + X2*X3]/2 где X3 <> 0
10. F = [(X1 * X3) + X2 - X3] / X1 где X1 <> 0

Логические выражения:

1. Y = (X1 AND (NOT X2)) XOR (X3 OR X4)
2. Y = ((NOT X1) XOR (X2 AND X3)) OR X4
3. Y = (X1 AND (NOT X2)) XOR (X3 AND X4)
4. Y = ((NOT X1) AND X3) XOR (X2 AND X4)
5. Y = (X1 AND (NOT X2)) XOR (X3 AND X4)
6. Y = (X1 XOR X2) AND (X3 AND X4)
7. Y = (X1 AND (NOT X2)) OR (X3 XOR X4)
8. Y = (X1 OR (NOT X2)) XOR (X3 OR X4)
9. Y = (X1 AND (NOT X3)) XOR (X1 AND X4)
10. Y = ((NOT X1) AND (NOT X2)) OR (X3 XOR X4)

Описания кодовых последовательностей:

1. 0 – 10h – 0
2. 10h – 0 – 10h.
3. 20h – 30h –20h
4. 30h – 20h – 30h
5. 40h – 30h – 0 –10h
6. F8h – FFh – F0h – F8h
7. E0h – D0h – E0h
8. 0 – 10h – 0 – 10h – 0
9. 10h – 20h – 18h – 10h
10. 70h – 60h – 80h.

7.3. Программа второго посещения: "Работа с портами МК".

При выполнении работ второго посещения предварительно необходимо разработать структуру информационных связей контроллера с объектом и реализовать их, соединив разъемы портов МК с разъемами исследуемого объекта.

1. Разработать программу, реализующую опрос клавиатуры и запись в ячейки памяти с адресами 30h . . . 33h кодов состояния клавиш.

2. Модифицировать программу п.1, дополнив ее модулем, формирующим номер нажатой клавиши в памяти данных микроконтроллера.

3. Ознакомиться с организацией и принципом работы модуля жидкокристаллических индикаторов (ЖК-модуля). Описание ЖК-модуля приведено в опции “Справочная информация” оболочки Shell51.

Разработать программу, реализующую вывод на ЖКИ двух 20-байтных строк, содержание которых предопределено в программе.

4. Разработать программу, отображающую на ЖКИ номер нажатой клавиши, используя программы п.-п. 2 и 3, подключив их к разрабатываемой программе директивой INCLUDE.

5. Ознакомиться со структурой АЦП и особенностями программирования режимов его работы. Разработать программу вывода на ЖКИ цифровых кодов аналоговых сигналов параллельно с двух источников: потенциометра и выхода интегратора, подключенного к генератору периодического сигнала. При анализе результатов сопоставить коду сигнала величину напряжения, поступающего на соответствующий вход АЦП, пояснить результаты.

6. Модифицировать программу п.5 для организации отображения цифровых кодов аналоговых сигналов и одновременного представления выбранного пользователем аналогового сигнала на экране инструментальной ЭВМ. При демонстрации результатов выполнения программы использовать опцию “Окна управления” оболочки Shell51. Снятый график приложить к отчету.

7. Разработать программу, генерирующую ШИМ-сигнал переменной скважности, задаваемой с инструментальной ЭВМ, с частотой N*100 Гц, где N - номер рабочего места. Результаты следует наблюдать на осциллографе, показания которого необходимо задокументировать.

7.4. Программа третьего посещения: "Изучение таймеров и системы прерываний".

При выполнении работ третьего посещения предварительно необходимо разработать структуру информационных связей контроллера с объектом и реализовать их, соединив разъемы портов МК с разъемами исследуемого объекта.

1. Изучить работу таймеров-счетчиков микроконтроллера и принципы организации системы прерываний.

2. Разработать программу, генерирующую на выбранном Вами разряде порта МК меандр с частотой N*100 Гц (где N - номер Вашего рабочего места). Работу с линией порта проводить в обработчике прерываний, генерируемых таймером T0. Результаты следует наблюдать на осциллографе, показания которого необходимо задокументировать.

3. Модифицировать программу п.2 для управления генерируемой частотой с клавиатуры. Результаты следует наблюдать на осциллографе, показания которого необходимо задокументировать.

4. Разработать программу, определяющую интервал времени между прерываниями INT0 и INT1, генерируемых с клавиатуры.

5. Реализовать на микроконтроллере счетчик времени с отображением на ЖКИ минут и секунд работы. В качестве опорного программного генератора использовать прерывания таймера.

6. Организовать взаимодействие оболочки с микроконтроллером по прерываниям от приемопередатчика. При этом в качестве прерываемого процесса использовать программу п. 5. С использованием опции "Окна управления" следует наблюдать за изменением значений опорного программного генератора. Снятый график приложить к отчету.

7.5. Программа четвертого посещения: "Организация и исследование межпроцессорного обмена".

1. Изучить текст программы send_rec.asm.

2. Исследовать ее работу, применяя вкладку “Терминал”.

3. Создать программу, реализующую вывод на экран ЖКИ текста, передаваемого с инструментальной ЭВМ.

4. Реализовать на МК сервисную программу статистической обработки данных: с инструментальной ЭВМ в МК передается последовательность чисел с завершающим символом ‘.’, в ответ на которую МК в зависимости от запрашивающего кода (протокол сформировать самостоятельно) передает в ЭВМ: максимальное из чисел, минимальное из чисел, среднее арифметическое и дублирует информацию на ЖКИ. Количество чисел и формат их представления должен быть свободным.

7.6. Варианты индивидуальных заданий.

1. Многофункциональный генератор.

Формирует в зависимости от комбинации нажатых клавиш 6 видов периодических сигналов: гармонический, прямоугольный, треугольный, пилообразный, колоколообразный, трапецеидальный; для любого сигнала возможно формирование нескольких (не менее 5) значений периода. Выход генератора – ШИМ-сигнал, модулирующий заданный периодический сигнал.

2. Модель динамического объекта первого порядка:

- инерционно-интегрирующее звено;
- инерционно-дифференцирующее звено;

Требуется сформировать и программно реализовать разностные уравнения, реализующие указанные звенья. Входные сигналы поступают с инструментальной ЭВМ; там же отображаются результаты. Тип звена и его параметры задавать с клавиатуры стенда.

3. Статистическая обработка сигналов.

Программа должна реализовать накопление входного сигнала с аналогового входа АЦП, провести вычисления, необходимые для построения гистограммы, и предоставить инструментальной ЭВМ возможность приема полученной информации. Сведения о максимуме, минимуме и среднем арифметическом значении сигнала за время наблюдения должны отображаться на ЖКИ. Команда запуска очередного измерения должна формироваться с клавиатурного пульта МК.

4. Сжатие-восстановление информации.

Программа должна реализовать накопление входного сигнала с аналогового входа АЦП,

провести вычисления, необходимые для компрессии полученного набора данных, и предоставить инструментальной ЭВМ возможность приема информации до сжатия, после сжатия и после восстановления. Сведения о работе программы и качестве сжатия должны отображаться на ЖКИ. Команда запуска очередного цикла работы должна формироваться с клавиатуры МК.

5. Модель системы автоматического управления.

Программа должна реализовать:

- разомкнутую неустойчивую систему;

- звено, “зажимающее” неустойчивую систему в наперед заданной области, применяя алгоритмы релейного, П-, ПИ-регулирования.

Режимы работы программы задаются с клавиатурного блока МК, сведения о работе отображаются на ЖКИ. Программа должна допускать интерактивное наблюдение работы САУ с инструментальной ЭВМ.

6. Музыкальный синтезатор.

Программа должна реализовать следующие режимы работы:

- ввод мелодии с клавиатуры стенда и ее отработку динамиком платы МК;
- переключение октав;
- отображение названия ноты на ЖКИ;
- режим записи мелодии в память МК;
- режим воспроизведения ранее записанной мелодии;
- режим доступа инструментальной ЭВМ к мелодии.

7. Калькулятор.

Программа должна реализовать арифметическое вычислительное устройство с вводом цифровой информации с клавиатурного блока стенда, вывода исходных данных и результатов вычислений на ЖКИ.

Формат представления чисел – знаковый, с фиксированной точкой. Диапазон: -32768..+32767.

Дополнительные функции: занесение в память, округление, логические функции, взаимное преобразование кодов в 2, 8, 10, 16-ричной системах счисления.

8. Частотомер.

Программа должна определять частоту периодического сигнала, подаваемого на вход АЦП. Предусмотреть возможность измерения высокочастотного и низкочастотного сигналов наиболее точными методами. Режимы работы и результат вычислений отображается на ЖКИ.

9. Электронная записная книга.

Программа должна поддерживать следующие режимы:

- Часы-календарь с возможностью установки даты и времени (с отображением на ЖКИ);

- звуковой оповещатель о наступлении указанного пользователем момента времени с

выводом на ЖКИ заданной пользователем однострочной записи-подсказки (на русском языке, для каждого момента времени своя);

- телефонная база данных формата: “телефон – текстовая строка”. Режимы работы базы данных - просмотр, поиск, ввод новых, удаление.

10. Электронный экзаменатор.

Программа предлагает пользователю ответить на ряд вопросов с вариантами ответа, отображаемыми на ЖКИ. Проводится подсчет времени каждого ответа, количество правильных и неправильных ответов. Результаты теста отображаются на ЖКИ. По окончании теста возможен просмотр данных с инструментальной ЭВМ.

11. Игровой тренажер внимания.

По экрану ЖКИ слева направо перемещаются символы, соответствующие определенным клавишам. Задача игрока – отреагировать на очередной символ нажатием соответствующей ему клавиши. По ходу игры темп перемещения ускоряется. Предусмотреть вывод статистики и музыкального оформления тренажера.

7.7. Отчетность по лабораторному практикуму.

В ходе работы над предлагаемым лабораторным практикумом студенты документируют свою работу.

По окончании лабораторной работы студент предъявляет следующие документы:

- схемы соединений аппаратуры;
- схемы разработанных программ и их тексты;
- результаты исполнения программ (графики, полученные средствами Shell51, осциллограммы и др.).

В отчет должны входить следующие разделы:

- Титульный лист;
- Содержание;
- Введение;
- Глава 1: "Изучение вычислительных возможностей МК":
    - цель работы;
    - программа работы;
    - теоретические сведения;
    - задание №1:
        - название;
        - анализ задания;
        - схема соединений;
        - схема программы;
        - текст программы;
        - испытания программы;
        - обсуждение результатов;
    - задание №2:

    (далее все задания главы оформляются по аналогии с заданием 1);
    - …
    - выводы по главе;
- Глава 2: "Работа с портами МК":

(далее все главы оформляются по аналогии с главой 1);

- …
- общее заключение по отчету.

