

А. Г. Алексенко, А. А. Галицын,
А. Д. Иванников

ПРОЕКТИРОВАНИЕ
РАДИО-
ЭЛЕКТРОННОЙ
АППАРАТУРЫ
НА МИКРО-
ПРОЦЕССОРАХ

А. Г. Алексенко, А. А. Галицын, А. Д. Иванников

А. Г. Алексенко, А. А. Галицын,
А. Д. Иванников

ПРОЕКТИРОВАНИЕ
РАДИОЭЛЕКТРОННОЙ
АППАРАТУРЫ
НА МИКРОПРОЦЕССОРАХ
ПРОГРАММИРОВАНИЕ, ТИПОВЫЕ
РЕШЕНИЯ, МЕТОДЫ ОТЛАДКИ



Москва «Радио и связь» 1984

ББК 32.973.2

А47

УДК 681.325.5—181.4

Алексенко А. Г., Галицын А. А., Иванников А. Д.

А47 Проектирование радиоэлектронной аппаратуры на микропроцессорах: Программирование, типовые решения, методы отладки. — М.: Радио и связь, 1984. — 272 с., ил.

В пер. 1 р. 10 к.

Даются основные принципы проектирования цифровой аппаратуры на основе микропроцессорных средств. Основное внимание уделяется разработке технических средств и программного обеспечения для сопряжения микро-ЭВМ с управляемым объектом. ЗУ микропроцессорных систем, комплексной отладке микропроцессорных систем с использованием аппаратно-программных отладочных комплексов. Подробно рассматривается процесс проектирования систем числового программного управления и цифровых фильтров на основе микропроцессоров.

Для инженерно-технических работников, связанных с разработкой аппаратуры на микропроцессорах.

А $\frac{2403000000-003}{046(01)-84}$ 60—83

ББК 32.973.2

БФ7.3

Рецензенты: д-р техн. наук **А. А. Красилов** и канд. техн. наук **В. А. Шахнов**

Редакция литературы по электронной технике

Предисловие

В настоящее время партия и правительство уделяют большое внимание развитию средств автоматизации производства. Разработка современных автоматизированных систем предполагает использование микропроцессорной техники. В материалах XXVI съезда КПСС указано на необходимость «...развивать производство и обеспечить широкое применение автоматических манипуляторов (промышленных роботов), встроенных систем автоматического управления с использованием микропроцессоров и микро-ЭВМ»¹. Отечественной промышленностью освоен серийный выпуск микропроцессорных БИС, позволяющих создавать различную управляющую, информационную, связную, вычислительную, медицинскую и бытовую аппаратуру. Благодаря работам отечественных специалистов [1, 3—7, 15—19, 21, 24] информация об основных принципах построения микропроцессоров и микро-ЭВМ, их программировании и возможностях построения аппаратуры на их основе, описание некоторых комплектов микропроцессорных БИС и микро-ЭВМ доступны широкому кругу читателей. Однако информации, приводимой в опубликованных к настоящему времени работах, для проектирования микропроцессорной аппаратуры недостаточно. В книге делается попытка в определенной степени восполнить этот пробел.

Полное изложение всех этапов проектирования радиоэлектронной аппаратуры на основе микропроцессорных средств — задача слишком большая для одной книги. В связи с этим основное внимание уделяется подробному анализу функционирования микропроцессорных БИС, методике разработки программного обеспечения и отладке микропроцессорных систем — этапам, специфичным для процесса проектирования микропроцессорной аппаратуры.

В связи с ограниченным объемом книги и требованием полноты изложения в основу книги положено

¹ Материалы XXVI съезда КПСС. — М.: Изд-во политической литературы, 1981, с. 144.

проектирование микропроцессорных систем на базе одной серии БИС—БИС КР580. Этот выбор обусловлен следующими причинами:

1. Микропроцессорный комплект БИС КР580 является комплектом общего назначения и используется в управляющей, информационной, измерительной, связанной, медицинской, бытовой и другой аппаратуре самого различного применения.

2. Разработчики аппаратуры в указанных областях применения, как правило, ранее не сталкивались с использованием средств вычислительной техники. Освоение методов разработки аппаратуры на микропроцессорах для них является сложной задачей.

3. Комплект БИС серии КР580 представляет собой один из первых отечественных микропроцессорных комплектов. К настоящему времени уже накоплен достаточный опыт построения аппаратуры на его основе.

Хотя описание основных функций этих БИС имеется в ряде монографий [4, 13, 18], детальные сведения по их характеристикам и программированию (за исключением самого микропроцессора КР580ИК80А), необходимые разработчикам аппаратуры, в отечественной литературе еще не публиковались. В книге рассматривается разработка программного обеспечения микропроцессорных систем. Наряду с общими понятиями автоматизации программирования приведено подробное описание двух наиболее употребительных для серии КР580 языков программирования: ассемблера и ПЛ/М. Описана кросс-система разработки программного обеспечения на базе мини-ЭВМ СМ-4. Описаны часто используемые блоки микропроцессорных систем на БИС КР580: процессорный блок, блоки памяти, схемы сопряжения процессорного блока с распространенными внешними устройствами.

Разработка и комплексная отладка технических средств и программного обеспечения микропроцессорных систем рассмотрены на примере отладочного комплекса Intellec MDS-800 фирмы Intel (США). В книге содержится ряд примеров применения комплекта БИС серии КР580: реальной микропроцессорной системы числового программного управления чертежным автоматом и микропроцессорной реализации цифровых фильтров.

Введение написано А. Г. Алексенко; гл. 1 и 4 — А. Г. Алексенко и А. А. Галицыным; гл. 2 и § 5.2 —

А. Г. Алексенко и А. Д. Иванниковым; гл. 3, § 4.1, 5.1 и приложения — А. Д. Иванниковым.

Авторы искренне признательны рецензентам д-ру техн. наук А. А. Красилову и канд. техн. наук В. А. Шахнову за ценные замечания, способствовавшие улучшению качества книги, а также Г. В. Зеленко, С. Б. Бочарову и С. Е. Галкину за помощь в работе и оформлении рукописи.

Отзывы и замечания по книге просьба направлять по адресу: 101000, Москва, Почтамт, а/я 693, издательство «Радио и связь».

Введение

Рассмотрим основные понятия и определения, принятые в микропроцессорной технике и используемые в данной книге [1, 2, 7, 17].

Микропроцессор (МП) — программно-управляемое устройство, осуществляющее процесс обработки цифровой информации и управление им, построенное на одной или нескольких БИС. Примером МП является отечественный центральный процессор КР580ИК80А или аналогичный ему процессор 8080 фирмы Intel (США).

Микропроцессорная БИС — интегральная микросхема, выполняющая функцию МП или его части. По существу — это БИС с процессорной организацией, разработанная для построения микропроцессорных систем.

Микропроцессорный комплект (МПК) — совокупность микропроцессорных и других интегральных микросхем, совместимых по конструктивно-технологическому исполнению и предназначенных для совместного применения при построении МП, микро-ЭВМ и других средств вычислительной техники.

Микропроцессорная электронная вычислительная машина (микро-ЭВМ) — ЭВМ, состоящая из микропроцессора (микропроцессоров), полупроводниковой памяти, средств связи с периферийными устройствами и при необходимости пульта управления и источника питания, объединенных общей несущей конструкцией. Микро-ЭВМ — универсальное вычислительное устройство с законченной конструкцией и развитым стандартным программным обеспечением.

Однокристалльная микро-ЭВМ — микро-ЭВМ, выполненная в виде одной БИС. В этом случае на одном

кристалле размещаются процессор, постоянное запоминающее устройство (ПЗУ) для хранения программы, оперативное запоминающее устройство (ОЗУ) для хранения промежуточных результатов, каналы ввода — вывода, в ряде случаев таймер.

Одноплатная микро-ЭВМ — микро-ЭВМ, выполненная в виде одной печатной платы, предназначенная для встраивания в радиоэлектронную аппаратуру. Как правило, однокристалльные и одноплатные микро-ЭВМ допускают подключение дополнительных плат памяти и контроллеров устройств ввода — вывода.

Микропроцессорные средства — МПК БИС, однокристалльные и одноплатные микро-ЭВМ.

Микропроцессорная система — управляющая, информационная или иная специализированная цифровая система, построенная на базе микропроцессорных средств и включающая микро-ЭВМ и средства сопряжения с обслуживаемым объектом. Обычно микропроцессорная система выполняет ограниченный набор программ, хранящихся в ПЗУ.

Технические средства микропроцессорной системы (микро-ЭВМ) — микропроцессорные средства и схемы сопряжения с обслуживаемым объектом, размещенные на некоторой конструкции и соединенные согласно некоторой принципиальной схеме.

Программное обеспечение микропроцессорной системы (микро-ЭВМ) — последовательность команд (программа или набор программ), размещенная в ЗУ микропроцессорной системы и реализующая требуемый алгоритм ее функционирования.

Таким образом, микропроцессорная система — это комплекс технических средств и программного обеспечения.

Разработка цифровой аппаратуры на основе микропроцессорных средств имеет следующие особенности, отличающие ее от проектирования цифровой аппаратуры на ИС малой и средней степеней интеграции [1, 2, 6, 14]:

1. Необходимость совместной разработки и отладки технических средств и программного обеспечения, ориентированного на конкретную структуру технических средств.

2. Использование принципиально новых методов и средств разработки и отладки микропроцессорных си-

стем (внутрисхемных эмуляторов, логических анализаторов, отладочных тестеров [9]), а также средств автоматизации программирования.

3. Сильная взаимосвязь и даже интеграция этапов проектирования, при которой разработчик должен одновременно обладать опытом проектирования микропроцессорных систем, а также разбираться в конкретной области их применения (например, в числовом программном управлении, автомобильной электронике, измерительной технике).

Процесс проектирования микропроцессорной системы включает следующие этапы проектирования [3]: системный, структурно-алгоритмический и технический.

На этапе системного проектирования исходя из требований к микропроцессорной системе выбирают основные функциональные блоки ее технических средств и программного обеспечения, а также связи между ними. В частности, осуществляется выбор микропроцессорных средств, периферийных устройств, организации системы прерываний и системы ввода — вывода; задаются пропускные способности каналов. Системное проектирование является в основном эвристическим, его результат — функциональная схема микропроцессорной системы.

При выборе микропроцессорных средств могут использоваться два варианта: серийно выпускаемые микро-ЭВМ и специально разрабатываемые на основе МПК БИС микропроцессорные системы. При выборе одного из этих вариантов необходимо прежде всего учитывать стоимость, сроки выпуска аппаратуры, специальные требования и квалификацию разработчиков.

Основными факторами, определяющими выбор конкретных серий микропроцессорных средств, являются: производительность, разрядность, наличие средств проектирования и отладки, потребляемая мощность, надежность, число типов БИС в базовом комплекте, климатические и механические параметры, доступность, стоимость, опыт проектирования систем на основе выбираемых микропроцессорных средств.

Основу микропроцессорного комплекта БИС составляют либо МП с фиксированной разрядностью и фиксированной системой команд, либо микропрограммируемые процессорные секции с разрядностью 2, 4, 8, 16. В последнем случае параллельным соединением

секций можно строить микропроцессорные системы для обработки данных любой разрядности, кратной разрядности секции. Набор команд в этом случае может быть выбран разработчиком в соответствии с конкретным назначением системы и реализован с помощью микропрограммирования [2, 3, 5, 6]. Микропроцессорные системы на основе микропрограммируемых процессорных секций характеризуются высокой производительностью, однако сложность и стоимость проектирования цифровой системы на их основе существенно больше. Поэтому если комплект с фиксированной системой команд, например комплект серии КР580, удовлетворяет требованиям по производительности, то ему, безусловно, должно быть отдано предпочтение.

На системном этапе проектирования необходимо определить, какие функции микропроцессорной системы будут реализовываться аппаратным, а какие — программным способом. Как правило, первый способ увеличивает производительность системы, но усложняет и удорожает ее технические средства.

Функциональная схема микропроцессорной системы является основой для разработки структурной схемы технических средств и алгоритмов целевых программ, разрабатываемых на этапе структурно-алгоритмического проектирования. Этот этап проектирования микропроцессорной системы характеризуется широким использованием типовых схемных и программных решений и сильной взаимозависимостью технических и программных средств. Поэтому разработка технических средств и программного обеспечения должна осуществляться параллельно на всех этапах.

Если проектирование и отладку программного обеспечения и технических средств микропроцессорной системы проводить раздельно, то в результате проверки их работоспособности как системы могут быть выявлены ошибки их совместного функционирования. При этом трудно определить, чем вызвана неправильная работа системы (устройства): неисправностью технических средств или невыявленными ошибками в программе. Такая ситуация потребовала создания специальных отладочных средств — внутрисхемных эмуляторов, позволяющих одновременно отлаживать программные и технические средства разрабатываемой микропроцессорной системы.

Самыми эффективными средствами проектирования являются специализированные отладочные комплексы (см. гл. 4), которые позволяют без использования дополнительного оборудования разрабатывать программные и технические средства проектируемой микропроцессорной системы, а также одновременно и на реальном макете их отлаживать. Выполнение этих функций

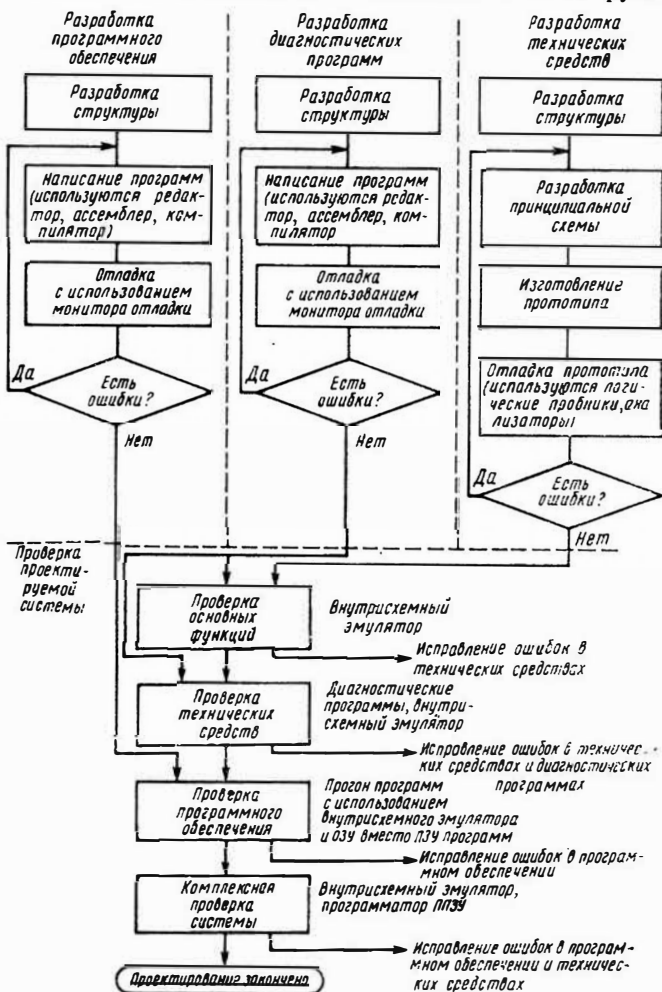


Рис. В.1. Последовательность проектирования микропроцессорных систем

возможно с использованием дисковой операционной системы (ДОС) и внутрисхемного эмулятора, входящих в состав комплексов. Дисковая операционная система позволяет работать с большими объемами информации и представляет собой набор вспомогательных программ, необходимых для разработки программного обеспечения и помогающих управлять ресурсами комплекса при разработке, отладке и выполнении целевых программ.

Внутрисхемный эмулятор — это совокупность программно-аппаратных средств, которые дают возможность разработчику активно контролировать ход вычислительного процесса в отлаживаемой системе, а также предоставлять в распоряжение разрабатываемой системы те аппаратные ресурсы, которые на этапе разработки в ней еще отсутствуют, но которыми обладает отладочный комплекс (память, внешние устройства и т. п.).

Отладка технических средств с помощью внутрисхемного эмулятора осуществляется на системе-прототипе, т. е. на микропроцессорной системе, в которую на ранних стадиях разработки из всех возможных функциональных блоков входят, по крайней мере, МП и схемы сопряжений с шиной управления (ШУ), шиной адресов (ША) и шиной команд-данных (ШК). Пошаговое исполнение программ и поэтапное введение в систему-прототип реальных интерфейсных устройств, которые обеспечивает внутрисхемный эмулятор, позволяют достаточно просто осуществлять тестирование и совместную отладку технических средств системы и ее программно-го обеспечения. Общая последовательность разработки и отладки микропроцессорных систем с использованием специализированных отладочных комплексов приведена на рис. В. 1.

Глава 1.

Микропроцессорный комплект БИС серии КР580

Микропроцессорный комплект БИС КР580 (табл. 1.1) предназначен для построения 8-разрядных контроллеров, микро-ЭВМ и микропроцессорных систем различного назначения. Функциональная законченность МПК, относительная простота разработки микропроцессорных систем на его основе, достаточно высокое быстродействие и сравнительно низкая потребляемая мощность обуславливают возможность широкого применения этого МПК в различных областях народного хозяйства. Микропроцессорный комплект КР580 выполнен на основе *n*-МОП технологии. Уровни электрических сигналов микросхем комплекта соответствуют уровням широко распространенной серии К155. Кроме того, с комплектом совместимы микросхемы серии К589, выполненные по ТТШЛ технологии [5]. Микросхемы КР580ИК80А, КР580ВВ51 и КР580ВВ55 являются схемотехническими аналогами микросхем К580ИК80, К580ИК51 и

Таблица 1.1

Состав МПК БИС серии КР580

Обозначение	Наименование	Максимальная тактовая частота, МГц	Напряжение питания, В	Тип корпуса	Существующие аналоги
КР580ИК80А	Микропроцессор	2,5	5; 12; —5	2123.40-2	К580ИК80
КР580ВВ51	Программируемый последовательный интерфейс	2,0	5	2123.40-2	К580ИК51
КР580ВВ55	Программируемый параллельный интерфейс	2,0	5	2123.40-2	К580ИК55
КР580ВВ53	Программируемый таймер	2,0	5	2123.24-3	
КР580ВТ57	Программируемый контроллер прямого доступа к памяти	2,0	5	2123.40-2	
КР580ВН59	Программируемый контроллер прерываний	2,0	5	2121.28-5	
КР580ВГ75	Программируемый контроллер электронно-лучевой трубки	3,0	5		

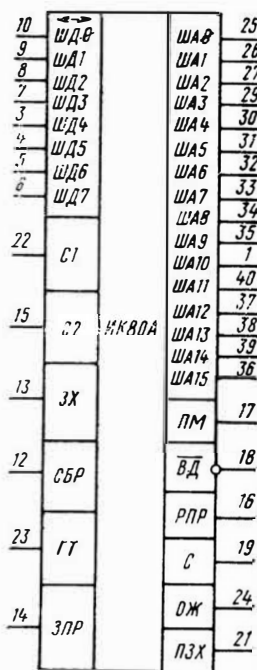
К580ИК55 и отличаются типом корпуса. Кроме того в МП КР580ИК80А по сравнению с МП К580ИК80 максимальная тактовая частота повышена с 2 до 2,5 МГц.

1.1. Микропроцессор КР580ИК80А

Восьмиразрядный однокристалльный МП КР580ИК80А с фиксированной системой команд предназначен для построения средств обработки данных, применяемых в устройствах автоматики, контроллерах и микро-ЭВМ. Он помещен в пластиковый корпус с 40 выводами (рис. 1.1) и потребляет мощность 750 мВт. Его тактовая частота составляет 2,5 МГц [18].

Микропроцессор КР580ИК80А (рис. 1.2) имеет 8-разрядную внутреннюю шину данных. Обмен информацией между внутренней шиной данных и шиной данных микропроцессорной системы ШД [0—7] осуществляется через двунаправленный буферный регистр данных. Шестнадцатиразрядная шина адресов ША [0—15] позволяет МП осуществлять непосредственную адресацию до 64 Кбайт памяти. Буферные регистры адресов и данных имеют выходы с тремя состояниями (высокий уровень, низкий уровень и состояние с высоким выходным сопротивлением), что позволяет отключать МП от шин системы при реализации режима прямого доступа к памяти со стороны внешних устройств.

Восьмиразрядное арифметико-логическое устройство (АЛУ) позволяет осуществлять операции над данными, представленными в двоичной форме и находящимися в регистре накопителя и регистре временного хранения. Результат выполнения операции поступает на внутреннюю шину данных или записывается в накопитель. Блок десятичной коррекции позволяет при необходимости осуществлять перевод содержимого накопителя из двоичной в двоично-десятичную форму. Информация о результате выполнения команды заносится в 5-разрядный регистр признаков. Разряд Z (Нуль) регистра признаков устанавливается в единицу при нулевом результате, разряд C (Перенос) — при возникновении переноса, разряд S (Знак) —



5B — 20
 -5B — 11
 12B — 28
 Общий — 2

Рис. 1.1. Нумерация выводов КР580ИК80А

при единичном значении старшего разряда, разряд P (Четность) — при четном числе единичных битов в результате, разряд AC (Дополнительный перенос) — при переносе из младших четырех разрядов в старшие).

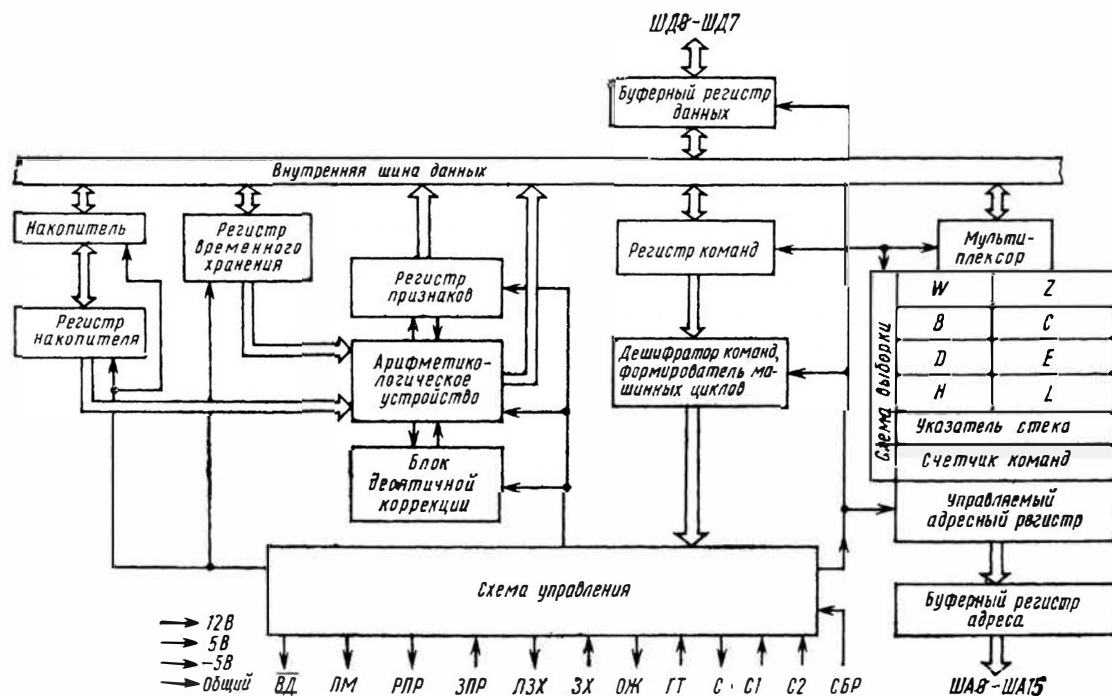


Рис. 1.2. Структура МП КР5801К80А

Блок регистров МП включает шесть (В, С, D, E, H, L) 8-разрядных регистров общего назначения, которые в зависимости от типа выполняемой команды могут использоваться либо как самостоятельные регистры, либо как 16-разрядные регистровые пары

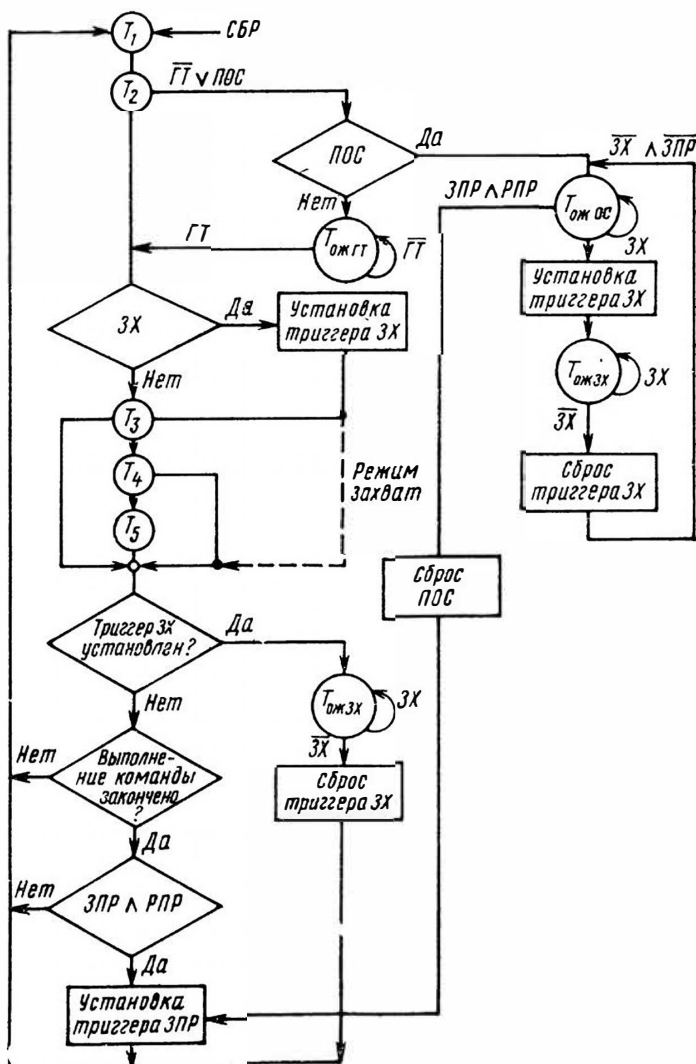


Рис. 1.3. Диаграмма состояний при выполнении машинного цикла МП КР5801К80А

(В-С, D-E, H-L). Регистры W, Z программно недоступны, они используются для хранения промежуточных результатов при выполнении команд. Управляемый адресный регистр предназначен для приема информации из других регистров и передачи ее в буферный регистр адреса. В процессе передачи этот регистр может увеличивать или уменьшать на единицу передаваемую информацию, что упрощает адресацию. В блок регистров входят также 16-разрядный регистр — указатель стека (УС) и 16-разрядный счетчик команд (СК).

Машинный цикл работы МП (рис. 1.3) включает от трех до пяти тактов и соответствует чтению или записи в ЗУ или во внеш-

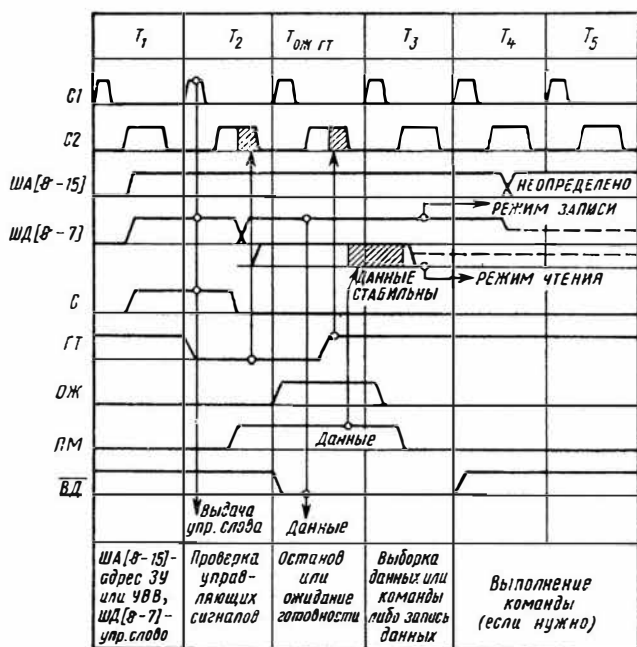


Рис. 1.4. Временные диаграммы командного цикла микропроцессора КР5801К80А

нее устройство одного байта информации. Команды МП (см. приложение 1) выполняются за несколько машинных циклов в зависимости от длины команды (1—3 байта) и реализуемых операций.

При выполнении команды (рис. 1.4) в такте T_1 содержимое счетчика команд выдается на шину адреса ША [8—15]. В этом же такте МП выдает на шину данных ШД [8—7] 8-разрядное управляющее слово (табл. 1.2) и формирует специальный сигнал С (Синхронизация) (табл. 1.3), по которому это слово записывается в буферный регистр внешних схем управления. Управляющее слово ука-

зывает тип выполняемого машинного цикла. В такте T_2 содержимое счетчика команд увеличивается на единицу и осуществляется анализ управляющих сигналов ГТ и ЗХ и сигнала ПОС управляющего слова. В зависимости от значений этих сигналов производится либо переход к такту T_3 машинного цикла, либо переход в одно из состояний ожидания $T_{ож}$, $T_{ож ос}$, $T_{ож зх}$, выход из которых возможен только при поступлении внешних управляющих сигналов. В такте T_3 данные принимаются из ЗУ или внешнего устройства на шину данных и записываются в регистр команд (если выполняется машинный цикл выборки команды) или в один из регистров МП. В тактах T_4 и T_5 , если они необходимы, выполняются действия над операндами. Если команда включает несколько циклов, то по завершении текущего машинного цикла процессор переходит к такту T_1 следующего цикла.

В последнем такте последнего цикла команды, а также в состоянии $T_{ож ос}$ проверяется уровень сигнала на входе ЗПР микропроцессора. При высоком уровне этого сигнала и установленном ранее триггере РАЗРЕШЕНИЕ ПРЕРЫВАНИЯ (РПР) МП по тактовому импульсу С2 (рис. 1.5) устанавливается внутренний триггер ПРЕРЫВАНИЕ (ТПР). Следующий цикл будет циклом ПРЕРЫВАНИЕ. В такте T_1 этого цикла содержимое счетчика команд не увеличивается на единицу, а сохраняется и далее записывается в стек. В такте T_3 устройство, запросившее прерывание, подает на шину данных код однобайтовой команды RST N (Переход к подпрограмме по адресу N*8). Число N может принимать значение от

Таблица 1.2

Значение разрядов управляющего слова МП КР580ИК80А

Разряд управляющего слова	Обозначение	Наименование сигнала	Название цикла									
			Выборка команды	Чтение из памяти	Запись в память	Чтение из стека	Запись в стек	Ввод по каналу ввода	Вывод по каналу вывода	Прерывание	Останов	Прерывание во время останова
ШД [0]	ППР	Подтверждение прерывания	0	0	0	0	0	0	0	1	0	1
ШД [1]	$\overline{ЗВ}$	Запись или вывод	1	1	0	1	0	1	0	1	1	1
ШД [2]	СТК	Операции со стеком	0	0	0	1	1	0	0	0	0	0
ШД [3]	ПОС	Подтверждение останова	0	0	0	0	0	0	0	0	1	1
ШД [4]	ВЫВ	Вывод	0	0	0	0	0	0	1	0	0	0
ШД [5]	Ц1	Первый цикл команды	1	0	0	0	0	0	0	1	0	1
ШД [6]	ВВД	Ввод	0	0	0	0	0	1	0	0	0	0
ШД [7]	ЧТ	Чтение из памяти	1	1	0	1	0	0	0	0	1	0

Управляющие сигналы МП КР580ИК80А

Обозначение	Наименование	Функциональное назначение и условия возникновения
<i>Сигналы управления шинами</i>		
ПМ	Разрешение приема информации с шины данных системы (выходной)	Шина данных ШД [0 — 7] МП находится в режиме приема информации
ВД	Признак выдачи информации на шину данных (выходной)	На шину данных ШД [0 — 7] МП выдана информация для записи в ЗУ или во внешнее устройство
<i>Сигналы управления прямым доступом к памяти</i>		
ЗХ	Захват шин (входной)	Запрос на прямой доступ к памяти со стороны внешнего устройства
ПЗХ	Подтверждение захвата шин (выходной)	Признак допуска внешнего устройства к шинам данных и адреса системы. Выходы шин ШД [0 — 7], ША [0 — 15] МП в состоянии с высоким выходным сопротивлением
<i>Сигналы управления прерываниями</i>		
ЗПР	Запрос прерывания (входной)	Запрос прерывания работы процессора. Проверяется в состояниях ожидания во время останова, а также в конце выполнения команды, если установлен триггер разрешения прерывания
РПР	Разрешение прерывания (выходной)	Разрешение прерывания. Соответствует состоянию внутреннего триггера разрешения прерывания
<i>Сигналы управления асинхронной передачей данных</i>		
ГТ	Готовность данных к передаче (входной)	Информация для обмена между ЗУ или внешним устройством и МП готова. При отсутствии сигнала готовности ГТ микропроцессор переходит в состояние ожидания, выходит из него при поступлении сигнала ГТ
ОЖ	Ожидание (выходной)	Микропроцессор находится в состоянии ожидания

Обозначение	Наименование	Функциональное назначение и условия возникновения
<i>Сигналы общего назначения</i>		
С	Синхронизация (выходной)	Начало машинного цикла. По этому сигналу на шину данных ШД [8—7] микропроцессор выдает управляющее слово
СБР	Сброс (входной)	Сигнал устанавливает в нуль счетчик команд, сбрасывает триггеры разрешения прерывания и захвата шин
С1, С2	Тактовые импульсы (входные)	Постоянно подаются на МП, обеспечивают тактирование

Ø до 7 и определяет начальный адрес одной из восьми возможных подпрограмм обработки прерываний. Два цикла ЗАПИСЬ В СТЕК, следующие за циклом ПРЕРЫВАНИЕ, осуществляют запись в стек текущего содержимого счетчика команд для последующего возврата к прерванной программе.

Микропроцессор реагирует на сигнал запроса прерывания ЗПР только при установленном в единицу внутреннем триггере разрешения прерывания. Его установка осуществляется с помощью команды EI (РАЗРЕШИТЬ ПРЕРЫВАНИЕ), а сброс может осуществляться командой DI (ЗАПРЕТИТЬ ПРЕРЫВАНИЕ). При выполнении

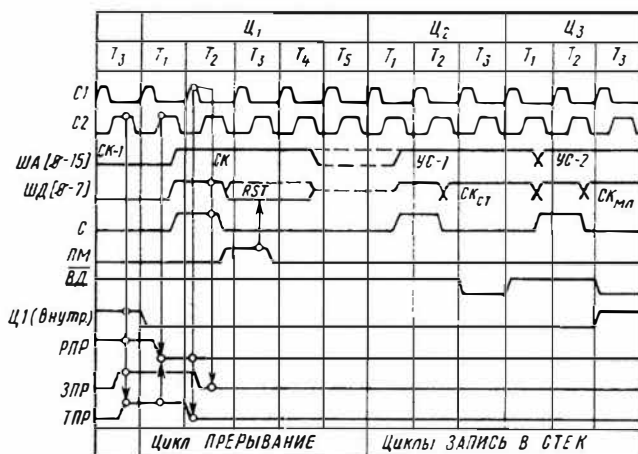
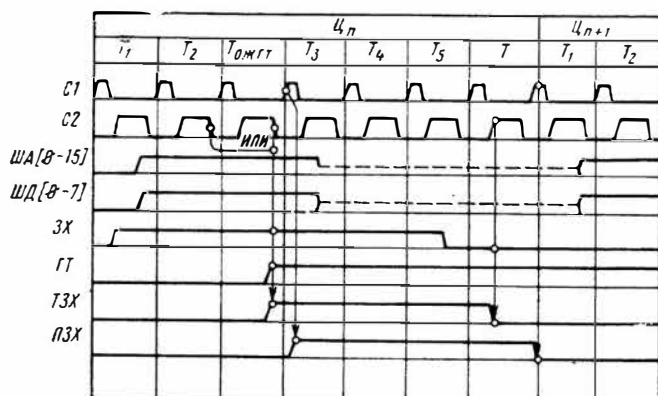
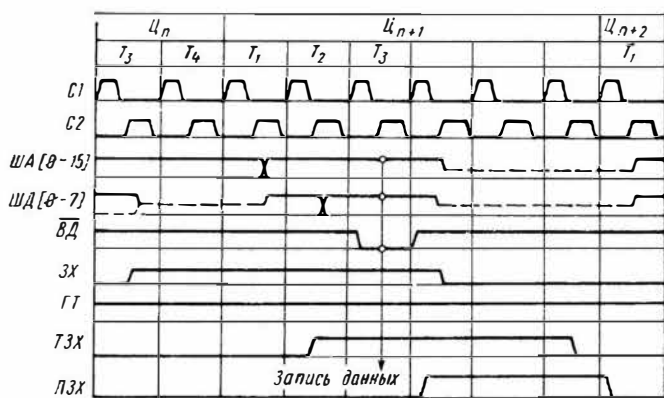


Рис. 1.5. Временные диаграммы процедуры предоставления прерывания МП КР5801К80А



а)



б)

Рис. 1.6. Временные диаграммы процедуры предоставления прямого доступа к памяти МП КР580ЙК80А:

а — для циклов ЧТЕНИЕ ИЗ ПАМЯТИ; б — для остальных циклов

команды RST, код которой поступает с внешнего устройства, вызвавшего прерывание, триггер разрешения прерывания сбрасывается в нуль. Для того чтобы МП вновь реагировал на сигнал ЗПР, необходимо в подпрограмме обслуживания прерывания выполнить команду EI.

Для непосредственного обмена информацией между ЗУ и внешним устройством используется режим прямого доступа к памяти. Его инициализация осуществляется асинхронным сигналом захвата шин (ЗХ), поступающим на МП от внешнего устройства. При этом

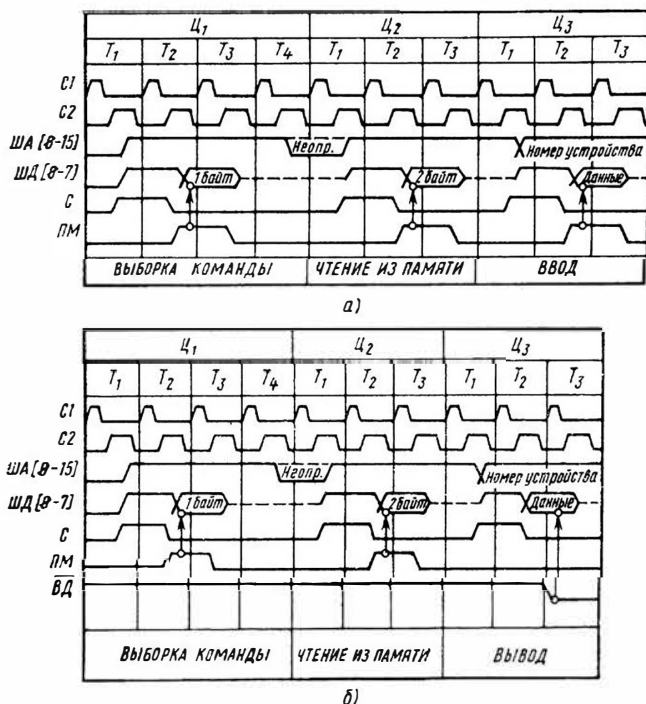


Рис. 1.7. Временные диаграммы выполнения команд ввода и вывода МП КР580ИК80А:

а — команда IN (ВВОД); б — команда OUT (ВЫВОД)

МП устанавливает свои выходы адресов и данных в состояние с высоким выходным сопротивлением, вырабатывает сигнал подтверждения захвата (ПЗХ), завершает выполнение внутренних операций текущего цикла и переходит в состояние $T_{ож\ зх}$.

В зависимости от типа машинного цикла, выполняемого во время поступления сигнала ЗХ, выработка сигнала ПЗХ осуществляет-

ся по-разному. Временная диаграмма процедуры предоставления прямого доступа к памяти во время исполнения машинных циклов ЧТЕНИЕ ИЗ ПАМЯТИ или ВВОД ПО КАНАЛУ ВВОДА приведена на рис. 1.6,а, аналогичная диаграмма для всех остальных циклов — на рис. 1.6,б. Выход МП из режима предоставления прямого доступа к памяти осуществляется в порядке, обратном описанному выше.

Ввод и вывод информации в МП осуществляется командами IN (ВВОД) и OUT (ВЫВОД). Команда IN предназначена для ввода байта данных из заданного канала ввода в накопитель, занимает два байта и требует для своей реализации трех циклов (рис. 1.7,а). В первом цикле осуществляется посылка адреса из счетчика команд в ЗУ, а по сигналу ПМ — прием первого байта команды в регистр команд. Во втором цикле в ЗУ посылается увеличенный на единицу адрес из счетчика команд и осуществляется прием второго байта команды в регистр временного хранения. Этот байт представляет собой номер внешнего устройства, к которому производится обращение. В третьем цикле этот номер в качестве адреса выводится на адресную шину, а по сигналу ПМ слово данных с канала ввода по шине данных записывается в накопитель.

При выполнении команды OUT (рис. 1.7,б) в третьем цикле формируется управляющее слово ВЫВОД ПО КАНАЛУ ВЫВОДА и по сигналу ВД осуществляется вывод информации на шину данных. На шину адреса выдается номер канала вывода. Если внешнее устройство работает с меньшей скоростью, чем МП, то в такте T_2 любого машинного цикла сигнал на входе ГТ имеет низкий уровень. В этом случае процессор после такта T_2 переходит в состояние $T_{ож ГТ}$, в котором и остается до получения от внешнего устройства сигнала ГТ. Состояние ожидания $T_{ож ГТ}$ имеет длительность, кратную периоду тактовых импульсов.

При включении питания содержимое регистров и счетчика команд МП устанавливается произвольным образом. Для запуска программы с нулевого адреса необходимо предварительно подать на вход сброса (СБР) микропроцессора сигнал высокого уровня, длительность которого должна быть не менее трех периодов тактовой частоты. Сигнал СБР осуществляет установку в нуль счетчика команд и сброс внутренних триггеров устройства управления. При этом сбрасываются и соответствующие выходные сигналы МП. Состояния регистров остаются неизменными, их содержимое будет неопределенным до тех пор, пока в них программно не будет записана та или иная информация. Выборка команд из нулевой ячейки памяти начинается сразу после снятия сигнала СБР.

1.2. БИС программируемого параллельного интерфейса КР580ВВ55

Микросхема КР580ВВ55 представляет универсальную программно-программируемую интерфейсную БИС, с помощью которой возможна организация обмена в параллельном формате практически с любым перифе-

рийным оборудованием. Она размещена в пластиковом корпусе с 40 выводами (рис. 1.8), потребляет мощность 350 мВт и требует напряжения питания 5 В. Микросхема может быть использована для организации синхронного и асинхронного обмена, обмена в режиме прерывания программы.

В состав БИС (рис. 1.9) входят три 8-разрядных канала (буферных регистра) ШКН1, ШКН2, ШКН3 и схема управления с регистром управляющего слова. Обмен информацией между каналами ввода — вывода и шиной данных микропроцессорной системы производится через двунаправленную шину ШД [$\emptyset$ —7] и буфер данных в соответствии с сигналами на входах $\overline{CT}$, $\overline{ЗП}$,

Таблица 1.4

Управляющие сигналы БИС КР580ВВ55

Обозначение	Наименование	Функциональное назначение и условия возникновения
$\overline{BM}$	Выбор микросхемы	Разрешение работы БИС. При отсутствии сигнала (высокий уровень) выводы шины данных ШД [$\emptyset$ —7] находятся в состоянии с высоким выходным сопротивлением
A [$\emptyset$], A [1]	Адрес, разряд $\emptyset$ Адрес, разряд 1	Обращение к одному из трех каналов или регистру управляющего слова
$\overline{CT}$	Чтение	Чтение данных из БИС КР580ВВ55 на шину данных ШД [$\emptyset$ —7]
$\overline{ЗП}$	Запись	Запись байта данных в БИС КР580ВВ55 с шины данных ШД [$\emptyset$ —7]
СБР	Сброс	Сброс БИС КР580ВВ55. При этом все регистры, включая регистр управляющего слова, устанавливаются в нулевое состояние. Все каналы переводятся в режим ввода

$\overline{BM}$, а также сигналами A [$\emptyset$], A [1], определяющими адрес (номер) канала ввода — вывода (табл. 1.4).

Перед началом работы программным способом путем передачи в БИС специальных команд инициализации назначаются режимы работы каждого из каналов.

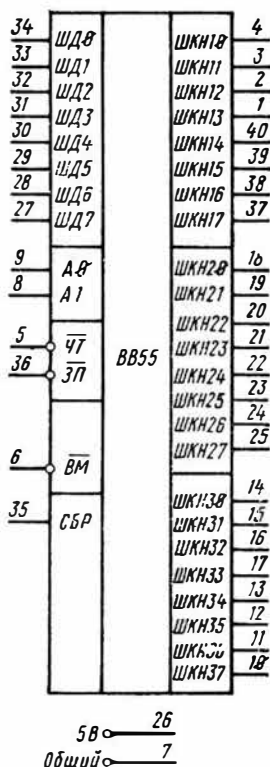


Рис. 1.8. Нумерация выводов БИС КР580ВВ55

Последующее обращение к каналам ввода — вывода осуществляется командами IN (ВВОД) и OUT (ВЫВОД). Однако возможно построение микропроцессорных систем, в которых обращение к каналам производится как к ячейкам памяти. В случае обращения по команде ввода —

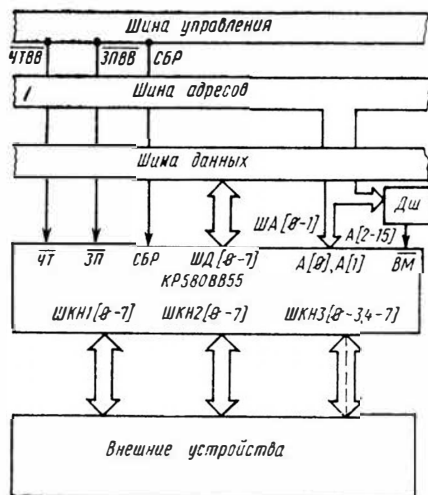


Рис. 1.9. Схема включения БИС программируемого параллельного интерфейса КР580ВВ55

вывода при наличии на старших разрядах ША [2—15] шины адреса нужного сочетания сигналов на входе ВМ БИС КР580ВВ55 с помощью внешнего адресного дешифратора (ДШ) формируется сигнал низкого уровня. Сигналы младших разрядов (А[0], А[1]) шины адреса поступают на БИС непосредственно и адресуют регистры каналов ШКН1, ШКН2, ШКН3 или регистр управляющего слова (табл. 1.5).

В процессе работы при необходимости изменения структуры ввода — вывода в управляющий регистр рассматриваемой БИС программным способом заносится соответствующее управляющее слово (рис. 1.10,а). Со-

держимое этого слова задает один из трех возможных режимов работы каналов ввода — вывода. основного (режим $\emptyset$); стробируемого однонаправленного (режим 1), стробируемого двунаправленного (режим 2). Хотя задание режимов для каналов ШКН1, ШКН2 осуществляется одной командой назначения режима, эти каналы могут одновременно работать в различных режимах. При изменении режима содержимое регистров каналов и триггеров регистра состояния устанавливается в нуль.

Таблица 1.5

Зависимость вида и направления передачи информации от управляющих сигналов БИС КР580ВВ55

Сигнал на входах					Вид передаваемой информации	Направление передачи информации
A [$\emptyset$]	A [1]	$\overline{3П}$	$\overline{ЧТ}$	$\overline{ВМ}$		
0	0	1	0	0	Данные	ШКН1 → ШД [$\emptyset$ —7]
1	0	1	0	0	"	ШКН2 → ШД [$\emptyset$ —7]
0	1	1	0	0	"	ШКН3 → ШД [$\emptyset$ —7]
0	0	0	1	0	"	ШД [$\emptyset$ —7] → ШКН1
1	0	0	1	0	"	ШД [$\emptyset$ —7] → ШКН2
0	1	0	1	0	"	ШД [$\emptyset$ —7] → ШКН3
1	1	0	1	0	Управляющее слово	ШД [$\emptyset$ —7] → Регистр управляющего слова
—	—	—	—	1	—	Нет передачи

При работе в режиме $\emptyset$ информация, поступающая на БИС КР580ВВ55 с ШД, запоминается в буферном регистре соответствующего канала и сразу же передается на его выходы. При вводе информации данные с каналов ШКН1—ШКН3 записываются в буферные регистры этих каналов без стробирования и далее при поступлении сигнала $\overline{ЧТ}$ (рис. 1.11,а) передаются на ШД. Обычно этот режим используется для ввода и вывода медленно изменяющихся данных или каких-нибудь постоянных значений.

В режиме $\emptyset$ направление передачи определяется отдельно для каждого из каналов.

Режимы работы каналов ШКН1 и ШКН2 могут быть заданы независимо друг от друга. Канал ШКН3

делится на две части Р7—Р4 и Р3—Р0, режимы работы которых определяются соответственно режимами работы каналов ШКН1 и ШКН2, если последние работают в режиме 1 или 2. Если каналы ШКН1 и ШКН2 работают в режиме 0, каждая половина канала ШКН3

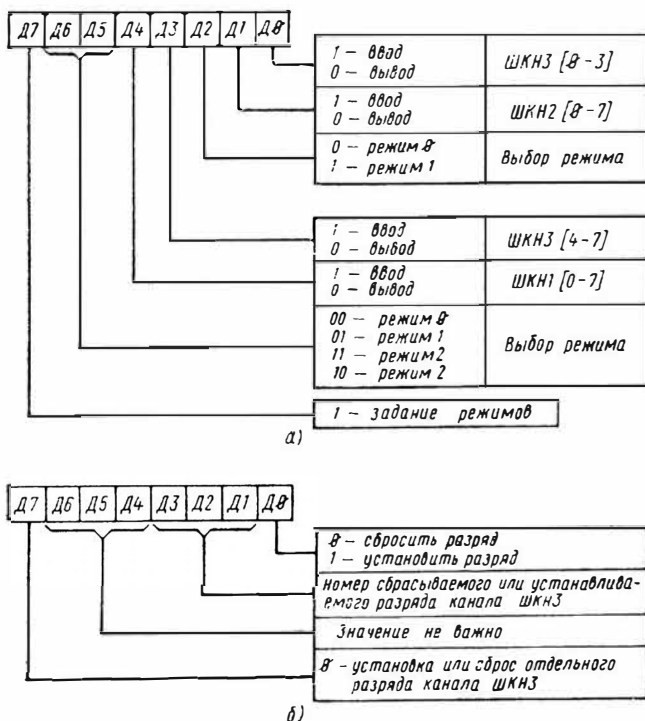


Рис. 1.10. Управляющее слово:

а — задающее режимы работы каналов ввода — вывода БИС КР580ВВ55; б — устанавливающее отдельные разряды канала ШКН3

может работать как отдельный канал в режиме 0. Режим 0 позволяет организовать синхронный обмен информацией между МП и внешними устройствами.

Режим 1 обеспечивает возможность организации однонаправленной асинхронной передачи информации

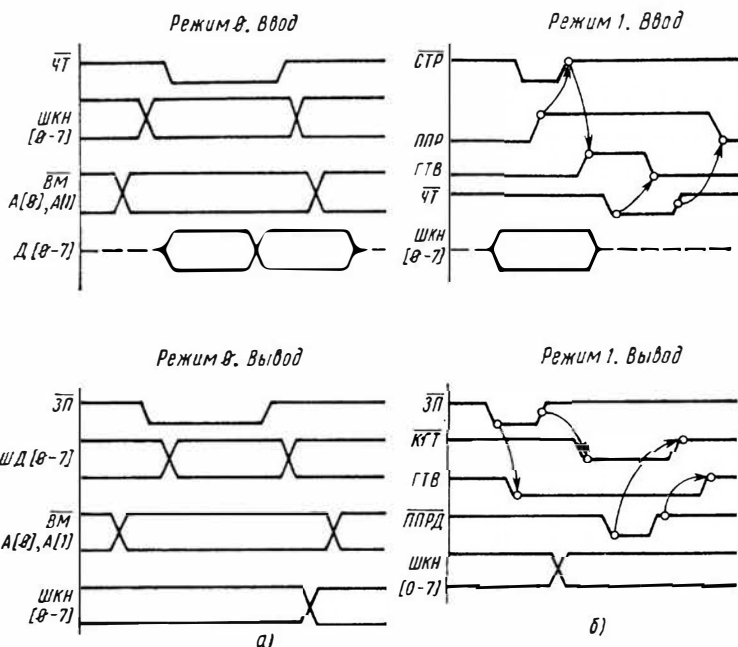
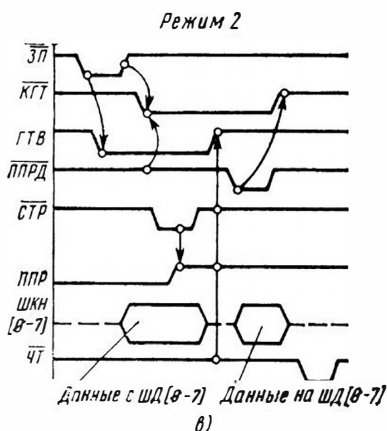


Рис. 1.11. Временные диаграммы работы БИС КР580ВВ55:

а — режим 0; б — режим 1; в — режим 2

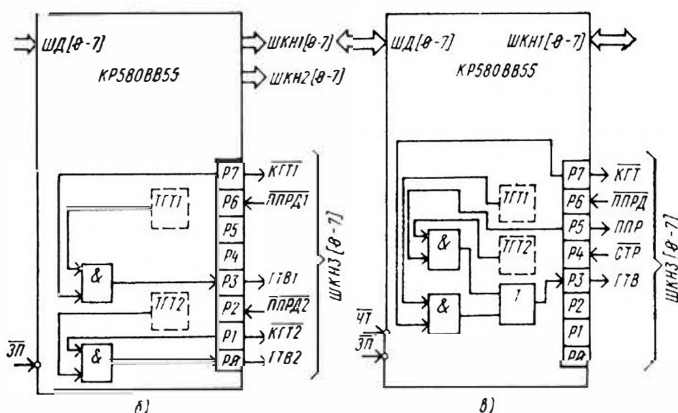
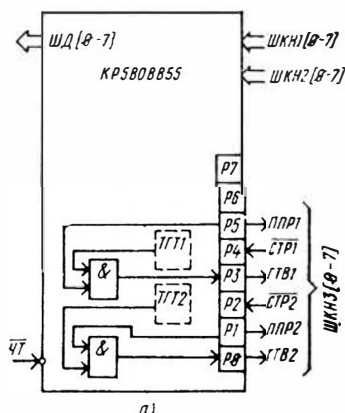


ных в регистры каналов ШКН1 и ШКН2). Для этого в состав канала ШКН3 входят два триггера (ТГТ1 и ТГТ2) разрешения готовности (рис. 1.12,а, б), допускающие установку и сброс, программным способом, а также схемы,

которые осуществляют асинхронную запись информации в регистры каналов ШКН1 и ШКН2 из внешнего устройства (либо наоборот) и формируют в зависимости от состояния ТГТ1 и ТГТ2 сигнал запроса на прерывание. Особенностью канала ШКН3 является возможность установки и сброса любого из его разрядов программным способом. Такая возможность существует для всех режимов работы канала, кроме нулевого, и осуществляется загрузкой в БИС КР580ИК55 кода команды установки/сброса разрядов (рис. 1.10,б) во время выполнения процессором команды OUT. Таким образом, каждый из разрядов канала ШКН3 может быть установлен либо сброшен независимо от остальных, что позволяет использовать канал ШК3 для хранения управляющего слова-состояния внешнего устройства при организации программно-управляемого обмена. Отметим, что при работе каналов ШКН1 или ШКН2 в режи-

Рис. 1.12. Схема реализации режимов передачи данных БИС КР580ВВ55:

а — режим 1 (ввод); б — режим 1 (вывод); в — режим 2



ме 1 или 2 каждый из разрядов канала ШКНЗ имеет строго определенное функциональное назначение и мнемоническое обозначение (табл. 1.6).

Таблица 1.6

Использование разрядов канала ШКНЗ для передачи управляющих сигналов

Разряд канала ШКНЗ	Режим 1 (Ввод)	Режим 1 (Вывод)	Режим 2	Разряд канала ШКНЗ	Режим 1 (Ввод)	Режим 1 (Вывод)	Режим 2
Разряд 0	ГТВ2	ГТВ2	—	Разряд 4	СТР1	—	СТР
Разряд 1	ППР2	КГТ2	—	Разряд 5	ППР1	—	ППР
Разряд 2	СТР2	ППРЛ2	—	Разряд 6	—	ППРЛ1	ППРЛ
Разряд 3	ГТВ1	ГТВ1	ГТВ	Разряд 7	—	КГТ1	КГТ

Рассмотрим реализацию процедуры асинхронного обмена между МП и внешним устройством с помощью БИС КР580ВВ55, работающей в режиме 1. Предположим, что обмен данными осуществляется через канал ШКН1. Для ввода информации канал должен работать в режиме 1 (Ввод) (рис. 1.11,б, 1.12,а). Внешнее устройство, будучи готовым к вводу информации в МП, формирует сигнал $\overline{\text{СТР1}}$ (Стробирование), по срезу которого осуществляется запись слова данных в канал ШКН1. При этом БИС КР580ВВ55 формирует сигнал ППР1 (Подтверждение приема). При высоком уровне сигнала ППР1 внешнему устройству запрещается формировать сигнал $\overline{\text{СТР1}}$ и записывать в канал новое слово данных. По фронту сигнала $\overline{\text{СТР1}}$, если сигнал ППР1 равен единице и установлен в единицу внутренний ТГТ1, формируется сигнал запроса на прерывание ГТВ1 (Готовность), посылаемый на МП. Получив этот сигнал МП переходит на соответствующую подпрограмму обслуживания прерываний, в процессе которой осуществляет считывание слова данных из канала ШКН1. Во время считывания на БИС КР580ВВ55 подается сигнал $\overline{\text{ЧТ}}$, по срезу которого слово данных передается из канала ШКН1 на шину команд/данных МП. По фронту сигнала $\overline{\text{ЧТ}}$ сбрасывается сигнал ППР1, что свидетельствует об окончании ввода данных в МП и разрешает внешнему устройству формировать новый сигнал $\overline{\text{СТР1}}$.

Отметим, что установка (сброс) ТГТ1 и ТГТ2 в ре-

жиме 1 (Ввод) осуществляется командой установки/сброса разрядов Р4 и Р2 канала ШКНЗ соответственно.

В режиме (Вывод) осуществляется запись слова данных из МП в регистры канала ШКН1 или ШКН2 (рис. 1.11,б, 1.12,б). Во время записи слова, например в канал ШКН1, микропроцессор подает на БИС КР580ВВ55 сигнал ЗП, по срезу которого снимается сигнал запроса на прерывание ГТВ1. По фронту сигнала ЗП/БИС КР580ВВ55 подает на внешнее устройство сигнал готовности канала к обмену КГТ1, подтверждая, что данные для выдачи записаны в регистре канала ШКН1. После получения этого сигнала внешнее устройство принимает данные из канала ШКН1 и формирует сигнал ППРД1 (Подтверждение передачи). По срезу сигнала ППРД1 снимается сигнал КГТ1. Если прерывания от данного внешнего устройства разрешены, т. е. предварительно программным способом установлен ТГТ1, то после приема информации внешним устройством (по фронту сигнала ППРД1) БИС КР580ВВ55 формирует для МП сигнал запроса на прерывание КГТ1, МП выполняет соответствующую подпрограмму обслуживания и т. д. Программная установка и сброс ТГТ1 и ТГТ2 осуществляются командой установки/сброса разрядов Р6 и Р2 канала ШКНЗ.

Используя режим 1 работы каналов, можно реализовать простой программно-управляемый обмен между МП и внешними устройствами. Для организации этой процедуры канал ШКНЗ должен служить регистром состояния внешнего устройства, каналы ШКН1 и ШКН2 — для передачи данных. Опрос и модификация содержимого регистра канала ШКНЗ могут осуществляться программным способом с помощью соответствующих команд МП и аппаратурным способом со стороны внешнего устройства.

В режиме 2 может работать только канал ШКН1, используемый в качестве буфера двунаправленной магистрали. Пять разрядов канала ШКНЗ применяются для формирования сигналов управления двунаправленной магистралью (рис. 1.11,в, 1.12,в). Процедура ввода либо вывода инициализируется сигналом запроса на прерывание ГТВ, который поступает из БИС КР580ВВ55 в МП. Микропроцессор в соответствии с

подпрограммой обработки запроса на прерывание начинает либо ввод, либо вывод информации из внешнего устройства. Вывод информации из МП начинается по срезу сигнала ЗП. По фронту сигнала ЗП БИС КР580ВВ55 формирует сигнал КГТ, свидетельствующий о том, что данные, предназначенные для выдачи, записаны в регистр канала ШКН1. В ответ на сигнал КГТ магистраль формирует сигнал ППРД. При активном сигнале ППРД БИС КР580ВВ55 выдает данные на двуправленную магистраль. Сигнал КГТ по срезу сигнала ППРД снимается. Сигнал запроса на прерывание ГТВ для МП формируется по фронту сигнала ППРД, если предварительно программно установлен ТГТ1, а сигналы КГТ и ЗП имеют единичный уровень. Сигнал ГТВ инициализирует выполнение микропроцессором очередного цикла обращения к двуправленной магистрали.

Ввод данных с двуправленной магистрали в МП осуществляется следующим образом. Когда магистраль готова к вводу информации в регистр канала ШКН1, она формирует сигнал СТР, по спаду которого осуществляется запись слова данных в регистр канала ШКН1. Микросхема КР580ВВ55 вырабатывает сигнал подтверждения приема ППР, который запрещает магистрали формировать сигнал СТР и записывать данные в канал. Сигнал запроса на прерывание ГТВ в МП выдается по фронту сигнала СТР при установленном ТГТ2 и наличии единичного уровня сигналов ЧТ и ППР. Получив сигнал запроса на прерывание, МП считывает информацию из канала ШКН1, формируя при этом сигнал ЧТ, по фронту которого сбрасывается сигнал ППР, сообщая магистрали об окончании цикла ввода информации. Программная установка/сброс ТГТ1 и ТГТ2 осуществляется командой установки/сброса разрядов Р6 и Р4 канала ШКН3 соответственно.

1.3. БИС программируемого последовательного интерфейса КР580ВВ51

Микросхема КР580ВВ51 представляет собой универсальное программируемое устройство преобразования и передачи данных и предназначена для организации

синхронного и асинхронного обмена в последовательном формате между микропроцессором (микропроцессорами) и периферийными устройствами. Универсальность БИС заключается в возможности программного изменения форматов передаваемых слов, режима и скорости передачи. Микросхема КР580ВВ51 размещена в пластиковом корпусе с 48 выводами (рис. 1.13), потребляет мощность 400 мВт, напряжения питания 5 В.

Для организации передачи данных в последовательном формате необходимы как минимум две БИС типа КР580ВВ51, одна из которых должна работать в режиме передачи и размещаться на передающем конце линии связи, вторая — работать в режиме приема и размещаться на приемном конце линии связи.

Режимы работы приемопередатчиков задаются теми устройствами (микропроцессорами, внешними устройствами), к шине данных которых приемопередатчики подключены. Задание режимов производится перед началом работы в процессе инициализации. Микросхема КР580ВВ51 позволяет передавать данные в двух основных режимах: синхронном и асинхронном.

В асинхронном режиме используется стартстопный принцип передачи. Каждое передаваемое слово передатчик сопровождает стартовым и стоповым битами, которыми осуществляется синхронизация запуска работы приемника (рис. 1.14, а).

В синхронном режиме данные передаются не по одному слову, а целыми массивами слов. Для синхронизации запуска приемника на прием массива используется не один бит, а одно или два слова — символы синхронизации (рис. 1.14, б). В синхронном режиме передатчик работает постоянно и выдает в линию данные в последовательном коде. При отсутствии (даже временном) данных передатчик непрерывно выдает в линию символы синхронизации. Приемник перед началом работы находится в режиме активного ожидания. В это время он побитно принимает поступающую информацию и осуществляет ее проверку на совпадение с кодом синхронизации. Совпадение свидетельствует о начале информационного массива. Сигналом совпадения синхронизируется запуск работы приемника, который начинает прием данных и их преобразование в параллельный формат. Один передатчик может работать на несколько приемников. Символом синхронизации осу-

Рис. 1.13. Нумерация выводов БИС КР580ВВ51

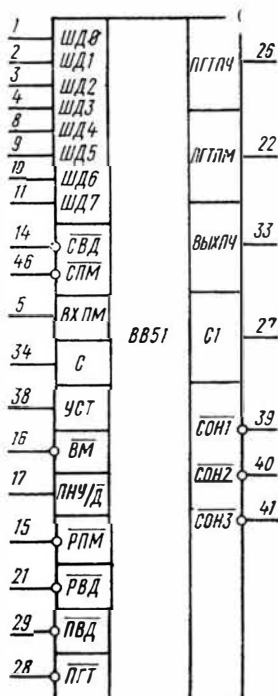
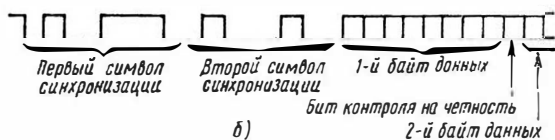
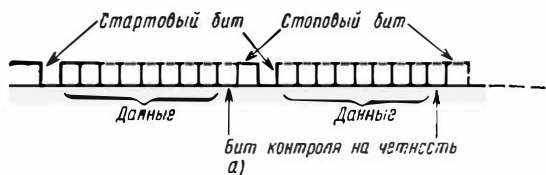
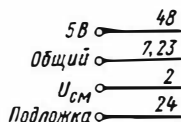


Рис. 1.14. Принцип передачи информации:

а — при асинхронной передаче; б — при синхронной передаче



ществляется адресация того или иного приемника. Новые символы синхронизации, вставляемые передатчиком в поток данных, позволяют повторно осуществлять процедуру синхронизации.

С точки зрения скорости передачи данных для информационных массивов малой размерности (менее 100 бит) более выгодным является асинхронный режим,

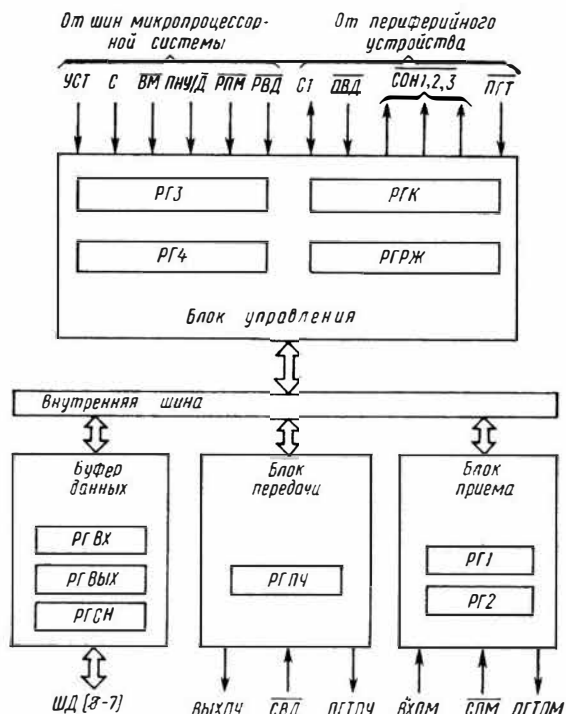


Рис. 1.15. Структура программируемого последовательного интерфейса связи КР580BV51

а для массивов большой размерности — синхронный. Первый обеспечивает скорость передачи до 9,6К бит/с, второй — до 56К бит/с. Для обоих режимов длина передаваемых символов является программируемой и может составлять 5, 6, 7 или 8 бит. При длине символа менее 8 бит неиспользуемые биты передаваемого байта заполняются нулями. Информация может выдаваться и при-

ниматься как с контролем на четность (нечетность), так и без него.

Функционально БИС КР580ВВ51 (рис. 1.15) состоит из блоков приема и передачи информации в последовательном коде, буфера данных и блока управления, соединенных внутренней шиной. Буфер данных обеспечивает прием и выдачу данных и команд в параллельном 8-разрядном коде через двунаправленные линии ШД $\overline{\text{О}}$ —ШД7. При отсутствии низкого уровня входного сигнала $\overline{\text{ВМ}}$ (Выбор микросхемы) (табл. 1.7) линии

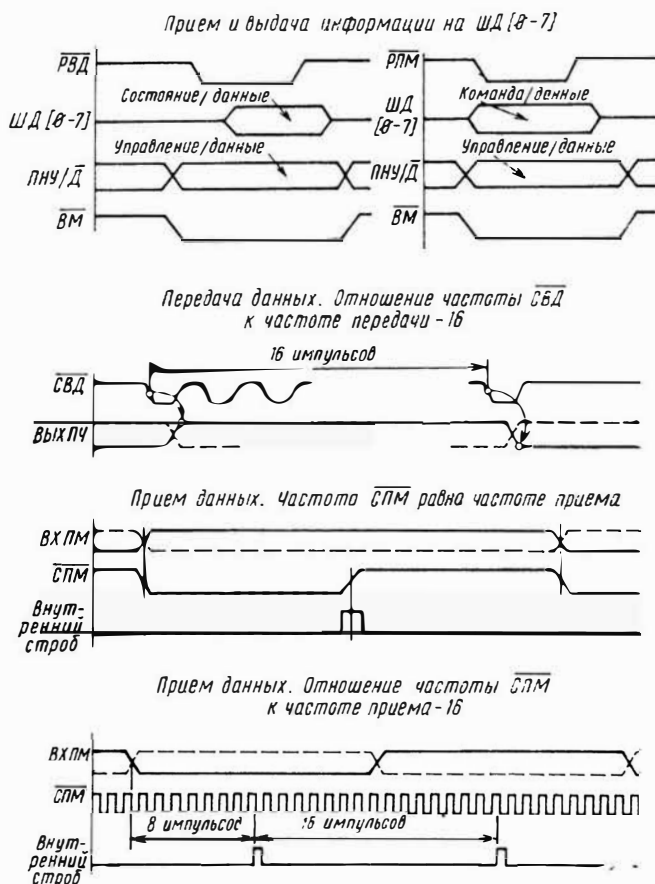


Рис. 1.16. Временные диаграммы работы БИС КР580ВВ51

ШДØ—ШД7 (рис. 1.16) находятся в состоянии с высоким выходным сопротивлением. Буфер данных включает регистр входной (РГВХ), регистр выходной (РГВЫХ) и регистр состояния (РГСН) (см. рис. 1.15).

