МИКРОПРОЦЕССОРЫ

3

СРЕДСТВА ОТЛАДКИ, ЛАБОРАТОРНЫЙ ПРАКТИКУМ И ЗАДАЧНИК

МИКРОПРОЦЕССОРЫ

МИКРОПРОЦЕССОРЫ

В ТРЕХ КНИГАХ

1 АРХИТЕКТУРА И ПРОЕКТИРОВАНИЕ МИКРО-ЭВМ. ОРГАНИЗАЦИЯ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

2 СРЕДСТВА СОПРЯЖЕНИЯ. КОНТРОЛИРУЮЩИЕ И ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ

3 СРЕДСТВА ОТЛАДКИ, ЛАБОРАТОРНЫЙ ПРАКТИКУМ И ЗАДАЧНИК

СРЕДСТВА ОТЛАДКИ, ЛАБОРАТОРНЫЙ ПРАКТИКУМ И ЗАДАЧНИК

Под редакцией
члена-корреспондента
АН СССР Л. Н. ПРЕСНУХИНА

Допущено Министерством высшего
и среднего специального образования СССР
в качестве учебника
для студентов высших
технических учебных заведений

МОСКВА «ВЫСШАЯ ШКОЛА» 1986

ББК 32.9732
М 59
УДК 681.322

Н. В. ВОРОБЬЕВ, В. Л. ГОРБУНОВ, А. В. ГОРЯЧЕВ, В. Р. ГОРОВОЙ, А. Е. КОСТИН, П. В. НЕСТЕРОВ, Д. И. ПАНФИЛОВ, Д. Л. ПРЕСНУХИН, В. Д. ВЕРНЕР, В. Ф. ШАНЬГИН, Г. И. ФРОЛОВ, А. А. ШИШКЕВИЧ

Рецензент — кафедра автоматизированных систем управления Московского высшего технического училища им. Н. Э. Баумана (зав. кафедрой — проф. *В. Н. Четвериков)*

М59 **Микропроцессоры:** В 3-х кн. Кн. 3. Средства отладки, лабораторный практикум и задачник: Учеб. для втузов/Н. В. Воробьев, В. Л. Горбунов, А. В. Горячев и др.; Под ред. Л. Н. Преснухина.— М.: Высш. шк., 1986.— 351 с.: ил.

Учебник является третьей книгой из серии, выпускаемой под общим названием «Микропроцессоры». В нем определены задачи и проблемы, возникающие при практическом использовании вычислительных систем на базе микропроцессоров; дана методика применения технических средств отладки и диагностирования микропроцессорных устройств; рассмотрены вопросы привития практических навыков в работе с микропроцессорной техникой в рамках лабораторных работ на выпускаемых промышленностью учебных микро-ЭВМ.

М $\frac{2405000000-467}{001(01)-86}$ 140—86

ББК 32.973.2
6Ф73

ВВЕДЕНИЕ

Данная книга является третьей из серии учебников под общим названием «Микропроцессоры». Она посвящена вопросам отладки, обнаружения, отыскания неисправностей в микропроцессорных системах, содержит лабораторный практикум, позволяющий привить практический навык работы с микропроцессорными системами, и теоретические задачи на материал, излагаемый во всех трех книгах учебника.

Несмотря на достигнутые в настоящее время успехи в области автоматизации проектирования, процесс создания дискретных систем носит творческий характер. Поэтому от того, насколько тщательно проведена проверка правильности поведения системы, зависит качество проектируемой системы.

Впервые в отечественной литературе авторами делается попытка описать методы и средства отладки микропроцессорных систем. Пока нет однозначного определения термина «отладка». Под этим термином авторы понимают процесс обнаружения неправильного поведения и локализации источника неправильного поведения системы или ее модели на этапе проектирования системы. Процесс отладки спроектированного электронного оборудования эффективно может быть организован на базе микропроцессорных вычислительных средств. Микропроцессоры вызвали к жизни новые методы и средства проектирования: логические анализаторы, генераторы слов, оценочные и отладочные комплексы, комплексы развития. В книге приводятся принципы построения имеющихся средств отладки, примеры пользования этими средствами; описывается лабораторное оборудование и излагается методика проведения лабораторных работ. Основу лабораторного оборудования составляют учебные микро-ЭВМ, разработанные в Московском ордена Трудового Красного Знамени институте электронной техники. Учебные микро-ЭВМ реализуются на микропроцессорных комплектах

серий К580, К589, К588, К1801 и отражают особенности функционирования основной номенклатуры микропроцессорных БИС.

Режимы работы, обмена и преобразования информации в микро-ЭВМ в основном осуществляются программными средствами. Поэтому при проведении лабораторного практикума основное внимание уделяется исследованию программных методов организации обмена и преобразования информации. Изучение программно-аппаратурных методов организации обмена информации целесообразно в тех случаях, когда схемотехническое и программное обеспечения неразрывно связаны друг с другом.

Приведенное в учебнике описание схемотехнических блоков и программного обеспечения позволяет читателю самостоятельно проектировать микро-ЭВМ и использовать их в технических разработках и для обучения неспециалистов в области вычислительной техники.

Примеры и задачи охватывают тематику трех книг. Многие задачи и вопросы имеют подробное решение. Некоторые задачи предлагается решить читателю самостоятельно.

В пособии помещен предметный указатель, охватывающий материал трех книг, который окажет существенную помощь читателю при нахождении конкретных вопросов.

Материал между авторами распределен следующим образом: гл. 1—7 написаны В. Р. Горовым, гл. 8 — Д. И. Панфиловым, предисловие, введение и гл. 9 — В. Л. Горбуновым. В написании задачника принимали участие все авторы.

Раздел 1. СРЕДСТВА ОТЛАДКИ

Микропроцессорные системы потребовали разработки принципиально новых методов и средств их отладки. От используемых методов и средств отладки зависят сроки разработки микропроцессорных систем и качество проекта. Практика показывает, что не выявленные на этапе проектирования ошибки проекта дорого обходятся пользователям систем. В предлагаемом разделе учебника рассматриваются существующие методы и средства отладки на различных уровнях представления: системном, программном, логическом, схемном; даются понятия проектных неисправностей, функций средств отладки, классификация средств отладки; показаны примеры применения отдельных средств.

Глава 1

Основные понятия

§ 1.1. Микропроцессорная система

Упрощенно микропроцессорную систему можно представить состоящей из микропроцессора *МП*, запоминающего устройства *ЗУ* и устройства управления вводом — выводом *УУВВ* (рис. 1.1). Микропроцессор выполняет основные функции процессора, но в отличие от последнего реализован на больших интегральных схемах (БИС). В *ЗУ* хранятся программы. Данные, или переменные, могут также храниться в *ЗУ* или поступать на входы устройства управления вводом — выводом. Запоминающее устройство может состоять из оперативного запоминающего устройства (ОЗУ), изменение содержимого которого доступно микропроцессору; постоянного запоминающего устройства (ПЗУ), «заполнение» которого осуществляется при изготовлении; программируемого ПЗУ (ППЗУ), однажды программируемого пользователем, и репрограммируемого ПЗУ (РППЗУ), содержимое которого изменяется с помощью программируемого устройства — программатора.

В микропроцессорных системах используется принцип программируемой логики или цикл управления фон-Неймана. По принципу программируемой логики функции реализуются последовательно, команда за командой, в отличие от комбинационной логики, где значения функций определяются комбинацией значений входов. В связи с этим микропроцессорные системы уступают по быстродействию системам, реализованным на произвольной логике. Цикл управления фон-Неймана состоит из нескольких фаз [1]. Так, в фазе выборки команды содержимое ячейки памяти, адресуемой счетчиком команд микропроцессора, выбирается из памяти и помещается в регистр команд микропроцессора. В следующей фазе код команды поступает с регистра команд в дешифратор команд, где преобразуется в систему управляющих сигналов. После этого содержимое счетчика команд увеличивается на 1; последней фазой является выполнение команды; после выполнения команды наступает опять фаза выборки команды.

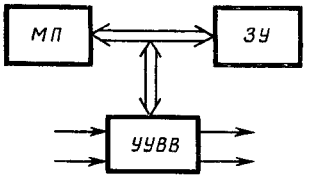


Рис. 1.1. Структура микропроцессорной системы

В зависимости от содержимого памяти микропроцессорная система выполняет ту или иную функцию. Поменяв в системе одно ПЗУ на ПЗУ с другим содержанием или изменив содержание РППЗУ системы, можно изменить функции системы. Например, если в системе, предназначенной для перевода с русского языка на английский, поменять ПЗУ, то эта система перестает переводить с русского языка на английский и начинает переводить, допустим, с японского языка на английский.

На сегодняшний день имеется несколько десятков различных типов микропроцессорных наборов и однокристальных микро-ЭВМ, отличающихся разрядностью, системой команд, быстродействием, потребляемой мощностью, номиналами питания и т. д. В качестве примера микропроцессорной системы рассмотрим одноплатную микро-ЭВМ.

Микро-ЭВМ «Электроника Н МС 11100.1». Эта микро-ЭВМ построена на базе микропроцессорного комплекта К1801, предназначена для встраивания в аппаратуру потребителя и может применяться в составе технологи-

ческого оборудования, в контрольно-измерительных и испытательных комплексах, в системах обработки цифровой информации общего назначения. Микро-ЭВМ унифицирована по конструктивному исполнению, системе команд, организации и интерфейсу магистрали с ЭВМ типа «Электроника-60» и позволяет наращивать технические возможности за счет подключения дополнительных типов функциональных и унифицированных устройств, разработанных пользователем.

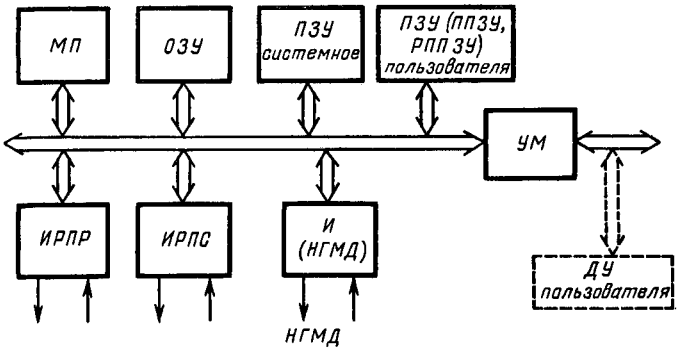


Рис. 1.2. Структура микро-ЭВМ «Электроника Н МС 11100.1»

На рис. 1.2 представлена структура микро-ЭВМ «Электроника Н МС 11100.1». В состав микро-ЭВМ входят следующие устройства: микропроцессор *МП*; *ОЗУ* емкостью 28К 16-разрядных слов; *ПЗУ* системное емкостью 4К 16-разрядных слов; контактирующее устройство *ПЗУ*, или *ППЗУ*, или *РППЗУ* пользователя; устройство интерфейса *И* накопителя на гибких магнитных дисках *НГМД*; устройство интерфейса для радиального подключения устройств с параллельной передачей информации *ИРПР*; устройство интерфейса для радиального подключения устройств с последовательной передачей информации *ИРПС*; умощнитель магистрали *УМ*.

Микропроцессор выполнен на основе БИС типа К1801 ВМ1, оперативное запоминающее устройство — на основе БИС К565 РУ3 и К1801 ВП1-030, К1801 ВП1-034, системное ПЗУ — на основе микросхемы К1801 РЕ1-000 и предназначено для хранения программ режима началь-

ного пуска микро-ЭВМ, пультового режима работы микро-ЭВМ, начального загрузчика с НГМД.

Устройство *ИРПР* выполнено на основе микросхемы К1801 ВП1-033, устройство *ИРПС* — на основе микросхемы К1801 ВП1-035, устройство интерфейса *НГМД* — на основе микросхемы К1801 ВП1-033 и предназначено для связи с НГМД «Электроника ГМД-7012». Контактирующее устройство представляет собой розетку типа РС24-7 и предназначено для установки ЗУ пользователя, например ПЗУ типа одной микросхемы К1801 РЕ1 емкостью 4К слов с программами пользователя. Запоминающее устройство пользователя может быть установлено в адресное пространство микро-ЭВМ вместо любого отключенного ОЗУ (4К 16-разрядных слов).

Умощнитель магистрали *УМ* выполнен на микросхемах К531 АП2П и предназначен для подключения дополнительных устройств пользователя *ДУ*.

Питание микро-ЭВМ осуществляется от внешних источников постоянного тока с номинальными значениями напряжений $+5$ и $+12$ В. Мощность потребления микро-ЭВМ от источника $+5$ W не более 12,6 Вт (ток не более 2,4 А); от источника $+12$ W — не более 2,5 Вт (ток не более 0,2 А). Габаритные размеры микро-ЭВМ $252\times$ $\times 296\times 12$ мм. Быстродействие микро-ЭВМ при выполнении команд типа «Сложение»: при регистровом методе адресации — (400 ± 100) тыс. операций/с; при косвенно-регистровом методе адресации — (180 ± 40) тыс. операций/с.

§ 1.2. Уровни представления микропроцессорной системы

Микропроцессорная система может быть описана на различных уровнях абстрактного представления.

Существующую микропроцессорную систему можно описать на любом известном уровне представления, но в начальной стадии проектирования ее можно описать только на концептуальном уровне представления. В процессе разработки системы происходит переход от одного уровня ее представления к другому, более детальному. Каждая абстракция несет в себе только информацию, которая соответствует данному уровню, и не содержит каких-либо сведений относительно более низких уровней. Микропроцессорная система может быть описана, например,

на одном из следующих уровней абстрактного представления [2—5]: 1) «черный ящик», 2) структурный; 3) программный; 4) логический; 5) схемный.

Примечание. Выше перечислены не все уровни представления микропроцессорных систем. Так, для определения технологических дефектов при производстве интегральных схем микропроцессорную систему необходимо рассматривать на уровне топологии кристалла.

На уровне «черного ящика» микропроцессорная система описывается внешними спецификациями; перечисляются внешние характеристики.

Структурный уровень создается компонентами микропроцессорной системы — микропроцессорами, запоминающими устройствами, устройствами ввода — вывода, внешними запоминающими устройствами, каналами связи. Микропроцессорная система на этом уровне описывается функциями отдельных устройств и их взаимосвязью, информационными потоками.

Программный уровень разделяется на два подуровня: *команд процессора* и *языковой*. Микропроцессорная система на этом уровне интерпретируется как последовательность операторов или команд, вызывающих то или иное действие над некоторой структурой данных.

Логический уровень присущ исключительно дискретным системам. На этом уровне выделяются два подуровня: переключательных схем и регистровых пересылок. Подуровень *переключательных схем* образуется вентилями и построенными на их основе операторами обработки данных. Переключательные схемы подразделяются на комбинационные и последовательностные; первые в отличие от последних не содержат запоминающих элементов. Поведение системы на этом уровне описывается алгеброй логики, моделью конечного автомата, вход — выходными последовательностями I и 0. Комбинационные схемы представляются таблицей истинности, в которой каждому входному набору значений сигналов ставится в соответствие набор значений сигналов на выходах. Последовательностные схемы могут описываться диаграммами или таблицами переходов — выходов, в которых определены взаимно однозначные соответствия между входами схемы, внутренними состояниями (комбинациями значений элементов памяти) и выходами. Подуровень *регистровых пересылок* характеризуется более высокой степенью абстрагирования и представляет собой описание регистров

и передачу данных между ними. Он включает в себя две части: информационную и управляющую. Информационная часть образуется регистрами, операторами и путями передачи данных. Управляющая часть обеспечивает зависящие от времени сигналы, инициирующие пересылку данных между регистрами.

Схемный уровень образуется резисторами и конденсаторами. Показателями поведения системы на этом уровне служат напряжение и ток, представляемые в функции времени или частоты. Этот уровень описания дискретной системы широко используется в описаниях аналоговых систем и не является ни наинизшим из возможных, ни достаточным для полной характеристики системы.

Для логического уровня представления универсальной формой описания системы являются временны́е диаграммы, которые широко используются разработчиками на практике. На рис. 1.3, *а* изображена временна́я диаграмма процедуры ввода данных магистрали, а на рис. 1.3, *б* — временна́я диаграмма процедуры вывода данных магистрали микро-ЭВМ «Электроника Н МС 11100.1». На диаграммах приняты следующие обозначения: *1* и *2*— передаваемый и принимаемый сигналы; * — значение сигнала не определено; *ДА* — сигналы данных, адреса; *СИА* — сигнал синхронизации активного устройства (указывает на то, что на магистрали установлен адрес); *СИП* — сигнал синхронизации пассивного устройства (информирует активное устройство о том, что данные приняты); *Ввод* — сигнал готовности приема данных активным устройством; *Вывод* — сигнал сопровождения данных (указывает на то, что на магистрали установлены данные); *ВУ* — сигнал выбора внешнего устройства; *Байт* — сигнал вывода байта. Стрелки на диаграмме указывают на причинно-следственную связь изменений значений сигналов. Как адрес, так и данные (слова и байты) передаются по одним и тем же 16 линиям магистрали данных. Любой цикл взаимодействия на магистрали начинается с адресации пассивного устройства.

Для описания микропроцессорных систем на уровне регистровых пересылок существуют языки, наиболее распространенными из которых являются BDL/CS [5], ISPL, DDL, CDL, AHPL [4].

Описание 8-битового мультиплексора с использованием языка ISPL имеет следующий вид:

на регистровом уровне абстракции [3]

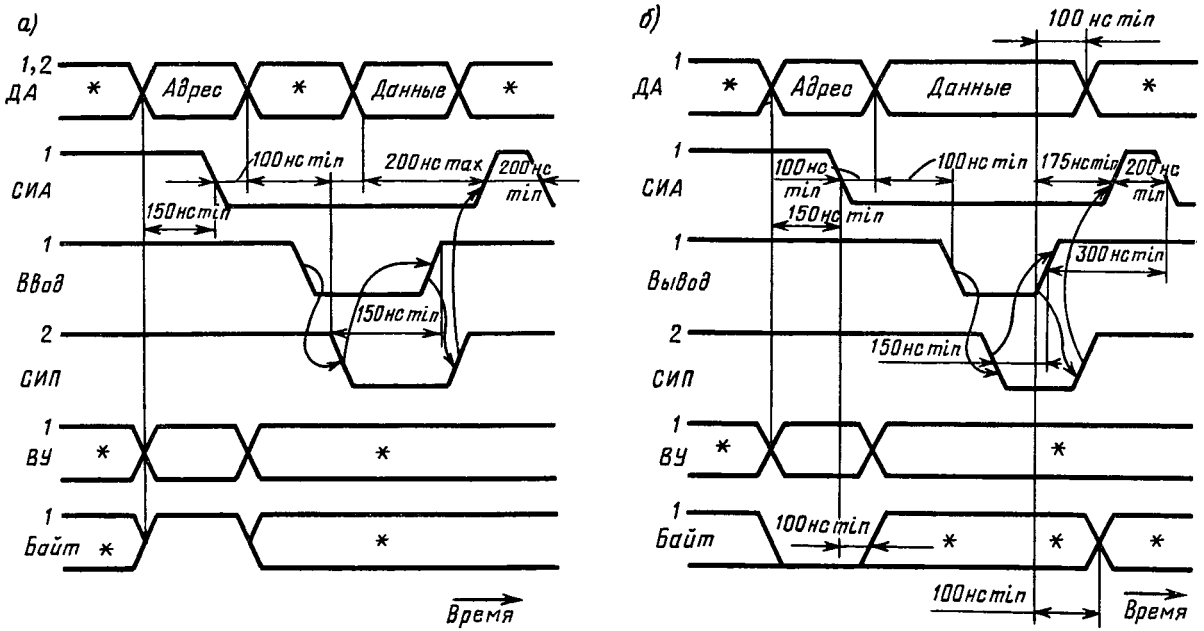


Рис. 1.3. Временны́е диаграммы процедуры ввода (*а*) и вывода (*б*) данных магистрали микро-ЭВМ «Электроника Н МС 11100.1»

```
mult (a<7:0> , b<7:0>) <15:0> :=
      BEGIN
      mult — 0 NEXT
      temp<15:0> —"00 @ b NEXT
      count — 8 NEXT
      run: =
          BEGIN
      IF a<0> =>
      mult — mult+temp NEXT
      a — a SRO 1 NEXT
      temp—temp SLO 1 NEXT
      count—count—1 NEXT
      IF (count GTR 0) =>
      RESTART run
      END
    END
```

а на программном уровне —

```
      mult (a<7:0> , b<7:0> ) <15:0> :=
          BEGIN
          mult — a*b
      END
```

§ 1.3. Ошибки, неисправности, дефекты

В жизненном цикле микропроцессорной системы, как любой дискретной системы, выделяются три стадии: проектирование, изготовление и эксплуатация. Каждая из стадий подразделяется на несколько фаз. Для каждой фазы существует вероятность возникновения конструктивных или физических неисправностей, приводящих систему в неработоспособное состояние. Поэтому на каждой фазе необходимы процедуры тестового контроля, направленные на обнаружение и локализацию неисправностей. Процедура тестового контроля может быть определена как проведение экспериментов с «черным ящиком». Дискретная система любой сложности или часть такой системы может рассматриваться как «черный ящик» с множеством входов и выходов. Правильность функционирования этого «черного ящика» должна устанавливаться путем подачи входных сигналов и наблюдения ответных выходных сигналов системы. В тех случаях, когда поведение «черного ящика» отличается от нормального, характеризуемого его спецификацией или представлениями человека, говорят о наличии ошибки. Ошибка вызывается некоторой неисправностью, представляющей собой некорректное состояние внутри «черного ящика». Неисправности классифицируют в соответствии с их причинами [2, 6]: физическая, если причиной ее служат либо дефекты элементов, либо физическое воздействие окру-

жающей среды; субъективная (внесенная, нефизическая), если ее причиной служат ошибки проектирования, неправильный монтаж элементов и грубые ошибки оператора. *Физические неисправности* — непредусмотренные, нежелательные изменения значения одной или нескольких логических переменных в системе. *Субъективные (внесенные) неисправности* — конкретные проявления недостатков программного и аппаратурного обеспечения и неправильных действий оператора, имеющих место при выполнении дискретной системой предписанных спецификацией действий.

Примечание. К субъективным неисправностям не относят физические неисправности, вызванные действиями людей (повреждения соединений, закорачивания выводов и т. п.). Проявления таких физических неисправностей аналогичны проявлениям неисправностей, вызванных естественными причинами; по этим соображениям они отнесены к тому же классу, что и другие физические неисправности.

Под субъективными неисправностями подразумеваются неисправности нефизические, вызванные недостатками различных схем, конструкций, программ, средств эксплуатации — компиляторов, ассемблеров, программ автоматизации проектирования, инструкций по эксплуатации, процедур и средств контроля и т. д. Субъективные неисправности делят на проектные и интерактивные.

Проектные неисправности вызваны недостатками, вносимыми в систему на различных стадиях реализации исходного задания — при структурном проектировании, разработке алгоритмов, написании программ, трансляции в машинный код, детальном логическом и техническом проектировании, а также при последующих модификациях аппаратурного и программного обеспечения. *Интерактивные неисправности* возникают, когда в процессе работы, или технического обслуживания, или отработки системы оператор вводит в систему через интерфейс человек — машина ложную информацию, не соответствующую текущему состоянию системы. Как правило, это происходит в результате непонимания инструкции для оператора или вследствие неточностей ввода информации.

Ошибка — проявление неисправности (физической или субъективной). В зависимости от уровня иерархической структуры системы термин «ошибка» может иметь различный смысл. Так, для дискретного устройства он означает появление неверных двоичных сигналов («0»

вместо «1» и «1» вместо «0»); для программы ошибка означает отклонение поведения программы от заданного, приводящее к выдаче неверных результатов.

Следует четко разграничивать понятия «ошибка» и «неисправность». Неисправность может приводить или не приводить к ошибке в зависимости от состояния системы. В то же время возникновение ошибки обязательно говорит о существовании какой-то неисправности. Одна и та же ошибка может быть вызвана множеством неисправностей, а одна неисправность может служить причиной целого ряда ошибок. Например, если триггер, предназначенный для хранения кода переполнения разрядной сетки ЭВМ, вследствие неисправности все время находится в состоянии «0», то ошибки из-за неисправности не будет до тех пор, пока в процессе вычислений реально не возникнет арифметическое переполнение, при котором триггер останется в состоянии «0» вместо перехода в состояние «1». Однако даже и в этом случае такая ошибка процессора не обязательно приведет к ошибке на программном уровне, если в программе условие переполнения не проверяется, и, следовательно, ни с какой стороны не влияет на ее дальнейшее поведение.

Дефекты — физические изменения параметров компонентов системы, выходящие за допустимые пределы. Их называют *сбоями*, если они носят временный характер, и *отказами*, если они постоянны.

Существует такая причинно-следственная связь: 1) дефект, представляющий собой изменение в значениях параметров компонентов, вызывает неисправность, т. е. отклонение от заданного значения (значений) логической переменной (переменных) в точке дефекта; 2) неисправность приводит к подаче неверных логических значений на вход (входы) остальной части системы и может вызывать ошибки, проявляющиеся при последующей работе других исправных логических схем; 3) ошибки приводят или к появлению неправильных результатов, или к отклонению от нормального хода исполнения программы.

§ 1.4. Отладка

О правильности функционирования микропроцессорной системы на уровне «черного ящика» с полностью неизвестной внутренней структурой можно говорить лишь тогда, когда произведены ее испытания, в ходе которых реализованы все возможные комбинации входных воздей-

ствий, и в каждом случае проверена корректность ответных реакций. Однако исчерпывающее тестирование имеет практический смысл лишь для простейших элементов систем. Следствием этого является тот факт, что ошибки проектирования встречаются при эксплуатации систем и для достаточно сложных систем нельзя утверждать об их отсутствии на любой стадии жизни системы. В основе почти всех методов испытаний лежит та или иная гипотетическая модель неисправностей, первоисточником которой служат неисправности, встречающиеся в практике. В соответствии с моделью в рамках каждого метода предпринимаются попытки создания тестовых наборов, которые могли бы обеспечить удовлетворительное выявление моделируемых неисправностей. Любой метод тестирования хорош ровно настолько, насколько правильна лежащая в его основе модель неисправности.

Важным моментом является правильный выбор соотношения между степенью общности модели неисправности и стоимостью и степенью сложности формирования и прогона тестов, ориентированных на моделируемые неисправности. Чем конкретнее модель, тем легче создать для нее систему тестов, но тем выше вероятность того, что неисправность останется незамеченной. Если же модель неисправностей излишне общая, то из-за комбинаторного возрастания числа необходимых тестовых наборов и (или) времени вычислений, требуемого для работы алгоритмов формирования тестов, она станет непрактичной и пригодной только для несложных систем.

Обнаружение ошибки и диагностика неисправности. Дефект не может быть обнаружен до тех пор, пока не будут созданы условия для возникновения из-за него неисправности, результат которой должен быть в свою очередь передан на выход испытуемого объекта, для того чтобы сделать неисправность наблюдаемой. Метод испытаний должен позволить генерировать тесты, ставящие испытуемый объект в такие условия, при которых моделируемые неисправности проявляли бы себя в виде обнаруживаемых ошибок. Если испытуемый объект предназначен для эксплуатации, то при обнаружении ошибки необходимо произвести локализацию неисправности с целью ее устранения путем ремонта или усовершенствования испытуемого объекта.

Диагностика неисправности — процесс определения причины появления ошибки по результатам тестирования. *Отладка* — процесс обнаружения ошибок и опреде-

ление источников их появления по результатам тестирования при проектировании микропроцессорных систем. Средствами отладки являются приборы, комплексы и программы.

Примечание. Иногда под отладкой понимают обнаружение, локализацию и устранение неисправностей.

Точность, с которой тот или иной тест локализует неисправности, называется его *разрешающей способностью.* Требуемая разрешающая способность определяется конкретными целями испытаний. Например, при испытаниях аппаратуры в процессе эксплуатации для ее ремонта часто необходимо установить, в каком сменном блоке изделия имеется неисправность. В заводских условиях желательно осуществлять диагностику неисправности вплоть до уровня наименьшего заменяемого элемента, чтобы минимизировать стоимость ремонта. В лабораторных условиях в процессе отладки опытного образца необходимо определять природу неисправности (физического или нефизического происхождения). В случае возникновения и проявления дефекта требуется локализовать место неисправности с точностью до заменяемого элемента, а при проявлении субъективной неисправности — с точностью до уровня представления (программного, схемного, логического и т. д.), на котором была внесена неисправность, и места.

Так как процесс проектирования микропроцессорной системы содержит неформализуемые этапы, то отладка системы предполагает участие человека.

Свойство контролепригодности системы. Успех отладки зависит от того, как спроектирована система, предусмотрены ли свойства, делающие ее удобной для отладки, а также от средств, используемых при отладке. Для проведения отладки проектируемая микропроцессорная система должна обладать свойствами [2] управляемости, наблюдаемости, предсказуемости.

Управляемость — свойство системы, при котором ее поведение поддается управлению, т. е. имеется возможность остановить функционирование системы в определенном состоянии, запустить систему. *Наблюдаемость* — свойство системы, позволяющее проследить за поведением системы, сменой ее внутренних состояний. *Предсказуемость* — свойство системы, позволяющее установить систему в состояние, из которого все последующие состояния могут быть предсказаны.

Функции средств отладки. Сроки и качество отладки системы зависят от средств отладки. Чем совершеннее приборы, имеющиеся в распоряжении инженера-разработчика, тем скорее можно начать отладку аппаратуры и программ и тем быстрее обнаружить ошибки, локализовать источники, устранение которых обойдется дороже на более позднем этапе проектирования.

Средства отладки должны: 1) управлять поведением системы или (и) ее модели на различных уровнях абстрактного представления; 2) собирать информацию о поведении системы или (и) ее модели, обрабатывать и представлять на различных уровнях абстракции; 3) преобразовывать системы, придавать им свойства контролепригодности; 4) моделировать поведение внешней среды проектируемой системы и ее составных частей.

Под *управлением поведением* системы или ее модели понимаются определение и подача входных воздействий для запуска или останова системы или ее модели, для перевода в конкретное состояние последних. Чтобы определить место субъективной неисправности, которая может быть внесена на любой стадии проектирования, необходимо уметь собирать информацию о поведении системы и представлять ее в тех формах, которые приняты для данного проекта. Например, это могут быть временны́е диаграммы, принципиальные электрические схемы, язык регистровых передач, ассемблер и др.

В общем случае нельзя локализовать источник ошибки проектируемой системы, имея информацию о поведении системы только на ее внешних выводах, поэтому проектируемую систему преобразовывают. Например, прежде чем изготовлять однокристальную микро-ЭВМ с теми или иными «зашивками» ПЗУ, программы отлаживают на эмуляционном кристалле, у которого магистраль выведена на внешние контакты и вместо ПЗУ установлено ОЗУ.

Глава 2
ТРЕБОВАНИЯ К СРЕДСТВАМ ОТЛАДКИ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

§ 2.1. Микропроцессорная система — объект отладки

Микропроцессорные системы имеют некоторые особенности, влияющие на методы и средства отладки: свойство «разумности»; магистральная организация; технология

изготовления; число контрольных точек; временна́я зависимость; многопроцессорная и многомашинная структуры.

Свойство «разумности». Это свойство позволяет использовать способ самотестирования для микропроцессорных систем, так как заложенный в систему принцип программируемой логики дает возможность ей генерировать входные воздействия на отдельные составные части и тем самым проверять работоспособность всей системы.

Магистральная организация. Взаимодействие между устройствами микропроцессорной системы осуществляется по магистралям адресов (МА), данных (МД) и управ-

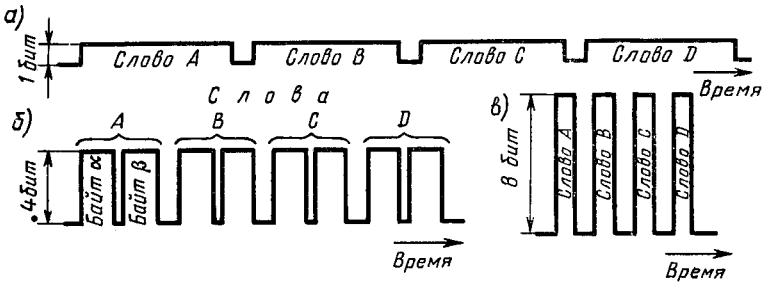


Рис. 2.1. Форматы данных

ления (МУ). Информация может передаваться по магистралям последовательным или параллельным способом. На рис. 2.1, *а* — *в* показаны форматы данных: поразрядный, по 4 бит и пословный, т. е. одно и то же слово, имеющее длину 8 бит, может быть организовано в восемь последовательных битов, или в два последовательных слога по четыре параллельных бита, или в восемь параллельных битов. На рис. 2.2, *а* — *в* приведены временны́е диаграммы передачи сообщения «DATA DOMAIN» в коде ASCII; показана передача этого сообщения по битам, по 4 бит и по словам [7].

При параллельном способе передачи информации важным параметром является разрядность. Так, имеются системы, у которых адреса передаются по 16, 18, 22 линиям (подмагистралям), а данные — по 8, 16, 32 линиям. Взаимодействие по магистралям может быть *синхронным* или *асинхронным*. Так, в одноплатной микро-ЭВМ «Электроника Н МС 11100.1» адрес и данные передаются синхронным способом, а взаимодействие на магистрали управления — асинхронное.

Взаимодействие между устройствами по магистралям требует от средств отладки умения наблюдать одновременно за всеми линиями магистрали данных, или (и) управления, или (и) адресов, а также умения управлять одновременно всеми этими линиями, реализовывать всевозможные интерфейсы. Вследствие этого существенным параметром средств отладки микропроцессорных систем является число входных и выходных каналов.

При магистральной организации имеет место *мультиплексирование*, т. е. по одним и тем же линиям могут передаваться, например, адрес и данные, как это имеет

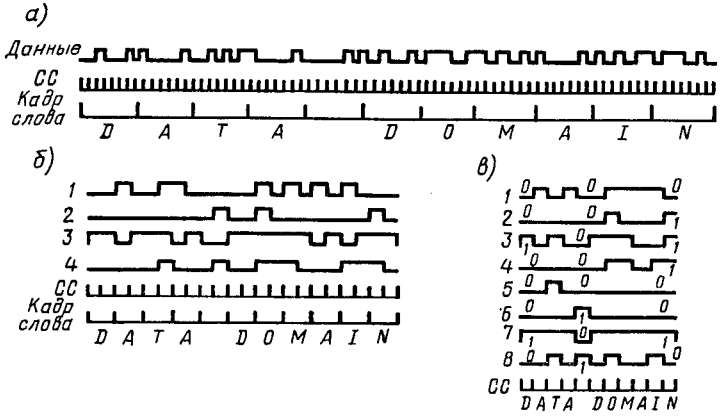


Рис. 2.2. Временны́е диаграммы синхронной передачи форматов данных

место в одноплатной микро-ЭВМ «Электроника Н МС 11100.1». Поэтому, для того чтобы представлять собранную информацию в удобном для человека виде, средства отладки должны отличать на магистрали данные, управление, адрес.

Другой особенностью магистральной организации является *двунаправленность* — по одной и той же линии устройство может посылать и принимать сигналы. Так, в микро-ЭВМ «Электроника Н МС 11100.1» данные по магистрали адресов — данных передаются процессором в ОЗУ и по этой же магистрали читаются процессором из ОЗУ. Средства отладки должны обеспечивать переключение с передачи сигнала на прием сигнала по одному и тому же каналу в течение такта синхронизации.

Технология изготовления. Различные технологии изготовления микропроцессорных наборов (ТТЛ, К-МОП, ЭСЛ) требуют различных средств сопряжения между средствами отладки и микропроцессорными системами. Так, микропроцессорные наборы, изготовленные по ТТЛ-технологии, характеризуются уровнем выходного напряжения («1») не менее 2,4 В, уровнем выходного напряжения («0») не более 0,4 В при максимальном токе нагрузки 15 мА; по К-МОП-технологии — значительно меньшей нагрузочной способностью, возможностью работать в широком диапазоне напряжений питания ($U_{\text{ип}}=3\div 10$ В), уровнем выходного напряжения («1»), равного $\approx U_{\text{ип}}$, и уровнем напряжения («0») порядка 0,01 В; по ЭСЛ-технологии — уровнем выходного напряжения «1» более 0,98 В и уровнем выходного напряжения («0») менее 1,63 В.

Число контрольных точек. Судить о поведении системы тем проще, чем больше доступных для наблюдения ее внутренних точек. Управлять поведением системы тем проще, чем больше доступных для подачи входных воздействий внутренних точек системы. Большие интегральные схемы выполняют функции, которые раньше реализовывались несколькими десятками микросхем малой и средней степени интеграции и которые имели в 3—4 раза больше внешних выводов.

Временна́я зависимость. Часто микропроцессорная система используется в режиме реального времени, поэтому важными параметрами оказываются скорость передачи на магистрали, время реакций на прерывание, временны́е характеристики отдельных устройств микропроцессорной системы. Кроме того, в микропроцессорной системе имеются неисправности, которые проявляются только при функционировании системы на предельных частотах. Приемосдаточные испытания микропроцессорной системы должны проводиться в реальных условиях с рабочими программами и реальным потоком данных.

В процессе отладки возникают противоречивые требования: с одной стороны, система должна проверяться в условиях, подобных тем, при которых она будет эксплуатироваться, так как неисправность может проявляться только при определенных, например экстремальных, условиях; с другой стороны, для локализации места источника ошибки может потребоваться доступ к ее внутренним элементам, что может повлиять на характеристики системы.

Многопроцессорные и многомашинные структуры. Микропроцессорные системы могут иметь многопроцессорную и (или) многомашинную структуру. Многопроцессорные и многомашинные проектируемые системы предъявляют дополнительные требования к средствам отладки — наблюдать и управлять несколькими магистралями.

§ 2.2. Этапы проектирования микропроцессорных систем

Микропроцессорные системы по своей сложности, требованиям и функциям могут значительно отличаться надежностными параметрами, объемом программных средств, быть однопроцессорными и многопроцессорными, построенными на одном типе микропроцессорного набора или нескольких, и т. д. В связи с этим процесс проектирования может видоизменяться в зависимости от требований, предъявляемых к системам. Например, процесс проектирования микропроцессорных систем, отличающихся одна от другой содержанием ПЗУ, будет состоять из разработки программ и изготовления ПЗУ. При проектировании многопроцессорных микропроцессорных систем, содержащих несколько типов микропроцессорных наборов, необходимо решать вопросы организации памяти, взаимодействия с процессорами, организации обмена между устройствами системы и внешней средой, согласования функционирования устройств, имеющих различную скорость работы, и т. д.

Ниже приведена примерная последовательность этапов, типичных для создания микропроцессорной системы:

1. Формализация требований к системе.
2. Разработка структуры и архитектуры системы.
3. Разработка и изготовление аппаратурных средств и программного обеспечения системы.
4. Комплексная отладка и приемосдаточные испытания.

Этап 1. На этом этапе составляются внешние спецификации, перечисляются функции системы, формализуется техническое задание (ТЗ) на систему, формально излагаются замыслы разработчика в официальной документации.

Этап 2. На данном этапе определяются функции отдельных устройств и программных средств, выбираются

микропроцессорные наборы, на базе которых будет реализована система, определяются взаимодействие между аппаратурными и программными средствами, временные характеристики отдельных устройств и программ.

Этап 3. После определения функций, реализуемых аппаратурой, и функций, реализуемых программами, схемотехники и программисты одновременно приступают к разработке и изготовлению соответственно опытного образца и программных средств. Разработка и изготовление аппаратуры состоят из разработки структурных и принципиальных схем; изготовления прототипа; автономной отладки.

Разработка программ состоит из разработки алгоритмов; написания текста исходных программ; трансляции исходных программ в объектные программы; автономной отладки.

Этап 4. См. § 2.4.

На каждом этапе проектирования микропроцессорной системы людьми могут быть внесены неисправности и приняты неверные проектные решения. Кроме того, в аппаратуре могут возникнуть дефекты.

Источники ошибок. Рассмотрим источники ошибок на первых трех этапах проектирования.

Этап 1. На этом этапе источниками ошибок могут быть: логическая несогласованность требований, упущения, неточности алгоритма.

Этап 2. На данном этапе источниками ошибок могут быть: упущения функций, несогласованность протокола взаимодействия аппаратуры и программ, неверный выбор микропроцессорных наборов, неточности алгоритмов, неверная интерпретация технических требований, упущение некоторых информационных потоков.

Этап 3. На этом этапе источниками ошибок могут быть: при разработке аппаратуры — упущения некоторых функций, неверная интерпретация технических требований, недоработка в схемах синхронизации, нарушение правил проектирования; при изготовлении прототипа — неисправности комплектующих изделий, неисправности монтажа и сборки; при разработке программных средств — упущения некоторых функций технического задания, неточности в алгоритмах, неточности кодирования.

Каждый из перечисленных источников ошибки может породить большое число субъективных или физических неисправностей, которые необходимо локализовать и

устранить. Обнаружение ошибки и локализация неисправности являются сложной задачей по нескольким причинам: во-первых, из-за большого числа неисправностей; во-вторых, из-за того, что различные неисправности могут проявляться одинаковым образом. Так как отсутствуют модели субъективных неисправностей, указанная задача не формализована. Имеются определенные успехи в области создания методов и средств обнаружения ошибок и локализации физических неисправностей. Эти методы и средства широко используются для проверки работоспособного состояния и диагностики неисправностей дискретных систем при проектировании, производстве и эксплуатации последних.

Субъективные неисправности отличаются от физических тем, что после обнаружения, локализации и коррекции больше не возникают. Однако, как следует из перечня источников ошибок, субъективные неисправности могут быть внесены на этапе разработки спецификации системы, а это означает, что даже после самых тщательных испытаний системы на соответствие ее внешним спецификациям в системе могут находиться субъективные неисправности.

Процесс проектирования — итерационный процесс. Неисправности, обнаруженные на этапе приемосдаточных испытаний, могут привести к коррекции спецификаций, а следовательно, к началу проектирования всей системы. Обнаруживать неисправности необходимо как можно раньше, для этого надо контролировать корректность проекта на каждом этапе разработки.

Проверка правильности проекта. Основные методы контроля правильности проектирования следующие: верификация — формальные методы доказательства корректности проекта; моделирование; тестирование.

В последнее время появилось много работ по верификации программного обеспечения, микропрограмм, аппаратуры. Однако эти работы пока носят теоретический характер. На практике используют моделирование поведения объекта и тестирование.

Для контроля корректности проекта на каждом этапе проектирования необходимо проводить моделирование на различных уровнях абстрактного представления системы и проверку правильности реализации заданной модели путем тестирования. На этапе формализации требований контроль корректности особо необходим, поскольку многие цели проектирования ее не формализуются или

не могут быть формализованы в принципе. Функциональная спецификация может анализироваться коллективом экспертов или моделироваться и проверяться в опытном порядке для выявления, достигаются ли желаемые цели. После утверждения функциональной спецификации начинается разработка функциональных тестовых программ, предназначенных для установления правильности функционирования системы в соответствии с ее функциональной спецификацией. В идеальном случае разрабатываются тесты, целиком основанные на этой спецификации и дающие возможность проверки любой реализации системы, которая объявляется способной выполнять функции, оговоренные в спецификации. Этот способ — полная противоположность другим, где тесты строятся применительно к конкретным реализациям. Независимая от реализации функциональная проверка обычно заманчива лишь в теоретическом плане, но практического значения не имеет из-за высокой степени общности.

Примечание. Существуют способы построения тестовых программ по функциональной спецификации с использованием информации о конкретной реализации системы [2].

Автоматизация утомительной работы по составлению тестовых программ не только сокращает продолжительность периода конструирования/отладки за счет получения тестовых программ на этапе конструирования (поскольку они могут быть сгенерированы сразу после формирования требований к системе), но и позволяет проектировщику изменять спецификации, не заботясь о переписывании всех тестовых программ заново. Однако на практике разработке тестов часто присваивают более низкий приоритет по сравнению с проектом, поэтому тестовые программы появляются значительно позже его завершения. Но даже если детальные тесты оказываются подготовленными, часто практически нецелесообразно запускать их на имитаторе, так как детальное моделирование требует больших затрат средств на разработку программ и времени на вычисление, в результате бóльшая часть работы по отладке должна откладываться до момента создания прототипа системы.

После обнаружения ошибки должен быть локализован ее источник, чтобы провести коррекцию на соответствующем уровне абстрактного представления системы и в соответствующем месте. Ложное определение источника ошибки или проведение коррекций на другом уровне

абстрактного представления системы приводит к тому, что информация о системе на верхних уровнях становится ошибочной и не может быть использована для дальнейшей отладки при производстве и эксплуатации системы. Например, если неисправность внесена в исходный текст программы, написанной на языке ассемблера, а коррекция проведена в объектном коде, то дальнейшая отладка программы ведется в объектном коде; при этом все преимущества написания программы на языке ассемблера сводятся на нет.

§ 2.3. Автономная отладка микропроцессорных систем

Процесс отладки прототипа проектируемой системы должен начинаться с отладки аппаратуры и отладки программ.

Отладка аппаратуры. Она предполагает тестирование отдельных устройств микропроцессорной системы — процессора, ОЗУ, контроллеров, блока питания, генератора тактовых импульсов — путем подачи тестовых входных воздействий и съема ответных реакций. Тестовые входные воздействия и ответные реакции определяются исходя из спецификаций на устройства, а также структурных схем устройств. При этом проверяются реальная аппаратура прототипа, спецификации, структурные схемы и отлаживаются тесты. После отладки отдельных устройств проверяется их взаимодействие. Процессор системы работает с магистралями адресов, данных и управления. Анализируя их сигналы, можно проконтролировать выполнение программы в процессоре. Поскольку МА и МД синхронные, их работу лучше всего проверить с помощью методов логических состояний. Перед анализом последовательностей данных на этих магистралях необходимо удостовериться в том, что сигналы, управляющие взаимодействием процессора с другими устройствами, выдаются в соответствующем порядке. Поскольку МУ состоит из линий, работающих асинхронно, необходимо просматривать сигналы многих линий в течение одного и того же промежутка времени. Для анализа асинхронной работы линий управления необходимо также наблюдать за сигналами на них при возникновении определенного события, чтобы можно было четко разделить и идентифицировать различные состояния линий. Например, среди сигналов

МУ могут быть сигналы длительностью всего несколько наносекунд, но могут также возникать кратковременные ложные узкие импульсы, вызванные перекрестными помехами или шумами [8].

После того как доказана работоспособность МУ, проводится дальнейшая проверка работы аппаратуры при различных режимах адресации процессора и кодах выбираемых данных. Для проверки выполнения процессором инструкций разрабатывается тестовая программа, которая помещается в ОЗУ или РППЗУ. При этом проверяется временна́я диаграмма сигналов и прохождения данных в системе (как осуществляется передача информации по отношению к строб-сигналам). Если тестовая программа — системный проверяющий тест — пройдет успешно, можно утверждать то, что автономно аппаратура отлажена.

При автономной отладке аппаратуры могут потребоваться приборы, умеющие: а) выполнять функции аналогового прибора, т. е. измерять напряжение и ток; воспроизводить форму сигнала, подавать импульсы определенной формы и т. д.; б) подавать последовательность сигналов одновременно на несколько входов в соответствии с заданной временно́й диаграммой или заданным алгоритмом функционирования аппаратуры, представленным в спецификации на языке высокого уровня, или другим способом; собирать значения сигналов многих линий в течение одного и того же промежутка времени, причем промежуток времени определяется задаваемыми, программируемыми событиями — комбинацией или последовательностью сигналов на линиях, например ложным сигналом на линии; обрабатывать и представлять собранную информацию либо в виде временно́й диаграммы, либо в виде диаграммы или таблицы логических состояний, либо на языке высокого уровня, например языке регистровых передач.

Для автономной отладки аппаратуры широко используются осциллографы, вольтметры, амперметры, частотомеры, генераторы импульсов, позволяющие отлаживать аппаратуру на схемном уровне. Чтобы автономно отладить аппаратуру микропроцессорных систем на более высоком уровне, применяют логические анализаторы, генераторы слов, пульты, комплексы диагностирования.

Отладка программ. Отладка программ микропроцессорной системы проводится, как правило, на тех же ЭВМ, на которых велась разработка программ, и на том же

языке программирования, на котором написаны отлаживаемые программы, и может быть начата на ЭВМ даже при отсутствии аппаратуры микропроцессорной системы. При этом в системном программном обеспечении ЭВМ должны находиться программы (интерпретаторы или эмуляторы), моделирующие функции отсутствующих аппаратурных средств. Так, разработка и автономная отладка программных средств может вестись на больших ЭВМ, мини-ЭВМ, микро-ЭВМ, система команд которых не совпадает с системой команд используемого микропроцессора. Кроме того, при отладке программ может отсутствовать внешняя среда микропроцессорной системы, и ее также необходимо моделировать.

Проверка корректности программ, т. е. проверка соответствия их внешним спецификациям, осуществляется *тестированием*. Программы проверяются на функционирование с различными исходными данными. Результаты функционирования программ сравниваются с эталонными значениями.

Отладка программ подразделяется на следующие этапы [9]: планирование отладки; составление тестов и задания на отладку, исполнение программ; информирование о результатах исполнения программ по заданным исходным данным; анализ результатов, обнаружение ошибок и локализация неисправностей.