ЛИТЕРАТУРА.

1. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах.- М.: Энергоатомиздат, 1990.- 224 с.

2. Боборыкин А.В., Липовецкий Г.П., Литвинский Г.В. и др. Однокристальные микроЭВМ.- М.: МИКАП, 1994.- 400 с.

3. Бродин В.Б., Шагурин И.И. Микроконтроллеры. Архитектура, программирование, интерфейс.- М.: Издательство ЭКОМ, 1999.- 400 с.

4. Хвощ С.Т., Варлинский Н.Н., Попов Е.А. Микропроцессоры и микроЭВМ в системах автоматического управления: Справочник.- Л.: Машиностроение, 1987.- 640 с.

5. Мясников В.А., Игнатьев М.Б., Кочкин А.А. и др. Микропроцессоры: системы программирования и отладки.- М.: Энергоатомиздат, 1985.- 272 с.

6. Уильямс Г.Б. Отладка микропроцессорных систем: Пер. с англ.- М.: Энергоатомиздат, 1988.- 253 с.

7. Архангельский Б.В., Черняховский В.В. Поиск устойчивых ошибок в программах.- М.: Радио и связь, 1989.- 240 с.

8. Щербаков Н.С. Достоверность работы цифровых устройств.- М.: Машиностроение, 1989.- 224 с.

9. Граф Ш., Гессель М. Схемы поиска неисправностей: Пер. с нем.- М.: Энергоатомиздат, 1989.- 144 с.

10. Клисторин И.Ф., Гремальский А.А. Функциональный контроль микропроцессорных устройств.- Кишинев: Штиинца, 1990.- 91 с.

11. Воробьев Н.И. Проектирование электронных устройств.- М.: ВШ, 1989.- 223 с.

12. Микропроцессорные системы. Учебное пособие для ВУЗов. Под общей редакцией Д.В. Пузанкова.- СПб.: Политехника, 2002.- 935 с.

13. Конструкторско-технологическое проектирование электронной аппаратуры. Учебник для ВУЗов. Под ред. проф. В.А. Шахнова.- М.: Изд-во МГТУ им. Баумана, 2002.- 528 с.

14. Николайчук О.И. x51-Совместимые микроконтроллеры фирмы Cygnal.- М.: ООО "ИД СКИМЕН", 2002.- 472 с.

15. Гладштейн М.А. Микроконтроллеры семейства Z86 фирмы ZILOG. Руководство программиста.- М.: ДОДЭКА, 1999.- 96 с.

16. Ульрих В.А. Микроконтроллеры PIC16C7X.- СПб.: Наука и техника, 2000.- 253 с.

17. Предко М. Справочник по PIC-микроконтроллерам. Пер. с англ.- М.: ДОДЭКА, 2002.- 512 с.

18. Андрэ Ф. Микроконтроллеры семейства SX фирмы "Scenix". Пер. с фр.- М.: ДОДЭКА, 2002.- 272 с.

19. Однокристальные микроконтроллеры PIC12C5x, PIC12C6x, PIC16x8x, PIC14000, M16C/61/62. Пер. с англ. и ред. Прокопенко Б.Я.- М.: ДОДЭКА, 2001.- 336 с.

20. Ремизевич Т.В. Микроконтроллеры для встраиваемых приложений: от общих подходов - к семействам HC05 и HC08 фирмы Motorola.- М.: ДОДЭКА, 2000.- 272 с.

21. Шагурин И.И. Микропроцессоры и микроконтроллеры фирмы Motorola.- М.: Радио и связь, 1998.- 560 с.

22. Куприянов М.С., Матюшкин Б.Д. Цифровая обработка сигналов: процессоры, алгоритмы, средства проектирования.- СПб.: Политехника, 1998.- 592 с.

23. Васильев А.Е. Микропроцессорные системы. Разработка устройств на микроконтроллерах. Учебное пособие.- СПб.: Изд-во СПбГТУ, 2001.- 132 с.

ПРИЛОЖЕНИЕ 1.

Сравнительные характеристики некоторых моделей микроконтроллеров.

В таблице приводятся некоторые, наиболее характерные представители микроконтроллеров ведущих фирм-производителей МК.

Более подробную информацию по представленным микроконтроллерам, а также обзор выпускаемых МК можно найти на сетевых ресурсах фирм:

Infineon: http://www.infineon.com

Cygnal: http://www.cygnal.com

Microchip: http://www.microchip.com

Atmel: http://www.atmel.com

Fujitsu: http://www.fujitsu.com

Scenix: http://www.ubicom.com

Motorola: http://www.motorola.com

Intel: http://www.intel.com

Winbond: http://www.winbond.com

Модель	Fosc, МГц	Пам. прогр.	Пам. данн.	Линии вв.-выв.	Таймеры общего. назнач., кол-во / разрядн.	АЦП, кан./ разр	ЦАП кан./ разр	Послед. интерфейсы	Корпус
4-разрядные микроконтроллеры									
Winbond W741E205	4	2Kx16	128x4	21	2 / 8	-	-	SSI	SOP28
8-разрядные микроконтроллеры									
Cygnal C8051F020	25	64Kx8	4352	64	5 / 16	8/ 12	1/12	SPI, I2C, 2xUSART	TQ100
Atmel ATmega128	16	128Kx8	4Kx8	53	2 / 8, 2 / 16	8/10	-	SPI	TQFP64
Infineon 80C515-N	12	8Kx8	256	48	3 / 16	8 / 8	-	USART	PLCC68
Scenix SX52BD100	100	4Kx12	256	40	1 / 8 2/ 16	ана-лог. комп	-	-	PQFP52

Модель	Fosc, МГц	Пам. прогр.	Пам. данн.	Линии вв.-выв.	Таймеры общего. назнач., кол-во / разрядн.	АЦП, кан./ разр	ЦАП кан./ разр	Послед. интерфейсы	Корпус
Microchip PIC16C433	10	2048x14	128x8	6	1 / 8	4 / 8	-	LIN	DIP18
16-разрядные микроконтроллеры									
Infineon C167CS-32FM	25	256Kx8	11Kx8	111	5 / 16	24/ 10	-	USART, SSC, 2xCAN	MQFP-144
Fujitsu MB90F574	16	256Kx8	10Kx8	97	1 / 16	8 / 8	2 / 8	I2C, 2xUART, 2xSSI	QFP-120
Intel 87C196NT	20	32Kx8	1000 x8	56	2 / 16	4/ 10	-	USART, SSI	PLCC68
32-разрядные микроконтроллеры									
Motorola MCF5282	66	512Kx8	64Kx8	150	4 / 32 1 / 16	8/ 10	-	CAN Ethernet I2C 3xUSART SPI	MAP-BGA256
Fujitsu MB91F360GA	64	512Kx8	32Kx8	114	2 / 16	16/ 10	2/ 10	3 USART 4 CAN 2 SSI I2C	PGA401
Infineon TriCore TC10GP	66	8Кx8 ОЗУ	32Кx8	24	3 / 32	-	-	2 USART SSC	PQFP-208

Группа команд пересылок

Мнемокод	КОП		Б	Ц	Функция
MOV A, Rn	1110	1rrr	1	1	(A) := (Rn)
MOV A, direct	1110	0101	2	1	(A) := (direct)
MOV A, @Ri	1110	011i	1	1	(A) := ((Ri))
MOV A, #data	1110	0100	2	1	(A) := #data
MOV Rn, A	1111	1rrr	1	1	(Rn) := (A)
MOV Rn, direct	1010	1rrr	2	2	(Rn) := (direct)
MOV Rn, #data	0111	1rrr	2	1	(Rn) := #data
MOV direct, A	1111	0101	2	1	(direct) := (A)
MOV direct, Rn	1000	1rrr	2	2	(direct) := (Rn)
MOV direct, direct	1000	0101	3	2	(direct) := (direct)
MOV direct, @Ri	1000	011i	2	2	(direct) := ((Ri))
MOV direct, #data	0111	0101	3	2	(direct) := #data
MOV @Ri, A	1111	011i	1	1	((Ri)) := (A)
MOV @Ri, direct	1010	011i	2	1	((Ri)) := (direct)
MOV @Ri, #data	0111	011i	2	1	((Ri)) := #data
MOV DPTR, #data16	1001	0000	3	2	((DPTR)) := #data16
MOVC A, @A+DPTR	1001	0011	1	2	(A) := ((A)+(DPTR)) Чтение памяти программ
MOVC A, @A+PC	1000	0011	1	2	(A) := ((A)+(PC)) Чтение памяти программ
MOVX A, @Ri	1110	001i	1	2	(A) := ((Ri)) Чтение памяти данных
MOVX @Ri, A	1111	001i	1	2	((Ri)) := (A) Запись памяти данных
MOVX A, @DPTR	1110	0000	1	2	(A) := ((DPTR)) Чтение памяти данных
MOVX @DPTR, A	1111	0000	1	2	((DPTR)) := (A) Запись памяти данных
PUSH direct	1100	0000	2	2	(SP) := (SP)+1 ((SP)) := (direct) запись в стек
POP direct	1101	0000	2	2	(direct) := ((SP)) (SP) := (SP)-1 чтение из стека
XCH A, Rn	1100	1rrr	1	1	(A) <-> (Rn)
XCH A, direct	1100	0101	2	1	(A) <-> (direct)
XCH A, @Ri	1100	011i	1	1	(A) <-> ((Ri))
XCHD A, @Ri	1101	011i	1	1	(A_{3-0}) <-> $((Ri)_{3-0})$
SWAP A	1100	0100	1	1	Обмен тетрад аккумулятора

Мнемокод	КОП		Б	Ц	Функция
MOV C, bit	1010	0010	2	1	(C) := bit
MOV bit, C	1001	0010	2	2	(bit) := (C)
CLR C	1100	0011	1	1	(C) := 0
CLR bit	1100	0010	2	1	(bit) := 0
SETB C	1101	0011	1	1	(C) := 1
SETB bit	1101	0010	2	1	(bit) := 1