Блок передачи обеспечивает преобразование параллельного 8-разрядного кода, поступившего из МП, в последовательный, добавление бита четности и, если нужно, генерацию стартового и стоповых битов. Выдача информации в последовательном коде осуществляется по линии выхода передатчика (ВЫХ ПЧ) путем сдвига данных в регистре передачи (РГПЧ) под действием синхроимпульсов СВД.

Блок приема предназначен для преобразования данных, поступающих в последовательном коде на вход приемника (ВХПМ), в параллельный 8-разрядный код. Это преобразование осуществляется в РГ1 и РГ2 под действием синхроимпульсов СПМ.

Блок управления обеспечивает прием и хранение управляющего слова в регистре режима (РГРЖ) и команды в регистре команды (РГК). Управляющее слово и команда, выдаваемые в БИС КР580ВВ51 микропроцессором, определяют функционирование устройства преобразования и передачи данных. В регистры РГ3 и РГ4 при работе в синхронном режиме микропроцессором записываются коды первого и второго символов синхронизации.

Микропроцессор обычно обращается к БИС КР580ВВ51 как к каналу ввода — вывода, однако может быть организовано обращение к этой БИС как к ячейке ЗУ. Перед началом работы устройство преобразования и передачи данных КР580ВВ51 должно быть установлено в исходное состояние сигналом установки (УСТ). Программирование режима приема или передачи данных осуществляется подачей на БИС управляющего слова (рис. 1.17) при единичном значении сигнала ПНУ/Д. Если задан синхронный режим работы, то вслед за управляющим словом подаются коды одного или двух символов синхронизации (рис. 1.18).

В процессе передачи данных на БИС КР580ВВ51 подаются команды (рис. 1.19) для оперативного указания конкретных операций: приема, передачи, перехода в режим активного ожидания. В частности, команда может быть использована для установки БИС в исход-

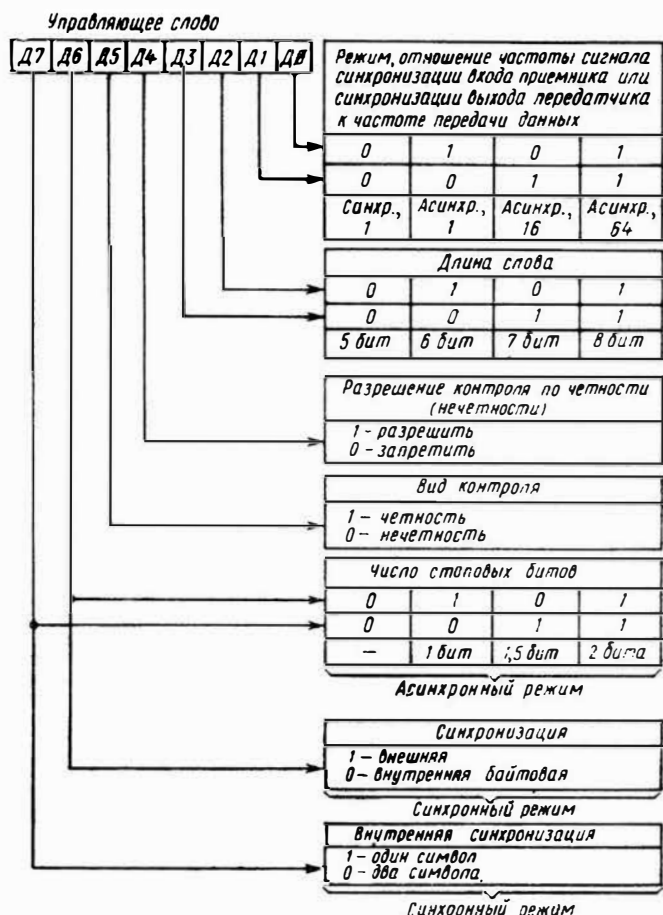


Рис. 1.17. Формат управляющего слова БИС КР580ВВ51

ное состояние и подготовки ее к приему управляющего слова.

Приемопередатчик КР580ВВ51 позволяет осуществлять как программно-управляемый обмен между МП и внешними устройствами, так и обмен в режиме прерывания программ. Для этого информация о состоянии приемного и передающего буферов выводится на внешние выводы БИС и может быть использована для инициализации прерываний МП и обмена в режиме прерыв-

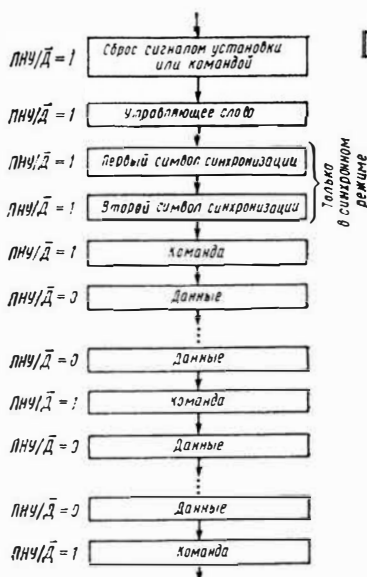


Рис. 1.18. Структура программного блока, управляющего работой БИС КР580BB51

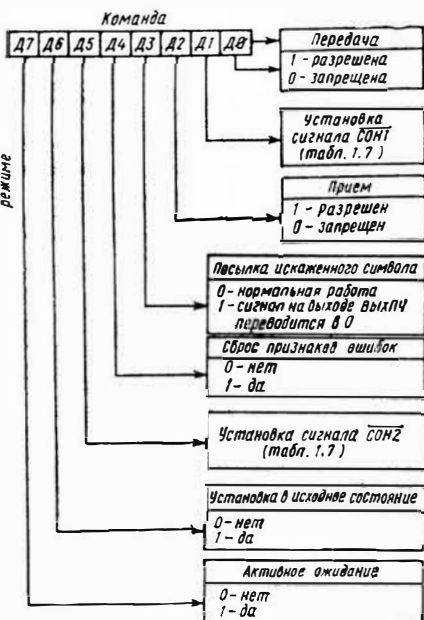


Рис. 1.19. Формат команды БИС КР580BB51

ваний. Для обеспечения программно-управляемого обмена предусмотрена возможность программного считывания слова-состояния БИС (рис. 1.20). Это слово содержит информацию о текущем состоянии приемного и передающего буферов, наличии ошибок приема и передачи. Считывание слова-состояния производится при единичном уровне сигнала на входе ПНУ/Д̄.

Работа БИС КР580BB51 в асинхронном режиме осуществляется со скоростью, равной частоте сигналов на входах СВД и СПМ или составлять 1/16 или 1/64 этой частоты. Скорость приема и передачи задается программным способом. При отсутствии данных передатчик выдает в линию высокий уровень напряжения, при их поступлении — низкий уровень. Длительность одной посылки (бита информации) определяется сигналами СВД (для передатчика) и СПМ (для приемника), которые должны быть настроены на одну частоту. Получив низ-

Управляющие сигналы БИС К580ИК51

Обозначение	Наименование	Функциональное назначение и условия возникновения
<i>Сигналы управления записью и считыванием со стороны микропроцессорной системы</i>		
УСТ	Установка (входной)	Установка в исходное состояние, БИС переходит в режим ожидания последовательности управляющих слов
С	Синхронизация (входной)	Тактовая частота. Обычно на вход С подается сигнал С2 от тактового генератора МП. При работе БИС в синхронном режиме частота сигнала С должна быть минимум в 30 раз выше частоты приема или передачи; при работе в асинхронном режиме — минимум в 5 раз
$\overline{BM}$	Выбор микросхемы (входной)	При высоком уровне сигнала буфер шины данных БИС находится в состоянии с высоким выходным сопротивлением, при низком уровне — подключается к шине данных микропроцессорной системы
ПНУ/ $\overline{D}$	Признак „Управление/Данные“ (входной)	Указывает тип вводимой в БИС с ШД информации (высокий уровень — ввод команды, низкий уровень — ввод данных). При выводе слова состояния БИС на этот вход подается высокий уровень сигнала ПНУ/ $\overline{D}$ (обычно на этот вход подается младший разряд адреса A[0])
$\overline{RPM}$	Разрешение на прием информации с ШД (входной)	Ввод команды или данных с ШД в БИС
$\overline{RVD}$	Разрешение на выдачу информации в ШД (входной)	Вывод данных или слова состояния на ШД

Обозначение	Наименование	Функциональное назначение и условия возникновения
ПГТПЧ	Признак готовности передатчика (выходной)	Передатчик готов к приему данных с ШД. Сигнал может быть использован как сигнал запроса на прерывание процессора. Состояние этого выхода может быть также опрошено программным способом посредством считывания слова состояния. Сигнал снимается после приема слова с ШД по срезу сигнала РПМ
ПГТПМ	Признак готовности приемника (выходной)	Приемник готов к выдаче принятого слова в параллельном коде на ШД. Может быть использован в качестве запроса на прерывание. Состояние этого сигнала может быть опрошено программным способом посредством считывания слова состояния

Сигналы управления приемом и передачей со стороны внешнего устройства

<u>СПМ</u>	Синхронизация входа приемника (входной)	Определяет скорость приема информации в последовательном коде. При синхронном режиме частота приема равна частоте сигнала. При асинхронном режиме частота приема задается программным способом и может составлять 1,1/16, 1/64 от частоты сигнала. Бит данных вводится в приемный буфер по срезу сигнала <u>СПМ</u>
<u>СВД</u>	Синхронизация выхода передатчика (входной)	Определяет скорость передачи информации в последовательном коде. При синхронном режиме частота передачи равна частоте сигнала, при асинхронном режиме частота передачи задается программным способом и может составлять 1,1/16, 1/64 от частоты сигнала. Каждый бит данных выводится из передающего буфера по фронту сигнала <u>СВД</u>
C1	Вид синхронизации	В синхронном режиме с внешней синхронизацией сигнал C1 является входным. Подача сигнала C1 на приемник инициирует ввод приемником слова в последовательном коде, начиная со следующего за C1 сигнала <u>СПМ</u> .

Обозначение	Наименование	Функциональное назначение и условия возникновения
		При работе с внутренней символьной синхронизацией сигнал C1 является выходным. Он устанавливается при приеме последнего бита символа синхронизации (если используются два символа синхронизации, то при приеме последнего бита второго символа синхронизации). При считывании слова состояния сигнал C1 сбрасывается. В асинхронном режиме сигнал C1 является выходным. Его единичное значение указывает на ошибку в принятых данных (на вход ВХПМ поступает постоянный низкий уровень). Значение сигнала C1 может быть опрошено программным способом посредством считывания слова состояния. Сброс сигнала C1, указывающего на ошибку, может быть осуществлен сигналом УСТ или подачей на вход ВХПМ сигнала высокого уровня
ПГТ	Признак готовности внешнего устройства принять данные (входной)	Низкий уровень сигнала инициирует передачу данных передатчиком на выход Вых ПЧ, если она предварительно разрешена соответствующей командой. Высокий уровень сигнала запрещает выдачу очередного слова данных, следующего за этим сигналом
СОН1	Сигнал общего назначения 1 (выходной)	Подтверждение готовности БИС КР580ВВ51 к обмену с периферийным устройством (модемом). Допускает установку и сброс программным способом
ПВД	Признак окончания выдачи данных (входной)	Внешнее устройство (например, модем) закончило обработки (выдачу в телефонную линию) данных и готово вновь принять данные с Вых ПЧ. Сигнал может быть опрошен программным способом посредством считывания слова состояния

Обозначение	Наименование	Функциональное назначение и условия возникновения
CON2	Сигнал общего назначения 2 (выходной)	Запрос на прием данных внешним устройством с выхода Вых ПЧ. Допускает установку и сброс программным способом
CON3	Сигнал общего назначения 3 (выходной)	Информация в передающем буфере отсутствует. Сбрасывается после приема данных от МП (по срезу сигнала РПМ). Состояние выхода может быть опрошено программным способом при считывании слова состояния БИС

кий уровень напряжения, приемник воспринимает его как стартовый бит, а последующие восемь посылок (или девять, если используется контроль на четность или нечетность) — как биты информации. После передачи байта информации передатчик передает, а приемник принимает стоповый бит, представленный высоким уровнем напряжения. Одновременно производится контроль принятого слова данных на четность (нечетность), проверка количества стоповых бит. При обнаружении

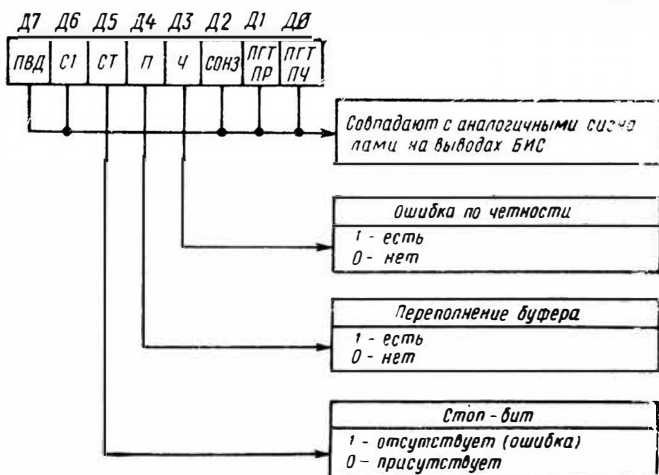


Рис. 1.20. Формат слова-состояния БИС КР580ВВ51

ошибки устанавливается соответствующий триггер ошибки.

После приема слова данных в приемный буфер сигнал ПГТПМ устанавливается в единичное состояние, подтверждая готовность слова данных для ввода в МП. Если МП не считывает из буфера это слово до начала

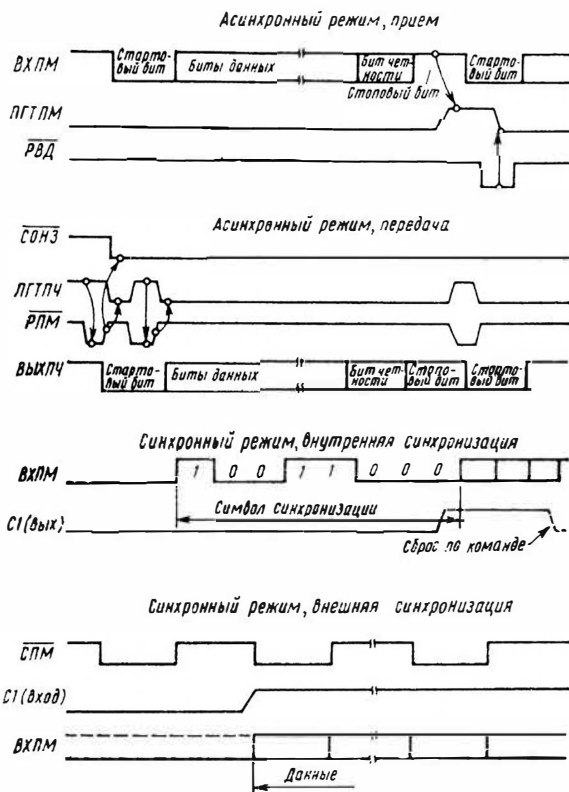


Рис. 1.21. Временные диаграммы синхронизации приема и передачи данных с использованием БИС КР580ВВ51

приема приемником следующего слова, автоматически устанавливается триггер ошибки по переполнению, а слово данных, находившееся в буфере, теряется. Сброс триггеров ошибок может быть осуществлен управляющей командой. Временные диаграммы формирования внешних управляющих сигналов во время приема и

передачи данных при асинхронном режиме работы приведены на рис. 1.21. На рис. 1.16 приведены диаграммы приема и передачи с различными скоростями.

При работе БИС КР580ВВ51 в синхронном режиме прием и передача данных возможны только на частоте, которая подается на входы СПМ и СВД приемника и передатчика. Если при инициализации задан синхронный режим с внутренней синхронизацией, приемопередатчик выдает на линию передачи символы синхронизации до тех пор, пока не будет загружен его выходной буфер. В это время внешний выходной сигнал СОНЗ установлен в единичное состояние. После загрузки слова данных из МП в выходной буфер приемопередатчика этот сигнал сбрасывается и начинается выдача слова данных на линию. Выдача очередного слова данных может быть запрещена подачей единичного сигнала на вход ПГТ передатчика.

Для задания работы приемника в режиме внутренней синхронизации вывод С1 приемника программируется управляющей командой на работу в качестве выхода. Затем управляющей командой (см. рис. 1.19) приемник переводится в режим активного ожидания и начинает работать в режиме поиска символов синхронизации. Обнаружив символ (символы) синхронизации, приемник последующую информацию начинает воспринимать как последовательность слов, формат которых заранее определен. При установлении синхронизации (во время поступления последнего бита символа синхронизации) устанавливается сигнал С1, который автоматически сбрасывается по команде считывания слова-состояния (см. рис. 1.21).

Для работы приемопередатчика в режиме с внешней синхронизацией назначение вывода С1 в БИС приемнике и БИС передатчике осуществляется по-разному. В первой этот вывод программируется на работу в качестве входа, а во второй — в качестве выхода. Вход С1 приемника и выход С1 передатчика соединяются. Синхронизация приема слов данных приемником осуществляется по фронту сигнала С1 от передатчика (см. рис. 1.21).

Типичным примером применения БИС КР580ВВ51 является ее использование для сопряжения микропроцессорной системы с модемом, обеспечивающим переда-

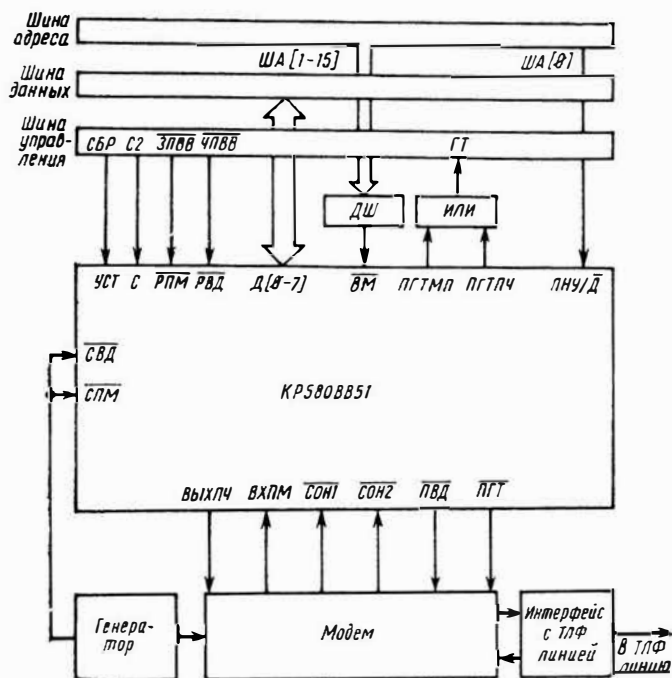


Рис. 1.22. Использование БИС КР580ВВ51 для сопряжения микро-процессорной системы с модемом

чу данных по телефонной линии. Представленный на рис. 1.22 вариант предусматривает обращение к БИС КР580ВВ51 как к ячейке ЗУ.

1.4. БИС программируемого контроллера прерываний КР580ВН59

Программируемый контроллер прерываний КР580ВН59 представляет собой законченное устройство, которое позволяет реализовывать 8-уровневую векторную систему прерываний с возможностью маскирования и динамического изменения дисциплины обслуживания. Для перехода к подпрограммам обслуживания прерываний контроллер формирует и подает на ШД процессо-

ра код команды CALL¹. Каскадированием БИС КР580ВН59 число обслуживаемых уровней прерывания может быть увеличено до 64. Контроллер может использоваться как для организации обмена информацией в режиме прерывания, так и для организации программно-управляемого обмена. В первом случае БИС КР580ВН59 на приоритетной основе формирует запрос на прерывание для МП и адрес подпрограммы обслужи-

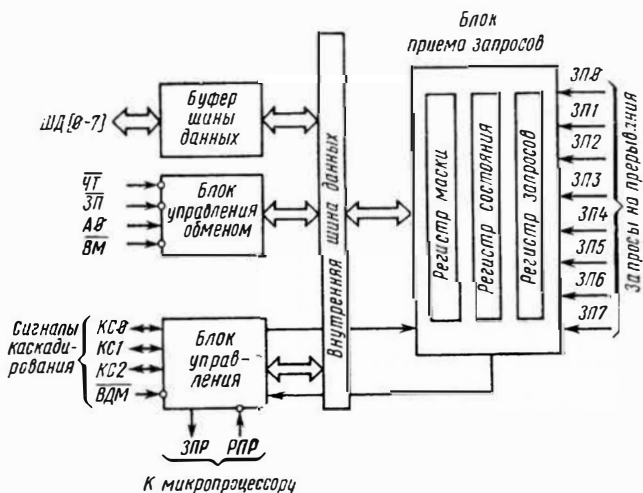


Рис. 1.23. Структура программируемого контроллера прерываний КР580ВН59

живания, во втором — процессор считыванием слова состояния контроллера определяет устройство с наивысшим приоритетом, готовое к обмену. Микросхема КР580ВН59 размещена в пластиковом корпусе с 28 выводами, требует напряжение питания 5 В и потребляет мощность 1 Вт.

Контроллер позволяет реализовать простой приоритетный режим и режим циклического приоритета обслуживания прерываний. При реализации простого приоритетного режима всем восьми входам запросов на прерывание присваиваются фиксированные приоритеты,

¹ В цикле прерывания вместо команды RESTART в МП может быть введена команда CALL, при которой счетчик команд перезагружается 16-разрядным адресом, указанным в команде CALL, а его текущее значение заносится в стек.

Таблица 1.8

Сигналы управления контроллера прерываний КР580ВН59

Обозначение	Наименование	Функциональное назначение и условия возникновения
<i>Сигналы управления записью/считыванием</i>		
$\overline{ЗП}$	Запись (входной)	Запись в БИС управляющих слов
$\overline{ЧТ}$	Чтение (входной)	Считывание из БИС содержимого внутренних регистров
$A\emptyset$	Адрес (входной)	Адресация регистров
$\overline{ВМ}$	Выбор микросхемы (входной)	Формируется при определенной комбинации сигналов на ША [1—15]. Осуществляет подключение ШД [0—7] БИС к шине данных системы. При отсутствии сигнала выводы ШД [0—7] БИС находятся в состоянии с высоким выходным сопротивлением
<i>Сигналы управления прерываниями</i>		
$ЗП\emptyset—ЗП7$	Запросы на прерывания (входные)	Запросы на прерывания от внешних устройств
$ЗПР$	Запрос на прерывание (выходной)	Запрос на прерывание, выдаваемый контроллером на МП
$\overline{РПР}$	Разрешение прерывания (входной)	После поступления этого сигнала от МП контроллер осуществляет ввод в МП команды CALL
<i>Системы управления каскадированием</i>		
$\overline{ВДМ}$	Ведомый (входной)	Высокий уровень, если контроллер является ведущим, низкий уровень—если ведомым
$КС\emptyset—КС2$	Каскадирование	Сигналы $КС\emptyset—КС2$ являются выходными, если контроллер является ведущим ($\overline{ВДМ}=1$), и входными, если—подчиненным ($\overline{ВДМ}=\emptyset$). Значения сигналов $КС\emptyset—КС2$ ведущего контроллера, подаваемые на входы $КС\emptyset—КС2$ подчиненных, указывают подчиненный контроллер, который формирует и выдает в МП адрес своей подпрограммы обслуживания

причем наивысший приоритет присваивается входу ЗПØ, наименьший — ЗП7. В режиме циклического приоритета после окончания обслуживания любого устройства приоритеты входов контроллера циклически изменяются таким образом, что устройству, обслуживаемому последним, присваивается низший приоритет. Кроме того, при работе в режиме циклического приоритета низший приоритет может быть присвоен программным способом любому входу запроса.

Запросы на прерывания от внешних устройств подаются на входы ЗПØ—ЗП7 контроллера (рис. 1.23, табл. 1.8) и запоминаются в регистре запросов. Регистр состояния, каждый разряд которого соответствует одному из входов ЗПØ—ЗП7, содержит все запросы на прерывания, обслуживаемые в текущий момент. Регистр маски содержит единицы в разрядах, соответствующих маскируемым в настоящий момент входам запросов. Запросы на прерывания по любому входу могут быть поданы в потенциальной или в импульсной форме. Однако каждый последующий запрос на прерывание воспринимается контроллером только после выполнения подпрограммы обслуживания текущего запроса по данному входу и сброса соответствующего разряда регистра состояния, что осуществляется программным способом (специальной командой). Установка в единицу того или иного разряда регистра маски блокирует передачу запроса на прерывание. Однако, если через некоторое время после подачи запроса на прерывание маска будет снята программой, запрос будет обслужен.

Программирование контроллера осуществляется двумя типами команд: командами инициализации и командами управления режимом. Ввод команд в контроллер прерываний осуществляется микропроцессором, как правило, командой OUT (ВЫВОД). Однако в системе может быть организовано обращение к контроллеру как к ячейкам ЗУ. Команды инициализации подаются перед началом работы контроллера. Эти команды задают стартовые адреса подпрограмм обслуживания прерываний, расстояния между соседними стартовыми адресами и указывают, если необходимо, на наличие других контроллеров в системе. Команды управления режимом служат для оперативного изменения режимов обслуживания прерываний и могут подаваться в любое время в процессе работы контроллера.

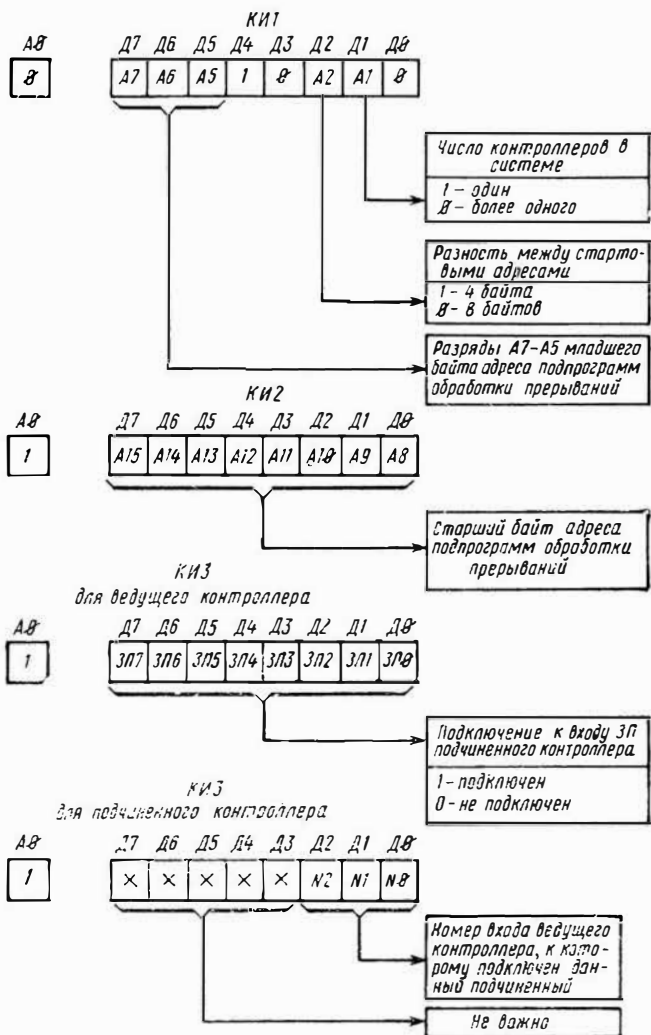


Рис. 1.24. Команды инициализации БИС КР580ВН59

Процедура инициализации заключается в последовательном вводе в контроллер команд инициализации (КИ1 и КИ2). Если система содержит несколько контроллеров прерываний, для каждого из них вслед за КИ1 и КИ2 вводится третья команда КИ3, которая

определяет соподчиненность контроллеров. Перед проведением инициализации прием прерываний микропроцессором должен быть запрещен с помощью команды DI (ЗАПРЕТИТЬ ПРЕРЫВАНИЕ).

Команда KI1 (рис. 1.24) осуществляет сброс всех разрядов регистра запросов на прерывания, очищает регистр маски, присваивает низший приоритет входу ЗП7, сбрасывает триггеры чтения состояния регистров и режима специального маскирования (см. ниже). Эта же команда определяет младшие байты адресов подпрограмм обслуживания прерываний (табл. 1.9), которые

Таблица 1.9

Младшие байты адресов подпрограмм для обслуживания прерываний

Вход	Расстояние между соседними стартовыми адресами равно 4 байт								Расстояние между соседними стартовыми адресами равно 8 байт							
	Д7	Д6	Д5	Д4	Д3	Д2	Д1	Д0	Д7	Д6	Д5	Д4	Д3	Д2	Д1	Д0
ЗП7	A7	A6	A5	1	1	1	0	0	A7	A6	1	1	1	0	0	0
ЗП6	A7	A6	A5	1	1	0	0	0	A7	A6	1	1	0	0	0	0
ЗП5	A7	A6	A5	1	0	1	0	0	A7	A6	1	0	1	0	0	0
ЗП4	A7	A6	A5	1	0	0	0	0	A7	A6	1	0	0	0	0	0
ЗП3	A7	A6	A5	0	1	1	0	0	A7	A6	0	1	1	0	0	0
ЗП2	A7	A6	A5	0	1	0	0	0	A7	A6	0	1	0	0	0	0
ЗП1	A7	A6	A5	0	0	1	0	0	A7	A6	0	0	1	0	0	0
ЗП0	A7	A6	A5	0	0	0	0	0	A7	A6	0	0	0	0	0	0

в цикле обработки прерываний выдаются на ШД [0—7] системы по второму сигналу $\overline{\text{РПР}}$ микропроцессора (по первому сигналу $\overline{\text{РПР}}$ микропроцессора контроллером прерываний на шину данных выводится код команды CALL). Команда KI2 вводится в контроллер при единичном значении сигнала A[0] и представляет собой старший байт адреса подпрограмм обслуживания прерываний. Этот байт выдается контроллером в цикле обработки прерываний в качестве третьего байта команды CALL на ШД [0—7] микропроцессорной системы. Выдача производится по третьему¹ сигналу $\overline{\text{РПР}}$ микропроцессора.

Управление режимами работы контроллеров, измене-

¹ Если в процессе обработки запроса на прерывание в МП введена команда CALL МП, то после ее ввода МП дважды формирует сигнал $\overline{\text{РПР}}$.

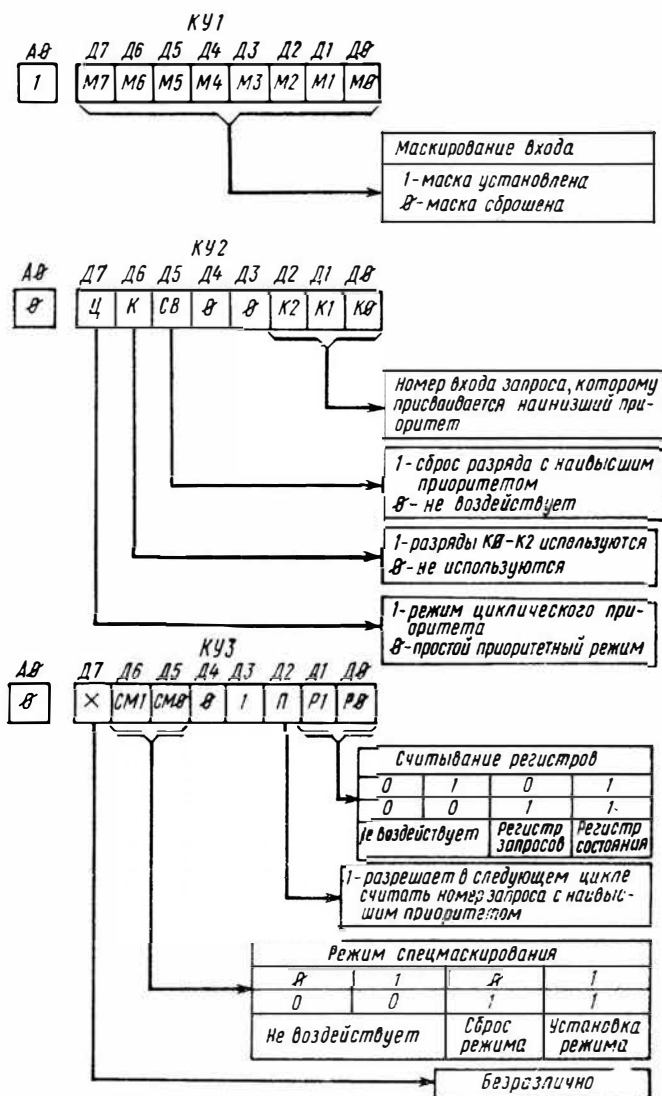


Рис. 1.25. Команды управления БИС КР580ВН59

ние дисциплины обслуживания, управление маскированием прерываний осуществляется командами управления режимами (КУ1, КУ2, КУ3, рис. 1.25).

Для задания простого приоритетного режима не требуется никаких дополнительных команд, кроме команд инициализации. После приема микропроцессором запроса на прерывание ЗПР и выдачи сигнала разрешения прерывания РПР триггер разрешения прерываний микропроцессора сбрасывается, запрещая тем самым прием прерываний. Поэтому для организации мно-



Рис. 1.26. Структура подпрограммы обслуживания прерывания при многоуровневой системе прерываний

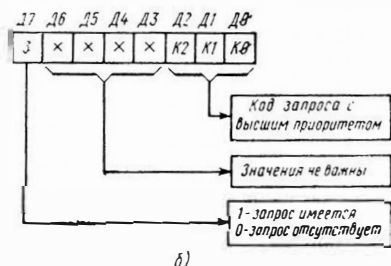
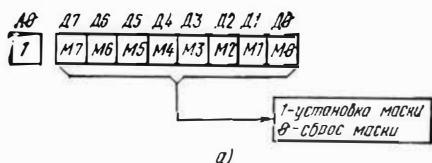


Рис. 1.27. Формат команды управления регистром маски (а) и слова считываемого с контроллера прерываний (б)

гоуровневой системы прерываний необходимо выполнить программную установку этого триггера командой EI (РАЗРЕШИТЬ ПРЕРЫВАНИЯ) МП. Об окончании выполнения подпрограммы обслуживания каждого запроса на прерывание МП должен сообщать контроллеру с помощью КУ2 (строка 1 или 2 табл. 1.10). При

этом осуществляется сброс либо указанного в команде разряда регистра состояния контроллера, либо разряда, соответствующего обслуженному прерыванию. Для реализации многоуровневой системы прерываний каждая из подпрограмм обслуживания должна иметь определенную структуру (рис. 1.26).

Таблица 1.10

Варианты использования команды КУ2 БИС КР580ВН59

Код команды	Выполняемые функции
0010 0000	Сброс разряда регистра состояния, имеющего наивысший приоритет среди разрядов, установленных в единицу
0110 0XXXX	Сброс разряда регистра состояния, номер которого указан в трех младших разрядах команды, содержимое регистра состояния не изменяется
1100 0XXXX	Присвоение низшего приоритета входу, указанному в трех младших разрядах команды, и циклическое изменение приоритетов без изменения содержимого регистра состояний
1010 0000	Сброс разряда регистра состояния, имеющего наивысший приоритет, присвоение входу, соответствующему этому разряду, наинизшего приоритета, циклическое изменение приоритетов
1110 0XXXX	Входу, указанному в трех младших разрядах команды, присваивается наинизший приоритет, соответствующий ему разряд регистра состояния сбрасывается, приоритеты остальных входов циклически изменяются

Для задания режима циклического приоритета и изменения приоритетов в этом режиме также используется команда КУ2 (строки 3—5 табл. 1.10).

Для управления регистром маски прерываний контроллера служит команда управления маскированием КУ1, формат которой приведен на рис. 1.27,а. Команда служит для установки и сброса разрядов регистра маски прерываний и может быть выполнена только после проведения процедуры инициализации контроллера.

В контроллере КР580ВН59 возможна организация режима специального маскирования обслуживаемого

прерывания. Режим спецмаскирования служит для того, чтобы разрешить обслуживание входов запросов, имеющих более низкий приоритет, чем маскируемый вход, во время выполнения подпрограммы обслуживания запроса на прерывание по маскируемому входу. В этом режиме входы с более низким приоритетом обслуживаются до тех пор, пока действует режим спецмаскирования. Режим спецмаскирования обслуживаемого прерывания назначается и снимается командой КУЗ (строка 1 табл. 1.11) после начала выполнения подпрограммы обслуживания.

Таблица 1.11

Варианты использования команды КУЗ БИС КР580ВН59

Код команды	Выполняемые функции
0110 1000 1110 1000	Установка режима спецмаскирования
0100 1000 1100 1000	Сброс режима спецмаскирования
0000 1100	Указывается на считывание из контроллера в следующем цикле кода запроса с наивысшим приоритетом
0000 1010	Указывается на считывание из контроллера в следующем цикле содержимого регистра запросов
0000 1011	Указывается на считывание из контроллера в следующем цикле содержимого регистра состояния

Контроллер может быть использован для организации программного опроса запросов на прерывания. В этом режиме контроллер принимает запросы на прерывания и формирует слово состояния опроса, в котором имеются разряд, указывающий на наличие запросов, и три разряда, указывающие номер запроса с наивысшим приоритетом. Режим программного опроса может быть реализован со всеми дисциплинами назначения приоритетов, описанными выше. Во время выполнения режима программного опроса прерываний прерывания МП должны быть запрещены.

Команду, инициализирующую работу контроллера в режиме опроса (строка 3 табл. 1.11), МП посылает в

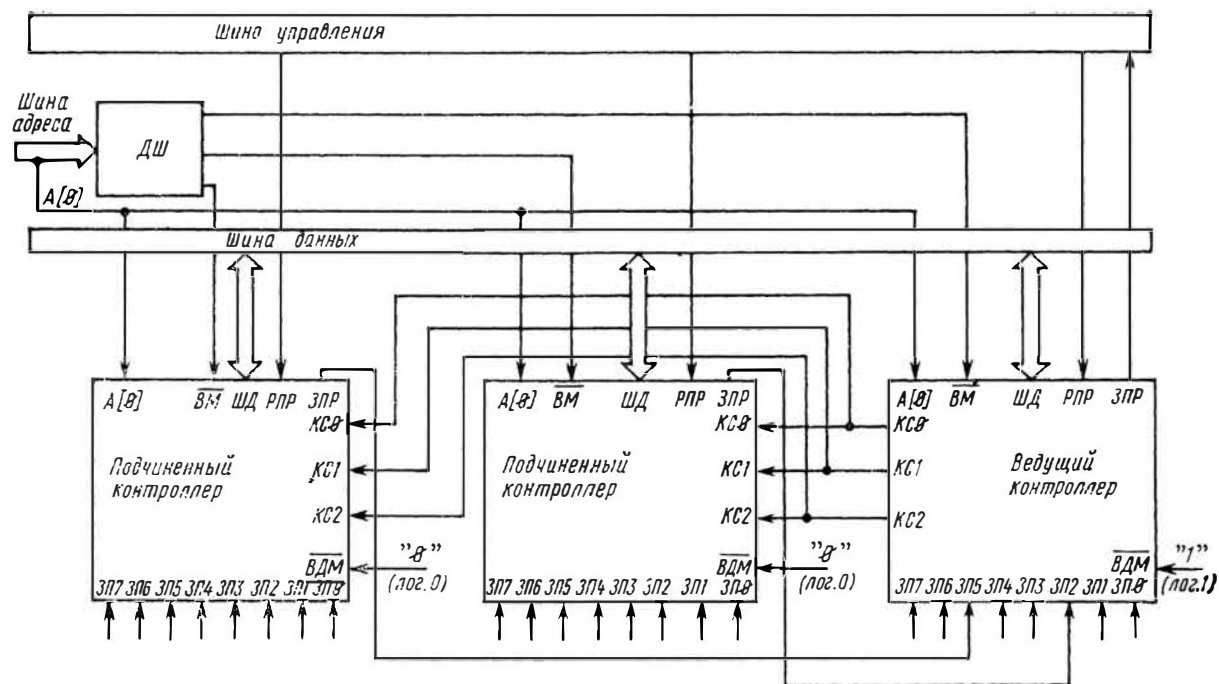
контроллер с помощью команды OUT ($A\emptyset=\emptyset$, $3П=\overline{\emptyset}$). Опрос контроллера со стороны МП осуществляется с помощью команды IN ($A\emptyset=\emptyset$, $\overline{ЧТ}=\emptyset$), получив которую контроллер выставляет на шину данных слово состояния опроса (рис. 1.27,б). Обслуживание запроса на прерывание заключается в декодировании программным способом слова состояния опроса и выполнении соответствующей подпрограммы.

Считывание содержимого регистра запросов и регистра состояния осуществляется путем загрузки в контроллер с помощью команды OUT ($A\emptyset=\emptyset$, $3П=\overline{\emptyset}$) соответствующей команды КУЗ (строки 4,5 табл. 1.11) с последующим выполнением команды IN ($A\emptyset=\overline{\emptyset}$, $\overline{ЧТ}=\emptyset$). Считывание содержимого регистра маски осуществляется выполнением команды IN ($A\emptyset=1$, $\overline{ЧТ}=\emptyset$) без предварительной загрузки команды КУЗ.

Число уровней запросов на прерывание увеличивается с помощью каскадного включения контроллеров (рис. 1.28). При этом один из контроллеров является ведущим, а другие — подчиненными. Любой из входов ведущего контроллера может быть использован для подключения к нему подчиненного.

Последовательность обработки сигнала запроса на прерывание выглядит следующим образом. Любой из подчиненных контроллеров, анализируя уровни приоритетов, поступающих на него запросов на прерывание, может сформировать сигнал запроса на входе ведущего контроллера. Определив, что данный запрос для ведущего контроллера имеет наивысший приоритет, последний формирует сигнал ЗПР для МП. По сигналу $\overline{РПР}$ от МП ведущий контроллер выдает на шину данных код команды CALL. Одновременно ведущий контроллер подает на входы $КС\emptyset$ — $КС2$ подчиненных контроллеров номер ведомого контроллера, запрос на прерывание которого должен обслуживаться. В цикле обработки прерываний по второму и третьему сигналам $\overline{РПР}$ микропроцессора указанный подчиненный контроллер подает на ШД начальный адрес своей подпрограммы обслуживания. При этом устанавливаются соответствующие разряды регистров состояния в ведущем и подчиненном контроллерах.

В начале работы процедура инициализации должна



51 Рис. 1.28. Каскадное включение БИС КР580ВН59 для увеличения числа уровней прерываний

быть проведена для каждого контроллера отдельно. Дисциплины обслуживания прерываний для подчиненных контроллеров могут различаться.

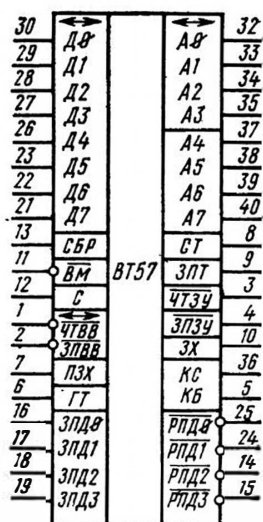


Рис. 1.29. Нумерация выводов программируемого контроллера прямого доступа к памяти КР580BT57

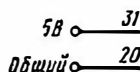
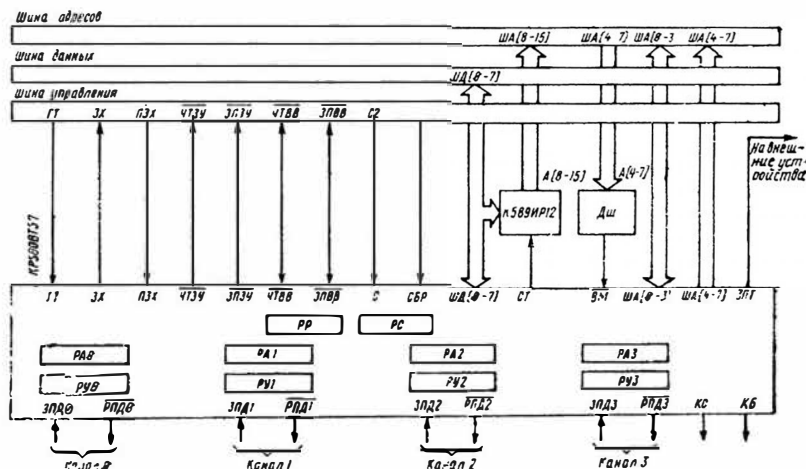


Рис. 1.30. Схема включения контроллера КР580BT57



1.5. Программируемый контроллер прямого доступа к памяти КР580ВТ57

Микросхема КР580ВТ57 предназначена для управления передачей информации между внешними устройствами и ЗУ микропроцессорной системы в режиме прямого доступа к памяти. Основное назначение контроллера — формировать в процессе обмена последовательность адресов ЗУ, а также сигналы управления обменом и процессом представления прямого доступа к памяти.

Микросхема КР580ВТ57

помещена в пластиковый корпус с 40 выводами (рис. 1.29), требует напряжения питания 5 В и потребляет мощность 1 Вт.

Контроллер прямого доступа (рис. 1.30) имеет четыре независимых канала обмена. Блок управления каждого из каналов включает два 16-разрядных регистра: регистр начального адреса (РА), в который заносится начальный адрес ЗУ для передаваемого массива данных, и регистр управления (РУ), в 14 младших разрядах которого помещается число на единицу меньше длины передаваемого массива. Таким образом, максимальная длина передаваемого массива составляет 16К байт.

Два старших разряда РУ определяют тип операции обмена (табл. 1.12). При выполнении операции контроля блок управления каналом формирует все сигналы, обычно вырабатываемые при передаче массива, за исключением сигналов чтения или записи информации в ЗУ и внешнее устройство ЧТЗУ, ЗПЗУ, ЧТВВ, ЗПВВ (табл. 1.13). Поэтому передачи информации не происходит. Однако МП отключается от шин системы и контроллер выдает сигнал разрешения прямого доступа (РПД) на внешнее устройство. Этот сигнал может быть использован периферийным устройством для разрешения внутреннего доступа к только что прочитанным из ЗУ данным для подсчета контрольной суммы.

Таблица 1.12

Возможные операции обмена в режиме прямого доступа к памяти

15-й разряд регистра управления	14-й разряд регистра управления	Операция
0	0	Контроль
0	1	Запись в ЗУ
1	0	Чтение из ЗУ
1	1	Сочетание запрошено

Управляющие сигналы БИС КР580ВТ57

Обозначение	Наименование	Функциональное назначение и условия возникновения
<i>Сигналы связи с периферийными устройствами</i>		
ЗПД $\emptyset$ — ЗПДЗ	Запросы прямого доступа к памяти (входные)	Формируется внешним устройством, требующим обслуживания в режиме прямого доступа; для передачи одного байта информации сигнал ЗПД $\emptyset$ должен быть снят внешним устройством сразу же после получения сигнала РПД от контроллера; для передачи массива данных сигнал ЗПД должен удерживаться внешним устройством до получения от контроллера сигнала КС
РПД $\emptyset$ — РПДЗ	Разрешение прямого доступа (выходные)	Информируют внешнее устройство, что ему предоставлен очередной цикл прямого доступа к памяти. Сигнал устанавливается и сбрасывается для передачи каждого байта данных
КС	Конец счета (выходной)	Указывает на то, что текущий цикл записи или чтения в режиме прямого доступа является последним
КБ	Конец блока данных (выходной)	Сигнал, выдаваемый на внешнее устройство в режиме прямого доступа во время цикла записи или чтения; указывает, что до конца передаваемого блока необходимо выполнить число циклов, кратное 128

Сигналы связи с шиной управления микропроцессорной системы

ГТ	Готовность (входной)	Отсутствие этого сигнала переводит контроллер в состояние ожидания для временного согласования работы памяти и контроллера
ЗХ	Захват (выходной)	Сигнал запроса захвата шин, выдаваемый контроллером на МП при наличии запроса от внешнего устройства
ПЗХ	Подтверждение захвата (входной)	Сигнал, поступающий с МП и разрешающий переход в режим прямого доступа

Обозначение	Наименование	Функциональное назначение и условия возникновения
ЧТЗУ	Чтение из запоминающего устройства (выходной)	Управляет считыванием данных из адресованной ячейки ЗУ в режиме прямого доступа в цикле чтения
ЗПЗУ	Запись в запоминающее устройство (выходной)	Управляет записью данных в ЗУ в режиме прямого доступа в цикле записи
ЧТВВ	Чтение с устройства ввода (в режиме программирования—входной, в режиме прямого доступа—выходной)	В режиме программирования БИС КР580ВТ57 сигнал обеспечивает считывание содержимого того или иного внутреннего регистра контроллера и вывод этой информации на шину данных. В режиме прямого доступа в цикле чтения сигнал осуществляет стробирование ввода данных от внешнего устройства на шину данных
ЗПВВ	Запись в устройство вывода (в режиме программирования—входной, в режиме прямого доступа—выходной)	В режиме программирования сигнал осуществляет запись информации в тот или иной внутренний регистр контроллера. В режиме прямого доступа в цикле записи осуществляет стробирование записи байта данных во внешнее устройство
С	Синхронизация (входной)	Тактовый сигнал, которым осуществляется синхронизация работы контроллера. Как правило, в качестве этого сигнала используется тактовый сигнал С2 МП
СБР	Сброс (входной)	Осуществляет установку в нуль содержимого внутренних регистров контроллера и сброс выходных управляющих сигналов

Обозначение	Наименование	Функциональное назначение и условия возникновения
<i>Сигналы связи с шиной адреса микропроцессорной системы</i>		
ША[Ø]— ША [3]	Адрес, разряды [Ø—3] (в режиме программирования—входные, в режиме прямого доступа—выходные)	В режиме программирования адресуют регистр, в который производится запись или считывание (см. табл. 1.14). В режиме прямого доступа задают четыре младших разряда адреса, формируемого контроллером
ВМ	Выбор микро-схемы (входной)	В режиме программирования—это вход разрешения работы БИС. При высоком уровне сигнала контроллер отключается от шин. В режиме прямого доступа данный вход отключен
А [4]— А [7]	Адрес, разряды [4—7] (выходные)	В режиме прямого доступа задают 4—7-й разряды адреса, формируемого контроллером. В остальное время находятся в состоянии с высоким выходным сопротивлением
СТ	Старший байт адреса (выходной)	Стробирует запись во внешний регистр К589ИР12 по ШД [Ø—7] старшего байта адреса ЗУ
ЗПТ	Запрет (выходной)	Формируется в режиме прямого доступа. Обычно используется для запрета ошибочной работы внешних устройств при этом режиме

Запись информации в РА и РУ каналов осуществляется МП, как правило, с помощью команд OUT (ВЫВОД). Однако возможна организация микропроцессорной системы, в которой обращение к БИС КР580ВТ57 осуществляется как к ячейкам ЗУ. В любом случае для занесения данных в любой из 16-разрядных регистров РА или РУ необходимо выполнить последовательно две команды при одних и тех же сигналах ША [Ø—3] (табл. 1.14): команду занесения в регистр младшего байта, а затем — старшего. В процессе выполнения программы МП может считать содержимое регистров РА и РУ любого канала, для этого также необходимо выполнить две команды IN (ВВОД) с

одним и тем же адресом (сначала считывается младший байт, а затем — старший).

Чтобы разрешить работу одного или нескольких каналов прямого доступа, после занесения требуемой информации в регистры РА и РУ необходимо записать в

Таблица 1.14

Обращение к регистрам БИС КР580ВТ57 в зависимости от сигналов на линиях адреса

Регистр	A3	A2	A1	A0
РА0	0	0	0	0
РУ0	0	0	0	1
РА1	0	0	1	0
РУ1	0	0	1	1
РА2	0	1	0	0
РА3	0	1	1	0
РУ3	0	1	1	1
РР (только запись)	1	0	0	0
РС (только чтение)	1	0	0	0

8-разрядный регистр режима (РР) БИС КР580ВТ57 управляющее слово (рис. 1.31,а). Рассмотрим диаграмму переходов контроллера прямого доступа (рис. 1.32). Длительность каждого состояния определяется тактовой частотой (рис. 1.33). Первоначально контроллер находится в состоянии S1. Сигналы запросов на прямой доступ к памяти ЗПД0—ЗПД3 поступают с внешних устройств. При наличии хотя бы одного такого сигнала от внешнего устройства контроллер выдает в МП сигнал захвата шин (ЗХ) и переходит в состояние S0. При поступлении с МП сигнала подтверждения (ПЗХ), свидетельствующего об отключении МП от шин системы, контроллер переходит в состояние S1 и выдает сигнал запрета (ЗПТ), который предохраняет другие внешние устройства от ошибочного воздействия при появлении на шине системы выработанных контроллером адресов ЗУ. Далее контроллер выдает сигнал РПД на внешнее устройство, запрашивавшее режим прямого доступа и имеющее наивысший приоритет. Если при записи управляющего слова режим циклического приоритета не задан, то нулевой канал имеет высший, а третий канал низший приоритеты. В режиме циклического приоритета

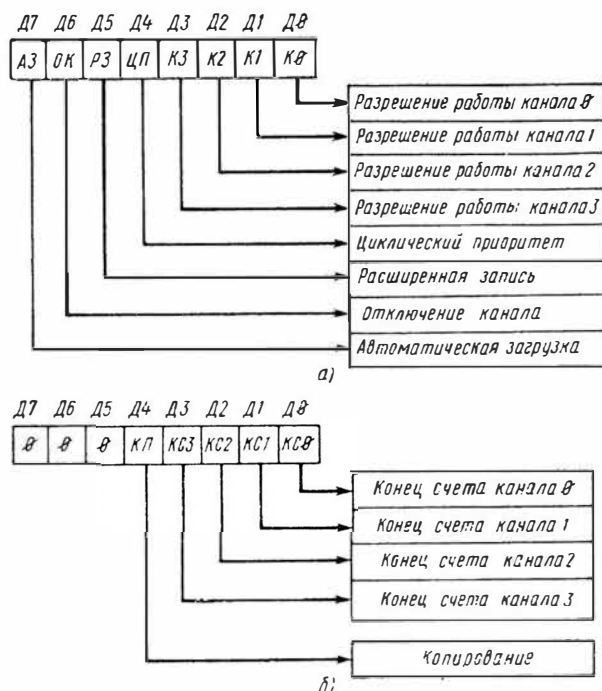


Рис. 1.31. Управляющее слово (а) и слово состояния (б) контроллера прямого доступа к памяти КР580ВТ57

после завершения передачи массива приоритеты каналов циклически изменяются.

В состоянии $S2'$ контроллер формирует также адрес ЗУ. Младший байт адреса выдается контроллером по линиям ША [0—7], старший под действием стробирующего сигнала (СТ) через шину данных ШД [0—7] (см. рис.1.30) записывается во внешний буферный регистр К589ИР12, откуда и подается на шину адреса. Необходимость передачи байта адреса по шине данных вызвана ограниченным числом выводов корпуса.

Цикл записи или чтения байта информации всегда состоит из состояний $S1—S4$. Если цикл работы ЗУ больше цикла контроллера, то последний переходит в состояние $S_{ож}$, выход из которого осуществляется при поступлении сигнала готовности (ГТ). При выполнении операции контроля переход в состояние $S_{ож}$ невозможен.

Рис. 1.32. Диаграмма работы контроллера прямого доступа к памяти КР580ВТ57

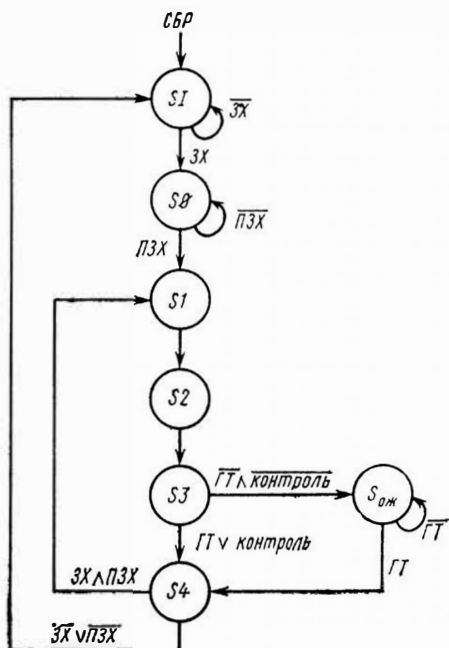
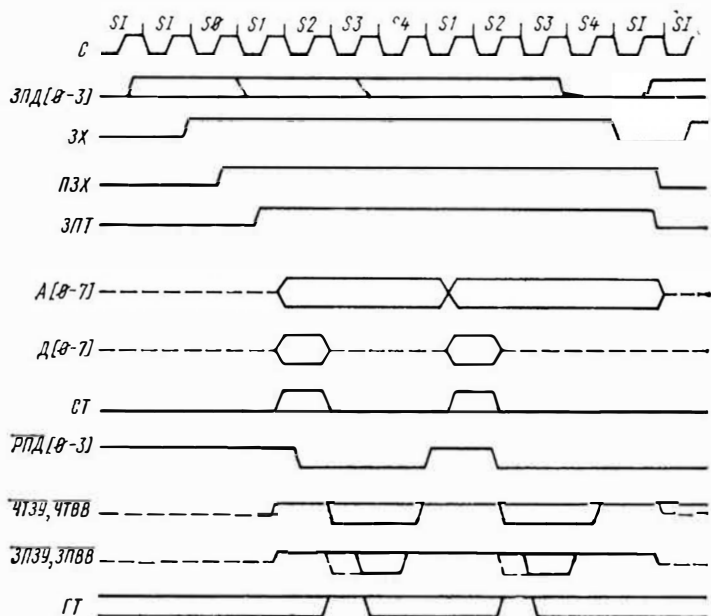


Рис. 1.33. Временные диаграммы программируемого контроллера прямого доступа к памяти КР580ВТ57



В состоянии S4 во время передачи последнего байта контроллер выдает сигнал конца счета (КС). В этом же состоянии на внешнее устройство выдается сигнал конца блока данных (КБ), который указывает, что до конца массива осталось кратное 128 (включая данный) число циклов записи или чтения.

При установке с помощью управляющего слова режима расширенной записи сигналы $\overline{\text{ЗПВВ}}$ и $\overline{\text{ЗПЗУ}}$ в цикле записи вырабатываются несколько раньше, что позволяет ускорить появление сигнала ГТ и избежать, где это возможно, перехода контроллера в состояние ожидания. Единичное значение бита отключения канала (ОК) управляющего слова обуславливает отключение канала прямого доступа при выработке сигнала КС и сброс бита разрешения работы канала в управляющем слове. Повторная работа этого канала возможна только после занесения нового управляющего слова в РР.

Если бит ОК управляющего слова равен нулю, появление сигнала КС не оказывает влияния на бит разрешения работы канала. Выход из режима передачи в этом случае осуществляется внешним устройством за счет снятия сигнала запроса (ЗПД).

Режим автоматической загрузки (разряд А3 управляющего слова) позволяет использовать канал 2 для повторной передачи блока или передачи последовательности блоков без вмешательства МП в промежутке между передачей блоков.

Если ранее в РР занесено управляющее слово с единичным битом А3, то при занесении данных в РА2 и РУ2 эти же данные будут загружены и в РА3 и РУ3. После начала работы канала 2 и передачи по нему массива данных в соответствии с содержимым РА2, РУ2 во время специального цикла копирования данные из РА3, РУ3 передаются в РА2, РУ2. Содержимое РА3, РУ3 при этом не изменяется. Следующим после цикла копирования будет первый цикл передачи байта нового массива данных. Повторная передача одного и того же блока данных во внешнее устройство используется, например, при регенерации изображения на экране электронно-лучевой трубки (ЭЛТ). Если после занесения параметров массива в РА2, РУ2, в регистры третьего канала РА3, РУ3 занести другие параметры, то будет осуществлена последовательная передача разных массивов данных.

Во время выполнения цикла копирования в РС БИС КР580ВТ57 устанавливается бит копирования (см. рис. 1.31,б). Считывая содержимое этого регистра, МП определяет конец передачи массива данных, что позволяет ему внести в РАЗ, РУЗ параметры следующего массива, подлежащего передаче. После окончания цикла копирования бит копирования слова состояния сбрасывается. При установленном в единицу разряде АЗ управляющего слова независимо от значения бита ОК отключение канала 2 при выработке сигнала КС не производится.

Разряды ДØ—Д3 слова состояния устанавливаются после появления сигнала КС по соответствующему каналу. При считывании слова состояния биты конца счета ДØ—Д3 сбрасываются.

1.6. БИС программируемого таймера КР580ВИ53

Микросхема КР580ВИ53 предназначена для задания временных интервалов в микропроцессорных системах, может использоваться как счетчик внешних событий, программируемый делитель частоты, одновибратор с программируемой длительностью импульса.

Размещается БИС КР580ВИ53 в пластиковом корпусе с 24 выводами (рис. 1.34), требует напряжения питания 5 В и потребляет мощность 1 Вт.

В состав БИС КР580ВИ53 (рис. 1.35) входят три независимых 16-разрядных счетчика (счетчики 0, 1, 2), работающих на вычитание в двоичном или двоично-десятичном коде. На входы счетчиков СØ, С1, С2 (табл. 1.15) поступают импульсы тактового генератора или подсчитываемые импульсы с внешних устройств. Прием этих импульсов разрешается сигналами стробирования СТРØ, СТР1, СТР2. При равенстве содержи-

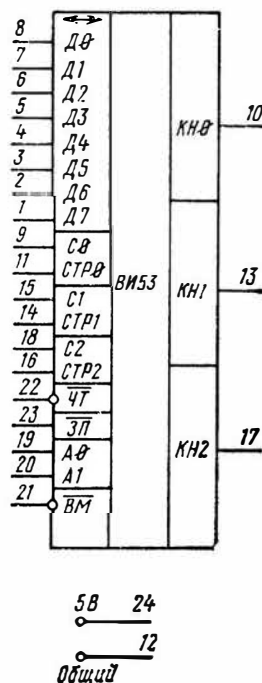


Рис. 1.34. Нумерация выводов БИС КР580ВИ53

мого какого-либо счетчика нулю вырабатывается один из сигналов КН0, КН1, КН2 (Конец 0, 1 или 2), которые могут использоваться как сигналы запросов на прерывание процессора. Режимы работы каждого из счетчиков задаются независимо друг от друга занесением управляющего слова в специальный регистр. Начальные значения счетчиков также загружаются в БИС через шину данных ШД [0—7]. Кроме того, содержи-

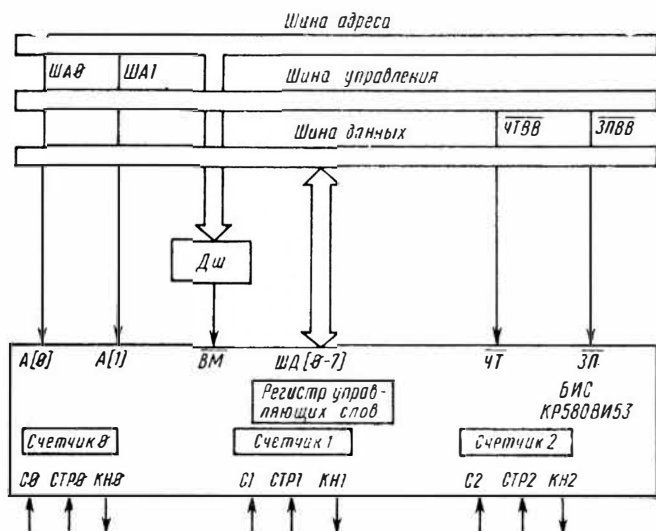


Рис. 1.35. Схема программируемого таймера КР580ВИ53

мое счетчиков может быть в любой момент считано МП (табл. 1.16). Считывание содержимого управляющего регистра невозможно. Обычно БИС КР580ВИ53 подключается к шинам микропроцессорной системы как внешнее устройство (см. рис. 1.35). Однако возможна организация микропроцессорной системы, при которой обращение к БИС осуществляется как к ячейкам ОЗУ. В любом случае БИС КР580ВИ53 соответствует четыре адреса.

Инициализация каждого счетчика осуществляется занесением управляющего слова (рис. 1.36) в предназначенный для этого регистр. Наряду с номером счетчика это слово задает запись и считывание одного или двух байтов содержимого счетчика, режим его работы и двоичную или двоично-десятичную систему счисле-

ния. Вслед за управляющим словом в указанный счетчик (табл. 1.16) должно быть занесено его начальное содержимое. При занесении обоих байтов содержимого четчика они заносятся последовательно при одина-

Таблица 1.15

Управляющие сигналы БИС КР580ВИ53

Обозначение	Наименование	Функциональное назначение и условия возникновения
<i>Сигналы шин микропроцессорной системы</i>		
A [1], A [0]	Адрес 1, адрес 0 (входные)	Выбор одного из трех счетчиков или управляющего регистра
$\overline{BM}$	Выбор микросхемы (входной)	При единичном уровне сигнала выходы ШЛ [0—7] находятся в состоянии с высоким выходным сопротивлением и никакие управляющие сигналы на БИС не воздействуют
$\overline{CT}$	Чтение (входной)	Чтение содержимого одного из трех счетчиков
$\overline{WP}$	Запись (входной)	Запись информации в один из трех счетчиков или в управляющий регистр
<i>Сигналы счетчиков</i>		
C0, C1, C2	Счетчик 0, счетчик 1, счетчик 2 (входные)	Подсчитываемые импульсы, максимальная частота 2 МГц
STR0, STR1, STR2	Стробирование 0, стробирование 1, стробирование 2 (входные)	Сигналы разрешения приема импульсов C0, C1, C2 в соответствующие счетчики
KN0, KN1, KN2	Конец 0, конец 1, конец 2 (выходные)	Конец счета нулевого, первого и второго счетчиков соответственно

ковых значениях сигналов A0, A1. Сигналы KN0, KN1 или KN2 вырабатываются после поступления на вход счетчика C0, C1 или C2 $n+1$ импульса, где n — начальное содержимое счетчика. Считывание байтов текущего содержимого счетчика микропроцессором при единичных значениях разрядов D4, D5 управляющего слова также осуществляется последовательно при одинаковых значениях сигналов A0, A1.

Каждый из счетчиков может работать в одном из шести режимов. В режиме 0 счетчик функционирует как программируемый таймер (рис. 1.37). Сигнал КН после занесения управляющего слова имеет низкий уровень. Уменьшение содержимого счетчика под действием импульсов, поступающих на вход счетчика С, начинается сразу же после занесения в счетчик начального значения, причем на счетчик воздействуют только импульсы, поступающие при высоком уровне сигнала СТР.

Таблица 1.16

Зависимость направления передачи информации от управляющих сигналов БИС КР589ВИ53

$\overline{EM}$	$\overline{CT}$	$\overline{3P}$	$A\emptyset$	$A1$	Направление передачи информации
0	1	0	0	0	ШД→Счетчик $\emptyset$
0	1	0	0	1	ШД→Счетчик 1
0	1	0	1	0	ШД→Счетчик 2
0	1	0	1	1	ШД→Регистр управляющего слова
0	0	1	0	0	Счетчик $\emptyset$ →ШД
0	0	1	0	1	Счетчик 1→ШД
0	0	1	1	0	Счетчик 2→ШД
0	0	1	1	1	Нет передачи; ШД [$\emptyset-7$] в состоянии с высоким выходным сопротивлением
1	×	×	×	×	
0	1	1	×	×	

Если во время работы счетчика МП осуществляет занесение нового начального содержимого, то после занесения первого байта счет останавливается, а после занесения второго байта счет начинается с нового начального значения.

В режиме 1 счетчик используется в качестве программируемого одновибратора. Сигнал КН снимается при поступлении первого импульса после установки сигнала СТР. При окончании счета сигнал КН устанавливается. Занесение в счетчик нового начального значения при низком уровне сигнала КН не изменяет длительности обрабатываемого импульса. Вновь занесенное начальное значение определяет длительность импульса на выходе КН после следующего положительного фронта сигнала СТР. Текущее содержимое счетчика может быть считано в МП в любой момент времени. Сигнал КН появляется при поступлении $n+1$ -го

импульса после последнего положительного фронта сигнала СТР, в связи с чем повторный запуск «одно-вибратора» может быть осуществлен до окончания отработываемого импульса.