Существует два способа начального тестирования программ: пошаговый режим, трассировка программ [10].

В пошаговом режиме программа выполняется по одной команде за один раз, а пользователь анализирует содержимое памяти, регистров и т. д., чтобы проверить, соответствуют ли результаты ожидаемым. Пошаговый режим может быть трудоемким, если средства отладки будут требовать отдельных команд после каждого шага для того, чтобы показать необходимую информацию в понятном для пользователя виде. Имеются средства отладки, автоматически показывающие после каждого шага содержимое регистров процессора и ячеек памяти, используемых в последней команде, и несколько следующих команд. Пошаговый режим является весьма мощным средством предварительного тестирования, так как позволяет обнаруживать неисправности, прежде чем они существенно исказят программу и данные. Кроме того, неоднократно проходя отдельными шагами через один и тот же участок объектной программы, программист может легко изменять содержимое регистров и ячеек памяти,

особенно если средства отладки имеют динамически обновляемый дисплей, и тем самым проверить работу программы в разных условиях. Этот интерактивный режим отладки программы позволяет разработчику постоянно упреждать, что будет делать его программа, и оперативно обнаруживать ошибку. Однако пошаговый режим с автоматическим показом результатов возможен только тогда, когда средства отладки содержат в своем составе дисплей с прямым доступом в память, так как после каждого шага на экране дисплея нужно показывать большой объем информации.

Исполнение программ осуществляется по шагам последовательно во времени и в соответствии с заданиями, содержащимися в операторах. При этом производится переработка значений переменных и определения оператора приемника. Если в ходе исполнения программы регистрируется последовательность операторов, реализуемых на каждом шаге процесса, то получается трасса или маршрут исполнения программы, который для конкретной программы зависит только от значений исходных данных [9].

Трассировка программ больше пригодна для отладочных средств, имеющих медленный, последовательный терминал. Программа-отладчик выполняет непрерывно команду за командой и выводит содержимое регистров процессора на терминал после каждого шага. Некоторые отладчики выводят также на терминал команды в дизассемблерной форме. Но при этом способе содержимое памяти не выводится на терминал и разработчик должен сам делать выводы об изменениях в ней. Отслеживание программы продолжается автоматически до тех пор, пока не будет остановлено извне. Результатом трассировки программы будут данные на экране дисплея или же в случае использования в качестве терминала печатающего устройства — длинная распечатка с ходом выполнения программы. Программист, анализируя эти данные, может обнаружить ошибки. Трассировка программ не дает, однако, возможности изменять содержимое памяти и регистров и может послужить причиной того, что программа разрушит себя или свои данные прежде, чем отслеживание будет остановлено.

Отдельные участки программы после проверки, используя пошаговый режим или трассировку, можно объединить и проверить с помощью установки контрольных точек, вводимых в программу и прерывающих ее исполне-

ние, для передачи управления программе-отладчику. По контрольным точкам можно по желанию выполнить избранные участки программы и проанализировать результаты. Контрольные точки устанавливаются обычно для конкретной команды, но в некоторых системах предусматриваются прерывания программы при чтении или записи данных в определенные ячейки памяти. Возможны и более сложные условия прерывания программы.

Расстановка контрольных точек предполагает, что программист связывает с ней точный адрес памяти. Для некоторых отладчиков программист задает абсолютный шестнадцатиричный адрес. Последние отладчики допускают символьные значения адресов, которые программист определяет в исходной программе; это позволяет значительно экономить время, распечатывая после каждого редактирования и транслирования программы новую копию листинга.

При тестировании можно планировать проверку всех возможных маршрутов исполнения программы для разных исходных переменных. Однако это реализуемо только для очень простых программ небольшого объема при малых диапазонах изменения исходных данных. Поэтому при планировании отладки программ применяют критерии полноты тестирования, которые, однако, не гарантируют полной проверки программ. Выбор критерия зависит от наличия ресурсов для тестирования и структурной сложности отлаживаемой программы. Критерии характеризуются глубиной контроля программ и объемом проверок [9].

В процессе отладки основная часть неисправностей в программах обнаруживается и затем устраняется. Однако всегда возможен пропуск нескольких неисправностей.

Средства отладки программ должны: а) управлять исполнением программ (останавливать, изменять порядок, запускать и т. д.); б) собирать информацию о ходе выполнения программы; в) обеспечивать обмен информацией (диалог) между программистом и ЭВМ на уровне языка программирования; г) моделировать работу отсутствующих аппаратурных средств микропроцессорной системы.

Отладка программ микропроцессорных систем ведется: на больших ЭВМ типа ЕС-1060, БЭСМ-6, «Эльбрус» в режиме разделения времени; на автоматизированных рабочих местах программиста (АРМ-П), построенных

на базе мини-ЭВМ СМ ЭВМ; на микро-ЭВМ, таких, как «Электроника-60», диалоговый вычислительный комплекс «Электроника НЦ-8020»; на отладочных комплексах и комплексах развития.

§ 2.4. Комплексная отладка микропроцессорных систем

Как правило, микропроцессорная система — это система реального времени, т. е. корректность ее функционирования зависит от времени выполнения отдельных программ и скорости работы аппаратуры. Поэтому *система считается отлаженной после того, как рабочие программы правильно функционируют на действительной аппаратуре системы в реальных условиях.* Дополнительным свойством, которым должны обладать средства комплексной отладки по сравнению со средствами автономной отладки, является возможность управления поведением микропроцессорной системы и сбора информации о ее поведении в реальном времени.

Тенденция развития средств отладки микропроцессорных систем состоит в объединении свойств нескольких приборов в одном комплексе, в создании универсальных средств, пригодных для автономной отладки аппаратуры, генерации и автономной отладки программ и комплексной отладки системы. Эти средства позволяют вести разработку и отладку, постепенно усложняя аппаратуру и программы. При этом разработка, изготовление и отладка планируются поэтапно с нарастанием сложности; новая, неотлаженная аппаратура и программа вводятся в создаваемую систему, присоединяются к проверенной ее части.

Если отладка программ ведется с использованием эмуляционного ОЗУ, а затем изготовляются микросхемы ПЗУ, то после этого микропроцессорная система должна быть протестирована.

Средства отладки на последних этапах не должны влиять на правильность функционирования системы, вносить задержки, дополнительные нагрузки.

При комплексной отладке наряду с детерминированным используется статистическое тестирование, при котором микропроцессорная система проверяется при изменении исходных переменных в соответствии со статистическими законами работы источников информации. Полнота контроля работоспособности проектируемой

системы возрастает за счет расширения диапазона возможных сочетаний переменных и соответствующих им логических маршрутов обработки информации.

Существует пять основных приемов комплексной отладки микропроцессорной системы: 1) останов функционирования системы при возникновении определенного события; 2) чтение (изменение) содержимого памяти или регистров системы; 3) пошаговое отслеживание поведения системы; 4) отслеживание поведения системы в реальном времени; 5) временно́е согласование программ.

Комплексная отладка завершается приемосдаточными испытаниями, показывающими соответствие спроектированной системы техническому заданию. Для проведения комплексной отладки микропроцессорной системы используют логические анализаторы и комплексы: оценочные; отладочные; развития микропроцессоров; диагностирования; средств отладки.

Глава 3
ЛОГИЧЕСКИЕ АНАЛИЗАТОРЫ

§ 3.1. Принцип действия логических анализаторов

Логические анализаторы — контрольно-измерительные приборы, предназначенные для сбора данных о поведении дискретных систем, для обработки этих данных и представления их человеку на различных уровнях абстракции. Они работают независимо и незаметно для испытуемых дискретных систем и применяются для их отладки и диагностирования (в первую очередь микропроцессорных систем) на всех этапах жизненного цикла.

Логические анализаторы характеризуются числом каналов, емкостью памяти на канал, частотой записи, способами синхронизации и запуска, формами представления данных.

Обобщенная структура логического анализатора (рис. 3.1) включает в себя компаратор уровней входных сигналов *КУ*, запоминающее устройство *ЗУ*, логический компаратор *КЛ*, генераторы задержки *ГЗ* и синхросигналов *ГСС*, переключатель режима *ПР*, устройства запуска *УЗ* и управления визуальным выводом *УУВВ*, дисплей *Д*. На входные каналы логического анализатора поступают сигналы с отлаживаемой или диагностируемой системы. Компараторы *КУ* распределяют их на соответ-

ствующие логические уровни. Сформированный компараторами уровней набор значений сигналов подается на входы запоминающего устройства и логического компаратора. Запоминающее устройство логического анализатора функционирует подобно группе сдвиговых регистров. Логический компаратор предварительно настраивается (программируется) на обнаружение определенной последовательности наборов значений сигналов. После

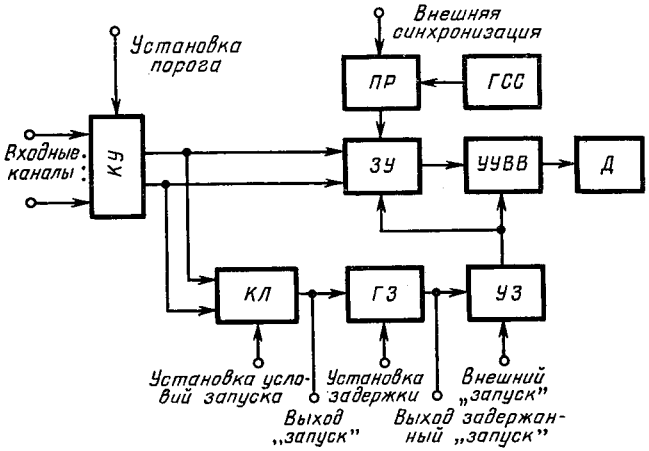


Рис. 3.1. Обобщенная структура логического анализатора

поступления запрограммированной последовательности входных наборов *КЛ* выдает сигнал генератору задержки, последний по истечении запрограммированного времени выдает сигнал на *УЗ*, которое инициирует или прекращает запись наборов значений входных сигналов в *ЗУ*. После прекращения записи в *ЗУ* устройство управления визуальным выводом *УУВВ* транслирует информацию на экран дисплея в удобном для интерпретации виде (таблица состояний, временны́е диаграммы, графы и т. д.).

§ 3.2. Фиксация данных о поведении микропроцессорной системы

Фиксация данных о поведении системы предполагает: подключение (контактирование) логического анализатора к испытуемому объекту; определение логических значений сигналов, поступающих от объекта; запись

сформированной информации в память логического анализатора.

Подключение логического анализатора к испытуемому объекту как в электротехническом, так и механическом смысле требует обеспечения доступа к ограниченным по размерам участкам дискретного устройства с большой плотностью компоновки, где размещается много элементов с сигналами различных уровней. В анализаторах используются миниатюрные щупы, или зонды, позволяющие подключиться к любым контактам одного ряда выводов корпуса типа DIP со стандартным шагом 2,5 мм. Некоторые логические анализаторы снабжаются специализированными зондами, ориентированными на определенный тип микропроцессоров.

По аналогии с измерительными системами, при проектировании логических анализаторов особое внимание обращается на то, чтобы их входные цепи не влияли на функционирование испытуемого объекта. Логические анализаторы обладают высоким входным сопротивлением (примерно 1 МОм) и малой входной емкостью (порядка 10—25 пФ). Для уменьшения входной емкости компараторы уровней логических анализаторов делаются выносными.

Логические анализаторы при *определении значений сигналов* в отличие от представления реальных временны́х функций при исследовании аналоговых сигналов с помощью осциллографа отображают нормированные по уровню цифровые сигналы: если значения входного сигнала превышают порог срабатывания входного компаратора, то значения сигнала на входе памяти соответствуют «1», а если не превышает, то «0». Вид входного сигнала $U(t)$ на входе и выходе компаратора уровней представлен соответственно на рис. 3.2, *а, б*. Схемотехническая реали-

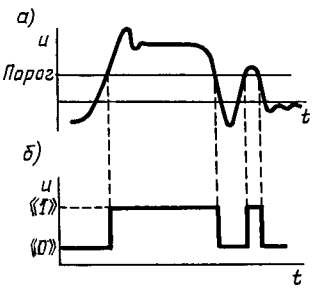
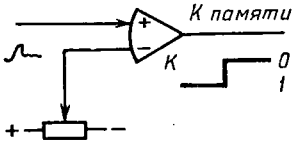


Рис. 3.2. Вид сигналов на входе (*а*) и выходе (*б*) компаратора

Рис. 3.3. Схема компаратора уровней



зация данного метода формирования сигналов осуществляется с помощью устройства, схема которого показана на рис. 3.3. Пороговое напряжение компараторов *К* регулируется у большинства приборов в пределах от —10 до +10 В [11].

Синхронный и асинхронный режимы записи. В тактовые моменты времени производится запись информации в память, в эти моменты времени фиксируются изменения состояния испытуемого объекта. Синхросигналы могут

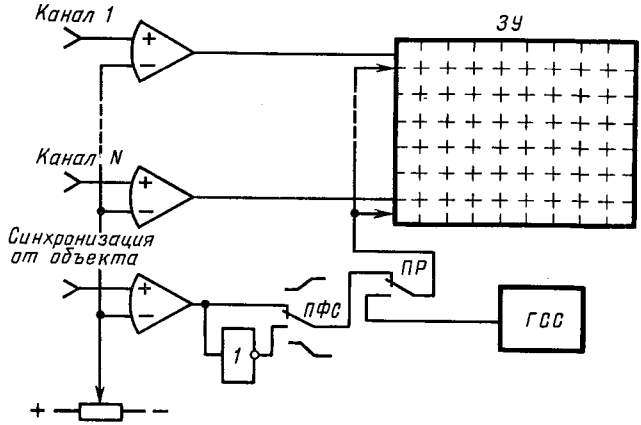


Рис. 3.4. Схема выбора режима записи данных и фронта синхросигнала

поступать как извне, с проектируемой или диагностируемой системы, так и с внутреннего генератора . В том случае, когда синхросигналы поступают от внутреннего генератора синхросигналов независимо от испытуемого объекта, реализуется асинхронный режим записи данных; если же для синхронизации работы логического анализатора используются сигналы испытуемого объекта, то реализуется синхронный режим записи данных. При этом набор значений сигналов («1», «0») записывается в память передним или задним фронтом синхросигнала от объекта. На рис. 3.4 показан пример реализации выбора режима записи данных и фронта синхросигнала. В зависимости от положения переключателя фронта синхросигнала *ПФС* данные в *ЗУ* будут записываться по переднему или заднему фронту синхросигнала. В зависимости от положения переключателя режима *ПР* данные в *ЗУ* будут записы-

ваться по синхросигналам, поступающим от испытуемого объекта, или генератора синхросигналов *ГСС* логического анализатора.

Логические анализаторы, в которых реализован асинхронный режим, называются *анализаторами временны́х соотношений* или *анализаторами логических временны́х диаграмм*. Логические анализаторы, имеющие синхронный режим, называются *анализаторами логических состояний*.

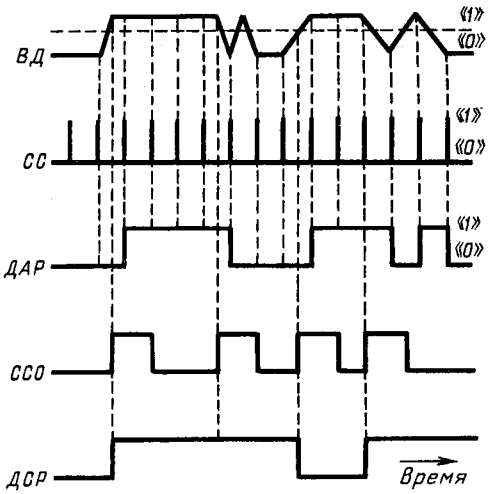


Рис. 3.5. Временна́я диаграмма записи данных в синхронном и асинхронном режимах

Современной тенденцией стало создание универсальных логических анализаторов, сочетающих в одном приборе возможности временно́го и логического анализа. Разница между синхронным и асинхронным режимами работы анализаторов поясняется временны́ми диаграммами (рис. 3.5). Входные данные *ВД* поступают на входной канал логического анализатора от испытуемого объекта. В асинхронном режиме данные *ВД* стробируются синхросигналом *СС* внутреннего генератора. В результате стробирования в память записываются данные *ДАР*. В синхронном режиме данные *ВД* стробируются передним фронтом синхросигнала испытуемого объекта *ССО*. В синхронном режиме в память записываются данные

ДСР. Каждый из режимов работы логического анализатора имеет свои преимущества и недостатки.

Асинхронный режим работы анализатора. Его обычно используют для нетактируемых информационных сигналов, для регистрации переходных процессов, т. е. там, где тактовая частота регистрации может не синхронизироваться с моментами изменения состояния входного сигнала. Достоинство этого режима работы анализатора — возможность применения тактовых частот, намного бóльших частоты повторения исследуемого сигнала. Тем самым обеспечивается весьма точная и детальная регистрация временнóго хода входного сигнала. Для обеспечения высокой разрешающей способности необходимо, чтобы частота синхронизации анализатора временны́х соотношений в 5—10 раз превышала частоту наблюдаемых событий. Для выбора тактовой частоты записи логического анализатора потребитель должен определить требуемую разрешающую способность, т. е. минимальный интервал времени, который может встречаться в проверяемой системе. Часто этот параметр определяется не частотой синхросигналов, а взаимным расположением во времени сигналов, действующих в системе. Если этот минимальный интервал равен 50 нс, то следует остановиться на анализаторе с тактовой частотой записи 20 МГц, при интервале 10 нс требуется анализатор с тактовой частотой записи 100 МГц.

Синхронный режим работы анализатора. Он рационален при исследовании систем на более высоком уровне описания (например, программном), так как интерес представляют состояния элементов системы в определенные моменты времени, в то время как фиксация моментов перехода из одного состояния в другое в данном случае не обязательна. Таким образом, в этом режиме работы логический анализатор регистрирует лишь установившиеся к моменту синхронизации логические уровни на выходах испытуемой системы.

Поступающие данные воспринимаются анализатором логических состояний точно так же, как они воспринимаются испытуемой системой. Это значит, что данные принимаются в анализатор логических состояний по тем же самым синхросигналам (появляющимся в те же самые моменты времени и имеющим те же самые фронты), которые управляют приемом данных в испытуемой системе. В типичной дискретной системе данные начинают изменяться почти непосредственно после окончания фронта

синхросигнала и снова становятся стабильными незадолго до начала фронта следующего синхросигнала. Вследствие этого анализаторы логических состояний имеют минимально необходимое время фиксации данных (период, в течение которого данные не должны изменяться после прихода фронта синхросигнала) и минимально возможное время установления данных (период, в течение которого данные должны иметь стабильные уровни до прихода синхросигнала).

Дополнительные возможности по сбору данных обеспечивают *квалификаторы* (квалификационные входы, определители) — отдельные каналы, значения сигналов которых не фиксируются в памяти, но определяют функции коммутации синхросигналов. В зависимости от значения сигнала на квалификаторе входные данные будут или не будут записываться в память. Использование подобных квалификаторов позволяет пользователю записывать данные выборочно и тем самым экономить емкость памяти логического анализатора. Например, если действие элемента *ЗУ* (дешифратора, накопителя и т. д.) проектируемой системы будет заподозрено как неверное во время записи, то сбор данных при прогоне программы только при активной линии признака записи позволяет значительно сократить несущественную информацию.

Идентификация информации. Важной характеристикой современных анализаторов логических состояний является возможность идентификации информации, передаваемой по одной магистрали [12]. Этой проблемы не существовало раньше, так как, например, в микропроцессорах 8080 и 6800 данные, адреса и команды управления передавались по своим магистралям. В современных микропроцессорах (например, К1801 и 8085) по линиям одной и той же магистрали передаются и данные и адреса (или часть адресной или управляющей информации). В этом случае анализатор, имеющий только один источник синхросигналов, не сможет различить данные, передаваемые по одной магистрали. Одним из необходимых условий разделения данных является наличие трех различных источников синхросигналов — режим многофазовой синхронизации. Этот режим работы анализатора может быть реализован следующим образом. Входные каналы анализатора разбиваются на две или три группы. По одной группе каналов записывается адрес, по другой — данные, по третьей — команды управления. Каждая группа каналов записывается по своему синхросигналу и в

свои разряды буферного регистра. После прихода всех синхросигналов данные с буферного регистра подаются в память и на логический компаратор одновременно.

Фиксация кратковременных сигналов. В реальных системах в промежутках между синхросигналами могут возникать ложные кратковременные сигналы и помехи, которые не фиксируются в памяти вне зависимости от того, в каком режиме (синхронном или асинхронном) работает анализатор. Обнаружение такого рода сигналов

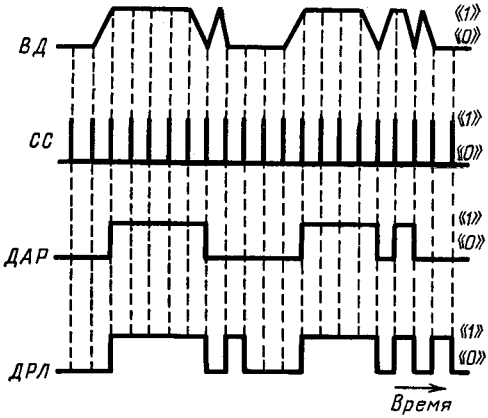


Рис. 3.6. Временны́е диаграммы фиксации кратковременных сигналов

осуществляется методами: 1) увеличения тактовой частоты в асинхронном режиме; 2) использования режима «ловушек».

Возможности использования первого метода ограничены, с одной стороны, конечным быстродействием применяемых элементов, а с другой стороны, синхронным (при необходимости) режимом работы анализатора. Кроме того, увеличение тактовой частоты в асинхронном режиме приводит к значительному увеличению емкости памяти.

Второй метод позволяет более просто решить даннную проблему: кратковременные сигналы, появляющиеся в промежутке между синхросигналами, фиксируются триггерами-защелками и следующим синхросигналом записываются в память. Процесс обнаружения кратковременных сигналов такого рода поясняется временны́ми диа-

граммами (рис. 3.6). Входные данные *ВД*, поступающие с испытуемого объекта, стробируются синхросигналами *СС* генератора логического анализатора в одном случае без использования режима «ловушек», в другом — с использованием режима «ловушек». В первом случае в память записываются данные *ДАР*, во втором — данные *ДРЛ*. Недостаток данного метода — невозможность определения длительности коротких сигналов и наличие дополнительного оборудования. В то же время при анализе дискретных систем чаще всего достаточно зафиксировать лишь наличие данного сигнала, не измеряя его временны́х параметров.

§ 3.3. Способы запуска логических анализаторов

Для целенаправленного поиска источника ошибки необходимо просматривать отдельные участки потока данных, характеризующих поведение испытуемой системы. Этот поток данных поступает непрерывно на входные каналы логического анализатора. Для выбора необходимого участка данных требуется проанализировать поступающие данные и в нужный момент прекратить или инициировать их запись в память. При этом в памяти будут храниться данные выбранного участка поведения системы. Анализ данных и обнаружение заданного события называют *запуском логического анализатора*. Если сигнал запуска инициирует запись данных в память, то такой запуск называется положительным, если сигнал запуска прекращает запись в память — отрицательным.

Способы запуска логического анализатора зависят от того, на каком уровне абстрактного представления поведения системы ищется неисправность. Поведение системы на программном уровне характеризуется последовательностью адресов, команд и данных, а также временны́ми соотношениями между отдельными последовательностями (между отдельными программами). При исследовании поведения системы на этом уровне запускающий сигнал логического анализатора должен определяться либо последовательностью адресов, либо последовательностью адресов и данных, либо последовательностью команд и данных, либо последовательностью всех трех составляющих, а также временны́ми соотношениями между последовательностями.

Поведение системы на логическом уровне характеризуется последовательностью наборов значений входов и выходов и сменой состояний, а также временны́ми соотношениями между последовательностями. Если диагностирование неисправности ведется на этом уровне, то сигнал запуска должен характеризоваться последовательностью наборов значений входов и выходов устройства системы при известном его начальном состоянии, временны́ми соотношениями между последовательностями (сигналами), временем существования набора (сигнала).

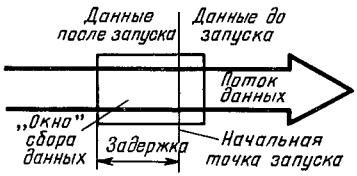


Рис. 3.7. Возможности генератора задержки

Способы запуска логического анализатора существенно отличаются от способов запуска развертки осциллографа. Для последней обычно формируется импульс внутреннего или внешнего запуска отдельно для каждого канала, причем если источник запуска задан, то остальные критерии выбора ограничиваются параметрами сигнала (положительным или отрицательным фронтом, уровнем запуска и т. д.), которые имеют большое значение при исследовании систем на схемном уровне.

В отличие от осциллографа логический анализатор имеет отрицательный запуск, который не является началом отслеживания. Отрицательный запуск прерывает сбор данных, и память будет удерживать данные, записанные до получения пускового сигнала. Эта особенность дает возможность вернуться назад и проанализировать логические состояния или временны́е соотношения сигналов, которые предшествовали пуску. Так как последовательности данных могут быть очень длинными, а емкость памяти ограничена, то желательно перемещать «окно» сбора данных (рис. 3.7) с помощью генератора задержки (когда пусковой сигнал распознается, то запись данных в память прекращается не сразу, генератор задержки отсчитывает заданное число синхросигналов, в результате чего пусковой сигнал смещается внутри «окна» сбора данных или даже вне его). Большинство современных анализаторов позволяет наблюдать либо предшествующие созданию условий запуска данные, либо данные, следующие после возникновения условий запуска, либо одновременно данные, предшествующие созданию условий за-

пуска и следующие после них. Используя, например, анализатор с емкостью памяти 16 слов и выбрав время задержки, соответствующее длительности восьми синхросигналов, можно наблюдать на экране дисплея семь слов, предшествующих запуску, одно слово, соответствующее запускающему коду, и, наконец, восемь последующих слов.

Запуск по кодовому слову. Самый простой способ запуска, применяемый уже в первых анализаторах,— *запуск по кодовому слову (по комбинации значений сигналов)*. Введение в логический анализатор схемы запуска

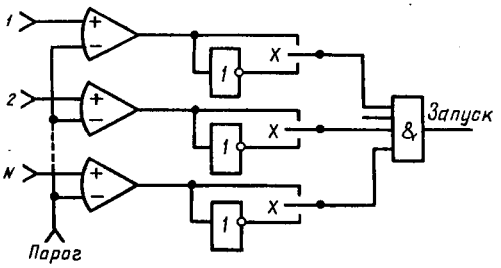


Рис. 3.8. Схема запуска по кодовому слову

такого типа позволяет производить запуск при появлении на входах логического анализатора определенного, заранее выбранного двоичного слова. Пример практической реализации такой схемы запуска показан на рис. 3.8. Сигнал запуска зависит от положения каждого переключателя. Если переключатель находится в положении *X*, то сигнал запуска не зависит от значения сигнала на соответствующем канале (1, 2, ..., *N*). Если анализатор подключен к 16-разрядной адресной магистрали и на ней появляется заранее определенное кодовое слово, то при отрицательном запуске регистрация прекращается и в памяти анализатора остается вся информация, прошедшая до момента запуска (в пределах емкости памяти), а при положительном запуске память будет заполнена данными, пришедшими после момента запуска.

Примечание. В ряде логических анализаторов для расширения длины запускающего слова используются квалификаторы.

Запуск по последовательностям слов. Программы, как правило, содержат циклы подпрограмм и даже вло-

женные циклы, так что выбранное запускающее слово при последовательных проходах цикла может встречаться многократно. Чтобы анализатор мог различать эти циклы, в устройство запуска добавляется счетчик проходов, который задерживает момент фиксации данных в памяти, отсчитывая не синхросигналы, а число появлений запускающего слова, так что запуск производится по n-му проходу подпрограммы. Программные средства со многими вложенными уровнями подпрограмм или рекурсивными подпрограммами или подпрограммами повторной входимости требуют очень широких возможностей трассировки. Единственным способом обнаружить конкретный участок программного кода может оказаться использование спецификаций на трассировку с несколькими уровнями последовательного запуска. Запуск по последовательностям слов позволяет выбрать в программе, имеющей множество ветвей, один определенный путь. На рис. 3.9 представлен граф программы, где вершина — адрес, а дуга — переход между адресами. При отладке программы необходимо убедиться в том, что каждый переход между операторами был пройден. В рассматриваемом примере имеются маршруты ABEFJKL, ACBEFJKL, ACHKL, ADL, которые можно различить, осуществляя запуск по последовательностям слов ABL, ACBL, ACHL, ADL. Если рассматривать программу, написанную для одноплатной микро-ЭВМ «Электроника Н МС 11100.1», то A, B, C,...— 16-разрядные адреса. После запуска в памяти анализатора будет храниться трасса того или иного участка программы в зависимости от выбранной последовательности слов.

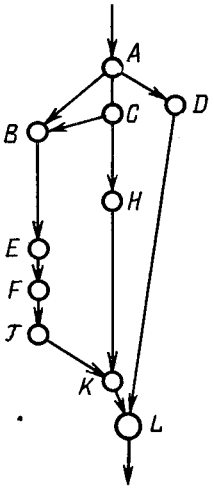


Рис. 3.9. Граф программы

Запуск по несовпадению. Больше всего хлопот доставляют перемежающиеся неисправности, которые появляются не часто, случайным образом, и при обычных периодически повторяющихся экспериментах могут не обнаруживаться. Для борьбы с перемежающимися неисправностями дискретных систем в некоторых моделях анализаторов предусмотрен *запуск по несовпадению.*

Для его реализации в логический анализатор вводится дополнительно ЗУ. Используя критерии запуска A, B и C, анализатор принимает блок данных из подозреваемой части программы и записывает его в свое второе, «эталонное», ЗУ. Далее, перейдя в режим проверки на несовпадение, анализатор сперва ожидает появления данных, удовлетворяющих критерию *ABC*. Когда эти данные приходят, он регистрирует их и сравнивает с данными, хранящимися в «эталонном» ЗУ. Если данные различны, прибор запускается. Если они одинаковые, он снова подготавливает эталонные данные и ожидает нового прихода данных, удовлетворяющих критерию *ABC*.

Цифровая защита запуска. В современных осциллографах предусматривается режим, в котором запуск развертки защищен от воздействия нежелательных сигналов [13]. Аналогичный режим имеется и в анализаторе временны́х соотношений; здесь он называется *режимом цифровой фильтрации, фильтрацией ложных сигналов* или *цифровой защитой запуска*. В этом режиме выполняется простое требование: запускающее слово должно оставаться действительным больше одного интервала квантования. Такой режим полезен при исследовании линий с тремя устойчивыми состояниями или систем, нечувствительных к весьма коротким ложным сигналам. В подобных случаях может потребоваться, например, осуществлять квантование пять раз за один период тактовой частоты и вместе с тем не допускать запуска ложными сигналами, имеющими длительность $^1/_5$ периода.

Способы запуска — важный показатель логического анализатора, так как они определяют требование к емкости памяти анализатора. От них зависят удобство и возможность использования логического анализатора для обнаружения и локализации места неисправности.

Необходимо отметить очень важную сторону логического анализатора: сигнал запуска может быть использован для запуска осциллографа или другого анализатора при их совместной работе.

§ 3.4. Формы представления данных о поведении микропроцессорных систем

Одной из основных характеристик логических анализаторов является форма представления данных о поведении отлаживаемой или диагностируемой системы на экране дисплея. Наиболее распространенные формы отображения

собранных данных логических анализаторов — *временны́е диаграммы и таблицы последовательности данных.* При первом представлении на экране изображается временна́я диаграмма 8, 16, 24 каналов. Пример временно́й диаграммы процедуры ввода микро-ЭВМ «Электроника Н МС 11100.1» дается на рис. 3.10 (здесь по отношению к рис. 1.3, *а* обозначения следующие: А0 — сигнал СИА; А1 — Ввод; А2 — Вывод; А3 — СИП; А4 — А6 — ДА; А7 — синхросигнал). Момент запуска индицируется на экране вертикальной прерывистой линией, называемой

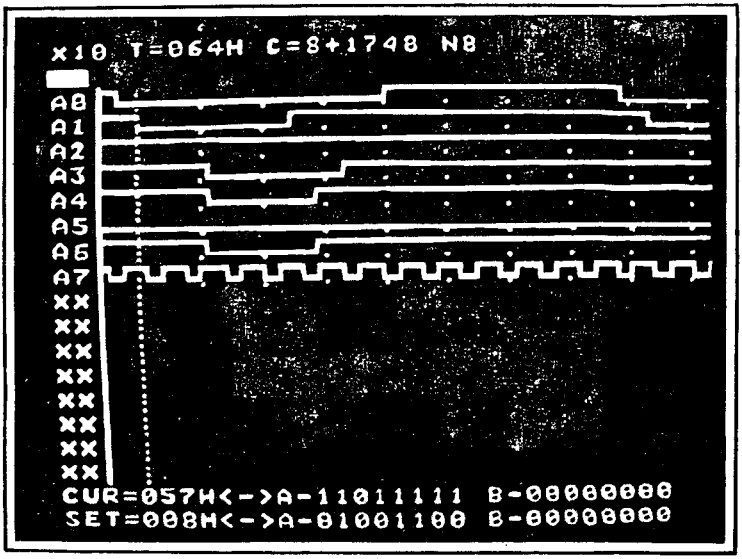
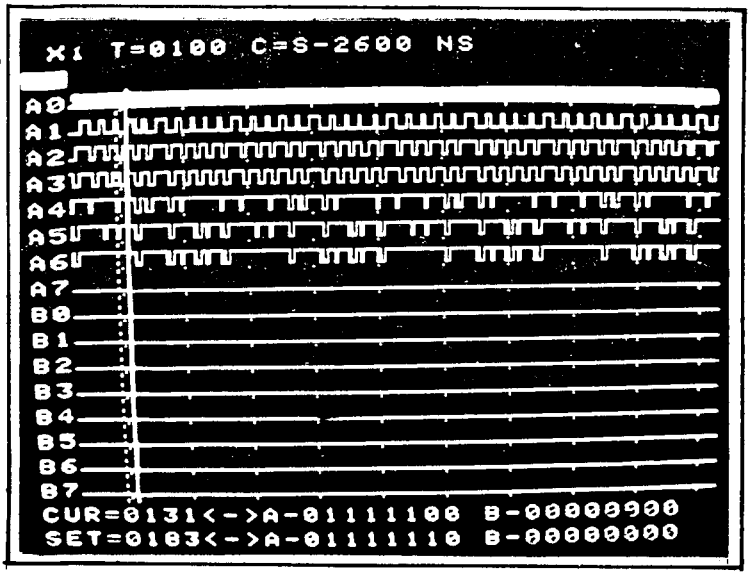


Рис. 3.10. Реальная временна́я диаграмма процедуры ввода магистрали микро-ЭВМ «Электроника Н МС 11100.1»

маркером запуска. Тем самым на экране наглядно фиксируется раздел между данными, поступившими до и после момента запуска.

В логических анализаторах имеется возможность изменять масштаб изображения по временно́й оси. На рис. 3.11, *а, б* (здесь по отношению к рис. 1.3, *а* обозначения такие: А0 — синхросигнал; А1 — СИА; А2 — Ввод; А3 — СИП; А4 — А6 — ДА) показана одна и та же временна́я диаграмма в различающихся в 10 раз масштабах по оси времени.

а)



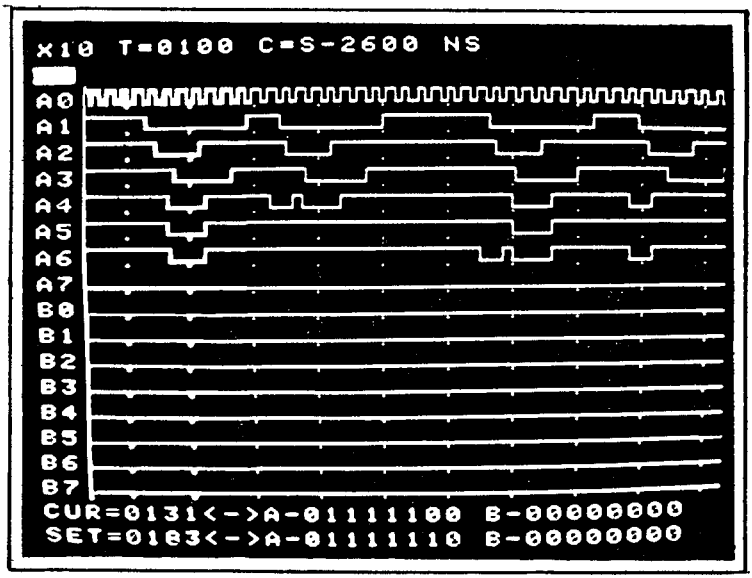


Рис. 3.11. Реальные временны́е диаграммы работы микро-ЭВМ «Электроника Н МС 11100.1»

Таблица последовательности данных может интерпретировать таблицы истинности комбинационного устройства, ввода—вывода и состояний устройства с памятью, трассу программы. Данные в таблице последовательности воспроизводятся в двоичном, восьмеричном, шестнадцатиричном кодах, в коде ASCII. Для удобства восприятия строки и столбцы данных в таблице последовательности группируются в блоки, разделенные промежуточными интервалами. Столбцы могут быть сгруппиро-

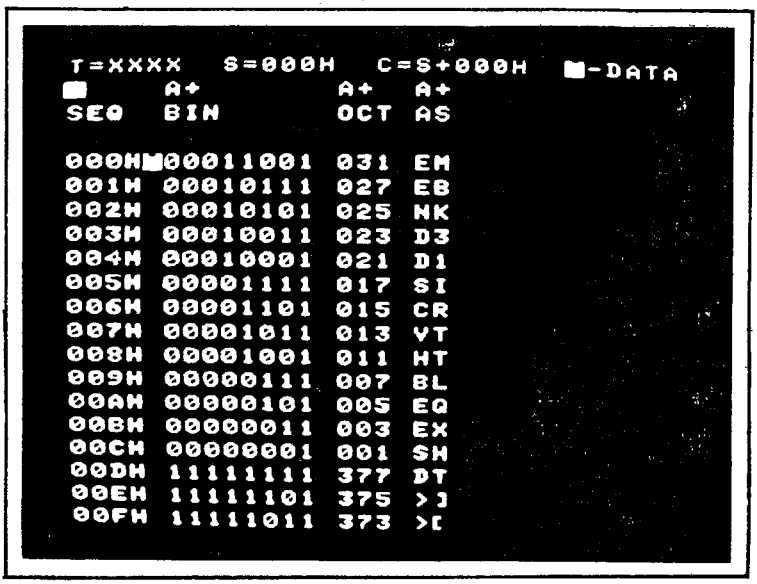


Рис. 3.12. Представление одних и тех же данных в двоичном, восьмеричном, ASCII кодах

ваны в блоки по три, если данные будут считываться в восьмеричном коде, или в блоки по четыре, если они будут считываться в шестнадцатиричной или двоично-кодированной десятичной форме. Имеется возможность перевода представления одних и тех же данных из одной формы в другую (рис. 3.12). Для удобства представления и возможности отладки программ, написанных на языке ассемблера, в некоторых логических анализаторах данные представляются на этом же языке, причем, как правило, адреса изображаются в шестнадцатиричном или восьме-

ричном коде, команды декодируются в абсолютный (шестнадцатиричный или восьмеричный) или мнемонический (ассемблерный) код, а управляющие линии, метки и линии состояний записываются в двоичном коде. На рис. 3.13 показано произвольное место программной трассировки.

Существуют логические анализаторы, воспроизводящие данные в виде карты состояний. В этом режиме на

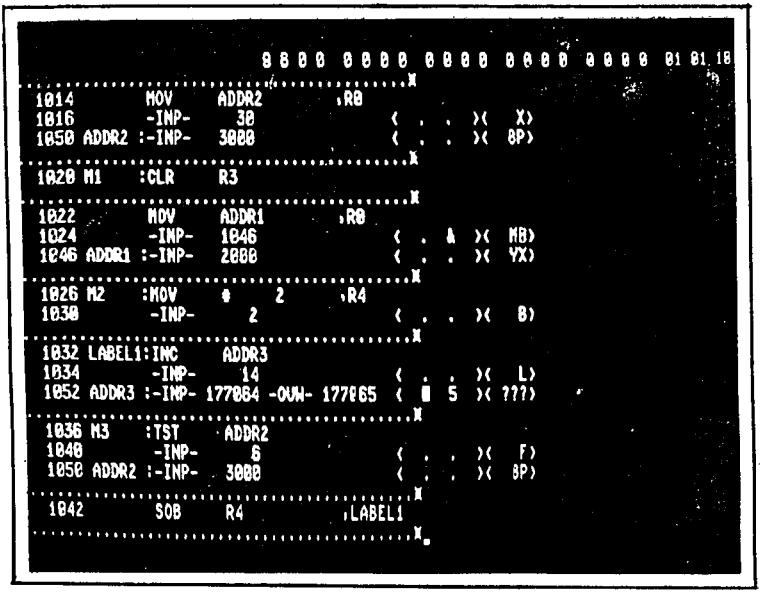


Рис. 3.13. Трассировка участка программы, написанной на языке макроассемблера ДВК «Электроника НЦ-8020»

экране вместо таблицы единиц и нулей воспроизводится матрица 2^N точек. Каждая точка представляет одну возможную комбинацию значений N входных линий, так что любое исходное слово — светящаяся точка. Слово, содержащее все нули, находится в верхнем левом углу экрана; слово, содержащее все единицы,— в нижнем правом углу. Точки взаимосвязаны таким образом, что можно наблюдать последовательность обработки данных. Отрезок линии становится более ярким по мере приближения к новой точке, указывая тем самым направление потока данных. Такая карта состояний облегчает общий

контроль работоспособности схемы с повторяющимся циклом. Этот метод контроля, с одной стороны, позволяет качественно оценить (по относительной яркости точек) повторяемость отдельных адресов в ходе выполнения программы, а с другой — дает возможность весьма быстрого обнаружения искажений программы путем сравнения с «образцовой» картой состояния.

Одним из самых полезных средств индикации является *курсор (экранный указатель)*. Это сплошная вертикальная линия на экране дисплея, которую оператор может перемещать, совмещая с любым требуемым участком изображения. Передвижной курсор используется на временно́й диаграмме для упрощения измерения сравнительных временны́х характеристик, а также для вызова конкретных адресов, что помогает при вычислении. Адрес курсора — положение курсора, представленное в памяти анализатора цифровым способом и воспроизводимое на экране дисплея. Чтобы найти сдвиг по времени между двумя точками изображения, оператор может поочередно поместить курсор в эти две точки и вычислить разность полученных отсчетов его положения.

§ 3.5. Способы управления логическими анализаторами

По мере возрастания функциональной сложности анализатора увеличивается и возможность неправильной установки его органов управления. Для преодоления этого противоречия логические анализаторы выпускаются с клавишным управлением. В логических анализаторах применяют также микропроцессоры, обеспечивающие выполнение функций самоконтроля, вычислений и отображения данных. Раньше оператор запоминал уставки органов управления и режимы запуска механически по положениям переключателей и надписям лицевой панели, в настоящее время в моделях анализаторов с клавишным управлением эти уставки отображаются на экране дисплея либо в формате списка команд, либо в режиме индикации состояния. В обоих случаях задачу контроля текущего режима выполняет микропроцессор.

Современные логические анализаторы снабжены стандартными интерфейсами IEEE-488 (ГОСТ 26.003—80), обеспечивающими диалог ЭВМ с периферийными устройствами. Интерфейсы позволяют осуществлять программи-

Таблица 3.1

Тип логического анализатора (фирма, страна)	Режим работы	Максимальная частота, МГц	Число входных каналов	Емкость памяти, бит/канал	Наличие стандартных интерфейсов	Наличие специализированных зондов	Примечание
1	2	3	4	5	6	7	8
648 («Paratronics», США)	Синхронный	15	48	256	+	—	Наличие микропроцессора
616 («Paratronics», США)	Асинхронный	50	16	1024	+	—	Возможность обнаружения паразитных сигналов
PM3540 («Philirs», Нидерланды)	Синхронный	10	16	512	—	—	Возможность работы в режиме аналогового осциллографа на частотах до 25 МГц
K100D («Biomation», США)	Синхронный Асинхронный	70 100	16 48	1024 10	+	—	Наличие микропроцессора, возможность самоконтроля
LAM4850 («Dolch», ФРГ)	Синхронный Асинхронный	25	48	1024	+	+	Разделение данных, передаваемых по одной магистрали; наличие «эталонного» ЗУ
1240 («Tektronix», США)	Синхронный Асинхронный	50 50 100	18 72 36	2048 512 512	+ + +	—	Применен экран сенсорного типа, позволяющий выбирать режим путем прикосновения к определенным участкам изображения. Позволяет собирать и обрабатывать данные при двух различных частотах синхронизации. Сменные платы позволяют менять число каналов
D132 («B. P. Instruments, США)	Синхронный Асинхронный	70 100 400	64 16 4	512 1024 2048	+	+	Наличие «эталонного» ЗУ

Продолжение табл. 3.1

Тип логического анализатора (фирма, страна)	Режим работы	Максимальная частота, МГц	Число входных каналов	Емкость памяти, бит/канал	Наличие стандартных интерфейсов	Наличие специализированных зондов	Примечание
1	2	3	4	5	6	7	8
К101 D («Biomation», США)	Синхронный Асинхронный	50 100	48	512	+	+	Наличие микропроцессора, «эталонного» ЗУ, мультиметра, частотомера; возможность самоконтроля
308 («Tektronix», США)	Синхронный Асинхронный	20 20	8 8	252 252	—	+	Возможность работы в режиме сигнатурного анализатора и анализатора последовательной передачи данных от 50 до 9600 бод; наличие «эталонного» ЗУ
К500D («Biomation», США)	Асинхронный	500	8	2048	+	+	
1610B («Hewlett-Packard», США)	Синхронный	10	32	64	+	+	Наличие семи уровней последовательностей запуска, определяет время выполнения программы и команды
1611A («Hewlett-Packard», США)	Синхронный	5	32	64	+	+	Ориентирован на отладку микропроцессорных систем
1615A («Hewlett-Packard, США)	Асинхронный Синхронный	20 16	8 24	256 256	+	+	Имеет запуск от ложного сигнала
806 СССР	Асинхронный	10	16	16	—	+	—
821 СССР	»	20	32	16	—	+	—

рование анализаторов от внешних устройств, обработку результатов анализа с помощью ЭВМ, использовать анализатор для дистанционной диагностики, создавать диагностические центры для выполнения и обработки данных с передачей с помощью контрольных измерений телефонного модема (подобные центры обеспечивают использование квалифицированного обслуживающего персонала, машинных методов диагностики для дистанционной локализации неисправностей. Обслуживающий персонал должен только подключить зонды к проверяемому оборудованию).

В табл. 3.1 приведены основные характеристики наиболее распространенных моделей логических анализаторов [12, 13].

§ 3.6. Примеры применения логических анализаторов

Пример 3.1. Рассмотрим поиск места неисправности в контроллере системы регулировки уличным движением, выполненном на базе микропроцессора [14].

Неисправность заключается в том, что на всех светофорах перекрестка шести улиц в начале часов «пик» почти каждые два дня одновременно загорается красный свет. При этом в счетчике команд процессора содержится, например, адрес ячейки 37416, а ячейки с таким адресом нет ни в оперативной, ни в постоянной памяти контроллера.

Для устранения этой неисправности в первую очередь необходимо определить точку в программе, с которой процессор делает переход по отсутствующему адресу. Это возможно с помощью анализатора логических состояний. В качестве кода запуска задается адрес ячейки 37416 и выбирается режим прекращения запоминания данных по коду запуска. После этого анализатор подключается к магистрали процессора. Как только контроллер уличного движения сработает неправильно, анализатор логических состояний запомнит и воспроизведет участок трассы выполняемой программы предшествующего перехода к ячейке с адресом 37416. Эта информация будет воспроизводиться на экране.

Контроллер всегда переходит к ячейке с адресом 37416 от ячейки с адресом 2173. В этой ячейке должна храниться команда И — в той части программы, которая изменяет циклограмму работы светофоров регулировки уличного движения применительно к условиям вечерних часов «пик». Конкретно содержимое ячейки с адресом 2173 вызывается в том случае, если в одном из светофоров горит желтый свет, т. е. ровно в 16 ч. 40 мин., когда начинается переход на циклограмму часов «пик». Поскольку это происходит примерно раз в два дня, ошибка только тогда и проявляется. Проверка программной документации показывает, что в ячейке с адресом 2173 хранится команда ПЕРЕХОД вместо команды И из-за ошибочного разряда в программном ПЗУ. Как только процессор получает команду ПЕРЕХОД, он воспринимает содержимое следующих двух ячеек как адрес 37416. А поскольку пустые или отсутствующие ячейки ПЗУ процессор воспринимает как команду СТОП, он просто останавливается по адресу 37416 и ждет перезапуска.

Пример 3.2. В микропроцессорной системе, содержащей микропроцессор, ОЗУ, ПЗУ, в нормальном режиме работы происходит зацикливание, в то же время в однотактном режиме программа всегда выполняется правильно.