Группа команд передачи управления

Мнемокод	КОП		Б	Ц	Функции
LJMP addr16	0000	0010	3	2	(PC):= $addr_{15-0}$
AJMP addr11	a_{10-8}0	0001	2	2	(PC):=(PC) + 2 $(PC)_{10-0}$:=$addr_{10-0}$
SJMP rel	1000	0000	2	2	(PC):=(PC) + 2 (PC):=(PC) + rel
JMP @A + DPTR	0111	0011	1	2	(PC):=(A) + (DPTR)
LCALL addr16	0001	0010	3	2	(PC):=(PC) + 2 INC(SP) ((SP)):=(PC_{7-0}) INC(SP) ((SP)):=(PC_{15-8}) (PC):=$addr_{15-0}$
ACALL addr11	a_{10-8}1	0001	2	2	(PC):=(PC) + 2 INC(SP) ((SP)):=(PC_{7-0}) INC(SP) ((SP)):=(PC_{15-8}) (PC_{10-0}):=$addr_{10-0}$
RET	0010	0010	1	2	(PC_{15-8}):=((SP)) DEC(SP) (PC_{7-0}):=((SP)) DEC(SP)
RETI	0011	0010	1	2	(PC_{15-8}):=((SP)) DEC(SP) (PC_{7-0}):=((SP)) DEC(SP)
JZ rel	0110	0000	2	2	(PC):=(PC) + 2 if (A) = 0 then (PC):=(PC) + rel
JNZ rel	0111	0000	2	2	(PC):=(PC) + 2 if (A)<>0 then (PC):=(PC) + rel
JC rel	0100	0000	2	2	(PC):=(PC) + 2 if (C) = 1 then (PC):=(PC) + rel
JNC rel	0101	0000	2	2	(PC):=(PC) + 2 if (C) = 0 then (PC):=(PC) + rel
DJNZ Rn, rel	1101	1rrr	2	2	(PC):=(PC) + 2 (Rn):=(Rn) - 1 if (Rn) <> 0 then (PC):=(PC) + rel
DJNZ direct, rel	1101	0101	2	2	(PC):=(PC) + 2 (direct):=(direct) - 1 if (direct) <> 0 then (PC):=(PC) + rel

CJNE A, direct, rel	1011	0101	3	2	(PC):=(PC) + 3
					if (A) <> (direct) then (PC):=(PC) + rel
					if (A) < (direct) then (C):=1 else (C):=0

Мнемокод	КОП		Б	Ц	Функции

CJNE A, #data, rel	1011	0100	3	2	(PC):=(PC) + 3
					if (A) <> #data then (PC):=(PC) + rel
					if (A) < #data then (C):=1 else (C):=0

CJNE Rn, #data, rel	1011	1rrr	3	2	(PC):=(PC) + 3
					if (Rn)<> #data then (PC):=(PC) + rel
					if (Rn) < #data then (C):=1 else (C):=0

CJNE @Ri, #data, rel	1011	011i	3	2	(PC):=(PC) + 3
					if ((Ri)) <> #data then (PC):=(PC) + rel
					if ((Ri)) < #data then (C):=1 else (C):=0

JB bit, rel	0010	0000	3	2	(PC) := (PC) + 3 if (bit) = 1 then (PC) := (PC) + rel
JBC bit, rel	0001	0000	3	2	(PC) := (PC) + 3 if (bit) = 1 then (bit) := 0, (PC) := (PC) + rel
JNB bit, rel	0011	0000	3	2	(PC) := (PC) + 3 if (bit) = 0 then (PC) := (PC) + rel
NOP	0000	0000	1	1	(PC) := (PC) + 1

Группа команд арифметических и логических операций

Мнемокод	КОП		Б	Ц	Функции
ADD A, Rn	0010	1rrr	1	1	(A) := (A) + (Rn)
ADD A, direct	0010	0101	2	1	(A) := (A) + (direct)
ADD A, @Ri	0010	011i	1	1	(A) := (A) + ((Ri))
ADD A, #data	0010	0100	2	1	(A) := (A) + #data
ADDC A, Rn	0011	1rrr	1	1	(A) := (A) + (C) + (Rn)
ADDC A, direct	0011	0101	2	1	(A) := (A) + (C) + (direct)
ADDC A, @Ri	0011	011i	1	1	(A) := (A) + (C) + ((Ri))
ADDC A, #data	0011	0100	2	1	(A) := (A) + (C) + #data
SUBB A, Rn	0011	1rrr	1	1	(A) := (A) - (C) - (Rn)
SUBB A, direct	0011	0101	2	1	(A) := (A) - (C) - (direct)
SUBB A, @Ri	0011	011i	1	1	(A) := (A) - (C) - ((Ri))
SUBB A, #data	0011	0100	2	1	(A) := (A) - (C) - #data

Мнемокод	КОП		Б	Ц	Функции
DA A	1101	0100	1	1	Десятичная коррекция содержимого аккумулятора
INC A	0000	0100	1	1	(A) := (A) + 1
INC Rn	0000	1rrr	1	1	(Rn) := (Rn) + 1
INC direct	0000	0101	2	1	(direct) := (direct) + 1
INC @Ri	0000	011i	1	1	((Ri)) := ((Ri)) + 1
INC DPTR	1010	0011	1	2	(DPTR) := (DPTR) + 1
DEC A	0001	0100	1	1	(A) := (A) - 1
DEC Rn	0001	1rrr	1	1	(Rn) := (Rn) - 1
DEC direct	0001	0101	2	1	(direct) := (direct) - 1
DEC @Ri	0001	011i	1	1	((Ri)) := ((Ri)) - 1
MUL AB	1010	0100	1	4	$(B)_{15\text{-}8}$, $(A)_{7\text{-}0}$:= (A)*(B)
DIV AB	1000	0100	1	4	$(A)_{цел}$, $(B)_{ост}$:= (A)/(B)
ANL A, Rn	0101	1rrr	1	1	(A) := (A) & (Rn)
ANL A, direct	0101	0101	2	1	(A) := (A) & (direct)
ANL A, #data	0101	0100	2	1	(A) := (A) & #data
ANL A, @Ri	0101	011i	1	1	(A) := (A) & ((Ri))
ANL direct, A	0101	0010	2	1	(direct) := (direct) & (A)
ANL direct, #data	0101	0011	3	3	(direct) := (direct) & #data
ORL A, Rn	0100	1rrr	1	1	(A) := (A) V (Rn)
ORL A, direct	0100	0101	2	1	(A) := (A) V (direct)
ORL A, @Ri	0100	011i	1	1	(A) := (A) V ((Ri))
ORL A, #data	0100	0100	2	1	(A) := (A) V #data
ORL direct, A	0100	0010	2	1	(direct) := (direct) V (A)
ORL direct, #data	0100	0011	3	3	(direct) := (direct) V #data
XRL A, Rn	0110	1rrr	1	1	(A) := (A) XOR (Rn)
XRL A, direct	0110	0101	2	1	(A) := (A) XOR (direct)
XRL A, @Ri	0110	011i	1	1	(A) := (A) XOR ((Ri))
XRL A, #data	0110	0100	2	1	(A) := (A) XOR #data
XRL direct, A	0110	0010	2	1	(direct) := (direct) XOR (A)

Мнемокод	КОП		Б	Ц	Функции
XRL direct, #data	0110	0011	3	3	(direct) := (direct) XOR #data
CLR A	1110	0100	1	1	Обнуление аккумулятора (A) := 0
CPL A	1111	0100	1	1	Инверсия аккумулятора
RL A	0010	0011	1	1	Сдвиг аккумулятора влево циклический
RLC A	0011	0011	1	1	Сдвиг аккумулятора влево через перенос
RR A	0000	0011	1	1	Сдвиг аккумулятора вправо циклический
RRC A	0001	0011	1	1	Сдвиг аккумулятора вправо через перенос
CPL C	1011	0011	1	1	Инверсия бита переноса
CPL bit	1011	0010	2	1	Инверсия прямоадресуемого бита
ANL C, bit	1000	0010	2	2	C:=C&bit
ANL C, /bit	1011	0000	2	2	C:=C&/bit
ORL C, bit	0111	0010	2	2	C:=C or bit
ORL C, /bit	1010	0000	2	2	C:=C or /bit

Список команд, воздействующих на флаги регистра PSW

Команда	Флаги			Команда	Флаги		
	C	OV	AC		C	OV	AC
ADD	+	+	+	ANL C, bit	+	-	-
ADDC	+	+	+	ANL C, /bit	+	-	-
SUBB	+	+	+	ORL C, bit	+	-	-
MUL	0	+	-	ORL C, bit	+	-	-
DIV	0	+	-	MOV C, bit	+	-	-
DA	+	-	-	CLR C	0	-	-
RRC	+	-	-	CPL C	+	-	-
RLC	+	-	-	CJNE	+	-	-
SETB C	1	-	-	POP PSW	+	+	+

ПРИЛОЖЕНИЕ 3.

ЧАСТО УПОТРЕБЛЯЕМЫЕ ДИРЕКТИВЫ ЯЗЫКА АССЕМБЛЕРА X8051.

Директивы ассемблера предназначены для управления процессами ассемблирования и компоновки, в ходе которых происходит перевод исходного текста программы в формат, пригодный для исполнения на микроконтроллере.

Директивы позволяют задавать используемые пользователем переменные, области памяти, организовать секционную структуру исполняемых программ, выполнять вычисления при трансляции программы и др.

Директивы управления адресами и файлами программ:

ORG VALUE

Устанавливает адрес программы. По умолчанию адрес устанавливается в 0000.

Пример: ORG 8100h

END VALUE

Отмечает конец программы или подключенного файла. Выражение, следующее за END, дополнительное и, если существует, то указывает стартовый адрес программы. Этот адрес заносится в выходной файл, если в выходном формате существует тип записи стартового адреса.

INCLUDE <имя файла>

Директива указывает ассемблеру на необходимость трансляции текста, расположенного в файле <имя файла>, начиная с адреса, следующего за адресом последней команды, расположенной в текущем файле.

Пример: INCLUDE asms\names.asm

Директивы определения символических имен:

LABEL: EQU VALUE

Присваивает имени LABEL значение VALUE. VALUE может быть другим символом или разрешенным математическим выражением.

Пример: X1: equ 30h

LABEL: VAR VALUE

Присваивает имени LABEL значение VALUE, но значение может изменяться по тексту программы. Имя, определенное как VAR, не следует переопределять директивой EQU.

LLCHAR CHARACTER

По умолчанию символ $ является символом для обозначения локальной метки. Эта директива изменяет символ-определитель локальной метки.

GLOBAL LABEL

Определив метку как глобальную, можно ссылаться на нее из других программ. Несколько меток

можно определить как удаленные, записав их через запятую.
GLOBAL SYM1 ;Объявляет символ SYM1 доступным для других программ.