В режиме 2 счетчик используется в качестве делителя частоты. Выходной сигнал КН имеет низкий уро-

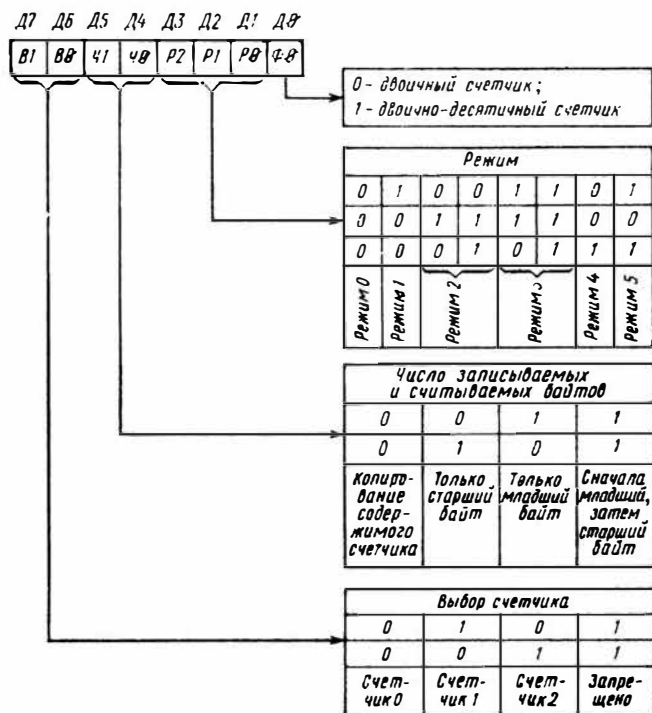


Рис. 1.36. Формат управляющего слова программируемого таймера КР580ВИ53

вень в течение одного периода поступающих на вход С импульсов. Период выходных импульсов КН составляет n периодов входных импульсов, где n — начальное содержимое счетчика (см. рис. 1.37). При занесении в счетчик нового начального значения отработываемый период выходных импульсов КН не изменяется. Занесенное значение определяет длительности последующих

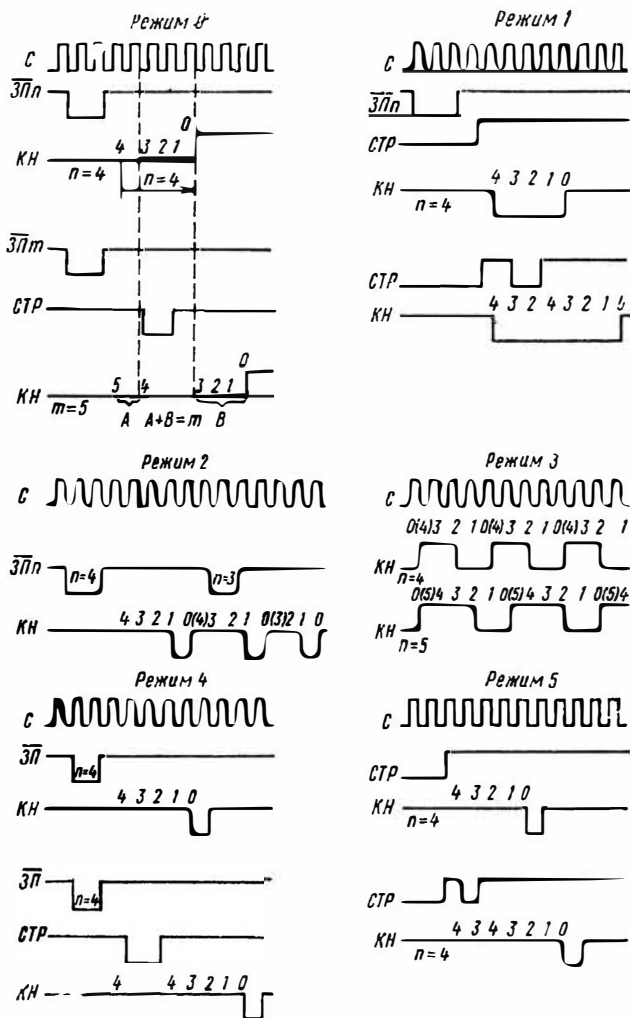


Рис. 1.37. Временные диаграммы программируемого таймера КР580ВН53

периодов. При поступлении на счетчик сигнала $СТР$ низкого уровня сигнал KH всегда имеет высокий уровень. Положительный фронт сигнала $СТР$ инициирует счет с начального значения. Таким образом, сигнал

СТР может быть использован для синхронизации работы счетчика.

Режим 3 отличается от режима 2 тем, что выходной сигнал КН в течение одной половины периода имеет высокий уровень, в течение другой половины периода — низкий уровень. Если загружаемое в регистр значение n нечетно, то высокий уровень сигнала КН сохраняется до поступления $(n+1)/2$ -го импульса. При занесении нового начального значения во время работы счетчика последний начинает функционировать в соответствии с этим значением после первого изменения уровня сигнала КН.

В режиме 4 после загрузки управляющего слова выход КН имеет высокий уровень. После занесения начального значения счетчик начинает работать. По окончании счета сигнал КН сбрасывается, а при поступлении следующего импульса на вход С вновь устанавливается. Занесение в счетчик нового начального значения влияет только на следующий временной интервал. Снятие сигнала СТР останавливает счет, а его повторная установка вызывает работу счетчика с начального значения.

В режиме 5 счет начинается после поступления положительного фронта на вход СТР. Сигнал КН сбрасывается после окончания счета и при поступлении следующего импульса на вход С вновь устанавливается.

Считывание содержимого любого счетчика может быть выполнено МП двумя способами. В простейшем случае считывание осуществляется одной или двумя командами IN (ВВОД) в зависимости от значения разрядов Д4, Д5, ранее записанного в специальный регистр управляющего слова.

При этом для получения стабильного результата во время операций считывания должно быть запрещено изменение содержимого счетчика. Запрет может осуществляться либо сигналом СТР, либо с дополнительными логическими схемами, запрещающими поступление сигналов на вход счетчика С.

Для считывания содержимого любого счетчика без нарушения операции счета МП заносит в БИС КР580ВИ53 управляющее слово с нулевыми разрядами Д4, Д5. Значение разрядов Д0—Д3 при этом несущественно. Содержимое заданного счетчика копируется в специальном буфере, откуда считывается обычным

образом. Следует отметить, что в любом случае операция считывания (1 или 2 байта) должна быть проведена полностью.

1.7. БИС программируемого контроллера видеотерминала КР580ВГ75

Микросхема КР580ВГ75 представляет программируемый контроллер ЭЛТ, предназначенный для связи растровых сканирующих дисплеев с микропроцессорными системами. Основным назначением БИС является согласование синхронного режима работы дисплея с асинхронным режимом обмена внутри микропроцессорной системы посредством буферизации информации, поступающей из памяти последней. Микросхема КР580ВГ75 размещается в пластиковом корпусе с 40

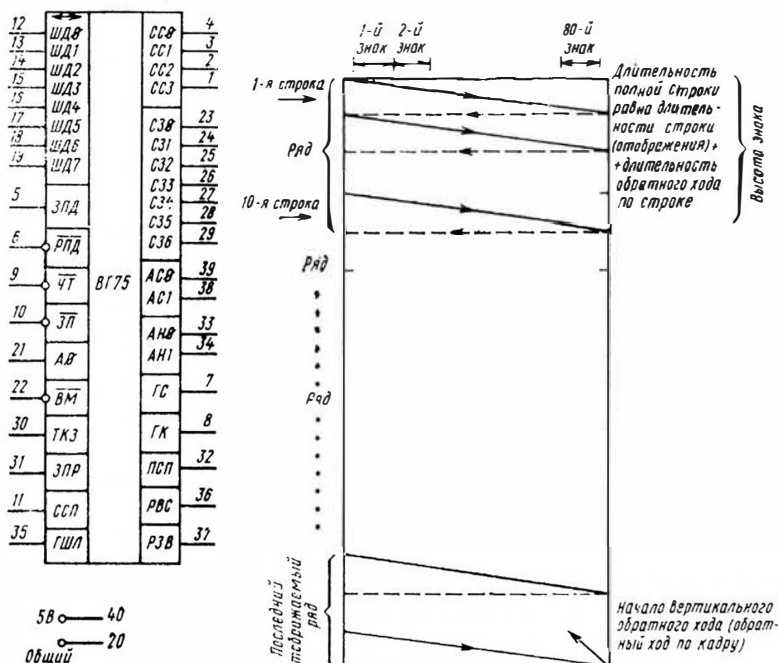


Рис. 1.38. Нумерация выводов программируемого контроллера видеотерминала КР580ВГ75

Рис. 1.39. Принцип движения луча по экрану ЭЛТ при формировании знаковых рядов

выводами (рис. 1.38), требует напряжение питания 5 В и потребляет мощность 1 Вт.

Дисплей обеспечивает отображение в виде буквенно-цифровых символов либо в графической форме информации, хранящейся в памяти микропроцессорной системы. Информация отображается в виде дискретов (точек), высвечиваемых на экране при движении луча слева направо по телевизионной строке (рис. 1.39). Изображение на экране состоит из знаковых рядов (рис. 1.40), в которые объединены телевизионные стро-

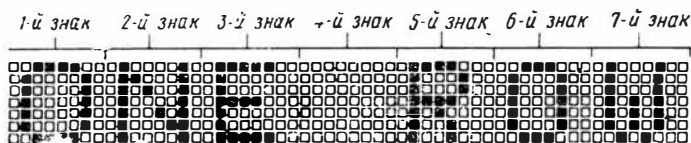


Рис. 1.40. Структура знакового ряда и отображение знаков

ки. Обратный ход луча по строке и кадру гасится. Число знаковых рядов в кадре, телевизионных строк и знаков в ряду, длительность обратного хода по строке и кадру являются программируемыми.

Контроллер ЭЛТ КР580ВГ75 обеспечивает буферизацию (накопление) отображаемой информации, формирование сигналов синхронизации строчной и кадровой развертки, генерацию курсора, обнаружение сигналов светового пера. Гибкость в использовании обеспечивается возможностью программной перестройки режимов работы контроллера.

Восьмиразрядный двунаправленный буфер ШД контроллера (рис. 1.41) имеет выходы с тремя состояниями и используется для передачи информации, которой контроллер обменивается с микропроцессорной системой. Блок управления чтением-записью и ПДП содержит регистры команд, параметров и состояния. Управление записью и считыванием осуществляется сигналами ЧТ, ЗП, ВМ, АФ (табл. 1.17). В режиме ПДП дополнительный сигнал выбора микросхемы является низкий уровень на входе РПД. Программируемый счетчик знаков задает число знаков в отображаемом ряду и в обратном ходе по строке. Управление счетчиком осуществляется по входу ТКЗ (тактыый сигнал знаковый). Программируемый счетчик строк задает

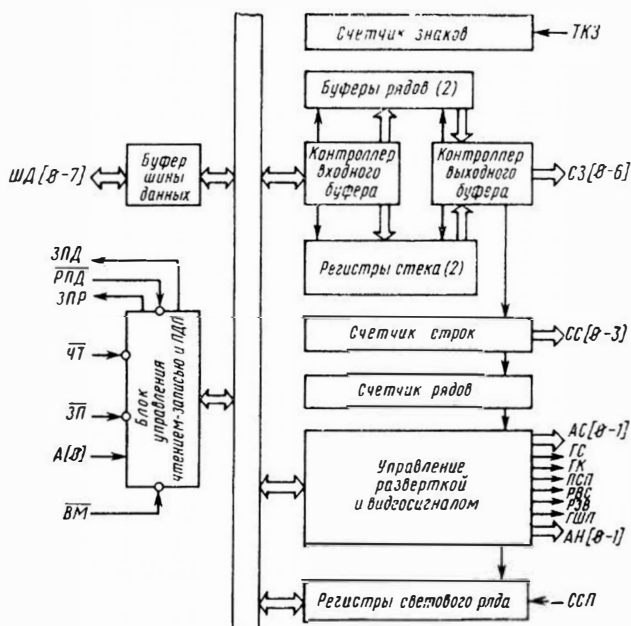


Рис. 1.41. Структурная схема программируемого контроллера видеотерминала КР580ВГ75

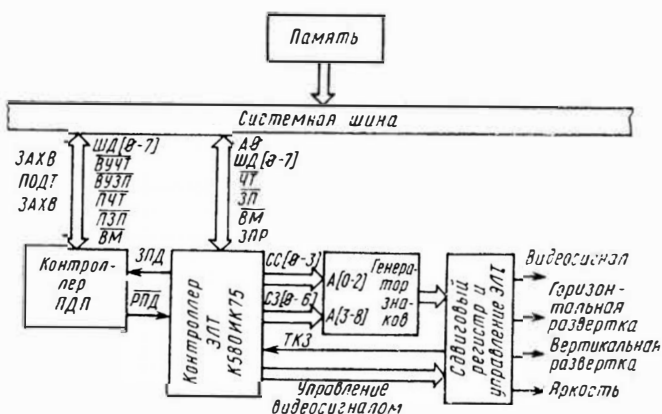


Рис. 1.42. Схема включения БИС КР580ВГ75 в систему отображения

число телевизионных строк в отображаемом ряду знаков. Выходы счетчика строк используются для адресации внешнего ПЗУ (ППЗУ) знакогенератора. Программируемый счетчик рядов задает число рядов знаков в одном отображаемом кадре и в обратном ходе по кадру. В регистры светового пера заносится инфор-

Таблица 1.17

Сигналы управления записью-считыванием БИС КР580ВГ75

А [Ø]	$\overline{CT}$	$\overline{ЗП}$	$\overline{ВМ}$	Операция
0	0	1	0	Чтение из регистра параметров
0	1	0	0	Запись в регистр параметров
1	0	1	0	Чтение слова состояния
1	1	0	0	Запись в регистр команд
×	1	1	0	Перевод буфера ШД в состояние с высоким выходным сопротивлением
×	×	×	1	

мация о положении светового пера на экране. Стробирование информации в эти регистры осуществляется по положительному фронту сигнала ССП (сигнал светового пера). Схемы управления разверткой и видеосигналом формируют сигналы ГС (гашение обратного хода по строке) и ГК (гашение обратного хода по кадру), а также сигналы АС [Ø—1], ПСП, РВС, РЗВ, ГШЛ и АН[Ø—1] (табл. 1.18).

Два буфера по 80 знаков заполняются кодами отображаемых знаков из памяти микропроцессорной системы. Два регистра стека емкостью по 16 знаков используются для хранения кодов атрибутов поля. Контроллеры входного и выходного буферов декодируют три старших разряда байтов информации, помещенной в буферы рядов. При обнаружении атрибута знака, поля или специального кода контроллеры формируют соответствующие сигналы, управляющие работой микросхем. Например, при обнаружении специального кода «Конец экрана — стоп ПДП» контроллер входного буфера останавливает дальнейшие запросы ПДП. Контроллер выходного буфера активизирует выход ПСП при обнаружении атрибута поля «Засвечивание».

При работе контроллера в системе (рис. 1.42) информация из памяти микропроцессорной системы через

Управляющие сигналы БИС КР580ВГ75

Обозначение	Наименование	Функциональное назначение и условия возникновения
<i>Сигналы управления ПДП</i>		
ЗПР	Запрос на прерывание (выходной)	Сигнал запроса на прерывание процессора
ЗПД	Запрос прямого доступа (выходной)	Сигнал запроса ПДП, подается на контроллер ПДП
РПД	Разрешение прямого доступа (входной)	Разрешение прямого доступа. Поступает от контроллера ПДП
<i>Сигналы управления разверткой и видеосигналом</i>		
АС [0—1]	Атрибуты строк (выходные)	Коды атрибутов строк. Расшифровываются внешними схемами и используются для отображения горизонтальных и вертикальных линий на экране в соответствии с видом графического изображения, определяемого кодами атрибутов знака
АН [0—1]	Атрибуты общего назначения	Коды атрибутов общего назначения. Активируются с помощью кодов атрибутов поля
ПСП	Подсветка позиций (выходной)	Сигнал для подсветки на экране позиций, определяемых кодами атрибутов знака или атрибутов поля
РВС	Ревверс видеосигнала (выходной)	Сигнал, сообщающий дисплею о реверсе видеосигнала. Он становится активным во время отображения курсора, если курсор запрограммирован в виде видеоблока, или в позициях экрана, определенных кодами атрибутов поля
РЗВ	Разрешение видеосигнала (выходной)	Сигнал, разрешающий прохождение видеосигнала на ЭЛТ. Выход является активным в позициях экрана, определяемых кодами атрибутов, и при программировании курсора в виде подчеркивания
ГС	Гашение обратного хода по строке (выходной)	Сигнал имеет высокий уровень во время обратного хода по строке. В это время сигнал РВС имеет высокий уровень, а сигнал РЗВ—низкий
ГК	Гашение обратного хода по кадру (выходной)	Сигнал имеет высокий уровень во время обратного хода по кадру. В это время сигнал РВС имеет высокий уровень, а сигнал РЗВ—низкий

Обозначение	Наименование	Функциональное назначение и условия возникновения
ГШЛ	Гашение луча (выходной)	Сигнал становится активным: во время обратного хода по строке и кадру; во время первой и последней строки в ряду; если запрограммировано подчеркивание строки с номером ≤ 8 ; когда обнаружен сигнал конца ряда и конца экрана; во время регулярных интервалов (1/16 частоты кадра для курсора, 1/32 частоты кадра для атрибутов знака и поля) для создания мерцающего изображения в соответствии с режимом, запрограммированным для курсора, атрибутов знака и поля
<i>Сигналы управления ПЗУ знакогенератора</i>		
СЗ [Ø—3]	Счетчик строк, разряды 0—3 (выходные)	Сигналы с этих выходов подаются на знакогенератор
СЗ [Ø—6]	Счетчик знаков, разряды 0—6 (выходные)	Информация из буферов рядов подается на знакогенератор
<i>Специальные сигналы</i>		
ТКЗ	Тактовый сигнал знаковый (входной)	Тактовый сигнал, по которому осуществляется вывод информации на ЭЛТ
ССП	Сигнал светового пера (входной)	Сообщение от дисплея об обнаружении сигнала светового пера

контроллер ПДП поступает в буферы рядов контроллера ЭЛТ КР580ВГ75. Два буфера рядов контроллера ЭЛТ работают поочередно, т. е. в то время, когда содержимое одного из них отображается на экране, другой накапливает поступающую информацию.

Растровый сканирующий дисплей отображает ряды знаков по одной строке в каждый момент времени, поэтому 6-разрядный код знака с выходов СЗ [Ø—6] контроллера ЭЛТ подвергается построчной дешифрации с помощью генератора знаков. Генератор знаков представляет собой ПЗУ, в которое записаны $2^6 = 64$ алфавитно-цифровых знака кода КОИ-7. На его адресные

входы А [3—8] поступают коды знака с выходов СЗ [0—6] контролера ЭЛТ, а на адресные входы А [0—3] — выход счетчика строк СС [0—3]. Семиразрядный выход ПЗУ соответствует семи точкам, которые составляют сегмент строки отдельного знака (см. рис. 1.40). Выход ПЗУ соединен со входом параллельного сдвигового регистра, тактируемого сигналом с частотой, равной частоте генерации точек на экране. Выход сдвигового регистра подается на видеовход ЭЛТ. Ряды знаков отображаются один за другим

Номер строки		Работа счетчика строк в режиме 0	Работа счетчика строк в режиме 1
0	□□□□□□□□	0 0 0 0	1 1 1 1
1	□□□□■□□□	0 0 0 1	0 0 0 0
2	□□□□■□□□	0 0 1 0	0 0 0 1
3	□□□□■□□□	0 0 1 1	0 0 1 0
4	□■□□□□□□	0 1 0 0	0 0 1 1
5	□■□□□□□□	0 1 0 1	0 1 0 0
6	□■□□■□□□	0 1 1 0	0 1 0 1
7	□■□□■□□□	0 1 1 1	0 1 1 0
8	□■□□□□□□	1 0 0 0	0 1 1 1
9	□■□□□□□□	1 0 0 1	1 0 0 0
10	□□□□□□□□	1 0 1 0	1 0 0 1
11	□□□□□□□□	1 0 1 1	1 0 1 0
12	□□□□□□□□	1 1 0 0	1 0 1 1
13	□□□□□□□□	1 1 0 1	1 1 0 0
14	□□□□□□□□	1 1 1 0	1 1 0 1
15	□□□□□□□□	1 1 1 1	1 1 1 0

Номер строки		Работа счетчика строк в режиме 0	Работа счетчика строк в режиме 1
0	□□□□□□□□	0 0 0 0	1 0 0 1
1	□□□□■□□□	0 0 0 1	0 0 0 0
2	□□□□■□□□	0 0 1 0	0 0 0 1
3	□□□□■□□□	0 0 1 1	0 0 1 0
4	□■□□□□□□	0 1 0 0	0 0 1 1
5	□■□□□□□□	0 1 0 1	0 1 0 0
6	□■□□■□□□	0 1 1 0	0 1 0 1
7	□■□□■□□□	0 1 1 1	0 1 1 0
8	□□□□□□□□	1 0 0 0	0 1 1 1
9	□□□□□□□□	1 0 0 1	1 0 0 0

Рис. 1.43. Программирование режимов работы счетчика строк

Номер строки		Работа счетчика строк в режиме 0	Работа счетчика строк в режиме 1
0	□□□□□□□□	0 0 0 0	1 0 1 1
1	□□□□■□□□	0 0 0 1	0 0 0 0
2	□□□□■□□□	0 0 1 0	0 0 0 1
3	□□□□■□□□	0 0 1 1	0 0 1 0
4	□■□□□□□□	0 1 0 0	0 0 1 1
5	□■□□□□□□	0 1 0 1	0 1 0 0
6	□■□□■□□□	0 1 1 0	0 1 0 1
7	□■□□■□□□	0 1 1 1	0 1 1 0
8	□■□□□□□□	1 0 0 0	0 1 1 1
9	□■□□□□□□	1 0 0 1	1 0 0 0
10	□□□□□□□□	1 0 1 0	1 0 0 1
11	□□□□□□□□	1 0 1 1	1 0 1 0

Номер строки		Работа счетчика строк в режиме 0	Работа счетчика строк в режиме 1
0	□□□□■□□□	0 0 0 0	0 1 1 1
1	□□□□■□□□	0 0 0 1	0 0 0 0
2	□□□□■□□□	0 0 1 0	0 0 0 1
3	□□□□■□□□	0 0 1 1	0 0 1 0
4	□■□□□□□□	0 1 0 0	0 0 1 1
5	□■□□□□□□	0 1 0 1	0 1 0 0
6	□■□□■□□□	0 1 1 0	0 1 0 1
7	□■□□■□□□	0 1 1 1	0 1 1 0

Рис. 1.44. Отображаемые знаки при работе контроллера в режиме подчеркивания строки

вплоть до последнего в кадре. Во время отображения последнего ряда контроллер КР580ВГ75 посылает запрос прерывания ЗПР в МП. Определив, что запрос

поступил от контроллера ЭЛТ, МП вызывает подпрограмму обработки прерываний, которая реинициализирует контроллер ПДП. Сигнал ЗПР снимается после считывания слова состояния контроллера.

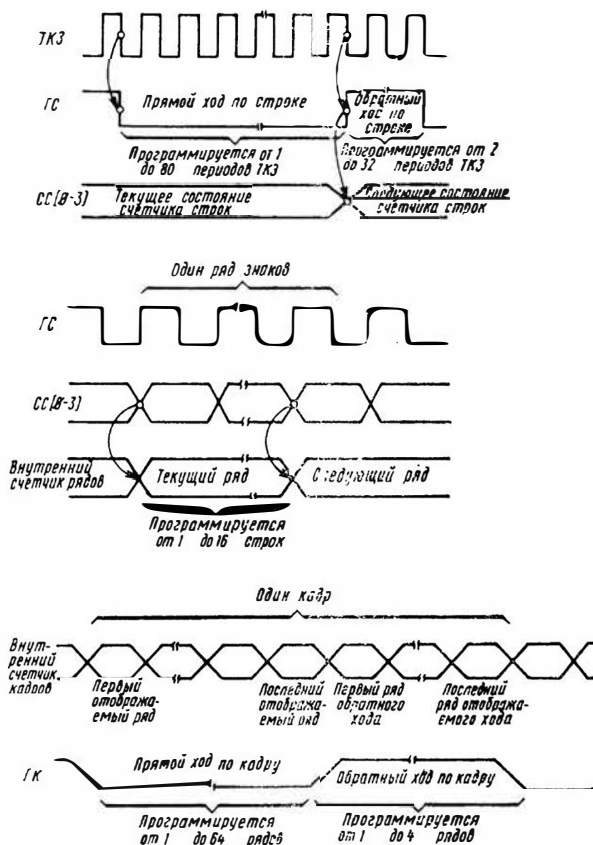


Рис. 1.45. Временные диаграммы формирования сигналов управления разверткой

Состояние счетчика строк остается неизменным при прямом ходе по строке, когда на выходы кодов знаков СЗ [Ø—6] выдается содержимое отображаемого буфера. Счетчик строк наращивается на 1 во время обратного хода по строке, а во время прямого хода по следующей строке коды знаков отображаемого ряда вновь

выводятся на СЗ [Ø—6]. Число строк в знаковом ряду программируется от 1 до 16.

Счетчик строк может быть запрограммирован для работы в одном из двух режимов. В режиме Ø коды счетчика строк соответствуют номерам строк. В режиме 1 коды счетчика строк меньше номера строки на 1. Номеру первой строки в этом режиме соответствует последнее состояние счетчика (рис. 1.43). Режим Ø рассчитан на генераторы знаков, адресация которых начинается с адреса 1. Режим 1 рассчитан на генераторы знаков, адресация которых начинается с адреса Ø.

Номер подчеркиваемой строки программируется от Ø до 15 и не зависит от режима счетчика строк. Если номер подчеркиваемой строки больше 7 (старший разряд номера строки равен 1), то верхняя и нижняя строки ряда бланкируются (рис. 1.44). Если номер подчеркиваемой строки больше максимального числа строк, то подчеркивания не происходит. Бланкирование осуществляется с помощью сигнала ГШЛ, подчеркивание — сигналом РЗВ.

Счетчики знаков, строк и рядов служат для управления разверткой. Временные диаграммы работы счетчиков приведены на рис. 1.45. Счетчик знаков считает число отображаемых знаков в ряду (рис. 1.46) и программируется на число знаков от 1 до 80. Управление счетчиком знаков осуществляется сигналом по входу ТКЗ. В конце строки он увеличивает содержимое счетчика строк на 1 и начинает считать число знаков в обратном ходе по строке (программируется от 2 до 32). Этот процесс продолжается до конца ряда.

Счетчик строк запускается от счетчика знаков. Он выдает номера строк на выходы СС [Ø—3] для генератора знаков. После того как счетчик строк сосчитает все строки в ряду (число строк программируется от 1 до 16), он увеличивает содержимое счетчика рядов на 1 и начинает считать сначала.

Счетчик рядов запускается от счетчика строк и считает число знаковых рядов в кадре (рис. 1.46), программируется от 1 до 64 и управляет буферами рядов. После того как счетчик рядов сосчитает все ряды в кадре, он начинает считать обратный ход по кадру (программируется от 1 до 4). Сигнал ГШЛ становится активным во время обратного хода по строке и кадру. В контроллере имеется возможность бланкировать че-

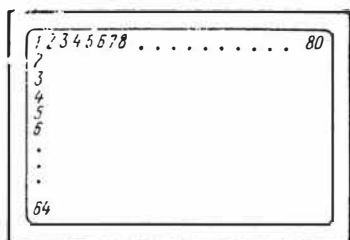


Рис. 1.46. Расположение знаковых позиций и знаковых рядов на экране

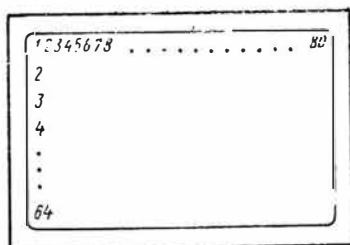


Рис. 1.47. Бланирование чередующихся рядов

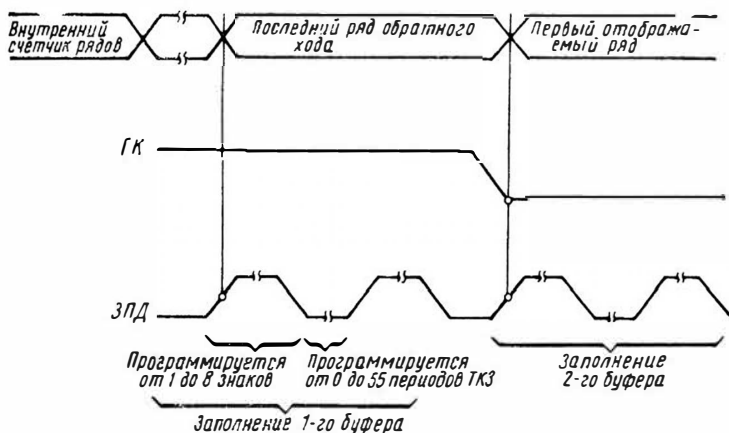


Рис. 1.48. Временная диаграмма формирования сигнала запроса на прямой доступ к памяти

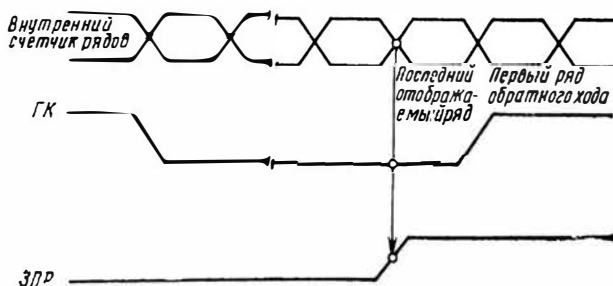


Рис. 1.49. Временная диаграмма формирования сигнала запроса на прерывание

редующиеся ряды. В этом режиме первый ряд отображается, второй — бланкируется, третий — отображается и т. д. (рис. 1.47).

Контроллер КР580ВГ75 может быть запрограммирован на запрос передачи пакета знаков (от 1 до 8) в режиме ПДП. Интервал между пакетами ПДП тоже программируется (от 0 до 55 периодов ТКЗ). Это позволяет согласовать работу контроллера с системными требованиями.

Первый запрос ПДП происходит во время обратного хода по кадру (рис. 1.48). В ответ на него от контроллера ПДП приходит сигнал подтверждения ПДП — РПД, который является дополнительным сигналом выбора микросхемы во время записи пакета знаков. Каждый байт знака в пакете записывается своим сигналом ЗП. Эти сигналы считаются в системе и после достижения запрограммированной длины пакета сигнал РПД снимается. Далее отсчитывается число периодов ТКЗ, на которое запрограммирован интервал между пакетами, и следует очередной запрос. Запросы ПДП продолжают до тех пор, пока не заполнится буфер ряда. Если буфер оказался заполненным в середине пакета, то контроллер КР580ВГ75 прерывает пакет, сбрасывает счетчик пакетов и больше запросы ПДП не вырабатываются до начала следующего ряда. В начале следующего ряда вновь вырабатываются запросы ПДП до тех пор, пока не будет заполнен другой буфер ряда. В конце текущего кадра контроллер ПДП обычно инициализируется для формирования следующего кадра.

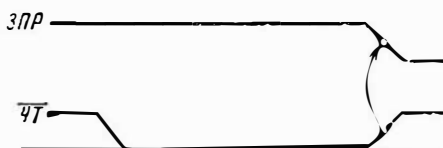
Контроллер КР580ВГ75 может быть запрограммирован на формирование запроса прерывания (ЗПР) в конце каждого кадра. Этот запрос может быть использован для того, чтобы реинициализировать контроллер ПДП. Если флаг разрешения прерывания контроллера установлен, то запрос на прерывание выдается в конце последнего отображаемого ряда (рис. 1.49). Запрос на прерывание сбрасывается после чтения регистра состояния (рис. 1.50). Другой метод реинициализации контроллера ПДП — заставить контроллер ПДП прерывать самого себя по окончании счета. При таком способе нет необходимости устанавливать флаг разрешения прерываний.

Знаки, поступающие в контроллер КР580ВГ75, явля-

ются 8-разрядными. Старший бит приходящего на контроллер байта служит для определения, является ли он отображаемым знаком (старший разряд = 0) или это визуальный атрибут или специальный код (старший разряд = 1). Существуют два типа визуальных атрибутов — атрибуты знаков и атрибуты поля.

Атрибуты знаков применяются для создания графических символов без использования генератора знаков. Это достигается избирательной установкой выходов атрибутов строк АС [0-1], выхода ГШЛ и выхода РЗВ (табл. 1.19). Атрибуты знаков могут быть запрограм-

Рис. 1.50. Временная диаграмма сброса сигнала запроса на прерывание



мированы на мерцание или засвечивание (рис. 1.51,а). Режим мерцания осуществляется выходом ГШЛ, который возбуждается с частотой, равной 1/32 частоты регенерации кадра. Режим засвечивания реализуется выходом ПСП (рис. 1.52).

Атрибуты поля используются для воздействия на визуальные характеристики поля знаков, начиная со знака, следующего за кодом атрибута поля, и кончая знаком, который предшествует следующему коду атрибута поля, или до конца кадра. Можно задать 6 атрибутов поля (рис. 1.51,б):

1. Мерцание. Вызывает мерцание знаков, следующих за кодом атрибута, осуществляется с помощью сигнала ГШЛ. Частота мерцания равна 1/32 частоты регенерации кадра.

2. Засвечивание. Вызывает засвечивание знаков, следующих за кодом атрибута, выполняется сигналом ПСП.

3. Негативное изображение. Знаки, следующие за кодом атрибута, изображаются негативно, что осуществляется сигналом РЗВ.

4. Подчеркивание. Вызывает подчеркивание знаков, следующих за кодом атрибута, осуществляется сигналом РЗВ.

Таблица атрибутов знаков

Код атрибута знака "СССР"	Положение луча относительно средней строки звукового ряда	Формируемые сигналы на выходах				Формируемое изображение
		АС [1]	АС [0]	ГШЛ	РЗВ	
0000	Выше строки	0	0	1	0	Левый верхний угол
	На строке	1	0	0	0	
	Ниже строки	0	1	0	0	
0001		0	0	1	0	Правый верхний угол
		1	1	0	0	
		0	1	0	0	
0010		0	1	0	0	Левый нижний угол
		1	0	0	0	
		0	0	1	0	
0011		0	1	0	0	Правый нижний угол
		1	1	0	0	
		0	0	1	0	
0100		0	0	1	0	Верхнее пересечение
		0	0	0	1	
		0	1	0	0	
0101		0	1	0	0	Правое пересечение
		1	1	0	0	
		0	1	0	0	
0110		0	1	0	0	Левое пересечение
		1	0	0	0	
		0	1	0	0	
0111		0	1	0	0	Нижнее пересечение
		0	0	0	1	
		0	0	1	0	
1000		0	0	1	0	Горизонтальная линия
		0	0	0	1	
		0	0	1	0	
1001		0	1	0	0	Вертикальная линия
		0	1	0	0	
		0	1	0	0	
1010		0	1	0	0	Пересекающиеся линии
		0	0	0	1	
		0	1	0	0	
1011		0	0	0	0	Не рекомендуется ¹⁾
		0	0	0	0	
		0	0	0	0	
1100		0	0	1	0	Специальные коды
		0	0	1	0	
		0	0	1	0	
1101						Запрещено
1110						
1111						

¹⁾ Код 1011 атрибута знака использовать не рекомендуется. Пока выходы атрибутов неактивны, генератор символов не действует и будут генерироваться неопределенные символы. Коды атрибутов знака 1101, 1110 и 1111 запрещены.

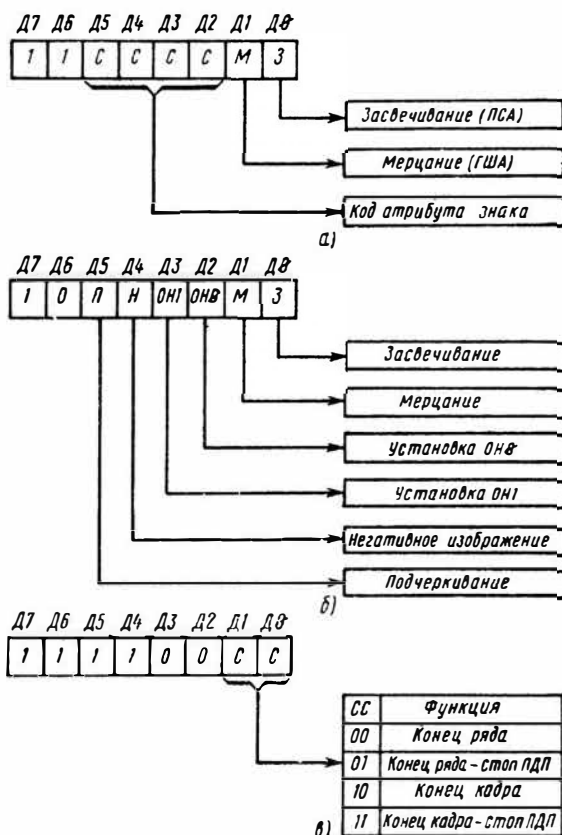


Рис. 1.51. Формат слова:

а — атрибутов знаков; б — атрибутов поля; в — специальных управляющих кодов

5,6. Атрибуты общего назначения. Это два независимо программируемых атрибута поля АН [Ø-1].

Контроллер КР580ВГ75 может быть запрограммирован на видимый и невидимый режимы атрибутов поля. В первом случае все места атрибутов в последовательности знаков на экране отображаются в виде пробелов, обусловленных активизацией сигнала ГШЛ. Выбранные визуальные атрибуты проявляются на знаках после пробела (рис. 1.53,а). Во втором случае включаются регистры стека (16 знаков по 7 разрядов), и когда во вре-

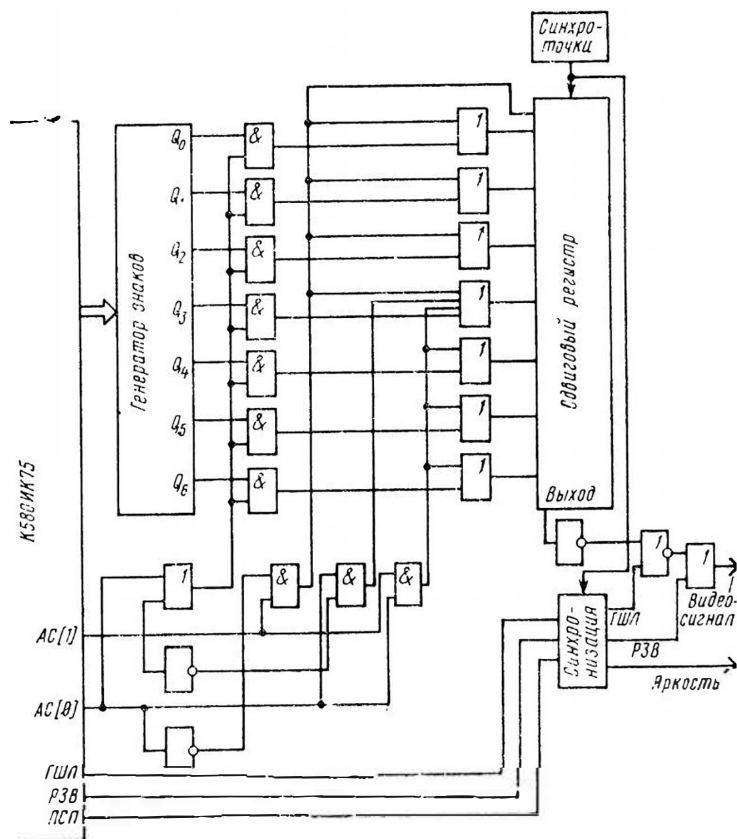


Рис. 1.52. Дополнительные схемы, необходимые для создания графических символов без использования генератора знаков

мя цикла ПДП атрибут поля помещается в буфер ряда, контроллер входного буфера распознает его и загружает следующий знак в соответствующий регистр стека. При отображении атрибут поля помещается в выходной буфер, который распознает его и устанавливает на выходы кода знака СЗ [Ø—6] знак из регистра стека, одновременно активизируются выбранные визуальные атрибуты. В невидимом режиме пробелов на экране нет (рис. 1.53,б). Можно использовать не более 16 атрибутов поля в одной строке, в противном случае первые знаки в регистрах стека теряются, а в слове состояния

устанавливается соответствующий разряд. Так как регистр стека оперирует 7-разрядными кодами знаков, старший разряд любого помещенного в него знака теряется и атрибуты знаков или специальные коды не могут сразу следовать за атрибутом поля. В этой ситуации они будут рассматриваться как обычный отображаемый знак.

На атрибуты знака воздействуют атрибуты поля «Негативное изображение» и «Атрибуты общего назначения» и не воздействуют «Подчеркивание», «Мерцание» и «Засвечивание», однако эти характеристики могут быть запрограммированы индивидуально для атрибутов знака.

Четыре специальных кода (см. рис. 1.51, в) помогают уменьшить объем памяти и упростить управление циклом ПДП. Код «Конец ряда» активизирует сигнал

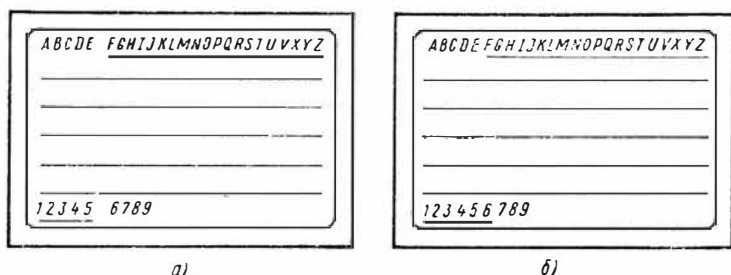


Рис. 1.53. Задание режима отображения атрибутов поля:
а — видимый; б — невидимый

ГШЛ и удерживает его до конца строки; «Конец ряда — стоп ПДП» заставляет логику управления ПДП прекратить запросы ПДП до конца ряда (на дисплей воздействие то же, что и при коде «Конец ряда»); «Конец кадра» активизирует сигнал ГШЛ и удерживает его до конца кадра; «Конец кадра — стоп ПДП» заставляет логику управления ПДП прекратить запросы ПДП до конца кадра (на дисплей воздействие то же, что и при коде «Конец кадра»).

Если «Стоп ПДП» не является последним знаком в пакете или ряду, то ПДП не останавливается до тех пор, пока не считан следующий знак.

Местоположение курсора на экране определяется курсорными регистрами ряда и столбца, которые загружаются в контроллер по команде. Курсор может быть запрограммирован в следующем виде: мерцающее подчеркивание; мерцающий негативный видеоблок; немерцающее подчеркивание; немерцающий негативный видеоблок.

Частота мерцания курсора равна 1/16 частоты регенерации кадра. Если курсор запрограммирован как немерцающий негативный видеоблок в немерцающем негативном поле, то курсор будет отображаться в виде

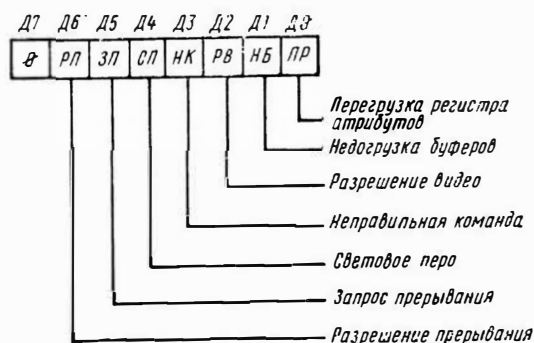


Рис. 1.54. Формат слова состояния контроллера видеотерминала

нормального видеоблока. Если немерцающий подчеркиваемый курсор запрограммирован в немерцающем поле с подчеркиванием, то курсора не будет видно.

Микросхема КР580ВГ75 обеспечивает работу дисплея со световым пером. Световое перо состоит из микропереключателя и микрофотоэлемента. Если световое перо давит на экран, то микропереключатель включает микрофотоэлемент, который защелкивает выход светового пера, когда его достигает развертка луча. Выход светового пера подается на вход ССП контроллера КР580ВГ75, а координаты положения светового пера (ряд и столбец) запоминаются в регистрах светового пера¹. При этом в регистре слова состояния уста-

¹ Содержимое этих регистров может быть считано программным способом. Из-за внутренних и внешних задержек координата пера по строке будет сдвинута на 2—3 позиции. Этот сдвиг должен быть скорректирован программным способом.

навливаются соответствующий разряд, что указывает на обнаружение сигнала светового пера.

Слово состояния контроллера (рис.54, табл. 1.20) может быть считано в любой момент, когда свободна шина данных. Для этого должны быть поданы сигналы $A\bar{O}=I$, $\bar{CT}=0$. После считывания слова состояния все разряды регистра состояний автоматически сбрасываются.

Т а б л и ц а 1.20

Значения разрядов слова состояния БИС КР580ВГ75

Обозначение разряда	Наименование	Функциональное назначение
РП	Разрешение прерывания	Установка и сброс программируются. Автоматически устанавливается командой „Старт дисплей“ и сбрасывается командой „Формат экрана“
ЗП	Запрос прерывания	Устанавливается в начале отображения последующего ряда кадров, если флажок разрешения прерывания установлен
СП	Световое перо	Устанавливается, когда вход ССП светового пера активен и загружены регистры светового пера
НК	Неправильная команда	Устанавливается, когда цепочка параметров слишком длинная или слишком короткая
РВ	Разрешение видео	Указывает, что разрешена подача сигналов на видеовход ЭЛТ
НБ	Недогрузка буферов	Указывает, что поступила команда на отображение, а буфер ряда не загружен полностью
ПР	Перегрузка регистров атрибутов	Устанавливается, если число атрибутов поля в одном ряду больше 16

Программирование контроллера осуществляется семью командами, каждая из которых состоит из байта кода команды и от нуля до четырех байтов параметров. Если контроллер не получит необходимого числа параметров до прихода следующей команды, то установится разряд НК слова состояния, указывающий на неправильно введенную команду.

Инициализация работы контроллера осуществляется с помощью команды ФОРМАТ ЭКРАНА (рис. 1.55). После записи команды ФОРМАТ ЭКРАНА запросы

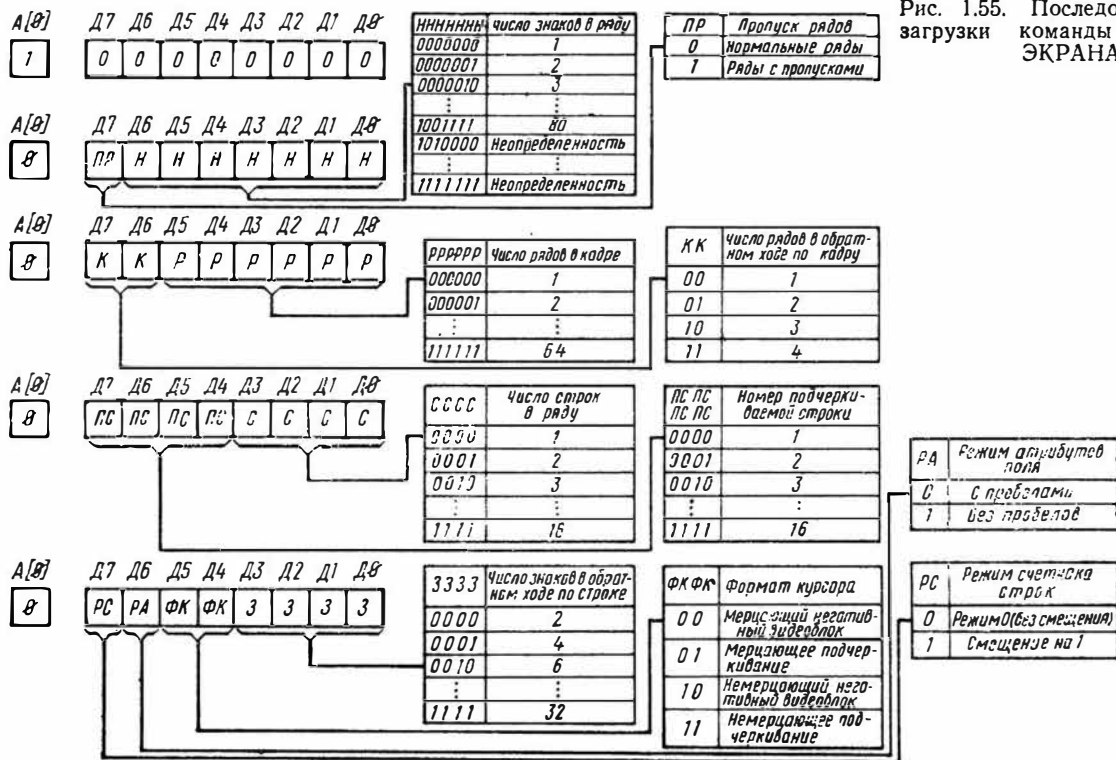


Рис. 1.55. Последовательность загрузки команды ФОРМАТ ЭКРАНА

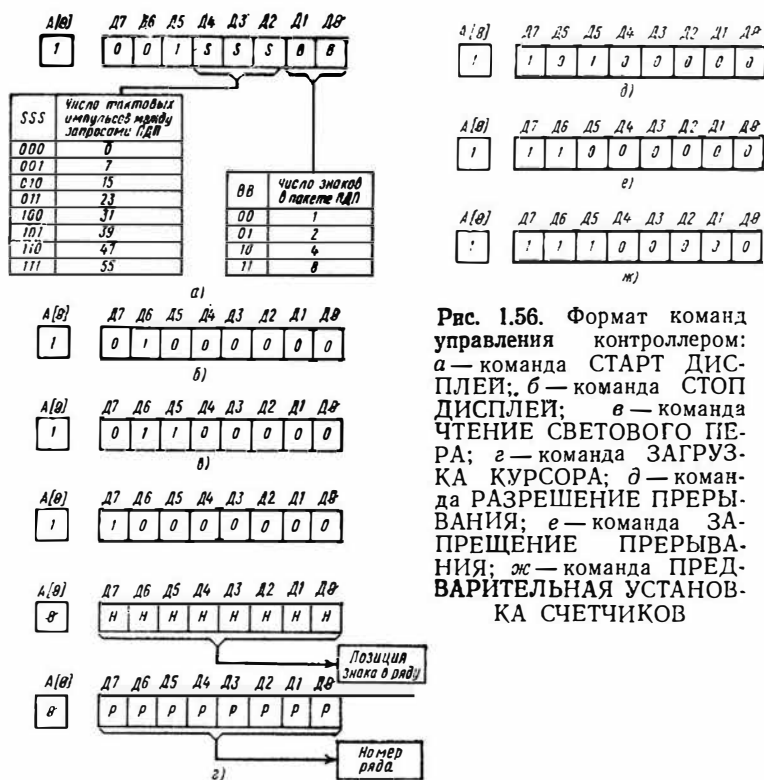


Рис. 1.56. Формат команд управления контроллером: а — команда СТАРТ ДИСПЛЕЙ; б — команда СТОП ДИСПЛЕЙ; в — команда ЧТЕНИЕ СВЕТОВОГО ПЕРА; г — команда ЗАГРУЗКА КУРСОРА; д — команда РАЗРЕШЕНИЕ ПЕРЫВАНИЯ; е — команда ЗАПРЕЩЕНИЕ ПЕРЫВАНИЯ; ж — команда ПРЕДВАРИТЕЛЬНАЯ УСТАНОВКА СЧЕТЧИКОВ

ПДП и прерывания запрещаются, выход ГШЛ бланкирует экран, сигналы ГС и ГК вырабатываются, но являются случайными относительно момента включения питания. После записи байтов параметров формат экрана считается заданным.

Командой СТАРТ ДИСПЛЕЙ (рис. 1.56,а) контроллеру разрешается начинать отображение, при этом прерывания разрешены, генерируются запросы ПДП, разряды РП и РВ регистра состояний устанавливаются. Команда СТОП ДИСПЛЕЙ (рис. 1.56,б) прекращает отображение, при этом прерывания запрещаются, вырабатываются сигналы ГС и ГК, разряд РВ регистра состояний сбрасывается. Возобновление отображения может быть осуществлено новой командой СТАРТ ДИСПЛЕЙ. Команда ЧТЕНИЕ СВЕТОВОГО ПЕРА (рис. 1.56,в) позволяет двумя следующими за ней циклами чтения (при $A[\emptyset] = 0$) вывести в процес-

сор содержимое регистров светового пера: сначала номер знака, затем — номер ряда. Содержимое регистра состояний при этом не изменяется. Команда ЗАГРУЗКА КУРСОРА (рис. 1.56,з) помещает два последовательных байта параметров в регистры позиции курсора. Содержимое регистра состояний при этом не изменяется. Команда РАЗРЕШЕНИЕ ПРЕРЫВАНИЯ (рис. 1.56,д) разрешает прерывание и устанавливает разряд РП регистра состояний. Команда ЗАПРЕЩЕНИЕ ПРЕРЫВАНИЙ (рис. 1.56,е) запрещает прерывания и сбрасывает разряд РП регистра состояний. Командой ПРЕДВАРИТЕЛЬНАЯ УСТАНОВКА СЧЕТЧИКОВ (рис. 1.56,ж) все внутренние счетчики устанавливаются в положение, соответствующее левой верхней позиции экрана, и остаются в этом положении до получения следующей команды. Для выполнения этой команды нужны два тактовых сигнала ТКЗ. Команда полезна при системной отладке и синхронизации группы дисплеев, подключенных к одному процессору.

Глава 2.

Разработка программного обеспечения микропроцессорных систем

Специализация микропроцессорной системы для решения конкретной задачи осуществляется за счет создания программного обеспечения, т. е. программы (или набора связанных программ) в системе команд МП, записываемой в ПЗУ или ППЗУ и реализующей заданный алгоритм функционирования системы. В ряде случаев блоки программного обеспечения могут размещаться во внешних ЗУ и загружаться в ОЗУ микропроцессорной системы по мере надобности.

Разработка программного обеспечения является одним из наиболее длительных и трудоемких этапов проектирования микропроцессорной системы; ее стоимость может достигать до 90% всей стоимости проектирования. Программное обеспечение микропроцессор-

ной системы постоянно совершенствуется и модифицируется, при этом среднегодовые затраты могут превышать 20% его первоначальной стоимости [13].

2.1. Основные понятия в области автоматизации программирования

Для ускорения и облегчения разработки и отладки программного обеспечения микропроцессорных систем широко используются специальные программы, называемые системными. Если системные программы реализованы на той же микропроцессорной системе (микро-ЭВМ), для которой они вырабатывают программное обеспечение, они называются резидентными, в противном случае — кросс-программами. Ниже рассматриваются основные системные программы.

Ассемблер. Машинно-ориентированный язык программирования, в котором используются мнемонические обозначения команд, отражающие их функции, и символические имена переменных. Например, в МП КР580ИК80А команда занесения содержимого ячейки ЗУ с адресом 15 в накопитель имеет код 0011 1010 0000 1111 0000 0000. В программе на языке ассемблера эта команда записывается так: LDA 15 или LDA ADR, если ранее символическое имя ADR присвоено ячейке с адресом 15. Мнемоника команды LDA является сокращением от LOAD ACCUMULATOR (загрузить в накопитель).

Системная программа, транслирующая исходную программу, написанную на языке ассемблера, в коды команд МП (объектную программу) называется ассемблером. При трансляции в объектную программу каждая команда ассемблера преобразуется в одну команду МП. В то же время для хранения исходной программы на языке ассемблера (где каждому символу соответствует один байт кода КОИ-7) требуется на порядок больше места, чем для хранения соответствующей объектной программы.

Совершенные ассемблеры обычно содержат также макропроцессор — системную программу, позволяющую включать в разрабатываемую программу группу команд с помощью присвоенной ей мнемоники. Такой ассемблер называется макроассемблером. Параметризованные макрокоманды макроассемблера повышают эффективность программирования и читабельность программ.

Трансляторы с языков высокого уровня. Языки высокого уровня: Бейсик, Фортран, Паскаль — существенно упрощают написание программы. В качестве операторов используются специальные (ключевые) слова английского или русского языка. Одному оператору на языке высокого уровня соответствует до нескольких десятков команд объектного кода, что сокращает трудоемкость программирования. Трансляторы с языков высокого уровня разделяются на компиляторы и интерпретаторы. Компилятор — это системная программа, транслирующая исходную программу на языке высокого уровня в объектный код. Компиляторы, как правило, являются многопроходными, т. е. требуют для работы нескольких просмотров исходного текста программы.

При большом объеме программы ее текст на языке высокого уровня занимает в ЗУ меньше места, чем объектный код. При специальном кодировании операторов языка высокого уровня программе можно не транслировать в объектный код, а вводить построчно и одновременно выполнять. При этом отпадает необходимость хранения в памяти объектного кода всей программы. Системная про-

грамма, обеспечивающая выполнение наборов команд в соответствии с операторами исходной программы, называется интерпретатором. Наибольшее распространение получили интерпретаторы с языка Бейсик.

Объем программы-интерпретатора обычно невелик и составляет от 2К до 8К байт. Компиляторы требуют для своей работы большого объема ОЗУ (32—64К байт) и ЗУ магнитных дисках. Резидентный макроассемблер занимает около 4К байт памяти.

Ассемблеры и компиляторы могут генерировать код объектной программы в абсолютном и перемещаемом форматах. В первом случае в коды команд объектной программы компилятором заносятся абсолютные адреса операндов и переходов, во втором — относительные по отношению к базовым адресам программных сегментов адреса. Перемещаемая объектная программа может быть загружена для исполнения в любую область памяти, для этого программой-загрузчиком должна быть выполнена соответствующая коррекция относительных адресов.

Редактор текста. Системная программа, которая в соответствии с командами программиста, как правило, вводимыми с терминала, вносит исправления в текст исходной программы, находящейся в памяти ЭВМ. Для хранения исходной программы в программе-редакторе предусматривается буфер объемом до 32К байт. Если исходная программа не помещается в буфер одновременно, то ее редактирование осуществляется по частям. Исправленная исходная программа выдается на перфоленгу или магнитный носитель для дальнейшего использования.

Обычно редактор позволяет удалять строку или строки, изменять порядок строк, стирать информацию в буфере, вводить сегмент программы в буфер и выводить его, добавлять текст в пустой или неполностью заполненный буфер, находить в тексте строки, совпадающие с заданной, и выдавать их номера на печать или заменять их на другие, изменять знаки в строке, не набирая ее полностью.

Загрузчик и редактор связей. Загрузчик — это системная программа, с помощью которой объектная программа вводится в память микропроцессорной системы. Если загружаемый объектный код является абсолютным, то он записывается по адресу, определенному при ассемблировании. Эта операция выполняется абсолютным загрузчиком, который весьма прост. Он может использоваться как самостоятельная единица, но часто входит в отладочный монитор. Если объектный код является перемещаемым; загрузчик должен осуществлять преобразование всех относительных адресных ссылок в абсолютные и осуществлять загрузку полученной абсолютной объектной программы в требуемую область памяти.

При разработке больших программ обычно удобнее разбить программу на части (модули), которые компилируются отдельно. Полученные перемещаемые объектные модули затем объединяются в общую программу. Объединение объектных модулей осуществляется программой-редактором связей, результат работы которой заносится в память с помощью загрузчика (рис. 2.1). Связывающий загрузчик одновременно выполняет функции редактора связей и заносит результирующую программу в память.

Отладочный монитор. При пробном выполнении программы могут быть обнаружены ошибки. Назначение отладочного монитора состоит в ускорении и облегчении поиска и исправления ошибок в объектной программе. Монитор обеспечивает возможность остано-

ва выполнения отлаживаемой объектной программы в любой наперед заданной точке (точке разрыва). После останова программы монитор позволяет выводить на экран или на печать содержимое регистров и памяти МП, модифицировать их содержимое, в чем и заключается отладка. Отладочные мониторы позволяют выполнять программу также и в пошаговом режиме, т. е. с остановом после выполнения каждой команды.

Прерывание отлаживаемой программы в точке разрыва может быть выполнено несколькими способами [8]. Одним из таких способов является замена команды программы с адресом $N+1$ командой перехода к специальной подпрограмме монитора. При этом команда из ячейки $N+1$ временно запоминается в некоторой буферной ячейке. После передачи управления монитору команда в ячейке $N+1$ восстанавливается. Естественно, что отладочный монитор должен находиться в памяти микропроцессорной системы вместе с отлаживаемой программой.

Эмулятор. Это системная программа для мини- или больших ЭВМ, которая моделирует выполнение команд микропроцессорной системы. С помощью эмулятора можно отлаживать программы для микропроцессорных систем при отсутствии самих систем. Эмулятор

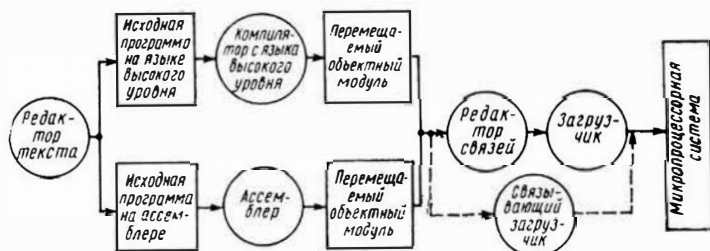


Рис. 2.1. Последовательность использования системных программ при разработке программного обеспечения микропроцессорных систем

также позволяет выполнять те же отладочные действия, что и монитор. Для эмуляторов характерно замедление выполнения отлаживаемой программы относительно реального времени в 10—20 раз [9, 13].

Дисковая операционная система. Операционной системой называется система программ, обеспечивающая взаимодействие ЭВМ с пользователем и выполняющая его команды. Ядром системы является управляющий монитор или супервизор, обеспечивающий прием команд пользователя и запуск соответствующих системных программ: трансляторов, редакторов, загрузчиков, которые также входят в операционную систему. Дисковой операционной системой (ДОС) называется операционная система, большая часть которой постоянно размещается на магнитном диске (для микро-ЭВМ на гибком диске). В полупроводниковой памяти (как правило, в ПЗУ) располагается только управляющий монитор ДОС. Он загружает с диска в ОЗУ те части операционной системы, которые необходимы для выполнения задаваемых пользователем команд ДОС.

2.2. Программирование на языке ассемблера МП КР580ИК80А

Программно-доступные элементы. В микропроцессорной системе на основе серии КР580 для программиста являются доступными [15, 19] 16-разрядный счетчик команд (PC), регистр-накопитель (A), шесть 8-разрядных регистров (B, C, D, E, H, L), однобитовые признаки результатов, 16-разрядный указатель стека (SP), триггер разрешения прерывания, каналы ввода — вывода, память.

Память представляется как последовательность ячеек размером один байт. Каждая ячейка имеет свой адрес в диапазоне от 0 до 65 536. При программировании на ассемблере обычно используется шестнадцате-

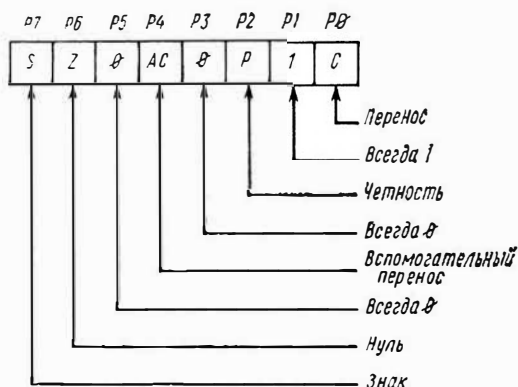


Рис. 2.2. Формат байта признаков результата

ричное значение адреса, тогда диапазон адресации составляет $0000\text{—}FFFF$. При считывании или записи информации в ячейку указывается ее адрес.

Каждая микропроцессорная система имеет свою структуру памяти. Количество ячеек памяти в специализированной системе часто меньше максимально возможного, причем в ряде случаев адреса имеющихся ячеек могут располагаться не подряд. Например, адреса могут иметь значения от 0000 до $0FFF$ и от $F800$ до $FFFF$. Кроме того, только часть памяти может быть выполнена в виде ОЗУ, остальная часть может представлять собой ПЗУ. При написании программы разработчик должен использовать адреса только реально имеющихся ячеек и следить за тем, чтобы запись ин-

формации осуществлялась только по адресам, соответствующим ОЗУ.

В ассемблере всем программно-доступным элементам поставлены в соответствие числа, использование которых возможно для обращения к соответствующим элементам:

Мнемоническое обозначение элемента...BCDEHLMA

Цифровой эквивалент обозначения...Ø 1 2 3 4 5 6 7

Цифрой 6 кодируется ячейка памяти, адрес которой находится в регистрах H, L (в H — старший байт адреса, в L — младший). Хотя ячейка ЗУ не является РОН, обозначение М (Память) приведено вместе с обозначениями регистров, так как ячейка ЗУ и РОН используются в одних и тех же командах.

Ряд команд МП обращается к парам регистров, что дает возможность оперировать 16-разрядными данными и использовать пары регистров (табл. 2.1) для косвенной адресации.

Таблица 2.1

Обозначения пар регистров

Обозначение регистров	Старший регистр пары	Младший регистр пары	Обозначение регистров	Старший регистр пары	Младший регистр пары
B D H	B D H	C E L	PSW SP	A Старший байт	Байт признаков результата Младший байт

С помощью сокращения PSW (Слово состояния программы) производится обращение к накопителю и байту, составленному из однокбитовых признаков результата (рис. 2.2). Пять однокбитовых признаков: C (Перенос), AC (Вспомогательный перенос), S (Знак), Z (Ноль), P (Четность) — устанавливаются в зависимости от результата операции при выполнении арифметических и логических команд (см. приложение 1).

Указатель стека SP является единым 16-разрядным регистром, обращение к нему осуществляется как к паре регистров.

Стек — это область ОЗУ микропроцессорной системы, последовательность считывания данных из которой обратна последовательности их записи. Иными словами,

первым может быть считано только число, записанное последним. Операции занесения и извлечения данных из стека широко используются при написании прикладных программ, а также необходимы при работе с подпрограммами и обслуживании прерываний. Стек по усмотрению программиста может быть размещен в любом месте ОЗУ. При этом в указатель стека SP заносится адрес последней занятой ячейки стека. В качестве информации при записи может выступать содержимое пары регистров или счетчика команд. На рис. 2.3 показана операция занесения в стек содержимого пары

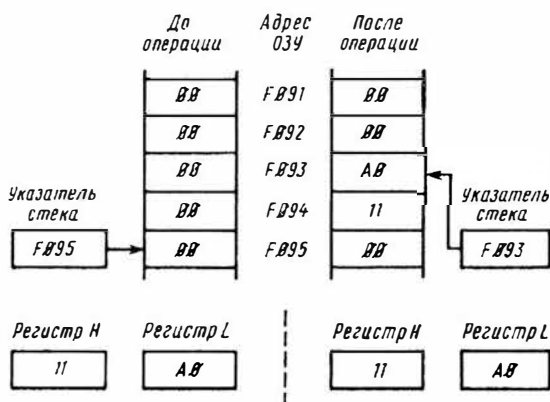


Рис. 2.3. Операция занесения в стек содержимого пары регистров

регистров H, L. При выполнении этой операции выполняются следующие действия: старший байт данных запоминается в ОЗУ по адресу, на единицу меньшему содержимого указателя стека; младший байт данных запоминается по адресу, меньшему содержимого указателя стека на два; содержимое указателя стека уменьшается на два. Таким образом, стек растет в сторону младших адресов. При выборке данных из стека содержимое двух его последних ячеек помещается в нужные регистры, а содержимое указателя стека увеличивается на два.

Если триггер разрешения прерывания МП установлен в единицу, то прием прерываний разрешен, если он установлен в нуль, то процессор на сигналы прерывания

не реагирует. Установка триггера в единичное состояние осуществляется командой EI (РАЗРЕШИТЬ ПРЕРЫВАНИЕ), в нулевое состояние — командой DI (ЗАПРЕТИТЬ ПРЕРЫВАНИЕ). Кроме того, триггер устанавливается в нуль при выполнении команды RST (ПОВТОРНЫЙ ЗАПУСК).

Микропроцессорная система на основе МП КР580ИК80А может иметь до 256 восьмиразрядных каналов ввода и до 256 восьмиразрядных каналов вывода. Каждый канал ввода и вывода имеет адрес от 0 до 255. Обращение к ним осуществляется командами IN (ввести байт канала ввода в накопитель) и OUT (вывести байт из накопителя в канал). В этих командах всегда указывается адрес соответствующего канала.

Структура строки ассемблера. Строка программы на ассемблере может содержать следующие поля: метку, команду или псевдокоманду, операнд или операнды, комментарии.

Команды ассемблера соответствуют командам МП, псевдокоманды задают информацию, необходимую для трансляции программы на ассемблере в объектный код. Мнемоники команд МП КР580ИК80А приведены в приложении 1.

Каждая команда или псевдокоманда может иметь метку (для ряда псевдокоманд наличие метки обязательно). Меткой может служить последовательность, начинающаяся с латинской буквы или символов @ и ? и содержащая не более пяти символов. В качестве символов метки (кроме первого символа) используются цифры и латинские буквы. Непосредственно за меткой ставится двоеточие. Метка предшествует команде или псевдокоманде. Мнемоники команд, псевдокоманд, имена регистров и пар регистров, обозначения арифметических и логических операторов не могут использоваться в качестве меток. Приведем примеры помеченных команд:

```
LAB: MOV A, B
      ?B11C: LDA1111H
```

```
@ : RLC
```

Поскольку метки выступают в качестве адреса команды, которую они помечают, то в поле метки каждая

из них должна встречаться не более одного раза. При наличии у одной команды или псевдокоманды нескольких меток каждая из них начинается с новой строки:

LAB1:

LAB2: RAR

В этом случае при передаче управления как на метку LAB1, так и на метку LAB2 следующей будет выполняться команда RAR.

В поле команды указывается ее мнемоника (см. приложение 1). Команда может иметь один или два операнда, обозначения которых отделяются от мнемоники команды одним или более пробелами. При наличии двух операндов они разделяются запятой. Для каждого операнда каждой команды МП КР580ИК80А возможен только один определенный метод адресации. В зависимости от способа адресации для задания операнда необходимо указать регистр, пару регистров, непосредственный байт данных, два непосредственных байта данных, двухбайтовый адрес (при обращении к ячейке памяти) или однобайтовый адрес (при обращении к устройству ввода или вывода) в поле операндов команды.

Операнд (адрес операнда) может задаваться одним из шести способов: константой, счетчиком размещения, константой в коде КОИ-7, меткой команды, меткой величины, выражением. Константа может быть шестнадцатеричной, десятичной, восьмеричной или двоичной. За каждой шестнадцатеричной константой должна следовать латинская буква H. Число должно начинаться с цифры. Если первой цифрой являются шестнадцатеричные A, B, C, D, E, F, то перед ними добавляется незначащий нуль. Например, JUMP $\emptyset F \emptyset 1 \emptyset H$ (Переход по адресу $F \emptyset 1 \emptyset_{16}$); MVI B, 22H (Занести в регистр B число 22_{16}). При использовании десятичных констант после них указывается латинская буква D, которая может и опускаться. Например, STA 32841D (Запомнить содержимое накопителя по адресу 32 841); CPI 28 (Сравнить содержимое накопителя с числом 28). Восьмеричная константа имеет суффикс O или Q. Например, IN 62Q (Ввести данные канала 62_8); LDAX SP $1 \emptyset 771 \emptyset$. (Занести в указатель стека число $1 \emptyset 771_8$). За двоичной константой всегда следует латинская буква B. Например, ANI $\emptyset 1111111 B$ (Поразрядное И с числом $\emptyset 1111111_2$). Если операнд, задаваемый константой, по смыслу команды является однобайтовым, то значение

константы должно быть от 0 до 255. При задании двухбайтовых операндов константами малой величины старшие разряды операнда заполняются нулями.

Счетчик размещения задается символом \$ его значение равно адресу текущей команды. Например, JMP \$ +20 означает передачу управления на адрес, следующий через 20 байтов после текущего значения счетчика команд.

При задании операнда константой в коде КОИ-7 указывается один или два символа (букв, цифр, служебных символов), которые заключаются в одинарные кавычки (апострофы). Двойной апостроф может быть использован для представления одного апострофа внутри константы. Для задания однобайтового операнда задается один символ, двухбайтового — два. Например, MVI C, '+' (Занести в регистр C код символа плюс); LXI D, 'AB' (Занести в пару регистров D, E коды букв A и B).