В программе инициализации, расположенной в ячейках с адресами 0—137, предусмотрен переход в ячейку с адресом 4052. Но вместо перехода в ячейку 4052 программа возвращается в ячейку с адресом 52. Из-за неисправности нагрузочного резистора разряда, соответствующего адресу 4000 процессора, уровень на этой адресной линии нарастает так медленно, что ПЗУ воспринимает адрес 4052 как адрес 52 и возвращается в эту ячейку. А поскольку в ячейке с адресом 52 содержится команда перехода в ячейку с адресом 57, система переходит в ячейку 57 и происходит зацикливание по программному счетчику. Однако при проверке системы в однотактном режиме неисправная адресная линия имеет достаточно времени для нарастания уровня напряжения и переход в ячейку с адресом 4052 выполняется правильно.

Эту неисправность обнаруживают с помощью осциллографа с запуском от анализатора логических состояний. Для этого необходимо определить ошибочный цикл программы, неверный переход в программе, что можно сделать либо с помощью анализатора логических состояний, подключившись щупами к адресной магистрали и просмотрев ход выполнения программы, либо с помощью визуального вывода карты состояний, либо путем трассировки программы. После того как установлено, что программа из ячейки с адресом 63 переходит в ячейку с адресом 52 вместо перехода в ячейку с адресом 4052, на логическом анализаторе устанавливается запуск по коду 63, по приходе которого осуществляется запуск осциллографа.

Глава 4

ГЕНЕРАТОРЫ СЛОВ И КОМПЛЕКСЫ ДИАГНОСТИРОВАНИЯ

§ 4.1. Генераторы слов

Генераторы слов (генераторы данных, генераторы тестовых последовательностей) — приборы, предназначенные для формирования и подачи входных воздействий на проектируемую или диагностируемую дискретную систему. Совместно с логическими анализаторами генераторы слов образуют системы подачи внешних стимулирующих сигналов и сбора ответных реакций как микропроцессорных модулей, так и схем произвольной логики. Генераторы слов применяются для тестирования дискретных систем, для эмуляции отсутствующих (например, еще не спроектированных или не изготовленных) дискретных устройств проектируемой системы. Известно несколько типов генераторов слов.

Примечание. Ниже не будут рассматриваться генераторы слов, входящие в комплексы диагностирования.

Генераторы слов характеризуются числом каналов, емкостью памяти, частотой подачи воздействий, называемой *тактовой*, способами подачи данных и формирования входных воздействий.

Обобщенная структура генератора слов (рис. 4.1) включает в себя *ЗУ*, драйверы *Др*, устройство управления *УУ*, генератор синхросигналов *ГСС*, устройство управления вводом *УУВ*, дисплей *Д*, клавиатуру *Кл*.

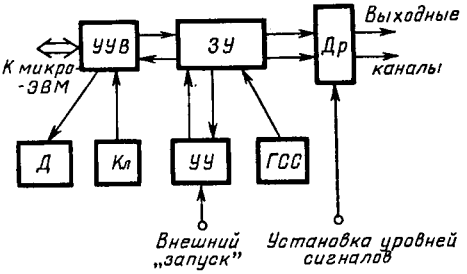


Рис. 4.1. Обобщенная структура генератора слов

Последовательность входных приборов, которую необходимо подать на систему диагностирования или проектируемую систему, заносится в *ЗУ*. Информация вводится в *ЗУ* либо с клавиатуры *Кл*, при этом дисплей *Д* используется как средство отображения вводимых или записанных в *ЗУ* данных, либо через стандартный интерфейс из памяти микро-ЭВМ. Устанавливаются частота тактирования, с которой входные наборы будут подаваться на систему диагностирования, уровни сигналов, соответствующие «1» и «0» (обеспечиваются драйверами, являющимися управляемыми источниками напряжения), режим цикличности подачи воздействия (один цикл, *n* циклов, непрерывный). К выходным каналам подключается объект диагностирования. Сигнал ПУСК подается либо с клавиатуры, либо от микро-ЭВМ, либо извне от объекта диагностирования, логического компаратора, либо от другого генератора слов. После сигнала ПУСК данные считываются из памяти и через драйверы *Др* поступают на выходные каналы генератора слов с заданной частотой.

Число выходных каналов — один из основных параметров генератора слов, так как при тестировании дискретных систем необходимо, чтобы на все входы были поданы вполне определенные последовательности сигналов. В ряде случаев можно вести тестирование разбиением всех входов на группы и подачу воздействий — последовательно, отдельно на каждую группу входов. Однако при этом

усложняется построение тестов, удлиняется время диагностирования. Число выходных каналов известных генераторов слов колеблется от 2 до 80 (2, 8, 16, 32, 64, 80).

Тактовая частота — следующий важный параметр генератора слов. Ряд неисправностей дискретной системы, как физических, так и нефизических, проявляется только на высоких частотах функционирования системы. Для обнаружения этих неисправностей необходимо вести тестирование на максимально возможной для конкретной проверяемой системы тактовой частоте. Максимальная частота известных генераторов слов составляет 50 МГц, тактовая частота изменяется от сотен герц до десятков мегагерц. Это свойство позволяет испытывать одним и тем же генератором слов различные системы.

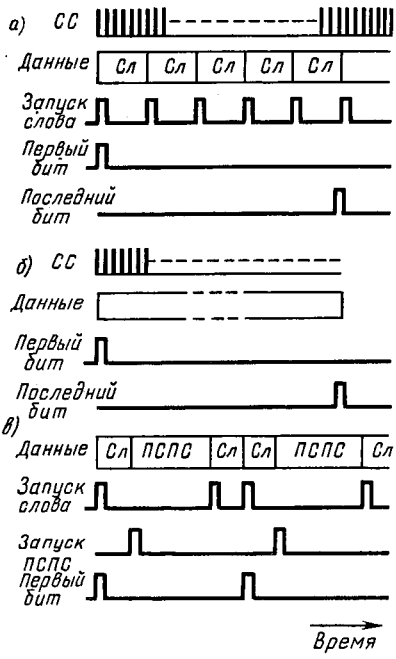


Рис. 4.2. Временные диаграммы работы генератора слов последовательного типа 8018А в различных режимах:
а — в режиме слов; *б* — в режиме данных; *в* — в смешанном режиме

Емкость памяти известных генераторов слов содержит от 16 до 2048 слов. Память реализуется на сверхоперативных БИС ЗУ, например микросхемах серии КР132РУ4, имеющих организацию 1×1К и время выборки 25 нс.

Драйверы подключают генератор слов к испытуемым объектам, выполненным по различной технологии (ТТЛ, ТТЛ-ДШ, К-МОП, ЭСЛ) и имеющим различные нагрузочные характеристики. Некоторые драйверы имеют выходы с тремя состояниями: высокого и низкого напряжений и высокого импеданса. В состоянии высокого импеданса выходной канал генератора слов «отключен» от объекта диагностирования. Эта особенность обеспечи-

вает непосредственное подключение генератора слов к двунаправленным магистралям.

По способу подачи воздействий генераторы слов подразделяются на генераторы слов последовательного и параллельного кодов. Существуют генераторы слов, имеющие тот же режим работы, что и генераторы последовательного и параллельного кодов. Режим последовательного кода используется при тестировании или имитации систем с последовательной передачей данных, а режим параллельного кода — во всех остальных случаях.

Генераторы слов последовательного кода имеют один или два информационных выходных канала, синхровыводы (первый бит, последний бит, синхросигнал) для обеспечения различных интерфейсов с объектом тестирования. Отдельные генераторы последовательного кода имеют такие ценные качества, как генерация псевдослучайной последовательности сигналов (ПСПС).

На рис. 4.2 (*Сл* — слова; *ПСПС* — псевдослучайные последовательности сигналов) показаны некоторые возможности генераторов слов последовательного кода 8018А фирмы «Hewlett—Packard». Этот генератор слов имеет три режима работы: слов, данных и смешанный. В режиме слов (рис. 4.2, *а*) могут генерироваться последовательности из нескольких слов, в режиме данных (рис. 4.2, *б*) — последовательности сигналов; в смешанном режиме (рис. 4.2, *в*) — последовательности слов и псевдослучайные последовательности сигналов.

По способу реализации устройства управления можно выделить три типа генераторов слов: 1) с буферной памятью (в этих генераторах данные из памяти считываются последовательно, начиная от конкретного начального адреса и кончая заданным конечным адресом ЗУ. Если необходимо организовывать циклы, то за конечным адресом на регистр адреса ЗУ заносится начальный адрес. Эти генераторы наиболее простые); 2) с управляющей памятью (память в этих генераторах делится на две части — данных и команд, имеющих общее управление и общий регистр адреса. Данные и команды считываются одновременно. Команды поступают на дешифратор команд, определяющий, что́ необходимо сделать со считанными данными. Подобная реализация устройства управления обеспечивает максимальное быстродействие); 3) с алгоритмическим генерированием последовательностей (основу этих генераторов составляет микропрограммируемый процессор, как правило, специализированный. Генератор

имеет память данных и память микропрограмм, работающих автономно. Частным случаем генератора с алгоритмическим генерированием последовательностей является генератор, у которого отсутствует память данных. Эти генераторы используют для контроля устройств с регулярной структурой, например БИС ОЗУ. Генераторы, основу которых составляют память данных и процессор или микропроцессор, позволяют эффективно формировать как алгоритмические, так и неалгоритмические последовательности слов или сигналов).

В микропроцессорных системах применяют синхронные и асинхронные магистрали (управляющие магистрали — асинхронные, магистрали адресов и данных — синхронные). Для обеспечения таких возможностей в генераторы слов вносятся команды, позволяющие считывать данные из памяти либо при поступлении сигнала внутреннего генератора, либо извне от объекта контроля. В первом случае обеспечивается режим, при котором очередной входной набор подается на объект диагностирования при поступлении синхросигнала либо от внутреннего, либо от внешнего генератора синхросигнала, либо от оператора с клавиатуры, а последовательность входных наборов определяется запрограммированным алгоритмом и не зависит от реакций объекта диагностирования. Во втором случае очередной входной набор может зависеть как от запрограммированного алгоритма тестирования, так и от реакции объекта на входную последовательность.

Программировать работу генератора слов в «1» и «0» утомительно, поэтому средства автоматизации программирования крайне желательны. Кроме того, при использовании генератора слов в процессе отладки аппаратуры или комплексной отладки микропроцессорной системы необходимо задавать последовательности входных наборов, подаваемых на систему, с применением языков, на которых велось проектирование. Например, если при проектировании группа каналов использовалась как МД или МА, то эти каналы должны таким образом и обозначаться. Если задание на проектирование интерфейса было задано в виде временно́й диаграммы, то при проектировании генератора слов для проверки этого интерфейса необходим язык временны́х диаграмм. Современные генераторы слов имеют в своем программном обеспечении такие средства программирования, как трансляторы, редакторы текстов, отладчики. В табл. 4.1 приведены параметры некоторых генераторов слов [15, 16].

Таблица 4.1

Тип генератора слов (фирма, страна)	Режим работы	Максимальная тактовая частота, МГц	Число выходных каналов	Емкость ЗУ, бит/канал	Наличие стандартных интерфейсов	Уровни выходного сигнала	Примечание
Г5-80 (СССР)	Последовательный	50	—	—	—	ЭСЛ	Наличие микроЭВМ
	Параллельный	50	4—16	2048	Есть	ТТЛ	—
8018А («Hewlett-Packard», США)	Последовательный	50	2	1024	»	ТТЛ ЭСЛ КМОП	Возможность генерации ПСПС
8170А («Hewlett-Packard», США)	Параллельный	2	8/16	1024/512	»	ТТЛ КМОП	Наличие микропроцессора 6800
IGA («Rohde & Schwarz», ФРГ)	Последовательный	1/1,8	16/32	1024	—	ТТЛ КМОП	Наличие кассетного накопителя памяти на магнитной ленте

§ 4.2. Обобщенная структура комплексов диагностирования

Комплексы диагностирования объединяют возможности логических анализаторов и генераторов слов; способны подавать входные воздействия на диагностируемую систему, собирать и анализировать ответные реакции системы. Они представляют собой не простое объединение любых логического анализатора и генератора слов, а имеют режим, при котором генератор слов и логический анализатор, входящие в состав комплекса, функционируют как единое целое под общим управлением микропроцессора, с общим программным обеспечением, с согласованными по времени распространения сигналами.

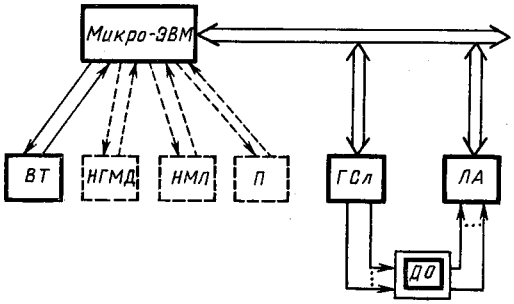


Рис. 4.3. Обобщенная структура комплекса диагностирования

Комплексы диагностирования используют главным образом при проектировании как микропроцессорных, так и других дискретных систем, а также для проверки работоспособного состояния и диагностики неисправностей систем при их производстве и эксплуатации. Однако комплексы диагностирования уступают средствам диагностирования, предназначенным для производства и эксплуатации систем, в производительности, массе и габаритных размерах.

Обобщенная структура комплекса диагностирования представлена на рис. 4.3. В состав комплекса входят *микро-ЭВМ* с периферией, устройства генератора слов *ГСл* и логического анализа *ЛА*. Устройство *ГСл* совместно с микро-ЭВМ, программным обеспечением и

периферией выполняет функции прибора — генератора слов. Устройство *ЛА* совместно с микро-ЭВМ, программным обеспечением и периферией выполняет функции контрольно-измерительного прибора — логического анализатора.

Примечание. Ниже устройства *ЛА* и *ГСл* для простоты будем называть логическим анализатором и генератором слов.

Микро-ЭВМ подготавливает тестовые наборы, загружает и настраивает на определенный режим работы *ГСл* и *ЛА*, анализирует результаты тестирования, обрабатывает информацию о поведении объекта диагностирования *ОД*, представляет информацию о ее поведении на языках, используемых при проектировании, осуществляет диалог с человеком.

Генератор слов в зависимости от исходных данных, полученных от *микро-ЭВМ*, выдает на заданной частоте входные воздействия на *ОД*. Некоторые генераторы слов умеют подавать входные воздействия, зависящие от реакции объекта. Логический анализатор собирает данные о поведении системы в режиме реального времени. Генератор слов и логический анализатор могут находиться как в одном, так и в отдельных блоках. Для взаимодействия с человеком в комплекс включен видеотерминал *ВТ*.

В комплекс могут включаться либо накопитель на гибких магнитных дисках *НГМД*, либо накопитель на магнитной ленте *НМЛ*. Последние, увеличивая емкость памяти, существенно расширяют функции комплексов: появляется возможность организации библиотеки тестовых программ, использования языков высокого уровня, проведения автоматических процедур диагностирования, запоминания процедур отладки и др.

Принтер *П* включается в комплекс для документирования отладки или приемосдаточных испытаний или контроля и диагностирования.

§ 4.3. Комплекс диагностирования «Электроника НЦ-603»

Технические характеристики комплекса «Электроника НЦ-603» следующие [17]:

Число каналов подачи тестовых воздействий	64(128)
Число каналов считывания ответных реакций	32(64)

Максимальная программируемая частота, мГц:	
подачи тестовых воздействий	до 10
считывания ответных реакций	до 10
Емкость памяти на канал, бит	1024
ТТЛ-уровни с нагрузочной способностью на выходе комплекса, мА	до 60
Напряжение на входе комплекса, В	—10 до +10
Ток на входе комплекса, мА . . .	2
Мощность потребления со штатной периферией, кВт	1,5
Габаритные размеры (без периферии), мм	1200×800×1020
Масса (без периферии), кг	70
Питание от однофазной сети тока напряжением	220 $В^{+10\%}_{-15\%}$, 50 Гц

Комплекс содержит микро-ЭВМ типа «Электроника НЦ-03Д», генератор слов и логический анализатор. В состав микро-ЭВМ входят ОЗУ емкостью 40К 16-разрядных слов, видеотерминал типа VT-340 или РИН-609, накопитель на гибких магнитных дисках «Электроника ГМД-70», принтер типа DZM-180.

В микро-ЭВМ предусмотрена возможность работы с фотоэлектрическим устройством FS-1501 и перфоратором ленточным ПЛ-150 через контроллер перфоленточных устройств. Генератор слов и логический анализатор образуют специализированный терминал — тестер, предназначенный для выдачи тестовых воздействий на объект диагностирования и восприятия его ответных реакций.

Генератор слов — устройство, способное подавать на объект диагностирования входную последовательность длиной до 1024 64-разрядных слов с программируемой частотой от 250 кГц до 10 мГц. Имеется возможность циклической подачи входной последовательности. Уровни значений сигналов подачи входных воздействий соответствуют ТТЛ; при дополнительных источниках питания уровни могут регулироваться в пределах от —10 до +10 В. В состав генератора слов входят ЗУ емкостью 1024 32-разрядных слов, соединенное с микро-ЭВМ и *драйверами*, а также регистры адреса, начала и конца и устройство управления *УУ*.

Микро-ЭВМ готовит тестовые воздействия порциями по 1024 32-разрядных слов, засылая их в память, и дает команду устройству управления. После этого тестовые наборы считываются потактно из памяти и посылаются на объект диагностирования. Регистр начала указывает,

с какого адреса (тестового набора) надо начать посылку, а регистр конца — каким адресом (тестовым набором) закончить. Регистры начала и конца позволяют организовать циклическую подачу входных воздействий, частным случаем которого является шаг. Тактовые сигналы генератора слов и один из программируемых его выходов выведены на высокочастотные розетки на внешней панели тестера.

Логический анализатор — устройство, предназначенное для считывания и запоминания до 1024 значений сигналов 32 линий диагностирования с максимальной частотой до 10 МГц. Запуск логического анализатора осуществляется при возникновении заданного, запрограммированного события (комбинации значения входных сигналов) на его входе. Останов логического анализатора происходит либо при переполнении памяти, либо при возникновении определенного события на его входе, либо по команде оператора. Кодовая комбинация, по которой осуществляется запуск или останов логического анализатора, может быть определена как на всех 32 входах, так и на любом их подмножестве.

Логический анализатор работает в режимах тестера, синхронном, асинхронном. В режиме тестера он работает синхронно с генератором слов. В этом режиме съем реакций объекта диагностирования производится синхронно с подачей входных наборов, при этом задержка между началом такта и моментом съема регулируется вручную путем введения аппаратурных задержек в цепи синхронизации между логическим анализатором и генератором слов.

В асинхронном режиме частота сканирования входов задается внутренним программно-управляемым генератором. В синхронном режиме запись в ЛА входной информации производится только при изменении сигнала из «0» в «1» на специальном его входе. Запись в ЛА осуществляется после его запуска, т. е. после появления заданной комбинации 32-разрядного кода на его входах.

Логический анализатор содержит память, компараторы уровней, устройство управления, устройство запуска и останова. Емкость памяти 1024 32-разрядных слов выполнена на БИС ЗУ серии КР132 РУ4. На входе памяти установлены аналоговые компараторы с регулируемым порогом срабатывания.

Устройство запуска и останова соединено с компараторами уровней и отслеживает входные наборы. Задав

определенные комбинации значений входных наборов, устройство позволяет установить момент записи входных наборов в память и момент останова записи. Например, можно запустить ЛА при появлении определенной команды или какой-либо помехи на одном из входов. После окончания записи микро-ЭВМ позволяет вывести информацию в удобном для интерпретации виде: либо в виде таблицы состояний, либо в виде временны́х диаграмм, либо в виде трассировки программ. Если представить себе поток данных в виде «1» и «0», то ЛА позволяет «высветить», «сфотографировать» определенную часть этой последовательности. Логический анализатор дает возможность увидеть, как ведет себя дискретная система, в частности микропроцессорная, в реальных условиях.

Имеется два варианта конструктивного исполнения комплекса: стоичный и настольный.

В состав программного обеспечения (ПО) комплекса входят: дисковая операционная система ИНТОЛ; монитор; программное обеспечение режима подготовки тестовых и испытательных программ, автоматического и полуавтоматического диагностирования, генератора слов, логического анализатора, управления аппаратурным уровнем, самоконтроля комплекса; тестовые программы пользователей.

Программное обеспечение режима подготовки тестовых и испытательных программ предназначено для их ввода в комплекс, редактирования, организации архива, вывода на листинг или экран дисплея. Тестовые и испытательные программы вводятся в комплекс и выводятся из него в виде символьного текста, записанного соответственно на языках ЯТП и ДИРЕКТИВ. В тестовых программах наборы значений входных сигналов должны подаваться на объект диагностирования. Испытательные программы определяют последовательность процедур диагностирования, которую необходимо выполнить автоматически. Последовательность этих процедур устанавливает требуемый порядок выполнения тестовых программ, различные способы подачи тестовых воздействий и анализа ответных реакций объекта диагностирования, вывода задаваемых сообщений оператору и т. д.

Программное обеспечение режима автоматического диагностирования обеспечивает автоматическое выполнение последовательности директив, оформленных в виде испытательных программ.

Программное обеспечение режима полуавтоматического диагностирования производит автономное выполнение отдельных процедур диагностирования по подаче входных воздействий на объект диагностирования, по съему ответных реакций объекта и по их анализу, формированию и выводу сообщений оператору о результатах диагностирования. Задание каждой процедуры производится с помощью отдельной директивы, вводимой оператором.

Программное обеспечение режима генератора слов позволяет автоматически подавать на объект диагностирования входные последовательности порциями по 1024 набора.

Программное обеспечение режима логического анализатора устанавливает в начальное состояние логический анализатор, обрабатывает информацию, записанную в его память, результаты выводит на экран дисплея либо в виде временны́х диаграмм, либо в виде трассировки программ.

Программное обеспечение режима управления аппаратурным уровнем производит автономное выполнение отдельных элементов процедур диагностирования, задаваемых оператором с помощью директив.

Программное обеспечение режима самоконтроля комплекса осуществляет проверку технического состояния комплекса и отдельных его устройств.

Тестовыми программами (ТП) пользователя являются тесты, оформленные в виде таблиц или программ, для проверки работоспособности и поиска места неисправности систем пользователя комплекса «Электроника НЦ-603». Программы представляют собой набор команд, организованных построчно, причем каждая строка содержит только одну команду.

Системное программное обеспечение хранится на системном диске, а тестовые программы пользователя — на пользовательских дисках.

Язык ЯТП предназначен для описания и ввода в комплекс тестовых наборов, содержащих входные наборы и эталонные ответные реакции разрабатываемой тестовой программы.

Подача входных наборов на объект диагностирования происходит потактно. Началом одного такта и концом предыдущего считается момент перехода от предыдущего входного набора к настоящему.

Управление комплексом «Электроника НЦ-603» осуществляется с помощью директив, вводимых с клавиатуры дисплея. Все директивы разбиваются на группы в соответствии с режимами работы комплекса. Формат директивы включает в себя трехсимвольное имя, начинающееся с буквы D, и признак конца директивы — символ LF (конец строки).

К режиму подготовки тестовых программ относятся директивы: транслировать ТП; редактировать канал ТП; исключить строки ТП; вставить строки ТП; создать ТП; сформировать эталонные ответные реакции ТП; печатать ТП; записать ТП в архив; считать ТП из архива.

К режиму полуавтоматического диагностирования относятся директивы: задания устройства вывода сообщений и частоты тактовых сигналов генератора слов; пошаговая подача входных воздействий; автоматическая и циклическая подачи входных наборов; останов циклической подачи входных наборов; циклическая подача входных наборов во временны́х интервалах; циклическая подача входных наборов заданное число раз; статическое и динамическое диагностирования; динамическое диагностирование с учетом разброса фронтов; сравнение в числовом диапазоне; задание номера временно́й диаграммы; анализ временны́х диаграмм; временна́я диаграмма; и т. д.

Директивы позволяют:

задать устройство, на которое будут выводиться сообщения; тактовую частоту; ТП и их участки, которые будут поданы на объект диагностирования в пошаговом режиме; ТП и их участки, которые будут поданы на объект диагностирования в автоматическом режиме: ТП и их участки, которые будут поданы на объект диагностирования в непрерывной последовательности; число циклов подачи ТП;

остановить циклически подаваемую последовательность тестовых кодов на объект диагностирования;

осуществить статическое тестирование, при котором на объект диагностирования подаются входные наборы каждого из заданных участков ТП с частотой порядка 0,5 кГц и после подачи каждого входного набора значения на выходах объекта диагностирования считываются и сравниваются с их эталонными значениями, заданными в ТП;

осуществить динамическое диагностирование, при котором на заданной тактовой частоте подаются входные наборы участка ТП, считываются и запоминаются реак-

ции объекта диагностирования, которые затем сравниваются с эталонными с учетом отклонения значений задержек реальных цепей;

проанализировать зафиксированные с помощью логического анализатора значения исследуемых сигналов в соответствии с алгоритмом, указываемым в виде параметра директивы;

выполнить в тестерном режиме задаваемые участки ТП и вывести в виде сообщения временные диаграммы, получаемые на входных и выходных выводах объекта диагностирования.

Программное обеспечение комплекса содержит директивы других режимов; кроме того, оно допускает расширение своих функций.

§ 4.4. Комплекс диагностирования DAS 9100 фирмы «Tektronix»

В состав комплекса диагностирования DAS 9100 [18] входят микро-ЭВМ на базе микропроцессора Z-80, дисплей на ЭЛТ, клавиатура, генератор слов и логический анализатор. Комплекс содержит ряд печатных плат, каждая из которых специализируется для реализации общего управления работой комплекса, сбора данных, генерации тестовых последовательностей, передачи цифровых или видеосигналов. Эти платы размещаются в одном корпусе размером $24 \times 43 \times 60$ см. Модульный принцип, заложенный при проектировании комплекса, позволяет менять его конфигурацию, устанавливая в шести его посадочных местах различные аппаратурные модули — платы генерации тестовых последовательностей или сбора данных. Технические возможности отдельных плат представлены в табл. 4.2.

Универсальность и удобство настройки комплекса на заданную конфигурацию достигаются диалоговым режимом работы с программным обеспечением. При этом способе используются программы-утилиты, каждая из которых предназначается для выполнения конкретного вида работы. Конкретную программу выбирают по *меню команд*.

При включении питания комплекса пользователю на экране дисплея представляется меню, информирующее о платах, установленных в приборе, их работоспособности. Если какая-либо плата неработоспособна, то

Таблица 4.2

Платы	Номер модели	Примечание
Генератора слов (платы генерации тестовых последовательностей; максимум 80 каналов)	91P16	Выдает сигналы данных по 16 выходным каналам плюс два строб-сигнала и два синхросигнала на частоте 25 МГц; микропрограммируемая
	91P32	Расширяет возможности генератора слов; содержит 32 выходных канала и четыре строб-сигнала; должна использоваться с платой 91P16
Логического анализатора (платы сбора данных; максимум 104 канала)	91A32	Работает с многоразрядной магистралью, собирает данные по 32 каналам с частотой 25 МГц синхронно или асинхронно, имеет емкость памяти 512 бит на один канал
	91A08	Собирает данные по восьми каналам с частотой 100 МГц синхронно или асинхронно, имеет емкость памяти 512 бит на один канал; предназначается для отладки аппаратурных средств
	91A04	Собирает данные либо по четырем каналам с частотой 330 МГц синхронно или асинхронно, либо по двум каналам с временной разрешающей способностью 1,5 нс (660 МГц), имеет емкость памяти 2048 бит на канал; предназначается для отладки высокоскоростных логических схем
	91AE04	Используется с платой 91A04, при этом добавляется еще четыре канала сбора данных с частотой 330 МГц или два канала с разрешением 1,5 нс (660 МГц)

пользователь может запустить более сложную диагностическую программу, нажав соответствующую кнопку.

С помощью *меню описания каналов* указываются последовательность и формат воспроизведения канальной информации, а также уровни логических сигналов для каналов.

С помощью *меню запуска* пользователь указывает, в какие моменты требуется собирать данные.

Все меню автоматически меняются в соответствии с возможностями, реализованными в системе в данный момент времени. Например, меню запуска позволяет

низкоскоростной (с частотой до 25 МГц) плате 91А32 запускать плату 91А08 сбора аппаратурных данных.

С помощью *меню определения мнемоники* пользователь может присваивать различным логическим комбинациям, которые появляются на определенных каналах, мнемонические обозначения, соответствующие этим комбинациям; примерами служат мнемонические обозначения команд и сигналов для различных микропроцессоров или для интерфейса IEEE-488.

Меню команд генератора слов дает оператору возможность выбирать простые программы, которые путем многократных циклов формируют различные тест-коды для подачи на входы и тестирования проверяемого изделия. С этим меню связано *меню строб-сигналов*, которое позволяет пользователю определять до десяти линий в качестве строб-сигналов, дает возможность, например, моделировать с помощью комплексов различные магистральные структуры.

Чтобы обеспечить передачу данных от комплекса DAS 9100 для воспроизведения и анализа в другом месте либо для запоминания тест-кодов и мнемонического обозначения для последующего использования, предусмотрено *меню ввода — вывода*.

Модульность комплекса DAS 9100 достигается организацией магистральной структуры прибора и микропрограммного системного обеспечения. Поскольку меню команд и функциональные возможности комплекса зависят от плат, установленных в данный момент времени, микропрограммное обеспечение, информирующее центральный процессор о возможностях платы и настраивающее соответственно меню команд, реализовано на каждой плате в ПЗУ объемом 8К байт.

Генератор слов комплекса — специализированный микропрограммируемый процессор, формирующий выходные тестовые последовательности с помощью процедуры трехуровневой конвейеризации. Плата 91Р16 содержит все схемы управления генерацией тестовых последовательностей и позволяет выдавать сигналы по 16 каналам и два независимых строб-сигнала. Структура этой платы представлена на рис. 4.4. Компонентами платы 91Р16 являются микропрограммная память *ПМ* емкостью 256×13 бит, управляющая логика *ЛУ*, мультиплексор инструкций *МИ*, стек 16×8 бит, векторная память *ПВ* емкостью 256×16 бит и схемы стробирования *СхСб*. Платы расширения увеличивают емкость памяти генератора слов

и рассчитаны на 32 дополнительных канала и четыре строб-линии каждая. В прибор можно устанавливать максимум две платы расширения, функционирующие под управлением основной платы.

Для работы с высокоскоростными магистральными структурами и обеспечения тестирования в утяжеленных и граничных режимах генератор слов должен быть в состоянии формировать и выдавать многочисленные тестовые последовательности быстро и четко. Устройство управления генератора слов выбирает ячейки векторной

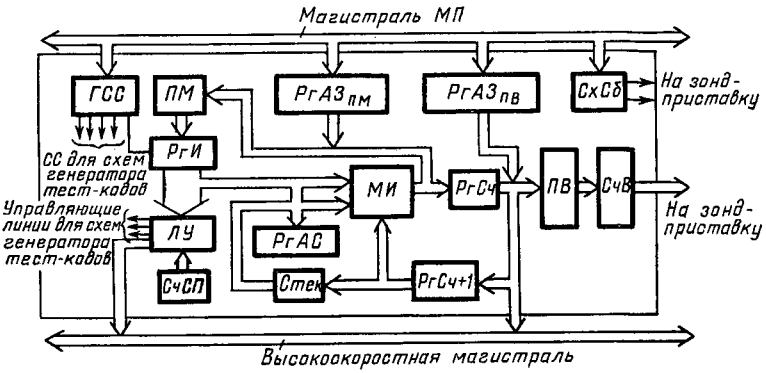


Рис. 4.4. Структура основной платы генератора слов:

ПМ — микропрограммная память; $РгАз_{ПМ}$, $РгАЗ_{ПВ}$ — регистры адреса загрузки микропроцессорной и векторной памяти; *СхСб* — схема стробирования; *ГСС* — генератор синхросигналов; *РгИ* — регистр инструкций; *МИ* — мультиплексор инструкций; *РгСч* — регистр счетчика; *ПВ* — векторная память; *СчВ* — выходной счетчик; *ЛУ* — управляющая логика; *РгАС* — регистр адреса стека; *СчСП* — счетчик событий/повторений; *РгСч+1* — регистр программного счетчика

памяти по алгоритмам, предусматривающим передачи управления, цикла и отсчеты событий, а также с помощью подпрограмм, реализующих последовательности установления связи по магистрали. Поэтому векторная память емкостью 256 слов может использоваться для генерации тестовых последовательностей, имеющих длину несколько тысяч слов и повторяющихся неопределенное число раз. В то же время фактические программы, применяемые для генерации этих тестовых последовательностей, имеют размер в среднем всего лишь 10—15 инструкций.

Генерация последовательностей осуществляется с помощью высокоскоростных аппаратурных средств гене-

ратора. Благодаря этому DAS 9100 может выдавать действительный тестовый набор каждые 40 нс.

Линии строб-сигналов генератора слов в сочетании с выходными каналами позволяют моделировать различные магистральные структуры. Например, при разработке контроллера на базе микропроцессора последовательность слов можно представить в формате МА и МД, а строб-сигналы использовать в качестве управляющих линий, чтобы моделировать внешние функции микропроцессоров. Это делается с помощью набора инструкций, который

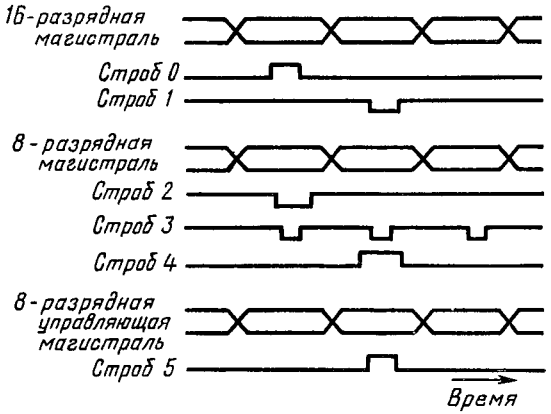


Рис. 4.5. Временна́я программа моделирования строб-сигналов

знаком разработчику, что исключает необходимость работы с различными языками ассемблера. Передние и задние фронты входных строб-сигналов можно программировать в пределах цикла обработки данных, как показано на рис. 4.5, а все параметры строб-сигналов устанавливать с помощью меню строб-сигнала.

Программируемые строб-линии генератора слов позволяют моделировать управляющие линии проектируемой системы. Строб-сигналы выдаются в пределах цикла тактовой частоты длительностью 40 нс.

Для обеспечения подачи строб-сигналов и сигналов данных на объект диагностирования к генератору слов подключаются специальные зонды-приставки двух типов: для подключения к ТТЛ- и МОП-схемам и для ЭСЛ-схем.

Каждую выходную линию можно перевести в режим с тремя состояниями. Приставка-зонд имеет десять выходов: восемь линий данных, одну линию строб-сигнала и одну линию синхросигнала. Кроме того, приставка имеет две линии опорных сигналов, определяющих высокий и низкий уровни выходных сигналов данных.

Платы сбора данных, входящие в состав прибора, разработаны таким образом, что могут выполнять различные функции при отладке аппаратурных и программных средств. Имеется три модуля сбора данных, которые можно комбинировать в разных вариантах: до 96 каналов на предельной частоте 25 МГц, до 32 каналов на максимальной частоте 330 МГц и до восьми каналов на максимальной частоте 660 МГц (временно́е разрешение 1,5 нс).

Пользователи комплекса DAS 9100 могут устанавливать в него самые разнообразные сочетания плат сбора данных, предназначенных для отладки аппаратурных и программных средств, при максимальном общем числе каналов 104.

При комплексной отладке микропроцессорных систем для отыскания неисправностей необходимо уметь устанавливать соответствия поведения системы на различных уровнях абстрактного представления. Например, если обнаружена ошибка на программном уровне, то в ряде случаев следует знать, как ведет себя аппаратура на конкретном участке программы. Для этих целей в приборе введен режим, при котором высокоскоростные платы запускаются от низкоскоростных при возникновении определенных событий. На основе собранных данных высокоскоростными платами микропроцессор восстанавливает картину фактических событий и выводит их на дисплей. На рис. 4.6 показан пример взаимосвязи аппаратурных и программных событий [18].

Платы сбора данных имеют различные режимы запуска: по кодовому слову, по счету событий, от помехи, от квалификаторов синхросигналов.

По особому заказу с комплексом DAS 9100 может быть поставлен накопитель на магнитной ленте DC-100, емкость памяти которого 16K байт. Этот накопитель позволяет сохранять тестовые последовательности, используемые для отладки изделия в процессе разработки, с тем чтобы в дальнейшем применять их для тестирования и проверки работоспособности готового изделия после его внедрения в производство. С помощью накопителя DC-100 можно расширить возможности комплекса по формам

представления информации: вводя соответствующий дизассемблер, можно представлять трассировку программ на языке микропроцессора проектируемой системы.

В комплексе DAS 9100 могут быть введены аппаратурные средства ввода — вывода для работы с интерфейсами RS-232 и IEEE-488. Это позволяет включать комплекс

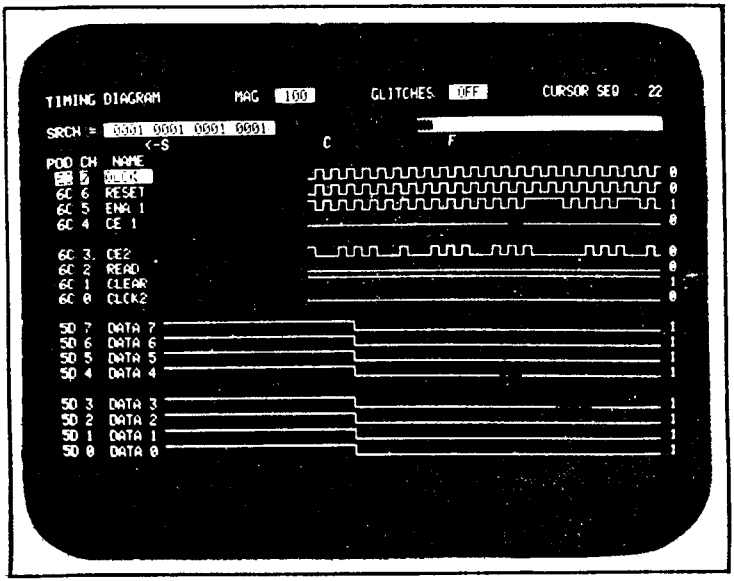


Рис. 4.6. Пример влияния выполнения программы (внизу) на событие более быстрых аппаратурных средств (вверху)

DAS 9100 в испытательную систему. Специальный язык дистанционного программирования дает возможность выполнять все операции, задаваемые с клавиатуры, и некоторые дополнительные команды для автоматизированного тестирования.

§ 4.5. Комплекс диагностирования фирмы «Hewlett—Packard»

Комплекс диагностирования фирмы «Hewlett-Packard» [19] состоит из генератора слов 8180A, логического анализатора 8182A и расширителя каналов генератора

слов 8181А. Комплекс может использоваться как в настольном, так и в стоечном варианте. Генератор слов и логический анализатор работают в диапазоне тактовой частоты 1 Гц — 50 МГц. Комплекс выполняет измерение уровней и осуществляет операции одно- или двухпорогового сравнения с сигналами ТТЛ-, К-МОП или ЭСЛ-схем. Число выходных каналов комплекса составляет 16, число входных каналов — 32. Число каналов генератора слов может быть увеличено с 16 до 64 с применением 24-канального расширителя 8181А. Емкость памяти на один канал генератора слов и логического анализатора составляет 1К. К объекту диагностирования комплекс подключается с помощью выносных зондов, у каждого канала которых три состояния. Имеются выходы видеосигналов для подключения монитора, который может использоваться при дистанционной работе, а также при генерации и отладке программ. Человек взаимодействует с комплексом через меню команд. Управление комплексом может осуществляться как непосредственно с клавиатуры, так и от микро-ЭВМ по интерфейсу IEEE-488.

Генератор слов обеспечивает высокие характеристики синхронизации временны́х параметров. В дополнение к восьми каналам данных без возвращения к нулю можно ввести восемь каналов с возвращением к нулю. Длительность и задержка в каждом канале с возвращением к нулю программируются независимо друг от друга с разрешением до 100 пс. В 24 канала каждого расширителя можно одновременно ввести задержку относительно каналов генератора без возвращения к нулю. Генератор слов имеет диапазон выходных напряжений 2—17 В. Уровни выходных напряжений могут регулироваться ступенчато с шагом 10 мВ (20 мВ на высокоомную нагрузку).

Логический анализатор может работать в синхронном и асинхронном режимах с максимальной частотой выборки 50 МГц. Логический анализатор 8182А имеет высокую разрешающую способность синхронного анализа. Например, при измерении задержки распространения сигнала момент выборки может быть задержан относительно активного фронта тактового сигнала на время, задаваемое с разрешением 100 пс. Сдвиг временно́й диаграммы по всем каналам составляет менее 2 нс. Важная характеристика логического анализатора 8182А — режим сравнения в реальном масштабе времени, в котором на максимальной частоте сравнивают собранные данные с ожидаемыми. Пользователь имеет возможность делать

сравнения во временны́х окнах, ширина которых (от 10 до 90% длительности периода) и задержка (от 0 до 90% периода) могут программироваться с шагом 100 пс. При сравнивании выходных сигналов с заданным или требуемым уровнем анализатор обеспечивает программирование порога срабатывания по входу с разрешающей способностью 10 мВ. Логический анализатор позволяет обнаруживать помехи длительностью до 5 нс в синхронном или асинхронном режиме по всем 32 каналам. Собранную информацию логический анализатор отображает в виде списка состояний или временно́й диаграммы.

Глава 5 *
ОЦЕНОЧНЫЕ И ОТЛАДОЧНЫЕ КОМПЛЕКСЫ

§ 5.1. Оценочные комплексы

Оценочные комплексы предназначены для проведения отладки микропроцессорных систем на программном уровне. Оценочные комплексы — это микро-ЭВМ в минимальном составе, на базе которой создается проектируемая микропроцессорная система с подключенными клавиатурой и дисплеем, а также с возможностью подключения аппаратуры пользователя. В комплексах используются как одноплатные микро-ЭВМ, предназначенные для встраивания в различное оборудование, так и специально спроектированные для этих целей микро-ЭВМ.

Оценочные комплексы при проектировании микропроцессорных систем являются хорошим средством обучения и оценки возможностей микропроцессоров, стендом для макетирования; дают возможность выполнять программы в реальном времени и непосредственно на реальном микропроцессоре, но практически не способны генерировать программное обеспечение; занимают ресурсы проектируемой системы (адресное пространство памяти, области ввода — вывода и прерываний); не позволяют собирать информацию о поведении и управлять поведением проектируемой системы в режиме реального времени.

Обобщенная структура оценочного комплекса представлена на рис. 5.1. В состав оценочного комплекса

* Гл. 5 и 6 написаны совместно с Н. П. Васильевым.

входят: микропроцессор *МП*; *ПЗУ* для хранения системных программ емкостью 256-4К байт; ОЗУ для хранения данных и отлаживаемых программ пользователя емкостью 128—4К байт; контроллер ввода — вывода *К* для подключения клавиатуры *Кл* и дисплея *Д*. В качестве клавиатуры используется простейшая клавишная панель, а дисплей *Д* представляет собой или набор семисегментных индикаторов, или набор индикаторов на светоизлучающих диодах.

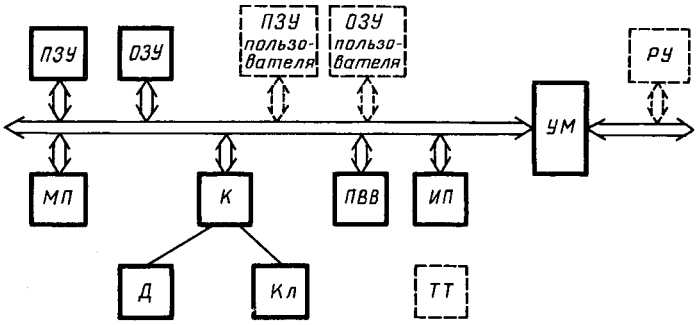


Рис. 5.1. Обобщенная структура оценочного комплекса

Примечание. В совокупности контроллер ввода — вывода, дисплей и клавиатура называются пультом. В зависимости от выполняемых функций пульт может быть пультом инженера или программиста. Последний называют *терминалом*.

Как правило, в оценочном комплексе имеется интерфейс последовательного асинхронного канала ввода — вывода *ИП* для подключения телетайпа *ТТ* или кассетного магнитофона и параллельный порт ввода — вывода *ПВВ*.

Как и в обычных одноплатных микро-ЭВМ, в оценочных комплексах часто предусматривается возможность установки дополнительных интегральных схем памяти: ПЗУ и ОЗУ пользователя в свободные гнезда на плате. Магистраль микро-ЭВМ через умощнитель *УМ* выводится на разъем, и к ней могут быть подключены разрабатываемые устройства *РУ*, дополнительная память, контроллеры ввода — вывода и внешних запоминающих устройств.

Программное обеспечение оценочных комплексов ограничивается монитором (в ПЗУ), который, несмотря на

небольшой объем, представляет достаточно гибкие средства для отладки программ (пошаговый режим, задание контрольных точек, загрузку и отображение содержимого регистров и памяти).

Возможности оценочного комплекса рассмотрим на примере комплекса, выполненного на базе одноплатной микро-ЭВМ «Электроника Н МС 11100.1» и комплекса «Microsystem Designer Series 1000» фирмы «Millennium».

§ 5.2. Оценочный комплекс на базе микро-ЭВМ «Электроника Н МС 11100.1»

Структура минимального оценочного комплекса на базе одноплатной микро-ЭВМ «Электроника Н МС 11100.1» приведена на рис. 5.2. Блок питания *БП* обеспечивает подачу напряжения +5 и +12 В. С помощью пульта *П* осуществляются инициирование процессора одноплатной микро-ЭВМ при включении питания, управление исполнением программ в пошаговом режиме и в режиме реального времени, а также прерывание отлаживаемой программы и переход к отладочному монитору, находящемуся в системном ПЗУ.

С клавиатуры пульта *П* разработчик может выполнить следующие команды.

По команде / — *Открыть ячейку* — отображается содержимое ячейки памяти, регистра общего назначения или слова состояния процессора. Команда обычно следует за адресом ячейки. Если команда / используется без задания адреса, то она отображает содержимое последней ячейки, которая открывалась ранее.

Команда ВК — *Закрыть ячейку* — подается после ввода нового значения ячейки, если требуется изменить содержимое открытой ячейки.

Команда ПС — *Закрыть ячейку и открыть следую-*

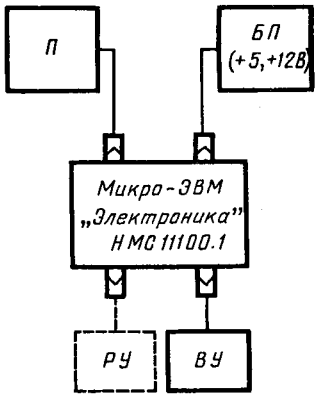


Рис. 5.2. Структура оценочного комплекса на базе одноплатной микро-ЭВМ «Электроника Н МС 11100.1»

щую — используется для отображения и (или) изменения содержимого массива последовательно расположенных ячеек. Для изменения содержимого открытой ячейки (или регистра общего назначения) команда подается после ввода нового состояния ячейки.

Команда ↑ — *Открыть предыдущую ячейку* — закрывает открытую ранее ячейку и открывает ячейку с адресом, уменьшенным на два, для регистра общего назначения — уменьшенным на единицу. Если требуется изменить содержимое закрываемой ячейки, новое ее содержимое должно предшествовать команде.

Команда @ — *Открыть ячейку с абсолютным адресом* — необходима для обращения к ячейке, адресом которой является содержимое ранее открытой ячейки или регистра общего назначения.

Команда — — *Открыть ячейку с относительным адресом* — необходима для обращения к ячейке, с адресом, определяемым как сумма трех слагаемых: содержимого уже открытой ячейки, ее адреса и числа 2.

Команда R — *Регистр* — используется для адресации регистров общего назначения путем последующего указания номера регистра (0, 1, ..., 7).

Команда RS — *Слово состояния процессора* — адресует регистр состояния процессора.

Команда G — *Пуск* — следует после указания адреса пуска программы и служит для запуска микро-ЭВМ на выполнение программы. Перед пуском программы по команде G вырабатывается сигнал сброса устройств, подключенных к системной магистрали процессора. Если клавиша режима работы находится в положении ПУЛЬТ, то после выработки сигнала сброса устройств процессор возвращается в режим связи с пультовым терминалом и отображает только что установленное состояние счетчика команд процессора.

По команде P — *Продолжение* — продолжается выполнение программы с адреса, определяемого текущим содержимым счетчика команд процессора. Если клавиша режима работы находится в положении ПУЛЬТ, то выполняется одна команда программы и процессор возвращается в режим связи с пультовым терминалом, при этом отображается содержимое счетчика команд процессора. Эта команда используется для пошагового исполнения программы.

Команда M — *Отладка* — применяется для установления причины передачи управления отладочному монитору.

При вводе команды М отображается восьмеричное число, младший разряд которого идентифицирует причину перехода: 0 — выполнение программной команды HALT (*Останов*) или перевод клавиши режима работы в положение ПУЛЬТ; 1 — нарушение интерфейса системной магистрали процессора при вводе адреса вектора прерывания; 3 — двойное нарушение интерфейса системной магистрали процессора (при нарушении интерфейса системной магистрали процессора в регистре указателя верхушки стека находится несуществующий адрес ячейки памяти).