EXTERNAL LABEL
Определяет метку как заданную в другой программе.

Директивы резервирования памяти:

LABEL: ASCII STRING
Запоминает STRING в памяти, но не включая оба символа окончания строки, а также вертикальную черту ("|", шестнадцатеричное 7C).
ASCII Hello

LABEL: DB VALUE
Помещает величины в последовательные ячейки памяти. Выражение DB позволяет смешивать операнды различного типа, разделенные запятыми. Строки символов следует заключать в апострофы. Если строка содержит апострофы, то следует поставить два апострофа подряд. Если после директивы не следует выражение, один байт резервируется и обнуляется. Метка не является обязательной.
.DB ;Отводит 1 нулевой байт
.DB 10 ;Отводит байт = 10
.DB 1,2,3 ;Отводит 3 байта=равные 1,2 и 3
.DB SYMBOL-10 ;Ищет в таблице символов метку SYMBOL, вычитает из нее 10,
;и запоминает результат
.DB 'HELLO' ;запоминает эквивалент символов ASCII слова HELLO

LABEL: DW VALUE
Помещает 16-битные величины в память. Несколько слов могут быть заданы посредством записи нескольких выражений через запятую. Если выражение не дано, резервируется и обнуляется 1 слово. Метка не является обязательной.

LABEL: LONG VALUE
Помещает 32-битные величины в память. Несколько слов могут быть заданы посредством записи нескольких выражений через запятую. Если выражение не дано, резервируются и обнуляются 2 слова. Метка не является обязательной.

LABEL: BLKB SIZE,VALUE
Резервирует число байтов, определяемое SIZE. Если поле VALUE присутствует, величина VALUE запоминается в каждом байте. В противном случае резервируемые байты обнуляются. Метка не является обязательной.
BLKB 20 ;Запоминает 20 нулевых байт
BLKB 20,0 ;Запоминает 20 нулевых байт
BLKB 20,FFH ;Запоминает 20 байт значения 0FFH

LABEL: BLKW SIZE,VALUE

Резервируется число 16-битных слов, определяемое SIZE. Если поле VALUE присутствует, величина VALUE запоминается в каждом слове. В противном случае, резервируемые слова обнуляются. Метка не является обязательной.

BLKW 20 ;Запоминает 20 нулевых слов

BLKW 20,0 ;Запоминает 20 нулевых слов

BLKW 20,FFFFH ;Запоминает 20 слов значения 0FFFFH

LABEL: BLKL SIZE,VALUE

Резервируется некоторое число 32-битных слов, определяемое SIZE. Если поле VALUE присутствует, величина VALUE запоминается в каждом слове. В противном случае, резервируемые слова обнуляются. Метка не является обязательной.

BLKL 20 ;Запоминает 20 нулевых двойных слов

BLKL 20,0 ;Запоминает 20 нулевых двойных слов

BLKL 20,FFFFH ;Запоминает 20 двойных слов значения 0FFFFH

Директивы работы с макросами:

LABEL: MACRO ARGS

Задает начало макроопределения.

ENDM

Задает конец макроопределения.

Пример реализации макроса:

```
        org 8100h
x1:     equ 40h
        sum3 #1,#2,#3
        sum3 50h,55h, 60h
        sum3 x1,#3,35h
        ret
sum3:   macro arg1,arg2,arg3
        mov a,arg1
        add a,arg2
        add a,arg3
        endm
```

В данном случае в памяти программ микроконтроллера, начиная с адреса 8100h будут сформированы три группы кодов, каждая из которых соответствует очередному макровызову.

```
        org 8100h
        mov a,#1
```

```
add a,#2
add a,#3
mov a,50h
add a,55h
add a,60h
mov a,40h
add a,#3
add a,35h
ret
```

ПРИЛОЖЕНИЕ 4. ОПИСАНИЕ СРЕДЫ ПРОЕКТИРОВАНИЯ SHELL51.

Система предназначена для проектирования программного обеспечения микропроцессорных систем управления на базе программируемых контроллеров (в том числе семейства МК-51). Система ориентирована на применение в составе комплекса, включающего инструментальную ЭВМ и микроконтроллер, подключенный к ЭВМ через последовательный интерфейс. Одним из направлений применения среды проектирования Shell51 является ее использование в учебном процессе.

После запуска пакета (двойным щелчком левой кнопки манипулятора "мышь" на пиктограмме с названием "Система Shell51") на экране отобразится окно программы (рис. П-1).

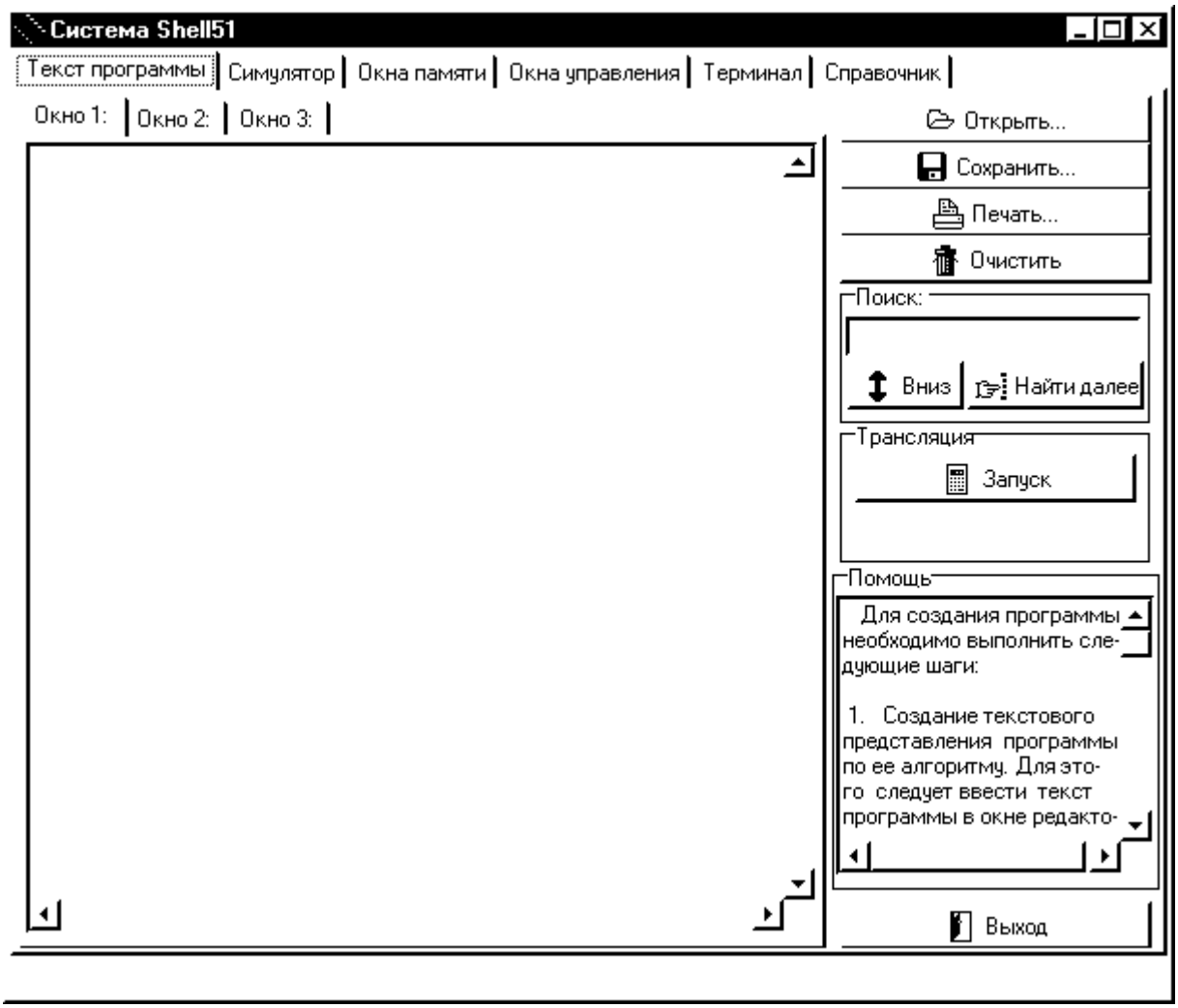


Рис. П-1. Окно среды Shell51.

Окно представляет собой набор вкладок-опций, каждая из которых ориентирована на реализацию одного из типовых действий программиста при проектировании программного обеспечения микроконтроллерных систем.

Ниже приведено более подробное описание опций.

ВКЛАДКА “ТЕКСТ ПРОГРАММЫ” (рис. П-1) - позволяет работать с исходным представлением программы в виде текста на языке ассемблера. Вкладка содержит три независимых окна текста (слева), поле справки по использованию данной опции (справа внизу), набор кнопок-панелей, реализующих функции данной опции, а также кнопку-панель "Выход". Активизация функции, назначенной кнопке-панели, производится путем нажатия левой кнопки манипулятора "мышь" над изображением этой панели.

Кнопка-панель "Открыть" активизирует диалог выбора необходимого пользователю файла текста программы, сохраненного на дисковых накопителях инструментальной ЭВМ, и копированию его содержимого в активное окно текста.

Кнопка-панель "Сохранить" активизирует диалог записи содержимого активного окна текста на диск инструментальной ЭВМ, предоставляя пользователю возможность указания имени файла сохраняемой программы.

Кнопка-панель "Печать" активизирует диалог назначения параметров печати информации, находящейся в поле активного окна текста программы, по окончании назначения которых производится вывод текста программы на печатающее устройство.

Кнопка-панель "Очистить" позволяет удалить из активного окна текста находящуюся в нем информацию.

Кнопка-панель "Запуск" в поле "Трансляция" активизирует выполнение процесса перевода исходного текста программы, находящегося в активном поле текста, в представление, пригодное для загрузки в микроконтроллер. Результат прохождения данного этапа отображается в виде двух диагностических сообщений: "Ок" - при отсутствии ошибок в программе и предупреждений со стороны транслятора и компоновщика, "Внимание" - в противном случае. Детальная информация о результатах трансляции и компоновки расположена на вкладке "Листинги", речь о которой пойдет ниже. Перед запуском трансляции текст разрабатываемой программы следует сохранить.

Пользовательский ввод в окно текста программы производится посредством клавиатуры инструментальной ЭВМ, при этом допускается использование следующих типовых сочетаний клавиш: Shift+→ (а также иные клавиши перемещения курсора) - выделение текста; Ctrl+C - копирование текста в буфер обмена; Ctrl+V - вставка текста из буфера обмена; Ctrl+X - вырезание текста в буфер обмена.