В трехбайтовых командах с прямой адресацией вместо адреса могут указываться метки команд. Например, JMP ADDR (Безусловный переход на метку ADDR). При этом необходимо, чтобы какая-либо команда была помечена меткой ADDR:

Язык ассемблера позволяет присваивать мнемонические обозначения (метки) числовым величинам. Правила написания этих мнемонических обозначений аналогичны правилам написания меток команд. Пусть, например, с помощью псевдокоманд ассемблера мнемонике (метке) ABC присвоено значение 1FH, а мнемонике K11 — значение 0FFFFH. Тогда команда ADI ABC складывает 1FH с содержимым накопителя, а команда LXI B, K11 заносит FFFFH в пару регистров B, C.

Операнды могут также задаваться выражениями. Выражения состоят из констант, обозначения счетчика размещения, констант в коде КОИ-7 и меток, связанных арифметическими и логическими операторами, а также круглых скобок. Используются следующие операторы: + (сложение), - (вычитание и знак минус), * (умножение), / (деление), MOD (остаток от деления), NOT (поразрядное отрицание), AND (поразрядное И), OR (поразрядное ИЛИ), XOR (поразрядное исключающее ИЛИ), SHR (сдвиг вправо), SHL (сдвиг влево). Все операции предполагают, что аргументы являются 16-разрядными двоичными целыми числами без знака и дают

В результате также 16-разрядное двоичное целое число без знака. Действия операторов +, —, *, /, NOT, AND, OR, XOR аналогичны общепринятым. При выполнении операции $X \text{ MOD } Y$ результатом является остаток от деления X/Y . Операторы SHR и SHL осуществляют сдвиг первого операнда вправо или влево на число позиций, указанное во втором операнде. Освободившиеся старшие или младшие биты заполняются нулями.

Операторы выражений вычисляются в следующем порядке:

1. Выражения в скобках.
2. *, /, MOD, SHL, SHR
3. +, —
4. NOT
5. AND
6. OR, XOR

Операторы MOD, SHL, SHR, NOT, AND, OR, XOR должны отделяться от операндов, по крайней мере, одним пробелом. Необходимо, чтобы результат вычисления выражения соответствовал требуемой разрядности операнда, например, второй операнд команды MVI должен содержать 8 двоичных разрядов. Поэтому запись $\text{MVI H, NOT } \emptyset$ содержит ошибку, так как операция $\text{NOT } \emptyset$ дает число 111111111111111B. Однако запись $\text{MVI H, NOT } \emptyset \text{ SHR } 8$ верна, так как старший байт результата равен нулю.

Ниже приведены примеры правильно записанных выражений.

1. MOV K/2, REG1 . Если мнемоникам REG1 и K присвоены значения 4 и 5 соответственно, то выполняется пересылка данных из регистра H в регистр D.

2. $\text{SUI } 32+8\emptyset/2$ обеспечивает вычитание числа 72 из содержимого накопителя.

3. $\text{SHLD BASE} - 3 * (1\emptyset\emptyset - \text{K112})$. Если метки BASE и K112 имеют значения $3\emptyset\emptyset\emptyset$ и 19 соответственно, то содержимое регистров H, L заносится в память по адресу 2757 (содержимое регистра L по адресу 2757, а регистра H по адресу 2758).

4. $\text{CPI (NOT (M1 AND } \emptyset\text{FH) SHL } 4 \text{ XOR } 55\text{H) AND } \emptyset\text{FFH}}$. Если метке M1 присвоено значение 2FC4H, то с содержимым накопителя будет сравниваться величина $\emptyset\text{EАН}$.

Поле комментариев может содержать любой пояснительный текст, который ассемблером не воспринимается,

а появляется только в распечатках. Перед комментарием ставится точка с запятой. Строка программы на ассемблере может содержать только комментарий, только метку, только команду с операндами или без них, а также любое сочетание указанных единиц. Слева и справа от текста в строке, а также между указанными полями допустимо любое число пробелов.

Псевдокоманды. Для каждой символически обозначенной переменной, используемой в программе, необходимо зарезервировать одну или несколько ячеек памяти. Выполняется это с помощью псевдокоманды DS (Определить память), имеющий следующий формат:

<метка> : DS <выражение>

причем значение выражения может быть от Ø до ØFFFFH. Значение выражения определяет число байтов памяти, которые должны быть зарезервированы для переменной. Так любая из приводимых ниже псевдокоманд резервирует 11 байтов под именем M1 (если ранее метке K присвоено значение 8):

M1: DS 11
M1: DS ØBH
M1: DS K+3
M1: DS 2*K-5

При обработке псевдокоманды DS содержимое зарезервированных ячеек памяти не определено. Все метки, встречающиеся в выражении, должны быть определены до их появления в псевдокоманде DS. Наличие метки в этой псевдокоманде не обязательно.

Псевдокоманда DB (ОПРЕДЕЛИТЬ БАЙТ) резервирует некоторое число байтов памяти и определяет их начальное содержимое. Псевдокоманда имеет следующий формат:

<метка> : DB <список>

где <список> — это перечисляемые через запятую выражения или строки символов, заключенные в одинарные кавычки (апострофы). Псевдокоманда вычисляет значение выражений (значение результата должно находиться в диапазоне от Ø до 255) и помещает их в последовательные байты памяти. При наличии в списке строки символов, заключенной в кавычки, в последовательные байты памяти помещаются коды КОИ-7 этих символов. Наличие метки в псевдокоманде не обяза-

тельно. В приводимом ниже примере предполагается, что метке K ранее присвоено значение 12, метке F1 — значение 28H, а метка M2 соответствует адресу 00F0H.

Текст на языке ассемблера	Шеснадиатеричный адрес	Ассемблированные данные (шеснадиатеричный код)
M2: DB 0F1H	00F0	F1
DB K+10, 1FH-3/2, 'A'	00F1	16
	00F2	1E
	00F3	41
	00F4	42
LAB: DB \$+10 AND 0FFH, (\$+10) SHR8	00F5	FF
	00F6	00

Псевдокоманда DW (ОПРЕДЕЛИТЬ СЛОВО) аналогична псевдокоманде DB, за исключением того, что она формирует двухбайтовые слова данных. Формат псевдокоманды DW:

<метка> : DW <список>

где <список> — это перечисляемые через запятую выражения, значения которых от 0 до 0FFFFH. При занесении значения выражения в память по адресу, соответствующему метке, заносится младший байт результата, а по следующему адресу — старший байт. Наличие метки у псевдокоманды не обязательно.

В приведенном ниже примере предполагается, что метке K2 ранее присвоено значение 1F00H, а метке K3 — значение 1000H. Метке JMТAB соответствует адрес 1800H.

Текст на языке ассемблера	Шеснадиатеричный адрес	Ассемблированные данные (шеснадиатеричный код)
JMТAB: DW K2, K2+5, K2+10	1800	00
	1801	1F
	1802	05
	1803	1F
	1804	0A
	1805	1F
DW \$+(K3 MOD 7*5)	1806	0B
	1807	18

Псевдокоманда ORG (НАЧАЛО) имеет следующий формат:

<метка> : ORG <выражение>

Значение выражения должно лежать в пределах $\emptyset$ — $\emptyset$ FFFFH. Все метки, входящие в выражение, должны быть определены до псевдокоманды ORG. Метка в псевдокоманде обязательна. При обработке псевдокоманды счетчик адреса принимает значение выражения. Следующая генерируемая ассемблером команда или байт данных будет размещаться по адресу равному значению выражения. Обычно псевдокоманда ORG располагается в начале программы и задает ее начальный адрес. Если программа не содержит в начале псевдокоманды ORG, то она размещается, начиная с нулевого адреса.

Псевдокоманда ORG может встречаться и в теле программы. Так, например, ниже она выполняет функцию псевдокоманды DS.

Текст на языке ассемблера (вариант 1)	Текст на языке ассемблера (вариант 2)	Шестнадцатеричный адрес	Ассемблированные данные (шестнадцатеричный код)
ORG 2C00H	ORG 2C00H	2C00	06
MVI B, 0FFH	MVI B, 0FFH	2C01	FF
MP LAB	JMP LAB	2C02	03
		2C03	09
		2C04	2C
DS 4	ORG \$ +4	2C05	
		2C06	
		2C07	
		2C08	
LAB: XRA A	LAB: XRA A	2C09	AF

Псевдокоманда EQU (РАВНО), имеющая формат

<метка> EQU <выражение>

присваивает метке числовое значение, равное значению выражения.

Так, в приведенном выше примере на использование псевдокоманды DW меткам K2 и K3 могут быть присвоены значения следующим образом:

K2 EQU 1F00H

K3 EQU 1000H

Метка в псевдокоманде EQU является обязательной, причем после нее не ставится двоеточие. Каждая метка может появляться только в одной псевдокоманде EQU, т. е. ее нельзя переопределить.

Псевдокоманда SET (УСТАНОВИТЬ), имеющая формат

<метка> SET <выражение>

присваивает метке числовое значение, равное значению выражения. Действия этой псевдокоманды аналогично действию псевдокоманды EQU, за исключением того, что значения метки могут быть переопределены.

Текст на языке ассемблера	Шестнадцатеричный адрес	Ассемблированные данные (шестнадцатеричный код)
NUM SET 5		
MVI A, NUM	1000	3E
.	1001	05
NUM SET 0FH-2	.	.
MVI A, NUM	1100	8E
	1101	0D

Так же, как в псевдокоманде EQU, в псевдокоманде SET метка обязательна и двоеточие после нее не ставится.

Все метки, входящие в выражения псевдокоманд ORG, EQU и SET, должны быть определены до того, как они появляются в указанных псевдокомандах.

Псевдокоманда END (КОНЕЦ) имеет формат

<метка> : END <выражение>

Псевдокоманда указывает на конец текста программы. Значение выражения в этой псевдокоманде определяет пусковой адрес программы. Если этот адрес опущен, то предполагается, что он равен нулю. Метка не является обязательной.

Примеры программ на ассемблере микропроцессора КР580.

1. Подсчет числа единиц в байте. Программа располагается, начиная с ячейки 1000. Предполагается, что исследуемый байт заносится в накопитель А заранее. В примере использован стандартный метод организации цикла. Вначале в регистр В (счетчик цикла) заносится число повторений цикла. После каждого выполнения тела цикла, т. е. сдвига содержимого А влево и увеличения на единицу содержимого счетчика единиц (регистра С), если это необходимо, осуществляется уменьшение на единицу содержимого регистра В. При отличном от нуля результате (признаке Z, установленном в нуль) осуществляется переход на начало цикла (метку M1), при нулевом результате (признаке Z, установленном в нуль) выполнение цикла завершается. Текст программы приведен ниже.

```

; исходный байт в А, результат в С
; регистр В используется в качестве счетчика цикла
ORG 1000
MVI C,0; обнуление счетчика единиц

```

```

MVI B,8; загрузка счетчика цикла
M1: RLC
    JNC M2
    INR C; добавление единицы к счетчику единиц
M2: DCR B; уменьшение на единицу содержимого счетчика цикла
    JNZ M1
    HLT
    END 100

```

2. Подпрограмма сложения двух 8-разрядных десятичных чисел. Для кодирования одного десятичного разряда необходимо 4 бита, поэтому два десятичных разряда занимают 1 байт. Восьмиразрядное десятичное число занимает 4 байта, причем младшие два разряда занимают байт с адресом ADDR следующие два разряда — байт с адресом ADDR+1 и т. д. Текст подпрограммы имеет вид:

```

; подпрограмма сложения двух 8-разрядных десятичных чисел
S1: LXI H, ADDR1 ; занесение адреса 1-го числа в H, L
    LXI D, ADDR2 ; занесение адреса 2-го числа в D, E
    XRA A ; сброс признака переноса
    MVI C, 4 ; загрузка счетчика цикла
M1: LDAX D ; занесение в A двух разрядов 2-го числа
    ADC M ; прибавление двух разрядов 1-го числа
    ; с учетом переносов из предыдущих раз-
    ; рядов
    DAA ; преобразование результата в десятич-
    ; ный код
    STAX D ; запоминание двух разрядов результата
    INX H ; получение в H, L и D, E адресов сле-
    ; дующих двух разрядов слагаемых
    DCR C ; уменьшение содержимого счетчика цик-
    ; ла
    JNZ M1 ; переход на начало цикла
    RET ; возврат из подпрограммы
;
;
ADDR1: DS 4 ; место размещения 1-го числа
ADDR2: DS 4 ; место размещения 2-го числа

```

Сложение пары десятичных разрядов осуществляется в цикле, начиная с младших разрядов. Результат помещается на место второго числа (адрес ADDR2). Счетчиком цикла служит регистр C. После сложения с учетом переноса из предыдущего разряда (команда ADC) для получения двоично-кодированного десятичного кода результата производится десятичная коррекция (команда DAA). Перед началом цикла командой XRA A (поразрядное исключающее ИЛИ содержимого накопителя самого с собой) сбрасывается признак переноса C. Эта же команда часто используется при программировании для загрузки числа нуль в накопитель, так как ее результат всегда равен нулю. Команда XRA A занимает 1 байт и выполняется за 4 такта, в то время как команда MVI A, 0 (Посылка нуля в накопитель) занимает 2 байта и выполняется за 7 тактов.

3. Программа управления светофором. На перекрестке дорог 1 и 2 установлен светофор, который функционирует по жесткой программе (табл. 2.2). Микропроцессорная система управляет светофором подачей на него управляющего слова (рис. 2.4) через выходной канал с адресом 0. При единичном значении бита соответствующий

Таблица 2.2

Алгоритм работы светофора

Номер состояния	Дорога 1	Дорога 2	Время, с
1	Зеленый	Красный	30
2	Желтый	Красный и желтый	5
3	Красный	Зеленый	15
4	Красный и желтый	Желтый	5

сигнал зажигается, при нулевом — гаснет. Для отсчета временных интервалов предусмотрен таймер, подключенный к выходному каналу 1. Через этот канал МП осуществляет запуск таймера подачи

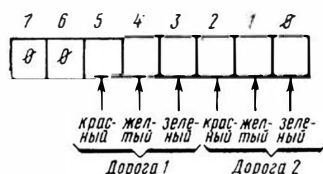


Рис. 2.4. Формат управляющего слова



Рис. 2.5. Пример набора логических функций, вычисляемых с помощью подпрограммы PLA

на его вход временного интервала в секундах. После окончания временного интервала на выходе таймера появляется сигнал, который поступает на нулевой входной канал микропроцессорной системы.

Программа управления светофором размещается с нулевого адреса. Ее текст имеет вид:

```

; программа управления светофором
LIGHT EQU 0 ; канал подачи управляющего слова
CLOCK EQU 1 ; канал запуска таймера
ALARM EQU 0 ; канал приема сигнала с таймера
LOOP: LXI B, COL
      LXI D, TIME
      MVI H, 4 ; загрузка счетчика цикла
M1: LDAX B ; загрузка в A управляющего слова
    OUT LIGHT ; выдача управляющего слова
    LDAX D ; загрузка в A значения интервала
M2: OUT CLOCK ; запуск таймера
    IN ALARM ; ввод сигнала с выхода таймера
    ANA A ; установка признака нуля
    JZ M2 ; переход на повторный ввод при нуле
    INX B ;
    INX D ;
    DCR H ; уменьшение содержимого счетчика
      ; цикла
    JNZ M1
    JMP LOOP ; переход на начало программы
TIME: DB 30,5,15,5 ; временные интервалы для четырех со-
                  ; стояний
COL: DB 14Q,26Q,41Q,62Q ; управляющие слова для четырех со-
                  ; стояний
END

```

Выполнение программы прерывается только при выключении питания.

4. Подпрограмма моделирования программируемой логической матрицы (ПЛИМ) [15]. Подпрограмма PLA, моделирующая работу ПЛИМ, позволяет вычислять до восьми логических функций восьми и менее переменных. Вектор значений входных переменных перед обращением к подпрограмме помещается в накопитель A. Вектор вычисленных значений логических функций F (рис. 2.5) в конце работы подпрограммы также размещается в A. Каждая функция представляет собой логическую сумму термов. В подпрограмме производится последовательное вычисление термов, входящих хотя бы в одну функцию. Для каждого терма с номером i из вектора входных переменных с помощью поразрядной операции И выделяются значимые переменные (рис. 2.6). Так, если терм $\overline{b}c$ включает только вторую и третью переменные, то двоичное значение выделяющей маски $D_i = 01100000$. Далее разряды результата, соответствующие вхождению в терм переменных без инверсии, инвертируются путем поразрядной операции Исключающее ИЛИ с байтом T_i . Так, при вычислении терма $\overline{b}c$ байт $T_i = 00100000$. Нулевой результат операции Исключающее ИЛИ указывает на единичные зна-

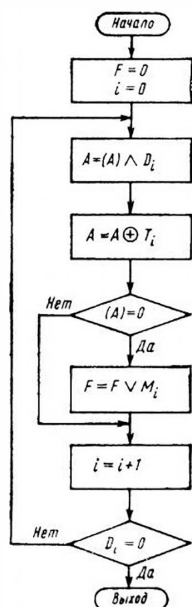


Рис. 2.6. Алгоритм вычисления логической функции

чения вычисленного термина. В этом случае в вычисляемый вектор логических функций F с помощью поразрядной операции ИЛИ добавляется байт M_i . Этот байт имеет единицы в двоичных разрядах, соответствующих функциям, в которые входит вычисленный терм. Так, если терм bc входит в первую, пятую и седьмую функции, то $M_i = 10001010$. Процесс повторяется для всех термов, входящих хотя бы в одну из вычисляемых функций, причем для каждого термина задаются свои D_i , T_i , M_i .

Текст подпрограммы имеет вид:

```

; подпрограмма PLA
; программный аналог программируемой логической матрицы на во-
; семь входов и восемь выходов
; вектор переменных в A, вектор вычисленных функций в A

PLA:      MOV B,A      ; запомнить аргумент в B
          MVI C,0      ; загрузить нуль в C для последующего
                      ; формирования результата
NEXT:     MOV A,B      ; восстановление аргумента в A
          ANA M        ; выделение значимых переменных
          INX H
          XRA M        ; в результате нуль, если терм равен
                      ; единице
          INX H
          JNZ CHECK    ; переход, если значение термина равно
                      ; нулю
          MOV A,C
          ORA M        ; логическое добавление единиц в функ-
                      ; ции, включающие вычислительный терм
CHECK:    MOV C,A
          INX H
          SUB A        ; загрузка нуля в A
          ADD M        ; установка бита нуля по значению сле-
                      ; дующей маски значимых переменных
          JNZ NEXT     ; переход к вычислению следующего тер-
                      ; ма, если его маска отлична от нуля
          MOV A,C      ; посылка вычисленного вектора функ-
                      ; ций в A
          RET

```

Для приведенного на рис. 2.5 набора логических функций константы D_i , T_i и M_i задаются следующим образом:

TABL: DB 0C0H,0C0H,98H	; D,T,M	для термина	ab
DB 60H,20H,8AH	; D,T,M	»	bc
DB 81H,81H,40H	; D,T,M	»	ah
DB 0D4H,10H,52H	; D,T,M	»	abd
DB 0EH,0EH,20H	; D,T,M	»	efg
DB 0BH,03H,20H	; D,T,M	»	egh
DB 03H,01H,30H	; D,T,M	»	qh
DB 1FH,1FH,0AH	; D,T,M	»	$defgh$
DB 0F0H,0F0H,05H	; D,T,M	»	$abcd$
DB 00H			

Признаком того, что все термины вычислены и значения функций установлены, является равенство очередного D_i нулю (последняя псевдокоманда DB). Перед обращением к подпрограмме PLA не-

обходимо в регистры H и L занести адрес таблицы TABL с помощью LXI H, TABL.

В качестве другого примера применения рассмотренной подпрограммы укажем изменение порядка следования двоичных разрядов в слове на обратный, используемое при выполнении быстрого преобразования Фурье. Логические функции в этом случае задаются следующим образом:

```
FFT:  DB 128, 128, 1
      DB 64, 64, 2
      DB 32, 32, 4
      DB 16, 16, 8
      DB 8, 8, 16
      DB 4, 4, 32
      DB 2, 2, 64
      DB 1, 1, 128
      DB 0
```

Обращение к подпрограмме осуществляется так:

```
LXI H, FFT
CALL PLA
```

Макрорасширения и условное ассемблирование.

В реальной программе часто встречаются одинаковые группы команд, отличающиеся операндами. В таких случаях повторяющейся в различных местах программы группе команд присваивается некоторое имя. Если после этого в программе в качестве команды будет указано это имя, то автоматически вместо него будет поставлена требуемая группа команд. Присвоение группе команд имени называется макроопределением, а употребление этого имени при написании программы — макровыводом. Примером макроопределения является:

```
SHF      MACRO REG, N      ; заголовок макроопределения
        MVI  REG, N        ; загрузка счетчика циклов
M1:      RRC                ; циклический сдвиг вправо
        ANI  7FH            ; очистить старший бит
        DCR  REG            ; уменьшение на единицу содержимого
        JNZ  M1             ; жимого счетчика циклов
        ENDM               ; конец макроопределения
```

Здесь имя SHF присваивается группе команд, осуществляющей сдвиг содержимого накопителя вправо на N позиций, при этом в качестве счетчика циклов используется регистр с именем REG. Заголовок макроопределения содержит имя SHF, псевдокоманду MACRO и набор параметров (в данном случае REG и N), которые в общем случае не являются обязательными. Последней строчкой макроопределения является псевдокоманда

да ENDM, указывающая на его конец. Если теперь в программе встретится макровывоз SHF (текст слева), то вместо него будут подставлены соответствующие команды (текст справа):

:	:
LDA T	LDA T
SHF B,5	MVI B,5
STA T	M1: RRC
:	ANI 7FH
	DCR B
	JNZ M1
	STA T

При нескольких одинаковых макровывозах внутри одной программы закодированная именем группа команд будет повторяться, однако метки этих команд (в нашем случае M1) будут считаться различными.

При программировании на ассемблере используются также псевдокоманды условного ассемблирования IF и ENDIF. Так, приведенное ниже макроопределение SHF1 осуществляет сдвиг содержимого накопителя влево или вправо:

SHF1	MACRO REG,N,DIR	; заголовок макроопределения
	MVI REG,N	; загрузка счетчика циклов
	IF DIR—'L'	; если направление сдвига не
		; влево
M1:	RRC	; циклический сдвиг вправо
	ANI 7FH	; очистить старший бит
	ENDIF	; конец участка условного ас-
		; семблирования
	IF DIR—'R'	; если направление сдвига не
		; вправо
M1:	RLC	; циклический сдвиг влево
	ANI 0FEH	; очистка младшего бита
	ENDIF	; конец участка условного ас-
		; семблирования
	DCR REG	; уменьшение счетчика циклов
	JNZ M1	
	ENDM	; конец макроопределения

Ассемблер вычисляет выражение, стоящее в поле операнда псевдокоманды IF. Если результат равен нулю, то команды между IF и ENDIF в результирующую программу не включаются. Если же результат вычисления выражения отличен от нуля, то команды между IF и ENDIF ассемблируются обычным образом. Так, если в макровывозе направление сдвига задано параметром DIR как 'L', т. е. влево (текст слева), то выражение в первой псевдокоманде IF равно нулю и команды сдвига

вправо и очистка старшего бита не ассемблируются. Выражение же во второй псевдокоманде IF отлично от нуля, и команды сдвига влево и очистки младшего бита будут включены в результирующую программу (текст справа):

:	:
:	:
:	LDA T
LDA T	MVI B,5
SHF1 B,5,'L'	M1: RLC
STA T	ANI 0F0FH
:	DCR B
:	JNZ M1
:	STA T
:	:
:	:

Условное ассемблирование может использоваться и вне макроопределений.

В качестве еще одного примера рассмотрим макроопределение для сложения двух целых положительных чисел без знака. Результат и слагаемые размещаются в ЗУ по адресам ADR1, ADR2, ADR3 соответственно. Каждое из этих чисел независимо друг от друга может занимать один или два байта. В первом случае параметры макроопределения BA1, BA2 и BA3 имеют значения 1, во втором — 2. Байты первого операнда перед сложением помещаются в регистр B, байты второго — в накопитель. Если параметр BA1=1 (результат занимает один байт), то складываются только первые (младшие) байты операндов. В противном случае вычисляется также второй (старший) байт результата. Если при этом какой-либо из операндов занимает только один байт, то при сложении его старший байт заполняется нулями. Текст макроопределения имеет вид:

PLUS	MACRO ADR1, BA1, ADR2, BA2, ADR3, BA3	
	LXI H, ADR2	; занесение младшего бай-
	MOV B, M	; та первого операнда
		; в регистр B
	LDA ADR3	; занесение младшего бай-
		; та второго операнда
	ADD B	; в накопитель
	ADD B	; сложение младших бай-
		; тов
	STA ADR1	; занесение младшего бай-
		; та результата
	IF BA1-1	; если результат занимает
		; два байта

IF BA2-1	, если первый операнд за- нимает два байта
INX H	
MOV B,M	; занесение старшего бай-
ENDIF	; та первого операнда
	; в регистр B
IF BA2-2	; если первый операнд за-
	; нимает один байт
MVI B,Ø	; занесение нуля в ре-
	; гистр B
ENDIF	
IF BA3-1	; если второй операнд за-
	; нимает два байта
LDA ADR3+1	; занесение старшего бай-
	; та второго операнда
	; в накопитель
ENDIF	
IF BA3-2	; если второй операнд за-
	; нимает один байт
MVI A,Ø	; занесение нуля в нако-
	; питель
ENDIF	
ADC B	; сложение старших бай-
	; тов с учетом переноса
STA ADR1+1	; занесение старшего бай-
	; та результата
ENDIF	
ENDM	

Псевдокоманды перемещаемого ассемблера. Ассемблер, используемый в развитом аппаратно-программном отладочном комплексе INTELLEC MDS-800 фирмы INTEL, является перемещаемым. Такой ассемблер позволяет отлаживать разрабатываемые программы в одной области памяти, например ОЗУ, а затем перемещать их полностью или частично в другую область памяти, например ПЗУ. Кроме того, написание и отладка программного обеспечения может осуществляться по частям одновременно несколькими программистами.

Перемещаемые программные модули используют три счетчика адресов, управляемые псевдокомандами ассемблера ASEG (Абсолютный сегмент), DSEG (Сегмент данных) и CSEG (Сегмент команд). Перед началом ассемблирования содержимое всех трех счетчиков равно нулю. Появление одной из указанных псевдокоманд в программе включает один из счетчиков и в дальнейшем приводит к размещению следующих далее команд и данных в соответствующем сегменте. При появлении в программе псевдокоманды другого сегмента

содержимое счетчика предыдущего сегмента запоминается, а в работу включается счетчик нового сегмента. С помощью псевдокоманды `ORG` активному в момент ее появления счетчику можно присвоить любое значение. Сегменты команд `CSEG` и данных `DSEG` являются перемещаемыми. Их абсолютные адреса определяются программой-загрузчиком. При этом сегмент команд размещается в ПЗУ, а сегмент данных — в ОЗУ. Содержимое абсолютного сегмента `ASEG` ассемблируется в абсолютном формате и впоследствии не перемещается. Если в начале программы отсутствуют псевдокоманды `ASEG`, `CSEG` и `DSEG`, то предполагается, что программа размещается в абсолютном сегменте.

При разбиении программного обеспечения на отдельные модули в последних могут применяться метки и переменные, определяемые в других модулях. Все метки и идентификаторы переменных, к которым осуществляется обращение в данном модуле, но определяемые вне его, должны быть указаны через запятую в псевдокоманде `EXTRN` (Внешние). Все метки и идентификаторы переменных, определяемые в данном модуле и используемые также вне этого модуля, должны быть указаны через запятую в псевдокоманде `PUBLIC` (Общие). Таким образом, идентификаторы, применяемые для связи между модулями, могут быть указаны в псевдокоманде `PUBLIC` только в одном модуле. В остальных модулях они указываются в псевдокомандах `EXTRN`.

Псевдокоманда `NAME` (Имя) присваивает имя программному модулю для его идентификации при связи с другими модулями. Именем модуля может быть любой идентификатор. Псевдокоманда `NAME` должна предшествовать появлению первой команды и первого байта данных в модуле. Если в программе псевдокоманда `NAME` опущена, то модулю присваивается имя `MODULE` (Модуль).

2.3. Программирование на языке ПЛ/М-80

Язык ПЛ/М-80 является языком высокого уровня, предназначенным для написания как системных, так и прикладных программ. Этот язык был разработан фирмой `INTEL` (США) для МП 8080 и 8085 [28]. Он применяется также для разработки программного обеспече-

ния микропроцессорных систем на основе серии КР580. Ниже приводится описание основных возможностей языка ПЛ/М-80.

Типы данных. Все переменные в этом языке являются целыми без знака и могут занимать один [переменные типа BYTE (Байт)] или два [переменные типа ADDRESS (Адрес)] байта. В первом случае переменная может принимать значения $\emptyset$ —255, во втором — $\emptyset$ —65 535. Идентификатором переменной может служить любая последовательность латинских букв и цифр, начинающаяся с буквы и содержащая не более 31 символа. В языке используются также одномерные массивы, причем наименьшим значением индекса является нуль. Так, если в программе предусмотрен массив F, состоящий из 100 элементов, то он включает элементы от F ($\emptyset$) до F (99). Перед использованием простых переменных и массивов в программе они должны быть введены в нее с помощью специального оператора DECLARE (Описание). Так, операторы

```
DECLARE F(1 $\emptyset\emptyset$ ) BYTE;  
DECLARE (W1,L1,H1,V1), ADDRESS;
```

описывают 100-элементный массив F, каждый элемент которого занимает один байт, и переменные W1, L1, H1, V1, занимающие в памяти по два байта.

В языке ПЛ/М с помощью одного идентификатора возможна ссылка к группе переменных различных типов, называемой структурой. Так, оператор

```
DECLARE PLANE STRUCTURE (SPEED BYTE, ALT  
ADDRESS);
```

описывает структуру с идентификатором PLANE (Самолет). Структура содержит однобайтовую переменную SPEED (Скорость) и двухбайтовую переменную ALT (Высота). Ссылкой к первой переменной является PLANE. SPEED, а ко второй — PLANE. ALT.

Более сложным примером может служить описание
DECLARE STUDENT (15 $\emptyset$) STRUCTURE (
NAME (15) BYTE,
SNAME (15) BYTE,
FNAME (15) BYTE,
AVR BYTE,
RUBL BYTE);

В данном случае описывается массив из 150 структур (с $\emptyset$ -й по 149-ю), каждая из которых соответствует

одному студенту. В структуру входят массивы NAME, SNAME, FNAME по 15 однобайтовых элементов каждый, в которые заносятся фамилия, имя и отчество студента, переменная AVR для хранения среднего балла студента за последнюю сессию и переменная RUBL, указывающая размер стипендии в рублях.

Для языка ПЛ/М характерно использование базированных переменных, т. е. переменных, адресом (базой) которых является значение другой переменной. Базированная переменная задается сначала описанием базы, обязательно имеющей тип ADDRESS (в данном случае POINTER), а затем описанием самой базированной переменной (C) с указанием ее базы (POINTER):

```
DECLARE POINTER ADDRESS;  
DECLARE C BASED POINTER BYTE;  
После этих описаний операторы
```

```
POINTER=88H,  
C=20H,
```

заносят шестнадцатеричное число 20 по адресу 88H.

При трансляции для базированной переменной память не выделяется, так как при выполнении различных частей программы она может занимать различные адреса памяти.

Следующие два оператора:

```
DECLARE P ADDRESS;  
DECLARE A BASED P (100) ADDRESS;
```

описывают переменную P и 100-элементный массив A типа ADDRESS, нулевой элемент которого располагается по адресу P.

Операторы присваивания. Как в любом языке программирования, одним из основных операторов ПЛ/М является оператор присваивания. Например,

$$V1 = W1 * L1 * H1;$$

Справа от знака равенства может стоять любое выражение, состоящее из описанных ранее простых переменных, элементов массивов и структур, числовых и символьных констант, круглых скобок и знаков действий: + (плюс), - (минус), / (разделить), * (умножить), MOD (остаток от деления), логических операторов NOT, AND, OR, XOR и операторов отношения < (меньше), > (больше), <= (меньше или равно), >= (больше или

равно), $\langle \rangle$ (не равно), $=$ (равно). При этом запись числовых и символьных констант аналогична принятой при программировании на ассемблере (см. § 2.2). Если значение числовой константы меньше 256, а символьная константа включает один символ, то они имеют тип BYTE. Числовые константы от 256 до 65 535 и символьные константы, состоящие из двух символов, имеют тип ADDRESS.

Результат операций «унарный минус» и NOT имеют тот же тип, что и операнд. При выполнении сложения, вычитания и логических операторов результат операции имеет тип BYTE, если оба операнда имеют тип BYTE, и тип ADDRESS в противном случае. Если один из операндов имеет тип ADDRESS, а другой тип BYTE, то перед выполнением операции операнд типа BYTE дополняется до 16 разрядов нулями в старших разрядах. Результат операций умножения, деления и получения остатка (MOD) всегда имеет тип ADDRESS. Если результат умножения превышает 65 535, то он не определен. Не определены также результаты деления и получения остатка, если деление производится на нуль. При выполнении операций отношения производится сравнение двух операндов. При истинном отношении результат равен OFFH , в противном случае — OFFH . В любом случае результат имеет тип BYTE.

Результат вычисления выражения присваивается переменной, стоящей слева от знака равенства. Если эта переменная имеет тип ADDRESS, а результат вычисления выражения — тип BYTE, то последний дополняется нулями в старших разрядах. Если переменная имеет тип BYTE, а результат вычислений — тип ADDRESS, то старший байт результата отбрасывается.

При вычислении выражений действия производятся в следующем порядке: вычисление выражений в скобках; унарный минус; умножение, деление, получение остатка; сложение и вычитание; операторы отношения; оператор NOT; оператор AND; операторы OR, XOR.

Пусть в программе описаны переменные K и Y типа BYTE, переменные X, VAL и массив HT (10) типа ADDRESS, причем $K = \text{OFFH}$, а $\text{HT}(9) = 13$. Тогда после выполнения операторов присваивания

```
X = (K <= 88H) * 100H OR 11B;  
VAL = HT(9) MOD 5;  
Y = K - OFF001H;
```

переменная X примет значение OFFH , переменная VAL — значение 3, а переменная Y — значение FH .

Часто удобно присваивать одно и то же значение нескольким переменным одновременно. В языке ПЛ/М это осуществляется заданием слева от знака равенства списка переменных, разделенных запятыми. С помощью одного оператора можно, например, присвоить значение выражения $X * \text{VAL}/3$ переменным K1, F, G:

```
K1, F, G, = X*VAL/3;
```

В качестве элементов выражений могут использоваться также обращения к ячейкам памяти. Так, если после описаний

```
DECLARE P ADDRESS;  
DECLARE A BASED P (100) ADDRESS;
```

записать оператор

```
K = .A(100);
```

то переменной K будет присвоено значение адреса десятого элемента массива A, т. е. $P+100$. Результат выполнения этого оператора всегда имеет тип ADDRESS.

Обращение к ячейке памяти может использовать частично определенное обращение к переменной. Например, если задан массив структур STUDENT (150) (с. 116), то оператор

```
N1=STUDENT. (100). FNAME -2;
```

присваивает двухбайтовой переменной N1 значение, на 2 меньшее, чем адрес нулевого элемента массива FNAME в десятой структуре STUDENT. Оператор $K1=STUDENT$ присваивает переменной K1 начальный адрес размещения нулевой структуры STUDENT.

Другой формой обращения к ячейке памяти является обращение к списку констант, в том числе символьных строк, разделенных запятыми. Так, оператор

```
PNT = ('TEXT1', 'TEXT2');
```

присваивает переменной PNT адрес ячейки памяти, начиная с которой в коде КОИ-7 располагаются символы TEXT1, TEXT2. При этом компилятор с языка ПЛ/М выделяет 100 ячеек памяти, в которые и заносит указанные символы.

Обращения к ячейкам памяти весьма удобны при присвоении значений базам переменных. Пусть в раз-

личных местах программы используются массивы NAME (15), CITY (15) и RANG (15), содержащие различные текстовые переменные. В процессе их обработки в различных местах программы нам хотелось бы обращаться к ним с помощью одного и того же имени TEXT. Это обеспечивается следующим образом:

```
DECLARE (NAME(15), CITY(15), RANG(15)) BYTE;  
DECLARE PNT ADDRESS;  
DECLARE TEXT BASED PNT(15) BYTE;  
:  
PNT = .NAME;
```

После этого базе PNT массива TEXT присваивается начальный адрес массива NAME и обращение к элементу массива TEXT [например, TEXT (3)] фактически будет обращением к соответствующему элементу массива NAME [в данном случае NAME (3)]. Если далее в программе встретится оператор PNT=.CITY; то после него обращение к массиву TEXT фактически является обращением к массиву CITY и т. д.

Внутри операторов ПЛ/М возможно применение так называемого вложенного присваивания. Например, значение выражения $X * (VAL := W1 * L1 * H1) / 3$ равно значению выражения $X * W1 * L1 * H1 / 3$. Однако в первом случае кроме вычисления значения выражения значение произведения $W1 * L1 * H1$ запоминается по адресу переменной VAL. Этот промежуточный результат может быть использован в программе позднее без нового его вычисления.

В языке ПЛ/М имеется предварительно объявленный массив OUTPUT, состоящий из 256 элементов типа BYTE. Каждый элемент этого массива соответствует одному из 256 выходных каналов. Идентификатор OUTPUT может появляться только в операторе присваивания слева от знака равенства, причем индексом может быть только числовая константа. Оператор

$OUTPUT(102) = K/2 - 10;$

обеспечивает выдачу на 102-й выходной канал значения 20 при $K=60$.

Исполнительные операторы языка ПЛ/М, в том числе операторы присваивания, могут быть помечены. Меткой может являться любой идентификатор. Метка помещается перед оператором и отделяется от него двоеточием

точием. Например,

```
LOOP: K=K+1;  
F1: L1=B1 AND 10111001B;
```

Любой исполнительный оператор может иметь более одной метки. Например,

```
M1: COUNT: BEG: NEXT = X(1);
```

Внутри одного программного блока не должно быть операторов, помеченных одинаковыми метками.

Операторы управления ходом выполнения программы. Основными операторами языка ПЛ/М, осуществляющими управление последовательностью выполнения программы, являются операторы DO, IF и GO TO.

Операторы DO (ВЫПОЛНИТЬ) всех типов являются началом блока DO, который всегда заканчивается оператором END (КОНЕЦ). За оператором END может следовать метка (без двосточия), которой помечен оператор DO. В этом случае повышается наглядность текста программы. Таким образом, операторы DO и END всегда употребляются совместно и действуют как операторные скобки, выделяющие некоторый программный блок. Оператор DO имеет следующие разновидности: простой оператор DO, оператор DO WHILE (ВЫПОЛНЯТЬ ПОКА), итеративный оператор DO, оператор DO CASE (ВЫПОЛНИТЬ ВАРИАНТ).

Пример простого оператора DO приведен ниже:

```
DO;  
    NV=OV+TEMP;  
    N=N+1;  
END;
```

Блок простого оператора DO можно рассматривать как один оператор и использовать его в любом месте программы, где возможно применение простого оператора, например в блоках DO CASE.

Блок оператора DO WHILE, например,

```
DO WHILE (I+3)*2<K;  
    X(I)=Y(I) MOD 16;  
    I=I+1;  
END;
```

выполняется следующим образом.

Если значение выражения оператора DO WHILE истинно (фактически только младший бит результата проверяется на единицу), то выполняются все операторы блока DO WHILE. Если же значение выражения в

операторе DO WHILE ложно (младший бит результата равен нулю), то операторы блока не выполняются, а управление передается оператору, следующему за оператором END.

Если на предыдущем шаге операторы блока выполнялись, то после этого вновь вычисляется значение выражения оператора DO WHILE и процесс повторяется. Выполнение блока прекращается только в том случае, если младший разряд значения выражения равен нулю.

Так же, как и простой блок DO, блок DO WHILE можно рассматривать как один оператор и использовать его как таковой в блоках DO CASE и операторах IF.

Итеративный блок DO

DO I=1 TO N*2+3 BY 2;

S=S+A(I)—A(I)/8*8;

END;

выполняет следующие действия: вычисляется начальное значение индекса, в данном случае индексу цикла I присваивается значение 1. Это делается только один раз до первого прохода блока; вычисляется значение ограничивающего выражения ($N \cdot 2 + 3$), которое сравнивается со значением индекса цикла. Если значение индекса больше значения ограничивающего предложения, то действия, описываемые в следующих пунктах, опускаются, а управление передается оператору, следующему за оператором END; выполняются операторы блока; значение индекса цикла увеличивается на значение выражения, следующего после служебного слова BY. Если служебное слово опущено, то предполагается, что шаг изменения индекса равен единице. Если новое значение индекса меньше старого (это возможно, так как все действия выполняются по модулю 256 или 65 536), то управление передается оператору, следующему за оператором END. В противном случае осуществляется переход ко второму пункту, описанному выше.

Блок DO CASE начинается оператором DO CASE и выполняет только один из операторов блока:

DO CASE <выражение>;

оператор 1;

оператор 2;

.....

оператор n;

END;

При этом сначала вычисляется значение выражения, которое должно быть от 0 до $n - 1$. Значение 0

соответствует выполнению первого, а значение $n-1$ — выполнению n -го оператора блока. После выполнения выбранного оператора блока управление передается оператору, следующему за оператором END. В качестве операторов блока DO CASE могут, в частности, использоваться любые блоки DO. Например,

```
DO CASE N=3;
    K=K+1;           /*N = 3*/
    DO;              /*Начало при N=4*/
        X=X+5;
        Y=Y+2;
    END;             /*Конец при N=4*/
    ;               /*Пустой оператор при
                    N=5*/
    DO I=1 TO G+1;   /*Начало при N = 6*/
        B(I)=Ø;
    END;             /*Конец при N = 6*/
END;                /*Конец блока DO CASE*/
```

Перед входом в этот блок переменная N должна иметь значение от 3 до 6. При $N=4$ и $N=6$ выполняется простой или итеративный блок DO соответственно. При $N=5$ выполняется пустой оператор, т. е. никаких действий не производится.

Как видно из приведенных выше примеров, после каждого оператора ставится точка с запятой. Оператор может начинаться в любом месте строки, что позволяет записывать текст программы весьма наглядно. Язык предусматривает использование комментариев в тексте программы. Комментарии слева ограничиваются символами /*, а справа — */. Комментарии могут располагаться в любом месте программы.

Оператор IF (ЕСЛИ) имеет форму
 IF <выражение> THEN <оператор 1>;
 ELSE <оператор 2>;

Оператор вычисляет значение выражения. Если значение выражения равно ØFFH (фактически проверяется только младший бит), то выполняется оператор 1, если значение выражения равно ØØH (фактически, если значение младшего бита равно нулю), то выполняется оператор 2. Часть оператора ELSE (Иначе) может быть опущена. В качестве операторов 1 и 2 могут использоваться блоки DO. Например,

```
IF K>N      THEN  S=S+G(I);
ELSE        DO;
```



```

L1=L1+1;
P=P+G(I);

```

```

END;

```

Оператор IF вместе с частью ELSE рассматривается как отдельный оператор языка ПЛ/М. Он может использоваться в блоках DO CASE и в других операторах IF. Таким образом, операторы IF могут быть вложены друг в друга. Однако если оператор IF вложен в часть THEN внешнего оператора IF, то этот внешний оператор не может содержать часть ELSE. Вложение оператора IF в другой оператор IF, содержащий часть ELSE, осуществляется блоком DO:

```

IF (G+Q)/2 < > K11 THEN
DO;
    IF K < 3 THEN S=S+2;
    ELSE S=S-1;
END;
ELSE S=S+1;

```

Оператор GO TO (ПЕРЕЙТИ НА) вызывает передачу управления на оператор с указанной меткой. Например,

```

GO TO M1;

```

передает управление оператору с меткой M1.

Оператор HALT (ОСТАНОВ) соответствует команде HLT ассемблера, т. е. вызывает останов выполнения программы. Из состояния останова процессор может быть выведен либо сигналом сброса, либо поступлением запроса на прерывание. При выполнении этого оператора триггер разрешения прерывания устанавливается в единицу.

Процедуры. Аналогом подпрограммы на языке ассемблера в языке ПЛ/М является процедура. Все современные технологии программирования предполагают разбиение программы на несложные, связанные друг с другом подпрограммы или процедуры. Описание процедуры состоит из оператора PROCEDURE (ПРОЦЕДУРА), последовательности операторов, составляющих тело процедуры, и оператора END. Например,

```

SUM: PROCEDURE (N, VAL);
    DECLARE (I,N) BYTE, VAL ADDRESS;
    S=0;
    DO I=1 TO N;
        IF G(I) <= VAL THEN S=S+G(I);
    END;
END SUM,

```

Здесь идентификатор SUM является именем процедуры, а переменные N, VAL — формальными параметрами.

рами. В последнем операторе (здесь END SUM) после оператора END может указываться имя процедуры. Это увеличивает наглядность текста программы и облегчает работу с ней. Однако это имя может и опускаться. Формальными параметрами процедуры могут являться только скалярные переменные, причём они должны быть описаны в теле процедуры до появления первого выполняемого оператора. В частном случае процедура может не иметь ни одного параметра. Тогда в операторе PROCEDURE список формальных параметров, включая скобки, опускается.

Обращение к процедуре из основной программы или процедуры более высокого уровня осуществляется оператором CALL (ВЫЗВАТЬ). Например, в нашем случае этот оператор будет иметь вид

```
CALL SUM(1Ø,F/2+2);
```

где после имени процедуры в скобках указываются фактические параметры, которые могут быть любыми выражениями. При выполнении оператора CALL вычисляются значения выражений, являющихся фактическими параметрами. Затем эти значения присваиваются формальным параметрам, объявленным внутри процедуры, и выполняются операторы тела процедуры. После этого управление передается оператору, следующему за оператором CALL. Выход из процедуры осуществляется тогда, когда управление переходит к оператору END, указывающему на конец процедуры, или когда в теле процедуры встречается оператор RETURN (ВОЗВРАТ). В теле процедуры может быть несколько операторов RETURN.

Выше была описана структура процедуры без значения. В языке ПЛ/М предусмотрены также процедуры со значением. Например,

```
AVR: PROCEDURE (N) ADDRESS;
      DECLARE (I,N) BYTE, (MAX, MIN) ADDRESS;
      MIN = G(Ø);
      MAX = G(Ø);
      DO I = 1 TO N;
          IF MIN > G(I) THEN MIN = G(I);
          IF MAX < G(I) THEN MAX = G(I);
      END;
      RETURN (MAX + MIN)/2;
END AVR;
```

Процедура со значением имеет тип либо BYTE, либо ADDRESS. Это означает, что после ее вызова в основ-

ную программу или процедуру более высокого уровня возвращается значение переменной AVR, имеющей в данном случае тип ADDRESS. Процедура со значением должна обязательно содержать оператор RETURN с выражением, значение которого присваивается переменной, являющейся именем процедуры. Обращение к процедуре со значением осуществляется с помощью функциональной ссылки. Так, в основной программе может быть записан оператор

MIDDLE = AVR (1Ø);

При этом при обращении к процедуре будет вычислена середина диапазона 11 элементов (с Ø по 1Ø) массива G(I) и полученное значение присвоено переменной MIDDLE. Функциональная ссылка может являться одним из операндов в выражении.

В языке ПЛ/М предусмотрен ряд встроенных процедур, в частности процедура INPUT (ВВОД) типа BYTE. Обращение к ней имеет форму INPUT (<константа>), где действительный параметр должен быть числовой константой в диапазоне от Ø до 255. Так, оператор

B1 = INPUT (5);

обеспечивает ввод байта данных с 5-го канала ввода в накопитель и передачу этого байта в ячейку B1.

Язык ПЛ/М имеет специальные средства для описания процедур обработки прерываний. Процедура баз значения, не имеющая параметров, объявленная на внешнем уровне программного модуля, может иметь описатель INTERRUPT *n* (Прерывание *n*), где *n* — числовая константа в диапазоне Ø—7 (при наличии в микропроцессорной системе контроллеров прерываний диапазон может быть расширен до Ø—255).

Управление работой процессора передается этой процедуре при возникновении в системе прерывания с соответствующим номером, т. е. выполнении команды RST *n*. Следует отметить, что эта команда обычно формируется устройством, вызывающим прерывание, и подается на шину данных МП. Прием прерываний процессором и активация процедуры обработки прерывания возможны только при единичном состоянии триггера разрешения прерываний. После включения микропроцессорной системы этот триггер установлен в нуль. Для установки его в единицу используется оператор

ENABLE (РАЗРЕШИТЬ), а для сброса в нуль оператор DISABLE (ЗАПРЕТИТЬ).

При передаче управления процедуре с описателем INTERRUPT триггер разрешения прерывания сбрасывается, выполняются операторы процедуры, триггер разрешения прерывания вновь устанавливается в единицу и управление возвращается в прерванную программу.

Область действия объектов. ПЛ/М является языком с блочной структурой. Программным блоком является всякий блок DO и всякое описание процедуры. Программные блоки ограничивают области действия объектов: процедур, переменных и меток.

Областью действия процедуры, переменной или метки является тот программный блок (блок DO или процедура), в котором она объявлена, за исключением тех вложенных в него программных блоков, в которых та же процедура, переменная или метка объявлены вновь. Причем описания переменных и процедур могут располагаться только внутри простого блока DO или внутри описания процедуры и должны предшествовать всем исполнительным операторам. Например,

```
M1: DO;                                /* Внешний блок */
    I=I+2;
    K = K+1;
    DO;                                /* Внутренний блок */
    DECLARE I BYTE;
    MAX = G(K);
    I = K+1;
M1: IFG(I)>MAX THEN MAX=G(I);
    I = I+1;
    IF I<N THEN GOTO M1;
    END;                                /* Конец внутреннего блока */
F = G(I);
END;                                    /* Конец внешнего блока */
```

Здесь предполагается, что переменные I, K, N типа BYTE и MAX, G(100), F типа ADDRESS объявлены в блоке, в который входит приведенный выше фрагмент программы. Переменная I типа BYTE, объявленная во внутреннем блоке DO, и переменная I внешнего блока DO являются разными переменными. Так, если I=6 перед началом выполнения приведенного выше фрагмента программы, то переменной F будет присвоено значение G(8). Переменная K, не описанная во внутреннем блоке DO, является общей для внешнего и внут-

ренного блоков. Метка M1 внешнего оператора DO и метка M1 оператора IF во внутреннем блоке воспринимаются как различные.

Модули. При программировании на ПЛ/М программа разбивается на модули, которые транслируются отдельно, а затем объединяются в программу с помощью редактора связей. Модулем является помеченный простой блок DO, который не вложен ни в какой другой блок. При этом необходимо, чтобы основной программный модуль был единственным. Неосновные модули содержат только описания процедур. Редактор связей позволяет объединять модули, полученные трансляцией с языка ПЛ/М и ассемблера.

При описании переменных внешнего уровня основного модуля используются описатели PUBLIC (Общий) и EXTERNAL (Внешний). Все переменные с описателем PUBLIC доступны из других программных модулей, имеющих декларацию тех же переменных с описателем EXTERNAL.

Например, если один модуль включает оператор

```
DECLARE (I, J) BYTE PUBLIC;
```

а другой

```
DECLARE (I, J) BYTE EXTERNAL;
```

то одни и те же переменные I, J применяются в обоих модулях. При объединении модулей в программу каждая общая переменная может иметь описатель PUBLIC только в одном модуле. Во всех остальных модулях, использующих эту же переменную, последняя должна иметь описатель EXTERNAL.

Если обращение к одной и той же процедуре производится при выполнении различных модулей, то в одном из модулей эта процедура должна иметь описатель PUBLIC. Например,

```
SUM: PROCEDURE (X,Y) ADDRESS PUBLIC;
```

```
    DECLARE (X,Y) ADDRESS;
```

```
    RETURN X+Y;
```

```
END SUM;
```

Во всех остальных модулях, где процедура SUM также используется, она должна иметь описатель EXTERNAL. В этих модулях из описания процедуры исключаются выполняемые ею операторы:

```
SUM PROCEDURE (X,Y) ADDRESS EXTERNAL;
```

```
    DECLARE (X,Y) ADDRESS;
```

```
END SUM;
```

Приведем несколько примеров программ на ПЛ/М-80.

1. Рассмотрим процедуру PLA, моделирующую работу ПЛМ (см. пример 4 § 2.2). В процедуре PLA байт аргумента обозначен как X. Константы D_i , T_i , M_i располагаются в том же порядке, что и в § 2.2, и образуют массив TABLE:

```
PLA:  PROCEDURE (X) BYTE;
      DECALARE (I,X,F) BYTE;
      I=0;
      F=0;
      DO WHILE TABLE (I) <> 0;
        IF X AND TABLE (I) XOR TABLE (I+1)=0
          THEN F=F OR TABLE (I+2);
          I=I+3;
      END;
      RETURN F;
      END PLA;
```

Если предположить, что в основной программе имеются следующие описания:

```
DECLARE (Y,Z) BYTE;
DECLARE PNT ADDRESS;
DECLARE TABLE BASED PNT (28) BYTE;
```

то набор логических функций Z (см. рис. 2.5) от вектора аргументов Y может быть вычислен следующим образом:

```
:
:
Y = .....;
PNT=(0C0H, 0C0H, 98H, 60H, 20H,..., 0F0H, 0F0H,
      05H, 00H);
Z=PLA(Y);
:
:
```

При использовании в основной программе процедуры PLA для изменения порядка двоичных разрядов в слове, эти описания могут быть оставлены без изменения [так как число элементов таблицы FFT (пример 4 § 2.2) меньше, чем таблицы TABL]. Обращение же к подпрограмме производится так:

```
:
:
Y = ...;
PNT=(128, 128, 1,64, 64,2, 32, 32,4, 16, 16,8, ..., 1,1, 128, 0);
Z=PLA(Y);
:
:
:
```

При необходимости вычисления 16 логических функций от 16 аргументов достаточно процедуру PLA и переменные X и F описать как имеющие типа ADDRESS.

2. Обычно при разработке транслятора с какого-либо языка программирования стока исходного текста помещается в некоторый массив BUF объемом 80 символов. Если строка текста содержит ме-

нее 80 символов, то оставшиеся элементы массива BUF заполняются пробелами. Пусть каждая строка начинается оператором языка, содержащим не более 6 символов. В общем случае имя (это могут быть служебное слово, метка или идентификатор) всегда начинается с латинской буквы и не содержит символов, отличных от латинских букв и цифр. Необходимо написать программу, формирующую имя оператора и определяющую его порядковый номер NOP в таблице операторов OPTBL. При задании несуществующего оператора (оператор в таблице OPTBL не найден) NOP = $\emptyset$. Требуемая программа представлена в виде двух модулей:

```

M1: DO;
  DECLARE (ER, NAME(6)) BYTE PUBLIC;
  DECLARE (PTR, BUF (80)) BYTE EXTERNAL;
  DECLARE (IMG, LAST) BYTE;
  FNAME: PROCEDURE /*процедура формирования имени*/
    PUBLIC;
    DECLARE I BYTE;
    ER =  $\emptyset$ ;
    /*выборка отличного от пробела символа*/
    DO WHILE LAST = ' ' AND IMG < > 4;
      CALL NCH;
    END;
    /*заполнение массива NAME пробелами*/
    DO I =  $\emptyset$  TO 5;
      NAME(I) = ' ';
    END;
    /* проверка, буква ли первый символ имени */
    IF IMG < > 1 THEN
      DO;
        ER = 1;
        RETURN;
      END;
    NAME( $\emptyset$ ) = LAST;
    DO I = 1 TO 5;
      CALL NCH;
      IF IMG < > 1, AND IMG < > 2 THEN
        RETURN;
      ELSE NAME(I) = LAST;
    END;
  END FNAME;
NCH: PROCEDURE; /* процедура выборки следующего символа
из массива BUF в LAST*/
  PTR = PTR + 1;
  IF PTR > 79 THEN
    DO;
      IMG = 4;
      RETURN;
    END; ELSE
    DO;
      LAST = BUF (PTR);
      IMG = 3;
      IF LAST > = 'A' AND LAST < = 'Z' THEN
        IMG = 1;
      IF LAST > = ' $\emptyset$ ' AND LAST < = '9' THEN
        IMG = 2;

```

END;

END NCH;
END M1;

Приведенный выше модуль M1 включает процедуры выборки следующего символа из массива BUF (процедура NCH) и формирования имени (процедура FNAME).

Основная программа располагается в модуле PROG:
PROG: DO;

```
/* модуль основной программы */
DECLARE (ER, NAME (6)) BYTE EXTERNAL;
DECLARE (PTR, BUF(80)) BYTE PUBLIC;
DECLARE (OPTBL(120), NOP, I, J) BYTE DATA ('BACK
FORWARDELAY ... EXICUT')
/* с помощью служебного слова DATA присваиваются значения
декларируемым переменным */
/* декларации использования процедур */
FNAME: PROCEDURE EXTERNAL;
      DECLARE I BYTE;
      END FNAME;
LNIN:  PROCEDURE EXTERNAL;
      DECLARE I BYTE;
      END LNIN;

/* присвоение указателю начального значения */
PTR=-1;
/*запись строки исходного текста в массив BUF */
CALL LNIN;
/* формирование имени */
CALL FNAME;
IF ER < > 0 THEN
  DO;
    /* выдача ошибки */
    OUT(3)='E';
    OUT(3)='R';
    OUT(3)='=';
    OUT(3)='1';

    END;
  ELSE
    NOP=0;
/* поиск имени в таблице OPTBL */
DO I=0 TO 119 BY 6;
  DO J=0 TO 5;
    IF NAME(J) < > OPTBL (I+J) GO TO LEB;
  END;
  NOP=I/6+1;
LEB: END;
/* выдача номера оператора */
OUT(3)='N';
OUT(3)='O';
OUT(3)='P';
OUT(3)='=';
OUT(3)=NOP;
END PROG;
```

Кроме того, предполагается наличие третьего модуля, содержащего процедуру LNIN, которая обеспечивает ввод строки исходного

текста в массив BUF. Результатом работы программы является значение NOP, выдаваемое в канал вывода с адресом 3. Если первым отличным от пробела символом в строке исходного текста является не буква, то переменной ER присваивается значение 1, которое выдается в канал вывода с адресом 3. При отсутствии ошибки ER=0.

При выборке символа из массива BUF используется указатель на последний считанный символ PTR. Символ считывается в переменную LAST. При этом переменной IMG присваивается значение 1, если в LAST находится код латинской буквы, 2 — если код цифры, 3 — если код какого-нибудь другого символа, 4 — если весь массив BUP просмотрен, т. е. PTR>79. Формирование имени осуществляется в массиве NAME (6). Если имя содержит менее 6 символов, то оставшиеся элементы массива заполняются пробелами.

2.4. Организация разработки программного обеспечения микропроцессорных систем

Процесс разработки программного обеспечения микропроцессорных систем выполняется в следующей последовательности: формулировка требований; выбор языка программирования; разработка схемы алгоритма; написание текста программы; трансляция программного обеспечения в коды микропроцессорных средств; отладка программного обеспечения.

Первые три этапа выполняются разработчиком вручную, на четвертом и пятом этапах используются системы автоматизированной разработки программного обеспечения.

Системы автоматизированной разработки программного обеспечения могут быть реализованы на большой, мини- или микро-ЭВМ. Системы на больших и мини-ЭВМ, называемые кросс-системами, позволяют проводить отладку программ без микропроцессорных средств и не требуют затрат на приобретение специализированных аппаратно-программных отладочных комплексов. В простейшем случае эти системы включают кросс-ассемблер или кросс-компилятор с языка высокого уровня и эмулятор с встроенными средствами отладки. Подготовка исходного текста программ на ассемблере или языке высокого уровня и его корректировка осуществляются редактором операционной системы большой или мини-ЭВМ. Кросс-ассемблер или кросс-компилятор обеспечивают генерацию объектного кода программ, а эмулятор позволяет провести их отладку. Существенным недостатком кросс-систем является невозможность совместной отладки программного обеспечения и технических средств.

Резидентные системы, как правило, реализуются в виде аппаратно-программных отладочных комплексов, построенных на используемом в проектируемой системе МПК БИС. Возможности отладочных комплексов, определяемые объемом полупроводниковой памяти, наличием внешних ЗУ, набором периферийных устройств и составом системного программного обеспечения, достаточно подробно рассмотрены в гл. 4.

В отличие от универсальных ЭВМ, аппаратно-программные отладочные комплексы являются специализированными. В связи с этим для резидентных систем разработки программного обеспечения начальные затраты несколько выше, чем для кросс-систем. Однако при проектировании ряда систем и систем повышенной сложности использование резидентных средств более экономично. В настоящее

время наблюдается преобладающее развитие резидентных систем разработки программного обеспечения.

Выбор языка программирования. При выборе языка программирования необходимо иметь в виду следующее.

Текст программы на ассемблере обычно содержит в 2—10 раз больше строк, чем текст функционально эквивалентной программы на языке высокого уровня. Производительность труда среднего программиста составляет 5—20 отлаженных строк в день независимо от того, пишется программа на ассемблере или языке высокого уровня. В связи с этим применение языков высокого уровня существенно ускоряет написание и отладку программного обеспечения [15, 19].

Программы, написанные на языках высокого уровня, обычно занимают на 15—200% больше памяти и требуют для выполнения на 15—300% больше времени, чем программы, написанные на ассемблере опытным программистом [13, 24].

С точки зрения уменьшения объема (стоимости) памяти и затрат на разработку программного обеспечения программирования на ассемблере может быть экономически обосновано при объеме выпуска, превышающем несколько сотен экземпляров. В ближайшие годы усложнение задач и снижение стоимости БИС памяти, по-видимому, приведут к преимущественному использованию языков высокого уровня. Следует отметить, что при создании программного обеспечения приблизительно 2/3 средств затрачивается на этапах постановки задачи и разработки схемы алгоритма, которые не зависят от языка программирования.

При выполнении программ наибольшая часть времени обычно тратится на многократное выполнение повторяющихся участков программ. Весьма эффективными являются выявление таких участков и программирование их на языке ассемблера, в то время как основная часть программы разрабатывается на языке высокого уровня.

При выборе метода программирования следует также учитывать стоимость системы разработки программного обеспечения (см. § 4.1).

Разработка схемы алгоритма. Схема алгоритма программного обеспечения обычно составляется в три этапа. На первом этапе разрабатывается укрупненная схема алгоритма, указывающая последо-



Рис. 2.7. Укрупненная схема алгоритма

зательность функций, которые должны выполняться программным обеспечением (рис. 2.7). Укрупненная схема определяет основные блоки программного обеспечения и составляется на этапе системного проектирования микропроцессорной системы.

На втором этапе блоки детализируются и формируется подробная схема алгоритма (рис. 2.8), в которой указываются конкретные

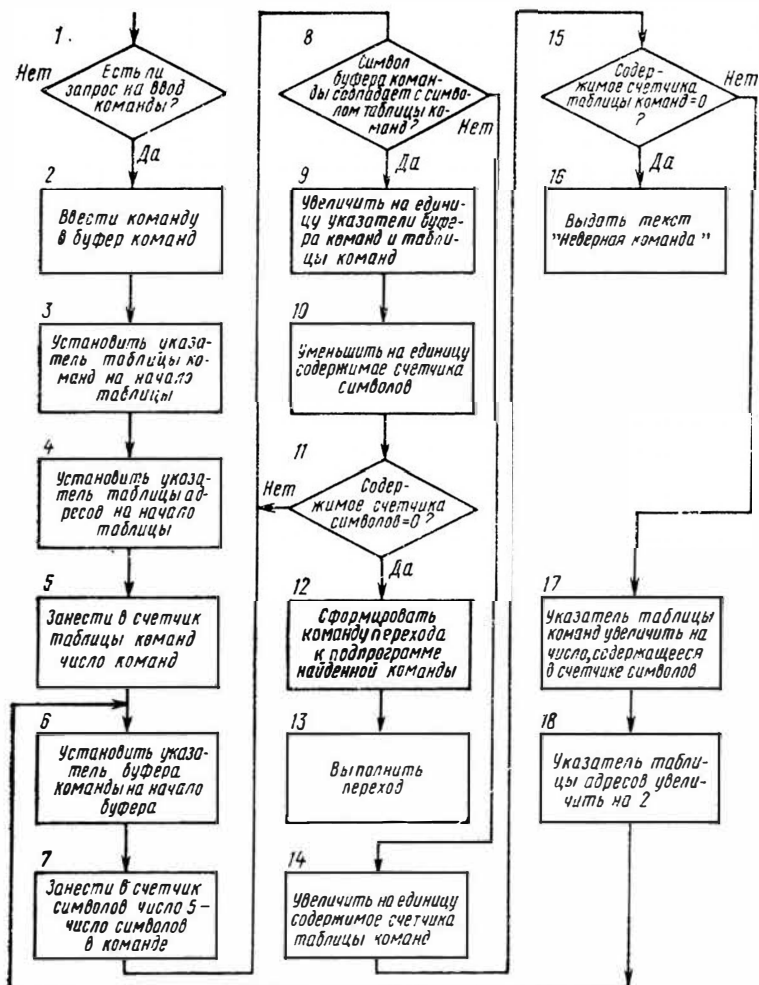


Рис. 2.8. Подробная схема алгоритма для блока 2 укрупненной схемы

шаги, необходимые для достижения желаемого результата. При этом выбирается база данных. Пусть все возможные команды, занесенные в таблицу команд, состоят из пяти символов и вводятся в пятибайтовый буфер команд. Имеется также таблица адресов подпрограмм, обеспечивающих выполнение введенных команд (рис. 2.9). Для работы с буфером команды и таблицы команд вводятся однокбайтовые счетчики и двухбайтовые указатели. Работа с таблицей адресов осу-

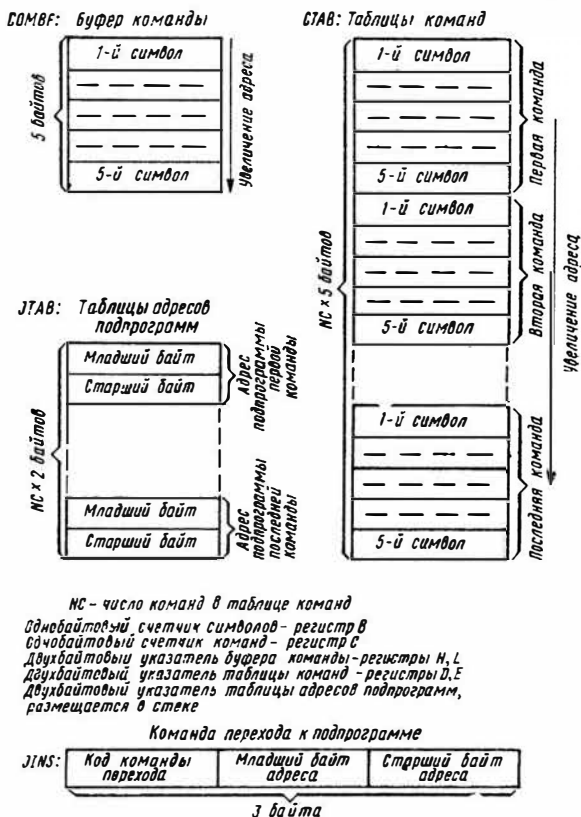


Рис. 2.9. База данных для ввода и идентификации команды

существляется также с помощью двухбайтового указателя. На этом этапе схема алгоритма должна быть как можно более детальной, но еще не привязанной к системе команд МП. Принято считать, что она подробнее укрупненной схемы приблизительно в 20 раз [30].

На третьем этапе происходит дальнейшая детализация схемы алгоритма, в которую вносятся специфика системы команд, режимов адресации и архитектуры микропроцессорных средств (рис. 2.10). Производится размещение элементов выбранной базы данных в ЗУ и регистрах МП и присвоение имен. Так, при использовании микро-

процессора КР580ИК80А роль счетчика символов может выполнять регистр В, а счетчика команд — регистр С. Указатель буфера команд удобно разместить в паре регистров Н—L, указатель таблицы команд — в паре регистров D—E, а указатель таблицы адресов запоминать в стеке. Начальному адресу буфера команд присвоим

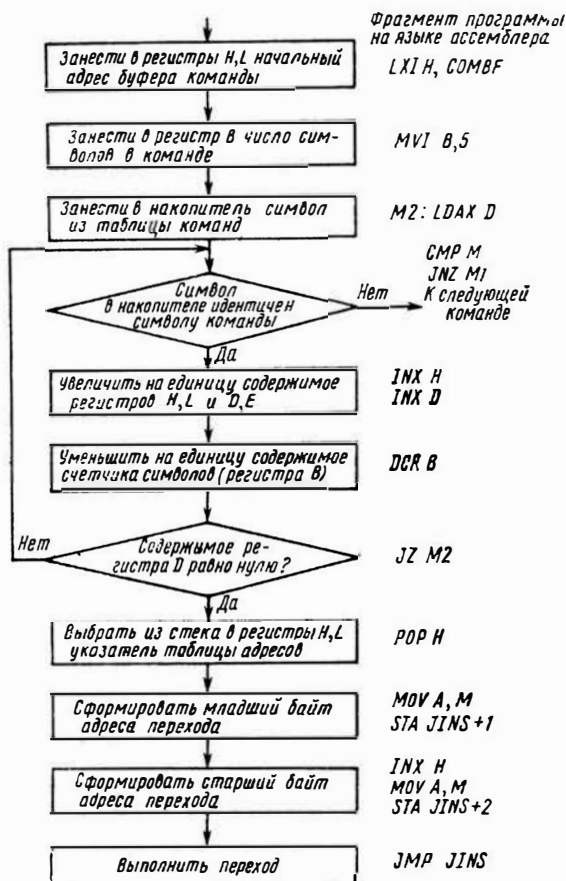


Рис. 2.10. Схема алгоритма, учитывающая особенности МП КР580ИК80А, являющаяся детализацией блоков 6—13 рис. 2.8

имя COMBF, начальному адресу таблицы команд — имя СТАВ, начальному адресу таблицы адресов — JТАВ. Число команд в таблице обозначим именем NC. Для осуществления перехода к подпрограмме выполнения команды введем команду безусловного перехода, размещающую по адресу JINS. Схема алгоритма этого уровня подробнее

схемы предыдущего уровня не более чем в 3 раза; она является исходной для написания программы. Каждый блок этой схемы может быть выполнен не более чем тремя командами.