Команда ЗБ — *Забой* — используется для отмены последнего введенного с клавиатуры терминала знака.

Пример 5.1. Рассмотрим процедуры ввода в память микро-ЭВМ и отладки программы, осуществляющей формирование массива С, элементами которого являются максимальные числа из соответствующих элементов входных массивов А и В, имеющих одинаковую размерность $10_{(8)}$ слов. Массив А расположен в ОЗУ с адреса $2000_{(8)}$, массив В — в ОЗУ с адреса $3000_{(8)}$. Результирующий массив С должен быть расположен в ОЗУ начиная с адреса 2000_8.

Пусть X — число, значение которого несущественно для данного примера; выделенные символы выводит на дисплей микро-ЭВМ, невыделенные символы вводит с клавиатуры пульта пользователь.

Ввод начальных значений массивов:

2000/X 0 ПС		3000/X 7 ПС
2002/X 1 ПС		3002/X 6 ПС
2004/X 2 ПС		3004/X 5 ПС
2006/X 3 ПС	A	3006/X 4 ПС
2010/X 4 ПС		3010/X 3 ПС
2012/X 5 ПС		3012/X 2 ПС
2014/X 6 ПС		3014/X 1 ПС
2016/X 7 ВК		3016/X 0 ВК

Ввод программы формирования массива (программу разместим в ОЗУ начиная с ячейки 1000_8):

```
1000/X 12701  ПС
1002/X 2000   ПС  Занести в R1 число 2000₈
1004/X 12702  ПС  Занести в R2 число 3000₈
1006/X 3000   ПС
1010/X 12703  ПС  Занести в R3 число 10₈
1012/X 10     ПС
1014/X 12200  ПС  Занести в R0 содержимое ячейки ОЗУ, адрес
                  которой в R2. Увеличить содержимое R2 на 2
1016/X 22100  ПС  Если R0 ⩾ содержимого ячейки
1020/X 100002 ПС  ОЗУ с адресом, находящимся в R1, то перейти
                  к команде с адресом 1024₈, в противном слу-
                  чае выполнить следующую команду. Увеличить
                  содержимое R1 на 2
1022/X 10041  ПС  Уменьшить содержимое R1 на 2. Занести
                  в ячейку ОЗУ с адресом, находящимся в R1,
                  содержимое R0
```

1024/X 5721	ПС	Увеличить содержимое R1 на 2
1026/X 77306	ПС	Уменьшить содержимое R3 на 1. Если содержимое R3 > 0, то перейти к команде с адресом 1014_8, в противном случае выполнить следующую команду
1030/X 0	ВК	Останов

Отладка программы. Так как массивы имеют малые размеры, можно выполнить всю программу в режиме покомандного исполнения, для чего клавишу режима работы необходимо установить в положение ПУЛЬТ.

1000 G	1000	Начальный пуск программы
@	P 1004	Выполнение первой команды
@	R1/2000P 1010	Отображение содержимого R1
@	R2/30000P 1014	
@	R3/10 ВК P 1016	
@	R0/7 ВК P 1020	
@	RS /11 ВК P 1022	—N=1. В следующей команде не должно быть перехода
@	P 1024	
@	2000/7 ВК P 1026	Первый элемент массива C изменяется
@	R1/2002 ПС R2/3002 ВК P 1014	
@	R3/7	

Ограничимся исполнением в покомандном режиме одного цикла программы. Дальнейшее выполнение программы проведем в реальном времени и оценим результаты работы программы. Для этого установим клавишу режима работы в положении ПРОГРАММА, введем команду P, на дисплее отобразится адрес 1032.

Проверка причины перехода в отладочный монитор:
M 10 Была выполнена команда *Останов*
Проверка значения регистров после выполнения программы:
R1 /1020 ПС
R2 /2020 ПС
R3 /0 ВК

Проверка содержимого массива C после выполнения программы:

2000/7 ПС	2010/4 ПС
2002/6 ПС	2012/5 ПС
2004/5 ПС	2014/6 ПС
2006/4 ПС	2016/7

Содержимое регистров R1—R3 и массива C свидетельствует о правильном выполнении программы.

§ 5.3. Оценочный комплекс «Microsystem Designer Series 1000»

«Microsystem Designer Series 1000» является наиболее развитым оценочным комплексом [20]. В отличие от других он позволяет оценивать возможности нескольких распространенных микропроцессоров: 8085A, 8088, 8086, Z80A, Z8000, 6800, 8049, 6801, Z8. Конструктивно и функционально комплекс разбит на две части: универсальную, которая является постоянной и независимой от испытуемого микропроцессора, и специализированную (модуль персональности), которая является сменной и зависимой от конкретного применения.

Внешний вид оценочного комплекса «Microsystem Designer Series 1000» представлен на рис. 5.3. Это настольный прибор со встроенным источником питания. Пользователь осуществляет диалог с прибором с помощью шестнадцатиричной клавиатуры *11*, 16-клавишной функциональной клавиатуры *13*, клавиши инициации прерываний *14*, светодиодных индикаторов состояния выходного порта *1*, а также 16-значного алфавитно-цифрового дисплея *10*.

Испытываемый микропроцессор устанавливается в модуле персональности *15*. Сигналы микропроцессора выводятся через плоский кабель *3*, соответствующий определенному модулю персональности,— разъем на конце кабеля совпадает с цоколевкой выводов процессора.

Клавиши функциональной клавиатуры *13* позволяют производить автоматическое чтение и запись данных ввода — вывода, выполнение программы в пошаговом режиме, установку аппаратурных контрольных точек, синхронизацию работы программ пользователя, а также пересылки, загрузку и отображение содержимого —*дампинг* — памяти.

Пользователю предоставляются возможности управления работой проектируемой системы. Обеспечивается связь с устройствами ввода — вывода оценочного комплекса, а сигналы прерывания могут вырабатываться либо вручную, либо с помощью системного таймера.

Программа вводится вручную с шестнадцатиричной клавиатуры. Затем пользователь с помощью команд функциональной клавиатуры может проверить и изменить содержимое памяти и регистров, установить контрольные точки и выполнить программу в пошаговом режиме, т. е. произвести отладку программы. Далее программу можно

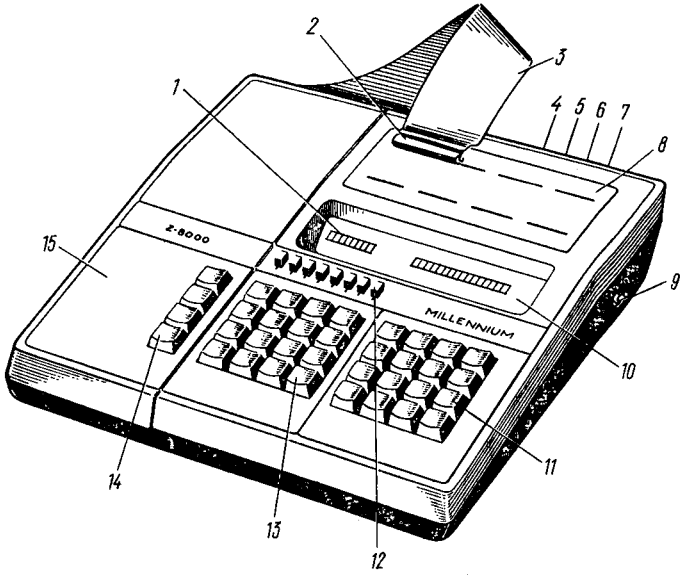


Рис. 5.3. Внешний вид оценочного комплекса «Microsystem Designer Series 1000»:

1 — светодиодные индикаторы состояния выходного порта; *2* — разъем кабеля микропроцессора; *3* — кабель микропроцессора для подключения к проектируемой системе; *4* — клавиша включения/отключения сигнала частоты 60 Гц; *5* — разъем интерфейса RS232; *6* — разъем кассетного магнитофона; *7* — внешний разъем магистрали; *8* — панель для монтажа без пайки; *9* — клавиша включения/отключения питания; *10—16*-значный алфавитно-цифровой дисплей; *11* — шестнадцатиричная клавиатура; *12* — клавиши входного порта пользователя; *13* — функциональная клавиатура; *14* — клавиши инициации прерываний; *15* — модуль персональности

записать на ленту кассетного магнитофона. На случай, если пользователь имеет доступ к какой-либо вычислительной системе, предусмотрена также возможность взаимного обмена информацией с этой системой с помощью последовательного интерфейса RS 232.

Как видно из структуры оценочного комплекса, представленной на рис. 5.4, основное устройство (ОУ) содержит универсальную системную магистраль *УСМ*, к которой подключается периферийный контроллер *К* (для управления клавиатурой *Кл* и дисплеем *Д*), ОЗУ, контроллер последовательного ввода — вывода *КВВ* (для сопряжения с кассетным магнитофоном *ИКМ* и интерфейсом *И* RS 232),

параллельный порт ввода — вывода *ПВВ*, таймер *ТМ* и схемную плату макетирования *ПМ*. Магистраль *УСМ* сопрягается также с кабелем расширения, который позволяет увеличить допустимую область системной памяти ввода — вывода. В пределах этой универсальной части комплекса команды и форматы остаются постоянными, не зависящими от типа испытываемого микропроцессора.

Зависящий от типа микропроцессора сменный модуль персональности включает в себя испытываемый микропроцессор *МП*, а также генератор синхросигналов *ГСС* этого микропроцессора *МП* и электрически программируемое постоянное запоминающее устройство (ЭППЗУ), в котором хранится системный монитор *СМ*. Модуль персональности содержит также логическую схему преобразования сигналов магистрали *ЛСП* и логическую схему переходов для аппаратурной передачи управления на программы монитора *ЛАП*. Этот отдельный сменный модуль персональности подключается к основному устройству с помощью 86-контактного разъема умощнителя магистрали *УМ*.

Периферийный контроллер *КВВ* выполняет функции управления дисплеем *Д* оценочного комплекса и сканирования клавиатуры *К*. Микропроцессор, не занятый этими функциями, может полностью программироваться поль-

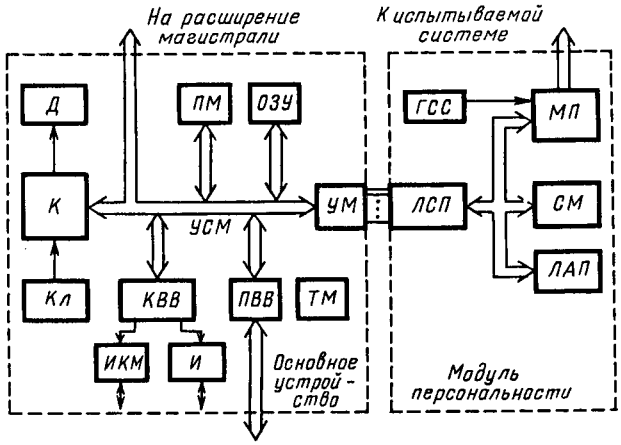


Рис. 5.4. Структура оценочного комплекса «Microsystem Designer Series 1000»

зователем и использовать параллельные порты ввода — вывода *ПВВ* оценочного комплекса.

Оперативное запоминающее устройство *ОЗУ* общим объемом 1К байт является общим для всех модулей персональности. В основном устройстве предусмотрены внутренние гнезда для расширения этой памяти до 4К байт. Это *ОЗУ* предназначено для хранения программ пользователя и данных.

Системный монитор, хранящийся в ЭППЗУ, занимает в сменном модуле персональности место, зависящее от типа микропроцессора, и содержит программы, управляющие работой комплекса.

Логическая схема аппаратурных переходов обеспечивает автоматический вход в монитор после включения питания или после нажатия клавиши функциональной клавиатуры MON вызова монитора; обеспечивает также возврат в монитор при обнаружении контрольной точки или при пошаговом режиме исполнения программы пользователя.

Клавиша контрольной точки дает возможность пользователю устанавливать и проверять аппаратурную контрольную точку (точку прерываний). Проверка выполнения условия для контрольной точки осуществляется с помощью 16 бит регистра, который загружается главным процессором и синхронизируется его сигналами чтения и записи памяти.

Пошаговый режим, задаваемый клавишей пошагового режима, реализуется с помощью аппаратурного перехода на монитор после выполнения одной команды. Постоянное нажатие клавиши одношагового режима приводит к поочередному выполнению команд программы пользователя с временной задержкой от 20 мс до 6 с. Величина задержки может быть установлена пользователем.

Ввиду различных требований синхронизации микропроцессоров, на которые рассчитан оценочный комплекс, каждый модуль персональности содержит собственный генератор синхросигналов ГСС. Это означает, что каждый микропроцессор может работать с максимальной собственной скоростью.

После подключения микропроцессорного кабеля, выходящего из модуля персональности, к макету разрабатываемой системы пользователю предоставляется возможность вызывать команды оценочного комплекса. Клавиша ввода — вывода «1/0» функциональной клавиатуры позволяет вручную проверять средства ввода — вывода, чтобы удо-

стовериться в работоспособности аппаратуры проектируемой системы. После такой проверки можно задать контрольные точки для программы (если даже она уже располагается в постоянной памяти) и проверить работу программы в пошаговом режиме.

Однако пользователь комплекса обязан помнить о двух очень важных ограничениях: во-первых, главный процессор в устройстве должен нагружаться на кабель, а также на проектируемую систему (такая нагрузка может сказаться на уменьшении быстродействия); во-вторых, так как модуль персональности использует некоторую часть области ввода — вывода и ячеек памяти для монитора, то проектируемая система не может занимать эту область адресного пространства. Поэтому проектируемая система должна быть небольшой, чтобы на ее работе не отражались эти ограничения по памяти и средствам ввода — вывода.

§ 5.4. Отладочные комплексы

Отладочные комплексы предназначены для отладки микропроцессорных систем на программном уровне описания. Они отличаются от оценочных комплексов развитым программным обеспечением, увеличенной емкостью памяти и усложненным интерфейсом, позволяющим использовать более широкий диапазон устройств ввода — вывода. Изготовляются отладочные комплексы в виде конструктивно законченного блока, объединяющего 2 — 15 плат и имеющего пульт. Так же как и в оценочных комплексах, основой системы отладочных комплексов является микро-ЭВМ на базе комплекта БИС, который будет применяться в проектируемой системе [21], и системная магистраль выводится на внешний разъем для подключения контроллеров, разрабатываемых пользователями.

Использование отладочных комплексов при проектировании микропроцессорных систем дает следующие преимущества: возможность программирования на языке ассемблера или языках высокого уровня; широкий набор внешних устройств (включая НГМД); развитую дисковую операционную систему. Недостатки: предназначаются для одного типа микропроцессора; накладывают ограничения на архитектуру проектируемой системы; занимают системные ресурсы; не позволяют собирать информацию о поведении системы и управлять поведением системы в режиме реального времени.

Увеличение емкости памяти отладочных комплексов по сравнению с емкостью памяти оценочных комплексов обусловлено большим объемом системного программного обеспечения и увеличением объема отлаживаемых программ.

Системное программное обеспечение включает в себя системный монитор и систему программирования (ассемблер или макроассемблер, редактор текста, редактор связей, загрузчик, отладчик).

Наличие НГМД существенно упрощает работу с отладочным комплексом и позволяет дополнительно использовать трансляторы языков высокого уровня ФОРТРАН, ПАСКАЛЬ, КОБОЛ, PL/1, АДА и т. д., а также дисковую операционную систему, обеспечивающую работу с файлами.

Для увеличения производительности труда разработчика к отладочному комплексу подключается видеотерминал, обеспечивающий удобную отладку программ в интерактивном режиме.

Отладочные комплексы часто имеют интерфейсы для подключения внешних устройств, однако сами устройства не включаются в состав поставляемого комплекта. Пользователь отладочного комплекса закупает и подключает к нему те устройства ввода — вывода, которые считает нужными.

При разработке программного обеспечения с использованием отладочных комплексов разработчику предлагается система программирования, поддерживаемая операционной системой комплекса. Дадим краткую характеристику основных программных компонентов системы программирования, входящих в отладочные комплексы, в том порядке, в каком они нужны при разработке программы.

Компиляторы и ассемблеры. Они преобразуют исходную программу в объектную, выдают листинг программы, на котором напечатаны обе ее версии (исходная и объектная), список сообщений об ошибках и другие виды диагностической информации. Входной информацией для ассемблера служит исходная программа на символьном языке (языке ассемблера), подробно отображающая архитектуру микропроцессора (написание этой программы весьма трудоемкое дело). Входной информацией для компилятора является исходная программа на языке высокого уровня, не связанная с архитектурой микропроцессора. Компилятор использует грамматику и словарь команд, легко усваиваемые программистами и разработанные для про-

граммирования конкретного класса задач. Это позволяет разработчику забыть о внутренней структуре микропроцессора и сконцентрировать все силы на логике своей программы. К сожалению, языки высокого уровня ведут к потере скорости выполнения программ и увеличению объема памяти для их хранения и выполнения.

При оценке ассемблеров необходимо учитывать:

а) возможности ассемблеров, далеко превосходящие простую способность транслировать код вызова длинной (заданной пользователем) последовательности команд с помощью одной команды (макрорасширения); реализацию сложных математических и логических выражений; генерацию таблиц данных, констант и текстовых цепочек; генерацию перемещаемого объектного кода; среду условной трансляции — средства для указания ассемблеру о трансляции или нетрансляции (в объектный код) конкретных блоков предложений исходной программы;

б) влияние свойств ассемблеров на основную структуру программы;

в) способность ассемблеров ввести в практику программирования такую форму кода, которая будет пригодна для последующих проектов (например, тщательно разработанный набор макрокоманд, предназначенный для конкретных задач программирования, можно рассматривать как специальный язык высокого уровня);

г) быстродействие ассемблера (особенно важно при разработке больших программ, так как во время отладки программы обычно требуется целый ряд повторных трансляций для исправления ошибок, и если время каждой трансляции будет большим, то масса времени тратится на ожидание трансляции. Типовая скорость трансляции с гибкими дисками составляет 200—2000 байт объектного кода в минуту).

Оценить компиляторы с языков высокого уровня еще труднее, так как нужно учитывать: какой используется язык; эффективность компилятора в терминах размеров объектной программы и скорости ее выполнения; обеспечивает ли компилятор возможность ручной оптимизации критических областей программы; а также связь с программами на языке ассемблера.

Пример 5.2. Листинг программы формирования массива (см. пример 5.1), написанной на языке ассемблера, имеет следующий вид:

```
MAXAR   MACRO V3.02B 11-JUL-83 PAGE 1

     1                          ;
     2                          ;       ПРОГРАММА ФОРМИРОВАНИЯ МАССИВА
     3                          ;
     4                          .TITLE MAXAR
     5                          ;
     6           000010                 L = 10
     7  000000                  .ASECT
     8           001000         .=1000
     9                          ;
    10  001000   012701  START:: MOV     #A, R1
                 002000
    11  001004   012702          MOV     #B, R2
                 003000
    12  001010   012703          MOV     #L, R3
                 000010
    13  001014   012200  RPT:    MOV     (R2)+, R0
    14  001016   022100          CMP     (R1)+, R0
    15  001020   100002          BPL     GE
    16  001022   010041          MOV     R0, -(R1)
    17  001024   005721          TST     (R1)+
    18  001026   077306  GE:     SOB     R3, RPT
    19  001030   000000          HALT
    20                          ;
    21           002000         .=2000
    22  002000   000000  A:      .WORD 0,1,2,3,4,5,6,7
        002002   000001
        002004   000002
        002006   000003
        002010   000004
        002012   000005
        002014   000006
        002016   000007
    23           003000         .=3000
    24  003000   000007  B:      .WORD 7,6,5,4,3,2,1,0
        003002   000006
        003004   000005
        003006   000004
        003010   000003
        003012   000002
        003014   000001
        003016   000000
    25                          ;
    26           001000         .END    START
```

Листинг программы, написанной на языке ФОРТРАН-IV, имеет следующий вид:

```
FORTRAN IV      V02.1-10    MON 11-JUL-83 00:00:00      PAGE 001

      C     ПРОГРАММА ФОРМИРОВАНИЯ МАССИВА
      C
0001        PROGRAM MAXAR
      C
0002        INTEGER A(8),B(8),L,I
```

```
      C
0003        DATA A/0,1,2,3,4,5,6,7/
0004        DATA B/7,6,5,4,3,2,1,0/
0005        DATA L/8/
      C
0006           DO 1 I=1,L
0007           IF(A(I).LT.B(I)) A(I)=B(I)
0009  1        CONTINUE
0010        STOP 'END MAXAR'
0011        END
```

Редактор текста. Вводится исходный текст программы в отладочный комплекс с помощью программы — *редактора текста*,— которая позволяет пользователю ввести исходную программу, исправить ошибки и записать ее в память, например на НГМД, для последующего вызова. Бóльшая часть ввода осуществляется один раз, а все исправления при отладке производятся путем вызова из памяти исходной программы ее коррекции и записи обратно в память. Редактор текста — программа для обработки текста вообще; ему безразлично, чтó редактировать: исходную программу или журнальную статью.

Редакторы текста ранних разработок — *последовательные редакторы текста* — проектировались для использования в вычислительных системах, имеющих телетайп в качестве терминала оператора. Многие отладочные комплексы все еще применяют такие редакторы текста. Малая скорость работы терминала при последовательном редактировании ограничивает возможности по отображению редактируемого текста. Кроме того, когда пользователь редактирует строку, последовательный редактор отпечатает эту измененную строку только после того, как пользователь введет ряд команд редактору текста, и поэтому пользователь должен уметь мысленно прослеживать, каков будет результат работы за пультом. Последовательный редактор недостаточно удобен для пользователя.

Использование алфавитно-цифровых дисплеев в качестве терминалов позволяет совершенно по-новому проектировать редакторы текстов. *Экранный редактор текста* может одновременно отображать от 16 до 24 смежных строк текста (число строк определяется типом дисплея), поэтому любую конкретную строку легко найти по контексту, а не по неудобным номерам строк, часто используемым в последовательных редакторах. Когда нужно изменить какую-либо строку текста, светящийся курсор на экране дисплея показывает место изменений, а редактор моментально выдает результаты коррекции и любые ошибки пользователя будут сразу обнаружены.

Правильно спроектированный экранный редактор текста способен сократить время редактирования в три раза по сравнению с временем редактирования с использованием последовательного редактора текста. Если учесть, что 30—50 % времени разработчика тратится на редактирование [10], то выгода использования экранных редакторов текста становится очевидной.

Редактор связи. Если ассемблер или компилятор генерирует перемещаемый объектный код, то требуется еще одна программа — *редактор связей*. На основе перемещаемого кода программу можно составлять из модулей, причем трансляцию каждого модуля выполняют раздельно, но со ссылкой на другие модули через символьные (глобальные) метки. Редактор связей модифицирует объектные программы для разрешения перекрестных ссылок.

Загрузчик. Загрузка программы, скомпонованной редактором связей, осуществляется *программой-загрузчиком* — программой, переносящей программу с некоторой внешней среды (например, с НМЛ, НГМД или линии связи) в ОЗУ микро-ЭВМ.

Отладчик. Отладка программы на микро-ЭВМ производится *программами-отладчиками*. Последние принимают от пользователя команды на выполнение: а) отображения содержимого памяти или регистров процессора; б) изменения содержимого памяти или регистров процессора; в) программы с указанного адреса; г) приостанова программы по достижении команды, находящейся в определенной ячейке памяти или при выполнении заданного условия; д) пошагового (покомандного) исполнения программы; е) трассировки программы с разнообразными условиями отображения диагностической информации; ж) отладки по заранее составленной программе, и т. д.

Некоторые программы-отладчики позволяют вести отладку программ в терминах исходного языка написания программы. Операции, выполняемые программой-отладчиком, занимают существенно больше времени, чем работа отлаживаемой программы, вследствие чего отладка в реальном масштабе времени с помощью отладочных комплексов невозможна.

Отладочный комплекс практически имеется для каждого микропроцессорного набора. Например, для микропроцессора К1801 ВМ1 — отладочный комплекс ДВК «Электроника НЦ-8020», для микропроцессора К580 — отладочный комплекс «Ока», для микропроцессорного набора К587 — отладочный комплекс «Интол», и т. д.

Глава 6

КОМПЛЕКСЫ РАЗВИТИЯ

§ 6.1. Обобщенная структура комплексов развития

Комплексы развития предназначены для отладки микропроцессорных систем на программном уровне описания и позволяют на программном уровне управлять поведением системы, собирать информацию о поведении системы, моделировать (эмулировать) недостающие устройства проектируемой системы (микропроцессор, ЗУ, контроллеры, внешние устройства и т. д.) в режиме реального времени или в режиме, близком к этому.

Комплексы развития применяют для разработки программ; проведения комплексной отладки; в ряде случаев для диагностирования неисправностей при изготовлении в процессе производства спроектированной системы.

Комплексы развития характеризуются типом и числом эмулируемых микропроцессоров, числом одновременно работающих пользователей, емкостью ОЗУ пользователя, емкостью внешних запоминающих устройств (ВЗУ), типом печатающего устройства, составом системного программного обеспечения, отладочными возможностями.

Обобщенную структуру комплекса развития можно представить состоящей из микро-ЭВМ с периферией (видеотерминал, внешние запоминающие устройства, принтер) и внутрисхемного эмулятора (ВСЭ). Взаимодействие между микро-ЭВМ и ВСЭ осуществляется по магистрали комплекса развития. Внутрисхемный эмулятор выполняет следующие функции: эмулирует поведение и электрофизические характеристики микропроцессора проектируемой системы и ЗУ (ОЗУ, ПЗУ, ППЗУ); эмулирует периферийные устройства, контроллеры и другие устройства проектируемой системы; собирает информацию о поведении системы на программном уровне представления; управляет поведением проектируемой системы.

Внутрисхемный эмулятор может прервать (остановить) функционирование проектируемой системы при появлении заданного события; пускать проектируемую систему с заданной команды; выполнять программу в пошаговом или автоматическом режиме; изменять состояние

памяти, внутренних регистров микропроцессора, портов ввода — вывода проектируемой системы.

Внутрисхемный эмулятор подключается к проектируемой системе на то место, где должен находиться микропроцессор проектируемой системы (микропроцессор устанавливается на свое место в системе, после того как прекращается отладка системы с помощью комплекса развития). Эмуляция ЗУ и других устройств проектируемой системы позволяет использовать ресурсы комплекса развития для поэтапной отладки по мере разработки и изготовления аппаратуры проектируемой системы. Например, с помощью ОЗУ комплекса можно физически моделировать ПЗУ проектируемой системы и вести отладку, помещая программы пользователя в ОЗУ.

В части сбора информации о поведении системы ВСЭ обладает возможностями логических анализаторов с синхронной записью данных. Кроме того, ВСЭ позволяет собирать статистические данные о времени выполнения тех или иных участков программы.

Микро-ЭВМ подготавливает информацию для ВСЭ, управляет ВСЭ и остальными ресурсами комплекса, обрабатывает собранную информацию и представляет ее пользователю на языке, используемом при разработке программ.

Стоимость комплекса развития определяется стоимостью периферийных устройств. Видеотерминал характеризуется числом строк (16, 24 у современных) и числом символов в строке (64 — 80). В качестве ВЗУ используется НГМД с емкостью памяти 0,25 — 2 М байт, кассетные накопители на магнитной ленте с емкостью до 0,5 Мбайт; в многопользовательских комплексах развития используются накопители на жестком магнитном диске с емкостью 5 — 120 Мбайт.

Кроме стандартных внешних устройств комплексы развития содержат программируемые устройства — программаторы для «зашивки» отлаженных программ в ППЗУ или РППЗУ.

Основные различия архитектурных решений комплексов развития определяются числом магистралей и микропроцессоров, входящих в их состав. По этому признаку комплексы развития делятся [22] на однопроцессорные и многопроцессорные одномагистральные и многопроцессорные многомагистральные.

Однопроцессорные одномагистральные комплексы развития наиболее простые и дешевые. Достоинство этих ком-

плексов состоит в том, что они используют одну неразделенную память ЗУ. Так как комплексы содержат один микропроцессор ЦП, то этот микропроцессор должен быть таким же либо близким к микропроцессору в разрабатываемой системе. Серьезным недостатком таких комплексов является то, что микропроцессор должен выполнять как функции эмулятора, так и системные функции комплекса развития: трансляцию программ, редактирование, загрузку и т. д. Поэтому значительная часть адресного поля, по крайней мере один адрес порта ввода — вывода и код прерывания, выделена под эти системные нужды. Это существенно уменьшает возможность эмуляции.

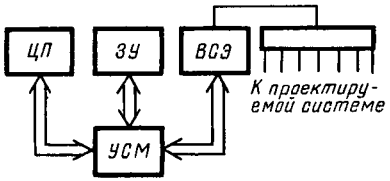


Рис. 6.1. Структура двухпроцессорного одномагистрального комплекса развития

В многопроцессорных одномагистральных структурах комплексов развития используют несколько микропроцессоров, работающих на общую магистраль (рис. 6.1). Обычно в состав комплекса развития входит два микропроцессора: один — системный *ЦП* — для выполнения необходимых функций комплекса развития, другой в составе внутрисхемного эмулятора *ВСЭ* — для эмуляции. Оба микропроцессора имеют общую память *ЗУ*. Недостатком многопроцессорной одномагистральной структуры комплекса развития является то, что в данный момент времени может работать только один микропроцессор. Поэтому, если в процессе эмуляции необходимо передать какую-либо инструкцию в систему, микропроцессор эмулятора необходимо останавливать, что бывает недопустимо для некоторых процедур управления. Другой недостаток систем подобного типа — сложная операционная система.

Многомагистральные многопроцессорные структуры лишены перечисленных недостатков. Каждый внутрисхемный эмулятор (рис. 6.2) имеет микропроцессор в составе эмулятора микропроцессора *ЭМП*, собственную память *ЭП* и магистраль *МЭ*, что позволяет ему вести эмуляцию одновременно с другими эмуляторами и независимо от них.

Системный процессор *ЦП* имеет собственную магистраль *МУС*, память *ЗУ* и контроллер ввода — вывода *КВВ*, может обращаться к микропроцессору любого эму-

Таблица 6.1

Фирма	Модель	Эмулируемый процессор	Число пользователей	ОЗУ пользователя, К байт	Емкость ВЗУ, М байт
1	2	3	4	5	6
«Advanced Micro Computers»	AMSYS, 8/8	Z8000	1	64	0,5—1,0
«Diversified Technology Inc.»	EL.COMP, Model DS-2	1802	1	9—32	0,5—2,0
«Fairshild Camera and Instrument Corp.»	FORMULATOR MARK II &, MARK III	F8 F3870 F3872 F3876 F3878	1	16—64	0,5—1,0
«Gen Rad/Future-Data»	Advanced Development System	8080 8085 8086 Z80 6800 6802	8	48—256	10—20
«Hewlett-Packard»	HP64000	8080 8085 6800 Z80	6	8—64	120
«Intel Corp.»	MDS-210 MDS-220 MDS-230 MDS-240	8021 8035 8039 8041 8048 8049 8080 8085 8086	1	32—64	0,25—7,5
«Tektronix Inc.»	8002A	8080 6800 Z80 9900	1	64	0,6—1,2
«Texas Instruments Inc.»	990/10	9900 9980 9981 9940 9985	16	16—10000	0,25—9,0
«Zilog Inc.»	ZDS-1/40	Z80 Z80A	1	60	0,6

Печатающее устройство	Дополнительные внешние устройства	Программное обеспечение	Отладочные особенности
7	8	9	10
132 столбца, 120 знак/с	Программатор ППЗУ	Дисковая операционная система, ассемблер, ФОРТРАН, БЭЙСИК, КОБОЛ	Точки останова, трассировка, дизассемблер
132 столбца, 150 знак/с	То же	БЭЙСИК, интерактивный отладчик	Точки останова, трассировка
128 столбцов, 120 строк/мин	»	Операционная система, ассемблер, система управления файлами	То же
132 столбца, 120 знак/с, или 300 строк/м	Программатор ППЗУ, логический анализатор	Операционная система, ассемблер, БЭЙСИК, ПАСКАЛЬ	Точки останова, эмуляция в реальном масштабе времени, дизассемблер, пошаговый режим
136 столбцов, 70 или 400 строк/м	Программатор ППЗУ, кассетная магнитная лента	Дисковая операционная система, ассемблер	Точки останова, эмуляция в реальном масштабе времени, трассировка
—	Программатор ППЗУ	Дисковая операционная система, ассемблер, ФОРТРАН, БЭЙСИК, КОБОЛ	Точки останова, эмуляция в реальном масштабе времени, трассировка, символьная отладка, дизассемблер
—	То же	Дисковая операционная система, ассемблер, БЭЙСИК, ФОРТРАН,	Точки останова, трассировка, сбор данных (анализатор данных)
132 столбца, 150 знак/с	»	Ассемблер, ПАСКАЛЬ, БЭЙСИК, ФОРТРАН, AMPL	Трассировка, точки останова, эмуляция в реальном масштабе времени
132 столбца, 120 знак/с	»	—	Трассировка, точки останова, символьная отладка, дизассемблер

Продолжение таблицы 6.1

Фирма	Модель	Эмулируемый процессор	Число пользователей	ОЗУ пользователя, К байт	Емкость ВЗУ, М байт
1	2	3	4	5	6
«Mosctek Corp.»	Mosctek Matril	Z80 3870 3872 3876 3874	1	56	0,25—0,5
«Motorola Microsystems»	Exoterm 220 Exorisel II	6801 6805 6809 14100 3870	1	64	0,5—2,0

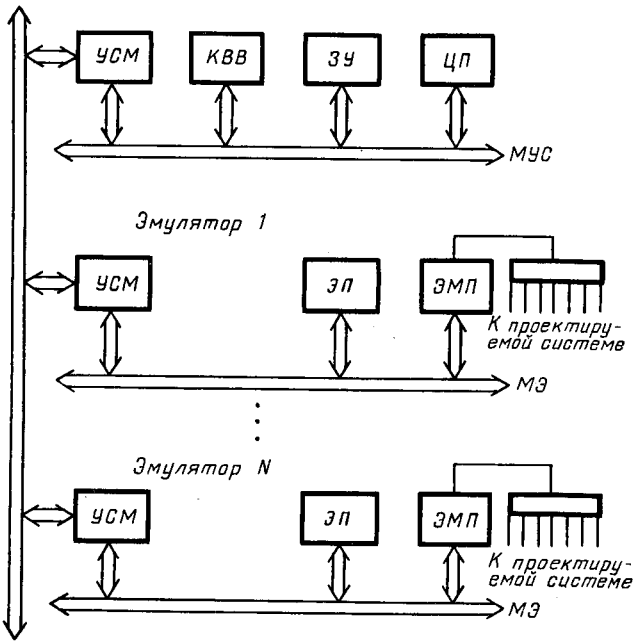


Рис. 6.2. Структура многопроцессорного многомагистрального комплекса развития

Печатающее устройство	Дополнительные внешние устройства	Программное обеспечение	Отладочны особенности
7	8	9	10
132 столбца, 120 знак/с	»	Дисковая операционная система, ассемблер, БЭЙСИК, ФОРТРАН, ПАСКАЛЬ	Трассировка, точки останова, пошаговый режим
132 столбца, 225 строк/м	»	Дисковая операционная система, БЭЙСИК, ФОРТРАН, MPL, КОБОЛ, ПАСКАЛЬ	Трассировка, точки останова

лятора для ввода или вывода данных, чтения или изменения содержимого регистров и портов ввода — вывода через блок управления магистралью *УСМ*. Такая архитектура, при которой один микропроцессор используется для системных целей, а остальные под его управлением занимаются лишь вопросами эмуляции, называется архитектурой типа «управляющий — подчиненный» («ведущий — ведомый»).

Программное обеспечение комплекса развития обычно состоит из дисковой операционной системы, системы управления файлами, редакторов текста, кросс-ассемблеров и кросс-компиляторов, обеспечивающих разработку программ соответственно на языке ассемблера или языке высокого уровня для конкретного микропроцессора, драйвера внутрисхемного эмулятора, редактора связей, загрузчика, системного монитора. В табл. 6.1 приведен краткий перечень комплексов развития ведущих зарубежных фирм.

§ 6.2. Внутрисхемный эмулятор

Функционально ВСЭ можно представить состоящим из нескольких устройств, объединенных общими цепями управления и обмена данными: *эмулятора микропроцессора*, предназначенного для физического моделирования микропроцессора проектируемой системы; *памяти трассировки*, служащей для накопления информации о пове-

дении проектируемой системы в реальном масштабе времени; *логического компаратора*, с помощью которого задаются условия останова программы пользователя; *карты адресов* (карты памяти), обеспечивающей распределение и защиту адресного пространства проектируемой системы; *эмуляционной памяти*, предназначенной для физического моделирования отсутствующей памяти проектируемой системы; *таймера*, используемого для определения времени выполнения программы пользователя или отдельных ее участков.

Существующие эмуляторы микропроцессоров (ЭМП) строятся по одному из трех принципов: 1) на базе дискретной логики; 2) на основе микропроцессорных наборов; 3) на основе микропроцессора, для физического моделирования которого предназначен данный эмулятор.

При построении ЭМП на базе дискретной логики и микропроцессорных наборов трудно получить характеристики ЭМП, адекватные эмулируемому микропроцессору. Такой ЭМП может отличаться электрофизикой, временем выполнения отдельных команд, функциональными возможностями. Для того чтобы ЭМП совпадал по своим возможностям с эмулируемым микропроцессором, в качестве ЭМП используют эмулируемый микропроцессор. К таковым относятся ЭМП для однокристальных микропроцессоров 8080, К1801 ВМ1, 8085, Z80, 6800, 6802, 2800, TMS9900 и др., для многокристальных микропроцессоров — 3870, 300, К1802, К587 и др. Перечисленные микропроцессоры имеют выведенные наружу магистрали адресов, данных, управления, позволяют наблюдать за своим поведением; считывать и изменять внутреннее состояние.

Однако не всегда можно построить ЭМП на базе эмулируемого микропроцессора, так как не каждый микропроцессор обладает свойством контролепригодности. Для того чтобы улучшить технические характеристики микропроцессорных систем, а также в связи с увеличением степени интеграции стали выпускаться однокристальные микро-ЭВМ (8048, 8049, 8051, TMS 1000 и др.), содержащие внутреннюю магистраль, процессор, ПЗУ программ, ОЗУ данных, контроллеры ввода — вывода. Внутренняя магистраль этих микро-ЭВМ недоступна пользователю. По сигналам на внешних выводах этих БИС нельзя судить о правильности выполнения программы. Для эмуляции подобных микро-ЭВМ необходимо: использовать внешнее ОЗУ программ вместо внутреннего ПЗУ, чтобы иметь возможность изменять и корректировать отлаживаемые програм-

мы; вывести наружу магистрали для работы с внешним ОЗУ программ; иметь доступ ко всем внутренним частям микро-ЭВМ для анализа и изменения состояния. Стали выпускаться микропроцессоры, в которых предусмотрена асинхронная работа многих функциональных узлов, таких, как предварительная выборка команд и организация очереди опережающей обработки, асинхронные связные контроллеры. Хотя эти функциональные узлы могут быть различными, все они связаны с командами микропроцессора и меняют последовательность выполнения команд таким образом, что это нельзя определить наблюдением за внешними выводами БИС, несмотря на то что магистрали данных, адреса и управления микропроцессора выведены. К подобным микропроцессорам относятся 8086, К1801 ВМ2, 8088 и др.

Существует несколько способов эмуляции однокристальных микропроцессоров и микро-ЭВМ. Один из основных способов — создание контролепригодных микропроцессоров и микро-ЭВМ; необходимо на этапе проектирования закладывать свойства контролепригодности. Второй способ — подключение внешних схем, дублирующих работу определенных узлов кристалла. Третий способ заключается в создании специального диагностического варианта кристалла, названного эмуляционным, имеющего дополнительные контакты, на которые выведена внутренняя магистраль микропроцессора или микро-ЭВМ.

§ 6.3. Комплекс развития «Электроника Т3»

Комплекс развития «Электроника Т3» [23] является универсальным средством отладки на программном уровне микропроцессорных систем на базе различных типов микропроцессоров. Данный комплекс включает в себя следующие аппаратурные и программные модули: набор плат внутрисхемных эмуляторов (ПВСЭ), число которых соответствует числу типов эмулируемых 4, 8 и 16-разрядных микропроцессоров; плату управления (ПУ) однострочным дисплеем и клавиатурой; две платы модулей программаторов БИС ППЗУ и БИС ПЛМ (программируемых логических матриц); пакеты отладочных программ (ПОП) на гибких магнитных дисках, соответствующие числу типов эмулируемых микропроцессоров; базовый вычислительный комплекс (БВК).

В качестве БВК могут быть использованы серийно вы-

пускаемые вычислительные средства с системой команд микро-ЭВМ «Электроника-60», имеющих интерфейс МПИ (ОСТ 11 305. 903—80): 15 ВУМС — 028—025, ДВК «Электроника НЦ 8020/2».

Аппаратурные и программные модули комплекса «Электроника Т3» позволяют создавать следующие конфигурации отладочных средств: комплекс развития, состоящий из ДВК «Электроника НЦ 8020/2», ПВСЭ, модульного программатора и ПОП; автономное отладочное устройство, включающее ПВСЭ, ПУ и блок питания, размещенные в едином конструктиве.

Автономное отладочное устройство реализует ввод программ в машинных кодах, опрос и модификацию рабочих регистров и ЗУ, пошаговый режим отладки с остановом в контрольных точках, эмуляцию памяти проектируемой системы, подсчет времени выполнения программ и подпрограмм. Структура автономного отладочного устройства представлена на рис. 6. 3. Платы управления *ПУ* и внутрисхемого эмулятора *ПВСЭ* связаны между собой, а при необходимости с БВК через интерфейс МПИ. В состав *ПУ* входят однострочный дисплей *Д*, на который выводятся адрес команд, код команды и данные, 20-клавишная клавиатура *Кл* для ввода программ, данных и управления режимами работы, *ОЗУ* для хранения программ и данных объемом 2К байт, *ПЗУ* для хранения программы-монитора, *МП* К580ИК80 с тактовым генератором *ТГ* и адаптер МПИ. Связь отдельных функциональных узлов платы управления осуществляется через магистраль МП К580ИК80 *(МАДУ)*.

Плата ПВСЭ содержит БИС МП, аналогичную используемой в проектируемой системе, тактовый генератор *ТГ*, *ОЗУ* для хранения отлаживаемой программы, *ПЗУ* для хранения управляющих программ, *ЗУ* контрольных точек останова, компаратор *К*, схему управления циклом эмуляции *СУЦЭ*, карту памяти *КП*, таймер, регистр команд данных *РгКД*, адаптер *МПИ*, схему управления магистралями эмулятора и проектируемой микропроцессорной системы *СУ*, формирователь *Ф*, который с помощью гибкого кабеля и штыревого разъема соединяется с розеткой БИС микропроцессора проектируемой системы.

Управление внутрисхемным эмулятором, а также обмен между его памятью и микро-ЭВМ осуществляется с помощью четырех регистров команд управления и данных с фиксированными адресами, задаваемыми на ПВСЭ с помощью перемычек. Первые два регистра используются в

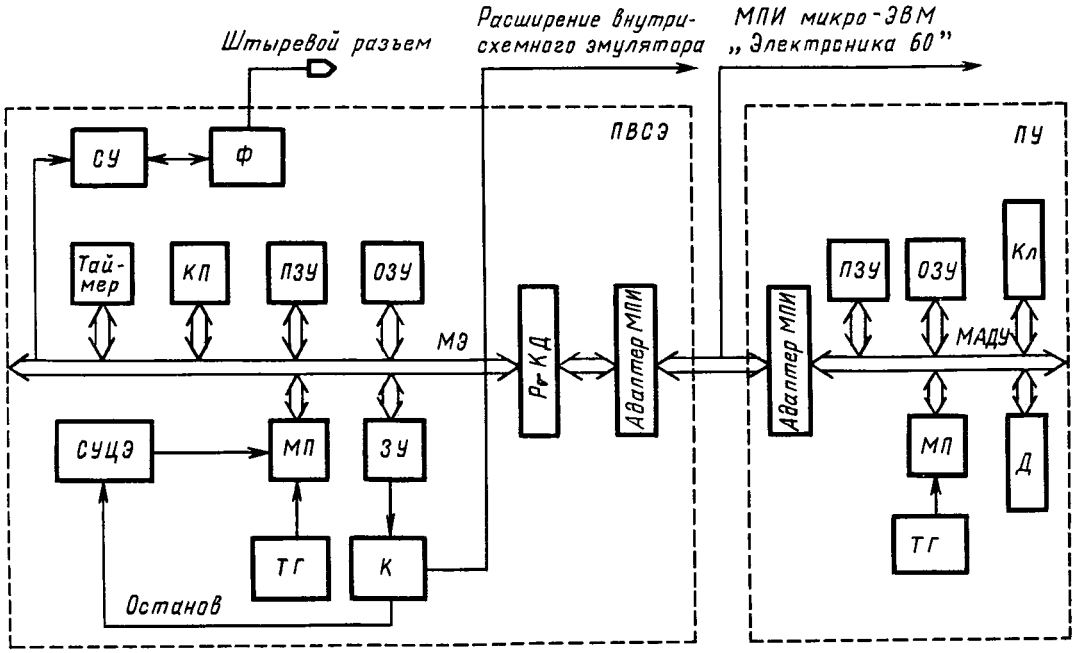


Рис. 6.3. Структура автономного отладочного устройства комплекса развития «Электроника Т3»

качестве регистров адреса и данных памяти, а вторые — в качестве регистров команд и статуса.

Запоминающее устройство *ЗУ* контрольных точек останова, компаратор *К* и схема управления циклом эмуляции *СУЦЭ* позволяют реализовать пошаговый режим отслеживания за состоянием *МЭ* с остановом в контрольных точках. С помощью таймера ведется подсчет времени выполнения программ или подпрограмм. Карта памяти *КП* служит для распределения адресного пространства проектируемой системы. Схема управления магистралями необходима для организации связи микропроцессора через формирователи *Ф* с магистралями проектируемой системы. Магистраль эмулятора выводится на внешний разъем для подключения внешних периферийных устройств (кассетного *ЗУ*, программатора БИС ППЗУ) или для расширения отладочных функций.

При подключении автономного отладочного устройства к БВК к перечисленным отладочным функциям добавляются ввод программ на языке ассемблера микропроцессора, трассировка (до 64 состояний магистралей), документирование информации, долговременное хранение и ввод программ с гибкого магнитного диска, программирование БИС ППЗУ и т. д.

§ 6.4. Комплекс развития HP 64000 фирмы «Hewlett—Packard»

Структура комплекса развития HP 64000 [24, 25] представлена на рис. 6.4. В состав комплекса развития HP 64000 входят до шести отдельных комплексов развития HP 64000A (*1*), соединенных через интерфейс HP-IB с жестким дисковым накопителем *11* и построчным принтером *10*. Интерфейс HP-IB представляет собой вариант стандартного интерфейса IEEE-488, оптимизированного для быстрого обмена данными. Пропускная способность интерфейса HP-IB составляет от 150К байт/с до 1 М байт/с. Дисковый накопитель винчестерского типа может содержать от 20 до 120 Мбайт в едином конструктиве с возможностью расширения до 960 Мбайт. Многоканальная структура, быстрый интерфейс и большая внешняя память комплекса HP 64000 позволяют коллективу разработчиков проектировать и отлаживать программное обеспечение сложных систем. На комплексе развития HP 64000 одновременно могут работать шесть разработчиков, обмениваться между собой программами и данными.

Комплекс развития HP 64100A представляет собой конструктивно законченный прибор, в котором размещаются: дисплей на электронно-лучевой трубке *7*; клавиатура *6*; микро-ЭВМ *8*; в зависимости от конфигурации комплекса тот или иной внутрисхемный эмулятор *2*; устройство программирования ППЗУ *4*; кассетный накопитель на магнитной ленте *3*; источник питания *9*. Кроме того, к прибору подключается электрофизическое сопряжение *ЭФС* (рис. 6. 5), выполненное в виде отдельного, небольшого по размерам, конструктива. Оно помещается рядом с проектируемой системой, связывая последнюю с комплексом. Электрофизи-

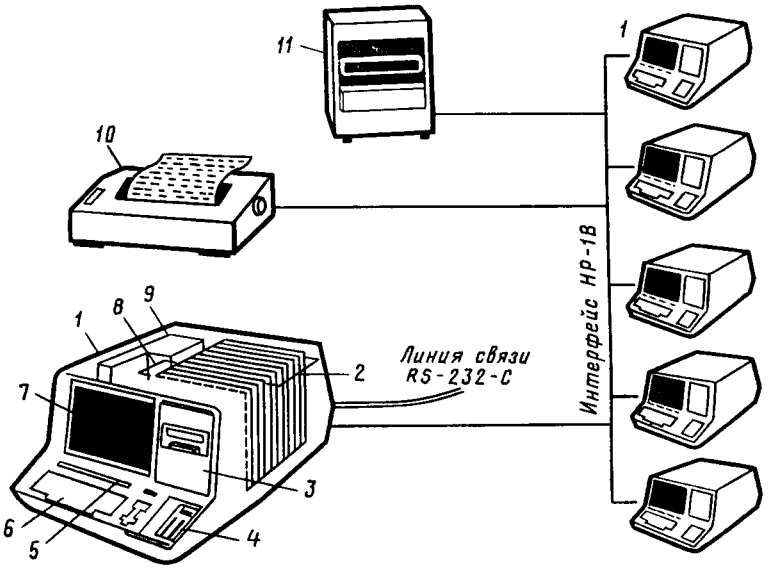


Рис. 6.4. Структура комплекса развития HP 64000 фирмы «Hewlett—Packard»

ческое сопряжение для каждого эмулируемого микропроцессора свое.