Для облегчения работы с текстами большого объема (в том числе для поиска строк, указанных транслятором как содержащие ошибки) используется опция "Поиск".

Текстовую информацию, подлежащую поиску, следует поместить в поле ввода, с помощью кнопки-панели "Вверх" ("Вниз") задать направление поиска (от начала документа к концу или наоборот). По нажатию кнопки-панели "Найти далее" в активном окне текста будет выделена строка, содержащая указанный текст. Каждое следующее нажатие этой кнопки-панели будет продолжать поиск в выбранном направлении (рис. П-2).

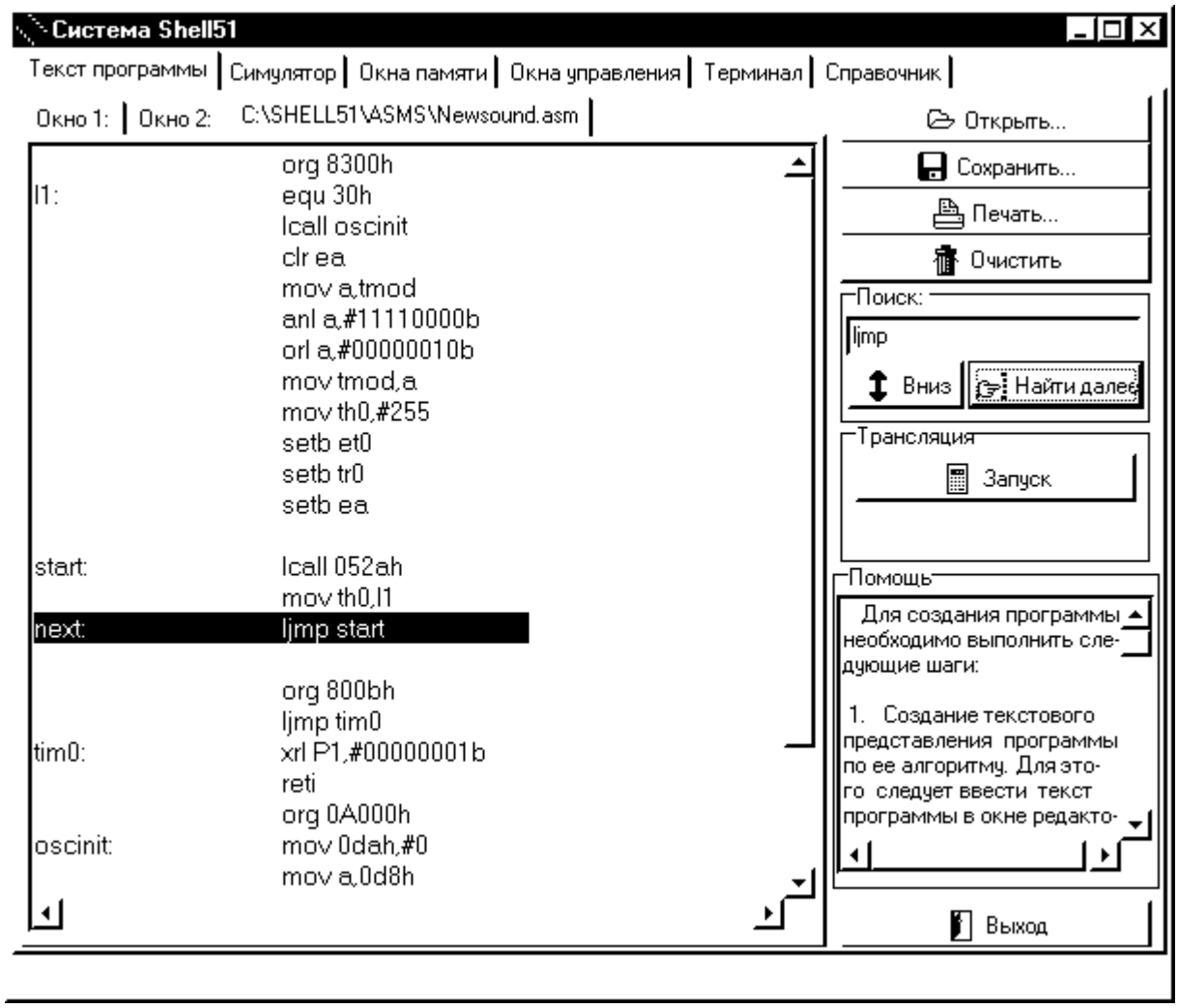


Рис. П-2. Поиск в тексте программы.

Отсутствие выделенной строки означает отсутствие введенного текста в данной программе.

ВКЛАДКА “ЛИСТИНГИ” предназначена для анализа результатов выполнения этапа трансляции и компоновки.

Активизация данного окна-вкладки происходит при выполнении пользователем двойного щелчка манипулятором "мышь" на строке-сообщении системы о результате трансляции (рис. П-3).

Рис. П-3. Текст с сообщением о результате трансляции.

Изображение окна-вкладки показано на рис. П-4.

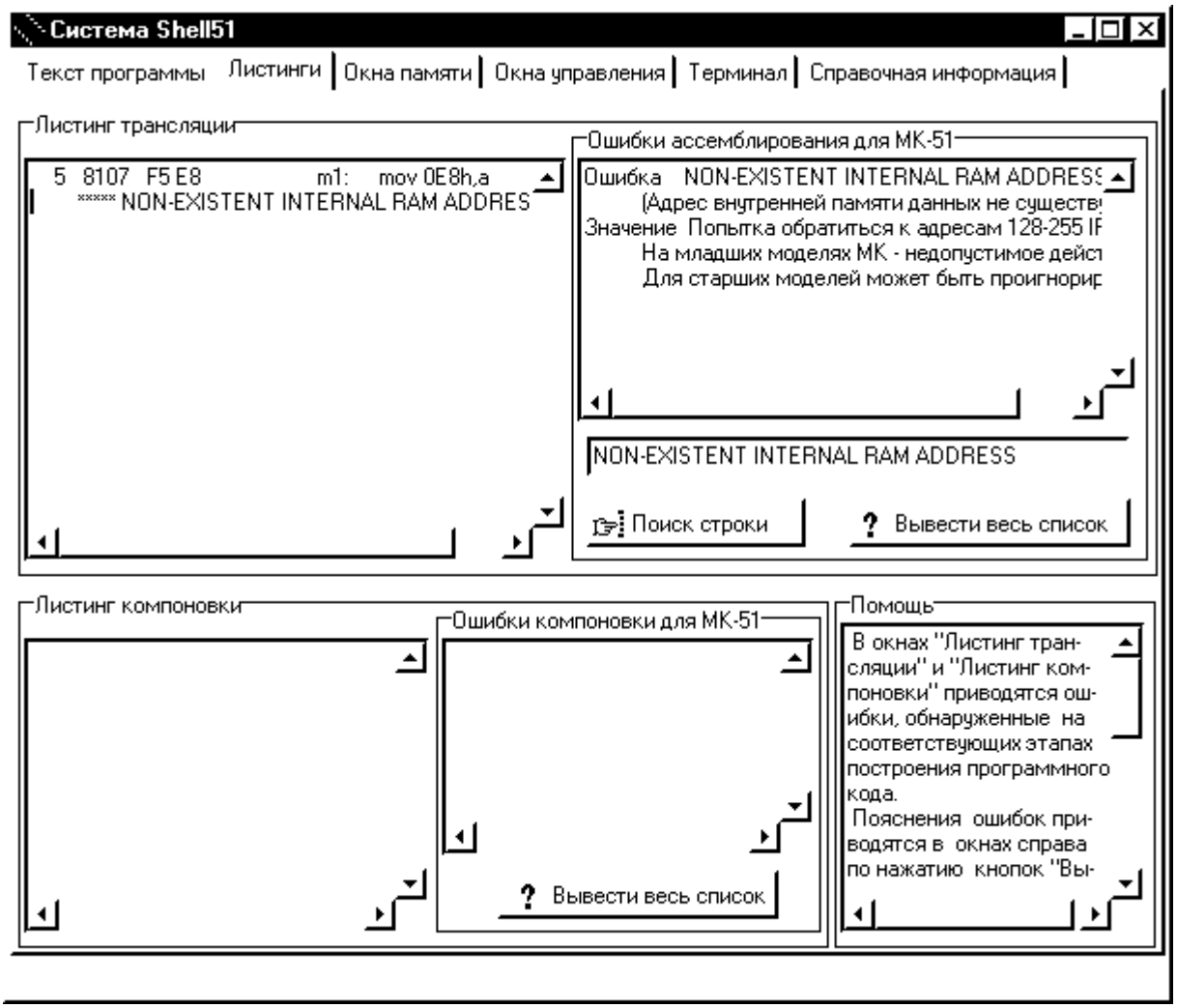


Рис. П-4. Окно-вкладка листингов трансляции компоновки.

Вкладка содержит поле вывода листинга результатов трансляции (слева вверху), поле вывода листинга результатов компоновки (слева внизу), поле перечня ошибок, обнаруживаемых ассемблером (справа вверху), поле перечня ошибок, обнаруживаемых компоновщиком (по центру внизу), поле помощи по использованию

данной вкладки (справа внизу), поле ввода строки контекстного поиска, а также три кнопки-панели.

В полях листинга отображаются результаты прохождения пользовательской программой этапов ассемблирования (т.е. преобразования программы из исходного текста в объектный код) и компоновки (т.е. преобразования программы из объектного кода в шестнадцатеричный исполнимый код). Наличие информации в данных окнах свидетельствует о присутствии ошибок в программе пользователя, ошибках объектных библиотек либо присутствии предупреждений со стороны транслятора и/или компоновщика. В подобном случае необходим анализ текста программы, выявление недопустимых элементов пользовательской программы и их исключение.

Для повышения эффективности процесса отладки система Shell51 предоставляет пользователю возможность просмотра полного перечня ошибок этапов ассемблирования и компоновки, выявляемых средой (с помощью соответствующей кнопки-панели "Вывести весь список"), а для транслятора с языка ассемблера - дополнительно возможность контекстного поиска информации по данной ошибке. Проведение контекстного поиска требует наличия текста сообщения об ошибке в поле строки поиска, что производится следующим образом:

- с помощью манипулятора "мышь" либо с использованием комбинации клавиш Shift+→ выделяется строка сообщения (в процессе выделения изменит фон);
- посредством клавиш Ctrl+C данная строка копируется в буфер обмена;
- с помощью однократного нажатия манипулятора мышь курсор переносится в поле строки поиска, а комбинация клавиш Ctrl+V вставляет текст из буфера обмена в искомое поле.