Структура программы и распределение памяти. При написании текста программы для удобства ее отладки и последующего сопровождения рекомендуется придерживаться определенной последовательности. Заголовком программы должен являться текст комментариев, включающий название программы, дату разработки, номер версии, назначение программы и краткие сведения о требованиях к памяти и организации ввода — вывода. Далее размещаются псевдокоманды определения символических констант и имен, каналов ввода — вывода. В ряде микропроцессорных систем

при распечатке протоколов работы в начале каждой страницы выводится наименование программы или организации-разработчика. Выводимая текстовая информация представляет собой последовательность символических констант. Вслед за псевдокомандами определения символических констант располагаются псевдокоманды резервирования элементов памяти для переменных. Здесь же обычно размещается стек. Следующими по порядку в тексте программы располагаются таблицы переходов, определяющие обращение к блокам программного обеспечения и подпрограммам ввода — вывода. Далее в текст заносят основную программу, которая может содержать только присвоение начальных значений переменным и последовательность вызова подпрограмм. Подпрограммы размещаются вслед за основной программой в порядке обращения к ним. Сначала располагаются подпрограммы, вызываемые основной программой, затем те, обращение к которым осуществляется из уже размещенных подпрограмм. В заключение приводятся таблицы данных. При написании текста программы рекомендуется широко использовать комментарии.

При разработке программного обеспечения важно внимательно провести распределение памяти (рис. 2.11). Процессорно-зависимая область включает небольшое число ячеек для обслуживания прерываний и функций ввода — вывода. В МП КР5801К80А, например, первые 64 байта обычно зарезервированы для обработки прерываний. Резервируемая область адресов памяти предназначена для размещения таблиц переходов, стеков, символьных буферов и параметров подпрограмм. Область временного хранения служит для кратковременного запоминания промежуточных результатов. Все переменные, размещаемые в этих областях, должны быть указаны на диаграмме распределения памяти. Если структура резервируемой области достаточно сложна, то ее обслуживание должно выполняться отдельной программой, которая предохраняет различные сегменты информации от наложения.

В области резидентных программ размещаются постоянно хранящиеся в ЗУ программы: мониторы, загрузчики. В специализирован-

Процессорно-зависимая область

Резервируемая область

Область временного хранения

Область загружаемых программ

Область резидентных программ

03У
03У
03У
03У
Не используется
ПЗУ

Рис. 2.11. Диаграмма распределения памяти

ной микропроцессорной системе в этой области могут размещаться все программы. В большинстве случаев эта область помещается в ПЗУ или ППЗУ. Если в связи с ограниченным объемом полупроводниковой памяти все программное обеспечение одновременно там же не помещается, то отдельные программы вводятся с внешнего носителя в область загружаемых программ.

2.5. КСИ-1 — кросс-система разработки программного обеспечения на мини-ЭВМ СМ-4

Кросс-система КСИ-1 на мини-ЭВМ СМ-4 предназначена для разработки программного обеспечения микропроцессорных систем на базе комплекта БИС серии КР580. В состав КСИ-1 входят: транслятор с языка ассемблера в коды МП КР580ИК80А; имитатор системы команд МП КР580ИК80А; отладочный монитор.

Кросс-система КСИ-1 функционирует в рамках ДОС мини-ЭВМ СМ-4 (ДОС СМ) и фактически является ее подсистемой. Для ввода текстов программ на языке ассемблера или в системе команд МП и их записи в виде файлов в дисковое ЗУ, редактирования текстов программ, вывода программ на печать или перфоленту, ведения библиотек отлаживаемых программ используются стандартные средства ДОС СМ. Все программы КСИ-1 написаны на Фортране.

Ассемблер системы КСИ-1 является абсолютным и не позволяет применять псевдокоманды MACRO и условное ассемблирование. В остальном он соответствует ассемблеру, описанному в § 2.2. Транслятор с языка ассемблера занимает 20К слов.

Имитатор представляет собой набор подпрограмм, соответствующих группам команд МП. Эти подпрограммы объединяются программой-дешифратором команд. Интерпретатор начинает свою работу при задании пользователем директивы Е (Выполнить) или S (Шаг) (табл. 2.3). Аргументом директивы Е является адрес запуска программы. В ходе выполнения команд анализируются заданные режимы трассировки и выполняются нужные действия, например контроль остановов, запоминание состояния программы, вывод содержимого регистров и ячеек памяти. При написании имитатора использовалась система перекрытий ДОС СМ. Подпрограммы имитации команд разбиты на 20 файлов, размещаемых на магнитном диске. Резидентная часть имитатора занимает 15К слов ОЗУ, каждое перекрытие — до 5К слов. Общий объем имитатора — около 115К слов.

Отладочный монитор обеспечивает взаимодействие пользователя с кросс-системой интерпретацией директив, вводимых с клавиатуры дисплея. Все директивы монитора делятся на управляющие и отладочные. Управляющих директив две: R (Чтение ассемблированного модуля в буфер) и O (Переход в отладочный режим). По директиве R объектная программа, полученная транслятором с языка ассемблера или введенная в библиотеку программ в двоичном виде, переписывается в ОЗУ СМ-4. Директива O осуществляет передачу управления интерпретатору отладочных директив.

Использование КСИ-1 рассмотрим на примере программы быстрого умножения двух 8-разрядных положительных чисел (рис. 2.12) [29]. Множимое размещается в регистре D, а множитель — в регистре E. Умножение представляется как четыре операции умножения 4-разрядных чисел, выполняемых табличным методом, с последующим сложением. При этом в регистре D формируется номер строки

Отладочные директивы КСИ-1

Т а б л и ц а 2.3

Вид директивы	Описание
E <адрес>	Запуск программы с заданного адреса. Если адрес не указан, то с адреса в РС
S	Выполнение одной команды
C	Запоминание состояния программы
V	Восстановление состояния программы
D <адрес 1>, <адрес 2>	Вывод содержимого ЗУ, начиная с адреса 1 до адреса 2
D <адрес>:<байт>	Запись байта в ЗУ по заданному адресу
L	Вывод содержимого регистров
L <регистр>:<байт>	Запись байта в указанный регистр
B <адрес 1>, ..., <адрес n>	Установка точек контрольного останова
F <адрес 1>, <адрес 2>: <константа>	Поиск одно-, двух- или трехбайтовой константы
O <адрес 1>, ..., <адрес n>	Задание адресов переменных, выводимых при трассировке
T1	Задание 1-го режима трассировки: печати меток
T2	Задание 2-го режима трассировки: печати кодов команд передачи управления
T3<адрес 1>, ..., <адрес n>	Задание 3-го режима трассировки: печать содержимого регистров и ячеек ЗУ после выполнения команд с указанными адресами
TØ	Отмена режимов трассировки
W	Отображение выходной информации
P <адрес 1>; <адрес 2>	Защита ячеек ЗУ, начиная от адреса 1 до адреса 2 на запись
Z <адрес 1>; <адрес 2>	Отмена защиты на запись
A <PR:A	Ввод отлаживаемой программы с перфоленты
A <LP:	Вывод программы на печать

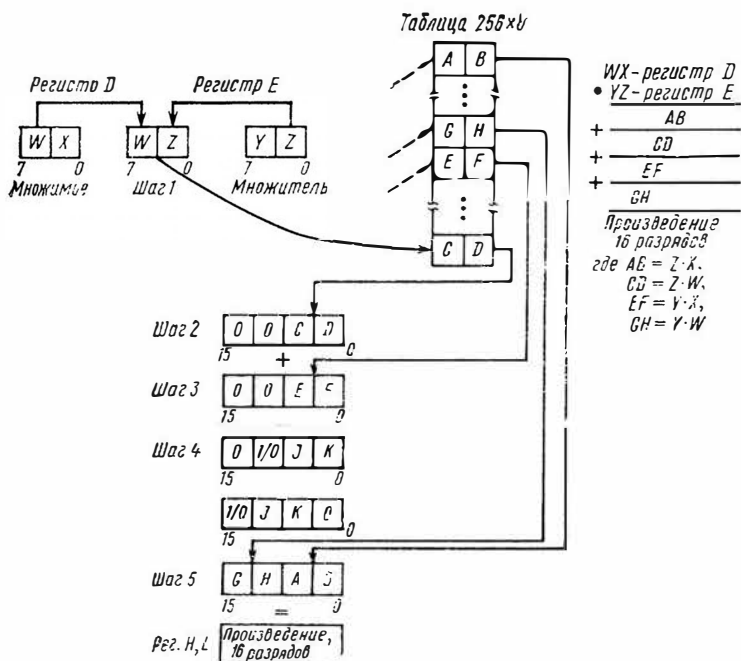


Рис. 2.12. Алгоритм умножения двух 8-разрядных положительных чисел

таблицы TBL, содержащей произведение $Y \cdot X$; регистра L — номер строки, содержащей произведение $W \cdot Z$; в регистре B — номер строки, содержащей произведение $W \cdot Y$; регистре E — номер строки, содержащей произведение $Z \cdot X$. Обращение к таблице TBL для поиска частных произведений осуществляется занесением в регистр H старшего байта адреса таблицы TBL (адрес метки TBL задается равным $4E00H$), а в регистр L — номера строки, содержащей нужное произведение. После четырех обращений к таблице (с помощью команды $MOV r, M$, где r — нужный регистр) в регистр C заносится частное произведение CD, в регистр B — произведение GH, в регистр E — произведение AB, в регистр L — произведение EF, в регистр D — произведение GH. После этого в паре регистров H, L осуществляется суммирование частных произведений с нужным сдвигом. Команда $DAD H$ используется для сдвига содержимого пары регистров H, L на один разряд влево.

После ассемблирования и ввода программы в ОЗУ ЭВМ СМ-4 директивой R осуществляется отладка программы. Сначала в регистры D и E заносится множимое 02 и множитель 03 и осуществляется запуск программы с нулевого адреса:

*L D: 02, E: 03 <BK>

*E 0000 <BK>

КРОСС-АССЕМБЛЕР К-58В			М И З М	СТРАНИЦА	У
			1 КОНТРОЛЬНЫЙ ПРИМЕР №1		
0000	218FFB	1	ORG	B	
0003	7A	2	LXI	H, BFFBFFH	
0004	A5	3	MOV	A, D	
0005	4F	4	ANA	L	
0006	7A	5	MOV	C, A	
0007	A4	6	MOV	A, D	
0008	47	7	ANA	H	
0009	78	8	MOV	B, A	
000A	A4	9	MOV	A, E	
000B	B1	10	ANA	H	
000C	57	11	ORA	C	
000D	78	12	MOV	D, A	
000E	A5	13	MOV	A, E	
000F	88	14	ANA	L	
0010	6F	15	ORA	B	
0011	78	16	MOV	L, A	
0012	BF	17	MOV	A, E	
0013	BF	18	RRC		
0014	BF	19	RRC		
0015	BF	20	RRC		
0016	5F	21	RRC		
0017	E6BF	22	MOV	E, A	
0018	88	23	ANI	BFFH	
0019	47	24	ORA	B	
001A	78	25	MOV	B, A	
001B	A4	26	MOV	A, E	
001C	B3	27	ANA	H	
001D	5F	28	ORA	E	
001E	268B	29	MOV	E, A	
001F	4E	30	MVI	H, TBL AND BFFH	
0020	68	31	MOV	C, M	
0021	46	32	MOV	L, B	
0022	68	33	MOV	B, M	
0023	46	34	MOV	L, E	
0024	68	35	MOV	E, M	
0025	5E	36	MOV	L, D	
0026	6A	37	MOV	L, M	
0027	6E	38	MOV	D, B	
0028	58	39	SUB	A	
0029	97	40	MOV	H, A	
002A	67	41	MOV	B, A	
002B	47	42	DAD	B	
002C	89	43	DAD	H	
002D	23	44	DAD	H	
002E	29	45	DAD	H	
002F	29	46	DAD	H	
0030	29	47	DAD	D	
0031	19	48	HLT		
0032	75	49	GRB	4E0BH	
4E20	000000	50	DB	0BH, 0BH, 0BH, 0BH	
4E23	00	51	DB	0BH, 0BH, 0BH, 0BH	
4E24	000000	52	DB	0BH, 0BH, 0BH, 0BH	
4E27	00	53	DB	0BH, 0BH, 0BH, 0BH	
4E28	000000	54			
4E29	00	55			

Рис. 2.13. Листинг программы умножения двух 8-разрядных положительных чисел

Здесь и далее информация, выводимая монитором на экран дисплея подчеркивается. После выполнения программы и останов по команде HLT (Останов) (строка 49 рис. 2.13) монитор выводит текст:

ОСТАНОВ

ВАША ПРОГРАММА ВЫПОЛНИЛАСЬ ЗА 270 ТАКТОВ

Затем с помощью директивы L контролируется содержимое регистров процессора:

*L <BK>

A=00 B=00 C=00 D=00 E=00 H=00 L=00
PC=0032 SP=0000

В паре регистров H-L должен содержаться результат. Видно, что программа работает неправильно. Поэтому после команд формирования номеров строк, содержащих частичные произведения (строки 12, 16, 25 и 29 программы), устанавливаются точки контрольных остановов. По этим же адресам с помощью директивы T3 задается контрольная выдача содержимого регистров. Директива O указывает на отсутствие необходимости трассировки каких-либо ячеек ЗУ:

* B 000D, 0011, 001B; 001F <BK>

*T3 000C, 0010, 001A, 001E <BK>

*O <BK>

ВВЕДИТЕ АДРЕСА 10 ЯЧЕЕК ПАМЯТИ, СОДЕРЖИМОЕ
КОТОРЫХ ВЫ ХОТИТЕ УЗНАТЬ ПРИ ТРАССИРОВКЕ

I: <BK>

:

:

10: <BK>

ВВЕДИТЕ ПРИЗНАКИ КОНТРОЛЬНОЙ ПЕЧАТИ

PPP = 1 <BK>

PPM = 0 <BK>

После занесения в регистры D и E множимого и множителя и запуска программы на каждой контрольной точке осуществляется выдача содержимого регистров. Для продолжения работы программы после контрольных остановов используется директива E:

* LD :02, E:03 <BK>

* E 0000 <BK>

A=02 B=00 C=02 D=02 E=03 H=F0 L=0F
PC=000D SP=0000

* E <BK>

A=03 B=00 C=02 D=02 E=03 H=F0 L=03
PC=0011 SP=0000

* E <BK>

A=00 B=00 C=02 D=02 E=30 H=F0 L=03
PC=001B SP=0000

* E <BK>

A=00 B=00 C=02 D=02 E=00 H=F0 L=03
PC=001FSP=0000

В результате анализа содержимого регистров в контрольных точках можно сделать вывод, что номера строк таблицы, содержащих частичные произведения, формируются неверно. После отмены всех контрольных точек и установки контрольной точки останова на последнем, видимо верном, операторе осуществляется прогон про-

граммы до контрольного останова:

* B <BK>

* B ØØ1B <BK>

* L D:Ø2, L:Ø3 <BK>

* E ØØØØ <BK>

Затем программа выполняется в пошаговом режиме с помощью директивы S:

* S <BK>

.

.

Предположим, что в результате найдена ошибочная команда по адресу ØØ1D (команда ORA E вместо ORA C). Осуществляется исправление команды (B1 — шестнадцатеричный код команды ORA C):

* D ØØ1D: B1 <BK>

Далее производится отмена контрольного останова и запуск программы:

* B <BK>

* L D:Ø2, E:Ø3 <BK>

* E ØØØØ <BK>

ОСТАНОВ

ВАША ПРОГРАММА ВЫПОЛНИЛАСЬ ЗА 27Ø ТАКТОВ

* L <BK>

A=ØØ B=ØØ C=ØØ D=ØØ E=Ø6 H=ØØ L=Ø6

PC=ØØ32 SP=ØØØØ

В данном случае в паре регистров H-L получен правильный результат.

Глава 3.

Проектирование основных блоков микропроцессорной аппаратуры

Микропроцессорный комплект БИС серии КР580 позволяет строить микропроцессорные системы различной конфигурации и назначения. Описываемые в настоящей главе блоки являются типовыми и используются в системах различной сложности — от простейших до самых сложных. Обозначение сигналов всех микросхем дается в соответствии с [18].

Процессорный блок, схемы сопряжения с устройством ввода с перфоленты и бытовым магнитофоном разработаны на основе аналоговых блоков отладочного комплекса, описанного в [9, 10].

3.1. Процессорный блок на основе микропроцессора КР580ИК80А

Основой любой микропроцессорной системы на базе серии КР580 является процессорный блок, построенный на МП КР580ИК80А. Кроме самого МП при построении этого блока используются микросхемы контроллера прерываний КР589ИК14, многорежимного буферного регистра К589ИР12 и шинного формирователя К589АП16. Подробное описание этих микросхем приведено в [5]. Описание микросхем серии К155, также используемых при построении процессорного и других блоков микропроцессорной аппаратуры, можно найти в [23]. Встречающееся на принципиальных электрических схемах обозначение «1» указывает на подключение вывода микросхемы к источнику питания 5 В через резистор 1 кОм.

Процессорный блок содержит (рис. 3.1): МП КР580ИК80А; генератор тактовых импульсов, схемы управляющих сигналов, буфер шины данных, буфер шины адреса, схему сброса, схему готовности, блок обработки прерываний.

Генератор тактовых импульсов предназначен для формирования двухфазных синхроимпульсов. Основой генератора импульсов служит симметричный мультивибратор, собранный на элементах D1.1 и D1.2 (рис. 3.2,а), вырабатывающий положительные прямоугольные импульсы с частотой 5 МГц, которые поступают на вход делителя частоты на 4, собранного на триггерах D2.1 и D2.2. Элемент D1.4 формирует импульсы C1. В связи с тем, что амплитуда тактовых импульсов, подаваемых на МП КР580ИК80А, должна составлять 12 В, инвертированные импульсы $\overline{C1}$ и C2 подаются на преобразователи уровня D3.1 и D3.2, что обеспечивает формирование импульсов амплитудой 12 В и частотой 1,25 МГц (рис. 3.2,б). Кроме того, с генератора снимаются импульсы $\overline{C1TTЛ}$ и $\overline{C2TTЛ}$ обычной для ТТЛ-схем амплитуды. Частота получаемых импульсов определяется кварцевым резонатором и его заменой может быть увеличена до 2,5 МГц.

Схемы управляющих сигналов обеспечивают формирование и буферизацию сигналов управления. В начале каждого цикла процессор выдает на шину данных ШД [Ø—7] управляющее слово, указывающее на действия, которые будут выполняться в данном цикле (см.

табл. 1.2). По сигналам С (Синхронизация) процессора и С1ТТЛ генератора тактовых импульсов это управляющее слово записывается в многорежимный буфер регистр К589ИР12 (D1, рис. 3.3) и хранится в нем до

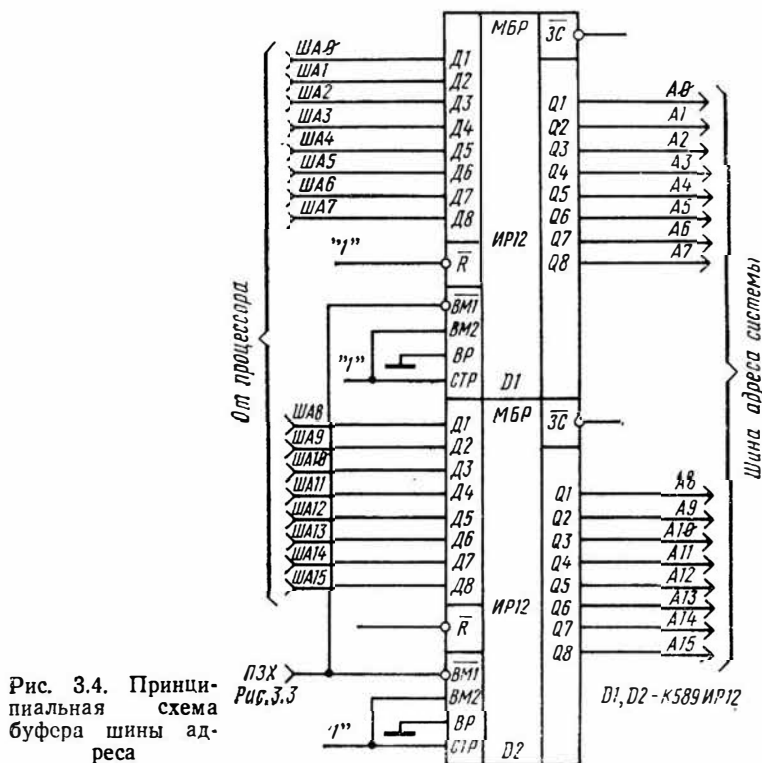
Таблица 3.1

Сигналы шины управления

Обозначение	Название	Условия возникновения	Источник сигнала
$\overline{\text{ЗПВВ}}$	Запись данных в канал вывода	Вывод данных в канал вывода по команде OUT (ВЫВОД)	Процессорный блок
$\overline{\text{ЗПЗУ}}$	Запись данных в ЗУ	Запись данных в ЗУ	То же
$\overline{\text{ЧТВВ}}$	Чтение данных с канала ввода	Ввод данных из канала ввода по команде IN (ВВОД)	„ „
$\overline{\text{ЧТЗУ}}$	Чтение данных ЗУ	Чтение данных из ЗУ	„ „
ВХГТ	Вход готовности	Готовность блока ЗУ к приему или передаче данных	Блок ЗУ
$\overline{\text{ОЖ}}$	Ожидание	Данные в блоке ЗУ не готовы, процессор в состоянии ожидания	Процессорный блок
$\overline{\text{ЗХ}}$	Захват шин системы	Запрос на режим прямого доступа к памяти	Внешнее устройство или контроллер прямого доступа
ПЗХ	Предоставление захвата шин	Разрешение режима прямого доступа к памяти	Процессорный блок
СБР	Сброс	Сброс системы в начальное состояние	Схема сброса процессорного блока
$\overline{\text{ЗП}\overline{\text{О}}}-\overline{\text{ЗП}}\overline{7}$	Запросы прерывания	Запросы прерывания с приоритетами $\overline{\text{О}}-7$	Внешние устройства
С2ТТЛ	Тактовые импульсы	Постоянно формируются тактовым генератором	Тактовый генератор процессорного блока

начала следующего цикла. С выводов регистра D1 управляющие сигналы ППР (Подтверждение прерывания), ВЫВ (Вывод), Ц1 (Первый цикл команды), ВВД (Ввод), ЧТ (Чтение из памяти), входящие в управляю-

щее слово, передаются на схемы И—НЕ D5, D6. Сигналы ППР, ВВД, ЧТ на элементах D5.1, D6.2, D6.4 стробируются управляющим сигналом ПМ (Прием байта), который поступает с процессора через усилители мощности D2.3 и D2.4. Аналогично сигнал ВВВ стробируется



управляющим сигналом ВД (Выдача), подаваемым на элемент D5.2 через усилитель мощности D3.1.

Полученные сигналы $\overline{\text{ЧТЗУ}}$ (Чтение данных ЗУ), $\overline{\text{ЧТВВ}}$ (Чтение с устройства ввода), $\overline{\text{ЗПЗУ}}$ (Запись данных в ЗУ), $\overline{\text{ЗПВВ}}$ (Запись данных на устройство вывода) поступают на шинный формирователь D7 (микросхема К589АП16). При поступлении с МП сигнала ПЗХ (Подтверждение захвата шин) выходы формирователя D7 устанавливаются в состояние с высоким выходным сопротивлением.

В табл. 3.1 приведены сигналы шины управления типовой микропроцессорной системы на базе МПК КР580.

Буфер шины адреса служит для повышения нагрузочной способности ША [Ø—15] микропроцессора. Он выполнен на базе схем многорежимных буферных реги-

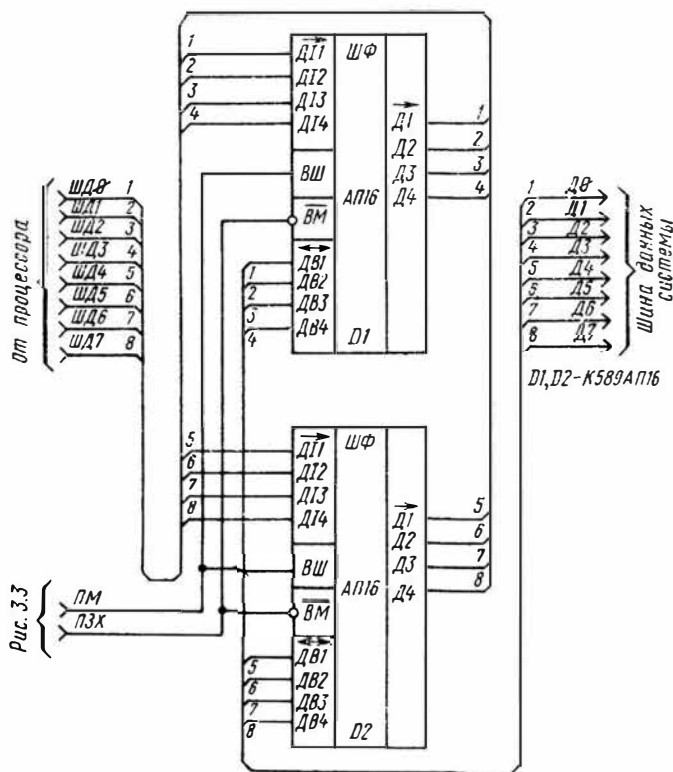


Рис. 3.5. Принципиальная схема буфера шины данных

стров К589ИР12 (рис. 3.4). В нормальном режиме работы системы сигнал ПЗХ, поступающий на входы ВМ1 (Выбор микросхемы 1) схем D1 и D2, имеет низкий уровень и сигналы адресных выходов ША [Ø—15] микропроцессора передаются на шину адреса микропроцессорной системы А [Ø—15]. В режиме же прямого доступа к памяти высокий уровень сигнала ПЗХ вызы-

вает переход выходных линий Q1—Q8 элементов D1, D2 в состояние с высоким выходным сопротивлением.

Буфер шины данных на двух шинных формирователях K589АП16 (элементы D1, D2, рис. 3.5) служит для увеличения нагрузочной способности выводов данных МП. При вводе данных в МП на вход ВШ (Выбор шины) подается сигнал ПМ (Прием) высокого уровня со схемы формирования управляющих сигналов (см. рис. 3.3). При этом информация с шины данных системы ШД [Ø—7] (выводы ДВ [1—4] элементов D1, D2) проходит на шину данных процессора (выводы Д [1—4]). При выводе данных из процессора сигнал ПМ имеет низкий уровень, что обеспечивает передачу данных от процессора (с входов Д1 [1—4] элементов D1, D2) на шину данных системы (выводы ДВ [1—4]).

При поступлении на входы $\overline{ВМ}$ (Выбор микросхемы) элементов D1, D2 сигнала ТПЗХ высокого уровня выходы ДВ [1—4] переходят в состояние с высоким выходным сопротивлением и шина данных МП отключается от шины данных системы.

Схема сброса используется для установки микропроцессорной системы в начальное состояние. Сигнал сброса поступает на микропроцессорную систему с внешнего кнопочного переключателя, расположенного на пульте микро-ЭВМ, или от какого-то другого источника и является асинхронным. Схема формирования сигнала сброса обеспечивает согласование сигнала $\overline{ВХСБР}$ (Вход сброса) с тактовым импульсом С2. Сигнал $\overline{ВХСБР}$ с кнопочного переключателя через инвертор D1.1 (рис. 3.6) поступает на вход D триггера D2.1 и заносится в него в момент поступления импульса С2ТТЛ, который подается на вход синхронизации триггера.

Выходной сигнал триггера D2.1 является сигналом сброса СБР и подается на соответствующий вход микропроцессора. Кроме того, этот сигнал в качестве одного из сигналов управляющей шины микропроцессорной системы подается на другие блоки системы. С инверсного выхода элемента D2.1 сигнал СБР поступает на многорежимный буферный регистр K589ИР12 (элемент D5, рис. 3.8), служащий для запоминания вектора прерывания.

Схема готовности. Входной сигнал готовности ВХГТ

с медленно действующего устройства является асинхронным и может поступить на процессор в произвольный момент времени, его согласование с тактовым импульсом С2 обеспечивает схема готовности. Сигнал ВХГТ поступает на вход элемента D1.1 (рис. 3.7) и проходит его при наличии разрешающего сигнала с триггера D3.2. Этот триггер может быть установлен в единичное состояние двумя способами.

1. Микро-ЭВМ находится в автоматическом режиме работы, т. е. сигнал ШГРЖ (Пошаговый режим), поступающий на вывод 4 схемы D2.2, имеет низкий уро-

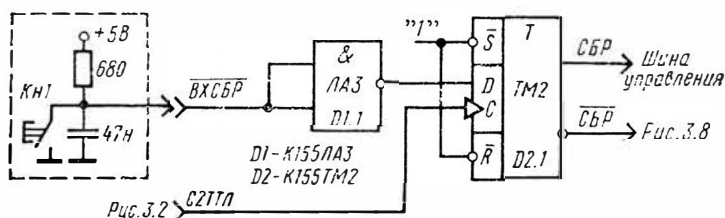


Рис. 3.6. Принципиальная схема формирования сигнала сброса

вень. В этом случае триггер D3.2 устанавливается в единицу по входу S.

2. Микро-ЭВМ находится в пошаговом режиме (верхнее положение переключателя S1). Поскольку на вход D триггера D3.2 постоянно подается высокий уровень, то с приходом на вход C сигнала с RS-триггера, собранного на элементах D1.2 и D2.1, триггер D3.2 устанавливается в единицу. Сигнал на выходе RS-триггера (вывод 3 схемы D2.1) изменяет свой уровень с нулевого на единичный при нажатии кнопки ШГ (Шаг). При этом на вывод 2 схемы D2.1 поступает низкий уровень сигнала. При отжатом состоянии кнопки ШГ низкий уровень сигнала поступает на вывод 3 схемы D1.2.

При наличии разрешающего сигнала с триггера D3.2 асинхронный сигнал ВХГТ проходит элементы D1.1 и D2.3 и в момент действия синхроимпульса С2 записывается в триггер D3.1 по входу $\overline{R}$. Выходной сигнал триггера D3.1 поступает на вход ГТ микропроцессора. После подачи сигнала ГТ высокого уровня МП выводится из состояния ожидания и прекращает выдачу

сигнала ОЖ. При этом через D2.2 осуществляется сброс триггера D3.2 по входу $\bar{R}$. Таким образом, отработав один цикл, микропроцессор снова останавливается в состоянии ожидания. Последовательное нажатие кнопки ШГ обеспечивает выполнение программы в пошаговом режиме.

Блок обработки прерываний состоит из БИС блока приоритетного прерывания К589ИК14 (элемент D3), регистра хранения кода команды RST на основе ИС К589ИР12 (элемент D5), дешифратора адреса блока

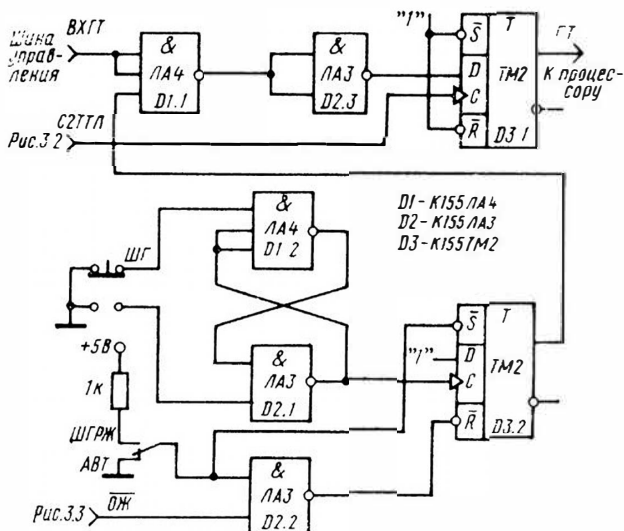


Рис. 3.7. Принципиальная схема формирования сигнала готовности

обработки прерываний в виде схемы совпадения на 8 входов (элемент D1) и вспомогательных схем (рис. 3.8).

Рассматриваемый блок обработки прерываний позволяет реализовать восьмиуровневую систему прерываний и обеспечивает формирование кода команд RST, выдаваемых на шину данных.

Регистр текущего состояния БИС К589ИК14 [5] для МП является устройством вывода с некоторым адресом (на рис. 3.8 с адресом FFH). Запись кода приоритета в этот регистр осуществляется командой OUT 0FFH. Перед выполнением этой команды в накопитель загрузки

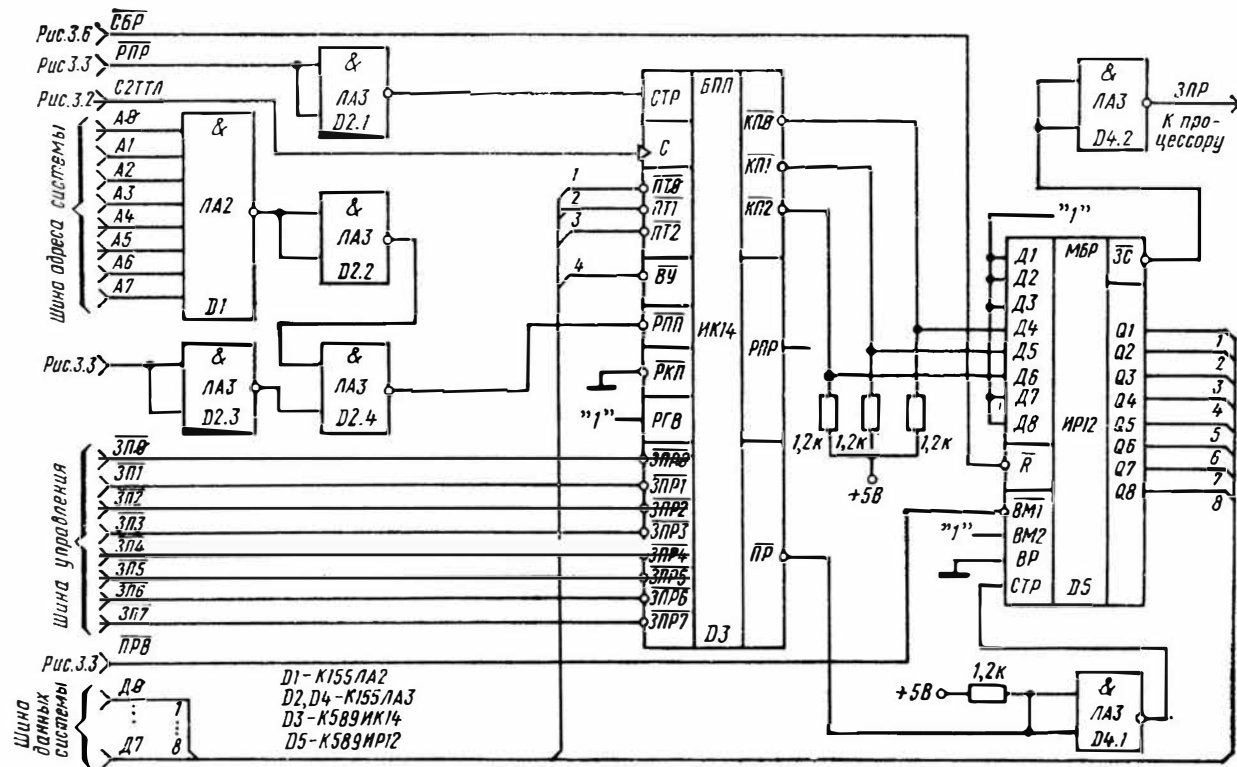


Рис. 3.8. Принципиальная схема блока обработки прерываний

жается управляющее слово, разряды $\emptyset$ —2 которого содержат код приоритета. Разряд 3 управляющего слова, поступающий на вход $\overline{ВУ}$ (Выбор группы уровней прерываний) микросхемы К589ИК14, должен иметь нулевое значение, а на входе $\overline{РПП}$ (Разрешение приема текущего приоритета) должен быть сигнал низкого уровня. При этом разрешается прием кода приоритета в регистр текущего состояния. Сигнал низкого уровня на входе $\overline{РПП}$ вырабатывается при совпадении на элементе D2.4 сигнала с дешифратора младшего байта адреса (элемент D1) и сигнала $\overline{3ПВВ}$ шины управления, возникающего при выполнении команды $\overline{OUT}$. Сигнал с дешифратора D1 возникает при выдаче на младшие 8 разрядов шины адреса кода FFH, что соответствует в описываемом процессорном блоке адресу блока обработки прерываний.

Вход $\overline{РКП}$ (Разрешение выдачи информации на выходы) БИС К589ИК14 заземлен. Поэтому сформированный код принятого прерывания сразу же передается на выходы $\overline{КП\emptyset}$ — $\overline{КП2}$, а на выходе $\overline{ПР}$ (Прерывание) появляется сигнал низкого уровня. Эти сигналы поступают на входы Д4—Д6 и СТР (Стробирование) регистра формирования команды RST (элемент D5). Поскольку вход ВР (Выбор режима) этого элемента заземлен, то код принятого прерывания, поступающий на входы

Таблица 3.2

Формирование команды RST

Вход, на который поступил запрос прерывания	Команда	Код команды								Адрес подпрограммы обслуживания прерывания
		Разряд 7	Разряд 6	Разряд 5	Разряд 4	Разряд 3	Разряд 2	Разряд 1	Разряд $\emptyset$	
$\overline{3ПР\emptyset}$	RST 7	1	1	1	1	1	1	1	1	$7\emptyset_8 = \emptyset\emptyset 38H$
$\overline{3ПР1}$	RST 6	1	1	1	1	$\emptyset$	1	1	1	$6\emptyset_8 = \emptyset\emptyset 3\emptyset H$
$\overline{3ПР2}$	RST 5	1	1	1	$\emptyset$	1	1	1	1	$5\emptyset_8 = \emptyset\emptyset 28H$
$\overline{3ПР3}$	RST 4	1	1	1	$\emptyset$	1	1	1	1	$4\emptyset_8 = \emptyset\emptyset 20H$
$\overline{3ПР4}$	RST 3	1	1	$\emptyset$	1	1	1	1	1	$3\emptyset_8 = \emptyset\emptyset 18H$
$\overline{3ПР5}$	RST 2	1	1	$\emptyset$	1	$\emptyset$	1	1	1	$2\emptyset_8 = \emptyset\emptyset 1\emptyset H$
$\overline{3ПР6}$	RST 1	1	1	$\emptyset$	$\emptyset$	1	1	1	1	$1\emptyset_8 = \emptyset\emptyset \emptyset 8H$
$\overline{3ПР7}$	RST $\emptyset$	1	1	$\emptyset$	$\emptyset$	$\emptyset$	1	1	1	$\emptyset$

Д4—Д6 регистра К589ИР12, запоминается в этом регистре. Одновременно по входам Д1—Д3, Д7—Д8 в регистр записываются единицы.

Таким образом, в регистре К589ИР12 формируется код команды RST (табл. 3.2). В связи с тем, что с выходов КПØ—КП2 код приоритета выдается в инверсном коде, сигналу прерывания низшего приоритета (вход ЗПРØ) соответствует команда RST 7, т. е. подпрограмма обслуживания этого прерывания располагается по адресу ØØ38H, а сигналу прерывания высшего приоритета (вход ЗПР7) — команда RST Ø. На выходе ЗС (Запуск) появляется сигнал низкого уровня, который после инверсии на элементе D4.2 подается на вход ЗПР (Запрос прерывания) микропроцессора. При поступлении на вход ВМ1 элемента D5 сигнала ПРВ (Прерывание) низкого уровня, указывающего на готовность процессора к приему команды RST, ее код из регистра К589ИР12 по выходам Q1—Q8 выдается на шину данных. До поступления сигнала низкого уровня на вход ВМ1 элемента D5 его выходы Q1—Q8 находились в состоянии с высоким сопротивлением, т. е. регистр D5 был отключен от шины данных.

3.2. Проектирование блоков ЗУ микропроцессорных систем

Используемые в микропроцессорных системах ЗУ во многом определяют стоимость последних. При выборе типов БИС ЗУ необходимо учитывать [20] требуемую структуру и информационную емкость блоков памяти, требования к временным и энергетическим параметрам ЗУ, стоимость элементов памяти и предполагаемую тенденцию ее изменения, планируемый объем производства микропроцессорных систем.

Блок ОЗУ статического типа емкостью 4К байт. Построен на основе микросхем К565РУ2А. Эти микросхемы представляют собой ОЗУ статического типа емкостью 1024×1 бит [20]. В состав блока входят массив ИС памяти, состоящий из 32 микросхем К565РУ2А общей емкостью $4К \times 8$ бит, дешифратор старших разрядов адреса и выходной буфер данных.

Массив памяти состоит из элементов DS1—DS32 типа К565РУ2А. Весь массив разбит на 4 линейки по 1К бит

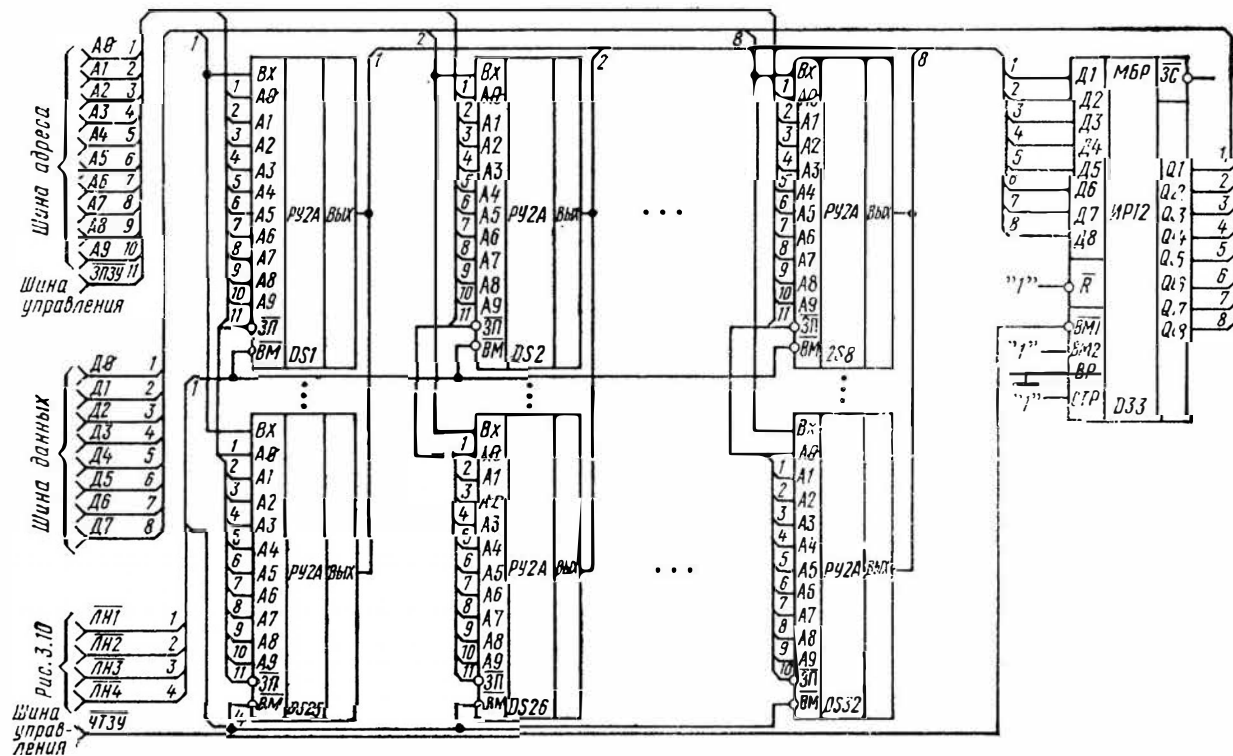


Рис. 3.9. Принципиальная схема массива ОЗУ

(рис. 3.9). Линейка состоит из 8 микросхем (DS1—DS8, DS9—DS16, DS17—DS24, DS25—DS32), что соответствует формату шины данных. Разряды адреса A_0 — A_9 , определяющие выбираемую в БИС ОЗУ ячейку памяти, поступают на все элементы памяти массива. Разряды данных D_0 — D_7 поступают на входы данных микросхем памяти, причем разряд D_0 подается на входы всех первых элементов линеек (DS1, DS9, DS17, DS25), разряд D_1 — на входы вторых элементов (DS2, DS10, DS18,

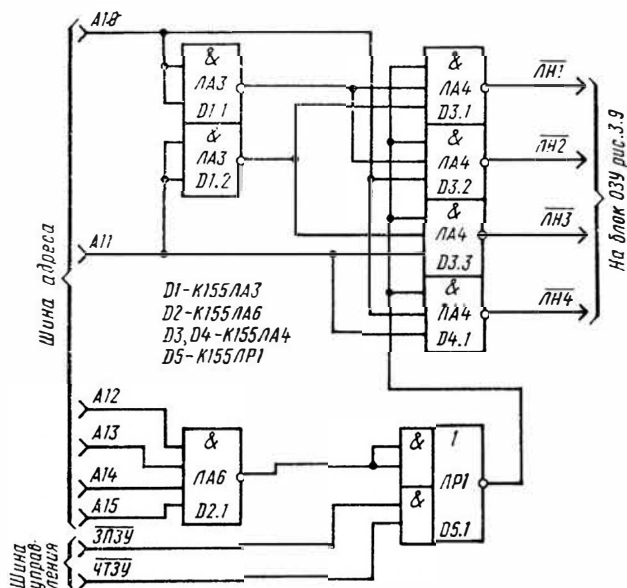


Рис. 3.10. Принципиальная схема управления блоком ОЗУ

DS26) и т. д. Аналогично соединены выходы микросхем. Выходы первых микросхем линеек объединены и образуют нулевой разряд выходных данных, выходы вторых микросхем образуют первый разряд выходных данных и т. д.

Управляющий сигнал $\overline{ЗПЗУ}$ с шины управления системы поступает на входы управления записью $\overline{ЗП}$ всех 32 элементов памяти. При нулевом уровне этого сигнала осуществляется запись информации в ОЗУ, при единичном — считывание. Входы выбора кристалла $\overline{ВМ}$ объединены у всех элементов памяти, стоящих в одной ли-

нейке. При поступлении на них низкого уровня происходит обращение к выбранной линейке, и в зависимости от уровня сигнала ЗПЗУ производится запись или считывание байта данных.

В качестве выходного буфера данных используется многорежимный буферный регистр K589IP12, работающий во входном режиме. При отсутствии сигнала ЧТЗУ низкого уровня выходы регистра Q1—Q8 находятся в состоянии с высоким выходным сопротивлением. При поступлении сигнала ЧТЗУ низкого уровня на вход ВМ1 данные с D0—D7 поступают на выходы Q1—Q8 и далее на шину данных микропроцессорной системы.

Дешифратор старших разрядов адреса обеспечивает выборку одной из четырех линеек БИС ОЗУ. На рис. 3.10 приведена схема дешифратора в предположении, что рассматриваемый блок ОЗУ занимает адресное пространство F000—FFFF. Появление одного из сигналов ЛН1—ЛН4, низкий уровень которых обеспечивает обращение к соответствующей линейке БИС ОЗУ, возможно только при наличии сигналов ЗПЗУ или ЧТЗУ низкого уровня и сигналов высокого уровня на адресных линиях A12—A15.

Скорость работы блока ОЗУ (время цикла БИС K565PY2A составляет 0,4 мкс) согласуется с быстродействием МП. В связи с этим при использовании описанного блока линия ВХГТ шины управления может быть заземлена.

Блок начального загрузчика с перфоленты. В ряде микропроцессорных систем, в частности в аппаратно-программных отладочных комплексах (см. гл. 4), основные программы загружаются в ОЗУ с перфоленты. Чтобы осуществить такую загрузку, необходимо иметь блок ПЗУ или ППЗУ, содержащий программу начальной загрузки с перфоленты. В частности, такой блок может быть построен на основе ИС ПЗУ K155PE3 с пережимаемыми перемычками [23], имеющих организацию 32×8 бит. Эта микросхема имеет выходы с открытым коллектором, что позволяет объединять выходы нескольких схем. Блок начальной загрузки (рис. 3.11) имеет память объемом 64 байта на двух микросхемах K155PE3 (элементы DS5, DS6). Микросхеме DS5 соответствуют адреса от 0 до 31, а микросхеме DS6 — от 32 до 63. При установке переключателя в положение ВКЛ (Включе-

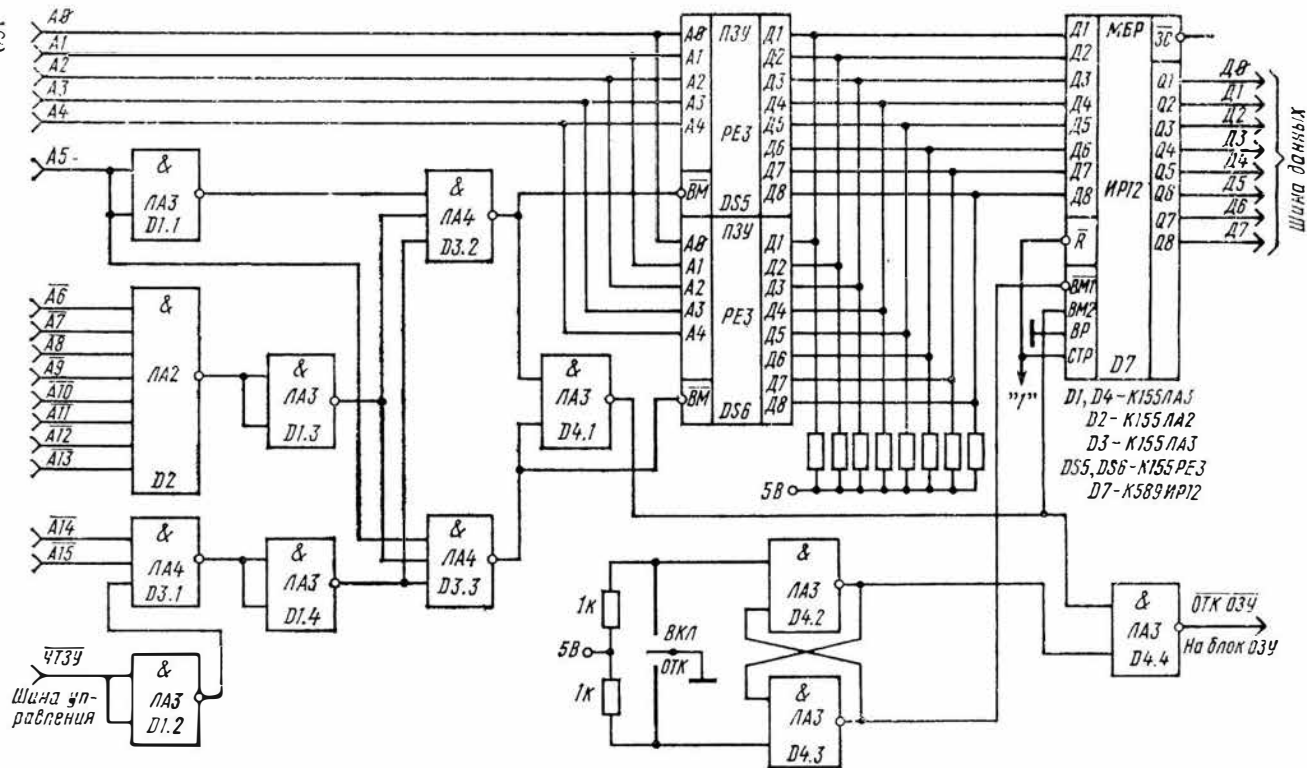


Рис. 3.11. Принципиальная схема блока начальной загрузки

но) на выходе элемента D4.2 сигнал имеет высокий уровень. При адресах в диапазоне 0—63 и наличии сигнала чтения из памяти $\overline{CT3Y}$ на выходе элемента D4.1 сигнал также имеет высокий уровень. При этом на входы D1—D8 буферного регистра D7 поступают данные из одной из схем ПЗУ, на входы выбора микросхемы ВМ2 этого регистра поступают разрешающие сигналы и данные проходят на шину данных системы. Одновременно с элемента D4.4 снимается сигнал запрета работы ОЗУ ОТК ОЗУ. При обращении к ЗУ по адресу, большему 63, происходит обычное обращение к ОЗУ системы. При установке переключателя в положение ОТК (отключено) сигнал на выходе D4.3 имеет высокий уровень, выходы Q1—Q8 буферного регистра D7 переходят в состояние с высоким выходным сопротивлением и блок отключается от шины данных микропроцессорной системы. Программа начальной загрузки приведена в § 3.3.

3.3. Использование БИС программируемого параллельного интерфейса КР580ВВ55 для сопряжения микропроцессорной системы с устройствами ввода — вывода

Сопряжение МП с цифровыми датчиками и исполнительными устройствами, ввод в МП информации о положении двухпозиционных выключателей и управление сигнальными лампочками и светодиодами обычно осуществляется с помощью БИС КР580ВВ55. В таком случае восемь входных логических сигналов ($K\emptyset$ — $K7$, рис. 3.12) подключаются к одному каналу ввода (например, ШКН3). Аналогично к каждому выходному каналу (ШКН1 и ШКН2) подключаются по восемь линий управления ($Y\emptyset$ — $Y7$ и $Y8$ — $Y15$). Работа всех каналов БИС КР580ВВ55 осуществляется в режиме $\emptyset$. На рис. 3.12 БИС КР580ВВ55 использует адреса $F\emptyset$ — $F3$.

Определение состояния входного логического сигнала и вывод выходного сигнала осуществляются с использованием следующих программных блоков:

; программирование режима работы и направления передачи информации

MVI A, 89H ; управляющее слово в накопитель

OUT $\emptyset F3H$; запись в БИС управляющего слова

;
;
;

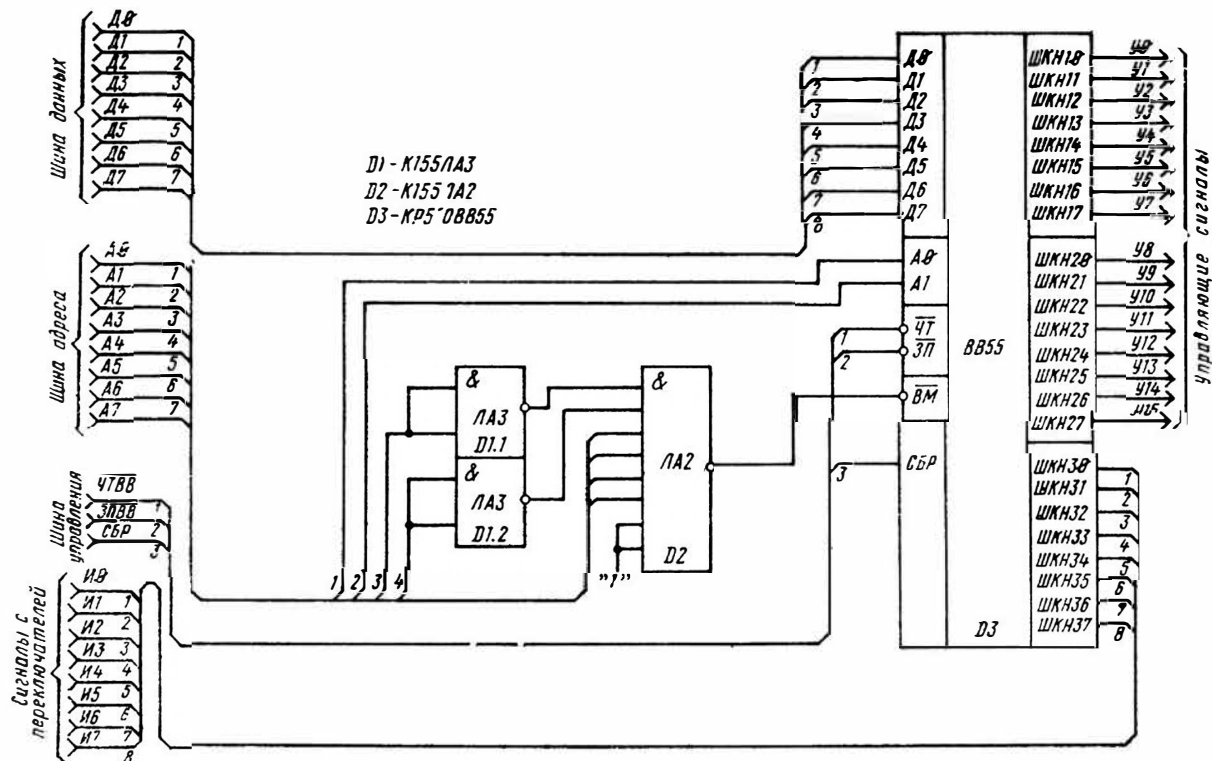


Рис. 3.12. Принципиальная схема сопряжения микропроцессорной системы с двухпозиционными переключателями и индикаторами

; определение состояния входного сигнала
 IN ØF2H ; ввод данных с канала ШКНЗ
 ANI Ø4H ; выделение второго бита
 JNZ M1 ; переход при единичном значении сигнала

; вывод сигнала
 MVI A, Ø8H ; установка в единицу определенного бита
 OUT ØFØH ; выдача содержимого накопителя в канал
 ; ШКН1.

Программатор ППЗУ с ультрафиолетовым стиранием. Одной из отечественных БИС ППЗУ с ультрафиолетовым стиранием является микросхема К573РФ1 [20]. Она выполнена по p -МОП технологии с плавающим затвором, имеет информационную емкость 8192 бит (1024×8 бит), размещается в корпусе с 24 выводами. Для ее работы необходимы напряжения 5; —5 и 12 В.

На входы АØ—А9 (рис. 3.13,а) подается 10-разрядный двоичный адрес считываемого числа. Дешифратор строк ДШХ выбирает одну из 64 строк запоминающей матрицы (каждая строка хранит 128 бит информации), а сигнал с одного из 16 выходов дешифратора ДШУ выбирает из строки 8 бит информации — код одного числа, который через выходные буферы поступает на линии данных ДØ—Д7. Устройство выбора микросхемы обеспечивает выбор нужной БИС при создании большого модуля ППЗУ. При считывании записанной информации на входы АØ—А9 подается код нужного адреса, а на вход $\overline{\text{ВМ}}\text{—ЗП}$ (Выбор микросхемы — Запись) — импульс разрешения выборки БИС; через 900 нс на выходах ДØ—Д7 устанавливается 8-разрядный код считанного числа (рис. 3.13,б).

Первоначально, а также после каждого цикла стирания информации по всем адресам и разрядам записана 1, поэтому программирование заключается в записи 0 в выбранные ячейки матрицы. В режиме записи информации на вход $\overline{\text{ВМ}}\text{—ЗП}$ подается напряжение 12 В. Адрес выбирается так же, как и в режиме считывания. Записываемые в ППЗУ данные подаются на шину ДØ—Д7. После того как сформированы код адреса и код числа, на программирующий вход ПРГ подается импульс 26 В мощностью не менее 0,4 Вт (рис. 3.13,в) и длительностью t . Программирование осуществляется бо-

осуществляется с помощью программатора, который работает под управлением МП. Основой программатора является БИС программируемого параллельного интерфейса КР580ВВ55, работающая в режиме $\emptyset$ (рис. 3.14). Данные, подлежащие записи, поступают на К573РФ1 через канал ШКН2. Младшие 8 разрядов адреса поступают через канал ШКН1, два старшие разряда адреса — через канал ШКН3 [$\emptyset$ —1]. Разряды 4—7 канала ШКН3 используются для подачи на БИС ППЗУ управляющих сигналов.

В данном случае БИС КР580ВВ55 подключается к микропроцессорной системе как внешнее устройство и имеет адреса Е8Н—ЕВН. В режиме записи разряды канала ШКН3 устанавливаются следующим образом: ШКН3[4]=1, ШКН3[5]= $\emptyset$, ШКН3[6]= $\emptyset$, ШКН3[7]= $\emptyset$. С выхода элемента D2.3 с открытым коллектором напряжение 12 В поступает на вход ВМ-3П БИС ППЗУ.

Управление импульсом записи производится разрядами 5,4 и 7 канала ШКН3. Импульс на входе программирования ПРГ микросхемы памяти появляется при ШКН3 [4]=1 и ШКН3 [7]= $\emptyset$. При этом ни при записи управляющего слова в БИС КР580ВВ55, когда все сигналы каналов ШКН1—ШКН3 сбрасываются, ни при поступлении сигнала сброса СБР, что вызывает появление единичных сигналов на линиях каналов, не возникает непредусмотренного сигнала программирования.

В режиме считывания данных с микросхемы К573РФ1 для контроля записи устанавливаются следующие управляющие сигналы: ШКН3 [4]= $\emptyset$, ШКН3 [5]=1, ШКН3 [6]=1, ШКН3 [7]= $\emptyset$. Схема формирования импульса записи отключена от входа программирования сигналом ШКН3 [4]= $\emptyset$.

Информация, которая подлежит записи в ППЗУ, находится по адресу START в ОЗУ микропроцессорной системы. Ячейка с адресом LOOP хранит идентификатор режима работы подпрограммы. Режим 2 (LOOP=2) означает запись в ППЗУ всего массива данных без контрольного чтения. Режим 1 (LOOP=1) означает запись отдельных ячеек ППЗУ при неправильной их записи в режиме 2. Режим 1 позволяет повторять процесс записи информации в те ячейки, для которых недостаточен стандартный цикл записи. Переход от режима 2 к режиму 1 происходит по ходу выполнения подпрограммы.

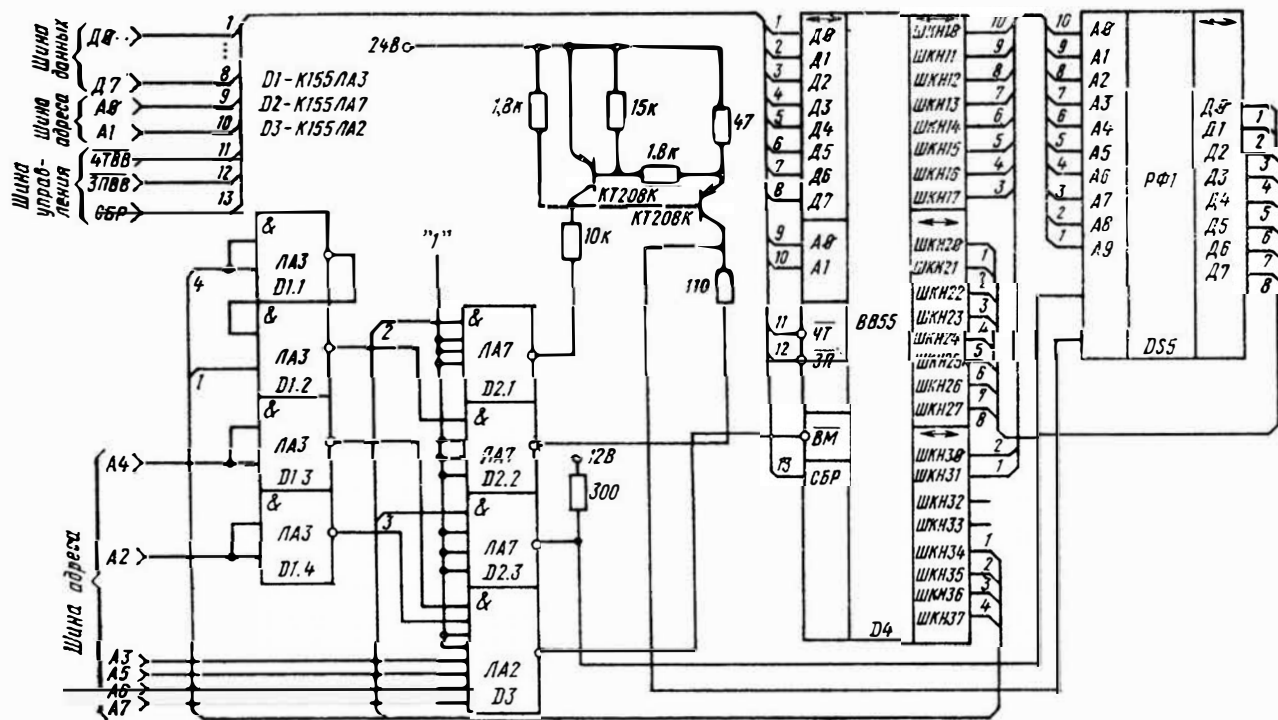


Рис. 3.14. Принципиальная схема программатора ППЗУ с ультрафиолетовым стиранием

Запись производится, начиная с последней ячейки модуля. В паре регистров H-L содержится адрес байта информации ОЗУ, подлежащего записи, а в паре регистров D—E — адрес, по которому он должен быть записан в ППЗУ. Программирование каналов на ввод или вывод информации осуществляется выдачей соответствующего управляющего слова по адресу CENTR.

Сначала в режиме 2 выполняется запись всего массива информации. Запись байта заключается в программировании всех каналов БИС КР580ВВ55 на вывод, установке адреса ячейки ППЗУ, в которую будет записана текущая информация, установке управляющих сигналов (подключение 12 В ко входу ВМ — 3П), загрузке накопителя байтом данных, который должен быть записан, и выдаче его на шину данных ППЗУ, подаче импульса записи определенной длительности.

Подобный цикл записи всего ППЗУ производится 33 раза (значение счетчика-регистра С). Длительность импульса записи устанавливается программным способом и составляет около 3 мс. Далее программа выполняется в режиме 1 (LOOP=1).

Регистр С в этом режиме используется для анализа ошибок. Если во время контрольного чтения обнаруживается ошибочная запись, то программа организует повторение цикла записи для этой одной ячейки, что фиксируется в регистре С и влечет за собой обязательно повторный цикл контрольного чтения. Процесс коррекции ошибочного байта повторяется также 33 раза. Максимальное число циклов контрольного чтения равно 16 (содержимое счетчика-регистра В).

После окончания работы подпрограммы в накопителе содержится специальная информация о результате записи: 00H указывает, что в процессе контрольного чтения не обнаружено ни одной ошибки, т. е. информация, подлежащая записи, полностью совпадает с содержимым ППЗУ; 80H указывает на неверную запись информации.

Подпрограмма управления работой программатора приведена ниже.

; подпрограмма программирования ППЗУ

CENTRL	EQU	0EBH	; адрес регистра управляющего
			; слова
PORTA	EQU	0E8H	; адрес канала ШКН1
PORTB	EQU	0E9H	; адрес канала ШКН2

PORTC	EQU	0EAH	; адрес канала ШКНЗ
START	EQU	1200H	; адрес записываемой
	ORG	1000H	; информации в ОЗУ системы
			; начальный адрес
			; подпрограммы
AGAIN:	LXI	B,1024H	; установка счетчиков циклов
	LHLD	CONST	; CONST = 1024 — объем
			; программируемой памяти
	XCHG		; занесение в DE последнего
			; адреса ППЗУ
	LHLD	START	; START — начальный адрес
			; информации в ОЗУ
	DAD	D	; в HL последний адрес
POL:	DCX	H	; информации в ОЗУ
	DCX	D	; (HL) — 1
TOL:	MVI	A,80H	; (DE) — 1
			; программирование всех
			; каналов на запись
			; в режиме 0
	OUT	CENTRL	; выдача управляющего слова
	MOV	A,E	; установка младшего байта
			; адреса
	OUT	PORTA	; выдача младшего байта адреса
	MOV	A,D	; установка 8,9 разрядов адреса
	ORI	20H	; установка управляющих
			; сигналов
	OUT	PORTC	
	MOV	A,M	; посылка в накопитель (HL)
	OUT	PORTB	; посылка (A) по адресу DE
	MOV	A,D	; включение импульса записи
	ORI	10H	
	OUT	PORTC	
	MVI	A,0FFH	; установка длительности
			; импульса 3 мс
TIME:	DCR	A	
	JNZ	TIME	
	MOV	A,D	; выключение импульса записи
	ORI	20H	
	OUT	PORTC	
	LDA	LOOP	; учет режима записи
	DCR	A	
	JNZ	DOL	; переход при общем цикле
			; записи
	DCR	C	; при записи одной строки
	JNZ	TOL	
	MVI	C,02H	; сигнализация о записи
	JMP	LON	; переход на чтение
DOL:	MOV	A,D	; проверка на конец цикла
			; записи
	ORA	E	
	JNZ	POL	; переход к записи в следующую
			; ячейку
	DCR	C	; счетчик циклов записи всего
			; массива

	JNZ	AGAIN	; переход к следующему циклу ; записи
	MVI	A, 01H	
	STA	LOOP	; изменение режима записи
	MVI	C, 01H	; установка счетчика анализа ; ошибки
READAL:	LHLD	CONST	
	XCHG		; занесение в регистры DE ; последнего адреса ППЗУ
	LHLD	START	
	DAD	D	; в HL последний адрес ; информации в ОЗУ
READ:	DCX	H	; (HL) — 1
	DCX	D	; (DE) — 1
	MVI	A, 82H	; программирование канала ; ШКИ2 на чтение
	OUT	CENTRL	
	MOV	A, E	; установка младшего байта ад- ; реса
	OUT	PORTA	
	MOV	A, D	; установка 8, 9 разрядов ; адреса
	ORI	60H	; установка управляющих ; сигналов
	OUT	PORTC	
	IN	PORTB	; чтение из ППЗУ в накопитель
	CMP	M	; сравнение (A) и ((HL))
	JZ	LON	; переход в случае совпадения
	MVI	C, 21H	; установка счетчика для ; записи одной строки
	JMP	TOL	; переход на исправление одной ; строки
LON:	MOV	A, D	; проверка на нулевую ячейку
	ORA	E	
	JNZ	READ	; на чтение содержимого ; следующей ячейки
	DCR	C	; учет ошибки при записи
	JZ	BOL	; переход на окончание ; подпрограммы
	DCR	B	; счетчик цикла контрольного ; чтения
	JNZ	READAL	; на новый цикл чтения всего ; модуля
	MVI	A, 80H	; сигнализация о возможной ; ошибке при записи
BOL:	RET		
	MVI	A, 00H	; сигнализация о правильной ; записи программы
CONST:	RET		
LOOP:	DW	0400H	; 1024
	DB	02H	; метка режима
	END		

Сопряжение микропроцессорной системы с алфавитно-цифровым дисплеем «Видеотон-340», перфорато-

ром ПЛ-150 и устройством ввода с перфоленты FS-1501. Для подключения указанных устройств к микропроцессорной системе на базе МПК КР580 используется БИС КР580ВВ55, работающая в режиме 1. При работе с микропроцессорной системой дисплей «Видеотон-340» используется в режиме ON LINE (Автоматический режим). Передача и прием информации производятся по байтно. Каждый байт соответствует одному алфавитно-цифровому символу в коде КОИ-7, старший бит используется для контроля на четность. Подключение дисплея к системе осуществляется через его параллельный интерфейс (табл. 3.3). Данные на вход ID [Ø—7] дис-

Таблица 3.3

Сигналы интерфейса алфавитно-цифрового дисплея
„Видеотон-340“

Обозначение	Наименование	Описание
OD [Ø—7]	Выходные данные, разряды Ø—7	Шина, по которой из дисплея выдается байт данных
STRBO	Стробирование выхода (выходной)	Указывает на наличие данных на выходе OD [Ø—7] дисплея при выводе из него информации
DMDO	Запрос на выдачу данных (входной)	Приемник информации готов принять данные с выхода OD [Ø—7]
ID [Ø—7]	Входные данные, разряды Ø—7	Шина, по которой осуществляется прием информации в дисплей
STRBI	Стробирование входа (входной)	Указывает на наличие данных на входах ID [Ø—7] при вводе данных
DMDI	Запрос на ввод данных (выходной)	Дисплей готов принять данные с входа ID [Ø—7]

плея (рис. 3.15) передаются с выхода канала ШКН1 через формирователь К589ИР12 с одновременной выдачей сигнала КГТ1 (разряд 7 канала ШКН3). Этот сигнал после инвертирования подается на вход STRBI дисплея. Изменение уровня сигнала DMDI, поступающего с дисплея, с высокого на низкий подтверждает передачу символа.