Комплекс развития HP 64100A имеет двухмагистральную организацию, что позволяет микро-ЭВМ и внутрисхемному эмулятору работать одновременно не мешая друг другу. Микро-ЭВМ размещается на трех ячейках и содержит: 16-разрядный микропроцессор *МП*; *ОЗУ* ем-

костью 64К байт, *ПЗУ* емкостью 16К байт; контроллеры интерфейсов HP-IB и RS-232-C; контроллер кассетного накопителя на магнитной ленте; контроллер дисплея.

Внутрисхемный эмулятор (рис. 6.5) содержит: эмулятор микропроцессора *ЭМП*, эмуляционную память *ЭП*, анализатор *А*. Комплекс развития HP 64100A имеет внутрисхемные эмуляторы для микропроцессоров 8080, 8085, 6800 и Z80. Эмуляционная память состоит из управления памятью *УП*, которая содержит карту адресов, и ячеек накопителя *Н*. Управление памятью связано с накопителем

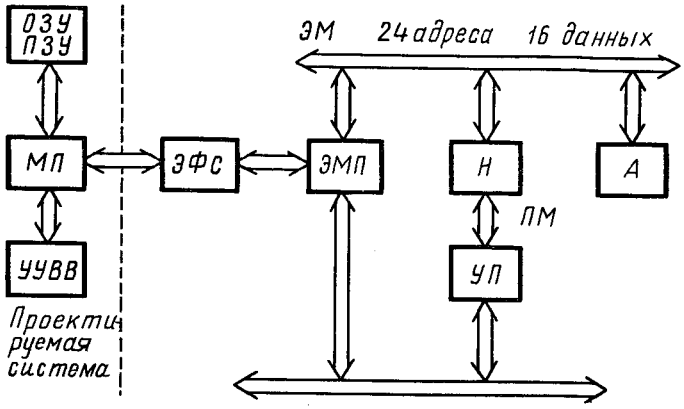


Рис. 6.5. Структура внутрисхемного эмулятора комплекса развития HP 64000 фирмы «Hewlett-Packard»

с помощью одной магистрали *ПМ*. Карта адресов позволяет разделить область памяти между проектируемой системой и эмулируемой памятью с точностью до блока объемом 1К байт. Емкость эмуляционной памяти может составлять от 8К до 128К байт в зависимости от конфигурации комплекса. Время цикла памяти 200 нс.

Анализатор *А* размещается на одной ячейке, имеет в своем составе логический компаратор и память трассировки. Логический компаратор позволяет отслеживать события на магистралях адреса, данных и управления эмулятора *ЭМП* и при обнаружении события подавать сигналы эмулятору микропроцессора и памяти трассировки. Память трассировки имеет емкость 256 40-разрядных слов.

Программатор обеспечивает программирование большинства существующих ППЗУ. Программатор состоит из универсальной управляющей ячейки и модулей персональностей семи типов. В зависимости от того, какое ППЗУ программируется, на передней панели комплекса устанавливается соответствующий модуль персональности.

Кассетный накопитель на магнитной ленте (емкость 225К байт до 128 файлов) обеспечивает загрузку операционной системы, запись файлов с накопителя на жестком диске, взаимодействие между отдельными комплексами развития HP 64100A.

На передней панели комплекса развития HP 64100A имеется восемь программируемых функциональных клавиш *5* (см. рис. 6.4). Эти клавиши позволяют пользователю настраивать комплекс на выполнение определенного вида работ, например, редактирование, компиляцию, соединение программных модулей, эмуляцию, программирование ППЗУ, организацию командного файла. На экране дисплея индицируются вид выполняемой работы и функция каждой клавиши. Если оператор хочет произвести какую-то работу, например редактирование составленной ранее программы, он нажимает на клавишу редактирования. При этом обозначения программируемых клавиш меняются, что позволит оператору выбрать соответствующие редактирующие функции. В то же время в оперативную память комплекса из накопителя на жестком диске загружается программа-редактор. При вызове программы-редактора программируемые функциональные клавиши выполняют следующие функции: вставить, исправить, изъять, найти, заменить, объединить, скопировать, извлечь, вызвать, перенумеровать, повторить, установить табулятор и перечислить. (Если существует более восьми возможностей выбора функции, как в приведенном случае, над восьмой клавишей на экране появляется метка ETC (и т. д.), при нажатии восьмой клавиши воспроизводятся следующие функции.) Если выбирается функция *изъять*, метки программируемых клавиш меняются на *включить*, *до*, *все*. Если затем выбирается функция *включить*, метки опять меняются, предоставляя на выбор *последовательность*, *номер строки*, *начало* или *конец*.

Возможности выбора для других функциональных клавиш ориентируют пользователя на конкретные наборы команд, которые синтаксически правильны для последовательностей работ независимо от сложности этих последовательностей. Такой направленный синтаксис програм-

мируемых функциональных клавиш обеспечивает простой и удобный интерфейс «человек — машина». Фактически сам комплекс говорит пользователю, какую команду он ожидает, вместо того чтобы просто выдавать сообщение об ошибке при вводе неправильной команды или данных.

§ 6.5. Проектирование с использованием внутрисхемного эмулятора

Оптимальная процедура проектирования аппаратурного обеспечения микропроцессорной системы разбивается на ряд последовательных этапов. Применение внутрисхемной эмуляции позволяет разработчику свободно пользоваться этим подходом [26].

Типичная аппаратурная часть проектируемой системы при внутрисхемной эмуляции строится следующим образом. Сначала проектируемая система не содержит ничего, кроме гнезда микропроцессора и магистрали системы. Разработчик составляет схему распределения всей программной памяти и системы ввода — вывода, указывает, что вся память пользователя и вся его система ввода — вывода будут физически размещены в комплексе развития (рис. 6.6, *а*). Как только отлажена программа, можно использовать комплекс развития для перевода отлаженного кода из памяти эмуляции в полупостоянную (программируемую) память, добавляемую к проектируемой системе. По мере развития проектируемой системы к ней может добавляться и память данных. Каждый раз, когда в проектируемую систему переходит блок памяти, разработчику необходимо перестроить свою схему распределения памяти, чтобы показать новое расположение какого-либо ее сегмента (рис. 6.6, *б*). Наконец, к проектируемой системе могут быть подсоединены необходимые контроллеры ввода — вывода. Здесь разработчик также может перенести порты ввода — вывода из комплекса развития в проектируемую систему, используя специальные команды развития (рис. 6.6, *в*).

Возможность использования памяти комплекса развития и ввода — вывода в качестве компонентов проектируемой системы приобретает особенно важное значение при поиске слабых мест разработки. Например, когда при исполнении программы в памяти проектируемой системы появляется ошибка, разработчик должен определить, чем она порождена — разрабатываемым программным обеспе-

чением или аппаратурой проектируемой системы. Для этого исполнение программы осуществляется с использованием памяти комплекса развития. Чтобы выявить ошибки в операциях ввода — вывода, можно применять ввод — вывод комплекса развития вместо ввода — вывода проектируемой системы.

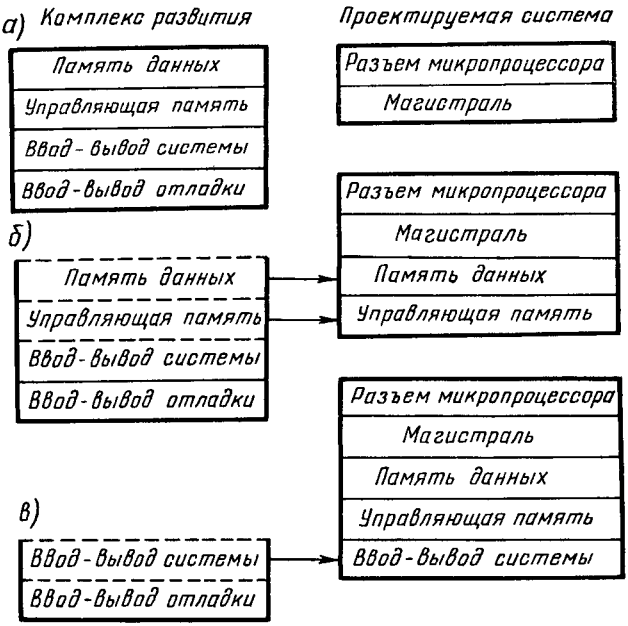


Рис. 6.6. Типичная схема разработки аппаратурной части проектируемой системы с использованием внутрисхемного эмулятора

Внутрисхемная эмуляция позволяет разработчику исполнять и отлаживать свое программное обеспечение в масштабе времени, близком к реальному, на технических средствах проектируемой системы.

Универсальные возможности внутрисхемного эмулятора можно показать на примере пяти отладочных приемов, используемых при разработке систем: останов, опрос — модификация содержимого памяти или регистров центрального процессора, пошаговое отслеживание, отслеживание в реальном масштабе времени и временно́е согласование программ [26].

Важный отладочный прием — останов. Программист может остановить эмуляцию в реальном времени на определенной команде. С помощью комплекса развития устанавливают несколько точек останова (число точек останова различно для различных комплексов развития). Пользователь может прервать доступ к любому порту ввода — вывода, вызвать команду из любой ячейки, записать или считать из памяти и т. д. Применение останова полезно в тех случаях, когда сбой в операции произошел из-за логической ошибки в программе или неправильного аппаратурного построения. При останове схемотехник с помощью команд комплекса развития может проверить содержимое ячеек памяти, регистров процессора и портов ввода — вывода, а программист изменить значения памяти или регистров, чтобы проверить их влияние на свою программу, или вставить изменения в сам программный код.

Более детальные сведения о работе программы разработчику обеспечивает пошаговое исполнение программы. С помощью пошаговой эмуляции разработчик может осуществлять индикацию содержимого регистров программы пользователя сразу же после выполнения каждой команды. Благодаря этому свойству, называемому *отслеживанием,* разработчик имеет возможность сравнить результаты выполнения каждой команды с результатом, который ожидался при написании программы. Отслеживание в масштабе времени, близком к реальному, представляет разработчику информацию о нескольких последних командах перед остановом. Если происходит останов и в регистрах или памяти появляются ошибочные данные, то было бы полезно знать, как программа вышла на эту точку останова. Отслеживание в реальном времени позволяет проектировщику «проиграть» проектируемую систему в реальном времени, сохраняя при этом возможность отслеживать ошибки.

При разработке программ синхронизация занимает важное место. Обычно внутрисхемные эмуляторы обеспечивают подсчет точного числа тактовых циклов, необходимых для выполнения некоторой части программы. Эти и другие свойства внутрисхемных эмуляторов делают их мощным средством отладки программ и позволяют проектировщику находить практически любые узкие места как в аппаратурной, так и программной части проектируемой системы.

Глава 7

КОМПЛЕКСЫ СРЕДСТВ ОТЛАДКИ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

§ 7.1. Обобщенная структура комплексов средств отладки

Рассмотренные в предыдущих главах средства отладки микропроцессорных систем: логические анализаторы, генераторы слов, комплексы развития, комплексы диагностирования, отладочные комплексы — автоматизируют либо один из этапов отладки, либо одну из функций средств отладки. Так, логические анализаторы наблюдают за поведением микропроцессорных систем, но не могут управлять их поведением, генераторы слов управляют поведением микропроцессорных систем, но не могут наблюдать за их поведением; комплексы развития ведут отладку микропроцессорных систем на программном уровне в режиме, близком к реальному, но не могут этого делать на логическом уровне и в реальном режиме, когда проектируемая система отключается от комплекса развития, а в проектируемую систему устанавливается микропроцессор; комплексы диагностирования управляют поведением и наблюдают за поведением микропроцессорных систем, однако не позволяют вести автономную отладку программных средств проектируемой системы; отладочные комплексы автономно отлаживают программные средства.

Если при переходе от одного этапа отладки к другому меняются средства отладки, то это создает большие неудобства. На рабочем месте у разработчика скапливается множество приборов: инструментальная ЭВМ, логический анализатор, комплекс развития, генератор слов, аналоговые приборы — осциллограф, частотомер, вольтметр, генератор импульсов. Каждый прибор требует знаний своего языка команд управления. Но основная сложность состоит в том, что необходимо уметь точно сопоставлять поведение проектируемой системы на различных уровнях описания. Например, если обнаружен участок программы, на котором поведение системы отличается от задуманного, от требуемого, от эталонного, то для локализации места неисправности нужно знать, как ведет себя отлаживаемая система при выполнении этого участка программы на логическом уровне, на схемном уровне. Это следует из того, что проектные неисправности, заложенные на одном уровне (например, схемном: неправильно рассчитано электри-

ческое сопряжение; не учтено влияние проводников друг на друга), или возникшие физические неисправности могут проявиться на другом уровне описания поведения системы (например, на логическом: нарушена вход-выходная последовательность разных элементов системы; на программном: нарушена последовательность адресов и данных). Различные приборы и комплексы, не объединенные общей базой данных, общим программным обеспечением, представляют информацию на разных языках, что составляет сложность для анализа. Кроме того, при наличии множества приборов сложность локализации места неисправности состоит в синхронизации этих приборов.

Степень сложности современных микропроцессорных систем такова, что один разработчик не в состоянии спроектировать систему полностью; обычно бригада разработчиков должна это делать совместно. Другой аспект сложности современных систем — это увеличение сложности и объема программного обеспечения. Обе эти особенности требуют модульного построения системы.

Существующие микропроцессорные системы содержат несколько асинхронно работающих модулей, каждый из которых базируется на МП. При отладке подобных систем возникают дополнительные сложности, связанные с наблюдением и управлением отдельными микропроцессорами и их взаимодействием между собой. Необходимо уметь собирать информацию по нескольким независимым синхросигналам, уметь анализировать и обнаруживать события нескольких независимых процессоров, управлять этими процессами. Для этого может потребоваться несколько логических анализаторов, внутрисхемных эмуляторов для 8- и 16-разрядных микропроцессоров, генераторов слов.

Средства отладки микропроцессорных систем, которые позволяют полностью проводить отладку систем на различных уровнях их представления, называются *комплексами средств отладки* (рабочие станции интеграции).

Существует два подхода к построению комплексов средств отладки микропроцессорных систем. Первый заключается в том, что отдельные комплексы, анализаторы, генераторы слов, осциллографы и другие объединяются через стандартные интерфейсы RS-232-C или IEEE-488 в систему, управляемую от одной центральной ЭВМ. В единую систему объединяются серийно выпускаемые приборы. Поэтому можно создавать различные комплексы средств отладки. Недостатком такого подхода является, во-первых, то, что сопряжение с помощью стандартных

интерфейсов, например RS-232-C, слишком медленное, когда надо передавать большой объем информации, а во-вторых, трудно, если вообще допустимо, осуществить взаимодействие отдельных приборов и комплексов между собой в режиме реального времени.

Второй подход к построению комплексов средств отладки заключается в создании специального комплекса с необходимыми функциями средств отладки. Второй подход рассмотрим на примерах описания комплексов «Электроника НЦ-803» и ATLAS.

§ 7.2. Комплекс средств отладки «Электроника НЦ-803»

Комплекс средств отладки «Электроника НЦ-803» [27] спроектирован как универсальное средство для отладки микропроцессорных устройств на базе различных микропроцессорных наборов, он может использоваться также для контроля и диагностирования дискретных устройств в процессе производства и эксплуатации. Комплекс имеет постоянную часть, независимую от проектируемой системы, и переменную часть, меняющуюся в зависимости от системы.

Комплекс средств отладки «Электроника НЦ-803» (рис. 7.1) состоит из *ЭВМ*, блоков логических устройств «Электроника Н МС 59401.1» *БЛУ*, модуля персональ-

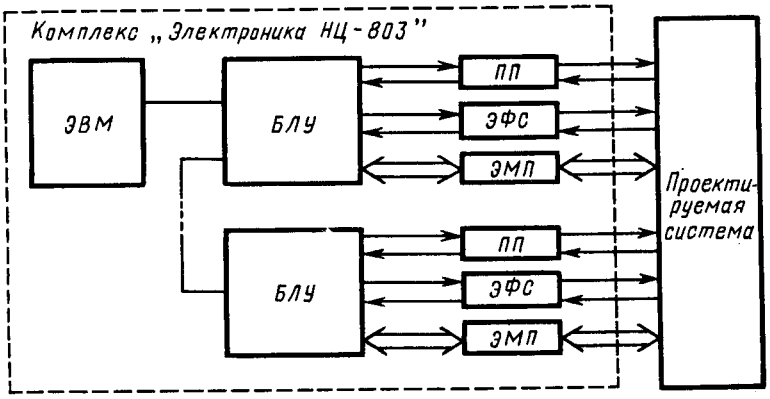


Рис. 7.1. Структура комплекса средств отладки «Электроника НЦ-803»

ности *ПП*, или/и электрофизического сопряжения *ЭФС*, или/и эмулятора микропроцессора *ЭМП*. В качестве ЭВМ может быть использована любая ЭВМ, совместимая по архитектуре с СМ ЭВМ с операционной системой ОС РАФОС: ДВК «Электроника НЦ-8020»; «Электроника-60», СМ-3; СМ-4; «Электроника 100-25»; «Электроника-79» и т. д. Электронно-вычислительная машина с БЛУ соединяется с помощью интерфейса МПИ (ОСТ 11 305.903 — 80). К одной ЭВМ может быть подключено до восьми блоков логических устройств «Электроника Н МС 59401.1». В зависимости от кабеля, соединяющего БЛУ, последние работают синхронно или автономно. Синхронно могут работать до четырех блоков логических устройств.

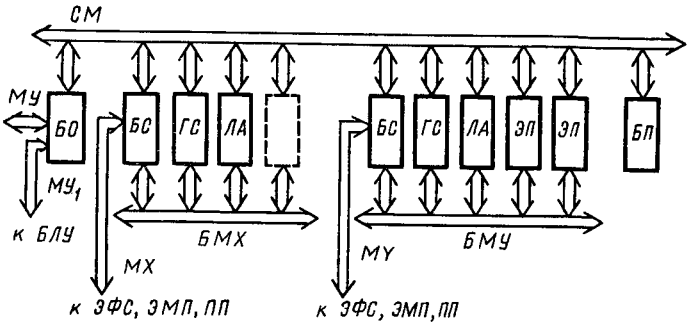


Рис. 7.2. Структура блока логических устройств «Электроника Н МС 59401.1»

Структура блока логических устройств «Электроника Н МС 59401.1» представлена на рис. 7.2. Блок логических устройств может содержать до десяти функциональных устройств, каждое из которых конструктивно выполнено на отдельной плате. На рисунке показан состав блока логических устройств, формируемый на заводе. Устройства *БО* и *БС*, а также блок питания *БП* представляют постоянную часть блока; остальные устройства могут входить в состав блока в разных количествах от одного до четырех или вообще отсутствовать. Устройства взаимодействуют между собой с помощью магистралей. Магистраль *СМ* обеспечивает обмен данными между ЭВМ и устройствами, а также взаимодействие устройств между собой в режиме реального времени. Магистраль *МУ* связывает ЭВМ с *БЛУ*.

магистраль $МУ_1$ — БЛУ с *БЛУ*. Магистрали *БМХ* и *БМУ* передают 32 разряда данных от устройств и принимают 36 разрядов данных (из них четыре квалификатора). Магистрали *МХ* и *МУ* связывают *БЛУ* с проектируемой системой через *ЭФС*, или *ПП*, или *ЭМП*. Магистраль *МХ (МУ)* выходит на четыре разъема, расположенных на стенке *БЛУ*.

Устройство *БО* обеспечивает взаимодействие между *БЛУ* и ЭВМ; устройство *БО* принимает радиальные запросы от отдельных устройств *БЛУ* и, если требуется, вырабатывает вектор прерывания ЭВМ. Устройство *БО* управляет режимами работы БЛУ: *опрос, контроль*. В режиме *опрос* ЭВМ доступны все программно-доступные регистры, память *БЛУ*. В этом режиме через ЭВМ можно устанавливать устройства в заданное состояние. В режиме *контроль* устройства *БЛУ* выполняют заданный алгоритм функционирования. Устройство *БО* содержит генератор синхросигналов для работы *БЛУ*, сигналы синхронизации выводятся на внешнюю панель *БЛУ*. Имеется возможность синхронизации работы *БЛУ* от внешнего генератора. В устройстве *БО* имеется таймер, который представляет собой 16-разрядный счетчик и отсчитывает время между заданными событиями, отслеживает заданные интервалы времени между заданными событиями, подсчитывает число заданных событий, а также может быть счетчиком времени. В качестве событий выступают прерывания от устройств *БЛУ*.

Устройство *БС* согласует магистрали *БМХ* и *БМУ* с *ЭФС, ПП* и *ЭМП*. Устройство *БС* содержит 32-разрядный порт ввода — вывода и 16-разрядный регистр фиксации. Устройство *БС* позволяет логически отключить *БЛУ* от проектируемой системы, а также замкнуть входные и выходные линии *БМХ* и *БМУ* для самотестирования.

Устройство *ГС* предназначено для генерации воздействий по 32 каналам на проектируемую систему и анализа реакций от проектируемой системы и позволяет эмулировать отсутствующие аппаратурные средства проектируемой системы. Устройство *ГС* содержит 34-разрядную память команд и данных емкостью 1024 бит, построенную на БИС КР132РУ4. Восемь разрядов этой памяти поступают на дешифратор команд микропрограммного управления. Устройство *ГС* может выполнять следующие микрокоманды: выдавать данные; изменять частоту выдачи данных; ожидать прихода заданной комбинации значений сигналов от проектируемой системы; устанавливать маску на вход-

ные сигналы, адрес возврата, режим шаг или автомат, безусловный переход; изменять интервал времени ожидания; осуществлять переход по регистру возврата; останавливать выполнение команд; выдавать сигнал радиального запроса устройства *ГС*. Допустимы комбинации микроопераций. Устройство *ГС* может быть запущено по сигналу *РР*, означающему переход из режима *опрос* в режим *контроль*, а также по сигналу от устройства *ЛА*. Максимальная частота подачи входных воздействий устройством *ГС* составляет 10 МГц.

Устройство *ЛА* состоит из памяти трассировки и логического компаратора. Память трассировки (объем 1024 32-разрядных слова, частота записи данных 20 МГц) предназначена для накопления информации о поведении проектируемой системы. Логический компаратор (ЛК) предназначен для формирования сигнала обнаружения заданного события. На вход ЛК поступают четыре квалификатора и 32 разряда данных. Логический компаратор может программироваться на обнаружение до четырех событий одновременно. Каждое условие имеет свой сигнал: ЛК1, ЛК2, ЛК3, ЛК4. Обнаруживаемое событие может представлять собой комбинацию значений 36 входных сигналов или последовательность этих комбинаций длиной 4 входных сигнала. Сигналы ЛК1, ЛК2, ЛК3, ЛК4 поступают на память трассировки, в устройство *БО*, в устройство *ГС*. На память трассировки поступают также сигналы от устройств *ГС* и *ЭП*.

Устройство *ЭП* предназначено для распределения адресного пространства памяти проектируемой системы; защиты памяти проектируемой системы; эмуляции отсутствующих аппаратурных средств проектируемой системы, ОЗУ или ПЗУ с интерфейсом МПИ (ОСТ 11 305. 903 — 80). Устройство *ЭП* содержит карту адресов и накопитель эмулирующей памяти. Карта адресов разбивает адресное пространство (2^{18}) проектируемой системы на 1024 зоны емкостью по 256 байт. Цикл эмуляции ОЗУ или ПЗУ не более 300 нс.

Модуль персональности представляет собой отдельный небольшой конструктив, который имеет плату с контактами для размещения микросхем, резисторов, конденсаторов и других изделий электронной техники и место для установки разъемов пользователями комплекса средств отладки «Электроника НЦ-803».

Электрофизическое сопряжение представляет собой также законченный конструктив, имеет восемь входов ин-

формационных, один вход-квалификатор и восемь выходов. Входное сопротивление составляет не менее 0,1 МОм, входная емкость не более 25 пФ, амплитуда входного сигнала 0 — 10 В, частота следования входных сигналов 10 МГц. Входные сигналы компарируются программно, шаг компарации составляет 0,1 В ± 10 %. Электрофизическое сопряжение содержит блок ловушек, позволяющий записывать и затем запоминать короткие импульсы. К проектируемой системе ЭФС подключается с помощью разъема или одноконтактных клипс. К БЛУ может быть подключено до восьми ЭФС.

Эмулятор микропроцессора серии 1801 представляет собой отдельный конструктив. При подключении к БЛУ другого эмулятора микропроцессора, например серии К580, необходимо заменить также устройство ЭП на другое, предназначенное для МП данной серии.

Программное обеспечение комплекса средств отладки «Электроника НЦ-803» ориентировано на систему команд СМ ЭВМ и включает следующие компоненты: средства разработки программ, входящих в состав ОС РАФОС (редакторы текста, ассемблер, компоновщик и т. д.); подсистему отладки, использующую аппаратуру БЛУ, а также подсистемы инструментального режима, тестового контроля и самодиагностирования.

Подсистема отладки обеспечивает: доступ к адресному пространству отлаживаемой программы по адресам и именам, определенным в исходном тексте программы на языке ассемблера; доступ к регистрам процессора, выполняющего программу; преобразование машинных кодов команды в текстовую строку, совпадающую с записью команды на языке ассемблера; задание точек останова отлаживаемой программы по чтению, записи, обращению по адресу (число точек останова определяется числом устройств ЛА, подключенных к проектируемой системе; каждое устройство реализует четыре точки останова); запоминание в реальном времени, а затем отображение трассы отлаживаемой программы (объем трассы определяется объемом памяти устройств ЛА, подключенных к проектируемой системе).

§ 7.3. Комплекс средств отладки ATLAS фирмы «Dolch»

Структура комплекса средств отладки ATLAS (adaptive test and logic analysis system) [28] представлена на рис. 7.3. Комплекс ATLAS состоит из основного конструк-

тива *ОК* и конструктива расширения *КР*. В основном конструктиве размещаются: микропроцессор *МП*, работающий под управлением операционной системы MP/M; контроллер дисплея *КД*; сопроцессор логического анализа *ПЛА*; накопитель на гибком магнитном диске *НГМД*, клавиатура *Кл*; контроллер стандартных интерфейсов *К*; программируемый контроллер связи *ПКС*. В основном конструктиве имеется две позиции *П1* и *П2*, в которых помещены устройства логического анализа, генератора последовательности и эмуляции. Магистраль *ПМ* обеспечивает взаимодействие с периферийными устройствами: диском, клавиатурой, дисплеем и др. Контроллер *К* связывает комплекс ATLAS по стандартным интерфейсам *СИ* RS-232-C или IEEE-488 или локальной сети с другими такими же

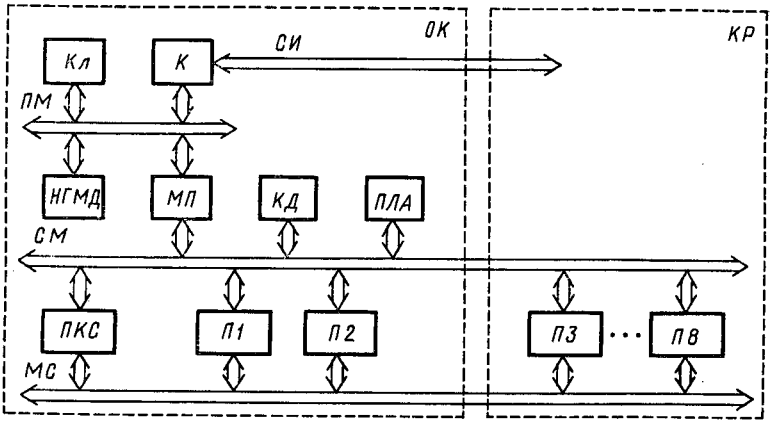


Рис. 7.3. Структура комплекса средств отладки ATLAS фирмы «Dolch»

комплексами или/и с логическими анализаторами фирмы «Dolch», а также с большой вычислительной ЭВМ. Системная магистраль *СМ* осуществляет взаимодействие основного процессора со встроенными устройствами, помещаемыми как в основном конструктиве (позиции *П1* и *П2*), так и в конструктиве расширения (позиции *П3* — *П8*). Магистраль *МС* позволяет каждому встроенному устройству узнать о наступлении события, обнаруженного другим устройством в режиме реального времени, и отвечать реакцией на это событие. Так, ЛА можно заставить запуститься сразу же после того, как эмулятор достигнет специальной

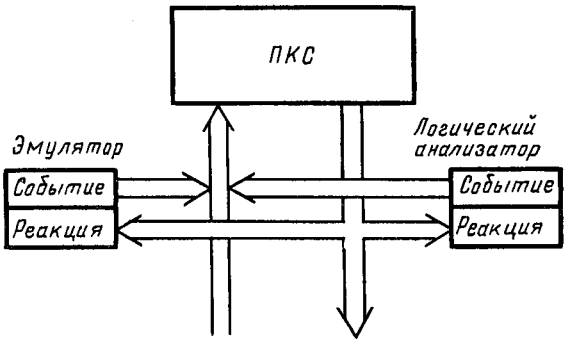


Рис. 7.4. Программируемый контроллер связи

точки прерывания. И наоборот, событие, обнаруженное ЛА, может вызвать соответствующую реакцию внутрисхемного эмулятора (рис. 7. 4). Программируемый с помощью отдельного меню контроллер связи *ПКС* объединяет все указанные средства обнаружения событий самыми разнообразными способами, что позволяет связывать эти события с различными встроенными устройствами, требующими соответствующей информации.

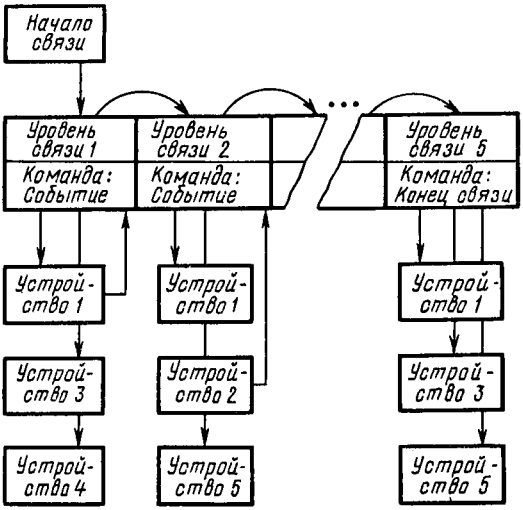


Рис. 7.5. Иллюстрация этапности обнаружения события

Если различные встроенные и другие устройства должны взаимодействовать друг с другом в режиме реального времени, то пользователи комплекса ATLAS могут использовать функцию связи системы. Программируемый контроллер связи организовывает последовательность из событий, обнаруженных отдельными устройствами. Каждое отдельное событие в последовательности определяется уровнем связи. На рис. 7. 5 проиллюстрирована организация последовательности отдельных событий, определяемая пятью уровнями связи. На каждом уровне связи ПКС определяет, какое устройство должно обнаружить событие, каким устройствам должен быть послан сигнал об этом и от какого устройства необходимо ожидать обнаружения события.

В комплексе средств отладки ATLAS используется дисплей увеличенного размера, на котором одновременно отображается несколько мини-экранов. Можно одновременно показывать трассу программы, временные диаграммы, комментарии, меню и результаты испытаний.

Программное обеспечение позволяет комплексу ATLAS эмулировать терминал VT-100 ЭВМ VAX или РДР-11, обеспечивая интерфейс между системой проектирования и ЭВМ.

Для работы с комплексом ATLAS можно использовать логические анализаторы, внутрисхемные эмуляторы, генераторы слов, анализаторы последовательностей данных, программаторы. Каждое такое встроенное устройство имеет программное обеспечение, записанное на дискете, руководство программиста и оператора. Концепция ATLAS и его документации позволяют любому спроектировать и встроить свое устройство.

Модульная гибкость комплекса ATLAS включает логический анализ с 48 каналами (на 10 и 20 МГц), 32 каналами (на 100 МГц) или 16 каналами (на 300 МГц); внутрисхемные эмуляторы 8- и 16-разрядных микропроцессоров; генераторы слов (20 и 100 МГц); программаторы ППЗУ.

Раздел 2. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Лабораторные работы, приведенные в этом разделе, позволяют привить практический навык проектирования вычислительных устройств и использования их в роли управляющих устройств. В качестве лабораторных стендов применяют учебные микро-ЭВМ УМПК 80 и МП 589, реализованные соответственно на микропроцессорных комплектах серий КР580 и К589. С помощью микро-ЭВМ УМПК 80 изучают особенности работы однокристальных микропроцессоров, а с помощью микро-ЭВМ МП 589 знакомятся со структурой микропрограммного устройства управления и особенностями микропроцессоров, управляемых на микрокомандном и командном уровнях.

Учебные микро-ЭВМ имеют встроенные сетевой источник питания, средства отображения информации в магистралях и клавиатуру для ввода исходных данных и программ. Сменные платы интерфейсных блоков позволяют подключать к микро-ЭВМ разнообразные приборы и устройства.

В зависимости от ориентации подготовки специалиста из всего перечня лабораторных работ могут быть рекомендованы наиболее близкие по профилю предполагаемых работ.

Глава 8

ОСНОВЫ ПОСТРОЕНИЯ МИКРО-ЭВМ НА МИКРОПРОЦЕССОРНЫХ КОМПЛЕКТАХ С ФИКСИРОВАННЫМ НАБОРОМ КОМАНД (НА ПРИМЕРЕ СЕРИИ К580)

§ 8.1. Учебная микро-ЭВМ для изучения проектирования вычислительных устройств на базе микропроцессорных больших интегральных схем с фиксированным набором команд

Учебная микро-ЭВМ (рис. 8.1) предназначена для знакомства с особенностями построения микро-ЭВМ на микропроцессоре с фиксированным набором команд и может

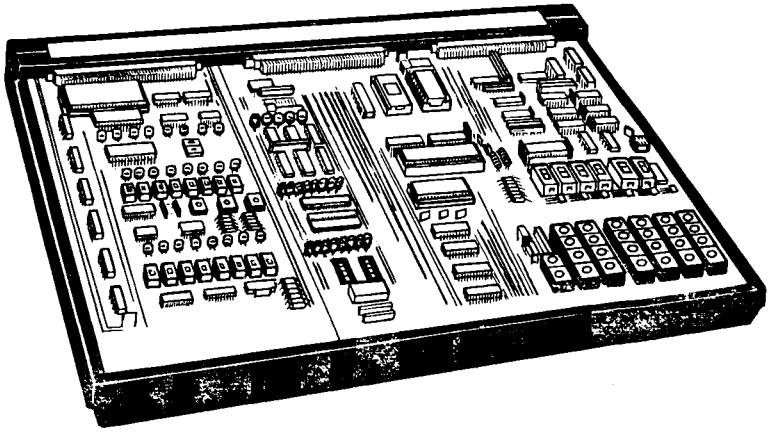


Рис. 8.1. Внешний вид учебной микро-ЭВМ на микропроцессорном комплекте серии К580

быть использована для исследования методов программирования и работы БИС, входящих в микропроцессорный комплект серии К580. Микро-ЭВМ может применяться как управляющая ЭВМ при создании и исследовании работы систем управления различными объектами. Она является легко осваиваемым и удобным средством для отладки относительно небольших (до 0,5К байт) программ пользователя. Открытая конструкция и наличие большого количества средств индикации позволяют наглядно исследовать процесс преобразования и передачи информации в микро-ЭВМ. Микро-ЭВМ имеет внутренний источник питания, обеспечивающий ее работу от сети 220 В, 50 Гц.

В задачу раздела не входит детальное описание макета, его принципиальной схемы и специфики построения операционной системы. Цель его — иллюстрация структуры и возможностей реально работающей микропроцессорной системы, ориентированной на определенный круг задач.

§ 8.2. Структура учебной микро-ЭВМ

На рис. 8.2 приведена структура микро-ЭВМ, где показаны ее отдельные блоки. Схема состоит из блока централь-

ного процессора со схемой тактового питания *СхТП;* формирователей магистралей данных *ФМД*, управления *ФМУ*, адреса *ФМА* микро-ЭВМ с магистралями: 8-разрядной МД для двунаправленного обмена данными между отдельными узлами микро-ЭВМ; 16-разрядной однонаправленной *МА*, управляемой *МП БИС* и используемой для адресации к памяти и отдельным узлам микро-ЭВМ; 5-разрядной однонаправленной магистрали управления *МУ*, управляемой *МП БИС* и служащей для определения режима работы микро-ЭВМ при выполнении каждого машинного цикла; блока памяти (БП), включающего в себя *ОЗУ* емкостью 1К байт и *ПЗУ* емкостью 2К байт; регистров ввода *РгВв* с переключателями *П* и вывода *РгВыв* с индикацией состояний светодиодами; схемы выдачи кода прерывания *СхВКПР;* клавиатуры управления и ввода данных *К*, состоящей из 9 клавиш управления, переключателя начальной установки микро-ЭВМ и 16 клавиш для введения шестнадцатеричных кодов чисел. Для ввода информации в микро-ЭВМ клавиатура имеет регистр чтения клавиатуры *РгЧК;* 6-разрядный восьмисегментный дисплей *Д*, четыре разряда которого служат для отображения кода адреса в шестнадцатеричной системе исчисления, а два — для отображения данных. Для вывода информации дисплей *Д*

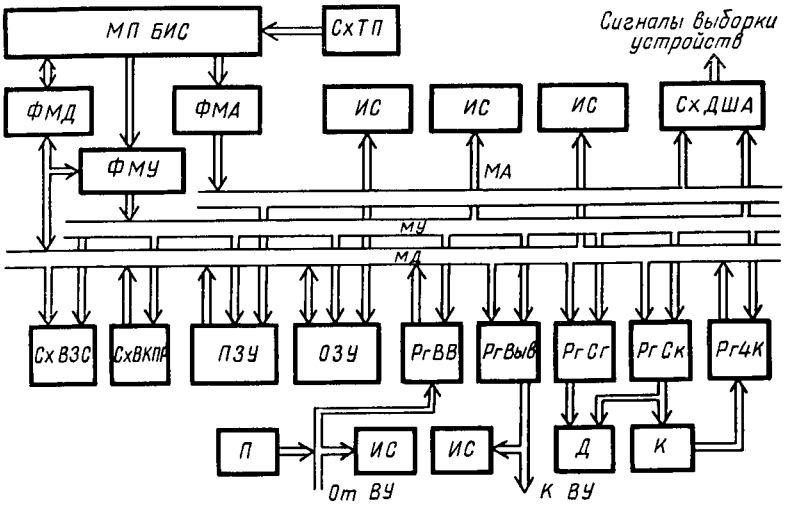


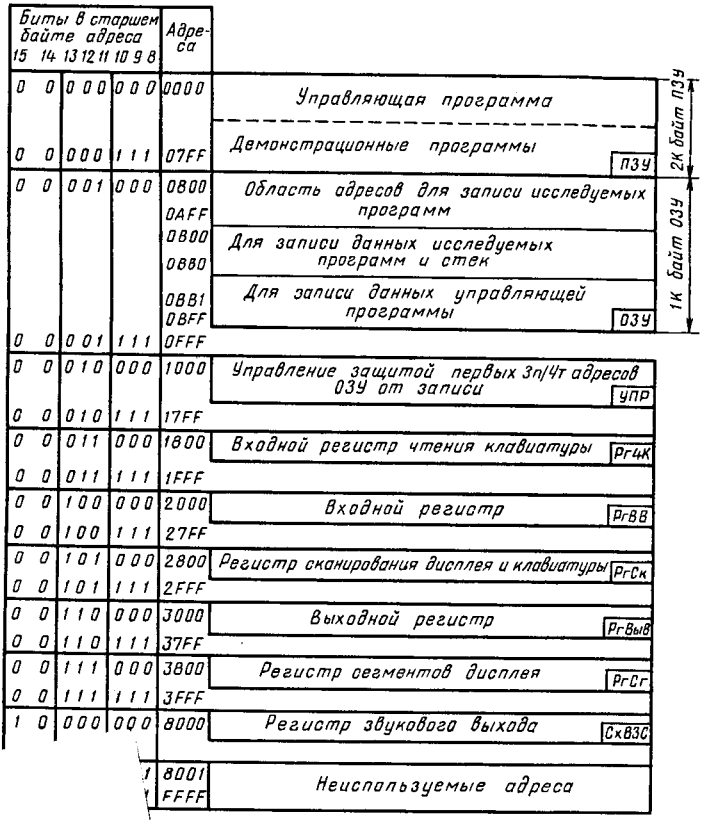
Рис. 8.2. Структура микро-ЭВМ

имеет два регистра: регистр сегментов дисплея *РгСг* и регистр скандирования *РгСк* и схемы выдачи звукового сигнала *СхВЗС*.

Магистрали микро-ЭВМ, а также входной и выходной регистры имеют светодиоды *ИС*, указывающие их состояние.

§ 8.3. Адресация в учебной микро-ЭВМ

В микро-ЭВМ реализована *адресация с помощью карты памяти:* каждому физическому устройству макета ставится в соответствие адрес, и обращение к нему при таком ти-



а памяти микро-ЭВМ

пе адресации аналогично обращению к ячейке памяти с использованием всего набора команд МП БИС. На рис. 8.3 приведена карта памяти микро-ЭВМ, из которой видно, что первые 2К байт адресов составляет *ПЗУ,* в котором записаны управляющие и демонстрационные программы; адреса с 0800_{16} по $0BFF_{16}$—*ОЗУ*; адрес 0BBO — начальный адрес стека, а адреса с OBBI по OBFF задействуются для временной записи данных во внутренних регистрах МП БИС при работе управляющей программы. Адрес 0800 является также начальным адресом *ОЗУ,* куда могут записываться исследуемые программы пользователя. Большинство из приведенных в лабораторном практикуме программ начинаются с адреса 0800 и могут быть без изменения адресов исследованы на данном макете. Следует отметить, что область *ОЗУ* с адреса 0800 по адрес 0AFF в учебной микро-ЭВМ схемотехнически защищена от случайной записи во время выполнения программ пользователя. Для записи данных при выполнении программ необходимо использовать область *ОЗУ* с адреса 0BOO по адрес 0BBO.

Такая адресация позволяет легко осуществить дешифрацию устройств на основе простого 3-разрядного дешифратора. Идея дешифрации ясна из рис. 8.3, на котором представлены также состояния старших восьми разрядов кода адреса, по которым осуществляется адресация к устройствам микро-ЭВМ. Как видно из рисунка, для дешифрации устройств можно использовать лишь 11, 12 и 13-й разряды адресной магистрали. Любая из восьми комбинаций состояний этих разрядов однозначно определяет вид устройства, с которым будет работать МП БИС на каждом машинном цикле.

§ 8.4. Режим работы учебной микро-ЭВМ и алгоритм управляющей программы

Управляющая программа микро-ЭВМ состоит из программы тестирования отдельных узлов микро-ЭВМ, начальной установки содержания внутренних регистров МП БИС и ячеек ОЗУ, сохранения содержания внутренних регистров МП БИС, выдачи сообщения на дисплей, чтения и декодирования клавиатуры, обслуживания звукового выхода и программ, обеспечивающих режимы:

1) ожидания ввода команд управления с клавиатуры;

2) отображения на дисплее любого адреса и его содержания;

3) ввода с клавиатуры кода любого адреса с одновременным отображением его на дисплее и после ввода последней цифры адреса микро-ЭВМ автоматического перехода к выполнению режима 2 (начало режима ввода кода адреса с клавиатуры инициируется нажатием клавиши ОТА — *отыскание адреса)*;

4) обращения к программному счетчику. Выполнение этого режима вызывается нажатием клавиши ПРСч *(программный счетчик)* и приводит к автоматическому вводу содержания программного счетчика на дисплей адреса с последующим переходом к режиму 2;

5) вывода на дисплей содержимого внутренних программно-доступных регистров МП БИС с указанием на дисплее названия выводимого регистра. Этот режим устанавливается нажатием клавиши ОТРг *(отыскание регистра)*, при этом на дисплей будет выведено содержание аккумулятора МП БИС. Последующие нажатия на клавишу ЗпУВ *(запись увеличить)* приведут к циклическому выводу содержания всех внутренних регистров МП БИС. Содержимое регистров будет выводиться на дисплей в такой последовательности: A, FL, B, C, D, E, H, L, SPH, SPL, PCH, PCL;

6) записи с клавиатуры в ячейки ОЗУ и внутренние программно-доступные регистры МП БИС нового кода. При работе микро-ЭВМ в режимах 2 и 5 после изменения кода чисел, представленных на дисплее данных, запись чисел инициируется нажатием на клавишу ЗпУВ;

7) увеличения или уменьшения на единицу адреса, представленного на дисплее при работе микро-ЭВМ в режиме 2. Эти режимы инициируются соответственно нажатием клавиш ЗпУВ и УМ *(уменьшить)*;

8) запуска программы с адреса, указанного на дисплее, с помощью клавиши П *(пуск)*;

9) выполнения команд по машинным циклам с отображением информации на магистралях микро-ЭВМ с помощью светодиодов. Режим устанавливается нажатием клавиши ШЦ *(шаг машинного цикла)*;

10) выполнения программы по командам. Режим устанавливается нажатием клавиши ШК *(шаг команды)*, при этом после каждой команды управляющая программа автоматически переходит к режиму 4;

11) останова программы пользователя. Режим устанавливается при нажатии клавиши СТ *(стоп)* с автоматическим сохранением в ОЗУ содержимого всех внутренних регистров МП БИС и переходом к выполнению режима 4.

Сохранение регистров позволяет анализировать их содержание на момент останова с помощью перехода к режиму 5. После останова программы микро-ЭВМ может быть переведена на режимы 9, 10;

12) возврата микро-ЭВМ из режима выполнения команд по машинным циклам на управляющую программу с автоматическим восстановлением содержания внутренних регистров МП БИС и переходом к режиму 4.

Работа микро-ЭВМ рассчитана так, что после подачи на нее питания запускается программа тестирования всех основных узлов. Эта программа начинается с ячейки ПЗУ с адресом 0000_{16}. В результате выполнения теста проверяются правильность записанной в ПЗУ информации и возможность ее верного считывания; содержимое ОЗУ на отсутствие ошибок при записи/считывании чисел; МП БИС на правильность выполнения им простейших арифметических и логических операций; запись чисел в регистр вывода и работоспособность всех его светодиодов состояния, а также работа всех сегментов дисплея. В случае обнаружения ошибок в ОЗУ или ПЗУ подается прерывистый звуковой сигнал и на дисплей выводится соответствующее сообщение: ОЗУ или ПЗУ. При успешном завершении теста микро-ЭВМ проводит начальную установку всех регистров МП БИС и на дисплей выводится сообщение НАЧАЛО, свидетельствующее о готовности ее к работе.

На рис. 8. 4 приведена схема возможных переходов от одного режима работы микро-ЭВМ к другому. В режиме ожидания ввода команд управления микро-ЭВМ может реагировать на нажатие трех клавиш: ОТА *(отыскание адреса)*, ОТРг *(отыскание регистра)* и ПРСч *(программный счетчик)*. Все последующие возможные переходы от режима к режиму можно проследить на рис. 8. 4, где сплошными линиями показаны возможные пути изменения режимов работы по командам пользователя, вводимым с клавиатуры управления, а штриховыми — изменения в состояниях микро-ЭВМ, вызванные выполнением соответствующих подпрограмм команд управления.

Для останова работы исследуемой программы или при возврате на управляющую программу при работе микро-ЭВМ в режиме выполнения программы по машинным циклам используется клавиша СТ *(стоп)*. Нажатие этой клавиши в указанных режимах приводит к останову режима, при этом на дисплей выводится содержание программного счетчика МП БИС. Нажатие клавиши СТ *(стоп)* при другом режиме приведет к переходу микро-ЭВМ в режим ожи-

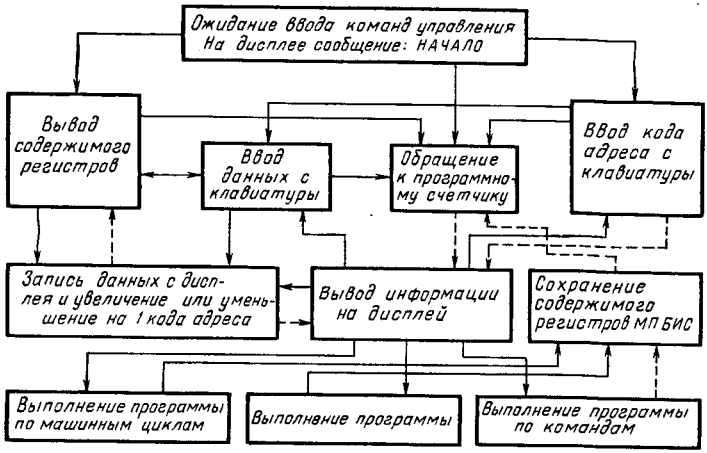


Рис. 8.4. Схема перехода к различных режимам работы в микро-ЭВМ

дания ввода команд управления, при этом на дисплей будет выводиться сообщение НАЧАЛО.