Получив в поле строки контекстного поиска строку сообщения об ошибке (сообщение можно ввести и обычным путем с клавиатуры), следует активизировать кнопку-панель "Поиск строки". В поле "Ошибки ассемблирования" появится детальное описание данной ошибки и рекомендации по ее устранению. Пустое поле свидетельствует о неточном вводе строки либо отсутствии описания данной ошибки.

При наличии ошибок результирующий исполнимый файл для микроконтроллера создан не будет.

Неправомочное игнорирование пользователем предупреждений может повлечь за собой работу с исполнимым файлом, не соответствующим исходной программе.

ВКЛАДКА “СИМУЛЯТОР” (рис. П-5) – предназначена для моделирования работы программы микроконтроллера.

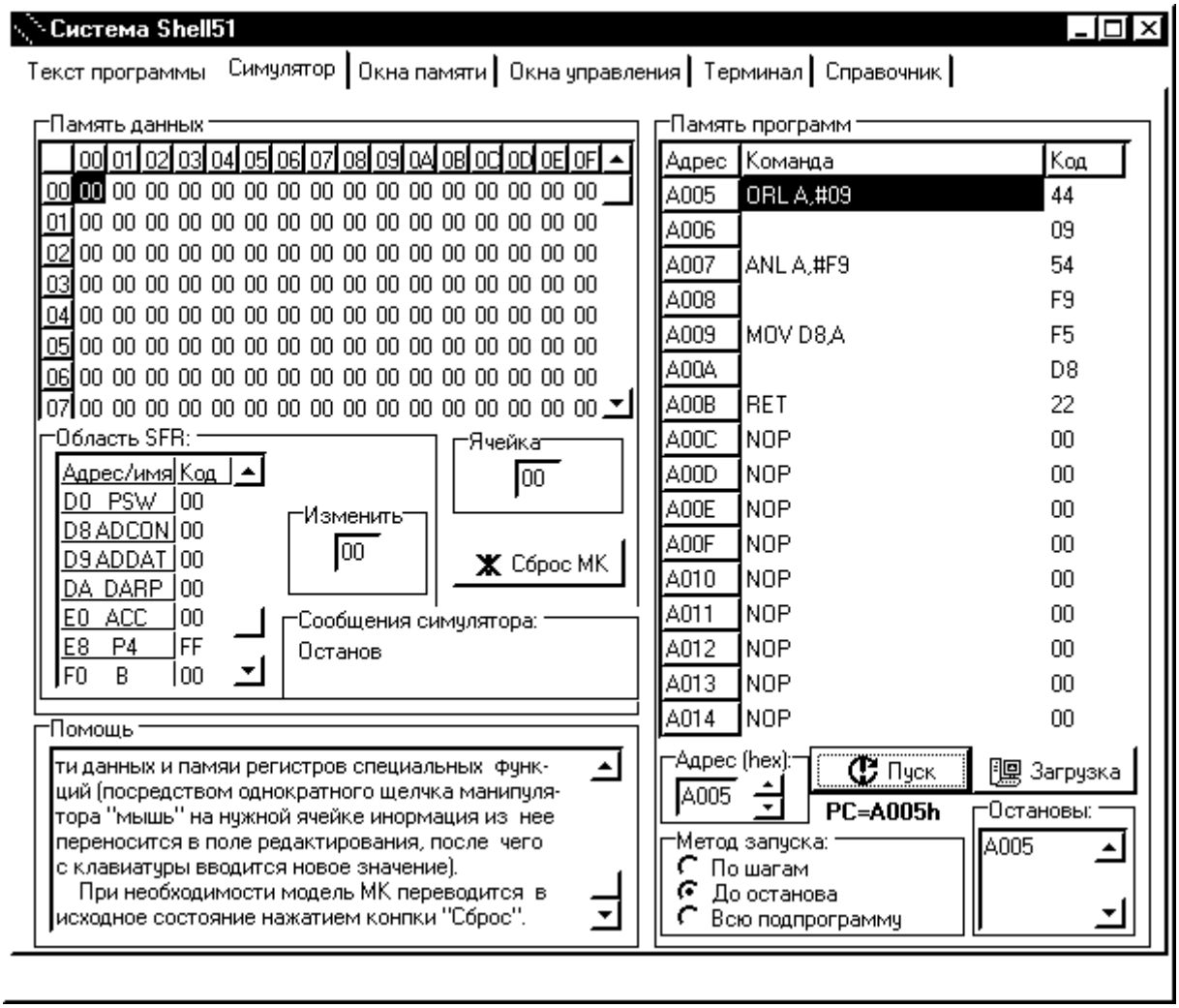


Рис. П-5. Вкладка "Симулятор".

Вкладка содержит окно памяти данных симулятора (слева), окно памяти программ симулятора (справа), окно помощи (слева внизу) и органы управления симулятором (справа внизу).

Вкладка "Симулятор" позволяет загрузить оттранслированную программу в память симулятора и отслеживать ее выполнение по шагам, до установленного пользователем адреса или целиком.

Загрузка программы производится по нажатию кнопки-панели "Загрузка". Необходимый вариант запуска на выполнение - выбором из опции "Метод запуска" с последующим нажатием кнопки-панели "Пуск".

При выборе метода "По шагам" останов симулятора будет производиться каждый раз после выполнения очередной команды, при выборе метода "До останова" программа будет выполняться до тех пор, пока значение счетчика команд симу-

лятора не совпадет с одним из значений, находящихся в перечне "Остановы", при выборе метода "Всю подпрограмму" программа будет выполняться до тех пор, пока очередная выбранная команда не окажется командой возврата из подпрограммы.

Остановы задаются/удаляются однократным щелчком манипулятора "мышь" на адресе необходимой команды. При этом они добавляются/исключаются из одноименного окна.

Счетчик команд (PC) устанавливается на необходимый адрес с помощью двойного щелчка манипулятора "мышь" на строке памяти программ, содержащей необходимую команду. По ходу выполнения программы он отображает адрес очередной команды из памяти программ симулятора.

Просмотр содержимого памяти программ производится с помощью окна "Адрес". Допускается как непосредственный ввод шестнадцатеричного значения участка памяти, подлежащего просмотру, так и использование бегунка.

При отладке пользователю предоставляется возможность скорректировать информацию в памяти данных и памяти регистров специальных функций - посредством однократного щелчка манипулятора "мышь" на нужной ячейке информация из нее переносится в поле редактирования, после чего с клавиатуры вводится новое значение.

Память данных симулятора представлена в виде двумерного массива ячеек, адрес которых определяется как $У_{16}*10_{16}+Х_{16}$, где У - номер строки, в которой расположена ячейка, Х - номер столбца, содержащего искомую ячейку. Значения Х и У - шестнадцатеричные. Память регистров специальных функций представляет собой линейный список с указанием символических обозначений регистров.

При необходимости модель МК переводится в исходное состояние нажатием кнопки-панели "Сброс".

ВКЛАДКА “ОКНА ПАМЯТИ” (рис. П-6) – предназначена для работы с содержимым памяти программ и данных микроконтроллера.

Вкладка содержит окно внутренней памяти данных микроконтроллера (слева вверху), окно текущей ячейки внутренней памяти данных (в центре), окно внешней совмещенной памяти программ-данных (справа), окно справки по использованию данной опции (слева внизу), поле состояния связи и ряд кнопок-панелей.

Внутренняя память данных микроконтроллера представлена в виде двумерного массива ячеек, адрес которых определяется как $У_{16}*10_{16}+Х_{16}$, где У - номер строки, в которой расположена ячейка, Х - номер столбца, содержащего искомую ячейку. Значения Х и У - шестнадцатеричные.

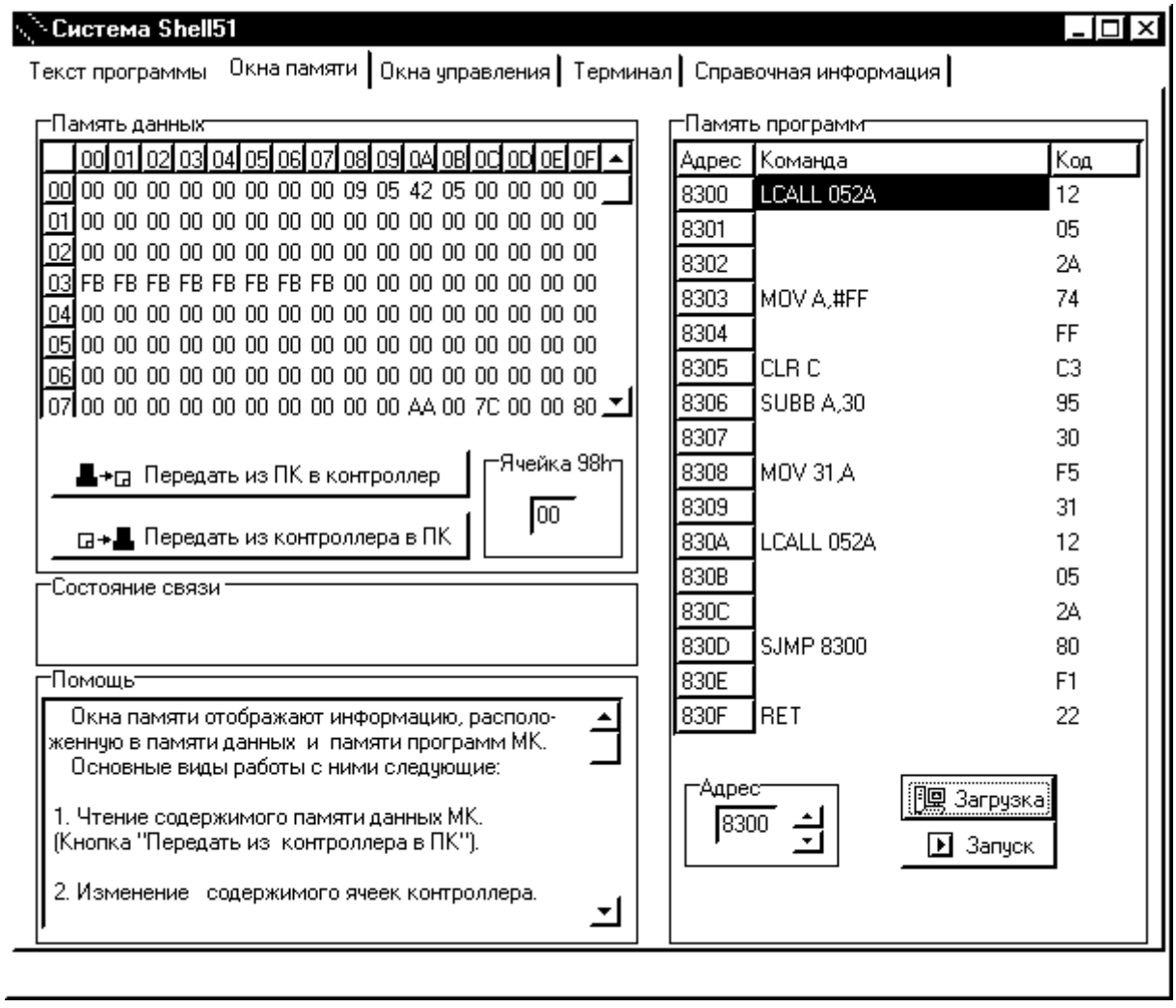


Рис. П-6. Вкладка "Окна памяти" системы.