Информация с выхода дисплея OD [Ø—7] поступает на вход канала ШКН2, работающего на ввод. Инфор-

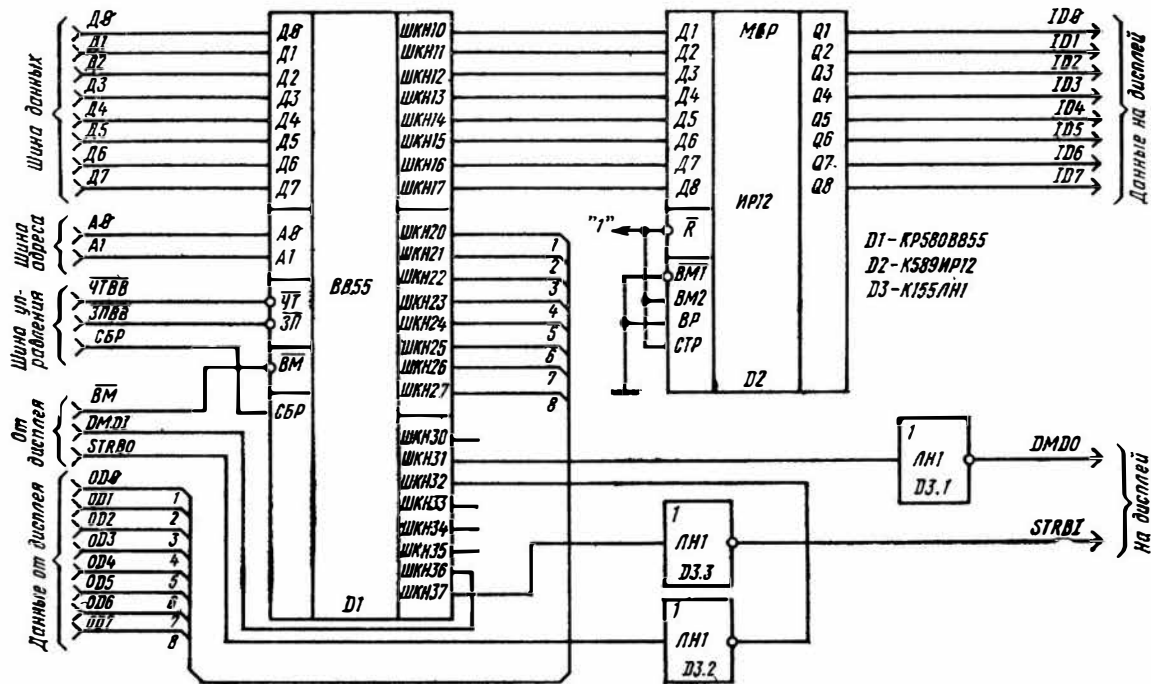


Рис. 3.15. Принципиальная схема сопряжения микропроцессорной системы с алфавитно-цифровым дисплеем «Видеотон-340»

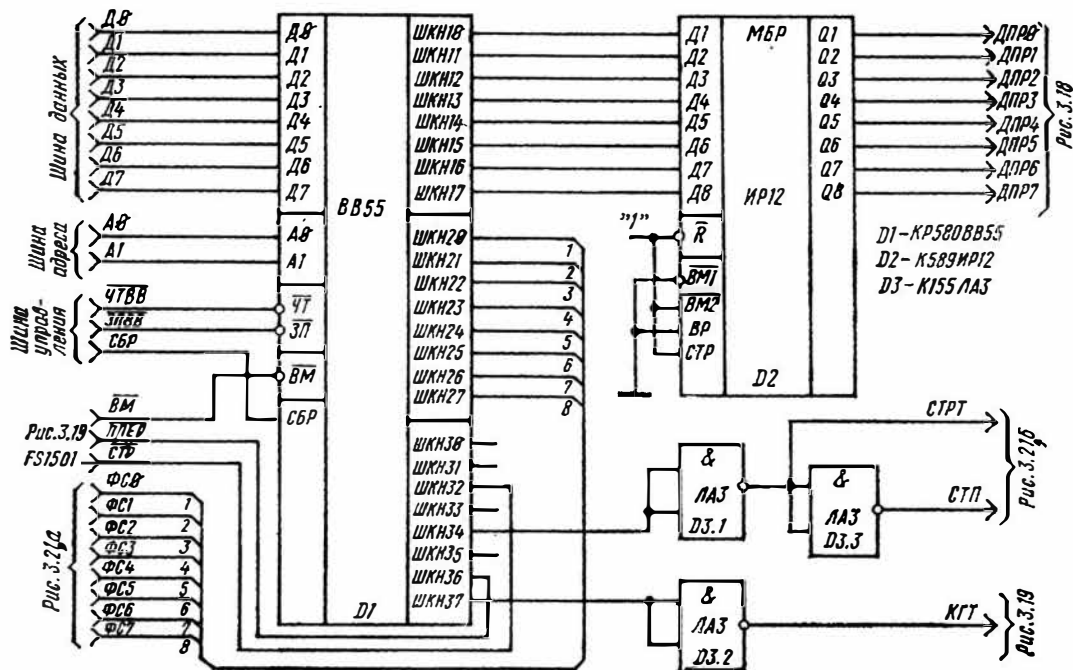


Рис. 3.16. Принципиальная схема сопряжения микропроцессорной системы с перфоратором ПЛ-150 и устройством ввода с перфоленты FS-1501

мация сопровождается сигналом STRBO, который после инвертирования преобразуется в сигнал $\overline{СТР2}$ (см. рис. 1.12) и подается на второй разряд канала ШКН3. Сигнал подтверждения приема (разряд 1 канала ШКН3) после инвертирования служит сигналом DMDO.

Обращение к дисплею осуществляется следующим образом:

```

; начальная установка режима
MVI A, ØA7H      ; посылка в накопитель управляющего
                  ; слова
OUT ØF7H          ; программирование канала ШКН1 на вывод,
                  ; а канала ШКН2 на ввод в режиме 1
MVI A, ØDH        ; установка триггера разрешения
                  ; готовности канала ШКН1
                  ; (разряд 6 канала ШКН3)
MVI A, Ø5H
OUT ØF7H          ; установка триггера разрешения
                  ; готовности канала ШКН2 (разряд 2 канала
                  ; ШКН3)
.
.
.
; выдача символа, расположенного в регистре С, на дисплей
M1: IN ØF6H       ; ввод байта канала ШКН3
    ANI Ø3H       ; выделение третьего разряда — сигнала ГТВ
                  ; канала ШКН1
    JZ M1
    MOV A, C      ; пересылка выдаваемого символа в накопитель
    OUT ØF4H      ; выдача символа через канал ШКН1
; ввод символа с клавиатуры дисплея и занесение в регистр С
M2: IN ØF6H       ; ввод байта канала ШКН3
    ANI Ø1H       ; выделение нулевого разряда — сигнала ГТВ
                  ; канала ШКН3
    JZ M2
    IN ØF5H       ; ввод символа через канал ШКН2
    MOV C, A      ; пересылка символа в регистр С

```

Перфоратор ПЛ-150 подключается к микропроцессорной системе через канал ШКН1, а устройство ввода с перфоленты FS-1501 — через канал ШКН2 БИС КР580ВВ55, работающей в режиме 1 (рис. 3.16).

При включенном двигателе перфоратора при каждом обороте вала вырабатываются синхронизирующие импульсы ДП1, ДП2, ДП3 (рис. 3.17). Для перфорации данных в момент действия импульса ДП2 входы соответствующих дорожек ВХД [Ø—7] с помощью токовых ключей КЛ [Ø—7] (рис. 3.18) заземляются, что вызывает протекание тока через соленоиды и пробивку от-

верстей на нужных дорожках. Заземление входа синхродорожки (ВХС) с помощью токового ключа синхродорожки (КЛС) обеспечивает ее перфорацию. Для транспортировки ленты в момент действия импульса ДПЗ (см. рис. 3.17) с помощью ключа транспортировки КЛТ заземляется вход транспортировки (ВХТ) перфоратора, что вызывает продвижение перфоленты на одну строку.

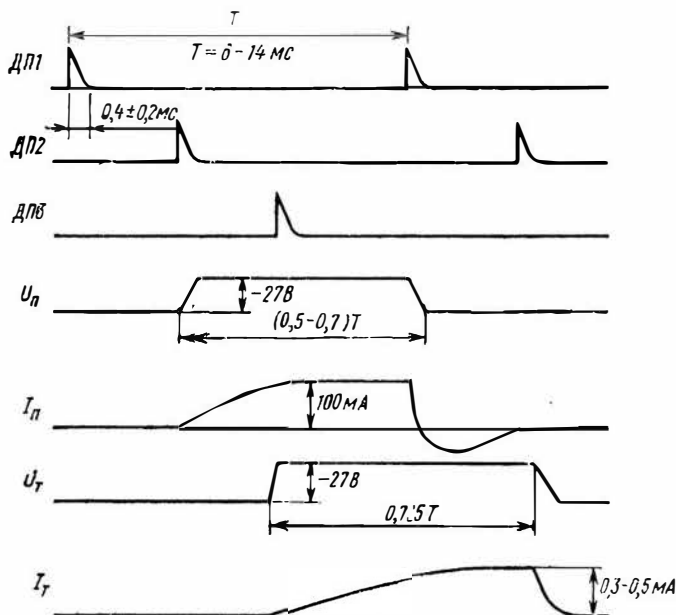


Рис. 3.17. Временные диаграммы работы перфоратора ПЛ-150:
 U_n , I_n — напряжение и ток соленоида перфорации; U_t , I_t — напряжение и ток транспортировки ленты; T — время одного оборота двигателя

Заземление входа двигателя (ВХДВ) обеспечивает включение последнего. В качестве токовых ключей используются ячейки управления перфоратором от мини-ЭВМ «Электроника-100/И».

Включение перфоратора производится переключателем ВК1, при этом на вход ключа КЛДВ поступает сигнал высокого уровня ВКПР (Включение перфоратора). При заправленной перфоленте на другой вход ключа КЛДВ поступает сигнал высокого уровня, свидетельст-

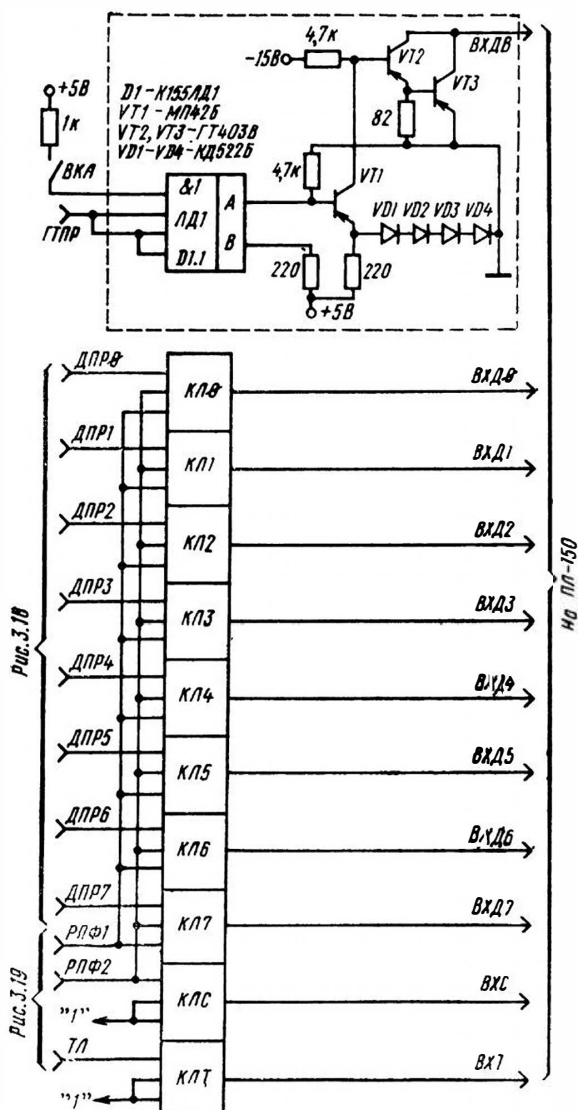


Рис. 3.18. Принципиальная схема токовых ключей для подключения перфоратора ПЛ-150

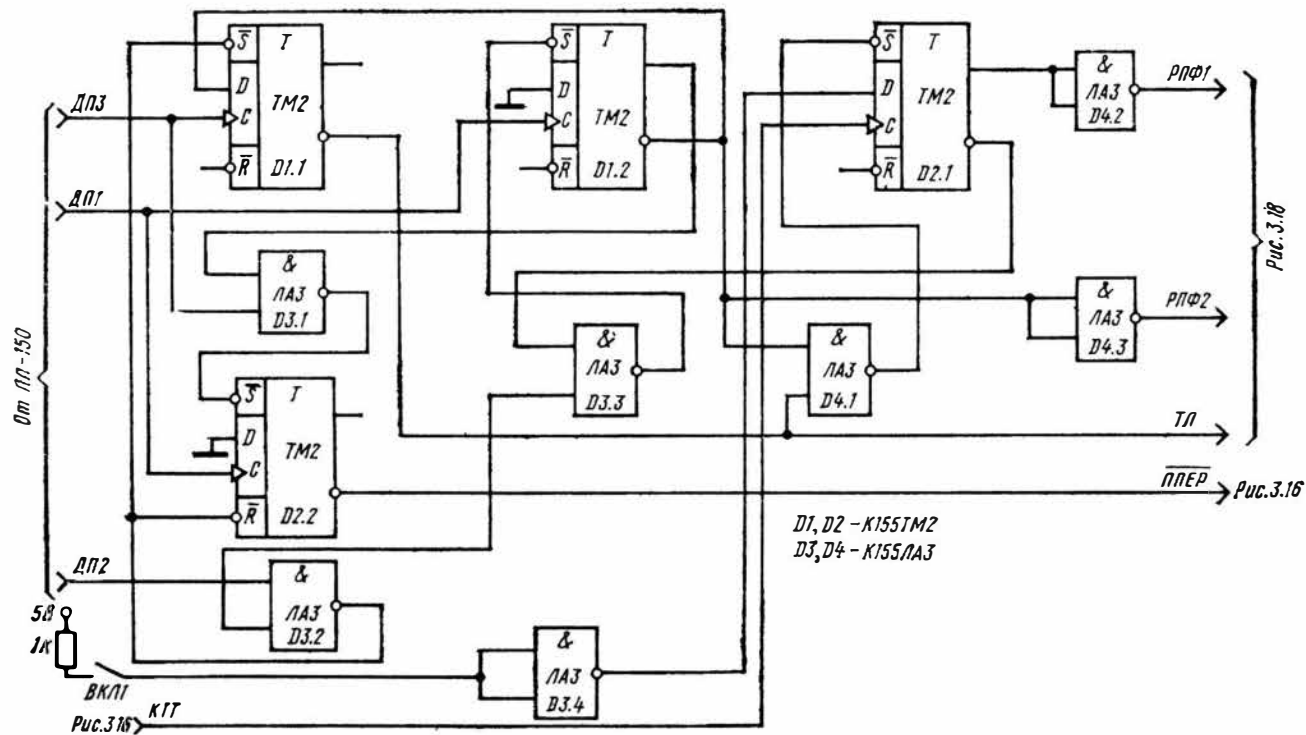


Рис. 3.19. Схема управления перфоратором ПЛ-150

вующий о готовности перфоратора к работе. При этом вход ВХДВ заземляется и двигатель включается. Через 1—2 с перфоратор готов к работе.

Перед выдачей данных на перфорацию микропроцессор программным способом устанавливает шестой разряд канала ШКНЗ БИС КР580ВВ55 (триггер готовности ТГТ) в единицу.

С переключателя ВК1 сигнал высокого уровня поступает также на элемент D3.4 (рис. 3.19) и после инвертирования на вход D триггера D2.1. При выдаче байта данных на перфорацию с разряда 7 канала ШКНЗ БИС КР580ВВ55 (см. рис. 3.16) поступает сигнал готовности КГТ, по которому триггер D2.1 устанавливается в нуль. При этом на токовые ключи КЛ1 [Ø—7] управления перфоратором (см. рис. 3.18) поступает сигнал разрешения перфорации РПФ1. С инверсного выхода триггера D2.1 сигнал высокого уровня поступает на вход элемента D3.3 и при поступлении с перфоратора импульса ДП2 устанавливает триггер D1.2 по входу S. С этого триггера снимается второй сигнал разрешения перфорации РПФ2 (рис. 3.20), при поступлении которого на входы ключей КЛ [Ø—7] и КЛС производится перфорация строки перфоленты. При поступлении с перфоратора импульса ДПЗ триггер D1.1 сбрасывается, а триггер D2.2 устанавливается. С инверсного выхода триггера D1.1 снимается сигнал транспортировки ленты (ТЛ), поступающий на токовый ключ КЛТ и перемещающий пер-

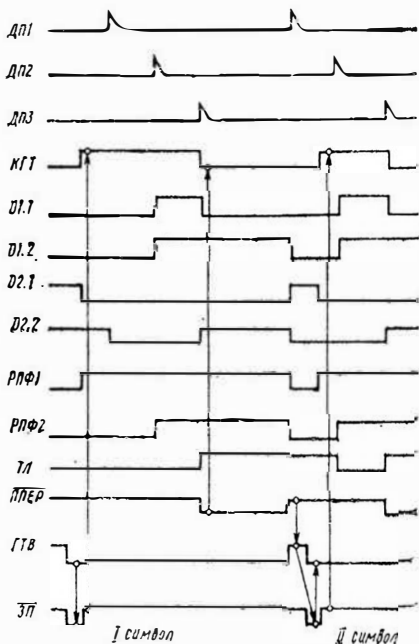


Рис. 3.20. Временные диаграммы работы схемы управления перфоратором ПЛ-150

фоленту на одну строку. С инверсного выхода триггера D2.2 снимается сигнал подтверждения передачи ППЕР, который поступает на 6-й разряд канала ШКНЗ БИС КР580ВВ55. Появление сигнала ППЕР сбрасывает сигнал КГТ (Канал ШКН1 готов). По окончании действия сигнала ППЕР сигнал ГТВ канала ШКН1 (разряд 3 канала ШКНЗ) устанавливается в единицу. Циклическим вводом байта информации канала ШКНЗ и анализом его 3-го разряда МП определяет момент появления сигнала ГТВ и записывает в канал ШКН1 БИС КР580ВВ55 следующий байт информации. При этом сигнал ГТВ вновь сбрасывается, а сигнал готовности канала КГТ поступает на схему управления перфоратором.

При вводе данных с фотосчитывателя FS-1501 (сигналы ФС [Ø—7]) в качестве сигнала СТР (см. рис. 3.16) используется сигнал с фотодиода синхродорожки. Выдаваемый по линии 1-го разряда канала ШКНЗ сигнал ППР, предназначенный для сброса сигнала СТР, в данном случае не используется, так как сигнал СТР после пробивки синхродорожки на перфоленте под фотодиодом автоматически сбрасывается. Для выдачи сигналов «Старт» (СТРТ) и «Стоп» (СТП) пуска и останова перфоленты используется 4-й разряд канала ШКНЗ, программируемый на вывод в режим Ø.

Лог. 1 на линиях устройства ввода FS-1501 соответствует напряжению —12 В, что делает необходимым согласование сигналов устройства FS-1501 с сигналами БИС КР580ВВ55, имеющими ТТЛ уровни. Это выполняется с помощью токовых ключей (рис. 3.21,а). При наличии пробивок на перфоленте импульсы с амплитудой —12 В поступают на линии данных ДР [Ø—7] и линию синхродорожки СДР токовых ключей. Транзисторы VT1 ключей открываются, и на выходе элементов К155ЛА3 появляются сигналы высокого уровня. Для преобразования ТТЛ уровней сигналов СТРТ и СТП в уровни —12 В используются токовые ключи, показанные на рис. 3.21,б.

Обращение к перфоратору и фотосчитывателю осуществляется следующим образом:

; программирование каналов

MVI A, ØA6H	; запись в накопитель управляющего слова
OUT ØFBN	; программирование канала ШКН1 на вывод,
	; канала ШКН2 на ввод в режиме I,

		; свободных разрядов канала ШКНЗ на вывод
MVI A, 09H		
OUT 0FBH		; установка в единицу четвертого разряда
		; канала ШКНЗ — выдача сигнала СТП
MVI A, 09H		
OUT 0FBH		; установка триггера ТГТ
		; канала ШКН2 (2-й разряд канала ШКНЗ)
MVI A, 0DH		
OUT 0FBH		; установка триггера ТГТ канала ШКН1
		; (6-й разряд канала ШКНЗ)
; считывание символа с перфоленты		
MVI A, 08H		
OUT 0FBH		; сброс в нуль 4-го разряда канала ШКНЗ —
		; выдача сигнала СТРТ
M1: IN 0FAH		; ввод байта канала ШКНЗ
ANI 01H		; выделение нулевого разряда — сигнала ГТВ
		; канала ШКН2
JZ M1		
MVI A, 09H		
OUT 0FBH		; установка в единицу 4-го разряда канала
		; ШКНЗ — выдача сигнала СТП
IN 0F9H		; ввод символа в накопитель с канала ШКН2
MOV C, A		; пересылка символа в регистр C
.		
.		
.		
; выдача символа на перфоратор		
M2: IN 0FAH		; ввод байта канала ШКНЗ
ANI 08H		; выделение 3-го разряда — сигнала ГТВ
		; канала ШКН1
JZ M2		
MOV A, C		; пересылка выводимого символа в накопитель
OUT 0F8H		; выдача символа через канал ШКН1

В качестве примера программирования приведенной схемы сопряжения с перфоратором рассмотрим программу блока начальной загрузки (см. § 3.2). Вводимая информация на перфоленте располагается следующим образом: заголовок 53H; начальный адрес — две строки; конечный адрес — две строки; вводимая информация побайтно на каждой строке.

Программа начальной загрузки загружает в регистр указатель стека SP код 48H. Затем осуществляется программирование БИС КР580ВВ55 на работу в режиме 1. После ввода заголовка, ввода в регистры H—L начального, а в регистры D—E конечного адресов в цикле (начиная с метки LOOP) осуществляются ввод и занесение в ОЗУ нужного числа байтов информации. После ввода каждого байта содержимое регистров H—L увеличивается на единицу. При равенстве текущего и конечного адресов ввод прекращается в счет-

чик команд с помощью команды RET (предпоследняя строка) заносится начальный адрес вводимой программы, который после ввода начального адреса был занесен в стек (команда PUSH H).

Ввод каждого символа осуществляется с помощью подпрограммы INP с начальным адресом 30H. Обращение к подпрограмме осуществляется командой RST 6 (Переход к адресу $6 \times 8 = 48$, т. е. 30H). Такое обращение требует меньше памяти, чем команда CALL, так как команда RST занимает один байт, а команда CALL — три.

Текст программы начальной загрузки приведен ниже:

```

; программа начальной загрузки
      LXI SP, 0048H      ; установка указателя стека на ОЗУ
      MVI A, 0A6H        ; программирование БИС
                          ; KP580BB55
      OUT 0FBH
      MVI A, 05H
      OUT 0FBH
      MVI A, 05H
      OUT 0FBH
; поиск заголовка
M1:   RST 6
      CPI 53H
      JNZ M1
; ввод начального адреса
      RST 6
      MOV H, A
      RST 6
      MOV L, A
      PUSH H              ; запись в стек начального адреса
; ввод конечного адреса
      RST 6
      MOV D, A
      RST 6
      MOV E, A
M2:   RST 6
      ORA A              ; установка признаков результата
      JNZ M2
; ввод очередного байта данных
LOOP: RST 6              ; ввод символа
      MOV M, A           ; посылка символа в ЗУ
; текущий адрес минус конечный
      MOV A, L           ; (H,L) — (D,E)
      SUB E
      MOV A, H
      SBB D
      JNC M3
      INX H
      JMP LOOP
; подпрограмма ввода байта

```

```

INP:      IN ØFAH
          ANI Ø1H
          JZ INP
          IN ØF9H
          RET
; остановка перфоленты
МЗ: MVI A, Ø9H
      OUT ØFBH
      RET
      HLT

```

3.4. Сопряжение микропроцессорной системы с бытовым магнитофоном

При записи на бытовой кассетный магнитофон двоичной информации используется частотная модуляция сигнала. Двоичной единице соответствует частота 2400 Гц, двоичному нулю 1200 Гц. Запись и считывание осуществляются в последовательном коле со скоростью 300 бит/с. Устройство сопряжения микро-ЭВМ с бытовым магнитофоном подключается к последнему стандартным кабелем, используемым обычно для подачи и съема сигналов звуковой частоты. Амплитуда сигналов, поступающих на вход «Звукосниматель» магнитофона и снимаемых с его линейного выхода, — не менее 250 мВ.

Коммутатором направления передачи информации служит элемент D5 типа K589АП16 (рис. 3.22). Его вывод DВ1 подключен к разряду 7 канала ШКН1 БИС КР580ВВ55. Через разряд ШКН1 [7] осуществляются как выдача, так и прием информации в последовательном коде. Для этого канал ШКН1 БИС КР580ВВ55 программируется на ввод или вывод информации в режиме Ø. Сигнал низкого уровня на выходе дешифратора D3 появляется при наличии на шине адреса определенных сигналов (в данном случае 3ØН—31Н) и появлении одного из сигналов 3ПВВ и ЧТВВ. Сигнал низкого уровня с выхода дешифратора адреса, поступаая на вход ВМ D5, разрешает работу коммутатора. Этот же сигнал после синхронизации срезом тактового импульса С2ТТЛ на D2.3 и инвертирования на D2.2 поступает на вход С триггера D4.1 и вызывает запись в него информации, поступающей на вход D. Если канал ШКН1 БИС КР580ВВ55 предварительно программирован на вывод информации в режиме Ø, то при наличии сигнала 3ПВВ, поступающего на вход ВШ (Выбор шины) элемента D5, информация из микропроцессорной системы через разряд ШКН1 [7] КР580ВВ55 и коммутатор направления передачи K589АП16 поступает на вход D триггера D4.1. Триггер D4.1 управляет работой симметричного мультивибратора на транзисторах VT1, VT2. При единичном уровне сигнала на выходе триггера мультивибратор генерирует импульсы с частотой 2400 Гц, а при нулевом — частотой 1200 Гц. Эти импульсы через эмиттерный повторитель на транзисторе VT3 поступают на вход магнитофона. С помощью потенциометра P1 рекомендуется установить значение сигналов около 1 В. С инверсного выхода D4.1 сигналы через инвертор поступают на светодиод индикатора записи. При подаче сигнала сброса СБР по шине управления микропроцессорной системы триггер D4.1 устанавливается в единицу по входу S.

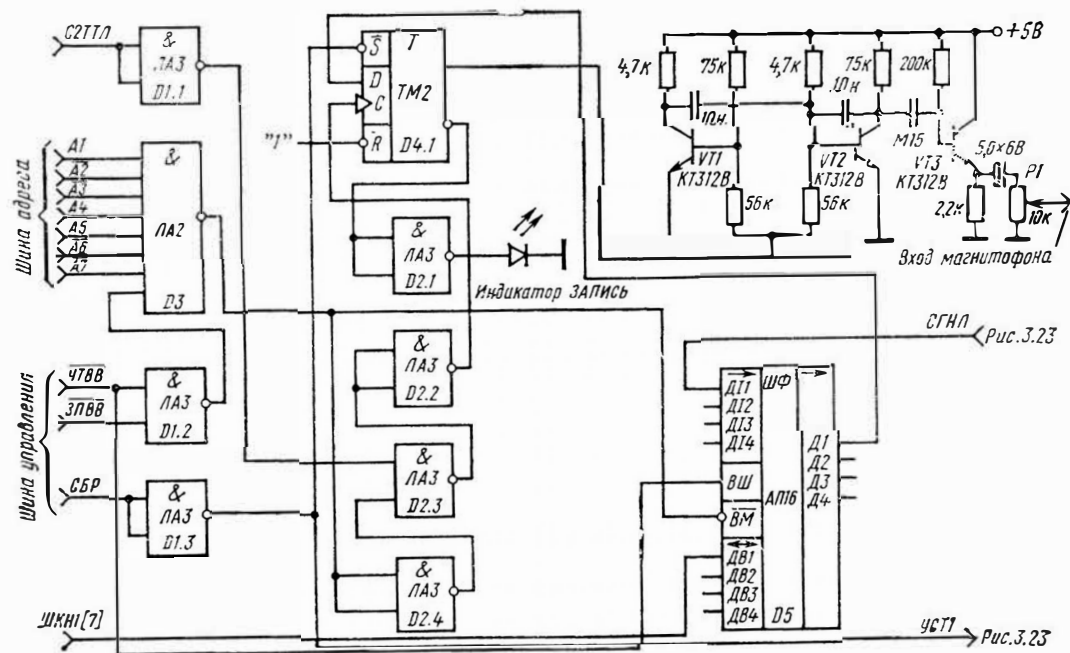


Рис. 3.22. Принципиальная схема преобразователя напряжение — частота

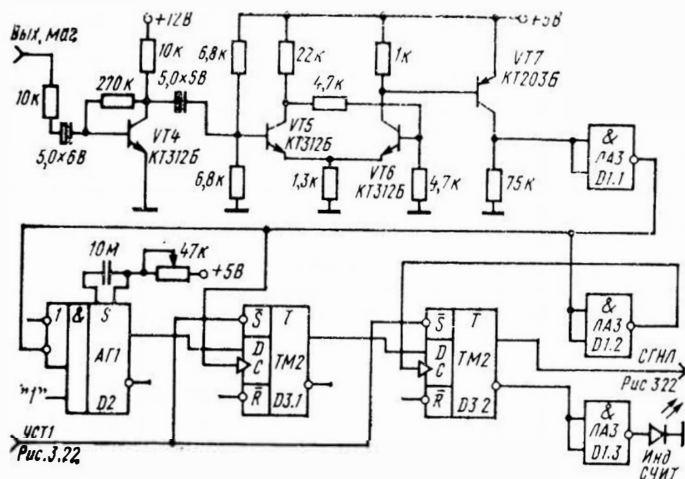


Рис. 3.23. Принципиальная схема преобразователя частота — напряжение

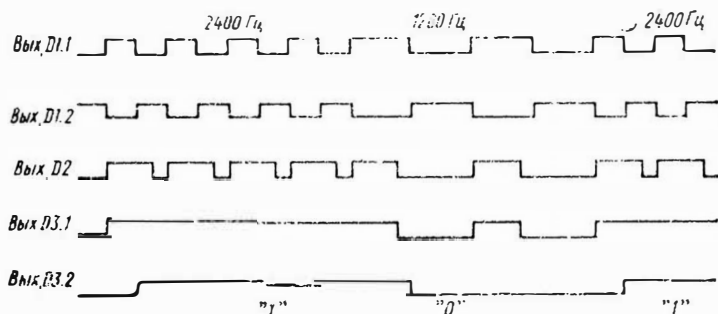


Рис. 3.24. Временные диаграммы преобразователя частота — напряжение

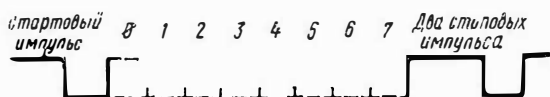


Рис. 3.25. Представление байта информации для записи на магнитофон

Для считывания данных с магнитофона необходимо предварительно осуществить программирование канала ШКН1 БИС КР580ВВ55 на ввод в режиме $\emptyset$. При наличии на шине управления сигнала $\overline{3ПВВ}$ сигнал $\overline{ЧТВВ}$ имеет высокий уровень и коммутатор направления передачи D5 передает данные со входа DI1 на вывод DB1 и далее на ШКН1 [7] БИС КР580ВВ55. Информация на вход DI1 элемента D5 поступает с преобразователя частота — напряжение (рис. 3.23).

При считывании данных с магнитофона сигнал с линейного выхода магнитофона поступает на вход усилительного каскада на транзисторе V14 и далее на триггер Шмидта (VT5, VT6, VT7), обеспечивающего восстановление формы прямоугольных импульсов. Далее сигналы через инвертор D1.1 поступают на одновибратор D2 и синхровход триггера D3.1 (рис. 3.24). Одновибратор при запуске отрицательным фронтом поступающих импульсов вырабатывает импульс длительностью, большей $1/2400$ и меньшей $1/1200$ с. Подстройка длительности импульсов одновибратора осуществляется потенциометром P2. Информация с выхода триггера D3.2 поступает на коммутатор направления передачи (элемент D5 рис. 3.22).

При обмене информацией с магнитофоном информация передается в последовательном коде (рис. 3.25). Фактически выдается только 7-й разряд накопителя. Для выдачи байта информации, предварительно помещенной в накопитель, необходимо программным способом осуществлять сдвиг последовательных битов в 7-й разряд. После выдачи каждого бита информации с помощью подпрограммы DEL осуществляется задержка (определяется константой TDEL), обеспечивающая запись информации со скоростью 300 бит/с:

```

; подпрограмма выдачи байта из накопителя на магнитофон
; (адрес 30H)
BWRT:  PUSH B
        PUSH PSW          ; запоминание вызываемого байта в
                           ; стеке
        XRA A              ; загрузка нуля в накопитель
        OUT 30H            ; выдача стартового бита
        CALL DEL
        POP PSW           ; восстановление выдаваемого байта
                           ; в накопителе
        MVI B,8            ; загрузка счетчика цикла
LOOP:   OUT 30H            ; выдача старшего бита накопителя
        CALL DEL
        DCR B              ; уменьшение содержимого счетчика
                           ; цикла
        JZ BEND            ; выход из цикла
        RAL                ; сдвиг содержимого накопителя влево
        JMP LOOP
BEND:   MVI A,0FFH
        OUT 30H            ; выдача первого стопового бита
        CALL DEL
        OUT 30H            ; выдача второго стопового бита
        POP PSW
        POP B
        RET
; подпрограмма задержки, задающей скорость записи 300 бит/с
DEL:    PUSH PSW
        PUSH D

```

MI:	LXI D, TDEL DCX D MOV, A, D ORA E JNZ MI POP D POP PSW RET	; загрузка в D, E числа циклов ; уменьшение содержимого D, E
-----	---	---

Запись информации на ленту осуществляется зонами по 128 байт. Перед началом зоны записываются символ S и длина зоны (если количество выводимой информации не кратно длине зоны, то последняя зона будет содержать менее 128 байт), в конце зоны записывается контрольная сумма. Перед началом записи информации на ленту записывается заголовок, включающий: символ S — знак начала записи, начальный адрес выводимой области памяти, синхронизирующую задержку 1; конечный адрес выводимой области памяти; синхронизирующую задержку 2; 4 байта $\overline{O}FFH$ и один байт $\overline{O}H$. При считывании информации необходимо по счетчику ленты установить нужный участок ленты около головки и начинать считывание.

Глава 4.

Аппаратно-программные комплексы для отладки микропроцессорных систем

4.1. Классификация отладочных комплексов

Для проектирования микропроцессорных систем широко используются специальные аппаратно-программные отладочные комплексы, позволяющие проводить совместную разработку и отладку программного обеспечения и технических средств. Все аппаратно-программные комплексы можно разделить на четыре типа [11]: оценочные модули; программно-аппаратные отладочные комплексы ограниченной мощности; развитые программно-аппаратные и универсальные аппаратно-программные отладочные комплексы.

Оценочные модули. Представляют собой одноплатные микро-ЭВМ с минимальной конфигурацией технических средств. Устройствами ввода — вывода являются клавиатура, светодиоды, семисегментные индикаторы и в ряде случаев телетайп. При использовании оценочных модулей программирование осуществляется в машинных командах, что эффективно при длине программ до 256 байт. Применяя технику программирования в машинных командах, можно вносить изменения и в более сложные программы. Программное обеспечение, как правило, ограничивается программой-монитором, размещенной в ПЗУ. Однако, несмотря на небольшой объем (256 байт — 4К байт), такой монитор предоставляет достаточно гибкие возможности для отладки программ, а именно: пошаговый режим, задание адресов контрольных точек останова, вывод на индикацию и изменение с клавиатуры содержимого регистров и ячеек ОЗУ, работу с командными клавишами и телетайпом.

Оценочные модули предназначаются для оценки возможностей микропроцессорных средств, а также для разработки прототипов сложных микропроцессорных систем с объемом программного обеспечения до 1К байта. Их преимуществом является низкая стоимость. Недостатками оценочных модулей как систем проектирования являются невозможность отладки технических средств микропроцессорных систем, ограниченные возможности устройств ввода — вывода, необходимость программирования в машинных командах и высокая трудоемкость разработки программ большого объема.

Аппаратно-программные комплексы ограниченной мощности. Конструктивно выполняются в виде законченного блока, объединяющего несколько печатных плат и имеющего пульт с клавиатурой управления и индикацией. Программирование ведется на ассемблере, генерирующем абсолютный либо перемещаемый объектный код. В ряде случаев в состав системного программного обеспечения включается интерпретатор программ на языке Бейсик. В ПЗУ таких комплексов хранятся только программа-монитор и программа-загрузчик. Ассемблер и редактор текста хранятся на перфоленте или магнитной ленте и при необходимости поочередно загружаются в ОЗУ.

Недостатком аппаратно-программных отладочных комплексов рассматриваемого класса являются ограниченные возможности отладки технических средств микропроцессорных систем.

Развитые аппаратно-программные отладочные комплексы. Принципиальными особенностями этих комплексов являются: возможность внутрисхемной эмуляции отлаживаемых программ; широкий набор внешних устройств, включая накопители на гибких магнитных дисках (НГМД); наличие развитой дисковой операционной системы (ДОС), компиляторов с языков высокого уровня.

Комплексы рассматриваемого класса поставляются как набор совместимых модулей, состав которого пользователь выбирает исходя из конкретных требований и стоимости. Такие комплексы обычно предназначены для разработки систем на базе не одного МП, а семейства различных микропроцессорных средств.

Высокая эффективность развитых аппаратно-программных отладочных комплексов во многом обусловлена наличием НГМД, что существенно увеличивает емкость ЗУ. Это, во-первых, позволяет использовать трансляторы с языков высокого уровня (упрощенные варианты языка PL/I, Фортрана, Кобола, Паскаля и др.); во-вторых, дает возможность создания библиотек отлаживаемых программ и стандартных подпрограмм. Трансляторы с языка ассемблера также широко используются в отладочных комплексах этого класса. Они, как правило, генерируют перемещаемый код, что позволяет отлаживать программы большого объема по частям. Оперативное запоминающее устройство таких комплексов имеет объем от 16К до 64К байт. Объем ПЗУ (ППЗУ) достигает 8К байт. Средства внутрисхемной эмуляции позволяют одновременно и на реальном макете осуществлять отладку программных и технических средств разрабатываемой микропроцессорной системы, что значительно повышает эффективность процесса разработки.

Универсальные аппаратно-программные отладочные комплексы. Для них характерны: возможность разработки систем на основе различных 8- и 16-разрядных микропроцессорных средств; наличие в комплексах нескольких терминалов; использование дорогих внешних устройств, обеспечивающих максимальную производительность, в частности ЗУ на жестких магнитных дисках.

Характеристики аппаратно-программных отладочных комплексов различных типов [13]

Характеристики комплексов	Оценочные модули	Отладочные комплексы ограниченной мощности	Развитые отладочные комплексы	Универсальные отладочные комплексы
Число моделей микропроцессорных средств	Одна	Одна	Несколько сходных моделей	Различные 8- и 16-рядные модели
Емкость ППЗУ или ПЗУ, байт	256—2К	1К—4К	2К—8К	2К—16К
Емкость ОЗУ, байт	128—4К	4К—16К	16К—64К	64К—128К
Типовые устройства ввода—вывода	Индикация на светодиодах или сегментных индикаторах, клавишная панель	Телетайп, печатающие устройства, перфоленточные устройства ввода—вывода, программаторы ППЗУ	Дисплей на ЭЛТ, печатающие устройства, перфоленточные устройства ввода—вывода, программаторы ППЗУ	Дисплей на ЭЛТ, высокоскоростные печатающие устройства, перфоленточные устройства ввода—вывода, программаторы ППЗУ
Внешнее ЗУ	Нет	Перфолента, кассетное ленточное ЗУ	НГМД, кассетное ленточное ЗУ, перфолента	ЗУ на жестком магнитном диске, НГМД, перфолента

Продолжение табл. 4.1

Характеристики комплексов	Оценочные модули	Отладочные комплексы ограниченной мощности	Развитые отладочные комплексы	Универсальные отладочные комплексы
Способ программирования	В машинных командах	На языке ассемблера	На языках высокого уровня, на языке ассемблера	На языках высокого уровня, на языке ассемблера
Системное программное обеспечение	Отладочный монитор	Отладочный монитор, абсолютный ассемблер, редактор текста	ДОС; отладочный монитор; ассемблер, генерирующий перемещаемый код; редактор связей; редактор текста; компиляторы с языков высокого уровня	ДОС; отладочный монитор; ассемблеры, генерирующие перемещаемый код; редакторы связей; редактор текста; компиляторы с языков высокого уровня
Внутрисхемная эмуляция	Нет	Нет	Да	Да
Многотерминальная работа	Нет	Нет	Нет	Да
Стоимость, дол.	100—500	500—5000	5000—15 000	15 000—80 000

Такие комплексы предоставляют пользователю все возможности развитых аппаратно-программных отладочных комплексов. Блоки внутрисхемных эмуляторов и часть системного программного обеспечения (ассемблеры, компиляторы) являются сменными и предназначены для работы с конкретными МП. Разработка программного обеспечения большого объема на таких комплексах обычно производится одновременно несколькими программистами.

Сравнительный анализ аппаратно-программных отладочных комплексов. При разработке систем на основе МПК БИС, имеющих нестандартную структуру технических средств, необходимо применение отладочных комплексов с блоком внутрисхемной эмуляции. Для разработки только программного обеспечения стандартных микро-ЭВМ пригодны все типы отладочных комплексов (табл. 4.1).

Оценочные модули, как правило, приобретаются для начального изучения микропроцессорных средств и характеризуются минимальной стоимостью. При нерегулярной разработке программного обеспечения небольшого объема большие затраты на приобретение сложных отладочных средств являются нецелесообразными. В этом случае обычно используются отладочные комплексы ограниченной мощности. Если разработка аппаратуры на основе конкретной модели (или моделей) микропроцессорных средств ведется постоянно, то затраты на приобретение и освоение развитых аппаратно-программных отладочных комплексов, как правило, оправданы. Универсальные отладочные комплексы обеспечивают наивысшую производительность, но их разработка или приобретение существенно дороже других комплексов. Универсальные комплексы целесообразно применять при проектировании систем на основе различных микропроцессорных средств в массовом масштабе, особенно если их программное обеспечение имеет большой объем и разрабатывается одновременно несколькими программистами.

4.2. Аппаратно-программный отладочный комплекс Intellec MDS-800

Развитый аппаратно-программный отладочный комплекс Intellec MDS-800 предназначен для разработки и отладки как программного обеспечения, так и технических средств микропроцессорных систем на основе МП 8080, 8085, 3000 фирмы Intel. Он может быть использован также для разработки систем на основе отечественных комплексов БИС серий КР580 и К589.

Комплекс представляет собой специализированную микро-ЭВМ на базе МП 8080 фирмы Intel (рис. 4.1) и включает модуль центрального процессора, модуль ОЗУ емкостью 16К байт, модуль ПЗУ с занесенным в него монитором, и панель управления. Для расширения возможностей комплекса к нему могут быть добавлены модули ППЗУ, до трех модулей ОЗУ емкостью по 16К байт, НГМД, модули внутрисхемных эмуляторов ICE-80 и ICE-30. В состав аппаратного обеспечения комплекса входит набор стандартных внешних устройств: телетайп, дисплей, устройство ввода с перфоленты, ленточный перфоратор, печатающее устройство, программатор ПЗУ.

¹ Краткий контекстный перевод английских терминов, используемых при описании отладочного комплекса и его операционной системы, приведен в приложении 2.

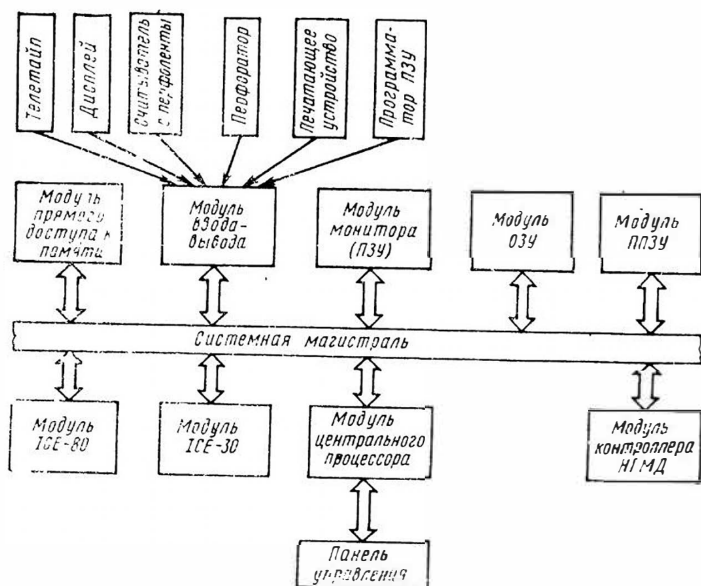


Рис. 4.1. Структурная организация отладочного комплекса Intellex MDS-800

Основным устройством, с помощью которого осуществляется взаимодействие пользователя с комплексом, является пультный терминал, в качестве которого может использоваться алфавитно-цифровой дисплей или телетайп.

Как правило, для отладки микропроцессорных систем применяется полный комплект комплекса Intellex MDS-800 с объемом ОЗУ 64К байта, ИГМД, модулем внутрисхемного эмулятора. В качестве пультного терминала используется алфавитно-цифровой дисплей.

Комплекс MDS-800 включает средства для разработки как программного обеспечения, так и технических средств микропроцессорных систем. Для разработки и отладки программного обеспечения служат программа-монитор и программы ДОС. Для комплексной отладки программных и технических средств используются модуль внутрисхемной эмуляции и соответствующее программное обеспечение, также входящее в состав операционной системы. Программное обеспечение комплекса совместно со средствами внутрисхемной эмуляции позволяют осуществлять запись программ (текстовых файлов) на том или ином входном языке, их редактирование, компиляцию, хранение исходных и объектных программ, осуществлять выполнение и отладку программ, а также отладку технических средств.

Программа-монитор постоянно расположена в ПЗУ и позволяет выполнять ряд функций, обеспечивающих взаимодействие с пультным терминалом, возможность загрузки и исполнения объектных программ, а также их контроль и отладку.

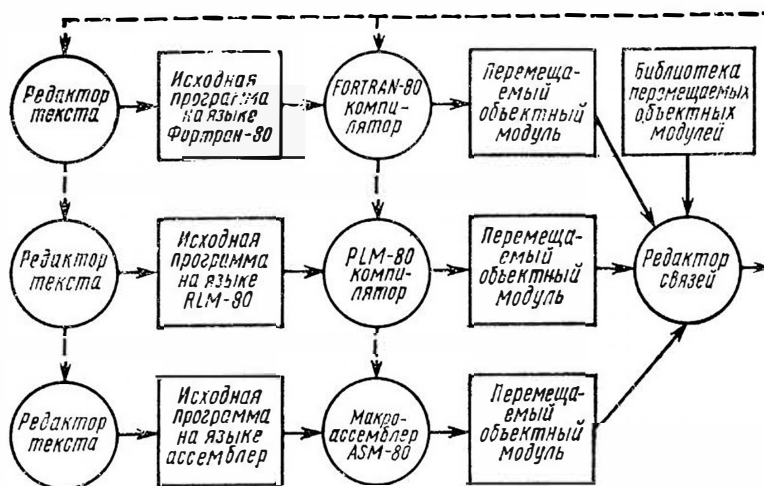


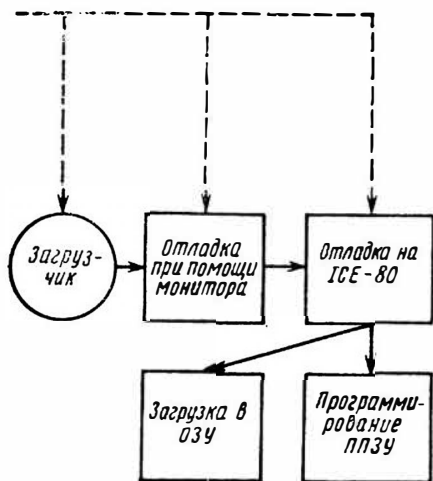
Рис. 4.2. Последовательность разработки программных и технических средств микропроцессорной системы с использованием комплекса Intellex MDS-800

Основу системного программного обеспечения комплекса составляет ДОС ISIS-II, которая позволяет работать с большими объемами информации, записываемой на ИГМД, и представляет собой набор системных программ, необходимых для разработки программного обеспечения и помогающих разработчику управлять ресурсами комплекса при разработке, отладке и выполнении целевых программ.

В MDS-800 применяется принцип модульного программирования. Модульное программирование предполагает написание программы в виде достаточно простых модулей (файлов), которые впоследствии используются основной программой пользователя. Мнемонические имена файлов определяет пользователь, а ДОС обеспечивает работу пользователя с файлами на мнемоническом уровне.

Основным инструментом для создания и редактирования текстовых файлов в коде КОИ-7 является программа-редактор текста. Процедура редактирования заключается в запуске программы редактора, введении текста, его редактировании (замене, удалении или добавлении некоторых символов) и выводе текстового файла на печать или записи его на диск.

В состав ДОС входит широкий набор команд управления файлами. Эти команды позволяют копировать, удалять файлы, осуществлять преобразование кодов, просматривать оглавление диска. Программные модули могут быть написаны на различных языках. Каждый программный модуль может быть записан, отредактирован и отложен независимо от других. Преобразование исходных текстовых файлов в объектные программы осуществляется компиляторами, входящими в состав ДОС. В операционную систему включены программа-макроассемблер, компиляторы с языков высокого уровня



(PL/M-80. Паскаль, Фортран, язык С). Кроме того, имеется интерпретатор программ на языке Бейсик.

Полученные в результате компиляции файлы объектного кода имеют перемещаемый формат. Для компоновки программ из перемещаемых объектных модулей и размещения программы в ЗУ в операционной системе имеются три системные программы: LINK (РЕДАКТОР СВЯЗЕЙ), LOCATE (ЗАГРУЗЧИК) и LIB (БИБЛИОТЕКАРЬ).

Программа-библиотекарь обеспечивает введение, удаление и хранение файлов объектных модулей. Эти модули могут быть использо-

ваны для построения более сложных программ. Для этого требуемые модули должны быть объединены в общую программу редактором связей, который объединяет объектные модули путем согласования относительных адресов межмодульных ссылок. Программа-загрузчик позволяет разместить полученную в перемещаемом формате объектную программу с учетом реального распределения адресов ОЗУ и ПЗУ, имеющихся в разрабатываемой микропроцессорной системе, что осуществляется введением соответствующих управляющих параметров в программу. Программа-загрузчик присваивает абсолютные адреса перемещаемым объектным файлам, после чего полученные абсолютные объектные программы готовы для исполнения. Исполнение любой программы можно инициализировать, указав имя соответствующего файла. Директива DEBUG (Отладка) дает возможность пользователю осуществлять загрузку и исполнение программ под управлением монитора.

Разработка программного обеспечения и технических средств при проектировании микропроцессорной системы должна осуществляться пользователем (группой пользователей) параллельно. Результатом этой работы является прикладная объектная программа, логическое функционирование которой проверено на системе MDS-800, а также микропроцессорная система-прототип. Система-прототип представляет собой микропроцессорную систему на ранних стадиях разработки, в которую из всех возможных блоков должны входить, по крайней мере, центральный процессорный элемент и схемы сопряжения с шинами управления, адресов и данных. Дальнейшая отладка как программного обеспечения, так и технических средств на реальном макете может быть проведена с помощью внутрисхемного эмулятора ICE-80 без использования какого-либо дополнительного отладочного оборудования.

Внутрисхемный эмулятор ICE-80 включается вместо МП системы-прототипа и выполняет все его функции — осуществляет эмуляцию. В то же время специальные программно-аппаратные средства,

заложенные в блок внутрисхемной эмуляции, дают возможность пользователю активно контролировать ход вычислительного процесса в системе-прототипе. Внутрисхемный эмулятор позволяет осуществлять исполнение, контроль и отладку программ в системе-прототипе в диалоговом режиме с терминала отладочного комплекса. При этом возможна отладка в реальном масштабе времени и в реальных условиях работы микропроцессорной системы.

Одним из важнейших качеств внутрисхемного эмулятора является возможность эмуляции им не только функций центрального процессорного элемента, но и функций ОЗУ (ППЗУ, ПЗУ) и периферийных устройств системы-прототипа. Для того, чтобы некоторая часть или вся память системы прототипа, а также часть или все внешние устройства системы-прототипа были эмулированы памятью и внешними устройствами комплекса, внутрисхемным эмулятором используется механизм виртуальной адресации. Сущность этого механизма заключается в том, что перед началом эмуляции программой ICE-80 пользователю дается возможность установить соответствие между виртуальными (логическими) адресами блоков памяти и внешних устройств, используемых в его программах, и их физическими адресами в системе-прототипе или отладочном комплексе.

Внутрисхемный эмулятор позволяет предоставить в распоряжение разрабатываемой системы те аппаратные ресурсы, которые на этапе разработки в ней еще отсутствуют, но которыми обладает отладочный комплекс (память, внешние устройства). При этом система-прототип может быть подключена к управляемому объекту.

Директивы внутрисхемного эмулятора позволяют производить в реальном масштабе времени при подключенном к системе-прототипу управляемому объекту все отладочные операции, обычно выполняемые монитором: инициализировать исполнение программ с указанием точек разрыва (останова), просматривать и редактировать содержимое ячеек ЗУ и регистров МП и т. п.

Таким образом, аппаратно-программный отладочный комплекс Intellec MDS-800 позволяет одновременно и на реальном макете осуществлять отладку как программного обеспечения, так и технических средств проектируемой микропроцессорной системы. Общая последовательность разработки и отладки программных и аппаратных средств микропроцессорной системы с использованием комплекса Intellec приведена на рис. 4.2.

4.3. Отладка объектных программ с использованием программы Монитор

Программа-монитор реализует наиболее важные функции ДОС — взаимодействие с пультовым терминалом, запуск, отладку и исполнение объектных программ. Программа-монитор расположена в ПЗУ, поэтому сразу после включения питания отладочный комплекс способен выполнять ряд мониторинговых функций. При этом монитор обеспечивает вывод точки (·) на устройстве пультового вывода (пультовый терминал) для указания готовности к приему директив.

Для работы программы-монитора в ОЗУ комплекса резервируются адреса $\emptyset\emptyset H - \emptyset FH$, а также 318 старших адресов первого непрерывного блока ОЗУ. Сам монитор занимает адресное пространство $\emptyset F8\emptyset\emptyset H - \emptyset FFFFH$.

Директивы имеют свое mnemonic обозначение и вводятся с клавиатуры дисплея. Выполнение директивы осуществляется после ее ввода и нажатия клавиши RETURN (Возврат каретки). Информацию, введенную до ее нажатия, пользователь может изменить с помощью специальных команд (клавиш) строчного редактирования. Так, удалить предыдущий символ можно, нажав клавишу RUBOUT (Стирание). Последовательное нажатие клавиш CTRL и X (CTRL-X) удаляет всю предыдущую запись. Вывод информации на экран можно остановить командой CTRL-S. Возобновление вывода осуществляется командой CTRL-Q.

Для нормальной работы монитора необходимы как минимум четыре внешних устройства, выполняющие

Таблица 4.2

Обозначения логических устройств

Устройство	Обозначение
Пультный терминал (CONSOLE)	C
Устройство ввода с перфоленты (READER)	R
Перфоратор (PUNCH)	P
Печатающее устройство (LIST)	L

Таблица 4.3

Обозначения физических устройств

Устройство	Обозначение
Телетайп	T или TTY
Дисплей	C или CRT
Печатающее устройство	L или LPT
Перфоратор	P или PTP
Устройство ввода с перфоленты	R или PTR
Устройство, определяемое пользователем	1 или 2
Считывание директив из файла директив с одновременным их выводом на печатающее устройство (пакетный режим)	B или BATCH

Отладочные директивы монитора

Директива	Описание
D<адрес 1>, <адрес 2>,	Вывод на печатающее устройство содержимого области ЗУ, начиная с адреса 1 до адреса 2 включительно, в шестнадцатеричной системе счисления
.F<адрес 1>, <адрес 2>, <константа>	Заполнение области ЗУ, начиная с адреса 1 до адреса 2 включительно константой, заданной в шестнадцатеричной системе счисления
.M<адрес 1>, <адрес 2>, <адрес 3>	Перемещение содержимого области ЗУ, начиная с адреса 1 до адреса 2 включительно, в область ЗУ, начиная с адреса 3
.S<адрес>	Вывод на пультовый терминал содержимого ячейки ЗУ с указанным адресом в шестнадцатеричной системе счисления и изменение содержимого этой ячейки
.X<регистр>	Вывод на пультовый терминал содержимого указанного регистра (табл. 4.5) в шестнадцатеричной системе счисления и модификация его содержимого. При отсутствии в директиве идентификатора регистра на терминал выводится содержимое всех регистров в порядке, указанном в табл. 4.5
.G<адрес 1>, <адрес 2>, <адрес 3>	Запуск программы, начиная с адреса 1, и останов ее выполнения при достижении адреса 2 или адреса 3. Если адрес 1 не указан, то выполнение программы начинается с адреса, находящегося в счетчике команд
.H<число 1>, <число 2>	Вычисление и вывод на пультовый терминал суммы и разности двух шестнадцатеричных чисел

функции пультового терминала (CONSOLE), устройства ввода с перфоленты (READER), перфоратора (PUNCH) и печатающего устройства (LIST). Монитором предусмотрена возможность назначения различных физических устройств для выполнения перечисленных выше системных функций. Для этого служит директива А (НАЗНАЧИТЬ), имеющая вид (табл. 4.2, 4.3).

.A < логическое устройство > = < физическое устройство >.

Процедура назначения системных устройств ввода — вывода может выглядеть, например, следующим образом:

.AC = T
.AR = P
.AP = T
.AL = T

Если назначение устройства не производится, то (по умолчанию) в качестве пультавого терминала назначается дисплей, а в качестве остальных устройств — телетайп. Ввод команды Q обеспечивает отображение на выбранном терминале списка логических и физических устройств ввода — вывода. Например,

.Q
C = C
R = C
P = T
L = T

Отладочные директивы монитора позволяют (табл. 4.4) выводить и модифицировать содержимое регист-

Т а б л и ц а 4.5

Идентификаторы регистров микропроцессора

Регистр	Идентификатор регистра	Регистр	Идентификатор регистра
Регистр А	А	Регистр Н	Н
Регистр В	В	Триггер разрешения прерывания	І
Регистр С	С	Регистр L	L
Регистр D	D	Регистровая пара Н—L	M
Регистр E	E	Счетчик команд	P
Регистр признаков результата	F	Указатель стека	S

ров МП и ячеек ЗУ, заполнять область ЗУ константой, перемещать содержимое областей ЗУ, запускать программу на исполнение с заданием адресов останова. При этом адреса задаются в шестнадцатеричной системе отчисления.

При выводе содержимого области памяти в каждой строке выдаваемой информации размещается содержимое до 16 ячеек ЗУ. В начале строки указывается адрес первой ячейки, содержимое которой размещается в данной строке. Например, по директиве .D 8000, 802F может быть выведена следующая информация:

```
8000 3E 2B 06 14 21 13 80 BE C2 0D 80 36 00 23 05 C2
8010 07 80 76 14 05 FF FF FF FF FF FF FF FF FF FF
8020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Если после этого задать директиву заполнения области ЗУ константой .F 800A, 8019, 54 и затем ввести директиву .D 8000, 802F, то на экран будет выведено

```
8000 3F 2B 06 14 21 13 80 BE C2 0D 54 54 54 54 54 54
8010 54 54 54 54 54 54 54 54 54 54 54 54 54 54 54
8020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

После введения директивы изменения содержимого ячейки ЗУ .S<адрес> вместо символа «Возврат каретки» вводится пробел. На терминал выводятся содержимое ячейки с указанным адресом и дефис. После этого пользователь с терминала может ввести новое содержимое ячейки. При необходимости сохранить содержимое ячейки ЗУ вместо нового ее содержимого вводится пробел. При этом на терминал выводится содержимое следующей ячейки памяти, сопровождаемое дефисом, предоставляя пользователю возможность для его изменения. Для завершения директивы необходимо ввести символ «Возврат каретки». Например,

```
.S8015 54—82
.S8016 54—54—54—14
.D8015,802F
8015 82 54 54 14 54 FF FF FF FF FF FF
8020 FF FF FF FF FF FF FF FF FF FF FF
```

Здесь символом указано нажатие клавиши «Пробел». Информация, выводимая на терминал монитором при выполнении директивы S, подчеркнута.

Работа с директивой изменения содержимого регистров .X<адрес> производится аналогично. Например,

.X $\square$ 05—13—2E—00—E8—56—F7—FF—E8—F7E8—FF92—
F7BC—
.XA $\square$ 05—03
.X $\square$ 03—13—2E—00—E8—56—F7—FF—E8—F7E8—
FF92—F7BC—
.XH $\square$ F7—
.XL $\square$ E8—
.XS $\square$ F7BC—
.XP $\square$ FF92—

Директива .N<число 1>, <число 2>, вычисляющая сумму и разность двух чисел, позволяет использовать вычислительные возможности монитора. Например,

.N10,10
0020,0000
.HIF,2A
0049,FFF5

Директива G позволяет запускать программу и останавливать ее по указанному адресу с передачей

Таблица 4.6

**Директивы работы с перфоленточными устройствами
ввода—вывода**

Директива	Описание
.B<адрес 1>, <адрес 2>	Вывод содержимого области ЗУ, начиная с адреса 1 по адрес 2 включительно, на перфсленту в двоичном формате
.L<адрес 1>, <адрес 2>	Ввод информации с перфоленты в область ОЗУ, начиная с адреса 1 по адрес 2 включительно, в двоичном формате
.W<адрес 1>, <адрес 2>	Вывод содержимого области ЗУ, начиная с адреса 1 по адрес 2 включительно, на перфоленту в шестнадцатеричном формате
.R<смещение>	Ввод информации с перфоленты в область ОЗУ в шестнадцатеричном формате
.E<адрес>	Конец шестнадцатеричного файла
.N	Перфорация 60 нулевых символов (отсутствие пробивок на перфоленте)

управления монитору. При этом пользователь может проанализировать и изменить содержимое ячеек ЗУ и регистров МП, а затем продолжить выполнение программы с помощью директивы G.

Монитор обеспечивает также работу с перфоленточными устройствами ввода — вывода (табл. 4.6). Если размеры загружаемого файла превышают размеры указанной в директиве области ЗУ, часть файла теряется. При вводе шестнадцатеричного файла по команде R загрузка осуществляется в область ЗУ, начиная с адреса, указанного в самом файле.

Если в директиве R указано ненулевое смещение, то монитор загружает шестнадцатеричный файл в область памяти, смещенную относительно адресов загрузки, указанных в самом файле. Естественно, что загруженная таким образом программа непосредственно выполняться не может.

При выводе информации на перфоленту в шестнадцатеричном формате возможно многократное использование директивы W, однако последний блок данных, выведенный на перфорацию, должен заканчиваться символом «Конец файла», формируемым директивой E. Пусковой адрес, указываемый в этой команде, заносится в шестнадцатеричный файл на перфоленте. При выполнении директивы ввода R этот адрес заносится в счетчик команд, что дает возможность начать выполнение программы, содержащейся во вводимом файле, сразу с нужного адреса. При нулевом значении пу-

Таблица 4.7

Директивы монитора для работы с программатором ППЗУ

Директива	Описание
.R<код> <разъем> <адрес 1>, <адрес 2>, <адрес ППЗУ>	Запись информации из области ЗУ начиная с адреса 1 по адрес 2 включительно, в БИС ППЗУ, начиная с адреса ППЗУ
.C<код> <разъем> <адрес 1>, <адрес 2>	Сравнение содержимого БИС ППЗУ с содержимым области памяти, начиная с адреса 1 по адрес 2 включительно
.T<код> <разъем> <адрес 1>, <адрес 2>	Перезапись информации из БИС ППЗУ в область памяти, начиная с адреса 1 по адрес 2 включительно

скового адреса при загрузке файла директивой R содержимое счетчика команд не изменяется.

Во время выполнения директив работы с программатором (табл. 4.7) необходимо, чтобы универсальный программатор ППЗУ был подключен к системе Intellex MDS-800, на него было подано питание, в один из двух разъемов программатора была установлена БИС ППЗУ. Параметр <код> определяет, в какой форме следует записывать в ППЗУ информацию из ОЗУ — прямой (Т) или инверсной (F). Параметром <разъем>, который может принимать значения X, Y, Z пользователь определяет, какой из разъемов программатора используется и какая половина байта памяти из ОЗУ системы должна записываться в ППЗУ, если размер слова ППЗУ составляет 4 разряда (табл. 4.8).

Таблица 4.8

Значение параметра <разъем> в директивах монитора для работы с программатором ППЗУ

Значение параметра <разъем>	Номер разъема программатора	В ППЗУ записывается младшая тетрада байта	В ППЗУ записывается старшая тетрада байта
X	2	Да	Да
Y	1	Нет	"
Z	1	Да	Нет

В процессе выполнения директивы P осуществляется контроль записанной в ППЗУ информации и при обнаружении ошибок на терминал выдается сообщение и адрес ППЗУ, при записи по которому допущена ошибка. Процесс программирования при этом прекращается.

4.4. Дисковая операционная система ISIS-II

Операционная система ISIS-II (INTEL SYSTEM IMPLEMENTATION SUPERVISOR) реализована на базе НГМД. Для работы ДОС необходимо 32К байт ОЗУ. При этом в ОЗУ постоянно находится только резидентная часть ДОС объемом 12К байт. На дисках хранятся системные программы (редактор, компиляторы и т. д.), отлаживаемые программы пользователя и файлы исходных данных [27].

Информация на дисках хранится в виде файлов произвольных размеров: либо текстовых в коде КОИ-7 (исходные тексты программ), либо шестнадцатеричных или двоичных (объектные файлы). Каждый файл имеет имя, определяемое пользователем, в виде $\langle \text{устройство} \rangle \langle \text{основное имя} \rangle . \langle \text{расширение} \rangle$ где $\langle \text{устройство} \rangle$ — два определенных алфавитно-цифровых символа внутри двоеточий, соответствующих внешнему устройству, на котором располагается файл (табл. 4.9, 4.10); $\langle \text{основное имя файла} \rangle$ — 1—6 произвольных алфавитно-цифровых символов;

Таблица 4.9

Стандартные внешние устройства

Мнемоническое обозначение	Наименование
:F<N>:	Дискковод с номером <N>
:TI:	Клавиатура телетайпа
:TO:	Печатающее устройство телетайпа
:TP:	Перфоратор телетайпа
:TR:	Устройство ввода с перфоленты телетайпа
:VI:	Клавиатура дисплея
:VO:	Экран дисплея
:HP:	Ленточный перфоратор
:HR:	Устройство ввода с перфоленты
:LP:	Печатающее устройство
:CI:	Пультовый терминал (ввод)
:CO:	„ „ (вывод)

Таблица 4.10

Дополнительные внешние устройства пользователя

Мнемоническое обозначение	Наименование
:R1:	Первое устройство ввода с перфоленты
:R2:	Второе устройство ввода с перфоленты
:P1:	Первый ленточный перфоратор
:P2:	Второй ленточный перфоратор
:I1:	Пультовый терминал (ввод)
:O1:	„ терминал (вывод)
:L1:	Печатающее устройство

<расширение>—1—3 произвольных алфавитно-цифровых символа.

Если <устройство> не является диском, то <основное имя> и <расширение> не указываются. Если в имени файла наименование устройства опущено, то предполагается, что файл располагается на диске, установленном на дисковомодуле :FØ:. Расширение в имени файла может отсутствовать.