Для удобства отладки программ в микро-ЭВМ предусмотрена возможность останова выполнения программ по любым заранее заданным адресам с помощью одного из векторов системы прерывания.

Программы, записанные в оставшейся части ПЗУ, можно условно разбить на два типа: 1) вспомогательные подпрограммы выполнения арифметических и алгебраических операций (например, умножение, деление двух восьмибитовых чисел и т. д.), которые могут применяться пользователем при создании более сложных программ; 2) демонстрационные подпрограммы (например, подпрограмма, используемая для записи и воспроизведения с помощью звукового выхода музыкальных произведений по нотам, коды которых можно хранить в виде чисел в ПЗУ или ОЗУ, подпрограмма-«секундомер» и т. д.).

Имеется открытый доступ к магистралям микро-ЭВМ, что позволяет исследовать его совместную работу с БИС, входящим в микропроцессорный набор серии КР580.

Приведенное ниже описание лабораторного практикума в основном охватывает исследование работы программного обеспечения микро-ЭВМ, построенных на микропроцессорном наборе серии КР580. Исследование программно-аппаратурных средств создания микро-ЭВМ про-

водится лишь в тех случаях, когда работа их взаимосвязана и не может быть рассмотрена в отдельности. Большинство из приведенных программ могут быть без каких-либо изменений исследованы на учебной микро-ЭВМ. При ее отсутствии они могут быть исследованы на любых других макетах или средствах отладки, имеющих сходные функции с описанной учебной микро-ЭВМ.

Для успешного выполнения лабораторных работ необходимо предварительное ознакомление с языком программирования и командами МП БИС КР580ИК80.

Примечание. Язык программирования и структура команд МП БИС КР580ИК80 изложены в книгах: *Балашов Е. П., Пузанков Д. В.* Микропроцессоры и микропроцессорные системы.— М.: Радио и связь, 1981; *Соучек Б.* Микропроцессоры и микро-ЭВМ: Пер. с англ.— М.: Советское радио, 1979; *Алексеенко А. Г., Галицын А. А., Иванников А. Д.* Проектирование радиоэлектронной аппаратуры на микропроцессорах.— М.: Радио и связь, 1984.

§ 8.5. Лабораторные работы

Лабораторная работа 8.1. Ознакомление с работой на учебной микро-ЭВМ

Цель работы: *ознакомление со структурой учебной микро-ЭВМ, картой памяти, органами управления и режимами работы.*

Краткие сведения из теории

Назначение микро-ЭВМ, ее структура и карта памяти рассмотрены выше. Здесь же еще раз укажем процессы, происходящие при включении и начальной установке микро-ЭВМ.

После включения учебной микро-ЭВМ или нажатия клавиши начальной установки *R* содержание программного счетчика МП БИС обнуляется и выполнение программы начинается с нулевой ячейки памяти, где записаны программы тестов узлов микро-ЭВМ.

После выполнения тестов производится начальная установка всех внутренних программно-доступных регистров МП БИС. Начальная установка состоит в том, что в программный счетчик записывается число 0800 (первый адрес ОЗУ), в указатель стека — число ОВВО, а остальные регистры МП БИС обнуляются.

Задания для домашней подготовки

1. Ознакомьтесь с описанием учебной микро-ЭВМ.
2. Ознакомьтесь с типовой минимальной структурой микро-ЭВМ, методами организации магистралей, подключения памяти и внешних устройств к магистралям.
3. Изучите алгоритм работы управляющей программы и возможные режимы работы по рис. 8.4.
4. Рассмотрите работу МП БИС в режиме ОЖИДАНИЕ и состояние магистралей микро-ЭВМ при его выполнении.
5. Изучите внутренние регистры МП БИС КР580ИК80 и временные диаграммы выполнения команд.

Задания к лабораторной работе

Задание 1. Исследовать порядок включения микро-ЭВМ. *Порядок выполнения задания:* 1. Подключить шнур питания к сети. 2. Включить тумблер СЕТЬ. В результате выполнения тестовых программ светодиоды выходного регистра и сегменты дисплея будут включены на время, равное $\sim$2 с, после чего микро-ЭВМ подает звуковой сигнал и выводит на дисплей сообщение НАЧАЛО. Находясь в этом режиме, микро-ЭВМ реагирует на нажатие клавиши управления. Вызов возможных режимов работы из этого состояния микро-ЭВМ определяется алгоритмом, приведенным на рис. 8.4. Из любого места управляющей программы можно вернуть микро-ЭВМ к начальному состоянию нажатием на клавишу *R*. В этом случае микро-ЭВМ начинает выполнять управляющую программу с нулевой ячейки памяти. В результате ее выполнения все ячейки ОЗУ будут обнулены и, следовательно, будет стерта вся программа пользователя, записанная ранее в ОЗУ. 3. Нажать на клавишу *R* и убедиться, что тестовые программы проходят заново.

Задание 2. Исследовать содержимое памяти.

Порядок выполнения задания: 1. Нажать на клавишу ОТА при этом на дисплее появится сообщение начала выполнения режима отыскания адреса (включены нижние сегменты дисплея). 2. Последовательно нажать на клавиши 0, 8, 0, 0. Убедиться при этом, что каждая цифра будет записана в младший разряд адресного дисплея и произойдет одновременный сдвиг всех знаков на адресном дисплее на один разряд влево. При введении четвертой цифры с клавиатуры микро-ЭВМ выведет на дисплей

число, записанное по этому адресу (рис. 8.4). На дисплее данного появится число 00. В случае ошибки при введении кода адреса следует нажать на клавишу ОТА и повторить ввод. **3.** Нажать на клавишу ЗпУВ. В этом режиме микро-ЭВМ увеличит на единицу адрес на адресном дисплее и выведет его содержимое на дисплей. Последовательно нажимая на клавишу ЗпУВ, проверить содержимое адресов ОЗУ. **4.** Нажать на клавишу ОТА и ввести код 0000 (первый адрес ПЗУ). На дисплее данного появится код 26. Последовательно нажимая на клавишу ЗпУВ, просмотреть содержимое нескольких ячеек ПЗУ. **5.** Убедиться, что содержимое памяти можно просмотреть с помощью последовательного нажатия на клавишу УМ. Таким образом, использование этого режима позволяет проверить содержимое всех ячеек памяти микро-ЭВМ.

Задание 3. Записать числа в память микро-ЭВМ.

Порядок выполнения задания: **1.** Включить тумблер СЕТЬ (или при включенной ранее микро-ЭВМ нажать на клавишу *R*). **2.** Нажать на клавишу ОТА и после этого набрать адрес 0800. На дисплее данного после введения адреса появится его содержимое 00. **3.** Нажать на клавишу 1. Микро-ЭВМ вводит значение цифры нажатой клавиши в младший разряд дисплея данного, при этом в этом разряде дисплея появляется десятичная точка. Она свидетельствует о том, что микро-ЭВМ находится в режиме ввода данных и число, представленное на дисплее данного, не является истинным содержимым адреса памяти, представленного на адресном дисплее. **4.** Нажать на клавишу 2. Убедиться при этом, что цифра 1 младшего разряда дисплея переместилась на старший разряд дисплея данного, а цифра 2 — на ее место (десятичная точка осталась в младшем разряде). Проверить, что последующие нажатия на цифровые клавиши приводят к вводу цифры нажатой клавиши в младший разряд дисплея данного и смещению в старший разряд предыдущей нажатой клавиши. **5.** Записать нажатием на клавишу ЗпУВ число, представленное на дисплее по адресу 0800. При этом на дисплее появятся адрес 0801 и его содержимое (00), а десятичная точка погаснет, показывая, что данные введены и микро-ЭВМ вышла из режима ввода данных. **6.** Записать любое число на дисплей данного при наличии на дисплее адреса 0801. **7.** Нажать на клавишу УМ. На адресном дисплее появится адрес 0800 с его содержимым. Убедиться при этом, что число было записано по этому адресу. Отсутствие десятичной точки в младшем разряде дисплея свиде-

тельствует о том, что это число микро-ЭВМ извлекла из памяти. 8. Нажать на клавишу ЗпУВ и проверить содержание адреса 0801. Убедиться при этом, что число, записанное на дисплее данного, не было записано в память при нажатии на клавишу УМ. 9. Осуществить вывод на дисплей содержимого адреса 0000 (ПЗУ). На дисплее будет 26. Записать по этому адресу на дисплее любое другое число. Убедиться, что при нажатии на клавишу ЗпУВ запись данных в ПЗУ невозможна и при этом микро-ЭВМ определяет это действие как ошибку, формирует звуковой сигнал, не увеличивает адрес и выводит на дисплей истинное число, записанное в ПЗУ.

Примечание. При выполнении данного задания показана методика ввода и проверки правильности ввода программ в микро-ЭВМ с помощью клавиш ОТА, ЗпУВ, УМ.

Задание 4. Записать числа в программно-доступные регистры МП БИС.

Порядок выполнения задания: **1.** Включить микро-ЭВМ. После вывода на дисплей сообщения НАЧАЛО нажать на клавишу ОТРг. На дисплей будет выведено содержание аккумулятора (А) МП БИС. Многократно нажимая на клавишу ЗпУВ, проверить последовательность вывода содержимого внутренних регистров МП БИС на дисплей. **2.** Изменить число, записанное в регистре при наличии на дисплее информации о его содержимом, с помощью цифровых клавиш клавиатуры. Нажать на клавишу ЗпУВ. **3.** Нажать на клавишу УМ и убедиться при этом в правильности записи числа в регистр.

Задание 5. Осуществить пуск программы.

Порядок выполнения задания: **1.** Для осуществления пуска программы с любого адреса памяти вывести этот адрес на дисплей. Пуск программы производится нажатием на клавишу П *(пуск)*, при этом в программный счетчик МП БИС записывается адрес, указанный на дисплее, и выполнение программы начинается с этого адреса. Записать на дисплей адрес 05В0, являющийся началом музыкальной программы, записанной в ПЗУ. **2.** Нажать на клавишу П, при этом микро-ЭВМ исполнит мелодию по нотам, записанным в ПЗУ. **3.** Убедиться, что выполнение программы может быть остановлено нажатием на клавишу СТ. Для этого повторить пуск музыкальной программы заново и при выполнении программы нажать на клавишу СТ. При останове программы на дисплей будет выводиться адрес, записанный в программном счетчике, и его

содержимое на момент останова (рис. 8.4). При останове программы проверить возможность осуществления режима вывода на дисплей содержимого регистров МП БИС на момент останова (рис. 8.4), для чего следует нажать на клавишу ОТРг, при этом на дисплее появится содержание аккумулятора МП БИС. Последовательным нажатием на клавишу *ЗпУВ* проверить вывод на дисплей содержимого регистра МП БИС. 4. Нажатием на клавишу ПРСч вывести на дисплей адрес памяти, на котором было прервано выполнение программы. Нажать на клавишу П *(пуск)* и продолжить выполнение программы.

Содержание отчета

Отчет должен содержать: 1. Схему структуры учебной микро-ЭВМ. 2. Карту памяти. 3. Информацию о содержании внутренних программно-доступных регистров МП БИС после программы начальной установки микро-ЭВМ.

Задания для самопроверки

1. Изобразите структуру учебной микро-ЭВМ.
2. Укажите функциональные части на принципиальной схеме учебной микро-ЭВМ.
3. Для какой цели в микро-ЭВМ используется работа МП БИС в режиме ОЖИДАНИЕ?
4. Что такое карта памяти микро-ЭВМ?
5. Какие адреса памяти микро-ЭВМ относятся к ОЗУ и ПЗУ?
6. Расскажите о возможных режимах работы микро-ЭВМ.
7. Как записать числа в программно-доступные регистры МП БИС или память микро-ЭВМ?
8. Что происходит в микро-ЭВМ при попытке записи данных в ПЗУ?
9. Какие узлы проверяются в микро-ЭВМ в процессе выполнения тестов при ее включении или нажатии на клавишу R?
10. Укажите возможные изменения режимов работы микро-ЭВМ, находящейся при выполнении программы по командам.
11. Опишите функции каждой клавиши управления на клавиатуре.
12. Укажите, какие значения записываются в регистры МП БИС и ОЗУ в процессе программы начальной установки микро-ЭВМ.

Литература

Горбунов В. Л., Панфилов Д. И., Преснухин Д. Л. Микропроцессоры. Основы построения микро-ЭВМ.— М.: Высшая школа, 1984.

Учебная микро-ЭВМ на базе микропроцессорного комплекта серии К580/*Панфилов Д. И., Красавин В. Н., Романенко О. А.*— Электронная промышленность, 1983, № 9.

Лабораторная работа 8.2.
Запись и выполнение простых программ

Цель работы: *исследование выполнения отдельных команд и простых программ; использование различных методов адресации в программах; запись программ.*

Краткие сведения из теории

Микропроцессор БИС КР580ИК80 имеет фиксированный набор команд. Время выполнения команды определяется процессом получения, декодирования и выполнения команды. Это время можно представить состоящим из ряда временны́х интервалов. Наиболее короткий временной интервал, равный периоду синхросигналов МП БИС, называется *машинным тактом*. Время, необходимое для извлечения 1 байт информации из памяти или внешнего устройства или выполнения команды, определяемой одним машинным словом, называется *машинным циклом*. Машинный цикл для МП БИС может включать в себя 3—5 машинных тактов. В зависимости от вида команды время выполнения может состоять из 1—5 машинных циклов. Для МП БИС имеется 10 различных типов машинных циклов: извлечение кода команды (цикл M_1), чтение данных из памяти, запись данных в память, извлечение данных из стека, запись данных в стек, ввод данных из внешнего устройства, запись данных во внешнее устройство, цикл обслуживания прерывания, останов, обслуживание прерывания в режиме останова. Первым машинным циклом при извлечении любой команды является цикл M_1.

На каждом машинном МП БИС проверяет состояние сигнала «Готов» на своем входе. Нулевой сигнал на этом входе приостанавливает нормальную работу МП БИС, при этом на магистралях микро-ЭВМ присутствует вся информация, передаваемая на рассматриваемом машинном цикле. В учебной микро-ЭВМ это используется для исследования выполнения команд по машинным тактам. В этом режиме информация на магистралях микро-ЭВМ отображается светодиодами состояния.

Программа записывается в микро-ЭВМ в последовательных ячейках памяти.

Рассмотрим простейшую программу (программа 8.1), извлекающую число из адреса памяти 0B00, инвертирующую его и записывающую результат в адрес памяти 0B01.

Программа 8.1 (в мнемокодах)

Мнемокод	Комментарий
LDA 0B00	получить число из адреса 0B00
CMA	инвертировать число
STA 0B01	записать результат по адресу 0B01
RST1	прервать выполнение программы

При записи программ все числа представляются в шестнадцатиричной системе счисления.

Для записи программы в память микро-ЭВМ необходимо перевести мнемокоды команд в машинные коды. Команды в программе могут быть одно-, двух- или трехбайтные и должны в памяти занимать соответственно один, два или три адреса.

Программа 8.1 (размещение по адресам памяти)

Адрес	Число	Комментарий
0800	3A	код команды LDA
0801	00	младший байт адреса
0802	0B	старший байт адреса
0803	2F	код команды CMA
0804	32	код команды STA
0805	01	младший байт адреса
0806	0B	старший байт адреса
0807	CF	код команды RST1

Предварительную запись программ удобно проводить в более компактной форме. В программе указывается начальный адрес каждой команды и при этом понимается, что в зависимости от длины (одно-, двух- или трехбайтная) команды в памяти будут занимать от одной до трех последовательных ячеек. При такой записи в левом столбце указываются лишь адреса команд в программе. Это позволяет сократить объем при описании программ и сделать более простым их анализ.

Программа 8.1 (общий вид записи)

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	3A 000B		LDA, 0B00	получить число
0803	2F		CMA	инвертировать число
0804	32 010B		STA, 0B01	записать по адресу 0B01
0807	CF		RST1	прервать выполнение программы

Здесь используется прямой способ адресации.

Рассмотрим программу, аналогичную программе 8.1, с использованием косвенного способа адресации (программа 8.2).

Программа 8.2

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	21 000B		LXIH 0B00	записать в регистры H,L число 0B00
0803	7E		MOV A, M	получить число из адреса, указанного в регистрах H, L
0804	2F		CMA	инвертировать число в аккумуляторе
0805	23		INX M	увеличить на 1 число в регистрах H, L
0806	77		MOV M, A	записать число из аккумулятора по адресу, указанному в H, L
0807	CF		RST 1	прервать выполнение программы

Задания для домашней подготовки

1. Ознакомьтесь с языком программирования и структурой команд МП БИС КР580ИК80.

2. Изучите режимы работы и временные диаграммы процесса выполнения команд МП БИС КР580ИК80.

3. Изучите методы программирования на языке ассемблера и в машинных кодах для МП БИС КР580ИК80.

4. Рассмотрите правила выполнения команд INR A (3C), DCR A (3D), ADD A (87), ANA A (A7), ORA A (B7), CMP A (BF), DAA (27).

5. Рассмотрите результат выполнения программы 8.1 при записи по адресу 0803 команд, приведенных в п. 4 задания. Результаты выполнения программы при различных командах, записанных в программе 8.1 по адресу 0803, занести в табл. 8.1.

Таблица 8.1

Число, записанное по адресу 0B00	Команда, записанная по адресу 0803	Число, записанное по адресу 0B01

6. Видоизмените и запишите программу 8.2 так, чтобы при ее выполнении исследуемое число первоначально записывалось по адресу 0B00.

7. Разработайте программы: а) увеличения на 5 числа, записанного по адресу 0B00, и записи результата по адресу 0B01 (программа 8.3); б) сложения чисел, записанных по адресам 0B00 и 0BA0, и записи результата по адресу 0B01 (программа 8.4); в) сравнения чисел в адресах 0B00 и 0B01 и записи большего из них в регистр B (программа 8.5).

Задания к лабораторной работе

Задание 1. Исследовать программу 8.1.

Порядок выполнения задания: **1.** Ввести в микро-ЭВМ программу 8.1. **2.** Записать по адресу 0B00 исследуемое число. **3.** Осуществить пуск программы 8.1 с адреса 0800. Проверить результат выполнения программы путем исследования числа, записанного по адресу 0B01. **4.** Исследовать процесс выполнения программы по командам. После выполнения каждой команды проанализировать содержание всех программно-доступных регистров МП БИС. **5.** Исследовать процесс выполнения команд в программе 8.1 по машинным циклам. Обратить внимание на последовательность передачи и преобразования информации в микро-ЭВМ при выполнении каждой команды. Представить временны́е диаграммы процесса выполнения любой одно- и трехбайтной команды в программе. **6.** Заменяя в программе 8.2 команду CMA на команды INR A (3C), DCR A (3D), ADD A (87), ANA A (A7), ORA A (B7), CMP A (BF), DAA (27), исследовать результат выполнения указанных команд по числу, записанному по адресу 0B01. Проверить табл. 8.1, заполненную при выполнении домашнего задания.

Задание 2. Исследовать программу 8.2.

Порядок выполнения задания: **1.** Ввести в микро-ЭВМ программу 8.2. **2.** Записать по адресу 0B00 исследуемое число. **3.** Осуществить пуск программы с адреса 0800. Проверить результат выполнения программы по числу, записанному по адресу 0B01. **4.** Исследовать процесс выполнения команды MOV A, M по машинным циклам. **5.** Ввести и исследовать выполнение микро-ЭВМ видоизмененной программы 8.2, позволяющей первоначально записывать исследуемое число по адресу 0B00.

Задание 3. Исследовать программу 8.3.

Порядок выполнения задания: **1.** Ввести в микро-ЭВМ программу 8.3, разработанную при домашней подготовке. **2.** Осуществить пуск программы 8.3 и проверить результат

ее выполнения по числу, записанному по адресу 0B01 при числах 05, FE, записанных по адресу 0B00.

Задание 4. Исследовать программу 8.4.

Порядок выполнения задания: **1.** Ввести в микро-ЭВМ программу 8.4, разработанную при домашней подготовке. **2.** Проверить результат выполнения программы по числу, записанному по адресу 0B01, последовательно записывая по адресам 0B00 и 0BA0 соответственно числа 0B и B0, FE и B5 и осуществляя пуск программы. Видоизменить и исследовать программу 8.4 для случая, когда сумма двух чисел будет превышать восьмиразрядное число.

Задание 5. Исследовать программу 8.5.

Порядок выполнения задания: **1.** Ввести в микро-ЭВМ программу 8.5, разработанную при домашней подготовке. **2.** Записать по адресам 0B00 и 0B01 исследуемые числа. **3.** Осуществить пуск программы 8.5. Проверить результат ее выполнения по числу, записанному в регистре B.

Работа на учебной ЭВМ

При выполнении лабораторной работы на учебной микро-ЭВМ следует иметь в виду, что: **1.** Ввод исследуемой программы осуществляется путем ее последовательной записи в ячейки памяти. Для записи числа по адресу используются клавиши ОТА, ЗпУВ и цифровые клавиши на клавиатуре. **2.** Пуск программы осуществляется нажатием на клавишу П. **3.** Выполнение программы по командам осуществляется нажатием на клавишу ШК. После выполнения каждой команды микро-ЭВМ позволяет выводить на дисплей содержание программно-доступных регистров МП БИС с помощью клавиш ОТРг и ЗпУВ. **4.** Выполнение программы по машинным циклам осуществляется последовательным нажатием на клавишу ШМЦ, при этом информация на дисплей микро-ЭВМ не выводится, а отображается лишь на светодиодах магистралей. Окончание режима производится нажатием на клавишу СТ, при этом на дисплей выводится адрес текущей программы, выполняемой микро-ЭВМ на момент выхода из режима выполнения команды по машинным циклам.

Содержание отчета

Отчет должен содержать: **1.** Заполненную табл. 8.1 для случаев выполнения программы 8.1 при использовании команд, представленных в п. 4 задания для домашней под-

готовки. 2. Временны́е диаграммы выполнения любой произвольно выбранной одно- и трехбайтной команды в программе 8.1. 3. Временну́ю диаграмму выполнения микро-ЭВМ команды MOV A, M в программе 8.2. 4. Видеоизмененную программу 8.2, записанную в машинных кодах, позволяющую первоначально записывать исследуемое число по адресу 0B00. 5. Разработанные в процессе домашней подготовки программы 8.3, 8.4 и 8.5, записанные в машинных кодах; результаты исследований работы программы по п. 3, 4, 5 заданий.

Задания для самопроверки

1. За сколько машинных тактов выполняется каждая команда в программах 8.1 и 8.2?
2. Укажите различия в способах адресации, используемых в микро-ЭВМ, построенной на основе МПК серии К580.
3. Укажите все возможные способы адресации, используемые при составлении программы по п. 7а задания для домашней подготовки.
4. При выполнении каких команд, приведенных в п. 4 задания для домашней подготовки, задействуются разряды регистра состояния МП БИС?
5. Сформулируйте правила выполнения МП БИС команд, приведенных в п. 4 задания для домашней подготовки.
6. Изобразите временны́е диаграммы процесса выполнения микро-ЭВМ следующих команд: LDA$<A_2>$ $<A_1>$, CMA, STA$<A_2>$ $<A_1>$, MOV M, A, MOV A, M, INX M.

Литература

Горбунов В. Л., Панфилов Д. И., Преснухин Д. Л. Микропроцессоры. Основы построения микро-ЭВМ.— М.: Высшая школа, 1984.

Прангишвили И. В. Микропроцессоры и микро-ЭВМ.— М.: Энергия, 1979.

Соучек Б. Микропроцессоры и микро-ЭВМ: Пер. с англ.— М.: Советское радио, 1979.

Лабораторная работа 8.3. Ввод—вывод, маскирование данных и организация условных переходов

Цель работы: *исследование методов подключения и организации обмена информацией с простейшими устройствами ввода — вывода. Изучение программных способов маскирования данных и организации условных переходов в микро-ЭВМ.*

К командам ввода — вывода МП БИС КР580ИК80 относятся команды IN<A_1> и OUT<A_1>. При выполнении команды IN<A_1> микро-ЭВМ считывает число из входного устройства с адресом (A_1)(A_1) и записывает его в аккумулятор. При выполнении команды OUT<A_1> МП БИС записывает число из аккумулятора в выходное устройство с адресом (A_1)(A_1). Так как адрес устройства указывается в одном байте, то с помощью этих команд микро-ЭВМ может обмениваться информацией не более чем с 256 внешними устройствами.

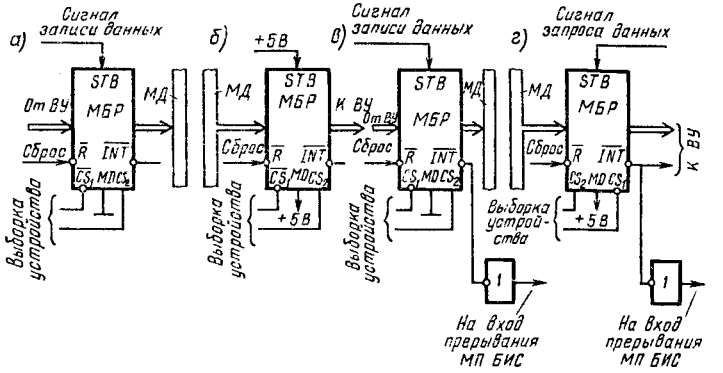


Рис. 8.5. Использование схемы многорежимного буферного регистра (МБР) К589ИР12 в качестве устройства ввода—вывода при различных способах обмена информацией

В качестве простейших устройств ввода — вывода могут использоваться 8-разрядные регистры (например, многорежимный буферный регистр (МБР) К589ИР12). Обмен данными между микро-ЭВМ и внешними устройствами может вызываться как в определенных местах в программе, так и по сигналам прерывания. В последнем случае подпрограмма обмена данными с внешним устройством будет вызываться за счет перевода микро-ЭВМ в режим обслуживания прерывания. Схемы подключения МБР К589ИР12 при использовании его в качестве устройства ввода — вывода и организации обмена информацией как по сигналам, формируемым микро-ЭВМ, так и по сигналам запросов прерывания приведены на рис. 8.5, *а—г*.

На рис. 8.5, *а*, *б* показаны схемы подключения МБР при работе его в качестве устройства ввода — вывода и осуществлении обмена информацией с ним по сигналам, формируемым микро-ЭВМ.

На рис. 8.5, *в*, *г* приведены схемы подключения МБР к микро-ЭВМ для обмена информацией по сигналам прерывания. В этом случае внешнее устройство записывает данные во входное устройство (рис. 8.5, *в*) по сигналу, подаваемому на вход STB многорежимного буферного регистра. Этим же сигналом формируется сигнал прерывания на выходе $\overline{\text{INT}}$ схемы, подаваемый на вход прерывания МП БИС. Вызванная подпрограмма обслуживания прерывания заставляет микро-ЭВМ обратиться ко входному устройству для получения данных. Схема подключения МБР и микро-ЭВМ для вывода данных во внешнее устройство приведена на рис. 8.5, *г*. При поступлении сигнала запроса от внешнего устройства (ВУ) на вход STB многорежимный буферный регистр вырабатывает сигнал низкого уровня на выходе $\overline{\text{INT}}$, который может подаваться на вход прерывания МП БИС. Подпрограмма обслуживания этого прерывания записывает данные в МБР по сигналам выборки. Этими же сигналами сбрасывается внутренний триггер запроса прерывания МБР, что приводит к появлению сигнала единичного уровня на выходе $\overline{\text{INT}}$, который сообщает внешнему устройству о приеме данных от микро-ЭВМ в МБР.

В качестве устройств ввода — вывода могут применяться и более сложные схемы, например программируемое устройство ввода — вывода информации в параллельном коде (КР580ИК55). Схема подключения к микро-ЭВМ входного устройства, выполненного на базе МБР К589ИР12 (*D1*) с переключателями, приведена на рис. 8.6, *а*. При замкнутом переключателе на вход регистра подается «0», а при разомкнутом — «1». Переключатели используются для имитации передачи данных от внешнего устройства. К регистру можно подключить светодиоды (HL_1—HL_8) для индикации чисел, записанных в нем. На рис. 8.6, *б* приведена схема подключения выходного устройства микро-ЭВМ, построенная на базе схемы К589ИР12 (*D1*). Светодиоды HL_1—HL_8 указывают число, записанное в выходном устройстве.

Простейшая программа (программа 8.6) перезаписи числа со входного устройства (с адресом 20) в выходное устройство (с адресом 30) имеет следующий вид:

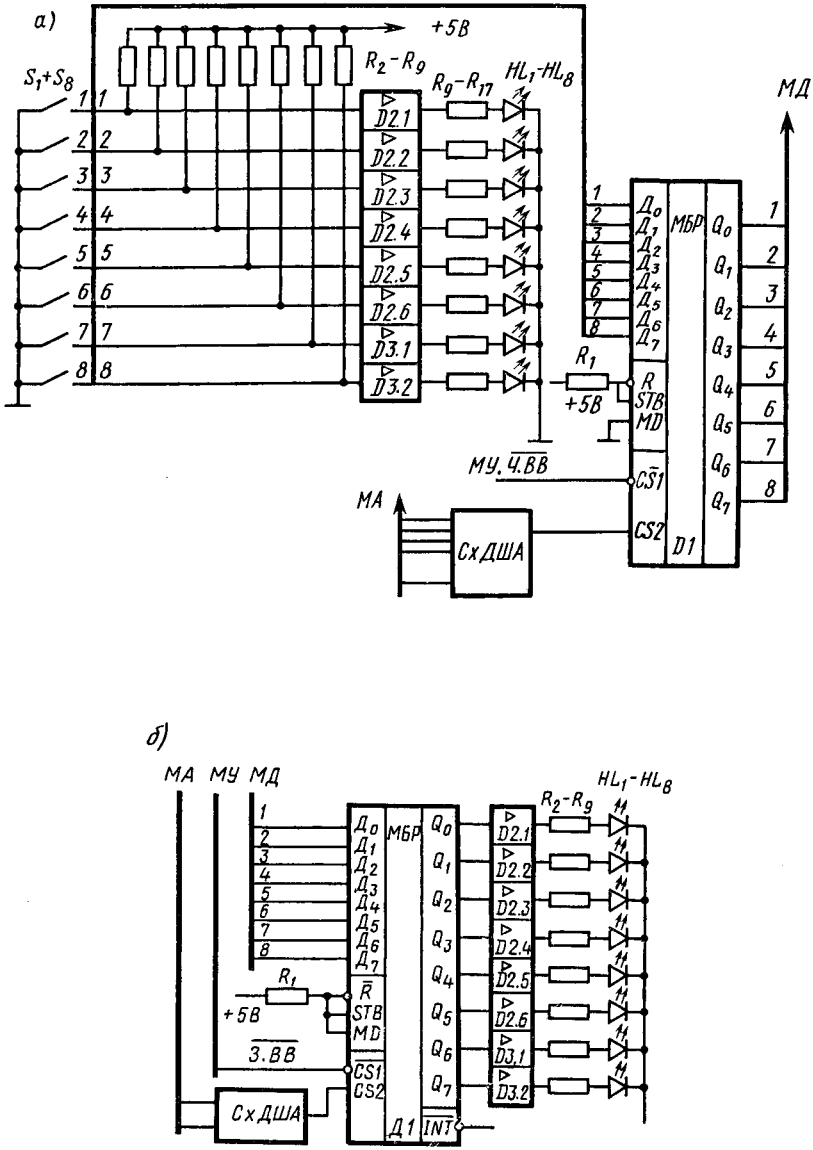


Рис. 8.6. Схемы подключения входного (*а*) и выходного (*б*) устройств к микро-ЭВМ

Программа 8.6

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	DB 20	CNT	IN 20	записать число из входного устройства с адресом 20 в аккумулятор
0802	D3 30		OUT 30	записать число из аккумулятора в выходное устройство с адресом 30
0804	C3 0008		JMP CNT	идти на CNT

Организация условных переходов в микро-ЭВМ осуществляется с помощью регистра признаков МП БИС.

Регистр признаков имеет пять разрядов, каждый из которых устанавливается по определенному правилу в соответствии с выполнением МП БИС последней команды. Этими разрядами являются:

1. Разряд переполнения C—CARRY. В него записывается 1, если при выполнении арифметической команды или команды сдвига было переполнение аккумулятора, в противном случае в разряд записывается 0.

2. Разряд знака S—SIGN. В него записывается 1, если при выполнении арифметической или логической команды в старшем, седьмом, разряде аккумулятора записана 1, в противном случае в разряд записывается 0.

3. Разряд нулевого результата Z—ZERO. В него записывается 1, если при выполнении арифметической или логической команды во всех разрядах числа в аккумуляторе имеются 0, в противном случае в разряд записывается 0.

4. Дополнительный разряд переполнения AC—AUX. CARRY. В него записывается 1, если при выполнении команд в аккумуляторе возникает единица переноса из третьего разряда числа.

5. Разряд четности P—PARITY. В него записывается 1, если при выполнении команды количество единиц в разрядах аккумулятора будет четным.

Во многих случаях при выполнении программ необходимо проверять или изменять (маскировать) состояние одного или нескольких разрядов числа в аккумуляторе. Это можно осуществить с помощью следующих операций:

1) логического умножения числа в аккумуляторе и маски, которое очищает разряд числа, если в соответствующем разряде маски будет записан 0, и не изменяет его, если в разряде маски записана 1;

2) логического сложения числа в аккумуляторе и маски, которое устанавливает разряд числа в 1, если в таком

же разряде маски будет записана 1, и не изменяет его, если в этом разряде записан 0;

3) логического «исключающего ИЛИ» числа в аккумуляторе и маски, которое инвертирует содержание разряда числа, если в соответствующем разряде маски записана 1, и не изменяет его, если в этом разряде записан 0. Примеры использования этих команд приведены в табл. 8.2.

Таблица 8.2

Мнемокод	Машинный код	Число в аккумуляторе	Маска	Комментарий	Результат в аккумуляторе
ANI <D_1>	E6 <D_1>	00111010 11111111 00000000 10101010 11110000 00001111 00100010	10101100 00100010 00100010 00100010 11111111 11111111 00000000	Логическое умножение содержимого аккумулятора с байтом D_1	00101000 00100010 00000000 00100010 11110000 00001111 00000000
ORI <D_1>	F6 <D_1>	00111010 00001111 11110000	10101100 00001111 00001111	Логическое сложение содержимого аккумулятора с байтом D_1	10111110 00001111 11111111
XRI <D_1>	EF <D_1>	00111010 00001111 11110000	10101100 00001111 00001111	Логическое «исключающее ИЛИ» содержимого аккумулятора с байтом D_1	10010110 00000000 11111111

Проведение логических операций возможно также с содержимым аккумулятора и внутренними регистрами МП БИС. В этом случае команды — однобайтные. При выпол-

нении всех логических команд задействуются разряды Z, S, P, AC регистра признаков (в разряд C записывается 0). Это позволяет проверять состояние любого разряда числа и выполнять условные переходы в программах. Программа маскирования отдельных разрядов числа (программа 8.7), записанного во входном устройстве, приведена ниже. Программа помещает результат маскирования в выходное устройство.

Программа 8.7

Адрес	Код	Метка	Мнемокод	Комментарий
0800	DB 20	CNT	IN20	получить число из входного устройства
0802	E6 20		ANI 20	выполнить логическую операцию
0804	D3 30		OUT 30	записать результат в выходное устройство
0807	C3 0008		JMP CNT	продолжать

Условные переходы организуются в программах с помощью команд условных переходов. При выполнении этих команд МП БИС проверяет состояние соответствующего разряда регистра состояния. Если при проверке состояния разряда регистра состояния условие не подтверждается, то выполняется следующая по порядку команда программы. Все команды условных переходов — трехбайтные: первый байт содержит код команды, второй и третий байты — адрес передачи управления. Таким образом, команды условных переходов позволяют строить ветвящиеся алгоритмы и в зависимости от текущего значения результата выполнения программы переходить на различные участки программы.

Ниже приведена программа (программа 8.8) для определения 1 в пятом разряде числа, записанного во входном устройстве. Программа использует маскирование числа и условный переход.

Программа 8.8

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	DB 20	WAIT	IN 20	получить число из входного устройства
0802	E6 20		ANI 20	проверить состояние пятого разряда числа
0804	CA 0008		JZ WAIT	идти на WAIT, если в пятом разряде был 0 (Z = 0)
0807	CF		RST 1	окончить выполнение программы

В представленных ранее программах имел место лишь один цикл, в котором работала микро-ЭВМ. Программа ожидания появления 1 во втором и пятом разрядах числа, записанного во входном устройстве (программа 8.9), содержит два цикла.

Программа 8.9

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	DB 20	WAITI	IN 20	получить число из входного устройства
0802	E6 04		ANI 00000100	выключен ли второй переключатель?
0804	CA 0008		JZ WAIT 1	если нет, продолжить WAIT 1
0807	3E FF		MVI A, FF	если да, зажечь светодиоды
0809	D3 30		OUT 30	выходного регистра
080B	DB 20	WAIT2	IN 20	получить число из входного устройства
080D	E6 20		ANI 00100000	выключен ли пятый переключатель?
080F	CA 0B08		JZ WAIT 2	если нет, продолжать WAIT2
0812	3E 00		MVI A, 00	если да, погасить светодиоды
0814	D3 30		OUT 30	выходного регистра
0816	C3 0008		JMP WAIT 1	повторить программу

Задания для домашней подготовки

1. Ознакомьтесь со схемой многорежимного буферного регистра и схемой программируемого устройства ввода — вывода информации в параллельном коде КР580ИК55.

2. Изучите способы организации обмена информацией между микро-ЭВМ и внешними устройствами. Рассмотрите схемы подключения устройств ввода — вывода данных при различных способах обмена.

3. Ознакомьтесь с командами ввода — вывода МП БИС КР580ИК80, а также временны́ми диаграммами их выполнения.

4. Изобразите схемы подключения к микро-ЭВМ устройств ввода — вывода, используемых при проведении лабораторной работы.

5. Изучите группу логических команд и команд условной передачи управления.

6. Ознакомьтесь с разрядами регистра признаков МП БИС и правилами записи в них 1.

7. Ознакомьтесь с программами 8.6, 8.7, 8.8, 8.9.

8. Самостоятельно разработайте программы: а) включения светодиодов выходного устройства, если число, записанное во входном устройстве, больше 3; б) включения светодиодов выходного устройства, если число, записанное во входном устройстве, больше 3, но меньше 8.

9. Видоизмените программу 8.8 так, чтобы микро-ЭВМ реагировала на 0 в пятом разряде при записанных 1 во всех остальных разрядах.

Задания к лабораторной работе

Задание 1. Исследовать программу 8.6.

Порядок выполнения задания: **1.** Ввести в микро-ЭВМ программу 8.6. Осуществить пуск программы. **2.** Убедиться, что при выполнении программы микро-ЭВМ постоянно переписывает данные со входного устройства в выходное. Для этого с помощью переключателей входного устройства изменить числа, записанные в нем. Информация о числах в устройствах ввода — вывода может отображаться светодиодами.

Задание 2. Исследовать программу 8.7.

Порядок выполнения задания: **1.** Ввести в микро-ЭВМ программу 8.7. Осуществить пуск программы и исследовать результат ее выполнения по числу, записанному в выходное устройство. **2.** Заменяя в программе 8.7 двухбайтную команду ANI<D> на однобайтные ANA A, XRA A, ORA A, исследовать результат их выполнения по числу, записанному в выходном устройстве.

Задание 3. Исследовать программу 8.8.

Порядок выполнения задания: **1.** Ввести в микро-ЭВМ программу 8.8. Осуществить пуск программы и убедиться, что при ее выполнении микро-ЭВМ реагирует лишь на те числа во входном устройстве, которые содержат 1 в пятом разряде. После окончания выполнения программы (выполнения в программе команды RST1) в разряде Z регистра состояния записана 1. **2.** Исследовать видоизмененную программу 8.8, позволяющую микро-ЭВМ реагировать на 0 в пятом разряде при записанных 1 во всех остальных разрядах.

Задание 4. Исследовать программу 8.9.

Порядок выполнения программы: **1.** Ввести в микро-ЭВМ программу 8.9. Осуществить пуск программы и убедиться, что при наличии 1 лишь во втором разряде числа входного регистра светодиоды выходного регистра вклю-

чены и микро-ЭВМ работает в цикле WAIT 2 ожидания появления единицы в пятом разряде числа. **2.** Записать 0 во второй разряд входного устройства при включенных светодиодах входного устройства. Записать 1 в пятый разряд входного устройства и убедиться, что светодиоды выходного устройства выключаются и микро-ЭВМ находится при выполнении цикла WAIT 1 программы. **3.** Установить 1 одновременно во втором и пятом разрядах числа во входном устройстве и проверить, что микро-ЭВМ последовательно выполняет оба цикла (WAIT 1 WAIT 2) программы.

Задание 5. Исследовать программы, разработанные в п. 8 задания для домашней подготовки.

Примечание. Программы, разработанные при выполнении п. 8 задания для домашней подготовки, исследуйте самостоятельно.

Работа на учебной ЭВМ

В учебной микро-ЭВМ входные и выходные устройства выполнены по схемам, приведенным на рис. 8.6 и 8.7. Адрес входного устройства — 20, а выходного — 30. Таким образом, все программы, приведенные в лабораторной работе, задания и порядок их выполнения могут быть без каких-либо изменений выполнены на учебной микро-ЭВМ.

Содержание отчета

Отчет должен содержать: **1.** Схемы подключения внешних устройств к микро-ЭВМ. **2.** Временные диаграммы процесса выполнения микро-ЭВМ команд ввода — вывода данных с внешних устройств. **3.** Самостоятельно разработанные и исследованные в процессе выполнения лабораторной работы программы, указанные в п. 8 задания для домашней подготовки. **4.** Полный перечень команд передачи управления по условию для МП БИС КР580ИК80. **5.** Полный перечень команд логических операций для МП БИС КР580ИК80.

Задания для самопроверки

1. С помощью каких команд микро-ЭВМ может осуществлять ввод — вывод информации?
2. За сколько машинных тактов осуществляется ввод — вывод данных по командам IN<A_1>, OUT<A_1>?
3. Приведите схемы подключения дешифраторов адреса и внешних

устройств при организации обмена информацией с внешними устройствами с помощью различных команд.

4. Укажите достоинства и недостатки различных методов адресации к внешним устройствам.

5. При выполнении каких команд, приведенных в программе 8.7, задействуются разряды регистра состояния МП БИС?

6. По каким условиям записывается 1 в каждый из разрядов регистра состояния МП БИС?

7. Перечислите виды логических операций, выполняемые МП БИС.

8. Перечислите режимы работы программируемого устройства ввода — вывода информации в параллельном коде КР580ИК55.

9. Укажите различия в управляющих сигналах схем К589ИР12 и КР580ИК55 при использовании их в режиме ввода — вывода информации в микро-ЭВМ.

10. Приведите различные варианты подключения внешних устройств к микро-ЭВМ с помощью схем К589ИР12 и КР580ИК55.

11. Рассмотрите возможные способы организации обмена информацией между двумя микро-ЭВМ в параллельном коде с помощью схемы КР580ИК55.

Литература

Горбунов В. Л., Панфилов Д. И., Преснухин Д. Л. Микропроцессоры. Основы построения микро-ЭВМ.— М.: Высшая школа, 1984.

Балашов Е. П., Пузанков Д. В. Микропроцессоры и микропроцессорные системы: Радио и связь, 1981.

Прангишвили И. В. Микропроцессоры и микро-ЭВМ.— М.: Энергия, 1979.

Алексеенко А. Г., Галицин А.А., Иванников А. Д. Проектирование радиоэлектронной аппаратуры на микропроцессорах.— М.: Радио и связь, 1984.

Лабораторная работа 8.4. Подпрограмма и стек

Цель работы: *исследование особенностей записи и обращения к подпрограммам; изучение методов использования стека при создании программ.*

Краткие сведения из теории

Память микро-ЭВМ, построенной на основе МПК серии К580, может иметь не более 65 536 однобайтных ячеек. Учитывая ограниченные возможности памяти при разработке программ, нужно стараться сделать их как можно короче. С этой целью часть программы, которая неоднократно повторяется, или программа, которая часто используется, могут быть оформлены в виде *подпрограмм* — последовательностей команд, выполнение которых может

быть вызвано из любого места программы любое количество раз. Процесс передачи управления к подпрограмме называется ее *вызовом.* Данные и адреса, требуемые для работы подпрограммы, называются *входными параметрами.* Результаты работы подпрограммы, передаваемые по окончании ее работы в основную программу, называются *выходными параметрами.*

Для вызова подпрограммы и возврата из них используются команды CALL $<A_2>$ $<A_1>$ и RET.

Команда CALL $<A_2>$ $<A_1>$ загружает в программный счетчик МП БИС содержимое байтов $<A_2>$ $<A_1>$, записанных в последующих двух адресах памяти после адреса, где записан код команды CALL (CD). Содержимое байта $<A_2>$ записывается в младший байт *PCL* программного счетчика, а третий байт $<A_1>$ команды — в старший байт *PCH* программного счетчика, при этом МП БИС автоматически сохраняет в стеке адрес основной программы, к которому она будет обращаться после выполнения подпрограммы.

Стек — специально организованная область ОЗУ, задействованная в микро-ЭВМ для временного сохранения данных или адресов. Число, записанное в стек последним, извлекается из него первым.

Команда RET (C9) помещает в программный счетчик последнее записанное на данный момент в стеке число. После этого выполнение программы будет осуществляться с этого адреса. Любая подпрограмма должна кончаться командой RET.

Автоматическое сохранение и восстановление адреса основной программы при выполнении подпрограмм позволяет сделать подпрограммы вложенными, т. е. осуществить вызов одной подпрограммы из другой. Уровень вложенности для данной микро-ЭВМ определяется лишь размером стека.

Существуют также команды условного вызова подпрограммы и возврата из них. Они позволяют вызвать подпрограмму и возвратиться из нее по определенному состоянию заданных разрядов регистра признаков (аналогично командам условных переходов). Все команды условного вызова подпрограммы — трехбайтные, во втором и третьем байтах сообщается начальный адрес подпрограммы. Команды вызова подпрограмм и возврата из них используют стек и внутренний регистр МП БИС SP (STACK POINTER) для адресации к стеку.

Помимо команд вызова подпрограмм и возврата из них

со стеком можно обмениваться информацией с помощью команд PUSH<R> (записать в стек содержание обозначенного регистра МП БИС) и POP<R> (записать данные из стека в обозначенный регистр МП БИС). Эти команды являются однобайтными, и в них содержится указание пары регистров МП БИС.

При записи в стек содержимого пары регистров или программного счетчика по адресу SP—1 записывается содержимое старшего регистра из указанной пары или старший байт PCH программного счетчика, а по адресу SP—2 в стек записывается содержимое младшего регистра из указанной пары младшего байта PCL программного счетчика.

При записи из стека данных в пару регистров или программный счетчик в младший регистр пары или PCL записывается число из адреса, указанного в указателе стека SP, а в старший регистр пары или PCH — число, записанное по адресу SP + 1. В результате выполнения команды содержимое указателя стека SP увеличивается на 2. Данные в памяти не изменяются, а лишь происходит их чтение и увеличение содержимого SP.

Таким образом, при записи данных адреса стека растут от больших к меньшим, а указатель стека SP всегда содержит последний адрес стека, в котором записано число.

При разработке программ необходимо первоначально назначать область стека, записывая в SP адрес с помощью команды LXI SP<A_2> <A_1> или команды SPHL.

Все операции со стеком должны быть сбалансированы, т. е. каждая подпрограмма должна содержать равное количество команд PUSH<R> и POP<R> и оканчиваться командой RET. В противном случае выполнение команды RET в конце подпрограммы приведет к записи в программный счетчик случайного числа из стека. Адрес возврата в основную программу будет потерян и нарушится последовательность ее выполнения.

Как правило, в начале каждой программы сохраняют в стеке содержимое всех задействованных при ее выполнении регистров с помощью команд PYSH<R>. В конце подпрограммы восстановление содержимого регистров осуществляется с помощью команд POP<R> и в обратной последовательности по отношению к их записи в стек.

Обычно в виде подпрограмм записываются многократно используемые фрагменты программ, например подпрограмма выдачи звукового сигнала, подпрограмма обслуживания клавиатуры и дисплея и т. д.