Любая ячейка внутренней памяти данных допускает модификацию. Для ее осуществления необходимо задать ячейку, подлежащую модификации, путем однократного щелчка левой кнопкой манипулятора "мышь" на нужной ячейке, а затем с

помощью клавиатуры ввести необходимое новое значение в виде шестнадцатеричного кода в окне текущей ячейки (рис. П-7).

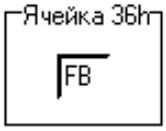


Рис. П-7. Ввод значения выбранной ячейки.

Для синхронизации информации между окном внутренней памяти данных среды Shell51 и внутренней памятью данных микроконтроллера служат кнопки-панели "Передать из ПК в контроллер" и "Передать из контроллера в ПК". Для записи новых значений в память данных контроллера следует использовать первую из них, для чтения содержимого внутренней памяти данных контроллера - вторую.

Занесение программы в исполняемом формате в память программ контроллера применяется кнопка-панель "Загрузить". По окончании загрузки в окне памяти программ будут расположены первые 16 байт пользовательской программы в виде дизассемблированного текста и ее кодовое представление, начиная с адреса, заданного директивой ассемблера "ORG <адрес>" в исходном тексте.

Кнопка-панель "Запуск" активизирует исполнение программы, находящейся в памяти программ контроллера, с адреса, указанного в окне "Адрес" в виде шестнадцатеричного значения. При необходимости значение адреса может быть скорректировано, при этом в окне памяти программ отобразится иной 16-байтный участок памяти программ, начинающийся с заданного адреса.

Результаты работы программы анализируются путем изучения измененных участков памяти.

В ходе обмена информацией между инструментальной ЭВМ и контроллером поле "Состояние связи" отображает статус линий связи. "Обмен с МК" свидетельствует о наличии очередного сеанса обмена, "Нет связи" - о невозможности взаимодействия ЭВМ и МК, при этом дальнейшие сеансы обмена блокируются. Необходимо выявить и устранить причину, а затем снять блокировку двойным щелчком манипулятора "мышь" по сообщению.

ВКЛАДКА “ОКНА УПРАВЛЕНИЯ” (рис. П-8) предназначена для интерактивного исследования систем управления и отладки пользовательских программ.

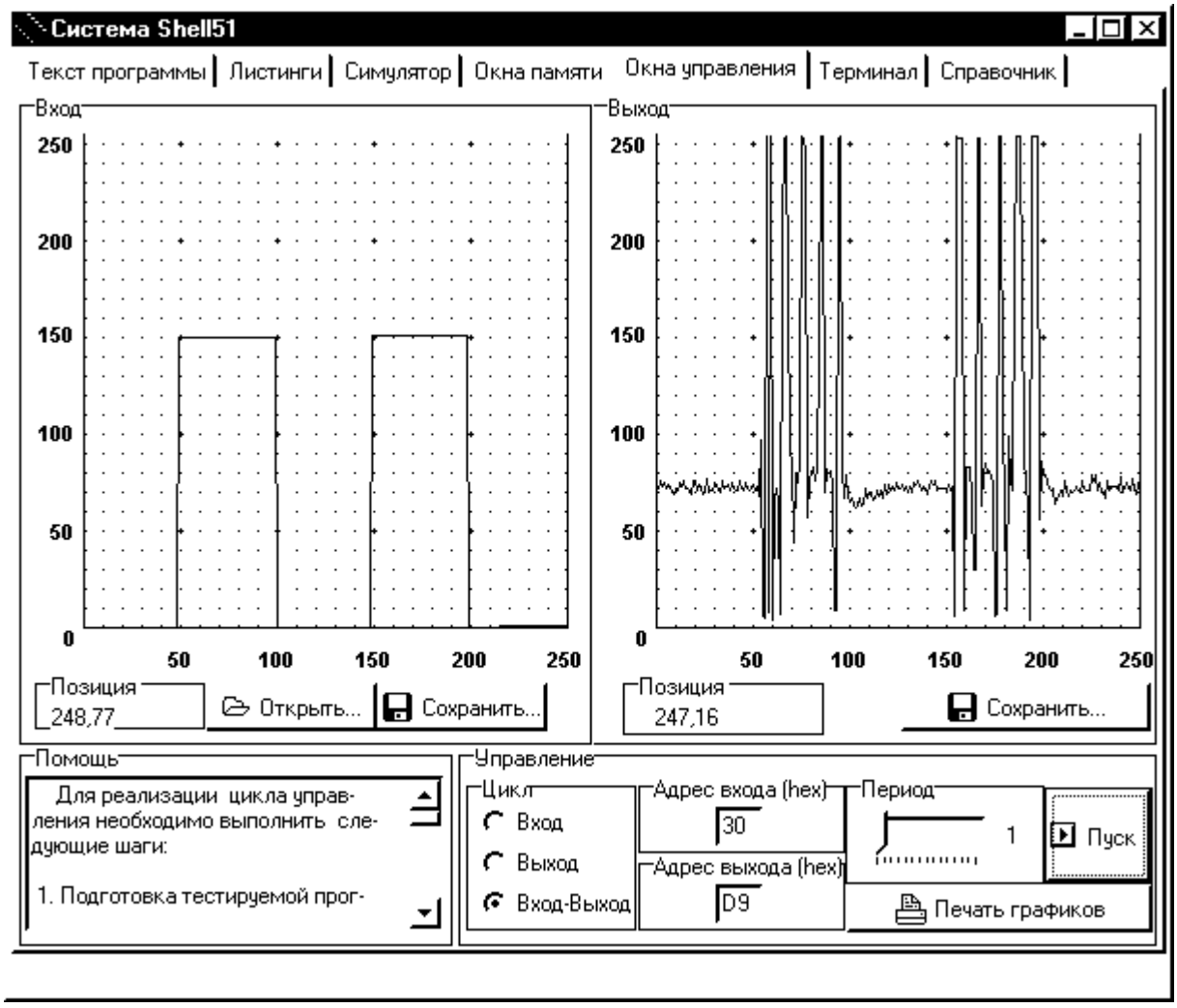


Рис. П-8. Вкладка "Окна управления".

Вкладка содержит: поле управляющего воздействия на контроллер (слева), поле реакции контроллера на управления (справа), поле справки (слева внизу), и поле настроек подсистемы (справа внизу).

В окне "Вход" пользователь, применяя манипулятор "мышь", строит график управляющего воздействия в виде кусочно-линейной однозначной функции Ф(t). Двойной щелчок означает начало/окончание режима ввода, однократный - фиксацию текущего отрезка графика. Для удобства в поле "Позиция" отображаются текущие координаты манипулятора "мышь".

Кнопки-панели "Открыть" и "Сохранить" позволяют использовать ранее созданные управляющие воздействия и запоминать на диске инструментальной ЭВМ вновь создаваемые управляющие воздействия и полученные на них реакции.

Окно "Выход" позволяет отобразить график функции M(t), являющейся результатом функционирования пользовательской программы на контроллере.

Возможны три варианта комбинаций функций Ф и М:

- только функция Ф (подача управляющих воздействий без ввода реакции);
- только функция М (опрос состояния контроллера, без вывода управления);
- совместное применение функций Ф и М.

Выбор необходимого варианта осуществляется установкой переключателя "Цикл" в требуемое положение.

Функции Ф и М представляют собой развернутые во времени значения задаваемых пользователем ячеек внутренней памяти данных, в связи с чем обе функции нужно адресовать, указав необходимые значения в полях "Адрес входа" и "Адрес выхода".

Длительность развертывания наблюдаемых процессов в условных временных единицах задается изменением положения движка "Период" посредством манипулятора "мышь". Графики функций Ф и М могут быть распечатаны на принтере с помощью кнопки-панели "Печать графиков". Начало сеанса интерактивного управления-регистрации привязывается к моменту активизации кнопки-панели "Пуск".

Варианты применения опции:

- в МК функционирует программа управления объектом, реализующая это управление на основе задатчика, которым является окно входа. В данном случае целесообразно в качестве реакции визуализировать отклики реального объекта;
- на МК исполняется программа, которую, перед использованием совместно с объектом, необходимо протестировать на соответствие заданию. В данном случае в окне входа следует задать предполагаемые сигналы с объекта, а визуализировать целесообразно реакцию со стороны МК.

Следует отметить, что программа, подлежащая исследованию в данной опции, должна быть модифицирована. В участок программы, принимающий функцию Ф либо формирующий функцию М, необходимо ввести команду перехода на подпрограмму однократного сеанса связи Lcall 052Ah, что позволит инструментальной ЭВМ выдать очередное значение управляющей функции и/или принять очередное значение результирующей функции М. Как правило, программное управление организуется в виде цикла, внутрь которого и следует поместить указанный вызов. Другим вариантом является организация взаимодействия МК и ЭВМ по прерываниям от приемопередатчика, в обработчике которых следует предусмотреть данную команду.

ВКЛАДКА “ТЕРМИНАЛ” (рис. П-9) предназначена для исследования последовательного интерфейса, стыкующего МК и ЭВМ, без участия связных компонент программы-монитора, и построения пользовательских связных компонент.