Директивы ДОС ISIS-II позволяют с помощью одного неполного имени задавать не один файл, а группу файлов. Так, основное имя файла, расширение или часть того или другого могут быть заменены звездочкой (*). Такое имя файла совпадает с именем, в котором звездочка заменяется на любые символы. Например, имя NAME. *означает все файлы, имеющие имя

Таблица 4.11

Директивы работы с файлами ДОС

Директива	Описание
—COPY<имя 1>,{<имя 2>,...} TO<имя нового файла> {переключатель}	Копирование одного или нескольких файлов, имена которых указаны через запятую до служебного слова TO, в новый файл
—DELETE<имя 1> {Q} {,<имя 2> {Q},...}	Стирание файлов с указанными именами
—RENAME<имя 1> TO <имя 2>	Изменение имени 1 файла на имя 2. Файлы с признаком ЗАЩИЩЕННЫЙ или ФОРМАТНЫЙ переименованы быть не могут. Номера дисководов в именах 1 и 2 должны быть одинаковыми
—ATTRIB<имя> <список операторов>{Q}	Для файла с указанным именем в соответствии со списком операторов (табл. 4.12) отменяются или устанавливаются признаки
—DIR{FOR <имя 1> {TO<имя 2>}}{<переключатель>}	Вывод оглавления для файла или группы файлов, определяемых именем 1, на устройство (файл), определяемое именем 2. Если имя 1 не указано, то выводится полное оглавление диска. Если не указано имя 2, то вывод осуществляется на пультовый терминал
—FORMAT{<дисковод> <имя>}{<переключатель>}	Форматирование диска, установленного на указанный дисковод (:F1: по умолчанию), и присвоение диску имени

NAME и какое-либо расширение. Имя АВ*. HEX определяет файлы, имена которых начинаются с символов АВ, имеющих расширение HEX. Если только один символ в имени не имеет значения, то на его месте ставится знак вопроса (?). Каждому файлу пользователем может быть присвоен один или несколько признаков (атрибутов). Возможны следующие признаки: НЕВИДИМЫЙ, ЗАЩИЩЕННЫЙ, ФОРМАТНЫЙ, СИСТЕМНЫЙ. Признак НЕВИДИМЫЙ обычно присваивается всем файлам, содержащим системные программы. При распечатке оглавления диска, если не принимается специальных действий, файлы с этим признаком не указываются. Файлы с признаком ЗАЩИЩЕННЫЙ не могут быть модифицированы, стерты, переименованы или переписаны на другой диск. Форматные файлы отличаются от защищенных тем, что они автоматически переписываются на другой диск при форматировании последнего (см. ниже). Файлы с признаком СИСТЕМНЫЙ копируются на диск, установленный на дисковом устройстве :F1:, только если в директиве форматирования указан переключатель S.

Директивы работы с файлами (табл. 4.11) позволяют создавать, модифицировать, уничтожать файлы, анализировать оглавление файлов, имеющихся на диске, проводить подготовку дисков к работе с ними. Синтаксические единицы в описании директив, взятые в фигурные скобки, могут отсутствовать. При рассмотрении примеров работы с директивами информация, выводимая на экран терминала операционной системой,

Таблица 4.12

Операторы назначения признаков файлов

Оператор	Назначение
IØ	Отмена признака НЕВИДИМЫЙ
I1	Установка признака НЕВИДИМЫЙ
WØ	Отмена признака ЗАЩИЩЕННЫЙ
W1	Установка признака ЗАЩИЩЕННЫЙ
FØ	Отмена признака ФОРМАТНЫЙ
F1	Установка признака ФОРМАТНЫЙ
SØ	Отмена признака СИСТЕМНЫЙ
S1	Установка признака СИСТЕМНЫЙ

подчеркивается. Признаком готовности ДОС принять директиву является вывод на экран символа тире (—) в начале строки.

Директива копирования файла COPY позволяет копировать файлы на дисках, объединять несколько файлов в один, вводить информацию с перфоленты или клавиатуры с записью на диск, выводить файлы с диска на перфоленту, печать или экран терминала. Так, например, директива — COPY TEXT TO :LP: обеспечивает вывод файла с именем TEXT, расположенного на диске дисковода :FØ:, на печатающее устройство.

Если в качестве имени нового файла (см. табл. 4.11) указывается имя существующего файла, то на экран терминала выводится текст: «Файл уже существует. Стереть?» При нажатии клавиши Y или y и клавиши «Возврат каретки» осуществляется его стирание, и директива COPY выполняется. Например, при объединении файлов TEXT1, TEXT2 в файл PAGE на дисководе :FØ: директива имеет вид

```
— COPY TEXT1,TEXT2 TO PAGE  
PAGE, FILE ALREADY EXISTS  
DELETE ? Y
```

Если в ответ на сообщение о существовании файла с указанным именем пользователь дает ответ, отличный от символа Y или y, то никакие действия не выполняются. При указании в директиве переключателя U сообщение о существовании файла с указанным именем не выводится и директива выполняется сразу. При этом содержимое нового файла добавляется к содержимому старого файла.

В директиве кроме переключателя U может быть указан один из переключателей S, N, P, Q, B. При наличии в директиве переключателя S признаки старого файла (файлов) сохраняются для нового, при отсутствии этого переключателя признаки для нового файла не устанавливаются. При наличии в директиве переключателя S будут копироваться только файлы с признаком СИСТЕМНЫЙ, при наличии переключателя N — только файлы, у которых не установлен признак СИСТЕМНЫЙ или ФОРМАТНЫЙ. Переключатель P обеспечивает паузу для смены диска в дисководе при копировании с одного диска на другой. Переключатель Q обеспечивает перед выполнением копирования посылку на экран запроса на разрешение копирования. Пере-

ключатель В выполняет те же функции, что и переключатель U, однако в случае совпадения имен файлов при копировании прежнее содержимое нового файла теряется.

Директива стирания файлов DELETE стирает файл или группу файлов, о чем выдается сообщение на экран терминала.

Например,

```
—DELETE :F1:PROG.V1  
:F1:PROG.V1,DELETED (Файл : F1:PROG.V1 удален)
```

Если в директиве удаления указано имя несуществующего файла, то на экран выводится соответствующее сообщение. Например,

```
— DELETE PROG. V1  
PROG.V1, NO SUCH FILE (нет такого файла)
```

Если подлежащий удалению файл имеет признак ЗАЩИЩЕННЫЙ или ФОРМАТНЫЙ, то он не удаляется. Например,

```
— DELETE TEXT1  
TEXT1, WRITE PROTECTED (защищен от записи)
```

При указании в директиве DELETE после имени файла переключателя Q ДОС выводит на экран название файла и вопрос «Удалить?». При нажатии клавиши Y или y и клавиши «Возврат каретки» производится удаление файла. Любой другой ответ приводит к переходу к следующему файлу.

При формировании нового файла он не имеет признаков. Для их установки, а также для модификации уже присвоенных признаков используется директива ATTRIB (АТРИБУТЫ) (см. табл. 4.11). При наличии в директиве переключателя Q перед изменением признаков каждого файла ДОС запрашивает у оператора подтверждение необходимости (нажатия клавиши Y или y) этой операции.

На каждом диске имеется оглавление имеющихся на нем файлов с указанием их размеров. Вызов этого оглавления для просмотра осуществляется директивой DIR (см. табл. 4.11). В качестве переключателя в этой директиве используются один из символов I, F или цифра в диапазоне 0—5. Цифра указывает номер дисковода, оглавление диска которого выводится. Если

цифра не указана, то предполагается, что файлы располагаются на диске дисководов :FØ:. Символ I обеспечивает вывод оглавления всех файлов, включая файлы с признаком НЕВИДИМЫЙ. При отсутствии символа I эти файлы в оглавлении отсутствуют. Символ F обеспечивает вывод сокращенного оглавления, содержащего только имена файлов.

Например,

```
— DIR 1
DIRECTORY OF DISKETTE :F1:IS.SYS (оглавление диска 1)
NAME .EXT BLKS LENGTH ATTR (имя, расширение
                               блоков, длина, признаки)
PROGA .HEX 75      9263      W
SUMS      51      6357
              126      (число использованных блоков)
936/2002 BLOCKS USED
```

Перед записью файлов на диск последний должен быть предварительно размечен — отформатирован директивой FORMAT (см. табл. 4.11). Форматируемому диску присваивается имя и на него копируются файлы с диска, установленного на дисковом :FØ:. В качестве переключателя в директиве FORMAT может быть указан один из символов A или S. При указании переключателя A на формируемый диск копируются все файлы с системного диска, установленного на дисковом :FØ:, при указании переключателя S — только файлы с признаком СИСТЕМНЫЙ и ФОРМАТНЫЙ. При отсутствии переключателя копируются только файлы с признаком ФОРМАТНЫЙ.

Создание и редактирование текстовых файлов производятся программой-редактором текста. Процедура редактирования заключается в запуске программы-редактора, загрузке (написании) текстового файла, подлежащего редактированию, редактировании текста (замене, удалении или добавлении некоторых символов и строк) и выводе текстового файла на печать или записи его на диск. В состав DOC ISIS-II входят две версии редактора текста: EDIT и CREDIT. Редактор текста CREDIT располагает более широкими возможностями редактирования.

Запуск программы-редактора текста CREDIT осуществляется директивой

```
CREDIT <имя 1> {TO <имя 2>}
{MACRO (<имя 3>)}
```

где <имя 1>—

имя создаваемого или уже имеющегося файла, подлежащего редактированию, или имя стандартного внешнего устройства, с которого должен быть считан текстовый файл;

<имя 2>—

имя файла, в который будет записан отредактированный текст. По умолчанию отредактированный текст записывается в файл, имя которого отличается от имени 1 расширением .BAK. Если имя 1 уже содержит расширение .BAK, то исходный файл теряется;

<имя 3>—

имя файла, в котором записана требующая выполнения последовательность директив редактора (директивного файла). Если имя 3 не указано, в качестве этого файла используется файл CREDIT. MAC.

Так, например, директива CREDIT :HR: TO MOD. A1 обеспечивает вызов редактора текста, ввод файла с перфоленты и его запись под именем MOD.A1 на диск, установленный на дисковом устройстве :FO:. Директива CREDIT MOD.A1 MACRO(PLM.MAC) осуществляет вызов редактора текста, выполняет редактирование в соответствии с директивами, имеющимися в файле PLM.MAC, и записывает полученный текст в файл MOD.BAK.

Для выхода из программы-редактора текста применяется директива EX <имя>, которая обеспечивает выход из редактора с записью отредактированного текста в файл с указанием имени. Если имя в директиве не указано, то отредактированный текст записывается в исходный файл. Директива EQ позволяет осуществить выход из программы-редактора текста без записи отредактированного текста в файл. В ответ на ввод команды EQ редактор выводит на экран дисплея вопрос QUIT? Нажатие клавиши Y или y приводит к выходу из редактора и передаче управления операционной системе. Введение любого другого символа позволяет продолжать редактирование.

Редактор текста CREDIT может работать в режимах экранного и командного редактирования. В режим экранного редактирования редактор входит сразу после запуска. В этом режиме редактор позволяет осу-

ществлять запись и отображение текста, его стирание, замену старых и вставку новых символов (табл. 4.13). При этом используется курсор — мерцающий знак подчеркивания одного символа. Перемещение курсора по

Таблица 4.1

Директива экранного редактирования

Директива	Описание
CTRL—V ¹	Переход к режиму экранного редактирования, вывод на экран текущей по отношению к положению курсора страницы (20 строк). Курсор считается установленным в начале третьей строки
CTRL—N	Вывод на экран следующей страницы текста
CTRL—P	Вывод на экран предыдущей страницы текста
CTRL—A	Вставка текста. При вводе директивы текст на экране, начиная с позиции курсора, удаляется. После ввода вставляемого текста и повторного ввода директивы удаленный текст восстанавливается
CTRL—C<символ>	Вставка символа после курсора
CTRL—Z	Удаление текста. Курсор устанавливается на первый символ удаляемого текста, затем вводится директива. Курсор перемещается на последний символ удаляемого текста и вновь вводится директива
CTRL—D	Удаление символа, на который указывает курсор
Клавиша HOME	Переход к режиму командного редактирования

¹ Запись CTRL—V означает одновременное нажатие клавишей CTRL и V.

экрану производится с помощью клавишей ↑, ↓, ←, →. Для замены символа внутри выведенной на экран страницы текста необходимо подвести курсор к заменяемому символу и ввести с клавиатуры новый символ. Результат подстановки отображается на экране.

Переход в режим командного редактирования осуществляется нажатием клавиши HOME. В этом режиме курсор не используется, его роль выполняет специальный указатель, имеющий вид ⊥. Указатель на эк-

Директивы командного редактирования

Директива	Описание
L <число>	Перемещение указателя на заданное число строк. Число может быть положительным (перемещение вниз) или отрицательным (перемещение вверх)
L <метка>	Перемещение указателя на символ с заданной меткой
J <число>	Перемещение указателя на заданное число символов. Число может быть положительным (перемещение направо) или отрицательным (перемещение налево)
J <метка>	Перемещение указателя на символ с заданной меткой
TS <номер>	Присвоение символу, заданному положением указателя, метки с указанным номером
TD <номер>	Снятие метки с указанным номером
P <число>	Вывод на экран заданного числа строк текста, начиная со строки, в которой находится указатель. Если число не указано, то выводится одна строка
P <метка>	Вывод на экран текста до символа с указанной меткой
I/<текст>/	Введение заданного текста перед текущим положением указателя
DL <число>	Удаление указанного числа строк относительно положения указателя
DC <число>	Удаление указанного числа символов относительно положения указателя
DC <метка>	Удаление текста от указателя до метки
XM <метка 1>, <метка 2>	Удаление текста, начиная с метки 1 до метки 2, и введение его в точку, определяемую положением указателя
XM <метка>, <число>	Удаление заданного числа строк, начиная со строки, содержащей символ с указанной меткой и введение удаленного текста в точку, определяемую положением указателя
XC <метка 1>, <метка 2>	Повторение текста, начиная с метки 1, до метки 2 в месте, определяемом указателем
XC <метка>, <число>	Повторение заданного числа строк, начиная со строки, содержащей символ с указанной меткой, в месте, определяемом указателем
F/<текст>/ <метка>	Поиск указанного текста в области файла между указателем и меткой
F/<текст>/ <число>	Поиск указанного текста в заданном числе строк, начиная со строки, в которой находится указатель
S/<текст 1>/ <текст 2>/ <число>	Замена текста 1 на текст 2 в указанном числе строк, начиная со строки, в которой находится указатель

Директива	Описание
S/<текст 1>/ <текст 2>/ <метка>	Замена текста 1 на текст 2 между указателем и меткой
SQ/<текст 1>/ <текст 2>/ <число> SQ/<текст 1>/ <текст 2>/ <метка>	Аналогичны директиве S, за исключением того, что перед каждой заменой на экран выводится знак вопроса. Замена осуществляется при нажатии клавиши Y или y. При нажатии любой другой клавиши замена отменяется

ране накладывается на символ, расположенный в той же позиции экрана. Указатель можно перемещать по редактируемому тексту с помощью директив L и J (табл. 4.14). При работе с редактором любому символу текста можно поставить в соответствие одну из меток TØ—T9, что делается директивой TS. Редактор автоматически снабжает метками начало и конец файла (метки TT, TE), а также начало и конец части файла, находящейся в ОЗУ комплекса (метки TB, TZ).

В директивах поиска и замены строки символов F, S, SQ при задании отыскиваемого или заменяемого текста вместо некоторых символов может быть указан знак вопроса (?), который интерпретируется как произвольный символ. Возврат к режиму экранного редактирования осуществляется директивой CTRL-V.

Редактор текста CREDIT предусматривает итеративные и условные процедуры редактирования, позволяющие выполнять нужные директивы заданное число раз или пока не будут выполнены определенные условия (достигнут конец файла). Для определения итеративной процедуры группа директив, подлежащих выполнению, должна быть заключена в угловые скобки. Перед первой скобкой в виде числа *n* указывается, сколько раз должно быть повторено исполнение данной группы директив. Если вместо *n* указывается восклицательный знак (!), то приведенный набор директив выполняется от положения указателя до конца файла. Программой допускается пять уровней вложения угловых скобок. Так, директива 6<S/ABC?/MOD/> задает поиск шести четырехсимвольных групп, начинаю-

щихся с символов ABC, и замену этих групп на символы MOD. Директива !<S/ABC?/MOD/> обеспечивает аналогичную замену в области файла от положения указателя до конца файла.

Условное выполнение директив предполагает их повторное исполнение до наступления определенных условий. Директива QU, которая обычно записывается в теле итеративной процедуры, выводит на экран знак «?» и требует от пользователя утвердительного (нажатия клавиши Y или y) или отрицательного (нажатие любой другой клавиши) ответа. Директива QT разрешает исполнение следующей за ней директивы только в том случае, если предварительно получен утвердительный ответ от пользователя. В противном случае следующая директива пропускается. Директива QF, наоборот, разрешает исполнение, если был получен отрицательный ответ.

Директива EL прекращает циклическое исполнение процедуры, заключенной в угловые скобки. Последовательная запись директив QV, QT(QF), EL позволяет осуществить выход из циклических процедур по желанию пользователя.

Циклической процедурой можно также управлять в зависимости от результатов выполнения директив F и S. Для этого служат директивы YF и YT, которые аналогично директивам QF, QT игнорируют или исполняют следующую за ними директиву в зависимости от результата поиска в директиве F или S. Например, итеративная процедура !<F/TTY/,YF,EX,P> осуществляет поиск всех комбинаций TTY и вывод строк, где эти комбинации встречаются. Выход из итеративной процедуры осуществляется после обнаружения последней комбинации TTY или при отсутствии таких комбинаций.

Во время работы с редактируемым файлом можно считывать информацию из любого другого файла, имеющегося в системе. Для того чтобы обратиться к дополнительному файлу, его необходимо предварительно «открыть» с помощью директивы

OR<имя файла>

При этом указатель этого дополнительного файла будет установлен на его начало. Директива

R[<число>]

обеспечивает считывание из этого файла указанного числа строк (если число строк не указано, то считывается одна строка, если число равно нулю, то никаких действий не производится; отрицательные значения числа не допускаются). Считываемые строки записываются в редактируемый файл в соответствии с текущим положением указателя. Указатель дополнительного файла перемещается на соответствующее число строк. При повторном обращении к дополнительному файлу считываются следующие по порядку строки, если указатель этого файла не возвращен к началу файла директивой В. В одно и то же время не могут быть открыты для чтения два и более дополнительных файла.

Во время работы с редактируемым файлом можно записывать информацию из этого файла в предварительно открытый для этого дополнительный файл. Открытие файла для записи осуществляется директивой

OW<имя файла>

Директива W[<число>] обеспечивает запись указанного числа строк в дополнительный файл, открытый для записи. При положительном числе записи подлежат строки, следующие за той, в которой располагается указатель редактируемого файла, при отрицательном — предыдущие строки.

Если в директиве OW указано число $\emptyset$, то в дополнительный файл записываются символы с начала текущей строки до положения указателя. Если число в директиве не указано, то записывается текущая строка. В одно и то же время не могут быть открыты для записи два дополнительных файла и более.

После проведения операций считывания и записи внешние файлы должны быть закрыты с помощью директив CR (файл, открытый для считывания) и CW (файл, открытый для записи).

Редактор текста CREDIT позволяет объединять директивы в последовательности (макрокоманды) для организации сложных процессов редактирования. Макросредства, заложенные в программу CREDIT, позволяют создавать, вызывать (выполнять), хранить и уничтожать такие макрокоманды (табл. 4.15).

Пользователь может создавать директивные файлы, т. е. файлы, содержащие последовательности директив редактора текста. Эти файлы могут быть вызваны ре-

дактором для исполнения с помощью директивы

G<имя директивного файла>

Кроме того, обращение к директивному файлу и выполнение соответствующей последовательности директив возможно сразу после загрузки редактора, если при вызове редактора указывается служебное слово MACRO.

Таблица 4.15

Директивы работы с макрокомандами редактора текста

Директива	Описание
MS <имя макрокоманды> / <тело макрокоманды> /	Создание макрокоманды. Именем макрокоманды может быть любой символ, кроме символов „Возврат каретки“, „Перевод строки“, „Окончание ввода“, „Пробел“, „Сноска“. Тело макрокоманды — это последовательность команд редактора текста, которая должна исполняться при вызове макрокоманды. Команду CTRL-V можно вводить в тело макрокоманды только в начале строки. Параметры макрокоманды в теле макрокоманды выделяются символами %
MF <имя макрокоманды> (<параметр 1> , . . . , <параметр N>)	Обращение к макрокоманде, указывают фактические значения параметров
MD <имя макрокоманды>	Уничтожение макрокоманды
MD <?*>	Уничтожение всех макрокоманд
?M	Вывод на экран имен и текста всех макрокоманд

Процедура ассемблирования в MDS-800 осуществляется резидентными средствами. В состав ДОС ISIS-II входят трансляторы с различных языков программирования: макроассемблера ASM-80 (§ 2.2), ПЛ/М (§ 2.3), FORT-80, Паскаль и др. В результате трансляции программы, написанной на том или ином языке, формируется файл программы в объектных кодах МП КР580ИК80А, имеющий перемещаемый или абсолютный формат. Затем осуществляются связывание (компоновка) программных модулей, если они имеют перемещаемый формат, и размещение их в памяти

(см. рис. 2.1). Разберем процедуру трансляции на примере программ, написанных на языке ассемблера.

Ассемблер в ДОС ISIS-II в общем случае генерирует перемещаемый код. Это позволяет писать и отлаживать различные части (модули) программы отдельно. При этом разные модули могут быть написаны как на ассемблере, так и на языках высокого уровня, например ПЛ/М. Общие для всех модулей переменные, определяемые в данном модуле, имеют описатель PUBLIC (Общий). Переменные, определяемые в других модулях, но используемые в данном, имеют описатель EXTRN (Внешний), а переменные, введенные и используемые только в данном модуле, — описатель USER (Внутренние).

Помимо собственно объектного файла программа-ассемблер формирует листинговый файл, в который входят текст исходной программы, текст соответствующей ей объектной программы, а также (по желанию пользователя) могут входить таблица символов, используемых в программе, таблица перекрестных ссылок и сообщения об ошибках. В таблице символов отдельно перечисляются все общие, внешние и внутренние символы, а также их значения (адреса соответствующих им ячеек памяти). В таблице перекрестных ссылок каждому символу, имеющемуся в программе, ставятся в соответствие строки программы, в которых он определяется, помечаются символы и строки, в которых имеется обращение к данному символу.

Вызов транслятора с языка ассемблера и инициализация процедуры ассемблирования осуществляются директивой

ASM80<имя> {параметры}

где <имя> —

имя текстового файла, подлежащего ассемблированию.

Управляющие параметры (табл. 4.16) указываются в директиве в произвольном порядке через пробел. В директиве ASM80 могут указываться также параметры INCLUDE, LIST, NOLIST, COND, NOCOND, MACRODEBUG, NOMACRODEBUG, GEN, NOGEN, TITLE, EJECT, SAVE, RESTORE, которые выполняют менее важные функции [27].

Ассемблер генерирует объектный файл в перемещаемом или абсолютном формате в зависимости от на-

**Функции возможных параметров в директиве
вызова ассемблера**

Параметр	Задаваемые функции
OBJECT {(имя)}	Объектному файлу, полученному в результате ассемблирования, присваивается указанное имя. Если имя не указано, объектному файлу присваивается имя исходного файла с расширением.OBJ
NOOBJECT	Объектный файл не формируется
DEBUG	В объектный файл, если он формируется, вводится таблица символов
NODEBUG	Ввод таблицы символов в объектный файл запрещается
PRINT{(<имя>)}	Указывается имя файла, в который выводится листинговый файл, формируемый при ассемблировании. Если имя не указано, то листинговому файлу присваивается имя исходного файла с расширением.LST
NOPRINT	Генерация листингового файла запрещается
SYMBOLS	В листинговый файл, если он формируется, включается таблица символов
NOSYMBOLS	Включение таблицы символов в листинговый файл запрещается
XREF	В листинговый файл, если он формируется, включается таблица перекрестных ссылок
NOXREF	Включение таблицы перекрестных ссылок в листинговый файл запрещается
MACROFILE	Указывает, что в ассемблируемой программе имеются макровыводы. В этом случае для работы ассемблера необходимо 48К байт ОЗУ
NOMACROFILE	Указывает на отсутствие макровыводов в ассемблируемой программе. Объем ОЗУ, необходимый для работы ассемблера, сокращается с 48К до 32К байт
PAGELNGTH (<число>)	Указывает число строк в странице листингового файла. По умолчанию страница содержит 66 строк
PAGEWIDTH (<число>)	Указывает число символов в строке листингового файла (от 72 до 132). По умолчанию в строке содержится 120 символов
PAGING	Разбиение листингового файла на страницы и печать заголовка в начале каждой страницы
NOPAGING	Разбиение листингового файла на страницы запрещается

личия в тексте программы специальных управляющих команд (ORG, CSEG, DSEG и т. п. см. § 2.2). При отсутствии этих команд полученная объектная программа будет иметь абсолютный объектный формат с нулевым начальным адресом.

В качестве примера рассмотрим ассемблирование программы замены числа 43 на нуль в области адресов 3У от MEM до MEM+20, предположив, что ее текст предварительно занесен в файл с именем :F1: SUBST.ONE. Директива имеет вид

—ASM80 :F1:SUBST.ONE MACROFILE XREF
SYMBOLS DEBUG

В результате ее выполнения на экран будет выдан текст¹

ISIS-II 8080/8085 MACRO ASSEMBLER V4.0

MODULE PAGE1

LOC OBJ LINE SOURCE STATMENT

```

1
2 ; программа для замены
3 ; числа 43 на число 0
4 ; в ячейках памяти с
5 ; адресами от MEM до
  ; MEM+20
6
7 CSEG
8 ORG 4000H
4000 3E2B 9 START: MVI A,43
4002 0614 10 MVI B,20
4004 211340 C 11 LXI H, MEM
4007 BE 12 LOOP: CMP M
4008 C20D40 C 13 JNZ NZERO
400B 3600 14 MVI M, 0
400D 23 15 NZERO: INX H
400E 05 16 DCR B
400F C20740 C 17 JNZ LOOP
4012 76 18 HLT
19
4013 1405 20 MEM: DW 1300
4000 00 C 21 END START

```

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS

LOOP C 4007 MEM C 4013 NZERO C 400D START C4000

¹ Символом # помечается номер строки, в которой метка встречается впервые. Символом C отмечаются команды, имеющие обращение к кодовому сегменту, D — к сегменту данных, S — к сегменту стека, E — к внешним переменным.

ASSEMBLY COMPLETE, NO ERRORS
ISIS-II ASSEMBLER SYMBOL CROSSREFERENCE, V2.1

LOOP	12#	17
MEM	11	20#
NZERO	13	15#
START	9#	21

CROSS REFERENCE COMPLETE

Редактирование связей и размещение объектных программ осуществляется следующим образом. Трансляторы ДОС ISIS-II формируют объектные программы, имеющие в общем случае перемещаемый формат, причем каждый объектный файл формируется транслятором в виде четырех сегментов (кодový сегмент, сегмент данных, сегмент стека и сегмент памяти). В кодовом сегменте транслятор располагает коды команды константы, которые могут быть расположены в ПЗУ. В сегменте данных транслятор помещает переменные и места для буферов ввода—вывода. Сегмент стека представляет собой область памяти, зарезервированную под стек, сегмент памяти—свободную область памяти. Перемещаемые объектные модули, полученные в результате трансляции, могут быть объединены директивой LINK (СВЯЗАТЬ) в одну общую программу, которая также будет иметь перемещаемый формат. Размещение программы в памяти с указанием конкретных адресов осуществляется директивой LOCATE (ЗАГРУЗИТЬ).

Директива LINK объединяет входные модули в один выходной модуль соединением всех кодовых сегментов в один кодовый сегмент, всех сегментов данных в один сегмент данных и так далее путем корректировки их относительных адресов и согласования всех внешних (межмодульных) связей. Это согласование осуществляется помещением в поле адреса каждой команды, обращающейся к внешнему символу, соответствующего значения абсолютного или относительного адреса этого символа в объединенном модуле. Директива имеет вид

LINK <список входных модулей> TO <имя входного модуля> {<параметры>}

где <список входных модулей> — последовательное (через запятую) перечисление следующих выражений:

<имя файла> {(<имя модуля 1>, <имя модуля 2>, ...)}

в которых

<имя файла> — имя обычного или библиотечного (см. ниже) файла;

<имя модуля> — имена модулей в файле, если файл библиотечный.

Если имена модулей отсутствуют, а файл библиотечный, из него выбираются только те модули, которые необходимы для полного согласования межмодульных связей между модулями, указанными в явном виде в директиве LINK.

Полученный в результате редактирования связей (компоновки) перемещаемый объектный модуль записывается в выходной файл, указанный в директиве LINK. Параметры в этой директиве указывают на выполняемые сервисные функции. Введение параметра MAP инициирует вывод на экран таблицы распределения памяти. Параметр PRINT (<имя файла>) вводится, если таблица распределения памяти должна быть выведена не на экран, а в указанный файл. Параметром NAME (<имя модуля>) можно переопределить имя выходного модуля, если в директиве указан библиотечный выходной файл. Следует отметить, что порядок соединения модулей соответствует порядку, в котором модули записаны в директиве LINK.

Программа, полученная с помощью директивы LINK, непосредственно выполняться не может, так как все адресные ссылки в программе отпределены относительно базовых адресов сегментов, а сами базовые адреса сегментов еще не определены. Для того чтобы разместить программные сегменты с учетом конкретного распределения адресов ОЗУ, ПЗУ и ППЗУ в микропроцессорной системе используется директива LOCATE, которая складывает каждый относительный адрес в сегменте с соответствующим конкретным базовым адресом сегмента и меняет адресные ссылки. Тем самым относительные адреса межсегментных ссылок преобразуются в абсолютные. В результате директивой LOCATE создается абсолютный объектный модуль, готовый к загрузке в микропроцессорную систему и исполнению. Эта директива имеет вид

```
LOCATE <имя 1>{ТО<имя 2>}{<параметры>}
```

где <имя 1> — имя файла, содержащего перемещаемый объектный модуль;

<имя 2>—имя файла, в который будет записан абсолютный объектный модуль. По умолчанию это имя совпадает с именем 1 без расширения; <параметры>—дополнительные указания для присвоения абсолютных адресов и сервисных функций.

Параметр ORDER({DATA(<адрес 1>),}{CODE(<адрес 2>),}{STACK(<адрес 3>),}{MEMORY(<адрес 4>),}, где адреса 1, 2, 3, 4—базовые адреса сегментов данных, команд, стека и памяти соответственно, которые задают начальные абсолютные адреса сегментов. Если этот параметр не указан, то директивой LOCATE осуществляется последовательное распределение сегментов в следующем порядке: кодовый сегмент, сегменты стека, данных, памяти.

Параметр MAP инициирует вывод на экран таблицы распределения памяти по завершении выполнения директивы LOCATE.

Параметры SYMBOLS, LINES, PUBLICS инициируют вывод на экран таблицы символов, т. е. таблицы имен межсегментных и межмодульных ссылок. Параметр SYMBOLS задает вывод в таблицу символов имен всех внутримодульных межсегментных ссылок с присвоенными им абсолютными адресами. Введение параметра PUBLICS обеспечивает вывод имен всех внешних ссылок. Параметр LINES позволяет включить в таблицу символов номера строк исходной программы и имена программных модулей, если исходная программа написана на языке PL/M-80.

Параметр PRINT <имя файла> позволяет осуществить вывод таблицы символов и таблицы распределения памяти не на экран, а в указанный файл.

Параметр NAME <имя> позволяет определить имя выходного модуля, если выходной файл библиотечный. Параметром STACKSIZE (<число>) задается число байтов в стековом сегменте в абсолютном объектном файле. Если пусковой адрес программы отличается от начального адреса кодового сегмента (сегмента команд), то в директиве LOCATE указывается параметр START (<пусковой адрес>). Параметр RESTART вызывает размещение в ячейках 0, 1, 2 команд безусловного перехода к пусковому адресу, которая используется в реальной микропроцессорной системе для запуска программы после окончания сигнала сброса.

В качестве примера рассмотрим директиву получения абсолютного файла программы для замены числа 43 на число Ø. Процедура ассемблирования этой программы рассмотрена выше.

Директива имеет вид

```
—LOCATE:F1:SYBST.OBJ MAP CODE(ØH) SYMBOLS  
PRINT(:LP:)
```

В результате ее выполнения на печать будет выдан текст

```
ISIS-II OBJECT LOCATER V3.Ø INVOKED BY:  
— LOCATE:F1:SUBST.OBJ MAP CODE(ØH) SYMBOLS  
PRINT(:LP:)
```

SYMBOL TABLE OF MODULE MODULE

READ FROM FILE :F1:SUBST.OBJ

WRITTEN TO FILE :F1:SUBST

VALUE	TYPE	SYMBOL
-------	------	--------

4ØØ7H	SYM	LOOP
-------	-----	------

4Ø13H	SYM	MEM
-------	-----	-----

4ØØDH	SYM	NZERO
-------	-----	-------

4ØØØH	SYM	START
-------	-----	-------

· MEMORY MAP OF MODULE MODULE

READ FROM FILE :F1:SUBST.OBJ

WRITTEN TO FILE :F1:SUBST

MODULE START ADDRESS 4ØØØH

START	STOP	LENGTH	REL	NAME
-------	------	--------	-----	------

ØØØØH	4Ø14H	15H	B	CODE
-------	-------	-----	---	------

4Ø15H	4Ø2ØH	CH	B	STACK
-------	-------	----	---	-------

4Ø21T	F68FH	B69FH	B	MEMORY
-------	-------	-------	---	--------

Организация библиотечных файлов. Удобным средством для хранения программных модулей, с которыми работает пользователь, является библиотека. Создание и ведение библиотечных файлов, в которые записываются программные модули в перемещаемом объектном формате, осуществляются программой LIB. Наличие библиотечных файлов позволяет освободить пользователя от перечисления большого числа модулей, объединяемых директивой LINK. Для этого модули, необходимые для создания какой-либо большой программы, по мере их готовности записываются в один библиотечный файл и впоследствии объединяются указанием этого библиотечного файла в качестве входного файла для директивы LINK. Пользователь может организовывать несколько библиотечных файлов, добавлять и удалять из них программные модули, просматривать оглавление любого библиотечного файла. Программа управления библио-

текой вызывается с помощью директивы LIB. Для создания нового библиотечного файла используется директива

CREATE <имя>

где <имя> — имя, присваиваемое новому библиотечному файлу.

Для того чтобы ввести тот или иной объектный модуль в библиотечный файл, следует воспользоваться директивой

ADD<имя 1> {(<имя объективного модуля 1>,
<имя объектного модуля 2>, ...) } TO
<имя 2>

где <имя 1> — имя файла (возможно библиотечного), содержащего объектный модуль;

<имя объектного модуля i > — имя i -го объектного модуля в исходном файле, если файл является библиотечным;

<имя 2> — имя библиотечного файла, к которому будет осуществляться добавление модулей.

Для удаления модулей из библиотечного файла используется директива

DELETE <имя> (<имя модуля 1>, <имя модуля 2>, ...)

где <имя> — имя библиотечного файла, из которого производится удаление модулей;

<имя модуля i > — имя i -го модуля, подлежащего удалению из данного библиотечного файла.

В состав директив управления библиотекой входит также директива вывода на экран оглавления библиотечного файла (списка имен, входящих в него модулей)

LIST<имя библиотечного файла> {(<имя модуля 1, ... >), ..., {TO<имя списочного файла>}
{PUBLICS}}

где <имя библиотечного файла> — это имя библиотечного файла, полное оглавление которого должно быть выведено на экран, если не указано <имя модуля i >;

<имя модуля i > — имя i -го модуля, информация о котором должна быть выведена в качестве оглавления;

<имя списочного файла> — это имя файла, в который должен быть произведен вывод оглавления библиотечного файла (по умолчанию оглавление выводится на экран).

Параметр PUBLICS вводится в том случае, если вме-

сте с оглавлением должна быть выведена информация о внешних символах для каждого модуля.

Для завершения работы с библиотечными файлами и для передачи управления ДОС используется команда EXIT.

Отладка абсолютных объектных программ производится с помощью монитора. Загрузка абсолютных объектных модулей в ОЗУ комплекса и обращение к монитору осуществляются с помощью директивы
DEBUG <имя>.

Директива DEBUG загружает программу с указанным именем в ОЗУ отладочного комплекса и передает управление монитору. Если имя загружаемого абсолютного объектного модуля не указано, то управление передается монитору без загрузки модуля.

Отладка программ в системе команд МП производится с использованием команд монитора. После выполнения директивы DEBUG и передачи управления монитору возможен запуск программы на исполнение с помощью директивы G монитора (§ 4.3). В директиве могут быть указаны пусковой адрес программы и до двух точек останова. В точках останова пользователь может использовать любые команды монитора для контроля и модификации содержимого регистров и памяти МП, модификации самой программы. После этого новой командой G пользователь может инициализировать дальнейшее выполнение программы с последующим остановом в новых контрольных точках. Таким образом последовательно осуществляется процедура отладки объектной программы. Результатом отладки является работоспособный абсолютный объектный модуль, который может быть перенесен в реальную микропроцессорную систему.

4.5. Использование внутрисхемного эмулятора ICE-80

Внутрисхемный эмулятор выполнен на базе МП 8080 и является по отношению к центральному процессору отладочного комплекса самостоятельным внешним устройством. Обмен информацией между процессором комплекса и процессором эмулятора осуществляется с помощью специально выделенной области ОЗУ комплекса, называемой памятью обмена. Через память обмена ДОС ISIS-II посылает внутрисхемному эмулятору директивы и начальные параметры, инициализирующие его работу, анализирует его состояния, принимая информацию о результатах выполнения каждой директивы. В свою очередь блок внутрисхемной эмуляции принимает от ДОС директивы пользователя и исполняет их, формируя при этом все информацион-

ные и управляющие сигналы для системы-прототипа. С точки зрения пользователя, внутрисхемный эмулятор может работать в режимах опроса, эмуляции и пошаговом режиме.

В режиме опроса процессор комплекса производит ввод директив и начальных параметров в память обмена, анализ результатов эмуляции, осуществляется также пересылка данных между системой-прототипом и отладочным комплексом.

В режиме эмуляции происходит выполнение программы в системе-прототипе. При этом внутрисхемный эмулятор посылает в систему-прототип все сигналы, которые посылал бы в нее реальный МП КР580ИК80А. Процесс эмуляции осуществляется до тех пор, пока он не будет аппаратным способом остановлен эмулятором и про-

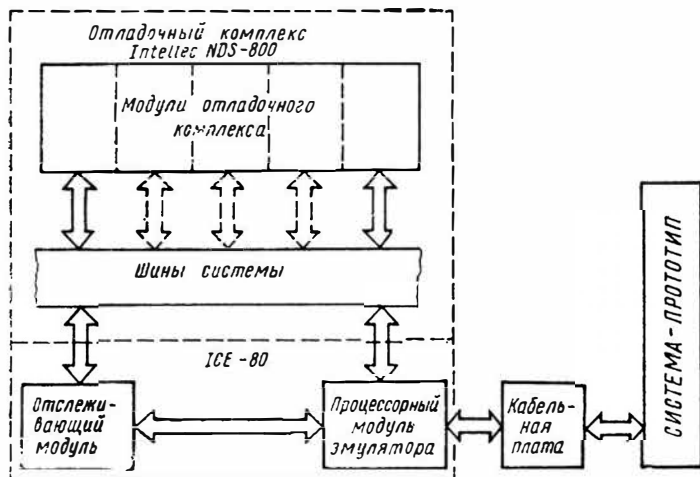


Рис. 4.3. Организация внутрисхемного эмулятора ICE-80 и его положение в системе

грамма ICE-80, расположенная в ОЗУ комплекса и сканирующая память обмена, не обнаружит, что достигнуты точки останова, указанные пользователем. После этого эмулятор переходит в режим опроса.

В пошаговом режиме после выполнения каждой команды процессор останавливается и на экран дисплея выдается содержимое регистров процессора, адресов ЗУ, к которым производилось обращение. Следующая команда выполняется по команде пользователя. При пошаговом режиме с автопродвижением внутрисхемный эмулятор автоматически переходит от команды к команде с задержкой 0,25 с.

Функционально внутрисхемный эмулятор ICE-80 (рис. 4.3) состоит из процессорного модуля, отслеживающего модуля, кабельной платы и разъема для подключения эмулятора к системе-прототипу. Процессорный и отслеживающий модули эмулятора выполнены в виде стандартных печатных плат, расположенных в корпусе отла-

дочного комплекса. К системе-прототипу внутрисхемный эмулятор подключается с помощью гибкого плоского кабеля с разъемом. Кабельная плата расположена непосредственно на кабеле, соединяющем платы эмулятора с системой-прототипом. Взаимодействие между платами внутрисхемного эмулятора осуществляется по внутренней шине эмулятора.

Системные функции между процессорным и отслеживающим модулями внутрисхемного эмулятора распределены следующим образом. Отслеживающий модуль обеспечивает прием команд от комплекса и инициирует исполнение процессорным модулем соответствующих подпрограмм обслуживания, записанных в ПЗУ отслеживающего модуля. Отслеживающий модуль обеспечивает также запоминание информации о выполнении 44 последних машинных циклов эмуляции. В функции отслеживающего модуля входит анализ кодов адресов, команд, данных, кодов управляющего слова, формируемых во время эмуляции процессором системы-прототипа, а также останов эмуляции в случае обнаружения условий, указанных пользователем. Процессорный модуль во время эмуляции представляет свой МП системе-прототипу, обеспечивая возможность выполнения программ пользователя. Во время останова эмуляции процессорный модуль обеспечивает запоминание состояния определенных входов и выходов МП, отключает МП от системы-прототипа и использует его в качестве процессора процессорного модуля внутрисхемного эмулятора для выполнения программ, хранящихся в ПЗУ отслеживающего модуля. Эти программы управляют процессом обмена информацией между внутрисхемным эмулятором и комплексом.

Отслеживающий модуль (рис. 4.4) непосредственно подключается к шинам отладочного комплекса и внутренней шине эмулятора. Основными структурными элементами модуля являются регистр команд, ОЗУ емкостью 256 байт, ПЗУ емкостью 1К байт, компаратор, мультиплексер. Регистр команд внутрисхемного эмулятора служит для приема управляющих директив от процессора комплекса. Регистр состояния содержит информацию о результатах выполнения эмулятором директивы комплекса. Об окончании выполнения эмулятором директивы процессору комплекса сигнализирует схема формирования запросов на прерывания. Регистр эмулятора, к которому осуществляет обращение процессор комплекса, определяется дешифратором обращений.

Одним из наиболее важных функциональных элементов модуля является быстросействующее ОЗУ емкостью 256 байт, 176 байт из которых служат для запоминания информации о 44 последних циклах эмуляции. Такой информацией для каждого машинного цикла являются код управляющего слова, код команды (данных), формируемый на шине данных, код адреса, формируемый на шине адресов. Формирование циклической последовательности адресов для записи этой информации осуществляется с помощью специального адресного генератора. Подача соответствующей информации одновременно с подачей адресов осуществляется с помощью мультиплексера. Мультиплексер служит также для подачи на компаратор адресных и управляющих кодов, формируемых в каждом машинном цикле.

Компаратор представляет собой два 24-разрядных регистра, в каждый из которых загружается определяемое пользователем условие останова эмуляции. В 24-разрядный формат входят 16-разрядный код адреса останова и 8-разрядный код управляющего сло-

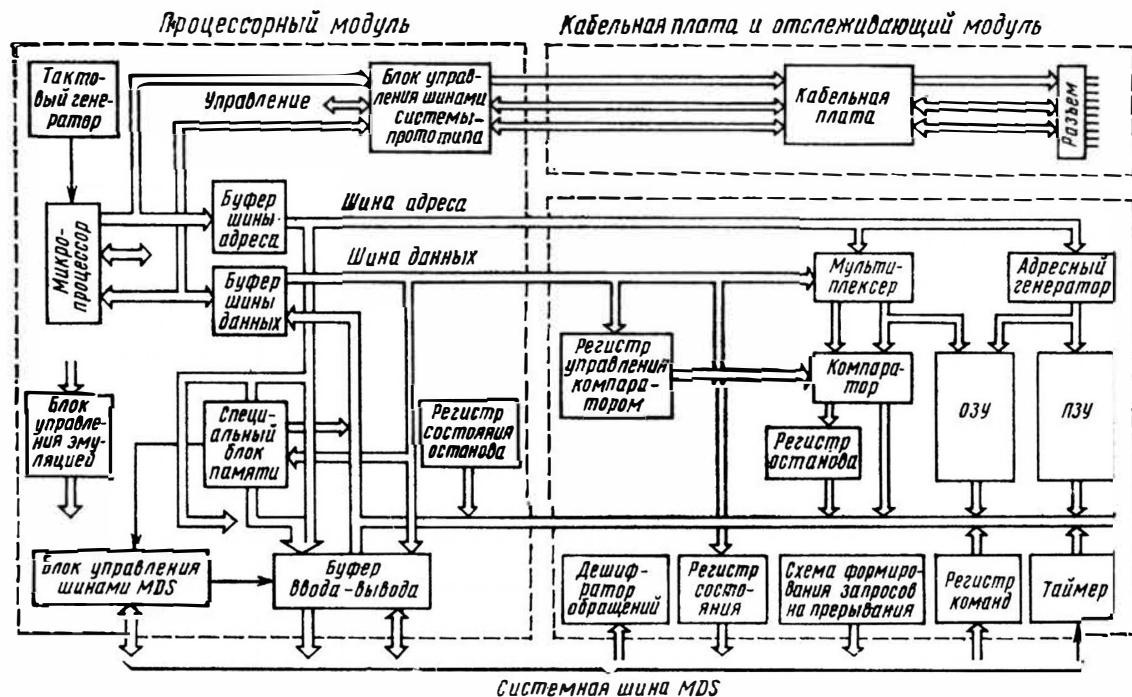


Рис. 4.4. Структурная организация процессорного и отслеживающего модулей внутрисхемного эмулятора ICE-80

ва. Во время эмуляции компаратор анализирует коды адресов и управляющих слов процессора, проходящие через мультиплексер. Совпадение этих кодов с кодом, записанным пользователем в любой из двух регистров компаратора, вызывает останов эмуляции по окончании выполнения текущей команды. Останов эмуляции может быть вызван и рядом других причин, например попыткой эмулируемой программы обратиться к несуществующим ячейкам памяти или устройствам ввода — вывода, при переключении сигнала, контролируемого пользователем, в случае попытки обращения со стороны комплекса к эмулятору во время эмуляции либо если во время эмуляции МП находится в состоянии ожидания или останова более 0,25 с. Информация о причине останова заносится в регистр останова, содержание которого может быть считано со стороны комплекса. В случае останова эмуляции процессор эмулятора переходит в командный режим, в котором он выполняет директивы, поступающие от комплекса.

В отслеживающем модуле расположено ПЗУ емкостью 1К байт, в котором хранятся программы, позволяющие процессору эмулятора в командном режиме отвечать на директивы, поступающие через регистр команд со стороны комплекса. При этом 64 байта быстродействующего ОЗУ отслеживающего модуля используются в качестве рабочей области для программ, расположенных в ПЗУ. В состав отслеживающего модуля входит таймер, позволяющий установить чистое время выполнения эмулируемой программы.

Процессорный модуль включает в себя МП 8080, который в зависимости от режима работы может подключаться либо к шинам системы прототипа, либо к внутренней шине эмулятора, либо к системной шине комплекса. В специальный блок памяти процессорного модуля эмулятора заносится таблица соответствия адресов блоков памяти и внешних устройств, которые комплекс предоставляет системе-прототипу. В функции этого блока памяти входит анализ адресов блоков памяти и внешних устройств, с которыми работает эмулируемая программа. В случае обращения программы пользователя к запрещенной области адресов блоком формируется сигнал, вызывающий останов эмуляции. Блоком памяти идентифицируется также обращение программы пользователя к блокам памяти и внешним устройствам, физическое расположение которых назначено в комплексе. При этом формируется сигнал для блока управления шинами комплекса, позволяющий подключить МП эмулятора к системной шине комплекса и сформировать все управляющие сигналы, необходимые для обращения этого МП к ОЗУ и внешним устройст-

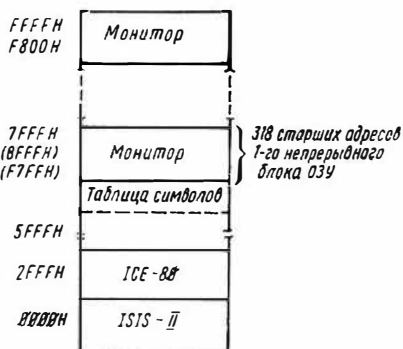


Рис. 4.5. Карта распределения памяти в системе Intellec MDS-800

вам комплекса. Физические адреса ОЗУ и внешних устройств формируются в буфере ввода — вывода процессора эмулятора подстановкой физических номеров (адресов) блоков памяти, взятых из таблицы соответствия, вместо их логических номеров (адресов).

Блок управления шинами системы-прототипа обеспечивает полную имитацию присутствия МП в системе-прототипе во время эмуляции. Этот блок принимает и передает в систему-прототип все требуемые сигналы в каждом машинном цикле эмуляции, за исключением цикла ввода информации из блока памяти или внешнего устройства, физическое расположение которого назначено в отладочном комплексе. Блок управления эмуляцией в основном осуществляет управление процедурами инициализации.

Управление работой внутрисхемного эмулятора осуществляется по командам пользователя с помощью программы ICE-8Ø, входящей в ДОО ISIS-II. Эта программа обеспечивает выполнение всех операций по отладке объектных программ в реальной микропроцессорной системе и по своим возможностям существенно превосходит монитор отладочного комплекса. Вызов программы ICE-8Ø производится командой ICE8Ø.

Все 64К логических адресов, которые могут быть адресованы программой пользователя, с точки зрения внутрисхемного эмулятора ICE-8Ø разделены на 16 логических блоков по 4К байта в каждом. Физическое расположение любого логического блока может быть определено пользователем в системе-прототипе либо в ОЗУ комплекса; 255 логических адресов, соответствующих буферам ввода — вывода внешних устройств, разделены на 16 блоков по 16 адресов в каждом. Физические номера блоков буферов ввода — вывода при их расположении в отладочном комплексе должны совпадать с их логическими номерами. При назначении физических блоков памяти в ОЗУ комплекса не следует использовать те блоки памяти, которые заняты ДОО и программой ICE-8Ø (рис. 4.5).

Назначение логических адресов блоков памяти и устройств ввода — вывода конкретным физическим адресам осуществляется с помощью директивы

```
XFROM MEMORY <блок 1> {TO <блок 2>}  
{GUARDED | UNGUARDED | INTO <блок 3>}1 или  
XFROM IO <блок 1> {TO <блок 2>} {GUARDED  
| UNGUARDED | INTO <блок 3>}
```

где <блок 1> — номер логического блока памяти или устройств физически существующих в системе-прототипе;

¹ Среди выражений, объединяемых символами «|», может одновременно использоваться только одно выражение.

<блок 2> — номер блока памяти или устройств ввода — вывода, физически существующих в системе-прототипе¹;

<блок 3> — номер блока памяти или устройств ввода — вывода, физически существующего в отладочном комплексе. При указании этого блока физический <блок 3> комплекса назначается логическому <блоку 1> системы-прототипа.

Параметр GUARDED — необязательный, он указывается для несуществующих блоков, чтобы обращение к ним приводило к выдаче сообщения об ошибке. Параметр UNGUARDED обязателен для блоков, которые логически и физически существуют в системе-прототипе. Назначение каждого физического блока комплекса логическому блоку системы-прототипа осуществляется отдельной директивой XFROM, причем логические и физические номера блоков ввода — вывода должны совпадать. Если ни один из параметров GUARDED, UNGUARDED, INTO <физический блок> не указан, то по директиве XFROM выводится таблица текущего соответствия физических блоков логическим блокам.

Загрузка эмулируемой программы в систему-прототип осуществляется директивой

LOAD <имя абсолютного объектного файла>

По этой директиве объектный код программы загружается в память системы-прототипа, физически определенную командой XFROM, а таблица символов загружаемой объектной программы помещается в отведенную для нее область ОЗУ комплекса.

Директива

```
GO{FROM<пробел>|<значение>}{UNTIL  
<пробел>|  
<условия останова>}{THEN<пробел>|  
<операции>}  
{CONTINUE<пробел>|<условия>}
```

инициирует исполнение объектной программы, начиная с адреса, указанного в формате команды, и продолжает ее до выполнения какого-либо одного из двух возможных условий останова. В команде GO могут быть указаны дополнительные операции, которые должны быть выполнены в точке останова, а также условия, при выпол-

¹ Выражение <блок 1>TO<блок 2> задает группу последовательных блоков, которая может быть помечена параметрами GUARDED или UNGUARDED.

нении которых эмуляция должна быть продолжена. Выражением FROM<пробел>|<значение> задается начальный адрес эмуляции. Если в этом выражении указан пробел, то в качестве начального выбирается адрес, который был начальным в последней процедуре эмуляции. Если символ <пробел> указан в первой же эмуляции, то в качестве начального адреса выбирается пусковой адрес программы. Значение адреса задается в шестнадцатеричной системе счисления или в виде метки. Начальным адресом может явиться также содержимое регистровой пары и двух соседних ячеек памяти. При этом перед именем регистровой пары вводится знак C, а адрес первой ячейки памяти заключается в квадратные скобки. По умолчанию в качестве начального адреса выбирается адрес, на котором эмуляция была прекращена ранее, либо стартовый адрес загруженной программы.

Выражением {UNTIL<пробел>|<условия останова>} определяются условия, при которых процесс эмуляции прекращается.

Таблица 4.17

Задание условий останова эмуляции

Выражение	Условие останова
<значение> EXECUTED	Выполнение команды с адресом
<значение> READ	<значение> Чтение из ячейки с адресом
<значение> WRITTEN	<значение>, если чтение не является стековой операцией или выборкой команды
<значение> STACKED	Запись в ячейку с адресом <значение>, если запись не является стековой операцией
<значение> UNSTACKED	Запись в стек по адресу <значение>
<значение> INPUT	Выборка из стека по адресу <значение>
<значение> OUTPUT	Ввод по каналу с адресом <значение>
	Вывод по каналу с адресом <значение>

Эти условия задаются одним или двумя выражениями (табл. 4.17). Если указываются два условия останова, то они должны быть разделены оператором OR. Если

после оператора UNTIL указан пробел, условия останова сохраняются от предыдущей эмуляции.

После указания условий останова возможно введение оператора NOBREAK, что позволяет проводить эмуляцию без остановов. Однако в момент появления заданных условий формируется специальный импульс, который может быть использован для запуска логического анализатора. Останов эмуляции возможен также по переключению любого сигнала в отлаживаемой системе. Для этого на кабельной плате имеется специальный шуп-пробник, подача на который нулевого логического уровня инициирует останов эмуляции перед выполнением очередной команды программы.

Выражением {THEN<пробел>|<операции>} вызываются операции, подлежащие выполнению после останова. При указании пробела после очередного останова эмуляции осуществляются операции, указанные в предыдущей директиве эмуляции, либо не выполняется никаких операций, если это первая эмуляция. В качестве операций, подлежащих исполнению, в точке останова может быть указано до четырех операций DUMP, MDCALL<значение> (в записи они разделяются оператором AND). Операция DUMP обеспечивает вывод на экран адреса последней выполненной команды и содержимого всех регистров эмулируемого процессора. Операция MDCALL <значение> инициирует вызов любой программы, записанной в памяти комплекса. Пусковой адрес данной программы указывается элементом <значение>. Выражение {CONTINUE <пробел>|<условия>} определяет условия, при которых эмуляция может быть автоматически продолжена. При указании пробела используются условия предыдущей эмуляции, однако если пробел указан в первой команде эмуляции, то продолжения эмуляции не будет. Пользователь может быть указано одно из следующих условий продолжения эмуляции:

COUNT <значение> — соответствует автоматическому продолжению эмуляции до *n*-го останова, где число *n* задается элементом <значение>;

FOREVER — соответствует продолжению эмуляции после каждого останова;

LOOPING — соответствует автоматическому продолжению эмуляции не с места останова, а с самого

начала с восстановлением состояния процессора при каждом повторе.

Условия продолжения эмуляции могут быть заданы в форме `CONTINUE WHILE` <логический оператор>. В этом случае эмуляция продолжается до тех пор, пока значение логического оператора истинно. Логический оператор имеет следующий синтаксис: <переменная> {`MASK` <значение 1>} {<логическая операция> <значение>}.

В свою очередь <переменная> задается одним из следующих выражений: `MEMORY` <значение>, `DOUBLE MEMORY` <значение>, `REGISTER` <имя>, `DOUBLE REGISTER` <имя>, `FLAG` <имя>, `PIN` <имя>, `INPUT` <имя>. Элемент <значение> определяет адрес ячейки памяти, содержимое которой анализируется, <имя> указывает соответственно регистр, признак, управляющий сигнал, канал ввода — вывода, состояние которого анализируется. Код переменной может быть маскирован константой, указанной после оператора `MASK`. В качестве логической операции могут быть использованы `=`, `<`, `=>`, `>`, `<>`, `=`. В директиве может быть указано до четырех логических операторов, разделенных операторами `OR` или `AND`, их результирующее значение определяется вычислением соответствующей логической функции.

В качестве примера приведем директиву `GO`, которая начинает эмуляцию с адреса с меткой `START` и выполняет ее до тех пор, пока не будет считано содержимое ячейки памяти, имеющей адрес `L1`. После этого содержимое регистров процессора выводится на печать:

`GO FROM START UNTIL L1 READ THEN DUMP`

Второй по значимости директивой внутрисхемного эмулятора является директива `STEP`. Эта директива позволяет осуществлять пошаговую эмуляцию, т. е. эмуляцию с остановами после выполнения определенного числа команд:

`STEP` {`BY` <пробел> | <шаг>} {`FROM` <пробел> | <значение>} {`THEN` <пробел> | <операция>} {`CONTINUE` <пробел> | <условия>}.

Оператором `BY` <пробел> | <шаг> в этой директиве указывается число команд, которые должны быть выполнены до первого останова эмуляции. По умолчанию шаг эмуляции выбирается равным одной команде. Если за оператором `BY` указан пробел, то в качестве ша-

га эмуляции выбирается шаг, который был при предыдущей эмуляции. Если пробел указан в первой команде эмуляции, то шаг эмуляции выбирается равным одной команде.

Шаг эмуляции может быть задан следующим образом: `<значение> {INSTRUCTION}|<значение> JUMPS|<значение> CALLS|<значение> RETURNS`. Параметр `<значение>` указывает число команд, команд передачи управления, вызовов подпрограмм или возвратов из подпрограмм соответственно в каждом шаге эмуляции. Все остальные элементы формата директивы STEP определяются точно так же, как и в директиве GO, и выполняют те же функции.

К директивам управления эмуляцией относится команда RANGE, которая является как бы вспомогательной для директив GO и STEP, поскольку она определяет (переопределяет) область их действия, будучи введенной перед любой из этих директив. Синтаксис директивы

`RANGE<адрес> {TO <адрес>} {<адрес> {TO <адрес>}} ...`

где параметр `<адрес> {TO <адрес>}` определяет отдельный адрес либо целую зону адресов памяти при эмуляции, в пределах которых должны выполняться действия, указанные оператором `{THEN <пробел> | <операции>}` в директивах GO и STEP.

Для того чтобы пользователь мог безошибочно осуществлять эмуляцию любой подпрограммы, в его распоряжении имеется директива

`CALL <значение>`

Элементом `<значение>` определяется пусковой адрес эмулируемой подпрограммы. По директиве CALL эмулятором запоминаются текущее состояние эмулируемого процессора и текущая точка останова эмуляции, затем инициируется эмуляция указанной подпрограммы. По окончании эмуляции подпрограммы (при обнаружении команды RETURN) эмуляция продолжается с того места, где была прервана предыдущая.

Следует отметить, что продолжение эмуляции с той точки, в которой был сделан ее останов, может быть инициировано не только директивами GO и STEP, но и директивой CONTINUE (ПРОДОЛЖИТЬ).

Останов эмуляции в произвольный момент времени (по желанию пользователя) может быть осуществлен

нажатием клавиши INTERRUPT 4 (ПРЕРЫВАНИЕ 4) на передней панели комплекса. Командой CONTINUE может быть инициировано возобновление эмуляции с той команды программы, перед которой произошел останов.

Описанные выше директивы внутрисхемного эмулятора служат для инициализации исполнения программ в системе-прототипе и осуществления остановов в контрольных точках программ. Для того чтобы в точках останова пользователь мог контролировать и изменять по своему усмотрению состояние процессора и программу (что собственно и необходимо для отладки), в состав директив внутрисхемного эмулятора включена большая группа директив, выполняющих мониторинговые функции.

Директива BASE определяет систему счисления выводимой на экран информации (независимо от указанной системы счисления адреса ячеек памяти выводятся в шестнадцатеричном формате)

BASE <указатель типа>

Возможны следующие указатели типа: DECIMAL (Десятичная), OCTAL (Восьмеричная), HEXADECIMAL (Шестнадцатеричная), BINARY (Двоичная), ASCII (В коде КОИ-7), SIMBOLIC (Символическая). Директива BASE обычно вводится после загрузки эмулируемой объектной программы.

Команда DISPLAY используется для вывода на экран информации о текущем состоянии системы-прототипа. Возможны 15 синтаксических вариантов команды DISPLAY. Директивы

DISPLAY MEMORY <адрес 1> {TO<адрес 2>}
DISPLAY DOUBLE MEMORY <адрес 1>
{TO<адрес 2>}

позволяют вывести на экран содержимое ячеек памяти в указанном диапазоне адресов, причем в том формате, который указан директивой BASE. Директивы

DISPLAY REGISTER <имя>
DISPLAY DOUBLE REGISTER <имя>
DISPLAY ALL REGISTERS
DISPLAY INPUT <имя>

позволяют вывести на экран текущее содержимое регистров, регистровых пар, буферов каналов ввода — вывода с указанными именами. Директивы

DISPLAY FLAG <имя>
DISPLAY PIN <имя>
DISPLAY ALL FLAGS
DISPLAY ALL PINS

обеспечивают вывод на экран информации о текущем состоянии флаговых триггеров и состоянии выводов HLTA (ПОС), HOLD (ЗХ), INT (ЗПР), READY (ГТ) эмулируемого процессора. Директива

DISPLAY <значение>

является по существу директивой преобразования формата, так как она указанное значение выводит на экран в формате, указанном директивой BASE. Директива

DISPLAY ALL SIMBOLS

позволяет вывести на экран таблицу символов эмулируемой программы. Директивой

DISPLAY STACK <число>

выводится на экран содержимое заданного числа ячеек верхней области стека. Директива

DISPLAY NESTING

позволяет вывести на экран пусковые адреса всех подпрограмм, выполняемых в настоящий момент, и адреса команд, вызвавших обращение к подпрограммам. Директива

DISPLAY CYCLES <число>

выводит на экран в виде списка управляющие слова процессора, коды, передаваемые по шинам адресов и данных в течение указанного числа последних эмулируемых циклов. Указанное число не должно быть больше 44.

Для изменения числового значения того или иного параметра (кода команды, кода оператора, адреса), что является крайне необходимым при выполнении отладочных операций, можно воспользоваться директивой CHANGE. Эта директива заменяет текущее значение элемента, указанного в формате команды слева от знака равенства, на новое значение, указанное справа от знака равенства. Семь синтаксических вариантов директивы CHANGE приведены ниже:

CHANGE MEMORY <адрес>=<значение>
 CHANGE DOUBLE MEMORY <адрес>=
 <значение>
 CHANGE REGISTER <имя>=<значение>
 CHANGE DOUBLE REGISTER <имя>=
 <значение>
 CHANGE FLAG <имя>=<значение>
 CHANGE OUTPUT <имя>=<значение>
 CHANGE VALUE <символ>=<значение>

Последняя директива изменяет текущее значение символа, указанного в таблице символов.

Директива SEARCH осуществляет ассоциативный поиск ячеек в пределах заданной области памяти, маскированное содержимое которых совпадает со значением, указанным пользователем, причем код маски также же задается пользователем. Синтаксис директивы SEARCH

SEARCH {DOUBLE} MEMORY <адрес>
 {TO <адрес>}
 {MASK <значение>}FOR <значение>

Первой строкой директивы задается область памяти, в которой осуществляется поиск. Параметром {MASK <значение>} определяется код маски. Оператором FOR <значение> задается искомое содержимое ячейки. В результате выполнения директивы адреса ячеек памяти, в которых содержатся искомые данные, выводятся на консоль.

В состав директив ICE-80 входят директивы FILL и MOVE, которые функционально эквивалентны соответствующим директивам монитора. Директива FILL позволяет заполнять определенную область памяти константой. Синтаксис директивы FILL

FILL {DOUBLE} MEMORY <адрес>
 {TO <адрес>}
 {WITH <значение>}.

Эта директива осуществляет заполнение области памяти, указанной оператором MEMORY <адрес> {TO <адрес>}, константой, которая в свою очередь указывается параметром WITH <значение>.

Директива MOVE позволяет переписывать информацию из одной области памяти в другую. Ее синтаксис

MOVE MEMORY <адрес>{TO<адрес>} INTO
 MEMORY<значение>

Первая часть директивы задает область адресов ячеек памяти, содержащее которых подлежит перемещению. Оператор INTO MEMORY <значение> определяет начальный адрес размещения перемещаемой информации. Если в директиве MOVE оператор MEMORY будет заменен оператором MDSMEM, то указанные адреса будут абсолютными адресами памяти комплекса.

Директивы SAVE и LIST внутрисхемного эмулятора обеспечивают запись отлаженных файлов и их таблиц символов на диск. Директива LIST позволяет сделать на диске копию объектного файла, имеющегося в системе-прототипе. Синтаксис директивы LIST

LIST <имя файла>

где <имя файла> — имя дискового файла, в который должна быть выведена объектная программа из системы-прототипа.

Директива SAVE позволяет вывести таблицу символов и содержимое некоторой области памяти системы-прототипа в дисковый файл. Синтаксис директивы SAVE

SAVE<имя файла><адрес>{TO<адрес>}

где <имя файла> — имя дискового файла, в который должен быть осуществлен вывод информации из указанной области памяти.

Добавление символов (с указанием их числовых значений) в таблицу символов программы пользователя возможно с помощью директивы EQUATE. Синтаксис директивы

EQUATE <символ>=<значение>

Для выхода из программы внутрисхемного эмулятора служит директива EXIT. Эта директива осуществляет передачу управления монитору комплекса. В заключение приведем общую последовательность отладки аппаратного и программного обеспечения с использованием внутрисхемного эмулятора.

1. Внутрисхемный эмулятор подключается к системе-прототипу. Программа ICE-80 загружается в ОЗУ комплекса.

2. Производится назначение блоков памяти и внешних устройств ввода — вывода системы-прототипа.

3. Осуществляется пошаговое выполнение программ либо выполнение программ с остановом в контрольных точках.

4. В точках останова с использованием директив внутрисхемного эмулятора производится анализ правильности выполнения программы. Если необходимо, осуществляется коррекция программы.

5. По мере отладки программного обеспечения в систему-прототип добавляются реальные блоки памяти и устройства ввода — вывода. Соответственно переназначаются физические адреса и проверяется работоспособность введенных устройств на реальной программе пользователя.

6. По завершении отладки технических средств и программного обеспечения отлаженная абсолютная объектная программа записывается на гибкий диск, система-прототип отсоединяется от внутрисхемного эмулятора, БИС МП устанавливается в соответствующий разъем системы-прототипа.

Глава 5.

Примеры микропроцессорных систем на основе комплекта БИС серии КР580

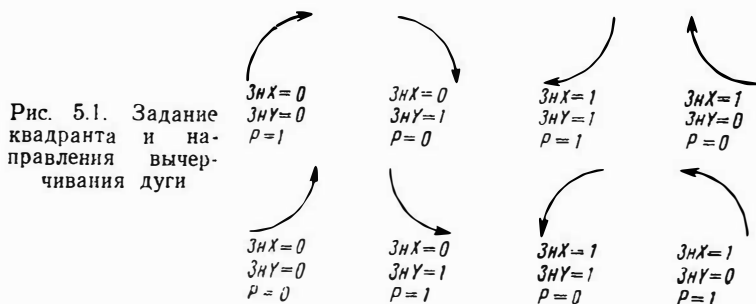
5.1. Проектирование микропроцессорной системы управления чертежным графическим автоматом

Рассмотрим проектирование микропроцессорной системы управления чертежным графическим автоматом (координатографом) «Итекан-2М». Этот автомат предназначен для вычерчивания графиков, схем, чертежей и другой графической документации в соответствии с программой, вводимой в устройство управления с перфоленты. Функции, выполняемые микропроцессорной системой, в этом случае являются типичными для оборудования с числовым программным управлением: станков и роботов.

Кодирование входной информации. Вычерчиваемое на чертежном автомате изображение разбивается на простейшие элементы: отрезки прямых и дуги окружности. Более сложные кривые аппроксимируются с помощью указанных элементов. Каждому отрезку прямой и каждой дуге окружности соответствует один кадр на пятидорожечной перфоленте. Такой кадр (рис. 5.1, табл. 5.2) включает код, определяющий вид вычерчиваемой линии, т. е. команду (ПРЯМАЯ, ДУГА Ц, ДУГА НЦ), номер

одного из двух имеющихся перьев, которые могут иметь разную толщину линии, положение пера (поднято, опущено), тип линии (непрерывная, штриховая, штрихпунктирная) и числовую информацию.

Для отрезка прямой (команда ПРЯМАЯ) указываются длины проекций отрезка на оси координат X и Y и знаки этих проекций ($ЗнX$ и $ЗнY$). Команда Дуга Ц обеспечивает вычерчивание дуги, содержащей целое число квадрантов. Она начинается и заканчивается на осях координат, проходящих через центр окружности. Для команды ДУГА Ц указываются число квадрантов и радиус окружности. За начальную точку дуги принимается



текущее положение пера. Направление вычерчивания дуги и ориентация задаются с помощью знаков приращения координат $ЗнX$ и $ЗнY$ для начального квадранта вычерчиваемой дуги и признака P , равного $\emptyset$ при удалении пера от оси Y и 1 при приближении к оси Y (рис. 5.1). Команда ДУГА НЦ обеспечивает вычерчивание дуги общего вида. В этом случае числовая информация содержит координаты начальной точки X_0 , Y_0 относительно центра окружности и сумму перемещений вдоль осей координат $D_x + D_y$ до конца дуги. Направление вычерчивания и ориентация дуги задаются величинами $ЗнX$, $ЗнY$ и P , так же как в команде ДУГА Ц. Числовая информация располагается, начиная с третьей строки кадра, и занимает три младших разряда. В двух старших разрядах строк располагаются признаки числовой информации, указывающей ее смысл (ΔX , ΔY , X_0 , Y_0 , R , $D_x + D_y$). Числовая информация вводится, начиная со старших разрядов, и может занимать различное число строк. Для прямой, параллельной оси координат, указывается только длина проекции на одну ось. Единица

Таблица 5.1

Кодирование входной информации для чертежного автомата

Тип информации	Информация	Код
Команда	ПРЯМАЯ	001
	ДУГА Ц	010
	ДУГА НЦ	011
	КОНЕЦ ЧЕРТЕЖА	111
Положение пера	Поднято	0
	Опущено	1
Номер пера	Перо 1	0
	Перо 2	1
Тип линии	Непрерывная	00
	Штриховая	01
	Штрихпунктирная	10
Число квадрантов	1 квадрант	00
	2 квадранта	01
	3 квадранта	10
	4 квадранта	11
Признак Р приближения к оси Y	Удаление от оси Y	0
	Приближение к оси Y	1
Признак числовой информации	Проекция на ось X— ΔX , радиус R или абсцисса центра дуги X_0	00
	Проекция на ось Y— ΔY , ордината дуги Y_0	01
	Сумма перемещений вдоль осей $D_x + D_y$	10
Кодовая комбинация	КОНЕЦ КАДРА	110
ЗнX — знак проекции на ось X	Движение вправо	0
	Движение влево	1
ЗнY — знак проекции на ось Y	Движение вверх	0
	Движение вниз	1
Конец команды	Конец команды	1111
Начало чертежа	ЗОНА	11111

младшего разряда соответствует перемещению пера на 0,1 мм. Размеры рабочего поля координатографа 600×850 мм, в связи с этим максимальное перемещение пера в каждом кадре не может превышать 8500, т. е. числовая информация не может занимать более пяти строк перфоленты.