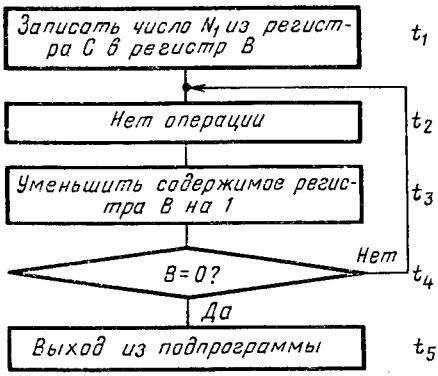


Рис. 8.7. Алгоритм простой подпрограммы временно́й задержки

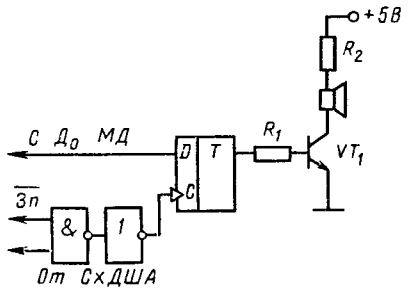


Рис. 8.8. Схема формирования звуковых сигналов в микро-ЭВМ

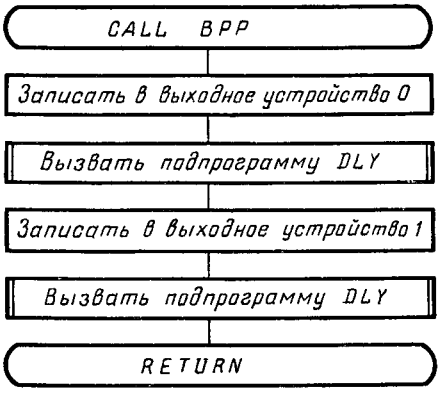


Рис. 8.9. Алгоритм подпрограммы выдачи звукового сигнала

Алгоритм работы простой подпрограммы временно́й задержки приведен на рис. 8.7. Общее время задержки вычисляется по формуле $T_D = t_1 + (t_2 + t_3 + t_4)N_1 + t_5$, где N_1 — число, первоначально записанное в счетчике. В качестве счетчика выбран регистр B, в который записывается число N_1 из регистра C.

Команда NOP нужна для увеличения времени выполнения цикла, а следовательно, и общей задержки. Вместо команды NOP может быть записана любая последовательность команд, выполнение которых не изменяет содержимого регистров микропроцессора. Время записи числа N_1 в регистр B и возврата из подпрограммы $t_1 + t_5$ фиксировано и в цикл не входит. Минимальная задержка для приведенной подпрограммы определяется при $N_1 = 0,1$ и равна $T_{D\,MIN} = t_1 + t_2 + t_3 + t_4 + t_5$. Максимальная задержка имеет место при $N_1 = 00$ и вычисляется по формуле $T_{D\,MAX} = t_1 + (t_2 + t_3 + t_4)256 + t_5$.

Подпрограмма DLY (подпрограмма 8.10) представляет подпро-

грамму временно́й задержки, записанную в соответствии с алгоритмом, представленным на рис. 8.7. Рассмотрим пример использования подпрограммы временно́й задержки при организации звуковых сигналов в микро-ЭВМ.

В микро-ЭВМ звуковые сигналы могут формироваться простейшей схемой (рис. 8.8), на вход которой со звуковой частотой записываются по очереди «0» и «1». Будем считать, что устройство формирования звуковых сигналов имеет адрес 80. Схема алгоритма работы подпрограммы генерации звуковых колебаний (подпрограмма 8.11) приведена на рис. 8.9.

Подпрограмма 8.10

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0900	41		MOV B, C	записать число из регистра C в регистр B
0901	00	DLY	NOP	нет операции
0902	05		DCR B	уменьшить число в регистре B на 1
0903	C2 0109		JNZ DLY	если число, записанное в регистре B, не равно нулю, то идти на DLY
0906	C9		RET	

Подпрограмма 8.11

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0A00	AF	BPP	XRA A	очистить аккумулятор
0A01	D3 80		OUT BP	записать 00 в выходное устройство
0A03	CD 0009		CALL DLY	вызвать подпрограмму задержки
0A06	2F		CMA	записать код ГГ в аккумулятор
0A07	D3 80		OUT BP	записать код ГГ в выходное устройство
0A09	CD 0009		CALL DLY	вызвать подпрограмму задержки
0A0C	C9		RET	

Программа MAIN (программа 8.12) представляет программу генерации сигналов с частотой, задаваемой числом с входного регистра.

Программа 8.12

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	DB 20	MAIN	IN 20	записать число из входного регистра в аккумулятор

0802	4F		MOV C, A	записать число в регистр C
0803	CD 000A		CALL BPP	вызвать подпрограмму BBP
0806	C3 0008		JMP MAIN	продолжать

При использовании подпрограмм за счет изменения входных параметров можно влиять на конечный результат выполнения подпрограмм. Рассмотрим это на примере подпрограммы регулируемой временно́й задержки (подпрограмма 8.13).

Программа состоит из двух подпрограмм: DELB (адрес 0430) и DELA (адрес 0429). Подпрограмма DELB осуществляет регулируемую временну́ю задержку, и входным параметром ее является двухбайтное число, записанное в паре регистров (B, C). Это число и определяет длительность задержки в миллисекундах. Частным случаем подпрограммы DELB является подпрограмма DELA, осуществляющая фиксированную задержку в 1 мс и не имеющая входных параметров. Таким образом, если необходимо иметь в программе фиксированную задержку в 1 мс, то можно обращаться с помощью команды CALL к подпрограмме DELA. При необходимости получения определенной заданной временно́й задержки в программе необходимо записать соответствующее число в регистры B, C, а затем вызвать подпрограмму DELB.

Подпрограмма 8.13

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0429	C5	DELA	PUSH B	сохранить содержимое регистров B и C в стеке
042A	01 0100		LXI B, 0001	установить длительность
042D	C3 3104		JMP DEL1	1 мс
0430	C5	DELB	PUSH B	сохранить содержимое регистров B и C в стеке
0431	F5	DEL1	PUSH PSW	сохранить содержимое PSW в стеке
0432	AF		XRA A	очистить аккумулятор
0433	D5		PUSH D	сохранить содержимое регистров D и E в стеке
0434	16 67	DEL 2	MVID, TIME	загрузить счетчик
0436	15	DEL 3	DCR D	1 мс задержки

			уменьшить содержимое счетчика на 1
0437	C2 3604	JNZ DEL3	если не 0, продолжать
043A	OB	DCX B	уменьшить содержимое счетчика длительности на 1
043B	B8	CMP B	
043C	C2 3404	JNZ DEL2	
043F	B9	CMP C	
0440	C2 3404	JNZ DEL2	если не 0, продолжать
0443	D1	POP D	восстановить содержимое регистров D, E
0444	F1	POP PSW	то же, PSW
0445	C1	POP B	то же, B, C
0446	C9	RET	

Задания для домашней подготовки

1. Изучите временны́е диаграммы выполнения микро-ЭВМ команд CALL <A2> <A1> и RET.

2. Ознакомьтесь с командами вызова и возврата из подпрограммы по условию для МП БИС.

3. Определите, при каких числах, записанных в регистре B, подпрограмма 8.9 будет осуществлять минимальное и максимальное время задержки. Определите эти времена, если машинный такт $T = 1$ мкс.

4. Разработайте подпрограмму 5- и 10-секундной временно́й задержки.

5. Составьте программу, последовательно включающую светодиоды выходного устройства на время соответственно 10 и 5 с. При разработке программы воспользуемся подпрограммами, разработанными в п. 4 задания (рис. 8.10).

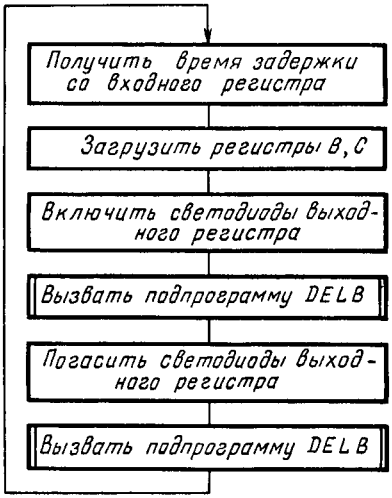


Рис. 8.10. Алгоритм программы генератора импульсов, заданной со входного регистра частоты микро-ЭВМ

6. Определите для подпрограммы 8.13, при каких числах в регистрах B и C будет максимальное и минимальное время задержки. Определите эти времена, если машинный такт $T = 1$ мкс.

7. Разберите программу 8.14, определяющую, какой из восьми переключателей входного устройства (адрес 20) микро-ЭВМ установлен в положение «0»: а) составьте алгоритм работы программы 8.14; б) запишите подпрограмму, используемую в программе 8.14; в) установите, в каком регистре МП БИС содержится информация о номере переключателя входного устройства, установленном в «0»; г) определите, как будет работать программа 8.14, если на входном устройстве будут установлены в положение «0» не один, а несколько переключателей.

Программа 8.14

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	31 B00B		LXI SP, 0 BB0	записать в указатель стека SP адрес 0900
0803	DB 20	WAITC	IN 20	получить число из входного устройства
0805	FE FF		CPI FF	содержит ли какой-либо разряд число 0
0807	CA 0308		JZ WAITC	если нет, то ждать
080A	CD 1008		CALL IDSW	если да, то вызвать подпрограмму определения разряда, в котором записан 0
080D	CF		RST 1	окончить выполнение программы
0810	06 FF	IDSW	MVI B, FF	записать в регистр B число FF
0812	04	SRCH	INR B	увеличить содержимое регистра B на 1
0813	0F		RRC	сдвинуть число в аккумуляторе вправо
0814	DA 1208		JC SRCH	если C = 1, то продолжать
0817	C9		RET	возврат из подпрограммы

Задания к лабораторной работе

Задание 1. Исследовать процесс выполнения команд вызова и возврата из подпрограммы, а также команд работы со стеком.

Порядок выполнения задания: 1. Ввести в микро-ЭВМ

подпрограмму 8.15. **2.** Выполнить программу 8.15 по командам, используя режим выполнения программы по командам. После каждой команды проверить содержимое всех регистров МП БИС. **3.** Выполнить команды CALL STDY, PUSH PSW, POP, RET по машинным циклам и построить временны́е диаграммы их выполнения микро-ЭВМ. **4.** Заменить в подпрограмме 8.15 команду POP PSW на команду NOP (00) и проследить, как будет выполняться подпрограмма 8.15. Объяснить происшедшие изменения.

Подпрограмма 8.15

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	31 B00B		LXI SP, 0BB0	записать в указатель стека SP адрес 0BB0
0803	CD 0708		CALL STDY	вызвать подпрограмму STDY
0806	CF		RST 1	остановить выполнение подпрограммы
0807	F5	STDY	PUSH PSW	записать слово состояние МП БИС в стек
0808	C5		PUSH B	записать содержимое регистров BC в стек
0809	D5		PUSH D	записать содержимое регистров D, E в стек
080A	E5		PUSH H	записать содержимое регистров H, L в стек
080B	3E 05		MVI A 05	записать в регистр A число 05
080D	47		MOV B, A	записать число из регистра A в регистр B
080E	87		ADD A	удвоить содержимое аккумулятора
080F	5F		MOV E, A	записать содержимое регистра A в регистр E
0810	67		MOV H, A	записать содержимое регистра A в регистр H
0811	E1		POP H	записать числа из стека в регистры H, L
0812	D1		POP D	записать числа из стека в регистры D, E

0813	C1	POP B	записать числа из стека в регистры B, C
0814	F1	POP PSW	записать слово состояния из стека в МП БИС
0815	C9	RET	возврат подпрограммы

Задание 2. Исследовать программу временно́й задержки на примере работы программы генерации звуковых колебаний (программа 8.12).

Порядок выполнения задания: **1.** Собрать и подключить схему выдачи звуковых сигналов к микро-ЭВМ так, как показано на рис. 8.8. **2.** Записать полный текст программы 8.12 для генерации звуковых сигналов с учетом подпрограмм 8.10 и 8.11. Ввести в микро-ЭВМ программу. **3.** Установить на входном устройстве число 00. **4.** Осуществить пуск программы с адреса 0800. **5.** Проследить за изменением тона звука, формируемого микро-ЭВМ в процессе выполнения программы, увеличивая число, записанное во входном устройстве.

Задание 3. Исследовать программу регулируемой временно́й задержки на примере программы, последовательно включающей и выключающей светодиоды выходного устройства на время соответственно 10 и 5 с.

Порядок выполнения задания: **1.** Ввести в микро-ЭВМ программу, разработанную в п. 5 задания для домашней подготовки. **2.** Осуществить пуск программы и проверить ее выполнение микро-ЭВМ. **3.** Изменить числа, записанные в регистрах B, C, в подпрограмме регулируемой временно́й задержки (подпрограмма 8.13). Проверить возможность изменения длительности задержки.

Задание 4. Исследовать программу 8.14.

Порядок выполнения задания: **1.** Ввести программу 8.14 в микро-ЭВМ. **2.** Установить на входном устройстве переключатели соответственно числу FF. Осуществить пуск программы. Убедиться, что микро-ЭВМ будет находиться в режиме ожидания появления 0 в любом разряде входного устройства. **3.** Установить 0 с помощью переключателей в любом из разрядов входного устройства. Проверить содержимое регистров МП БИС после окончания выполнения программы 8.14. **4.** Осуществить повторный запуск программы при наличии нулей в двух разрядах входного устройства. Какое число будет записано в регистре B МП БИС после окончания выполнения программы?

Работа на учебной ЭВМ

Учебная микро-ЭВМ содержит схему выдачи звуковых сигналов. Адрес внешнего устройства для записи данных в схему выдачи звуковых сигналов — 80.

Ряд подпрограмм записан в ПЗУ микро-ЭВМ. В частности, все записанные в ПЗУ и приведенные в описании лабораторной работы программы имеют адрес меньше 0800.

Все приведенные в лабораторной работе программы и порядок их выполнения могут быть без каких-либо изменений выполнены на учебной микро-ЭВМ.

Содержание отчета

Отчет должен содержать: 1. Временны́е диаграммы выполнения команд CALL $<A_2>$ $<A_1>$, RET. 2. Ответы на вопросы п. 4, 6 задания для домашней подготовки. 3. Полный перечень команд вызова и возврата из подпрограмм для МП БИС КР580ИК80. 4. Разработанные в процессе домашней подготовки программы 5- и 10-секундной задержки. 5. Перечень команд работы со стеком для МП БИС КР580ИК80. 6. Ответы на вопросы, поставленные в п. 7 задания для домашней подготовки.

Задания для самопроверки

1. Укажите количество машинных тактов выполнения команды CALL $<A_2>$ $<A_1>$.

2. В какой последовательности записывается и считывается из стека содержимое аккумулятора и регистра признаков МП БИС при выполнении команд PUSH PSW и POP PSW?

3. С помощью каких команд можно задать или переобозначить область памяти, отведенную под стек?

4. Укажите порядок выполнения микро-ЭВМ команды RET.

5. Сравните процесс выполнения микро-ЭВМ команд CALL и RST.

6. Как нужно изменять подпрограмму 8.13, чтобы временна́я задержка, получаемая в результате ее выполнения, определялась числами, записанными по адресам 0B00 и 0B01?

7. В какой последовательности сохраняется и извлекается содержимое регистров МП БИС в подпрограммах?

8. Какой минимальный адрес будет записан в указатель стека SP МП БИС при выполнении подпрограммы 8.15?

9. Как будет выполняться подпрограмма 8.15, если вместо команды POP B в ней будет записана команда NOP?

10. Какое максимальное время задержки может обеспечивать подпрограмма 8.10, если длительность машинного такта $T = 1$ мкс?

11. Какое максимальное и минимальное время задержки может обеспечить подпрограмма 8.13, если длительность машинного такта $T = 1$ мкс?

12. Как увеличить время задержки с использованием подпрограммы 8.13?

Литература

Балашов Е. П., Пузанков Д. В. Микропроцессоры и микропроцессорные системы.— М.: Радио и связь, 1981.

Соучек Б. Микропроцессоры и микро-ЭВМ: Пер. с англ.— М.: Советское радио, 1979.

Лабораторная работа 8.5. Выполнение арифметических операций

Цель работы: *изучение способов организации и исследование программ выполнения арифметических операций.*

Краткие сведения из теории

Из двух способов определения чисел с фиксированной и плавающей точкой первый получил наибольшее распространение при программировании микро-ЭВМ на МП БИС КР580ИК80. Это связано с отсутствием специальных команд, позволяющих работать МП БИС с числами с плавающей точкой. В свою очередь, двоичное восьмиразрядное число с фиксированной точкой можно представить как двоичное число со знаком, имеющее значение от -128_{10} до $+127_{10}$. При этом отрицательные числа представляются в дополнительном коде, а старший, седьмой, разряд числа используется как знаковый. Такое представление чисел не позволяет выполнять арифметические операции с использованием переноса при сложении и заема при вычитании.

Число с фиксированной точкой можно представлять также двоичными числами без знака, имеющими значения от 0 до 255_{10}.

Для МП БИС КР580ИК80 можно представлять такие числа в виде двоично-десятичного числа — BINARY — CODED — DECIMAL (BCD), при котором каждый байт рассматривается как два полубайта — две тетрады, каждая из которых кодирует десятичную цифру. Такое пред-

ставление позволяет закодировать в 1 байт числа от 0 до 99_{10}.

Проведение арифметических операций сложения, вычитания, умножения, деления, вычисления специальных функций рассмотрим на примерах соответствующих подпрограмм.

Подпрограмма 8.16

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	21 000B	MAIN	LXI H, 0B00	записать в регистры H, L адрес первого слагаемого
0803	06 05		MVI B, 0,5	загрузить в регистр B количество слагаемых
0805	CD 0908		CALL ADDB	вызвать подпрограмму сложения
0808	CF		RST 1	прервать выполнение программы
0809	AF	ADDB	XRA A	очистить аккумулятор
080A	4F		MOV C, A	очистить счетчик переносов
080B	86	CNT	ADD M	прибавить к содержимому аккумулятора число из массива слагаемых
080C	D2 1008		JNC TRM	если переноса нет, то идти на TRM
080F	0C		INR C	увеличить содержание регистра C на 1
0810	23	TRM	INX H	указать на следующий адрес слагаемого
0811	05		DCR B	уменьшить содержимое счетчика слагаемых
0812	D2 0B08		JNC CNT	если не все слагаемые, то идти на CNT
0815	C9		RET	

Программа сложения массива однобайтных чисел с получением двухбайтного результата — подпрограмма MAIN (подпрограмма 8.16). Слагаемые должны быть расположены в последовательных адресах памяти. Входными параметрами с подпрограммы ADDB являются адрес первого слагаемого, записанный в регистрах H, L, и число слагаемых, записанное в регистре B. Выходным параметром программы MAIN является сумма, старший байт которой записан в регистре C, а младший — в аккумуля-

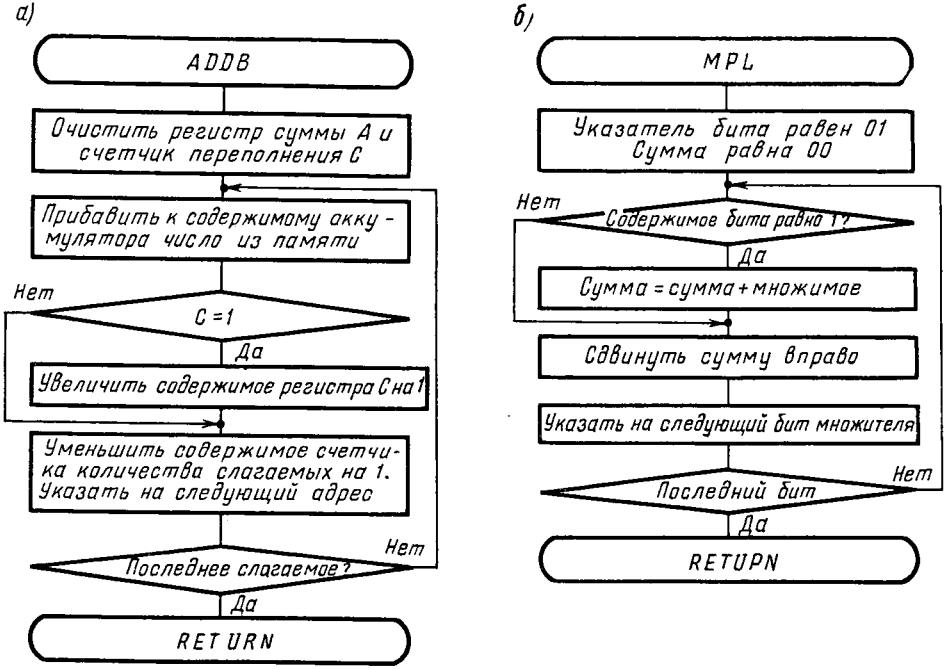


Рис. 8.11. Алгоритм подпрограммы сложения (*а*) и умножения (*б*)

торе А. Алгоритм программы сложения состоит в том, что после прибавления каждого элемента массива определяется переполнение аккумулятора (разряд С-1), и если это имеет место, то содержимое регистра С увеличивается на 1. Таким образом, за счет суммирования единиц переноса получается старший байт суммы (рис. 8.11, *а*).

Микро-ЭВМ может проводить арифметические операции с числами с двойной или большей длиной машинного слова. Так как МП БИС имеет 8-разрядное АЛУ, то операции с такими числами должны проводиться по байтам начиная с младших байтов. Так, операция сложения чисел 17F5+3411 будет проводиться в следующем виде:

Старший байт	Флаг С	Младший байт	Числа
00101111		11110101	17F5
+		+	+
00110100		00010001	3411
+1	←1		
01001100		00000110	4C06

Операция вычитания чисел 6F5C—13C5 будет осуществляться в таком виде:

Старший байт	Флаг С	Младший байт	Числа
01101111		01011100	6F5C
—		—	—
00010011		11000101	13C5
—1	←1		
01011011		10010111	5B97

Из приведенных примеров видно, что при суммировании (вычитании) младших байтов чисел необходимо применять команду ADD (SUB), а для сложения (вычитания) остальных — команду ADC (SBB), которая будет учитывать состояние разряда регистра С признаков МП БИС.

Программа нахождения разности чисел, имеющих одинаковую длину,— программа 8.17. Входные параметры: регистр С — длина чисел (в байтах), регистры H, L — адрес младшего байта вычитаемого, регистры D, E — адрес младшего байта уменьшаемого. Каждое из чисел записывается в последовательных адресах памяти начиная с младших байтов. Результат заносится в область памяти, отведенную под вычитаемое. В том случае, если уменьшаемое меньше вычитаемого, будет подаваться звуковой сигнал.

Программа 8.17

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0900	AF	SBN	XRA A	очистить аккумулятор и флаг C
0901	1A	CNT	LDAX D	записать в аккумулятор уменьшаемое
0902	9E		SBB M	вычесть из содержимого аккумулятора вычитаемое
0903	77		MOV M, A	записать разность на место вычитаемого
0904	23		INX H	указать на следующий байт вычитаемого
0905	13		INX D	указать на следующий байт уменьшаемого
0906	0D		DCR C	уменьшить содержимое счетчика длины числа
0907	C2 0109		JNZ CNT	если не последний (старший байт числа), то идти на CNT
090A	D0		RNC	если байт старший и результат без заема (C = 0), то возврат
090B	CD 1200		CALL BEEP	если был заем (C = =1), подать сигнал
090E	CF		RST 1	прервать выполнение программы

Выполнение команд MOV M, A; INX D не воздействует на разряд *C*. В программе SBN подпрограмма звукового сигнала (BEEP) начинается с адреса 0012. При исследовании программы SBN необходимо перед началом выполнения ее осуществить начальную установку всех входных параметров.

Умножение чисел. Существует несколько алгоритмов умножения чисел. При первом алгоритме умножение можно заменить многократным сложением, например $14 \cdot 3 = 14 + 14 + 14$. Существенный недостаток этого способа — значительная длительность процесса вычисления. При втором алгоритме умножение осуществляется в столбец. Этот алгоритм применим и для умножения двоичных чисел, например

$$
\begin{array}{r}
0110 = 6_{10} \\
0011 = 3_{10} \\
\hline
0110 \\
0110 \\
0000 \\
0000 \\
\hline
00010010 = 18_{10}
\end{array}
$$

При вычислении результата по второму алгоритму необходимо осуществить многократное суммирование со сдвигом влево множимого при одновременной проверке содержимого разрядов множителя начиная со стороны его младшего разряда. При этом если в очередном разряде множителя записана 1, то множимое прибавляется к сумме и сдвигается влево на один разряд, а если в разряде записан 0, то произойдет только сдвиг множимого. Сдвиг множимого влево можно заменить сдвигом суммы вправо. По этому алгоритму (рис. 8.11, *б*) работает подпрограмма умножения двух однобайтных чисел с получением двухбайтного результата (подпрограмма 8.18). Начальный адрес подпрограммы — 04E1; входные параметры: регистр D — множимое, регистр *E* — множитель. Результат перемножения записывается в регистры B, C.

Подпрограмма 8.18

Адрес	Машинный код	Метка	Мнемокод	Комментарий
04E1	010000	MPL	LXI B, 0000	очистить содержимое регистров B, C
04E4	3E 01		MVI A, 01	загрузить в аккумулятор указатель разряда
04E6	A7		ANA A	очистить флаг C
04E7	F5	MPL 1	PUS H PSW	сохранить указатель разряда в стеке
04E8	A3		ANA E	проверить содержимое очередного разряда множителя
04E9	78		MOV A, B	загрузить в аккумулятор старший байт суммы
04EA	CA EC04		JZ MPL2	если в очередном разряде записан 0, идти на MPL 2
04ED	82		ADD D	прибавить множимое к сумме
04EE	IF	MPL 2	RAR	сдвинуть сумму вправо (младший бит → C)
04EF	47		MOV B, A	сохранить содержимое аккумулятора в регистре B
04F0	79		MOV A, C	загрузить в аккумулятор младший байт суммы
04F1	1F		RAR	сдвинуть число в аккумуляторе вправо (C → старший бит)
04F2	47		MOV C, A	сохранить содержимое аккумулятора в регистре C
04F3	F1		POP PSW	получить из стека указатель разряда

04F4	17	RAL	указатель на следующий разряд
04F5	D2 E704	JNC MPL1	если разряд не последний, продолжать на MPL 1
04F8	C9	RET	если разряд последний, возврат

Деление чисел. Деление двоичных чисел, как и чисел, представленных в любой другой системе счисления, основывается на последовательном вычитании делителя из

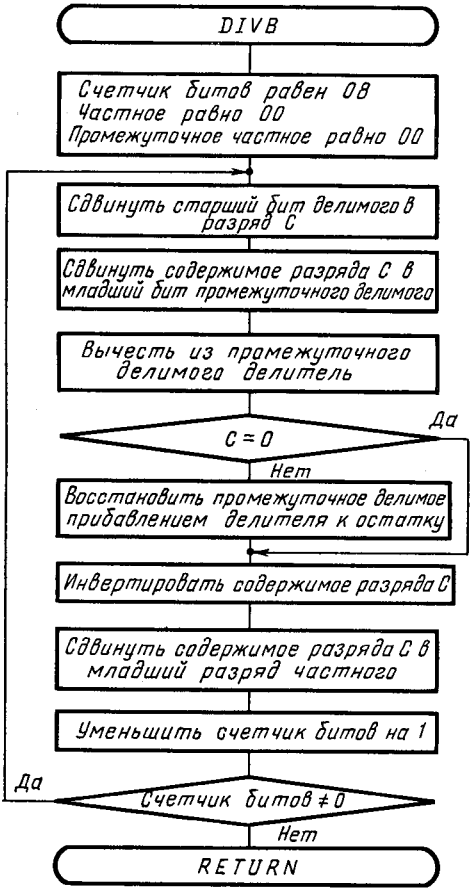


Рис. 8.12. Схема алгоритма подпрограммы деления двух восьмиразрядных чисел

делимого и остатков от деления. Однако двоичное деление реализуется проще, так как использование только двух цифр (0 и 1) исключает в каждом цикле деления необходимость определения числа делителей, содержащихся в текущем значении делимого или остатка (достаточно только сравнить их).

Схема алгоритма деления двоичных чисел приведена на рис. 8.12. Программа DIVB построена по этому алгоритму (программа 8.19). Входными параметрами этой программы являются делимое (в регистре E) и делитель (в регистре D); выходными параметрами — частное (в регистре H) и останов (в регистре C).

Программа 8.19

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0900	21 0800	DIVB	LXIH, 0008	загрузить счетчик битов (регистр L) и очистить регистр частного H
0903	0E 00		MVI C, 00	очистить регистр промежуточного делимого
0905	7B	MXT B	MOV A, E	загрузить делимое в аккумулятор
0906	17		RAL	сдвинуть старший бит в разряд C
0907	5F		MOV E, A	возвратить делимое в регистр E
0908	79		MOV A, C	загрузить в аккумулятор промежуточное делимое из регистра C
0909	17		RAL	сдвинуть разряд C в младший бит
090A	92		SUB D	вычесть из содержимого аккумулятора делитель
090B	D2 0F 09		JNC NOADD	если C=1, восстановить содержимое аккумулятора
090E	B2		ADD D	
090F	4F	NOADD	MOV C, A	возвратить промежуточное данное в регистр C

0910	3F	CMC	инвертировать разряд C
0911	7C	MOV A, H	сдвинуть разряд C в младший бит регистра частного H
0912	17	RAL	
0913	67	MOV H, A	
0914	2D	DCRL	проверены ли все восемь разрядов?
0915	C2 0509	JNZ MXTB	если нет, продолжать
0918	CF	RST1	если да, прервать выполнение программы

Вычисление специальных функций. Для вычисления специальных функций ($\sin x$, $\cos x$, $\operatorname{tg} x$, $\ln x$, $\sqrt{x}$) применяются специальные алгоритмы. Функции $\sin x$, $\cos x$, $\ln x$ можно вычислить, воспользовавшись их разложением в ряд:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$ для любого x (рад);

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$$ для любого x (рад);

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$$ для $0 < x \leqslant 1$.

Число членов ряда определяется из условия получения требуемой точности.

Для вычисления функции $\sqrt{x}$ с точностью до целых чисел можно применить алгоритм, основанный на том, что квадрат числа можно определить сложением последовательности нечетных чисел:

Число	Сумма нечетных чисел		Результат
1	1	=	1
2	1+3	=	2^2
3	1+3+5	=	3^2
4	1+3+5+7	=	4^2
5	1+3+5+7+9	=	5^2

Исходя из приведенного примера видно, что какое число необходимо возвести в квадрат, такое же количество

последовательных нечетных чисел начиная с 1 необходимо сложить.

Вычисление специальных функций по приведенным выражениям занимает длительное время и обеспечивает низкую точность. Это обусловлено сравнительно небольшой длиной машинного слова и ограниченным быстродействием МП БИС. Поэтому в тех случаях, когда ставятся жесткие требования по быстродействию и точности, применяется вычисление функций с помощью таблиц. Проиллюстрируем этот метод на примере программы вычисления квадрата числа x (программа 8.20).

Программа 8.20

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0900	2600	SQ:	MVI H, 0	очистить регистр H
0902	11 000A		LXI D SQTB	загрузить начальный адрес таблицы
0905	19		DAD D	получить адрес элемента
0906	66		MOV H, M	загрузить его в регистр H
0907	CF		RST 1	прервать выполнение программы
0A00	00	SQTB	00	таблица квадратов чисел
0A01	01		01	
0A02	04		04	
0A03	09		09	
0A04	10		10	
0A05	19		19	
0A06	24		24	
0A07	31		31	
0A08	40		40	
0A09	81		81	
0A0A	64		64	

Программа SQ определяет квадрат чисел от 0 до 10 включительно. Входной параметр программы — число x. Оно записывается в регистр L. Выходной параметр — значение x^2 в регистре H.

Задания для домашней подготовки

1. Изучите группу арифметических команд МП БИС КР580ИК80.

2. Ознакомьтесь с правилами выполнения команды DAA — десятичной коррекции аккумулятора.

3. Изучите программы 8.16—8.20, приведенные выше. Рассмотрите результат выполнения каждой программы на конкретных числовых примерах.

4. Проведите оценку времени выполнения подпрограмм умножения и программы деления 8-разрядных чисел.

5. Составьте программу для исследования результата перемножения двух чисел на основе подпрограммы 8.18.

6. Разработайте программу сложения двух 8-разрядных чисел с получением результата в двоично-десятичном коде (результат должен записываться в выходное устройство).

7. Оцените время выполнения подпрограммы 8.20.

Задания к лабораторной работе

Задание 1. Исследовать программу сложения однобайтных чисел с получением двухбайтного результата.

Порядок выполнения задания: 1. Ввести в микро-ЭВМ программу 8.16. 2. Записать в микро-ЭВМ последовательность из пяти чисел начиная с адреса 0B00. 3. Осуществить пуск программы и проверить ее выполнение по данным, записанным в регистре C и аккумуляторе МП БИС. 4. Изменить программу 8.16 так, чтобы результат выполнения записывался по адресу 0B06 и 0B07. Проверить результат ее выполнения. 5. Изменить в программе указатель количества слагаемых в сумме и выполнить программу заново.

Задание 2. Исследовать программу вычитания двух чисел, имеющих одинаковую длину.

Порядок выполнения задания: 1. Ввести программу 8.17 в микро-ЭВМ. 2. Записать в регистры D, E и H, L соответственно начальные адреса младших байтов уменьшаемого и вычитаемого. Вычитаемое должно быть записано в области ОЗУ, где нет защиты от случайной записи во время выполнения программ (в учебной микро-ЭВМ эта область занимает адреса 0B00 — 0BB0). Записать в регистр C длину числа в байтах. Для первого случая записать в регистр C число 01 (рассматривается вычитание двух 8-разрядных чисел). 3. Записать по адресам, указанным в регистрах H, L и D, E, уменьшаемое и вычитаемое. Выбрать при этом уменьшаемое большим вычитаемого. 4. Осуществить пуск программы 8.17 и исследовать результат ее выполнения по числу, записанному по адресу, где записано вычитаемое. 5. Видоизменить программу 8.18 так, чтобы результат вычисления разности

двух 8-разрядных чисел записывался в выходное устройство с адресом 30. **6.** Изменить числа, записанные по адресам, указанным в регистрах H, L и D, E, так, чтобы уменьшаемое было меньше вычитаемого. **7.** Осуществить пуск программы и проследить, что при этом получается. **8.** Исследовать в аналогичной последовательности результат вычитания двухбайтных чисел.

Примечание. Программа 8.17 построена так, что в том случае, если уменьшаемое меньше вычитаемого, микро-ЭВМ будет подавать звуковой сигнал. Если в используемой микро-ЭВМ блока звуковой сигнализации нет, то команда CALL BEEP в программе 8.17 может быть просто исключена.

Задание 3. Исследовать программу умножения двух 8-разрядных чисел с получением 16-разрядного результата.

Порядок выполнения задания: **1.** Ввести в микро-ЭВМ программу для исследования результата перемножения двух чисел, разработанную в п. 5 задания для домашней подготовки. **2.** Осуществить пуск программы и проверить результат перемножения двух чисел по числу, записанному в регистрах B, C.

Задание 4. Исследовать программы деления двух 8-разрядных чисел.

Порядок выполнения задания: **1.** Ввести в микро-ЭВМ программу 8.19. **2.** Записать в регистры E, D соответственно делимое и делитель. **3.** Осуществить пуск программы и проверить результат деления двух чисел по содержимому регистров H, C.

Задание 5. Исследовать программу вычисления квадрата числа с помощью таблицы.

Порядок выполнения задания: **1.** Ввести в микро-ЭВМ программу 8.20. **2.** Записать в регистр L число, квадрат которого необходимо вычислить. **3.** Осуществить пуск программы и проверить результат вычисления квадрата числа по содержимому регистра H. **4.** Видоизменить программу так, чтобы результат вычисления квадрата числа записывался в выходное устройство. **5.** Ввести в программу 8.20 проверку на значение числа, квадрат которого определяется в результате выполнения программы. Если задаваемое число больше 10, то микро-ЭВМ должна указывать на это, например, выдачей звукового сигнала, включением светодиодов выходного устройства и т. д.

Задание 6. Исследовать программу сложения двух

8-разрядных чисел с получением результата в двоично-десятичном коде.

Порядок выполнения задания: **1.** Ввести в микро-ЭВМ программу, разработанную при выполнении п. 6 задания для домашней подготовки. **2.** Осуществить пуск программы и проверить результат сложения следующих чисел: 33+25; 38+25; 98+25. **3.** Заменить в разработанной программе операцию сложения на операцию вычитания двух чисел. Проверить, возможно ли осуществлять десятичную коррекцию числа аккумулятора после команды вычитания.

Работа на учебной микро-ЭВМ

В учебной микро-ЭВМ подпрограмма записана в ПЗУ по тем же адресам, что приведены при описании подпрограммы 8.18. К ней можно обращаться с помощью команды CALL$<A_2>$ $<A_1>$. В микро-ЭВМ имеется схема звуковой сигнализации, описанная в лабораторной работе 4. Начало подпрограммы звуковой сигнализации — адрес 0012. Это позволяет исследовать программу 8.17 без каких-либо ее изменений. Все приведенные в лабораторной работе программы, а также порядок их выполнения могут быть исследованы на учебной микро-ЭВМ.

Содержание отчета

Отчет должен содержать: **1.** Полный перечень арифметических команд МП БИС КР580ИК80. **2.** Программу сложения двух 8-разрядных чисел с получением результата в двоично-десятичном коде, разработанного в п. 6 задания для домашней подготовки. **3.** Программу для исследования результата перемножения двух 8-разрядных чисел, разработанную в п. 5 задания для домашней подготовки. **4.** Видоизмененную программу 8.18, исследованную в задании 2. **5.** Видоизмененную программу 8.20, исследованную в задании 5. **6.** Результаты выполнения арифметических операций по всем заданиям.

Задания для самопроверки

1. Какие команды арифметических операций может выполнять МП БИС КР580ИК80?

2. Сформулируйте правило выполнения МП БИС команды DAA.

3. После каких команд можно осуществлять десятичную коррекцию числа аккумулятора (выполнять команду DAA)?

4. Как оценить максимальное время выполнения подпрограммы умножения двух чисел (программа 8.18)?

5. Можно ли непосредственно исследовать перемножения двух чисел по подпрограмме 8.18?

6. На чем основаны алгоритмы работы подпрограмм умножения и деления чисел?

7. В чем преимущество вычисления функций, заданных в виде таблиц?

8. Представьте числа от 0 до 20 в двоично-десятичном коде.

9. Укажите возможные способы представления чисел для МП БИС КР580ИК80.

10. Оцените максимальное время выполнения программы деления двух чисел, если время машинного такта для МП БИС $T=1$ мкс.

Литература

Прангишвили И. В. Микропроцессоры и микро-ЭВМ: М.: Энергия, 1979.

Соучек Б. Микропроцессоры и микро-ЭВМ: Пер. с англ. — М.: Советское радио, 1979.

Лабораторная работа 8.6. Подключение дисплея и клавиатуры к микро-ЭВМ

Цель работы: *изучение программно-аппаратурных методов подключения дисплея и клавиатуры к микро-ЭВМ.*

Краткие сведения из теории

В качестве устройства вывода информации, удобного для восприятия, часто используется дисплей. Рассмотрим метод подключения дисплея, состоящего из шести ячеек (семисегментных индикаторов), представляющих собой восемь светодиодов с общим анодом в одном корпусе. Каждый индикатор (рис. 8.13, *а*) имеет семь светодиодов для отображения сегментов цифр, а восьмой светодиод отображает десятичную точку (рис. 8.13, *б*). Индикатор может отображать цифры от 0 до 9, а также некоторые буквы.

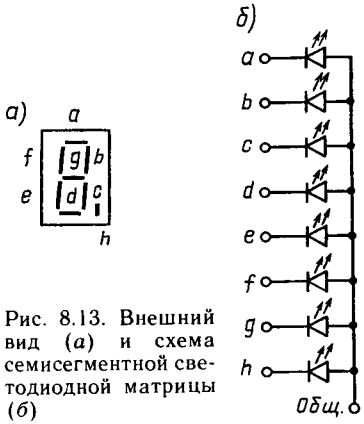


Рис. 8.13. Внешний вид (*а*) и схема семисегментной светодиодной матрицы (*б*)

Пронумеруем ячейки дисплея так, как показано на рис. 8.14.

Рис. 8.14. Нумерация ячеек дисплея

Для уменьшения схемотехнического обеспечения, необходимого для подключения дисплея к микро-ЭВМ, часто применяют мультиплексный режим работы индикаторов. При этом для вывода на дисплей информации используют два выходных регистра: *РгСг* (адрес 38_{16}) для записи семисегментного кода и *РгСк* (адрес 28_{16}) для записи номера индикатора (рис. 8.15).

Одинаковые сегменты каждой ячейки индикатора связаны общей шиной, которая соединена с одним из транзис-

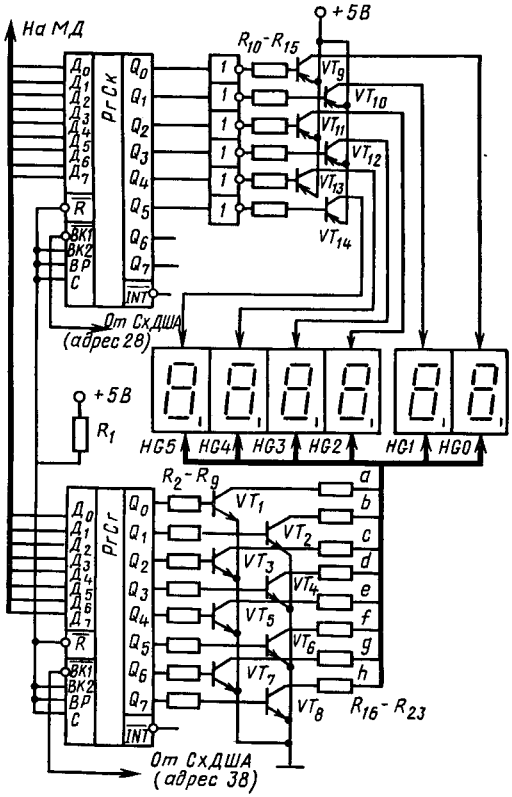


Рис. 8.15. Схема подключения дисплея к микро-ЭВМ

торных ключей VT_1 — VT_8 на выходе регистра *РгСг*. Общие аноды индикатора подключены к одному из транзисторных ключей VT_9—VT_{14} на выходе регистра сканирования *РгСк*. Включение индикатора и его сегментов при записи единицы в соответствующие разряды выглядит так:

для регистра *РгСг*

Номер разряда регистра сегментов дисплея	0	1	2	3	4	5	6	7
Включенный сегмент	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*

для регистра *РгСк*

Номер разряда регистра цифр дисплея	0	1	2	3	4	5	6	7
Включаемая цифра дисплея	0	1	2	3	4	5	—	—

Сигналы заведены на регистры *РгСг* и *РгСк* (К589ИР12) так, что при поступлении на вход *ВК1* сигнала выборки от дешифратора адреса данные с *МД*, подключенной по входам $Д_0$—$Д_7$, записываются в регистр и появляются на его входах Q_0—Q_7. Таким образом, например, при записи в регистр сегментов числа 0000 0110 отпираются транзисторные ключи VT_2 и VT_3, а при записи в регистр сканирования *РгСк* числа 0010 0000 отпирается транзисторный ключ VT_{14} и ток проходит по цепи +5 В — — VT_{14} — *HG1* — *b* и *c* — VT_2 и VT_3 — *земля*, при этом на левом индикаторе высветится 1. Если теперь с помощью программы высвечивать по очереди все ячейки, записывая их код в регистр *V2* и включая необходимую цифру с помощью регистра *РгСк*, то при достаточно высокой частоте переключений можно получать устойчивое изображение информации на дисплее.

Приведем простую программу включения сегментов пятой ячейки дисплея с помощью кода, задаваемого со входного устройства микро-ЭВМ (программа 8.21).

Программа 8.21

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	3E 20		MVI A, 20	загрузить в аккумулятор число 0010 0000
0802	D3 28		OUT SCAN	включить цифру 5
0804	DB 20	CNT	IN 20	считать данные из входного устройства
0806	D3 38		OUT DSP	записать их в регистр сегментов дисплея
0808	C3 0408		JMP CNT	продолжить

Программа 8.21 позволяет за счет изменения кода на

входном устройстве (адрес 20_{16}) включать различные сегменты пятого индикатора дисплея.

Организация мультиплексного режима работы дисплея. При мультиплексном режиме работы вывод информации на каждый индикатор дисплея выводится микро-ЭВМ последовательно. Цифра или символ на индикаторе высвечивается некоторый промежуток времени, задаваемый подпрограммой задержки. При большой частоте сканирования индикаторов на цифровом дисплее получается устойчивое изображение.

Схема алгоритма программы, обеспечивающей мультиплексный режим работы дисплея, приведена на рис. 8.16 (программа 8.22). Код цифр для вывода на каждую ячейку дисплея хранится в последовательных ячейках памяти

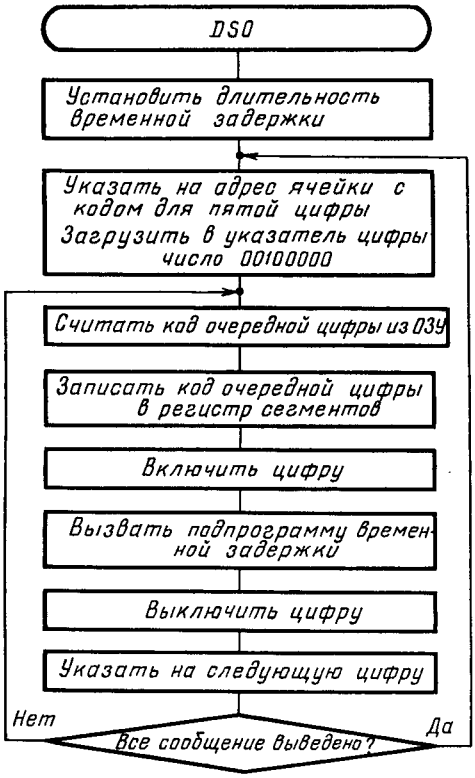


Рис. 8.16. Схема алгоритма программы мультиплексного режима индикации в микро-ЭВМ

с адресами 0900—0905. При этом полагается, что код цифры для вывода на 0 индикатора дисплея записан по адресу 0900. Начальный адрес подпрограммы временно́й задержки 0430.

Программа 8.22

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	01 0004	DSO	LXI B, 0400	загрузить в регистры B, C длительность задержки
0803	AF		XRA A	очистить аккумулятор
0804	21 0500	CNT 1	LXI H, 0905	указать на адрес кода цифры 5
0807	16 20		MVI D, 20	загрузить указатель цифры в регистр D
0809	7E	CNT 2	MOV A, M	получить из ОЗУ код очередной цифры
080A	D3 38		OUT DSP	записать его в регистр сегментов дисплея
080C	7A		MOV A, D	загрузить в аккумулятор указатель цифры
080D	D3 28		OUT SCAN	включить нужную цифру
080F	IF		RAR	указать на следующую цифру
0810	57		MOV D, A	сохранить указатель цифры в регистре D
0811	CD 3004		CALL DELB	вызвать подпрограмму временно́й задержки
0814	AF		XRA A	очистить аккумулятор
0815	D3 28		OUT SCAN	выключить цифру
0817	2D		DCRL	уменьшить на 1 содержание регистра L
0818	B2		ORA D	все ли сообщение выведено?
0819	C2 0908		JNZ CNT 2	если нет, продолжать
081C	C3 0408		JMP CNT 1	если да, то начать сначала

Подключение клавиатуры к микро-ЭВМ. Клавиатура является одним из широко распространенных устройств ввода данных и управляющих воздействий в микро-ЭВМ. С помощью клавиатуры можно вводить программу в ОЗУ, инициировать различные режимы работы микро-ЭВМ (пуск программы с заданного адреса, останов программы, выполнение программы по машинным циклам, вывод содержания регистров МП БИС на дисплей и т. д.).

Во всех случаях при организации ввода информации с клавиатуры в микро-ЭВМ перед разработчиком ставится ряд задач, к основным из которых можно отнести: 1) определение факта нажатия клавиши на клавиатуре; 2) нахождение номера нажатой клавиши; 3) осуществление передачи управления на соответствующую подпрограмму.

Первые две задачи являются специфическими при организации ввода информации с клавиатуры, и именно им будет уделено внимание в данном разделе. Последняя, как правило, решается программными методами.

Наиболее удобно организовывать клавиатуру в виде матрицы размером $n \times m$, где n и m — соответственно число строк и столбцов. При таком способе организации к микро-ЭВМ можно подключать $m \times n$ клавиш. Сопряжение клавиатуры с микро-ЭВМ производят с помощью устройств ввода — вывода данных. Для иллюстрации метода сопряжения клавиатуры с микро-ЭВМ рассмотрим клавиатуру 3×3, представленную на рис. 8.17 (где *СхДША* — схема дешифрации адреса, *УВв* — устройство

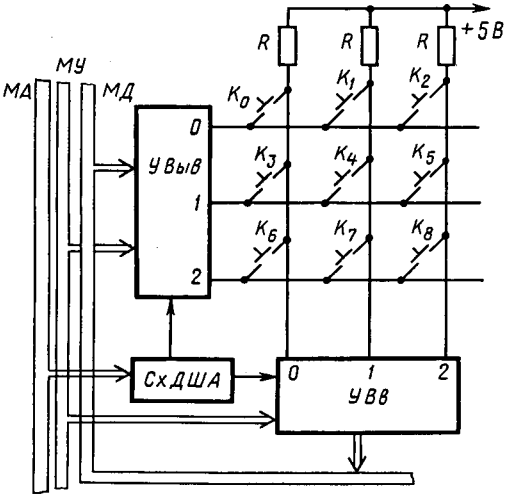


Рис. 8.17. Схема подключения клавиатуры к микро-ЭВМ

ввода, *УВыв* — устройство вывода), ряды которой подключены к трем младшим разрядам устройства вывода *УВыв* (символьное обозначение адреса KBDOT), а столбцы подключены к трем младшим разрядам устройства ввода *УВв* (символьное обозначение адреса KBDIN).