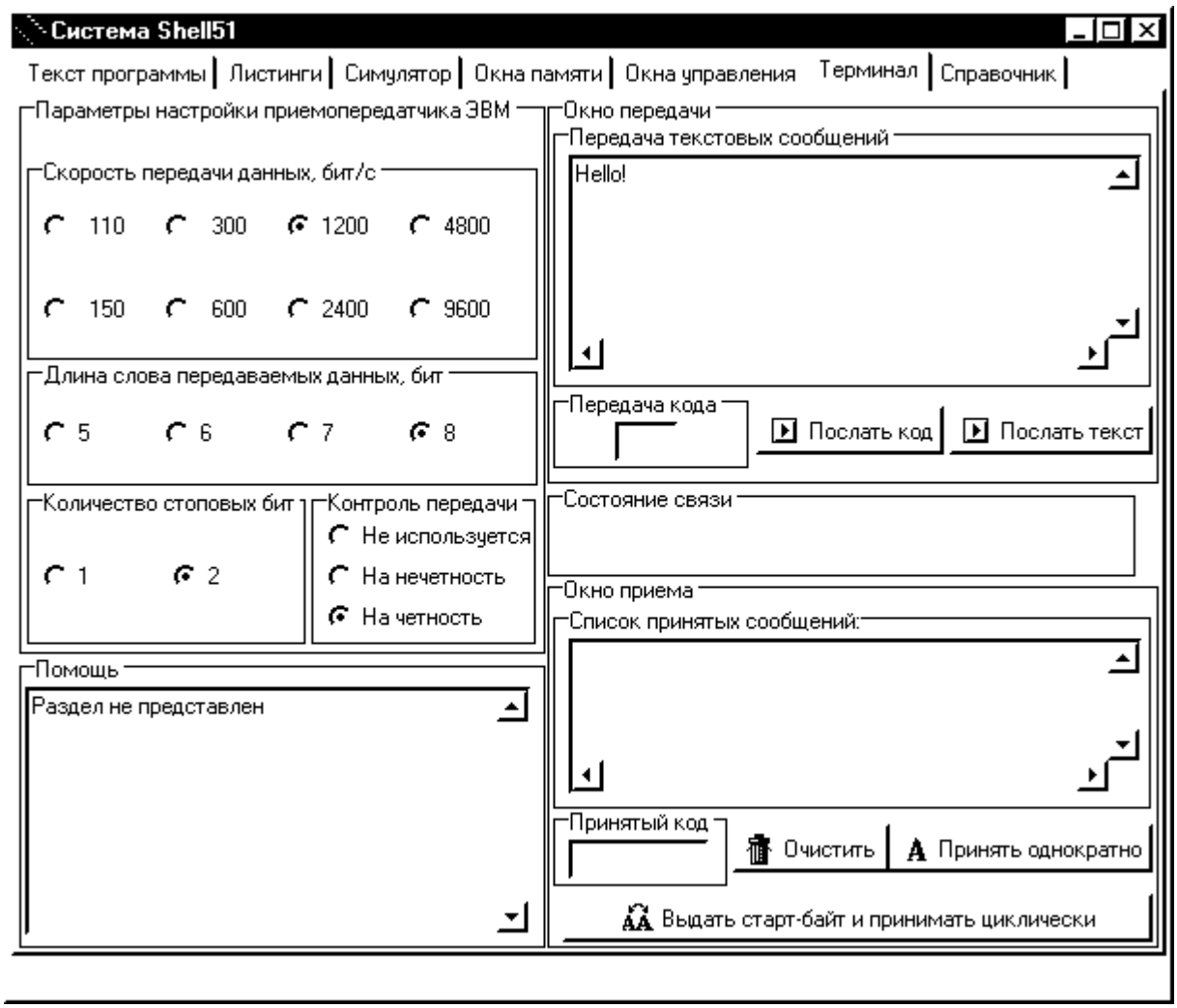


Рис. П-9. Вкладка "Терминал".

Вкладка содержит: поле настроек последовательного приемопередатчика инструментальной ЭВМ (слева вверху), окно справки (слева внизу), поле окна передачи (справа вверху), поле окна приема (справа внизу) и поле состояния связи (справа в центре).

Используя переключатели поля настроек приемопередатчика ЭВМ, пользователь задает необходимые режимы его работы (аналогичные значения скорости передачи, длины посылки и т.д., должны быть заданы и для приемопередатчика МК).

Следуя логике программы, функционирующей на МК (созданной и загруженной в МК штатными средствами системы), пользователь применяет следующие варианты работы с подсистемой:

1. Посылка информации в порт:
   - если посылка однобайтная, используется окно передачи шестнадцатеричных кодов и кнопка-панель "Послать код", активизирующая выдачу данного значения в последовательный порт на МК;
   - если посылка многобайтная, целесообразно использовать окно передачи текста, предварительно введенного с клавиатуры ЭВМ, и кнопка-панель "Послать текст", при этом производится выдача кодов символов, составляющих текст.

2. Прием информации из порта:

- при активизации кнопки-панели "Принять однократно", код, содержащийся в буфере приемника ЭВМ, будет отображен в окне "Принятый код", а также отображен как очередной эквивалентный символ в окне "Принятый текст". Кнопка-панель "Очистить" позволяет освободить окна приема от содержащейся в них информации;
- при активизации кнопки-панели "Выдать старт-байт и принимать циклически" система вышлет в последовательный порт код $A5_{16}$, а затем перейдет к циклу ожидания поступления входной информации из последовательного порта. Указанный старт-байт может быть использован программистом микроконтроллера как уведомление о необходимости передавать результаты в ЭВМ.

3. Комбинация вышеуказанных вариантов.

При наличии ошибок в логике взаимодействия пользователя и программы МК возможно нарушение обмена, обозначаемое сообщением "Нет связи" в поле статуса линии. Необходимо выявить и устранить причину, после чего разблокировать приемопередатчик путем двойного щелчка манипулятором "мышь" на указанном сообщении.

ВКЛАДКА "СПРАВОЧНИК" содержит сведения, необходимые для изучения аппаратуры и программирования микроконтроллерных систем. Вкладка организована в виде гипертекстового документа (рис. П-10).

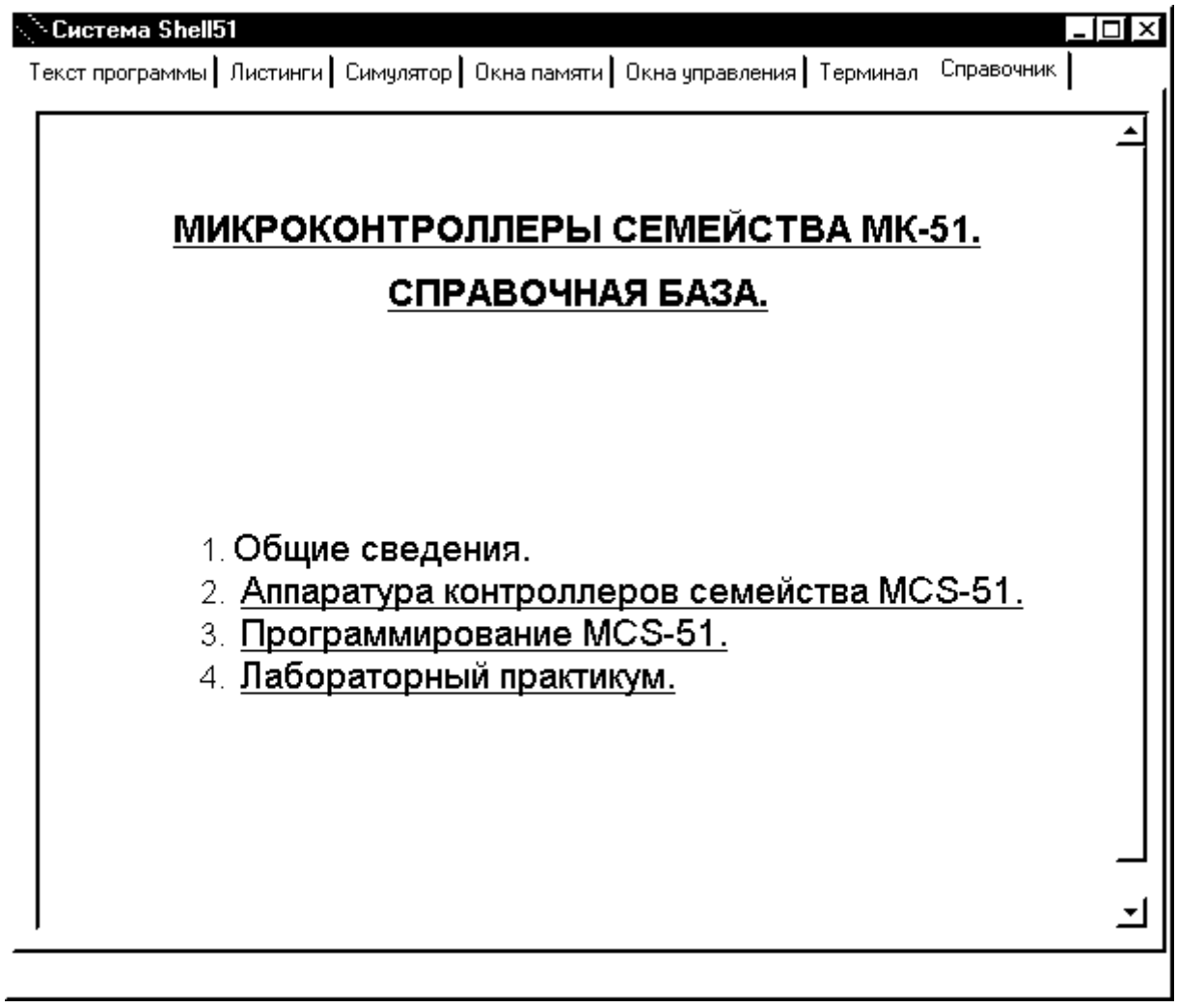


Рис. П-10. Вкладка "Справочная информация".

Выделенные цветом и подчеркиванием элементы документа являются ссылками на соответствующие разделы, переход на которые осуществляется щелчком левой кнопки манипулятора "мышь". Разделы в свою очередь состоят из подразделов с аналогичным принципом переходов. Для возврата на предыдущий уровень используются ссылки "Назад" или "Возврат", присутствующие во всех документах.

ВАСИЛЬЕВ Алексей Евгеньевич

МИКРОКОНТРОЛЛЕРЫ

Разработка встраиваемых приложений

Учебное пособие

Лицензия ЛР № 020593 от 07.08.97

Подписано в печать 19.05.2003. Формат 60x84/16

Печать офсетная. Усл. печ. л. 13,12. Уч.-изд. л. 13,12 Тираж 150. Зак.

Отпечатано с готового оригинал-макета, предоставленного автором, в типографии Издательства СПбГТУ.

195251, Санкт-Петербург, Политехническая ул., д. 29