Кадр информации на перфоленте замыкает строка, содержащая кодовую комбинацию КОНЕЦ КАДРА и знаки проекций на оси координат Z_nX и Z_nY . После считывания кода КОНЕЦ КАДРА чертежный автомат оста-

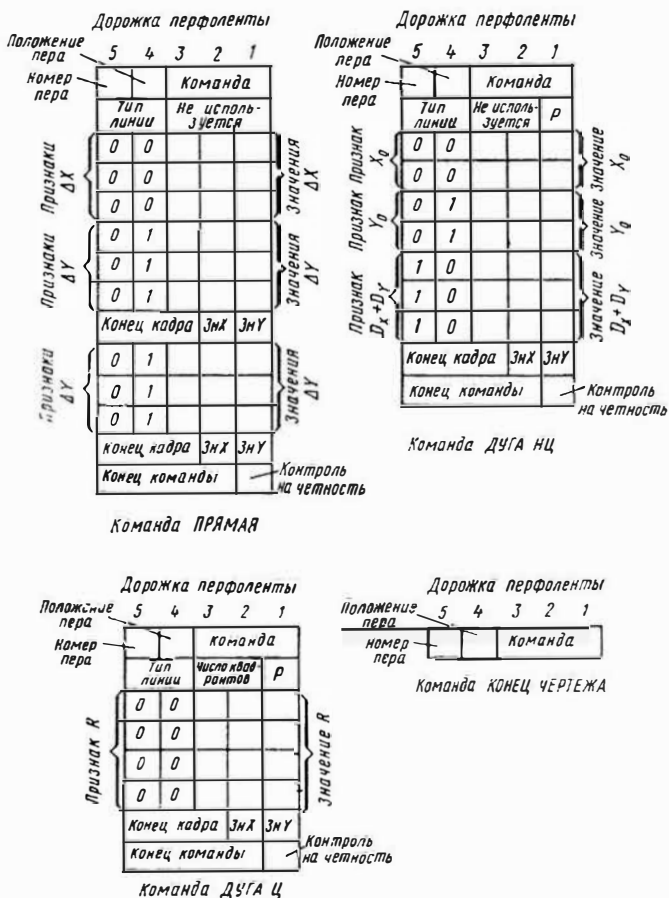


Рис. 5.2. Структура кадров на перфоленте

навливает ввод перфоленты и обрабатывает введенный кадр. Если следующий кадр не содержит первых двух строк, то предполагается, что код команды и тип линии не изменились. При необходимости изменить предыдущую команду на новую вслед за признаком **КОНЕЦ КАДРА** должна следовать строка с кодовой комбинацией **КОНЕЦ КОМАНДЫ**. Младший разряд этой строки служит для контроля правильности ввода информации. Его значение формируется таким образом, чтобы суммарное число пробивок на перфоленте, начиная со строки, задающей код команды, было четным. Если при считывании информации с перфоленты обнаруживается ошибка по четности, дальнейшая работа прекращается и загорается лампочка «Ошибка по четности».

В начале всего массива информации, относящегося к данному чертежу, должна следовать кодовая комбинация **ЗОНА**, а в конце — специальная команда **КОНЕЦ ЧЕРТЕЖА**, при выполнении которой осуществляется сброс всего устройства.

Органы управления и индикации чертежного автомата. На пульте управления имеются следующие органы:

1. Кнопочные выключатели «→», «←», «↓», «↑» служат для ручного перемещения чертежной головки в нужную точку чертежа. Перемещение головки в соответствующем направлении осуществляется при нажатом кнопочном выключателе.

2. Кнопочный выключатель «— —», нажатый одновременно с одной или двумя выключателями «→», «←», «↓», «↑», обеспечивает вычерчивание «Пером 1» штриховой линии.

3. Кнопочные выключатели «Перо 1» и «Перо 2» обеспечивают опускание соответствующих перьев. При отпускании кнопочного выключателя перо поднимается.

4. Кнопочный выключатель «Ввод». При нажатии начинаются ввод перфоленты и поиск символа **ЗОНА**. Загорается лампочка «Ввод».

5. Кнопочный выключатель «Останов» останавливает обработку введенного кадра. Загорается лампочка «Останов».

6. Кнопочный выключатель «Пуск» возобновляет обработку введенного кадра, лампочка «Останов» гаснет.

7. Кнопочный выключатель «Сброс» обеспечивает установку всех схем чертежного автомата в начальное состояние.

8. Лампочка «Ошибка по четности» загорается при обнаружении ошибки по четности. Кроме того, в этом случае останавливаются дальнейший ввод перфоленты и обработка введенного кадра. Одновременно загорается лампочка «Останов». Продолжение работы чертежного автомата возможно при нажатии кнопочного выключателя «Пуск».

9. Лампочка «Формат» загорается при выходе чертежной головки за поле чертежа. Причиной этого является замыкание одного из четырех концевых выключателей « $-X$ », « $+X$ », « $-Y$ », « $+Y$ ». При этом обработка введенного кадра останавливается и загорается лампочка «Останов». Дальнейшая работа чертежного автомата возможна только после отвода чертежной головки от края чертежа с помощью одной из кнопок « $\rightarrow$ », « $\leftarrow$ », « $\uparrow$ », « $\downarrow$ ».

10. Переключатели «Поправка X », «Поправка Y ». Для замены одного пера на другое во время вычерчивания необходимо опущенное перо поднять и переместить чертежную головку так, чтобы новое перо оказалось в точности над той точкой чертежа, над которой ранее находилось другое перо, т. е. на расстояние между перьями. В связи с тем что в чертежном автомате могут быть использованы различные головки, для задания расстояния между перьями предусматриваются два 10-разрядных тумблерных регистра: «Поправка X » и «Поправка Y ». Старшие разряды этих регистров указывают знак поправки, на оставшихся девяти разрядах набирается код абсолютной величины расстояния между перьями по данной оси. Единица младшего разряда соответствует элементарному перемещению головки (0,1 мм). Поправки X и Y указываются для перехода от «Пера 1» к «Перу 2».

Управление шаговыми двигателями и перьями. Два шаговых двигателя (ШДХ и ШДY) обеспечивают продольное и поперечное перемещение чертежной головки (рис. 5.3.). В чертежном автомате имеются два блока управления шаговыми двигателями (БУШД), которые осуществляют коммутацию обмоток статоров шаговых двигателей таким образом, что совершается левое или правое вращение их роторов в зависимости от того, на какой из двух входов БУШД поступают последовательности импульсов. Вращательное движение роторов преобразуется в продольное и поперечное перемещение чер-

тежной головки. Каждый поступивший импульс обеспечивает перемещение головки на 0,1 мм. Возможна одновременная подача импульсов на один из входов $-X$, $+X$ и на один из входов $-Y$, $+Y$. При этом осуществляется перемещение головки на один шаг по двум осям. На входы $-X$, $+X$, $-Y$, $+Y$ должны подаваться импульсы положительной полярности обычной для ТТЛ схем амплитуды. Максимальная частота импульсов не должна превышать 300 Гц.

При вычерчивании штриховой линии длина штриха должна составлять около 10 мм, пробел — около 2 мм.

Для штрихпунктирной линии длина большого штриха — около 20 мм, длина малого штриха и пробела — около 2 мм.

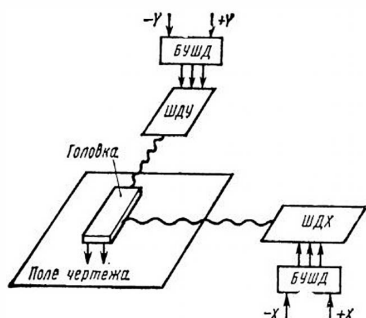


Рис. 5.3. Схема управления движением чертежной головки

Управление каждым из двух имеющихся перьев осуществляется с помощью соленоида и усилителя мощности. При подаче на усилитель мощности высокого уровня напряжения (уровни ТТЛ схемы) через обмотку соленоида протекает ток и перо опущено. При подаче низкого уровня ток через соленоид не протекает и перо поднято.

После изменения уровня на входе усилителя мощности должна быть предусмотрена задержка 0,4 с для срабатывания соленоида.

Алгоритм интерполяции прямых и окружностей. Этот алгоритм определяет порядок аппроксимации вычерчиваемых линий, т. е. такую последовательность элементарных перемещений пера по осям X и Y , при котором перо отклоняется от заданной линии не более чем на шаг. Используемый в данном случае метод оценочной функции характерен отсутствием в нем операций умножения и деления, что облегчает его реализацию на МП КР580ИК80А, не имеющем соответствующих команд.

Начальное положение пера примем за начало координат. Пусть интерполируемый наклонный отрезок лежит в первом квадранте. Тогда его уравнение имеет вид (рис. 5.4,а)

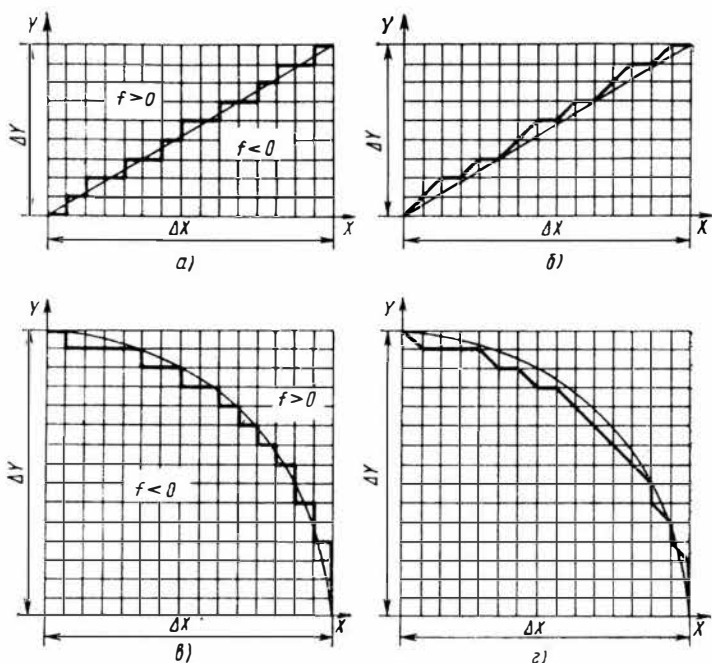


Рис. 5.4. Алгоритм интерполяции:

а — прямой при перемещении по одной оси; *б* — прямой при перемещении по двум осям; *в* — дуги при перемещении по одной оси; *г* — дуги при перемещении по двум осям

$$y = (\Delta y / \Delta x) x.$$

Оценочная функция

$$\bar{f} = y - (\Delta y / \Delta x) x.$$

На интерполируемом отрезке прямой функция $f=0$, выше этой прямой $f>0$, ниже — $f<0$. Так как координаты x и y могут меняться только на дискретное значение (в нашем случае 0,1 мм), то будем измерять координаты x и y в целых числах.

Алгоритм, основанный на вычислении оценочной функции, состоит в следующем:

для текущих значений x_i , y_i вычисляется оценочная функция

$$\bar{f}(x_i, y_i) = y_i - (\Delta y / \Delta x) x_i;$$

если функция $f(x_i, y_i) < 0$, то шаг делается по координате Y ;

если функция $f(x_i, y_i) \geq 0$, то шаг делается по координате X ;

вычисляется новое значение оценочной функции, и процесс повторяется. Вычисление этих операций заканчивается при попадании в точку $\Delta x, \Delta y$.

Поскольку в рассмотренном алгоритме важно не значение оценочной функции, а ее знак, то можно использовать функцию

$$f'(x_i, y_i) = y_i \Delta x - x_i \Delta y.$$

Пусть на j -м шаге алгоритма

$$f'_j(x_j, y_j) = y_j \Delta x - x_j \Delta y.$$

При шаге по оси Y

$$x_{j+1} = x_j; y_{j+1} = y_j + 1; \quad (5.1)$$

$$f'_{j+1}(x_{j+1}, y_{j+1}) = (y_{j+1} + 1) \Delta x - x_j \Delta y = f'_j(x_j, y_j) + \Delta x.$$

При шаге по оси X

$$x_{j+1} = x_j + 1; y_{j+1} = y_j; \quad (5.2)$$

$$f'_{j+1}(x_{j+1}, y_{j+1}) = y_j \Delta x - (x_j + 1) \Delta y = f'_j(x_j, y_j) - \Delta y.$$

Формулы (5.1), (5.2) позволяют вычислять оценочную функцию на каждом шаге алгоритма с использованием только операций сложения и вычитания. Начальное значение $f(0, 0) = 0$.

При интерполяции отрезков прямых, лежащих в других квадрантах, алгоритм остается тем же. Однако импульсы перемещения подаются на входы $+X, -Y$ (II квадрант); $-X, -Y$ (III квадрант); $-X, +Y$ (IV квадрант).

Для ускорения работы чертежного автомата предусмотрена возможность продвижения сразу по двум осям. Для этого определяется и запоминается направление перемещения на текущем шаге интерполирования, затем вычисляется следующее направление перемещения. Если эти два направления соответствуют разным осям, то оба перемещения выполняются одновременно (рис. 5.4, б).

При интерполяции дуг окружностей (рис. 5.4, в) используется оценочная функция вида $f(x, y) = R^2 - (x^2 + y^2)$, где R — радиус дуги; x, y — расстояние от центра до текущей точки дуги вдоль осей X и Y .

На линии окружности $f(x, y) = 0$, внутри окружности $f(x, y) < 0$, вне окружности $f(x, y) > 0$. Алгоритм интер-

поляции аналогичен алгоритму линейной интерполяции, изложенному выше. Однако при $f(x, y) < 0$ и $P=0$ или $f(x, y) \geq 0$ и $P=1$ шаг делается по оси X , а при $f(x, y) < 0$ и $P=1$ или $f(x, y) \geq 0$ и $P=0$ — по оси Y . Направления перемещения указывают коды $3nX$ и $3nY$.

Вычислим оценочную функцию при интерполяции дуг окружности. На j -м шаге интерполяции

$$f_j(x_j, y_j) = R^2 - (x_j^2 + y_j^2).$$

При шаге по оси Y

$$\begin{aligned} x_{j+1} &= x_j; y_{j+1} = y_j - 1; \\ f_{j+1}(x_{j+1}, y_{j+1}) &= R^2 - [x_j^2 + \\ &+ (y_j - 1)^2] = f_j(x_j, y_j) + 2y_j - 1. \end{aligned}$$

При шаге по оси X

$$\begin{aligned} x_{j+1} &= x_j + 1; y_{j+1} = y_j; \\ f_{j+1}(x_{j+1}, y_{j+1}) &= R^2 - [(x_j + 1)^2 + \\ &+ y_j^2] = f_j(x_j, y_j) - 2x_j + 1. \end{aligned}$$

Поскольку начальное положение пера всегда находится точно на дуге окружности, то начальное значение

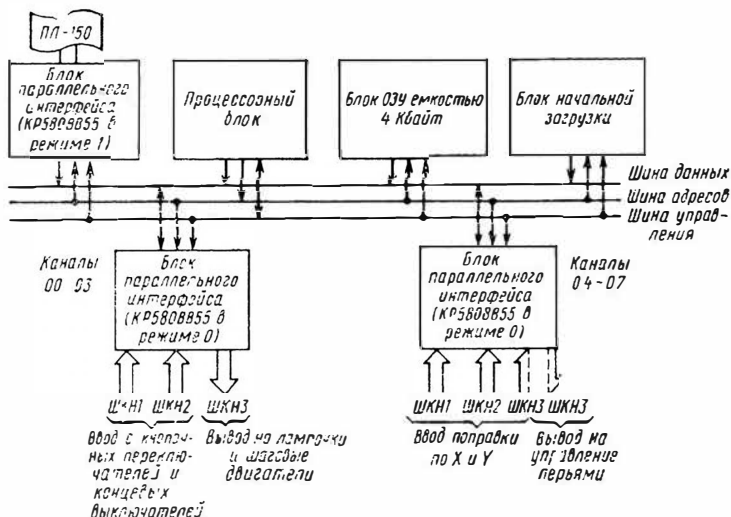


Рис. 5.5. Структурная схема микропроцессорной системы для управления чертежным автоматом

**Распределение разрядов БИС КР580ВВ55 сопряжения
микропроцессорной системы с органами управления
координатографом**

Разряд	Источник или приемник сигнала, направление передачи	
	БИС с адресами 00—03	БИС с адресами 04—07
ШКН1 [0]	Кнопочный переключатель „↑“, ввод	Поправка по X, разряд 1, ввод
ШКН1 [1]	„ „ „↓“, ввод	Поправка по X, разряд 2, ввод
ШКН1 [2]	„ „ „→“, ввод	Поправка по X, разряд 3, ввод
ШКН1 [3]	„ „ „←“, ввод	Поправка по X, разряд 4, ввод
ШКН1 [4]	„ „ „—“, ввод	Поправка по X, разряд 5, ввод
ШКН1 [5]	„ „ „П1“, ввод	Поправка по X, разряд 6, ввод
ШКН1 [6]	„ „ „П2“, ввод	Поправка по X, разряд 7, ввод
ШКН1 [7]	„ „ „Ввод“, ввод	Поправка по X, разряд 8, ввод
ШКН2 [0]	„ „ „Ост“, ввод	Поправка по Y, разряд 1, ввод
ШКН2 [1]	„ „ „Пуск“, ввод	Поправка по Y, разряд 2, ввод
ШКН2 [2]	Концевой выключатель —X, ввод	Поправка по Y, разряд 3, ввод
ШКН2 [3]	Концевой выключатель +X, ввод	Поправка по Y, разряд 4, ввод
ШКН2 [4]	Концевой выключатель —Y, ввод	Поправка по Y, разряд 5, ввод
ШКН2 [5]	Концевой выключатель +Y, ввод	Поправка по Y, разряд 6, ввод
ШКН2 [6]	Не используется	Поправка по Y, разряд 7, ввод
ШКН2 [7]	„ „	Поправка по Y, разряд 8, ввод
ШКН3 [0]	Лампочка „Ввод“, вывод	Поправка по X, разряд 9, ввод
ШКН3 [1]	Лампочка „Ост“, вывод	Знак поправки по X, ввод
ШКН3 [2]	Лампочка „Ош. четн“, вывод	Поправка по Y, разряд 9, ввод
ШКН3 [3]	Лампочка „Формат“, вывод	Знак поправки по Y, ввод

Разряд	Источник или приемник сигнала, направление передачи	
	БИС с адресами $\emptyset\emptyset-\emptyset3$	БИС с адресами $\emptyset4-\emptyset7$
ШКНЗ [4]	Вход — X шагового двигателя, вывод	Вход блока управления пером 1, вывод
ШКНЗ [5]	Вход $+X$ шагового двигателя, вывод	Вход блока управления пером 2, вывод
ШКНЗ [6]	Вход — Y шагового двигателя, вывод	Не используется
ШКНЗ [7]	Вход $+Y$ шагового двигателя, вывод	• •

оценочной функции равно нулю. Если дуга проходит через несколько квадрантов, то при их смене необходимо предусмотреть смену знаков перемещений по осям и изменение значения признака P .

Структурная схема микропроцессорной системы и укрупненная схема алгоритма. Основой микропроцессорной системы управления координатографом (рис. 5.5) является процессорный блок (см. § 3.1). Система прерываний в данном случае не используется. Для хранения управляющей программы обычно используется ПЗУ или ППЗУ. Однако на стадии отладки алгоритма целесообразно для этой цели использовать блок ОЗУ статического типа объемом 4К байт. Для ввода управляющей программы в ОЗУ с перфоленты предусматривается блок начальной загрузки (см. § 3.3). Сопряжение устройства ввода с перфоленты с процессорным блоком целесообразно строить на основе БИС КР580ВВ55, работающей в режиме 1 (см. § 3.3).

Управление индикаторными лампочками, перьями и шаговыми двигателями, а также ввод состояния кнопок и концевых переключателей осуществляется с помощью двух БИС КР580ВВ55, работающих в режиме $\emptyset$ (табл. 5.2). При нажатом кнопочном переключателе, замкнутом концевом переключателе или установленном в единицу разряде регистра «Поправка X » или «Поправка Y » на соответствующую линию канала подается нулевой потенциал, в противном случае подается высокий уровень. Выход на сигнальные лампочки осуществляется через инверторы с открытым коллектором, поэтому за-

горание лампочки происходит при Лог.1 на соответствующей линии. Сигнал с кнопочного переключателя «Сброс» подается на линию ВХСБР процессорного блока и далее после инвертирования—на входы СБР МП и всех трех БИС КР580ВВ55.

После включения источника питания осуществляются инициализация микропроцессорной системы, затем ввод состояний управляющих кнопочных переключений и кон-

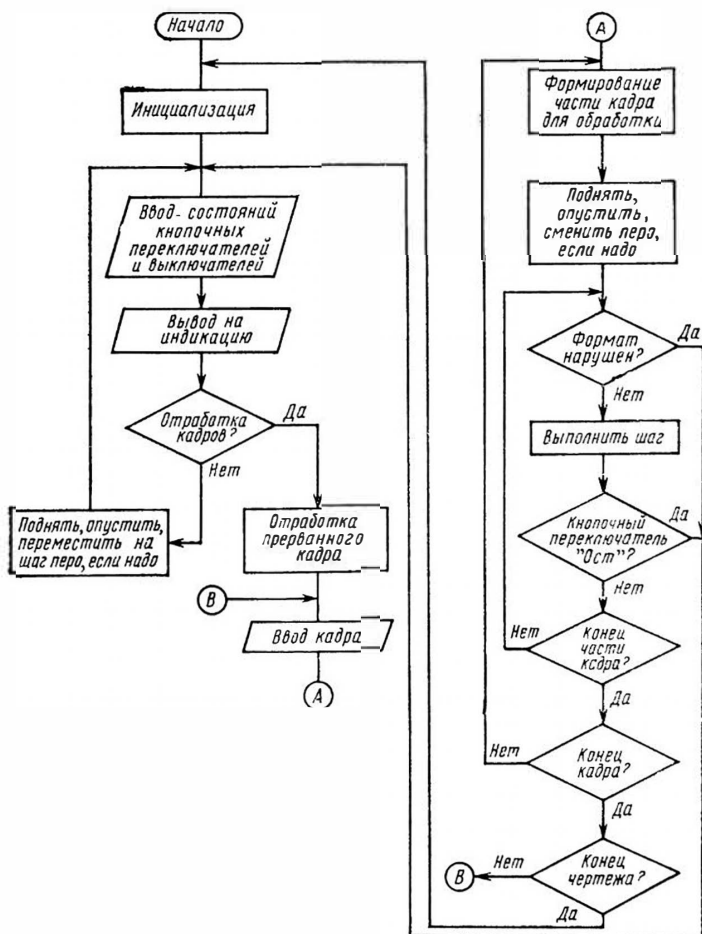


Рис. 5.6. Укрупненная схема алгоритма микропроцессорной системы для управления чертежным автоматом

цевых выключателей и вывод состояния координатографа на сигнальные лампочки (рис. 5.6). Если кнопочные переключатели «↓», «↑», «→», «←» задают перемещение какого-либо пера или отработка кадра остановлена кнопочным переключателем «Ост», одним из концевых выключателей или в связи с ошибкой по четности во введенном кадре, то осуществляются соответствующие действия, если они заданы (подъем или опускание пера, шаг по осям X , Y или обем осям одновременно, сброс признака останова при нажатии кнопочного переключателя «Пуск»). В противном случае осуществляется завершение отработки введенного кадра, если она была прервана.

Таблица 5.3

Переменные, используемые в блоке отработки части кадра

Информация	Идентификатор	Число байтов	Значения
Команда	COM	1	$\emptyset 1H$ — наклонная прямая, $\emptyset 2H$ — дуга $\emptyset 3H$ — прямая, параллельная оси X , $\emptyset 4$ — прямая, параллельная оси Y
Признак P	PFLG	1	$\emptyset \emptyset H - P = 0$, $8 \emptyset H - P = 1$, при отработке прямых $P = 0$
Направление перемещения по оси X	DIRX	1	$\emptyset 9H$ — перемещение в сторону $-X$, $\emptyset BH$ — перемещение в сторону $+X$
Направление перемещения по оси Y	DIRY	1	$\emptyset DH$ — перемещение в сторону $-Y$, $\emptyset FH$ — перемещение в сторону $+Y$
Количество оставшихся шагов по обем осям	DSTN	2	$\Delta X + \Delta Y$
Признак запомненного шага	PST	1	$\emptyset \emptyset H$ — нет, $\emptyset 1H$ — шаг по X , $\emptyset 2H$ — шаг по Y , $\emptyset 3H$ — шаги по X и Y

Далее после ввода очередного кадра формируется информация для обработки части кадра. При задании штриховой или штрихпунктирной линии очередной частью кадра является штрих или промежуток между штрихами. Для непрерывной линии обрабатывается сразу весь кадр. Перед выполнением каждого шага (по X , Y или X и Y одновременно) проверяется состояние концевых выключателей, а после каждого шага — состояние кнопочного переключателя «Ост». При нажатом кнопочном переключателе или замкнутых выключателях обработка кадра прерывается. При поступлении команды **КОНЕЦ ЧЕРТЕЖА** микропроцессорная система устанавливается в исходное состояние.

Разработка программного обеспечения. Вычисления при линейной и круговой интерполяции будем проводить над целыми 16-разрядными числами в дополнительном коде.

Рассмотрим подробно блок обработки части кадра, включая проверку состояния концевых выключателей и кнопочного переключателя «Ост». Пусть перед выполнением этого программного блока установлены нужные значения переменных COM, PFLG, DIRX, DIRY, DSTN (табл. 5.3). Значение PST=0. Кроме того, в стеке размещаются значения оценочной функции (верхушка стека), далее проекции обрабатываемого отрезка ΔY и ΔX или текущие координаты пера относительно центра дуги Y и X . Величины f , ΔY , ΔX , y , x представляют собой 16-разрядные числа в дополнительном коде. Начальные значения $f=0$.

; обработка части кадра PTFRM:	IN Ø1H	; ввод состояния органов управления ; поразрядное инвертирование ; содержимого накопителя
	CMA	
	ANI 3CH	; выделение сигналов с концевых ; выключателей
	JNZ LDCNT	
	CALL PROC	; прерывание обработки кадра и ; переход на ввод управляющих ; сигналов
	IN Ø1H	
	ANI 1H	; подпрограмма определения оси ; перемещения и его выполнения ; ввод состояния органов ; управления
	JZ LDCNT	
		; выделение состояния кнопочного ; переключателя «Ост» ; прерывание обработки кадра и ; переход на ввод управляющих ; сигналов

LHLD DSTN	; занести в H-L число оставшихся
	; шагов
MOV A,H	; проверка содержимого DSTN на
	; ноль
ORA F	
JNZ PTERM	; переход на начало подпрограммы,
	; если отработка части кадра не
	; закончена
LDA PST	; занести в накопитель признак
	; запомненного шага
CPI $\emptyset 1H$	; не был ли запомнен шаг по X?
CZ X	; подпрограмма шага по X
LDA PST	; занести в накопитель признак
	; запомненного шага
CPI $\emptyset 2H$	; не был ли запомнен шаг по Y?
CZ Y	; подпрограмма шага по Y
RET	
; подпрограмма определения оси перемещения и его выполнение	
PROC:	
LDA COM	; занесение команды в накопитель
CPI $\emptyset 3H$	
JZ M2	; переход при прямой, параллельной
	; оси X
CPI $\emptyset 4H$	
JZ M3	; переход при прямой, параллельной
	; оси Y
	; наклонная прямая или дуга окружности
POP H	; выборка из стека оценочной
	; функции f
PUSH H	; восстановление содержимого SP
LDA PFLG	; загрузка в накопитель признака P
XRA H	; определение оси перемещения
JM M1	; переход при $f \geq 0$ и $P=1$ или при
	; $f < 0$ и $P=0$
	; на шаг по оси X
	; шаг по оси Y
CALL FCHY	; подпрограмма коррекции f при
	; шаге по Y
M3:	
CALL STPY	; подпрограмма подготовки шага
	; по Y
RET	
	; шаг по оси X
M1:	
CALL FCHX	; подпрограмма коррекции f при
	; шаге по X
M2:	
CALL STPX	; подпрограмма подготовки шага
	; по Y
RET	

Структура подпрограмм коррекции оценочной функции f при шаге по X и шаге по Y одинаковы. Рассмотрим подпрограмму коррекции оценочной функции при шаге по оси Y

; коррекция оценочной функции при шаге по Y

FCHY:	POP H	; выборка из стека в H-L оценочной
		; функции f
	POP D	; выборка из стека в D-E ΔY (пря-
		; мая) или Y (дуга)

	POP B	; выборка из стека в В-С ΔX (прямая) ; или X (дуга)
	LDA COM	; загрузить в накопитель команду
	CPI Ø1H	; прямая?
	JZ LINE	; переход на обработку прямой
		; коррекция оценочной функции при интерполяции
		; дуги
	DAD D	; 8 команд для вычисления $f=f+$; $+2Y-1$
	DAD D	
	MOV A,L	
	SUI Ø1H	
	MOV L,A	
	MOV A,H	
	SBI ØH	
	MOV H,A	
		; коррекция абсолютной величины Y при интерполя-
		; ции дуги
	LDA PFLG	; анализ признака P и увеличение ; (при P=1) ; или уменьшение (при P≠0) Y
	RAL	
	JC M4	
	DCX D	
		; переключение знака X (DIRX) и признака P
		; (PFLG) при переходе в другой квадрант
	MOV A,D	
	ORA E	
	CZ SWY	; подпрограмма переключения DIRX
		; и PFLG
	JMP M5	
M4:	INX D	
	JMP M5	
		; коррекция оценочной функции при интерполяции
		; прямой
LINE:	DAD B	
M5:	PUSH B	; занесение в стек ΔX (прямая) или ; X (дуга)
	PUSH D	; занесение в стек ΔY (прямая) или ; Y (дуга)
	PUSH H	; занесение в стек оценочной функ-
		; ции f
	RET	
		; подпрограмма переключения DIRX и PFLG
SWY:	LDA PFLG	; занесение в накопитель признака P
	ADI 8ØH	; переключение PFLG
	STA PFLG	; запоминание нового значения P
	LDA DIRX	; занесение в накопитель знака X ; (DIRX)
	XRI Ø2H	; инвертирование 1-го разряда
	STA DIRX	; запоминание нового значения DIRX
	RET	
		; подпрограмма подготовки шага по X
STPX:	LDA PST	; занести в накопитель признак за- ; помненного шага

	CPI Ø1H	; не был ли запомнен шаг по X?
	JNZ M6	
	CALL X	; подпрограмма шага по X
	RET	
M6:	ORI Ø1H	; запоминание шага по X
	CPI Ø3H	; не запомнены ли шаги по X и Y?
	JNZ M7	
	CALL XY	; подпрограмма шага по X и Y одно- ; временно
	XRA A	; установка в нуль накопителя
M7:	STA PST	; запомнить невыполненный шаг
	LHLD DSTN	; уменьшить на 1 содержимое счет-
	DCX H	; чика шагов DSTN
	SHLD DSTN	
	RET	
; подпрограмма	подготовки шага по Y	
STPY:	LDA PST	; занести в накопитель признак за- ; помненного шага
	CPI Ø2H	; не был ли запомнен шаг по Y?
	JNZ M8	
	CALL Y	; подпрограмма шага по Y
	RET	
M8:	ORI Ø2H	; запоминание шага по Y
	CPI Ø3H	; не запомнены ли шаги по X и Y?
	JNZ M9	
	CALL XY	; подпрограмма шага по X и Y одно- ; временно
	XRA A	; установка в нуль накопителя
M9:	STA PST	
	LHLD DSTN	; уменьшить на 1
	DCX H	; содержимое счетчика
	SHLD DSTN	; шагов DSTN
	RET	
; подпрограмма	шага по X	
X:	CALL DEL	; подпрограмма задержки, обеспечи- ; вающая заданную частоту импуль- ; сов
	LDA DIRX	; загрузка в накопитель управляюще- ; го слова для шага по X ; при шаге по $-X$ DIRX = Ø9H, при ; шаге по $+X$ DIRX = ØBH
	OUT Ø3H	; установка в 1 ШКНЗ[4] или ; ШКНЗ[5] в зависимости от на- ; правления движения по оси X
	ANI ØFEH	; установка в нуль младшего разряда ; управляющего слова
	OUT Ø3H	; сброс ШКНЗ[4] или ШКНЗ[5]
	RET	
; подпрограмма	шага по Y	
Y:	CALL DEL	; подпрограмма задержки, обеспечи- ; вающая заданную частоту импуль- ; сов
	LDA DIRY	; загрузка в накопитель управляюще- ; го слова для шага по Y ; при шаге по $-Y$ DIRY = ØDH, при

		; шаг по $+Y$ DIRY = $\emptyset FH$
OUT $\emptyset 3H$		; установка в 1 ШКНЗ[3] или
		; ШКНЗ[7] в зависимости от на-
		; правления движения по оси Y
ANI $\emptyset FEH$		; установка в нуль младшего разряда
		; управляющего слова
OUT $\emptyset 3H$		; сброс ШКНЗ[6] или ШКНЗ[7]
RET		

; подпрограмма шага одновременно по X и Y
 XY: CALL DEL ; подпрограмма задержки
 LDA DIRX ; загрузка управляющего слова для
 ; шага по X
 OUT $\emptyset 3H$; установка в 1 ШКНЗ[4] или
 ; ШКНЗ[5]
 LDA DIRY ; загрузка управляющего слова для
 ; шага по Y
 OUT $\emptyset 3H$; установка в 1 ШКНЗ[6] или
 ; ШКНЗ[7]
 LDA DIRX ; загрузка управляющего слова для
 ; шага по X
 ANI $\emptyset FEH$; сброс младшего разряда управляю-
 ; щего слова
 OUT $\emptyset 3H$; сброс ШКНЗ[4] или ШКНЗ[5]
 LDA DIRY ; загрузка управляющего слова для
 ; шага по Y
 ANI $\emptyset FEH$; сброс младшего разряда управляю-
 ; щего слова
 OUT $\emptyset 3H$; сброс ШКНЗ[6] или ШКНЗ[7]
 RET

5.2. Микропроцессорная реализация цифровых фильтров

Каждый непрерывный сигнал, представленный изменяющимся во времени напряжением, характеризуется своим спектром частот. Любая частота или полоса частот может быть усилена, ослаблена, **исключена или выделена** фильтрацией. Фильтрация — это процесс изменения спектра частот сигнала. Примерами фильтрации сигналов могут служить исключение шумов, вызванных несовершенством каналов передачи, разделение двух или более сигналов, специально смешанных для увеличения пропускной способности канала связи, демодуляция сигналов, преобразование дискретных сигналов в непрерывные и в сигналы с ограниченным спектром. Любой непрерыв-

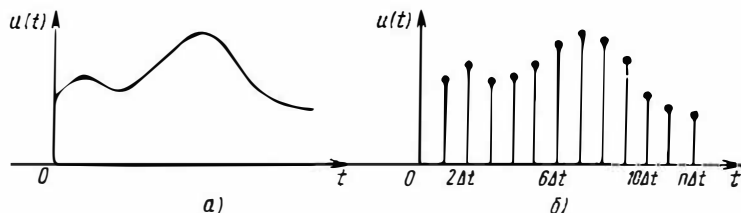


Рис. 5.7. Непрерывный сигнал (а) и его представление дискретными значениями (б)



Рис. 5.8. Схема цифрового фильтра

ный сигнал, спектр которого ограничен верхними частотами, может быть представлен в виде последовательности своих значений (рис. 5.7), отстоящих друг от друга на время Δt . Если $1/\Delta t \geq 2f$, где f — максимальная частота в спектре непрерывного сигнала $u(t)$, то по дискретным значениям $u(n\Delta t)$, $n=0, 1, \dots$, можно однозначно восстановить непрерывный сигнал $u(t)$ [21]. Таким образом, можно говорить о частотном спектре дискретного сигнала $u(n\Delta t)$, $n=0, 1, \dots$.

Цифровым фильтром называется цифровая система для изменения частотного спектра дискретных сигналов. При фильтрации сигналов в реальном времени (рис. 5.8) непрерывный сигнал $x(t)$ с помощью аналого-цифрового преобразователя (АЦП) преобразуется в дискретный $x(n\Delta t)$, который поступает на вход цифрового фильтра. Полученный сигнал $y(n\Delta t)$, если это необходимо, на цифро-аналоговом преобразователе (ЦАП) вновь преобразуется в аналоговый сигнал. Цифровой фильтр может быть построен на основе МПК БИС.

Цифровые фильтры имеют ряд преимуществ перед аналоговыми, построенными из резисторов, конденсаторов, индуктивностей и усилителей:

1. Нечувствительность характеристик фильтра к разбросу параметров входящих в него элементов, их временному и температурному дрейфам.

2. Малые размеры и высокая надежность работы фильтра, связанные с использованием БИС.

3. Легкость изменения параметров и характеристик цифрового фильтра, что при использовании МП осуществляется модификацией программного обеспечения или таблиц коэффициентов.

4. Возможность реализации адаптивных фильтров, т. е. фильтров с изменяющимися в процессе работы параметрами.

Основным недостатком цифровых фильтров, особенно реализованных на основе МП общего назначения (например, БИС серии КР580), является ограниченное быстродействие. Конечное время преобразования сигнала с помощью АЦП и ЦАП и особенно время, необходимое для работы программы вычисления $y(n\Delta t)$, ограничивают снизу период Δt между следующими друг за другом замерами входного сигнала и значениями выходного сигнала. Это в свою очередь ограничивает граничную частоту $f_{гр}$ сигнала $x(t)$, фильтрация которого может быть выполнена.

Рассмотрим простейший пример цифровой реализации фильтра нижних частот (рис. 5.9). Его уравнение имеет вид

$$U_{вых} + RC(dU_{вых}/dt) = U_{вх}.$$

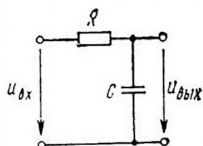


Рис. 5.9. Фильтр нижних частот

Рассматривая значения входного и выходного сигналов в дискретные моменты времени $n\Delta t$, $n=0, 1, \dots$, и заменяя производную конечной разностью

$$\frac{dU_{\text{вых}}}{dt} \approx \frac{U_{\text{вых}}[n\Delta t] - U_{\text{вых}}[(n-1)\Delta t]}{\Delta t},$$

получаем

$$\begin{aligned} U_{\text{вых}}[n\Delta t] &= \frac{1}{1 + RC/\Delta} U_{\text{вых}}[n\Delta t] + \frac{RC/\Delta}{1 + RC/\Delta} U_{\text{вых}}[(n-1)\Delta t] = \\ &= k_1 U_{\text{вых}}[n\Delta t] + k_2 U_{\text{вых}}[(n-1)\Delta t]. \end{aligned} \quad (1)$$

Реализовав зависимость (1) в виде программы для МП и используя схему (см. рис. 5.8) при $\Delta t \ll RC$, получим цифровой фильтр нижних частот.

Реально цифровые фильтры с использованием МП являются гораздо более сложными, причем один МП может быть использован для программной реализации нескольких фильтров с временным разделением входных и выходных сигналов. В общем случае уравнение цифрового фильтра имеет вид

$$y_n = \sum_{k=0}^K C_k x_{n-k} + \sum_{l=1}^L d_l u_{n-l}. \quad (2)$$

Таким же уравнением описываются передаточные функции систем автоматического управления и регулирования, представленные в дискретной форме. В общем случае уравнение (2) рассматривается как определение цифрового фильтра для обработки сигналов в реальном времени [21].

При управлении производственными процессами широко используется пропорционально-интегрально-дифференциальный (ПИД) закон регулирования. Аналоговая форма алгоритма имеет вид [2]

$$y(t) = kx(t) + \frac{k}{T_1} \int_0^t x(t) dt + kT_2 \frac{dx(t)}{dt}, \quad x(t) = u(t) - c(t),$$

где $y(t)$ — управляющее воздействие, подаваемое с регулятора на объект управления; $u(t)$ — требуемый выходной сигнал объекта; $c(t)$ — действительный выходной сигнал объекта; k — коэффициент усиления; T_1 — постоянная времени интегрального регулирования; T_2 — постоянная времени дифференциального регулирования.

Используя упрощенные формулы

$$\begin{aligned} dx(t)/dt &\approx (x_n - x_{n-1})/\Delta t; \\ \int_0^t x(t) dt &\approx \int_0^{t-\Delta t} x(t) dt + \frac{x_n + x_{n-1}}{2} \Delta t \end{aligned}$$

и соотношение

$$\int_0^{t-\Delta t} x(t) dt = \frac{T_1}{k} \left[y_{n-1} - kx_{n-1} - kT_2 \frac{x_{n-1} - x_{n-2}}{\Delta t} \right],$$

получаем закон ПИД регулирования в разностной форме:

$$y_n = y_{n-1} + k_1 x_n - k_2 x_{n-1} + k_3 x_{n-2}, \quad (3)$$

где $k_1 = k \left(1 + \frac{\Delta t}{2T_1} + \frac{T_2}{\Delta t} \right)$; $k_2 = k \left(1 - \frac{\Delta t}{2T_1} + \frac{2T_2}{\Delta t} \right)$; $k_3 = k \frac{T_2}{\Delta t}$.

Пусть масштабирование входного и выходного сигналов регулятора произведено таким образом, что x_n и y_n могут принимать значения в диапазоне 0—1023. Это соответствует десятиразрядным аналого-цифровому и цифро-аналоговому преобразователям, т. е. точности 0,1% от максимальной величины сигнала.

Для хранения в ЗУ 10-разрядных величин y_{n-1} , x_n , x_{n-1} , x_{n-2} , k_1 , k_2 , k_3 выделим по два байта (рис. 5.10). Ввод значения x_n осуществляется микропроцессором с аналого-цифрового преобразователя побайтно. После ввода каждого нового значения необходимо записать его в ЗУ (рис. 5.10), вычислить управляющее воздействие y_n , выдать его на цифро-аналоговый преобразователь, переместить величину x_n на место x_{n-1} , а x_{n-1} на место x_{n-2} . После этого микропроцессор готов к вводу нового значения x_n .

Вычисление y_n осуществляется подпрограммой FILTR. Величина y_n помещается на место y_{n-1} . В начале указатель стека содержит адрес младшего байта x_n .

;подпрограмма вычисления управляющего воздействия ПИД регулятора по формуле (3)

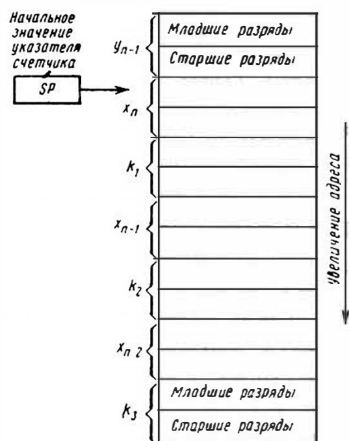


Рис. 5.10. Размещение в ЗУ коэффициентов и данных пропорционально - интегрально - дифференциального регулятора

```
FILTR:  POP H;
        POP D;
        CALL MULT
        LXI H, 0FFFFH
        DAD SP

        CALL PLUS
        POP H
        POP D
        CALL MULT
```

; вычисление $k_1 x_n$

; занесение в H, L числа -6
; в H, L адрес младшего байта
; y_{n-1}
; вычисление $y_{n-1} + kx_n$
; вычисление $k_2 x_{n-1}$

MOV A,C	; преобразование k_2x_{n-1} в до-
	; полнительный код
CMA	
MOV C,A	
MOV A,B	
CMA	
MOV B,A	
INX B	
LXI H, 0FFF6H	; занесение в H, L числа —10
DAD SP	; в H, L адрес младшего байта
	; y_{n-1}
CALL PLUS	; вычисление $y_{n-1} + k_1x_n -$
	; $-k_2x_{n-1}$
POP H	; вычисление k_3x_{n-2}
POP D	
CALL MULT	
LXI H, 0FFF2H	; занесение в H, L числа —10
DAD SP	; в H, L адрес младшего байта
	; y_{n-1}
CALL PLUS	; вычисление y_n
; пересылка x_{n-1} на место x_{n-2}	
LXI H, 0FFCH	; занесение в H, L числа —4
DAD SP	
XCHG	; в D, E адрес младшего байта
	; x_{n-2}
LXI H, 0FFF8H	; занесение в H, L числа —8
DAD SP	; в H, L адрес младшего байта
	; x_{n-1}
MOV A,M	
LDAX D	
INX H	; в H, L адрес старшего байта
	; x_{n-1}
INX D	; в D, E адрес старшего байта
	; x_{n-2}
MOV A,M	
LDAX D	
; пересылка x_n на место x_{n-1}	
XCHG	; в D, E адрес старшего байта
	; x_{n-1}
LXI H, 0FFF5H	; занесение в H, L числа —11
DAD SP	; в H, L адрес старшего бай-
	; та x_n
MOV A,M	
LDAX D	
INX H	
INX D	
MOV A,M	
LDAX D	
RET	
; подпрограмма умножения двух 10-разрядных целых положитель-	
; ных чисел	
; множимое в H—L, множитель в D—E	
; результат в B—C	
MULT:	LXI B, 0
	; обнуление B,C
	MVI A, 1
	; число повторений цикла

	PUSH PSW	; запомнить содержимое счетчи- ; ка циклов
		раз
	; далее тело цикла, повторяемое 10	
M1:	MOV A,B	; сдвиг частного произведения ; на один разряд вправо
	RAR	
	MOV B,A	
	MOV A,C	
	RAR	
	MOV C,A	
	MOV A,D	; сдвиг множителя на один раз- ; ряд вправо
	RAR	
	MOV D,A	
	MOV A,E	
	RAR	
	MOV E,A	; очередной разряд множителя ; в признаке C
	JNC M2	; переход при нулевом разряде ; множителя
	MOV A,C	; сложение множимого с част- ; ным произведением
	ADD L	
	MOV C,A	
	MOV A,B	
	ADC H	
	MOV B,A	
M2:	POP PSW	; уменьшение содержимого счет- ; чика цикла
	DCR A	
	RZ	; возврат из подпрограммы при ; нуле
	PUSH PSW	
	JMP M1	
	; подпрограмма сложения содержимого регистров B, C ; с содержимым ЗУ по адресу, расположенному в H, L	
PLUS:	MOV A,C	
	ADD M	
	MOV M,A	
	MOV A,B	
	INX H	
	ADC M	
	MOV M,A	
	RET	

Для выполнения подпрограммы FILTR при тактовой частоте 2 МГц необходимо около 2,6 мс, что соответствует частоте выдачи управляющих воздействий около 385 Гц.

Команды ассемблера микропроцессора КР580ИК80А

Команда	Длина ко- манды, байт	Число тактов	Описание	Изменяемые признаки
MOV r1, r2	1	5	Пересылка данных из регистра r2 в регистр r1	
MOV M r	1	7	Пересылка данных из регистра r в память	
MOV r, M	1	7	Пересылка данных из памяти в регистр r	
XCHG	1	4	Обмен данными между парами регистров H, L и D, E	
MVI r, (байт)	2	7	Занесение байта данных в регистр r	
MVI M, (байт)	2	10	Занесение байта данных в память	
LXI гр (два байта)	3	10	Занесение двух байтов данных в пару регистров (B, D, H, SP)	
LDAX гр	1	7	Загрузка в накопитель содержимого ячейки, косвенно адресуемых парой регистров гр (B, D)	
LDA (адрес)	3	13	Загрузка содержимого ячейки с указанным адресом в накопитель	
STAX гр	1	7	Занесение содержимого накопителя в ячейку, косвенно адресуемую парой регистров гр (B, D)	
STA (адрес)	3	13	Занесение содержимого накопителя в ячейку с указанным адресом	
LHLD (адрес)	3	16	Загрузка в регистры H, L содержимого ячеек с указанным адресом и адресом, на единицу большим	
SHLD (адрес)	3	16	Занесение содержимого регистра H, L в память	
ADD r	1	4	Сложение содержимых регистра r и накопителя	Z, S, P, C, AC
ADD M	1	7	Сложение содержимых ячейки памяти и накопителя	Z, S, P, C, AC
ADC r	1	4	Сложение содержимых регистра r и накопителя с учетом переноса C	Z, S, P, C, AC
ADC M	1	7	Сложение содержимых ячейки памяти и накопителя с учетом переноса C	Z, S, P, C, AC
SUB r	1	4	Вычитание содержимого регистра из содержимого накопителя	Z, S, P, C ¹ , AC ²
SUB M	1	7	Вычитание содержимого ячейки памяти из содержимого накопителя	Z, S, P, C ¹ , AC ²
SBB r	1	4	Вычитание содержимого регистра r из содержимого накопителя с заемом	Z, S, P, C ¹ , AC ²
SBB M	1	7	Вычитание содержимого ячейки памяти из содержимого накопителя с заемом	Z, S, P, C ¹ , AC ²
ANA r	1	4	Поразрядное И над содержимым регистра r и накопителем	Z, S, P, C=0, AC=0

Команда	Длина ко- манды, байт	Число тактов	Описание	Изменяемые признаки
ANA M	1	7	Поразрядное И над содержи- мыми ячейки памяти и накопи- теля	Z, S, P, C=0, AC=0
XRA r	1	4	Поразрядное ИСКЛЮЧАЮ- ЩЕЕ ИЛИ над содержимыми регистра r и накопителя	Z, S, P, C=0, AC=0
XRA M	1	7	Поразрядное ИСКЛЮЧАЮ- ЩЕЕ ИЛИ над содержимыми ячейки памяти и накопителя	Z, S, P, C=0, AC=0
ORA r	1	4	Поразрядное ИЛИ над содер- жимыми регистра r и накопи- теля	Z, S, P, C=0, AC=0
ORA M	1	7	Поразрядное ИЛИ над содер- жимыми ячейки памяти и нако- пителя	Z, S, P, C=0, AC=0
CMP r	1	4	Сравнение содержимых ре- гистра r и накопителя	(Z, S, P, C, AC) ¹
CMP M	1	7	Сравнение содержимых ячейки памяти и накопителя	(Z, S, P, C, AC) ¹
ADI (байт)	2	7	Сложение байта с содержи- мым накопителя	Z, S, P, C, AC
ACI (байт)	2	7	Сложение байта с содержи- мым накопителя с учетом пере- носа	Z, S, P, C, AC
SUI (байт)	2	7	Вычитание байта из содержи- мого накопителя	Z, S, P, C ¹ , AC ¹
SBI (байт)	2	7	Вычитание байта из содержи- мого накопителя с учетом заема	Z, S, P, C ¹ , AC ¹
ANI (байт)	2	7	Поразрядное И над содержи- мым накопителя и байтом	Z, S, P, C=0, AC=0
XRI (байт)	2	7	Поразрядное ИСКЛЮЧАЮ- ЩЕЕ ИЛИ над содержимым накопителя и байтом	Z, S, P, C=0, AC=0
ORI (байт)	2	7	Поразрядное ИЛИ над содер- жимым накопителя и байтом	Z, S, P, C=0, AC=0
CPI (байт)	2	7	Сравнение байта с содержи- мым накопителя	(Z, S, P, C, AC) ¹
DAD rp	1	10	Сложение содержимого пары регистров rp (B, D, H, SP) с содержимым пары регистров H, L	C
INR r	1	5	Увеличение содержимого ре- гистра r на единицу	Z, S, P, AC
INR M	1	10	Увеличение содержимого ячейки памяти на единицу	Z, S, P, AC
DCR r	1	5	Уменьшение содержимого ре- гистра r на единицу	Z, S, P, AC ¹
DCR M	1	10	Уменьшение содержимого ячейки памяти на единицу	Z, S, P, AC ¹
INX rp	1	5	Увеличение содержимого па- ры регистров rp (B, D, H, SP) на единицу	
DCX rp	1	5	Уменьшение содержимого па- ры регистров rp (B, D, H, SP) на единицу	
RLC	1	4	Циклический сдвиг содержи- мого накопителя влево	C ¹
RRC	1	4	Циклический сдвиг содержи- мого накопителя вправо	C ¹
RAL	1	4	Циклический сдвиг содержи- мого накопителя влево через перенос	C ¹

Команда	Длина ко- манды, байт	Число тактов	Описание	Изменяемые признаки
RAR	1	4	Циклический сдвиг содержи- мого накопителя вправо через перенос	C ⁴
DAA	1	4	Преобразование содержимого накопителя в двоично-десятич- ный код	Z, S, P, C, AC
CMA	1	4	Поразрядное инвертирование накопителя	
STC	1	4	Установка признака переноса C в единицу	C=1
CMC	1	4	Инвертирование признака пе- реноса C	C= $\overline{C}$
PCHL	1	5	Занесение содержимого ре- гистров H, L в счетчик команд	
JMP (адрес)	3	10	Безусловный переход по ука- занному адресу	
JC (адрес)	3	10	Переход при наличии переноса	
JNC (адрес)	3	10	Переход при отсутствии пере- носа	
JZ (адрес)	3	10	Переход при нуле	
JNZ (адрес)	3	10	Переход при отсутствии нуля	
JP (адрес)	3	10	Переход при плюсе	
JM (адрес)	3	10	Переход при минусе	
JPE (адрес)	3	10	Переход при четности	
JPO (адрес)	3	10	Переход при нечетности	
CALL (адрес)	3	17	Вызов подпрограммы	
CC (адрес)	3	11/17 ^a	Вызов подпрограммы при пе- реносе	
CNC (адрес)	3	11/17	Вызов подпрограммы при от- сутствии переноса	
CZ (адрес)	3	11/17	Вызов подпрограммы при нуле	
CNZ (адрес)	3	11/17	Вызов подпрограммы при от- сутствии нуля	
CP (адрес)	3	11/17	Вызов подпрограммы при плюсе	
CM (адрес)	3	11/17	Вызов подпрограммы при ми- нусе	
CPE (адрес)	3	11/17	Вызов подпрограммы при чет- ности	
CPO (адрес)	3	11/17	Вызов подпрограммы при не- четности	
RET	1	10	Возврат	
RC	1	5/11	Возврат при переносе	
RNC	1	5/11	Возврат при отсутствии пере- носа	
RZ	1	5/11	Возврат при нуле	
RNZ	1	5/11	Возврат при отсутствии нуля	
RP	1	5/11	Возврат при плюсе	
RM	1	5/11	Возврат при минусе	
RPE	1	5/11	Возврат при четности	
RPO	1	5/11	Возврат при нечетности	
RST (номер)	1	11	Повторный запуск с адреса 8×номер (0,8, . . . , 56)	
IN (канал)	2	10	Ввод данных из указанного канала в накопитель	
OUT (канал)	2	10	Вывод данных из накопителя в указанный канал	
PUSH гр	1	11	Занесение содержимого пары регистров гр (B, D, H, PSW) в стек	
POP гр	1	10	Выдача данных из стека в пару регистров гр (B, D, H, PSW)	(Z, S, P, C, AC) ⁶

Команда	Длина ко-манды, байт	Число тактов	Описание	Изменяемые признаки
XTHL	1	18	Обмен данными между вер- синой стека и парой регистров H, L	
SPHL	1	5	Занести в указатель стека со- держимое регистров H, L	
DI	1	4	Запретить прерывание	
EI	1	4	Разрешить прерывание	
NOP	1	4	Отсутствие операции	
HLT	1	7	Останов	

¹ Устанавливается при наличии заема в старший разряд, в противном случае сбрасывается.

² Устанавливается при наличии заема из старших четырех разрядов в младшие, в противном случае сбрасывается.

³ Z устанавливается, если содержимое регистра или байта данных равно содержимому накопителя; S и C — если больше содержимого накопителя; AC — если содержимое младших четырех разрядов регистра или байта дан-ных больше содержимого младших четырех разрядов накопителя; P — если байт разности между содержимым накопителя и содержимым регистра или байта данных содержит четное число единиц.

⁴ Состояние признака равно значению выдвигаемого из накопителя дво-ичного разряда.

⁵ В знаменателе дроби указано число тактов при выполнении условий, в числителе — при невыполнении.

⁶ По команде POP PSW признаки устанавливаются в соответствии со значением разрядов слова, запомненного в стеке, при других значениях чр признаки не изменяются.

ПРИЛОЖЕНИЕ 2

Иностранные термины, использованные при описании
отладочного комплекса MDS-800

Термин	Краткий контекстный перевод	Термин	Краткий контекстный перевод
ADD	Добавить	DELETE	Удалить
ALL FLAGS	Все разряды регистра признаков	DIRECTORY	Оглавление
ALL PINS	Все выходы микросхемы	DISPLAY	Отобразить на экран
ALL REGISTERS	Все регистры	DOUBLE	Две последовательные
ATTRIBUTE	Атрибут, признак	MEMORY	ячейки памяти
AND	Логическое И	DOUBLE	Регистровая пара
BASE	Система счисле- ния	REGISTER	Редктирование
CALL	Вызов подпрограммы	EDITION	
CHANGE	Изменить	EQUATE	Назначить
CONTINUE	Продолжать (пока)	EXECUTED	Выполнено
(WHILE)		EXIT	Выход
COPY	Копировать	FILL	Заполнить
COUNT	Считать	FLAG	Разряд регистра при- знаков
CREATE	Создать	FOREVER	Бесконечно
CYCLES	Циклы процессора	FROM	От
DEBUG	Отладка	DUMP	Вывести на экран (пе- чать)

Термин	Краткий контекстный перевод	Термин	Краткий контекстный перевод
ICE	Внутрисхемный эмулятор	OR	Логическое ИЛИ
INPUT	Вводить из канала	OUTPUT	Вызывать в канал
INSTRUCTION	Команда	PIN	Вывод микросхемы
INTO	В	QUIT	Подтверждение
ISIS	Супервизор операционной системы	RANGE	Диапазон, область
JUMP	Команда передачи управления	READ	Читать, считать
LINK	Связь	REGISTER	Регистр
LISTING	Печатаемая копия	RETURN	Возврат из подпрограммы
LOAD	Загружать	RENAME	Переименовать
LOCATE	Размещать	SAVE	Сохранить
LOOPING	Циклический	SEARCH	Поиск
MASK	Маска	STACK	Стек
MDS	Система проектирования	STEP	Шаг
MOVE	Переместить	THEN	Тогда
NAME	Имя	TO	По
NESTING	Вложение подпрограмм	VALUE	Значение
NOBREAK	Без остановки	UNGUARDED	Существующий
		UNTIL	До
		WRITTEN	Записано

Список литературы

1. Алексенко А. Г. Основы микросхемотехники. — М.: Сов. радио, 1977. — 408 с.
2. Алексенко А. Г., Галицын А. А. Микро-ЭВМ и особенности их организации. — М.: Машиностроение, 1982. — 68 с.
3. Балашов Е. П., Пузанков Д. В. Микропроцессоры и микропроцессорные системы. — М.: Радио и связь, 1981. — 326 с.
4. Бедрековский М. А., Волга В. В., Кручинкин Н. С. Микропроцессоры. — М.: Радио и связь, 1981. — 96 с.
5. Березенко А. И., Корягин Л. Н., Назарьян А. Р. Микропроцессорные комплекты повышенного быстродействия. — М.: Радио и связь, 1981. — 168 с.
6. Бушуев М. К., Галицын А. А., Шагурин И. И. Микропроцессоры и микропроцессорные наборы БИС как элементная база современных цифровых систем. — Известия вузов СССР, 1978, т. XXI, № 12, с. 36—48.
7. Васенков А. А., Коночкин Э. И., Малашевич Б. М., Шахнов В. А. Терминология в технике микропроцессорных интегральных схем и микро-ЭВМ. — Микроэлектроника и полупроводниковые приборы/ Под ред. А. А. Васенкова и Я. А. Федотова. — М.: Сов. радио, 1979, вып. 4, с. 17—29.
8. Зеленко Г. В., Иванников А. Д., Сыпчук П. П. Проектирование и отладка микропроцессорных систем. — М.: Машиностроение, 1982. — 56 с.
9. Зеленко Г. В., Кукин М. А., Панов В. В., Казанская М. В., Попов С. Н. Рабочее место проектировщика микропроцессорной аппаратуры. — Синтез управляющих устройств на основе микропроцессоров и однородных сред/ Труды II Всесоюзного семинара. — М.: Наука, 1980, с. 65—68.
10. Зеленко Г. В., Панов В. В., Попов С. Н. Процессорный модуль микро-ЭВМ. — Радио, 1983, № 2, с. 40—43.
11. Иванников А. Д. Аппаратно-программные комплексы для отладки микропроцессорных устройств. — Микроэлектроника и полупроводниковые приборы/ Под ред. А. А. Васенкова и Я. А. Федотова. — М.: Радио и связь, 1981, вып. 6, с. 62—82.
12. Иванников А. Д., Зеленко Г. В. Микропроцессорные средства. — Зарубежная электронная техника, 1979, вып. 15, с. 3—103.
13. Иванников А. Д., Старых В. А. Проектирование микропроцессорных систем. — Зарубежная электронная техника, 1980, вып. 11, с. 3—92.
14. Иванников А. Д., Сыпчук П. П. Автоматизация проектирования микропроцессорных систем. — В кн.: Автоматизация проектирования в радиоэлектронике и вычислительной технике. — М.: МДНТП им. Ф. Э. Дзержинского, 1981, с. 113—117.
15. Каган Б. М., Сташин В. В. Микропроцессоры в цифровых системах. — М.: Энергия, 1979. — 193 с.
16. Клингман Э. Проектирование микропроцессорных систем. Пер. с англ./ Под ред. С. Д. Пашкеева. — М.: Мир, 1980. — 569 с.

17. **Микропроцессорные БИС и микро-ЭВМ.** Построение и применение./ А. А. Васенков, Н. М. Воробьев, В. Л. Дшхунян и др.; Под ред. А. А. Васенкова. — М.: Сов. радио, 1980. — 280 с.

18. **Микропроцессорные комплекты интегральных схем.** Состав и структура: Справочник/ В. С. Борисов, А. А. Васенков, Б. М. Малашевич и др.; Под ред. А. А. Васенкова, В. А. Шахнова. — М.: Радио и связь, 1982. — 192 с.

19. **Прангишвили И. В.** Микропроцессоры и микро-ЭВМ. — М.: Энергия, 1979. — 232 с.

20. **Полупроводниковые запоминающие устройства и их применение/** Под ред. А. Ю. Гордонова. — М.: Радио и связь, 1981. — 344 с.

21. **Рабинер Л., Гоулд Б.** Теория и применение цифровой обработки сигналов. — М.: Мир, 1978. — 848 с.

22. **Соучек Б.** Микропроцессоры и микро-ЭВМ: Пер. с англ./ Под ред. А. И. Петренко. — М.: Сов. радио, 1979. — 520 с.

23. **Справочник по интегральным микросхемам/** Б. В. Тарабрин, С. В. Якубовский, Н. А. Барканов и др.; Под ред. Б. В. Тарабрин. — М.: Энергия, 1981. — 816 с.

24. **Хилбурн Дж., Джулич П.** Микро-ЭВМ и микропроцессоры: Пер. с англ./ Под ред. С. Д. Пашкеева. — М.: Мир, 1979. — 463 с.

25. **Gutman F.** 8080 Routine Emulates PLA Hardware. — EDN, v. 24, 1979, № 18, p. 81—82.

26. **In-Circuit Emulator/80. Microcomputer Development System. ICE-80 Operator Manual.** — Intel Corp., USA, 1976. — 120 p.

27. **ISIS-II System User's Guide.** — Intel Corp., USA, 1977. — 150 p.

28. **PL/M-80 Programming Manual.** — Intel Corp., USA, 1977. — 125 p.

29. **Shankar S., Wakhlu S.** Routine Quickly Multiplies Eight Bits. — EDN, 1980, v. 25, № 2, p. 80.

30. **Wester J. G., Simpcon W. D. et al.** Software Design for Microprocessors. — Texas Instruments Learning Center, 1976. — 286 p.

Оглавление

Предисловие	3
Введение	5
 Глава 1. Микропроцессорный комплект БИС серии КР580	
1.1. Микропроцессор КР580ИК80А	12
1.2. БИС программируемого параллельного интерфейса КР580ВВ55	21
1.3. БИС программируемого последовательного интерфейса КР580ВВ51	30
1.4. БИС программируемого контроллера прерываний КР580ВН59	44
1.5. Программируемый контроллер прямого доступа к памяти КР580ВТ57	57
1.6. БИС программируемого таймера КР580ВИ53	65
1.7. БИС программируемого контроллера видеотерминала КР580ВГ75	72
 Глава 2. Разработка программного обеспечения микропроцессорных систем	
2.1. Основные понятия в области автоматизации программирования	93
2.2. Программирование на языке ассемблера МП КР580ИК80А	96
2.3. Программирование на языке ПЛ/М-80	115
2.4. Организация разработки программного обеспечения микропроцессорных систем	132
2.5. КСИ-1 — кросс-система разработки программного обеспечения на мини-ЭВМ СМ-4	138
 Глава 3. Проектирование основных блоков микропроцессорной аппаратуры	
3.1. Процессорный блок на основе микропроцессора КР580ИК80А	144
3.2. Проектирование блоков ЗУ микропроцессорных систем	156
3.3. Использование БИС программируемого параллельного интерфейса КР580ВВ55 для сопряжения микропроцессорной системы с устройствами ввода — вывода	
3.4. Сопряжение микропроцессорной системы с бытовым телефоном	
 Глава 4. Аппаратно-программные комплексы микропроцессорных систем	
4.1. Классификация отладочных комплексов	

4.2. Аппаратно-программный отладочный комплекс Intellec MDS-800	190
4.3. Отладка объектных программ с использованием программы Монитор	194
4.4. Дисковая операционная система ISIS-II	201
4.5. Использование внутрисхемного эмулятора ICE-80	223

Глава 5. Примеры микропроцессорных систем на основе комплекта БИС серии КР580

5.1. Проектирование микропроцессорной системы управления чертежным графическим автоматом	238
5.2. Микропроцессорная реализация цифровых фильтров	256

Приложение 1. Команды ассемблера микропроцессора КР580ИК80А	262
---	-----

Приложение 2. Иностранные термины, использованные при описании отладочного комплекса MDS-800	265
--	-----

Список литературы	267
-----------------------------	-----

**Андрей Геннадьевич Алексенко, Алексей Александрович Галицын,
Александр Дмитриевич Иванников**

**ПРОЕКТИРОВАНИЕ РАДИОЭЛЕКТРОННОЙ АППАРАТУРЫ
НА МИКРОПРОЦЕССОРАХ: ПРОГРАММИРОВАНИЕ,
ТИПОВЫЕ РЕШЕНИЯ, МЕТОДЫ ОТЛАДКИ**

**Редактор М. М. Лисина
Художник Л. В. Брылев
Художественный редактор Н. С. Шенн
Технический редактор Г. З. Кузнецова
Корректоры Т. В. Жукова, Е. В. Вязова**

ИБ № 286

Сдано в набор 03.06.83	Подписано в печать 18.10.83	Т-20027
Формат 84×108 ¹ / ₃₂	Бумага типографская № 3	Гарнитура литературная
Печать высокая	Усл. печ. л. 14,28	Усл. кр.-отт. 14,28
Тираж 40 000 экз.	Изд. № 19953	Зак. № 3174
		Уч.-изд. л. 14,86
		Цена 1 р. 10 к.

Издательство «Радио и связь», 101000 Москва, Почтамт, а/я 693

Ордена Октябрьской Революции и ордена Трудового Красного Знамени
Первая Образцовая типография имени А. А. Жданова Союзполиграфпрома
при Государственном комитете СССР по делам издательств, полиграфии и
книжной торговли. Москва, М-54, Валовая, 28.

В МАГАЗИНЕ № 55 г. ЛЕНИНГРАДА

ИМЕЮТСЯ В ПРОДАЖЕ СЛЕДУЮЩИЕ КНИГИ:

Березин А. С., Мочалкина О. Р. Технология и конструирование интегральных микросхем.

Подробно рассмотрены современные технологические методы создания полупроводниковых и гибридных интегральных микросхем. Дается описание отдельных технологических процессов и приводятся полные типовые технологические циклы изготовления различных микросхем. Рассмотрены методы конструирования интегральных микросхем. Книга является продолжением учебных пособий: «Основы микроэлектроники» (М.: Сов. радио, 1980) и «Микросхемотехника» (М.: Радио и связь, 1982), изданных под общей редакцией проф. И. П. Степаненко.

Для студентов радиотехнических и радиофизических специальностей вузов.

Моделирование и оптимизация на ЭВМ радиоэлектронных устройств /Под ред. З. М. Бененсона.

Справочное пособие посвящено комплексному решению задач анализа и оптимизации радиоэлектронных устройств (РЭУ) различного назначения в системе автоматизации проектирования (САПР). Рассмотрены современные алгоритмы решения этих задач. Значительное внимание уделено универсальности, точности и экономности алгоритмов. Изложены методы макроmodellирования интегральных микросхем и функциональных узлов радиоэлектронной аппаратуры (РЭА). Рассмотрены смежные задачи теплового расчета РЭА и моделирования магнитных цепей. Приведены языки описания РЭУ. Изложены вопросы практического применения САПР.

Для инженеров-разработчиков РЭА, систем управления и САПР. Может быть полезна студентам вузов.

Адрес магазина: 197198 г. Ленинград ПС, Большой проспект, 34.