При программном способе дешифрации нажатой клавиши определение факта нажатия на клавишу может быть осуществлено с помощью такой последовательности операций: **1.** Записать нули в разряды выходного устройства. **2.** Считать содержание разрядов входного устройства.

3. Повторить снова, если во всех разрядах входного устройства записаны единицы.

Программа 8.23, написанная в соответствии с приведенным алгоритмом, позволяет определить факт нажатия на одну из клавиш, но не указывает номер нажатой клавиши.

Программа 8.23

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	3E F8	WAITK	MVI A, 11111000	записать 0 в младшие три разряда аккумулятора
0802	D3 KBDOT		OUT KBDOT	записать 0 в выходное устройство (адрес KBDOT)
0804	DB KBDIN		IN KBDIN	получить число со входного устройства (адрес KBDIN)
0806	E6 07		ANI 00000111	очистить старшие пять разрядов аккумулятора
0808	FE 07		CPI 00000111	есть ли в младших трех разрядах аккумулятора 0
080A	CA 0008		JZ WAITK	если нет, то идти на WAITK
080D	C3 0D08	DONE	JMP DONE	конец

Определить номер нажатой клавиши можно с помощью алгоритма, приведенного на рис. 8.18. Алгоритм основывается на последовательной записи нуля в каждый из рядов матрицы клавиатуры. При наличии нуля в каждом ряду микро-ЭВМ определяет факт нажатия на клавиши, находящиеся в столбцах клавиатуры, принадлежащих анализируемому ряду. Если какая-либо клавиша нажата, то определяется ее номер по номеру разряда, в котором записан ноль. Программа 8.24 реализует описанный выше алгоритм. Номер нажатой клавиши определяется по номеру ряда клавиатуры, в котором обнаружена нажатая клавиша, и номеру разряда входного устройства, в котором записан ноль.

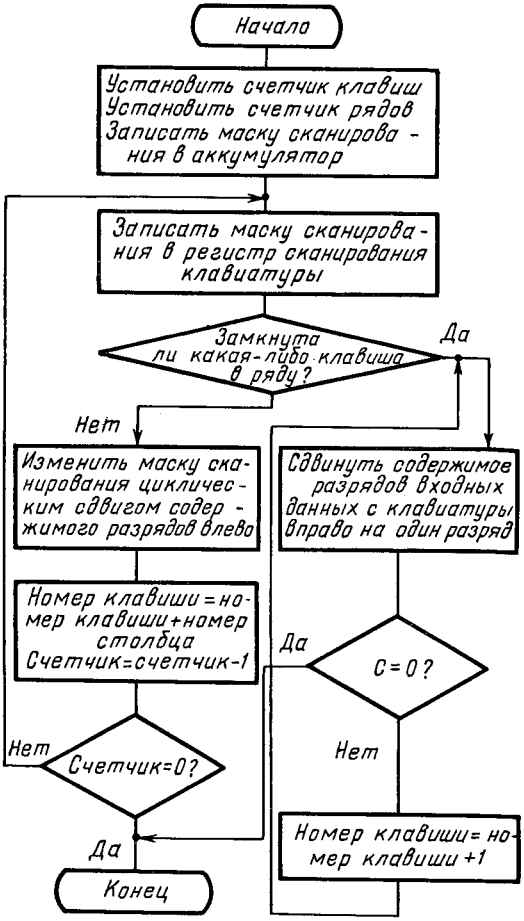


Рис. 8.18. Схема алгоритма определения нажатой клавиши при сканировании клавиатуры микро-ЭВМ

Программа 8.24

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	0600		MVI B, 00	обнулить счетчик клавиш
0802	OE FE		MVI C, 11111110	начальная установка маски сканирования рядов

```
0804   16 03          MVI D, 03        установить счетчик рядов
0806   79      FROW   MOV A, C         записать маску сканирования в аккумулятор
0807   D3 KBDOT       OUT KBDOT        записать маску сканирования в выходное устройство (адрес KBDOT)
0809   07             RLC              изменить маску сканирования
080A   4F             MOV C, A         сохранить маску в регистре C
080B   DB KBDIN       IN KBDIN         получить число со входного устройства (адрес KBDIN)
080D   E6 07          ANI 00000111     маскировать пять старших его разрядов
080F   FE 07          CPI 00000111     есть ли в трех младших разрядах 0
0811   C2 1F08        JNZ FCOL         если да, то идти на FCOL
0814   78             MOV A, B  }      изменить содержание счетчика номера клавиш
0815   C6 03          ADI 3     }
0817   47             MOV B, A  }
0818   15             DCR D            уменьшить содержимое счетчика рядов
0819   C2 0608        JNZ FROW         если не последний ряд, то повторить для следующего ряда
081C   C3 2708        JMP DONE         идти на окончание
081F   1F      FCOL   RAR       }       определение номера разряда, в котором записан 0, и определение номера нажатой клавиши
0820   D2 2708        JNC DONE  }
0823   04             INRB      }
0824   C3 1F08        JMP FCOL  }
0827   C3 2708 DONE   JMP DONE         конец
```

Часто при работе микро-ЭВМ ее операционная система строится таким образом, что информация, вводимая с клавиатуры, отображается на дисплее. С более подробными примерами построения таких программ можно ознакомиться по литературе, приведенной в конце лабораторной работы.

Задания для домашней подготовки

1. Ознакомьтесь со схемами подключения клавиатуры и дисплея к микро-ЭВМ, приведенными на рис. 8.15 и 8.17.

2. Изучите принцип мультиплексного вывода информации на дисплей.

3. Изучите принцип определения номера нажатой клавиши на клавиатуре.

4. Изучите программы 8.21—8.24.

5. Разработайте программу мультиплексного последовательного вывода лишь одной цифры, задаваемой со входного устройства микро-ЭВМ на разные ячейки дисплея. Для задания времени высвечивания цифры на каждой ячейке дисплея используйте подпрограмму временной задержки. Определите, на что будет влиять время задержки.

6. Дополните программу 8.24 так, чтобы номер нажатой клавиши отображался в одной из ячеек дисплея.

7. Разработайте программу передачи управления по заданным адресам в зависимости от нажатой клавиши на клавиатуре. Адреса передачи управления в зависимости от номера клавиши на клавиатуре задаются таблицей. Адрес передачи управления записывается в регистры H, L.

Задания к лабораторной работе

Задание 1. Исследовать программу 8.21.

Порядок выполнения задания: 1. Ввести в микро-ЭВМ программу 8.21. 2. Осуществить пуск программы. Проследить изменения во включаемых сегментах пятой цифры дисплея, меняя число на входном устройстве. Заполнить таблицу (табл. 8.3) соответствия кодов числа, записываемого в регистр сегментов дисплея, включаемым сегментам. 3. Записать и проверить коды букв Н, А, Ч, П, О.

Задание 2. Исследование программы вывода информации на дисплей.

Порядок выполнения задания: 1. Ввести в микро-ЭВМ программу 8.22. 2. Записать по адресам 0900—0905 коды букв для вывода на дисплей сообщения НАЧАЛО. 3. Осуществить пуск программы и проверить правильность вывода сообщения на дисплей. 4. Ввести в микро-ЭВМ разра-

Таблица 8.3

Символ	Код	Относительный адрес	Символ	Код	Относительный адрес	Символ	Код	Относительный адрес
0.0	3F	00	A	77	0A	P	73	14
1	06	01	B (малое)	7C	0B	П, Л (малое)	54	15
2	5B	02	C	39	0C			
3.3	4F	03	D (малое)	5E	0D	O (малое)	5C	16
4	66	04	E	79	0E	(нижняя	08	17
5.S	6D	05	F	71	0F	черта)		
6.Б	7D	06	ПРОБЕЛ	00	10	П, Л	37	18
7	07	07	H	76	11	(средняя	40	19
8	7F	08	L	38	12	черта)		
9	6F	09	Y	6E	13	Все сегменты	FF	1A
						R (малое)	50	1B
						I (левая)	30	1C

ботанную в п. 5 задания для домашней подготовки программу. **5.** Осуществить пуск программы и проверить, что на дисплей выводится лишь одна цифра. **6.** Установить в программе время включенного состояния цифры на каждой ячейке дисплея, равное 1 с (время задается подпрограммой задержки; проследить изменения в информации, выводимой на дисплей).

Задание 3. Исследовать программу обслуживания клавиатуры.

Порядок выполнения задания: **1.** Подключить к микро-ЭВМ клавиатуру 3×3 так, как показано на рис. 8.17. **2.** Ввести в микро-ЭВМ программу 8.24. **3.** Осуществить пуск программы и проверить содержимое регистра B после каждого нажатия на клавишу. **4.** Ввести в микро-ЭВМ программу, разработанную в п. 7 задания для домашней подготовки. **5.** Осуществить пуск программы и проверить соответствие адреса, записанного в регистры H, L, номеру нажатой клавиши.

Работа на учебной микро-ЭВМ

В учебной микро-ЭВМ дисплей и клавиатура подключены к магистралям так, как показано на рис. 8.15 и 8.19. Регистр сканирования *РгСк* (адрес *28*) используется как для сканирования дисплея, так и для сканирования клавиату-

ры. Адрес входного устройства чтения клавиатуры *РгЧК* — 18 (KBDIN). Программы 8.21, 8.22 могут быть выполнены без каких-либо изменений на учебной микро-ЭВМ без подключения дополнительного дисплея.

В ПЗУ микро-ЭВМ записаны программы, позволяющие определить код нажатой клавиши и выводить на дисплей сообщения. Так, подпрограмма KPU (адрес 0185) определяет, нажата ли клавиша на клавиатуре, и при любой

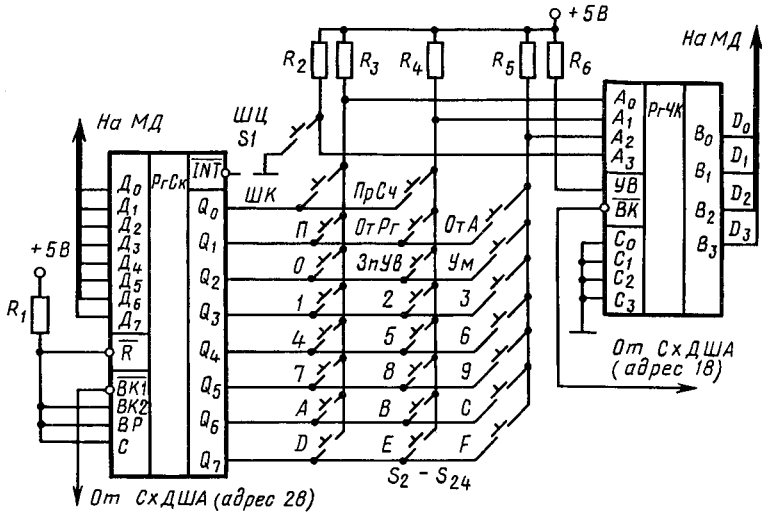


Рис. 8.19. Схема подключения клавиатуры ввода данных и управления к микро-ЭВМ

нажатой клавише устанавливает разряд признаков МП БИС Z в 0, в противном случае Z = 1. Входных параметров подпрограмма не имеет. Подпрограмма KIND (адрес 014В) производит сканирование клавиатуры, определяет факт нажатия клавиши, ее код по таблице и возвращает его. Одновременно подпрограмма KIND дешифрирует выдаваемое сообщение в семисегментный код по таблице и выводит его на дисплей. Входным параметром этой подпрограммы является сообщение, выводимое на дисплей, размещенное в ОЗУ по адресам 0BF0—0BF5, выходным параметром — код нажатой клавиши в аккумуляторе.

При выводе сообщения на дисплей подпрограмма KIND использует подпрограммы SDS (адрес 01C8) сканирования дисплея и DCD (адрес 01E9) декодирования сообщений для дисплея. Входной параметр SDS — семисегментные коды выводимого сообщения, размещенные по адресам OBFA — OBFF. Они же являются выходным параметром подпрограммы DCD, а входными ее параметрами — коды выводимого сообщения, размещенные по адресам 0BFO—0BF5. Дешифрирование в семисегментный код осуществляется с помощью специальной табл. 8.3 путем прибавления относительного адреса символа к начальному адресу таблицы.

Табл. 8.3 построена так, что при дешифрировании шестнадцатеричных цифр никаких дополнительных преобразований не требуется, так как значение цифры есть ее относительный адрес. Для перезаписи сообщения в область OBF0 — OBF5 применяется подпрограмма, вызываемая командой RST3. Входными параметрами ее являются адрес первого байта сообщения, помещенный в паре регистров D, E, и сообщение, записанное с этого адреса. Подпрограмма BLNK (адрес 0297) необходима для очистки дисплея, входных и выходных параметров не имеет.

Программа 8.25 производит декодирование и вывод на дисплей сообщения пользователя, записанного по адресу 0900—0905, с использованием подпрограмм RST3, DCD, SDS. Подпрограмма SDS выводит сообщение на дисплей только один раз, т. е. для получения изображения ее нужно вызывать многократно.

Программа 8.25

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	11 0009		LXI D, 0900	загрузить адрес начала сообщения
0803	DF		RST3	переписать сообщение по адресам OBF0—OBF5
0804	CD E901		CALL DCD	вызвать подпрограмму декодирования
0807	CD C801	CNT:	CALL SDS	вывести сообщение на дисплей
080A	C3 0708		JMP CNT	повторять вывод

Программа 8.26 показывает пример использования подпрограммы KPU. При нажатии на любую клавишу

в выходной регистр записывается число с входного устройства.

Программа 8.26

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	CD 8501	CNT	CALL KPU	нажата ли. клавиша?
0803	CA 0008		JZ CNT	если нет, продолжать ожидание
0806	DB20		IN 20	если да, получить число с входного устройства
0808	D3 30		OUT 30	записать число в выходное устройство
080A	C3 0008		JMP CNT	продолжать

Программа 8.27 использует подпрограммы KIND и BLNK и выводит на дисплей цифру, соответствующую нажатой клавише, для ввода кодов чисел (при нажатии на клавишу управления на дисплей будет выведено случайное сообщение, так как их кодов нет в таблице дешифрирования сообщения для дисплея). Кроме того, код нажатой клавиши будет выводиться на выходной регистр.

Программа 8.27

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	CD 9702		CALL BLNK	очистить дисплей
0803	CD 4B001	CNT:	CALL KIND	ожидать нажатия клавиши
0806	21 F00B		LXI H, OBFO	указать на нулевую цифру дисплея
0809	77		MOV M, A	записать код клавиши по адресу, указанному в регистрах H, L
080A	D3 30		OUT 30	вывести код клавиши в выходное устройство
080C	C3 0308		JMP CNT	продолжать

Задание 4. Исследовать подпрограммы вывода сообщения на дисплей.

Порядок выполнения задания: **1.** Ввести программу 8.25. Пользуясь табл. 8.3 (заполненной при выполнении задания 1), занести в ОЗУ начиная с адреса 0900 коды сообщения ПРОБА 1. **2.** Осуществить пуск программы и убедиться, что сообщение выведено верно.

Задание 5. Исследовать подпрограммы обслуживания клавиатуры.

Порядок выполнения задания: **1.** Ввести программу 8.26. Осуществить пуск программы и убедиться, что при нажатии на любую клавишу на клавиатуре число со вход-

ного устройства записывается в выходное устройство. **2.** С помощью рис. 8.21 определите, какой код нужно записать в регистр сканирования для определения нажатой клавиши 0. Какой код при этом поступит в аккумулятор при вводе от регистра чтения клавиатуры? **3.** Написать программу, аналогичную программе 8.26, но позволяющую микро-ЭВМ реагировать только на нажатие клавиши «0», ввести и проверить правильность ее выполнения.

Задание 6. Исследовать подпрограммы чтения и дешифрирования клавиатуры.

Порядок выполнения задания: **1.** Ввести программу 8.27. Осуществить пуск программы и убедиться, что при нажатии на клавиши ввода цифр в правой позиции дисплея высвечивается их значение, а в выходное устройство записывается их двоичный код. **2.** Нажимая на клавиши управления, исследовать их коды и объяснить происхождение символов, выводимых на дисплей при нажатии на эти клавиши. **3.** Переделать, воспользовавшись подпрограммой RST3, программу 8.27 так, чтобы на остальные позиции дисплея выводилось сообщение CODE—.

Содержание отчета

Отчет должен содержать: **1.** Схему подключения исследуемой клавиатуры к микро-ЭВМ. **2.** Программы, разработанные в п. 5, 6, 7 задания для домашней подготовки. **3.** Таблицу соответствия кодов чисел, записываемых в регистр сегментов дисплея, включаемым сегментам (таблица заполняется при выполнении задания 1). **4.** Программы, разработанные при выполнении заданий 5 и 6 в разделе «Работа на учебной микро-ЭВМ».

Задания для самопроверки

1. Какие коды необходимо записать по адресам 0900—0905 для вывода на дисплей чисел 1, 2, 3, 4, 5, 6?
2. Как следует изменить программу 8.22, чтобы изображение на дисплее начало равномерно перемещаться?
3. Как следует изменить программу 8.22 для изменения направления сканирования индикаторов дисплея?
4. Какая из клавиш будет определяться нажатой с помощью программы 8.24, если на клавиатуре будут одновременно нажаты: а) клавиши 3 и 5; б) клавиши 2 и 8?
5. Видоизмените программу 8.24 так, чтобы микро-ЭВМ определяла номер нажатой клавиши в клавиатуре, организованной в виде матрицы 8×3.
6. Видоизмените программу 8.24 так, чтобы сканирование клавиатуры происходило постоянно.

Литература

Кофрон Дж. Технические средства микропроцессорных систем: Пер. с англ.— М.: Мир, 1983.

Горбунов В. Л., Панфилов Д. И. Применение микропроцессорных устройств и микро-ЭВМ.— М.: Машиностроение, 1983.

Лабораторная работа 8.7. Исследование осциллограмм сигналов в микро-ЭВМ

Цель работы: *исследование временны́х диаграмм процессов передачи информации в микро-ЭВМ.*

Краткие сведения из теории

Процессы получения, преобразования и передачи информации в микро-ЭВМ во времени тактируются синхросигналами $Ф_1$ и $Ф_2$, поступающими на входы МП БИС. Период синхросигналов $Ф_1$ и $Ф_2$ называется *машинным тактом.* При анализе временны́х соотношений при обмене информацией в микро-ЭВМ используются также понятия *машинный цикл* и *время выполнения команды.* Возможные режимы работы, а также временны́е диаграммы процессов обмена информацией в микро-ЭВМ рассмотрены при описании МП БИС.

При выполнении программы на всех магистралях микро-ЭВМ процессы, как правило, не являются периодическими, что затрудняет их исследование с помощью простых технических средств. Наиболее простым и удобным техническим средством является осциллограф. Специфика изучения временны́х диаграмм передачи информации в микро-ЭВМ заключается в необходимости получения устойчивой картины на экране осциллографа, что возможно лишь для периодических сигналов. Все эксперименты, приведенные в заданиях лабораторной работы, позволяют проводить исследования временны́х диаграмм работы как МП БИС, так и сигналов на всех магистралях микро-ЭВМ. Приведенная программа в задании 3, по существу, позволяет исследовать процесс выполнения микро-ЭВМ любой команды. Для исследования процессов необходим лишь двухлучевой осциллограф. В дальнейшем входы осциллографа будут называться как входы А и В. Получение устойчивой картины исследуемых процессов на экране осциллографа достигается рациональным выбором сигнала, синхронизирующего запуск развертки осцил-

лографа. Во всех экспериментах этот сигнал — периодический.

Задания для домашней подготовки

1. Ознакомьтесь с временными диаграммами выполнения команд JMP<A_2> <A_1>; IN<A_1>; OUT <A_1>; MOVMA; PUSH B, POP B.

2. Рассмотрите типы машинных циклов для МП БИС КР580ИК80.

3. Изучите схему записи слова состояния МП БИС. Определите, в какой момент времени слово состояния МП БИС записывается в регистр слова состояния.

4. Ознакомьтесь с требованиями, предъявляемыми к параметрам синхросигналов $Ф_1$ и $Ф_2$ МП БИС.

5. Рассмотрите состояние магистралей микро-ЭВМ при работе МП БИС в режимах ОЖИДАНИЕ, ОСТАНОВ, ЗАХВАТ.

6. Ознакомьтесь с содержанием разрядов слова состояния МП БИС при выполнении различных машинных циклов.

Задания к лабораторной работе

Задание 1. Исследовать параметры синхросигналов $Ф_1$, $Ф_2$.

Порядок выполнения задания: 1. Подать на входы осциллографа А и В синхросигналы $Ф_1$ и $Ф_2$. 2. Измерить для каждого импульса следующие параметры: длительность сигнала, длительности фронтов, расстояние между сигналами, уровни «1» и «0».

Задание 2. Исследовать временные диаграммы выполнения команды JMP<A_2> <A_1>.

Порядок выполнения задания: 1. Ввести в микро-ЭВМ простейшую программу:

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	C3 0008	HERE	JPM HERE	идти на себя

2. Осуществить пуск программы. 3. Подать на вход А осциллографа сигнал с разряда $Д_5$ регистра слова состояния МП БИС. Единичный сигнал на этом выходе будет появляться в начале каждого машинного такта извлечения кода команды из памяти (такт M_1). 4. Осуществить синхронизацию осциллографа от этого сигнала. Определить число тактов и время, необходимое для выполнения коман-

ды JMP HERE. Зарисовать осциллограмму. **5.** Подключить вход В осциллографа к выходу «Синхр» МП БИС. Определить число и длительность сигналов на этом выходе. Зарисовать осциллограмму. **6.** Подключить вход В осциллографа к выходу «Прием» МП БИС. Определить число и длительность сигналов на этом выходе МП БИС при выполнении команды JMP HERE. Зарисовать осциллограмму. **7.** Определить с помощью осциллографа состояние разрядов регистра слова состояния МП БИС при выполнении машинного цикла получения кода команды (цикл M_1) и цикла чтения данных из памяти. Результаты занесите в таблицу. Выявите различия в определении МП БИС этих двух машинных циклов. **8.** Подключить вход *В* осциллографа к одной из подмагистралей данных микро-ЭВМ. Определить, сколько раз меняется информация на ней при выполнении команды JMP HERE. Объяснить, чем вызвано каждое изменение данных на этой подмагистрали. Зарисовать осциллограмму. **9.** Подключить вход В осциллографа к подмагистрали A_0 магистрали адреса микро-ЭВМ. Определить, сколько раз меняется информация на ней при выполнении команды JMP HERE. Зарисовать осциллограмму. **10.** Подключить вход В осциллографа к выходу $\overline{ЗП}$ МП БИС. Зарисовать осциллограмму сигнала на этом выходе при выполнении команды JMP HERE.

Задание 3. Исследовать осциллограммы процессов в микро-ЭВМ при выполнении различных команд.

Порядок выполнения задания: **1.** Ввести в микро-ЭВМ программу:

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	DB 20	STRT	IN 20	записать число с входного устройства
0802	C3 0008		JMP STRT	идти на начало

2. Осуществить пуск программы. **3.** Подключить вход А осциллографа к выходу $Д_6$ регистра слова состояния МП БИС. Задать синхронизацию осциллографа от сигнала, поступающего на вход *А*. Подключить вход В осциллографа к выходу $Д_5$ регистра слова состояния МП БИС. На этом выходе МП БИС записывает единичный сигнал в начале каждого машинного цикла M_1. Убедиться, что при выполнении программы имеется два цикла M_1. Определить длительность выполнения команды IN 20. Определить, сколько машинных тактов занимает выполне-

ние всей программы. Зарисовать осциллограмму сигнала на выходе $Д_5$ регистра слова состояния МП БИС. **4.** Определить с помощью осциллографа состояние разрядов регистра слова состояния МП БИС при выполнении машинного цикла чтения данных из внешнего устройства. Результаты занести в таблицу, составленную при выполнении п. 7 задания 2. **5.** Ввести в микро-ЭВМ программу:

Адрес	Машинный код	Метка	Мнемокод	Комментарий
0800	DB 20	STRT	IN 20	записать число с входного устройства
0802	D3 30		OUT 30	записать число в выходное устройство
0804	C3 0008		JMP STRT	идти на STRT

6. Осуществить пуск программы. Оставляя синхронизацию осциллографа по входу А от сигнала с выхода $Д_6$ регистра слова состояния МП БИС, исследовать временны́е диаграммы выполнения микро-ЭВМ команды OUT 30. Обратить внимание на момент появления и длительность сигнала на выходе $\overline{ЗП}$ МП БИС. Исследовать с помощью осциллографа состояние разрядов регистра слова состояния МП БИС при выполнении команды записи числа во внешнее устройство. Результаты занести в таблицу, составленную при выполнении п. 7 задания 2. Зарисовать осциллограммы процесса выполнения микро-ЭВМ команды OUT 30. **7.** Изменить в программе п. 5 команду записи числа в выходное устройство микро-ЭВМ на любую команду записи числа в память (например, MOV M, A; MOV M, B и т. д.). Ввести программу в микро-ЭВМ и исследовать осциллограммы процесса выполнения записи числа в память. Зарисовать осциллограммы. Исследовать с помощью осциллографа состояние разрядов регистра слова состояния МП БИС. Результаты занести в таблицу, составленную при выполнении п. 7 задания 2. **8.** Изменить в программе п. 5 команду OUT 30 на одну из команд работы со стеком (например, PUSH B, POP B и т. д.). Исследовать с помощью осциллографа временны́е диаграммы выполнения этих команд. Зарисовать осциллограммы. Исследовать с помощью осциллографа состояние разрядов регистра слова состояния МП БИС. Результаты занести в таблицу, составленную при выполнении п. 7 задания 2.

Работа на учебной микро-ЭВМ

Все задания, приведенные в лабораторной работе, могут быть выполнены на учебной микро-ЭВМ.

Содержание отчета

Отчет должен содержать: 1. Осциллограммы, снятые при выполнении пунктов заданий 1, 2, 3. 2. Таблицу содержания разрядов регистра состояния МП БИС при выполнении различных машинных циклов.

Задания для самопроверки

1. На каком такте слово состояния МП БИС записывается в регистр слова состояния МП БИС?
2. Сколько возможных вариантов слова состояния существует для МП БИС КР580ИК80?
3. Приведите схему записи слова состояния МП БИС в регистр слова состояния?
4. Какие сигналы формируются на магистрали управления микро-ЭВМ?
5. В чем отличие в слове состояния, выдаваемом МП БИС на машинных циклах чтения данных из внешнего устройства и памяти, записи данных во внешнее устройство и память?
6. Изобразите временны́е диаграммы выполнения микро-ЭВМ следующих команд: IN<A_1>, OUT<A_1>, JMP<A_2> <A_1>, MOV M, A, POSH PSW, POP PSW, HLT, NOP.
7. Укажите временны́е диаграммы для синхросигналов Φ_1 и Φ_2 МП БИС.
8. Укажите состояние магистралей микро-ЭВМ при работе МП БИС в режимах ОЖИДАНИЕ, ОСТАНОВ, ЗАХВАТ.
9. Укажите, каким сигналом, формируемым МП БИС, осуществляется в микро-ЭВМ управление двунаправленным шинным формирователем.
10. Укажите, для каких целей в микро-ЭВМ применяются режимы ОЖИДАНИЕ, ОСТАНОВ, ЗАХВАТ.

1. *Горбунов В. Л., Панфилов Д. И., Преснухин Д. Л.* Микропроцессоры. Основы построения микро-ЭВМ.— М.: Высшая школа, 1984.
2. *Соучек Б.* Микропроцессоры в микро-ЭВМ: Пер. с англ.—М.: Советское радио, 1979.

Глава 9

ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ И ПРОГРАММИРОВАНИЯ МИКРО-ЭВМ НА СЕКЦИОНИРОВАННЫХ МИКРОПРОЦЕССОРНЫХ БОЛЬШИХ ИНТЕГРАЛЬНЫХ СХЕМАХ

§ 9.1. Схемотехнические особенности микро-ЭВМ на базе секционированных микропроцессоров

Микропрограммный принцип управления. Основное отличие микропроцессорных систем, построенных на базе секционированных микропроцессов (МП), от микропроцессорных систем, построенных на базе однокристальных МП, заключается в применении микропрограммного способа управления. Рассмотрим этот способ подробно.

В однокристальных МП дешифрирование кода команды производят внутренние логические схемы, которые вырабатывают управляющие сигналы, организующие выполнение команды. Поэтому система команд однокристального МП фиксирована и задается при разработке кристалла.

В секционированных МП дешифрирование кода команды и выработка управляющих сигналов производятся отдельным устройством — микропрограммным устройством управления (УУ), которое содержит, как правило, постоянное запоминающее устройство микропрограмм (ПЗУМП); в нем каждая команда представлена в виде микропрограммы, реализующей выполнение команды [57].

Таким образом, в секционированных МП дешифрирование кода команды и выработка управляющих сигналов производятся не внутренними логическими схемами, а специальным УУ под управлением микропрограммы, хранимой в ПЗУМП.

В результате этого в МП-системе, построенной на базе секционированного МП, существует два уровня программирования: командный, на котором пользователь пишет программы, и микрокомандный, микропрограммы которого составляет разработчик системы.

В общем случае пользователь может и не знать, как реализуется та или иная команда, хотя при необходимости может изменить содержимое ПЗУМП, вводя новую команду или модифицируя уже имеющуюся.

Из вышесказанного следует, что программирование систем на базе микропрограммных секционированных МП с точки зрения пользователя имеет более широкие

возможности вследствие того, что система команд может быть дополнена или изменена в зависимости от конкретного применения.

Существуют также микропрограммные системы, в которых отсутствует командный уровень, а программирование ведется на уровне микрокоманд. Это позволяет составлять программы, обладающие наибольшей эффективностью, и получать максимальное быстродействие системы. Недостатком данного способа программирования является его сложность и трудоемкость, однако к нему зачастую приходится обращаться при построении систем, работающих в реальном времени.

Наиболее полно возможности микропрограммной системы раскрываются лишь для пользователя, знакомого

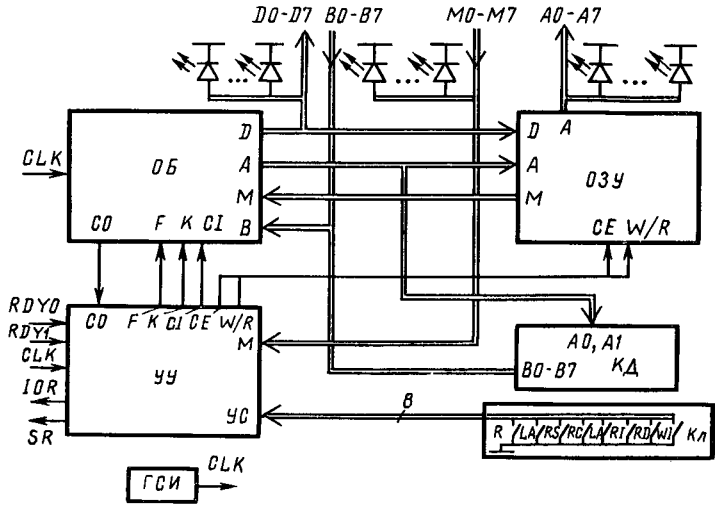


Рис. 9.1. Структурная схема микро-ЭВМ

с архитектурой и схемотехникой МП-системы, поэтому имеет смысл подробно рассмотреть архитектуру и схемотехнику микропрограммной микро-ЭВМ, построенной на базе секционированного МП-комплекта серии К589.

Структурная схема микро-ЭВМ. Схема микро-ЭВМ, реализованной в лабораторном стенде МП 589 на основе микропроцессорных БИС серии К589, представлена на рис.

9.1. Микро-ЭВМ состоит из операционного блока *ОБ*, устройства управления *УУ*, образующих в совокупности микропроцессор (МП), блока оперативного запоминающего устройства *ОЗУ*, клавиатуры *Кл*, задающей режим работы микро-ЭВМ, клавиатуры *КД* для ввода шестнадцатеричных кодов исходных данных и рабочих программ и светодиодов индикации.

Микро-ЭВМ синхронизируется импульсами CLK генератора синхронизирующих импульсов ГСИ, входящего в состав УУ.

Магистрали для обмена информацией с внешними устройствами выполнены с учетом возможностей микропроцессорных БИС, на которых реализован ОБ. В МПК серии К589 предусмотрена возможность формирования четырех магистралей: B, D, A, M. В рассматриваемой микро-ЭВМ магистраль B (B0 — B7) используется для считывания кодов шестнадцатеричной клавиатуры данных. Магистраль D (D0 — D7) используется для обмена информацией между выходом ОБ, возможными внешними устройствами (*ВУ*) и входами ОЗУ. Магистраль A (*A0 — A7*) формирует адрес ВУ, к которому обращается МП, или адрес ОЗУ. Магистраль M (M0 — M7) обеспечивает передачу информации ОЗУ в ОБ и УУ.

Светодиодные индикаторы отражают коды, передаваемые по магистралям микро-ЭВМ A, D, M.

Для программного формирования кодов шестнадцатеричной клавиатуры используются в качестве входных сигналов коды адресной магистрали A0, A1, A2, A3. Выходы клавиатуры *КД* подключены к шинам магистрали B0, B1, B2, B3. Клавиатура *Кл* режимов формирует управляющие сигналы *УС* для УУ, в соответствии с которыми микро-ЭВМ переходит в один из восьми предусмотренных режимов работы.

В микро-ЭВМ реализовано УУ с микропрограммной реализацией команд. При этом каждая команда представляется как последовательность микрокоманд. Микрокоманды выполняются с частотой поступления синхросигналов CLK. В процессе исполнения микрокоманды ОБ выполняет одну из предусмотренных его схемотехнической реализацией операций с содержимым внутренних регистров или внешних магистралей A, D, M, B в соответствии с кодом микроинструкций F, K, CI. В результате выполнения микрокоманды формируются сигналы переноса CO, ОБ, выходные коды A и D и преобразуется содержимое внутренних регистров.

Микрокоманда содержит в своем составе совокупность микроинструкций, обеспечивающих синхронную работу блоков микро-ЭВМ в соответствии с выполняемым действием. Например, микроинструкции CE, W/R обеспечивают управление схемами ОЗУ в режимах записи и считывания информации. Микроинструкция IOR используется для организации синхронного обмена информацией с ВУ.

Совокупность микрокоманд, обеспечивающих выполнение системы команд, принятой для каждой конкретной реализации микро-ЭВМ, записывается в ПЗУ микрокоманд, входящее в состав *УУ*.

В рассматриваемой микро-ЭВМ рабочая программа записывается в ОЗУ в виде последовательности команд, соответствующей алгоритму решаемой задачи. Адрес исполняемой команды фиксируется во внутренних регистрах ОБ, на которых организован программный счетчик *PC*. После выполнения очередной команды содержимое *PC* увеличивается на единицу (при последовательном исполнении команд) и по измененному содержимому *PC* из ОЗУ извлекается код очередной команды, по которому выполняется последовательность микрокоманд, соответствующих считанному коду команды.

Для работы с подпрограммами в микро-ЭВМ предусмотрена стековая память, построенная на специально отведенных для стека ячейках ОЗУ и использующая в качестве указателя стека SP внутренний регистр ОБ.

При обращении к стеку в ячейку, адрес которой определяется содержимым SP, записывается адрес команды выхода из подпрограммы и содержимое SP уменьшается на единицу. При возвращении к основной программе содержимое SP увеличивается на единицу и указывает адрес *ОЗУ*, в котором записан адрес команды выхода из подпрограммы.

Операционный блок. Представляет собой 8-разрядный блок обработки данных (рис. 9.2). Реализован ОБ на 4 БИС центрального процессорного элемента (ЦПЭ) К589ИК02 и БИС схемы ускоренного переноса (СУП) К589ИК03. Операционный блок имеет входные шины: данных от внешних устройств B; данных из памяти M; кода микрокоманды F; кода маски K и выходные шины данных D и адреса A. Кроме того, ОБ имеет вход переноса CI и выход переноса CO. Синхронизация работы ОБ осуществляется сигналом CLK.

Центральный процессорный элемент К589ИК02. Осно-

вой ОБ являются центральные процессорные элементы (ЦПЭ), представляющие собой 2-разрядные секции обработки данных. Поскольку ОБ обрабатывает 8-разрядные операнды, используется 4 ЦПЭ. В общем случае для построения N-разрядного ОБ требуется $N/2$ ЦПЭ. Таким образом, выполнение ОБ в виде массива ЦПЭ позволяет гибко варьировать разрядностью ОБ в зависимости от конкретной задачи: от простейших 4- и 8-разрядных контроллеров до мощных 16- и 32-разрядных ОБ мини-ЭВМ. В этом заключается одно из главных преимуществ

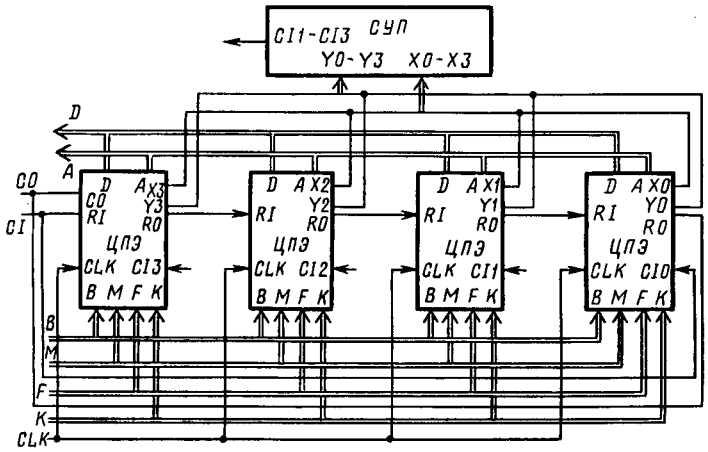


Рис. 9.2. Схема операционного блока

секционированных МП перед однокристальными МП, которые не позволяют изменять разрядность обрабатываемых слов или допускают обработку слов с удвоенной разрядностью путем их последовательной обработки, т. е. с понижением быстродействия.

Массив ЦПЭ выполняет следующие операции: двоичной арифметики, логические операции И, ИЛИ, НЕ и ИСКЛЮЧАЮЩЕЕ ИЛИ, +1, —1, сдвиг влево и вправо, проверку разрядов слова или всего слова на 0.

ЦПЭ содержит: сверхоперативное запоминающее устройство (СОЗУ), состоящее из 11 регистров общего назначения (РОН); накапливающий регистр-аккумулятор (АС); регистр адреса памяти (RA); арифметическо-логи-

ческое устройство (АЛУ); дешифратор *ДШ* микрофункций; мультиплексоры *А* и *В*.

Схема центрального процессорного элемента. Центральный процессорный элемент (рис. 9.3) выполняет арифметические, логические и регистровые функции 2-разрядного микропрограммного ОБ. Данные от внешних устройств поступают в ЦПЭ по шине В, данные из ОЗУ — по шине М. Выходные данные от ЦПЭ во внешние устройства передаются по шине D, а адрес внешнего устройства или ячейки ОЗУ определяется шиной А. Внутри ЦПЭ данные хранятся в одном из 11 регистров СОЗУ или в АС. Данные

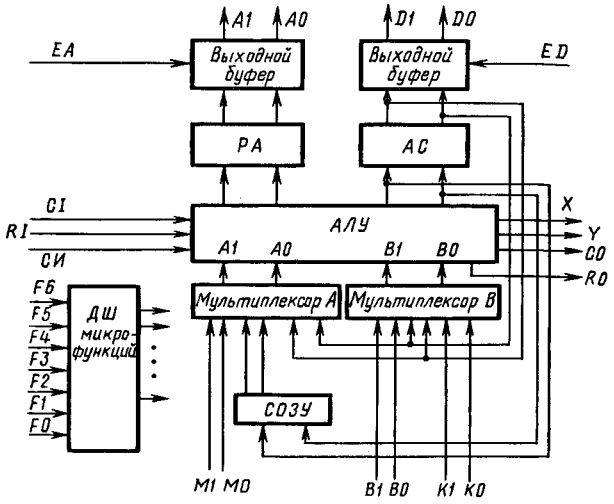


Рис. 9.3. Структурная схема ЦПЭ

от входных шин, регистров СОЗУ или АС поступают на входы АЛУ через два внутренних мультиплексора А и В. Дополнительные входы и выходы переноса и сдвига вправо служат для организации переносов и сдвигов.

Управление работой АЛУ, СОЗУ и мультиплексорами производит дешифратор *ДШ* микрофункций, декодирующий информацию на шине микрокоманд F.

Наличие двух входных шин В и М позволяет производить ввод информации в ЦПЭ из ОЗУ и внешних устройств по разным шинам, при этом повышается быстродействие

в процедурах обмена информации, кроме того, информация на шинах может подготавливаться заранее.

Сверхоперативное ЗУ(СОЗУ) содержит 11 РОН, обозначенных R0 — R9 и Т. Информация с выхода СОЗУ поступает через мультиплексор А на вход АЛУ, а из АЛУ, в свою очередь, на вход СОЗУ.

Для запоминания результатов вычислений в ЦПЭ есть еще независимый регистр АС-аккумулятор. Выход АС подключен через мультиплексор А ко входу АЛУ, кроме того, выход АС подключен к выходному буферу (с тремя состояниями) магистрали. Передача информации во внешние устройства и ОЗУ осуществляется через АС по магистрали D.

Мультиплексоры А и В выбирают входные данные для двух входов АЛУ в зависимости от кода микрокоманды. Особенностью мультиплексора В является то, что его выходные сигналы образованы как поразрядная конъюнкция кода на шине К и выбранных информационных сигналов. Таким образом, информация на каком-либо выходе мультиплексора В зависит от кода на соответствующем входе шины К. Это позволяет производить гибкое маскирование разрядов.

Арифметическо-логическое устройство способно выполнять арифметические и логические операции в дополнительном коде, $+1$, -1, поразрядное логическое сложение и умножение, поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ и поразрядное логическое дополнение. Результат операции АЛУ может быть записан в АС или в одном из РОН СОЗУ. Для выполнения операции сдвига вправо выведены отдельные «Вход сдвига вправо» RI и «Выход сдвига вправо» RO. Линии входа и выхода переноса CI и C0 предназначены для распространения последовательного переноса между секциями ЦПЭ. Данные на выходы C0 и R0 поступают через буферы с тремя состояниями (на схеме не показаны), причем разрешается выдача либо сигнала C0, либо сигнала R0, поэтому эти выходы могут быть объединены. Стандартные выходы для схем ускоренного переноса X и Y позволяют организовать ускоренный параллельный перенос при использовании БИС СУП. В этом случае выходы C0 не нужны, а информация о значении переноса в каждую секцию ЦПЭ поступает из СУП (рис. 9.4).

Маскирование входов АЛУ при помощи шины К значительно расширяет возможности АЛУ. При логических операциях выход переноса является логической сборкой по ИЛИ всех разрядов слова и позволяет фиксировать

нулевое значение результата или одного из операндов. При арифметических операциях шина K используется для маскирования частей обрабатываемых слов. Кроме того, шина K необходима для передачи констант из микропрограммного УУ в ЦПЭ.

Отдельный выход АЛУ поступает на регистр адреса RA и с него через выходной буфер с тремя состояниями на шину A. Таким образом, для задания адреса внешнего устройства или ячейки ОЗУ заносят информацию в RA.

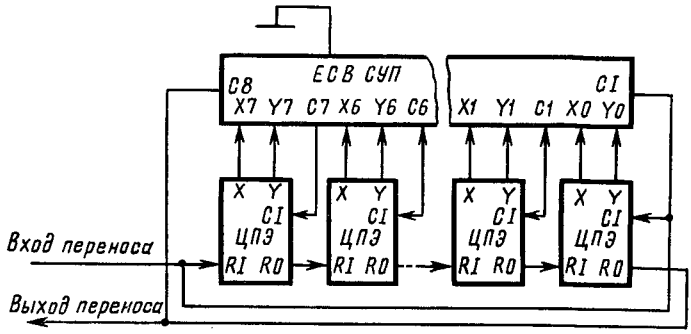


Рис. 9.4. Схема ОБ с ускоренным переносом

Описание функционирования ЦПЭ. На вход F ЦПЭ поступает микрокоманда, которая дешифрируется. Мультиплексоры выбирают операнды, и АЛУ производит заданную операцию. По отрицательному фронту сигнала синхронизации CLK результат операции заносится либо в AC, либо в СОЗУ, либо в RA. После поступления положительного фронта сигнала CLK может быть подана новая микрокоманда. Необходимо отметить, что цепи выработки сигналов переноса C0, сдвига вправо R0 и ускоренного переноса не стробируются сигналом GLK, что позволяет проверить содержимое регистров или определить наличие переноса при выполнении арифметической операции (например, переполнение разрядной сетки при сложении) без выполнения самой операции или изменения содержимого регистров. Проверка осуществляется с помощью запрета импульса синхронизации CLK в данном такте на управляющий вход ЦПЭ. При этом АЛУ выполняет заданную в микрокоманде операцию и выдает сигнал переноса, но

результат операции в регистры не заносится. Этот метод получил название *условной синхронизации* и заключается в стробировании сигнала синхронизации CLK одним из разрядов микрокоманды.

Содержание выполняемой микрокоманды определяется функциональной F- и регистровой R-группами кода микрокоманды, подаваемой на вход F ЦПЭ. При этом три разряда F-группы определяют выполняемую функцию, а четыре разряда R-группы — источник или приемник информации, т. е. регистр СОЗУ или АС. R-группа 1 включает в себя регистры R0 — R9, T и AC и обозначается символом R_n. R-группа 2 и R-группа 3 — регистры T и AC и обозначаются AT. Формат и кодировка F- и R-групп приведены в табл. 9.1.

В табл. 9.2 приведены микроинструкции ЦПЭ, в табл. 9.3 микроинструкции даны для случаев, когда все разряды

Таблица 9.1

R-группа	Регистр	F3	F2	F1	F0	$F_{(16)}$
1	P0	0	0	0	0	0
	R1	0	0	0	1	1
	R2	0	0	1	0	2
	R3	0	0	1	1	3
	R4	0	1	0	0	4
	R5	0	1	0	1	5
	R6	0	1	1	0	6
	R7	0	1	1	1	7
	R8	1	0	0	0	7
	R9	1	0	0	1	9
	T	1	1	0	0	C
	AC	1	1	0	1	D
2	T	1	0	1	0	A
	AC	1	0	1	1	B
3	T	1	1	1	0	E
	AC	1	1	1	1	F

F-группа	F6	F5	F4	$F_{(16)}$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	2
3	0	1	1	3
4	1	0	0	4
5	1	0	1	5
6	1	1	0	6
7	1	1	1	7

Таблица 9.2

F-группа	R-группа	Инструкция	F-группа	R-группа	Инструкция
0	1	$R_n + (AC \wedge K) + CI \rightarrow R_nAC$	4	1	$CI\{V\}(R_n \wedge AC \wedge K) \rightarrow CO$; $R_n \wedge (AC \wedge \wedge K) \rightarrow R_n$
	2	$M + (AC \wedge K) + CI \rightarrow AT$		2	$CI\{V\}(M \wedge AC \wedge K) \rightarrow CO$; $M \wedge (AC \wedge K) \rightarrow \rightarrow AT$
	3	$ATO \wedge (MO \wedge KO) \rightarrow RO$; $RIV[(I1 \wedge K1) \wedge AT1] \rightarrow AT1$; $[ATO(MO \wedge KO)] \vee [AT1 \vee (M1K1)] \rightarrow \rightarrow ATO$		3	$CI\{V\}(AT \wedge B \wedge K) \rightarrow CO$; $AT \wedge (M \wedge K) \rightarrow \rightarrow AT$
1	1	$KVR_n \rightarrow RA$; $R_n + K + CI \rightarrow R_n$	5	1	$CI\{V\}(R_n \wedge K) \rightarrow CO$; $K \wedge R_n \rightarrow R_n$
	2	$KVM \rightarrow RA$; $M + K + CI \rightarrow AT$		2	$CI\{V\}(DI \wedge K) \rightarrow CO$; $K \wedge DM \rightarrow AT$
	3	$(\overline{AT}VK) + (AT \wedge K) + CI \rightarrow AT$		3	$CI\{V\}(AT \wedge K) \rightarrow CO$; $K \wedge AT \rightarrow AT$
2	1	$(AC \wedge K) - 1 + CI \rightarrow R_n$	6	1	$CI\{V\}(AC \wedge K) \rightarrow CO$; $R_n \vee (AC \wedge K) \rightarrow R_n$
	2	$(AC \wedge K) - 1 + CI \rightarrow AT$		2	$CI\{V\}(AK \wedge K) \rightarrow CO$; $M \vee (AC \wedge K) \rightarrow R_n$
	3	$(I \wedge K) - 1 + CI \rightarrow AT$		3	$CI\{V\}(B \wedge K) \rightarrow CO$; $AT \vee (B \wedge K) \rightarrow AT$
3	1	$R_n + (AC \wedge K) + CI \rightarrow R_n$	7	1	$CI\{V\}(R_n \wedge AC \wedge K) \rightarrow CO$; $R_n \bar{\oplus} (AC \wedge \wedge K) \rightarrow R_n$
	2	$M + (AC \wedge K) + CI \rightarrow AT$		2	$CI\{V\}(M \wedge AC \wedge K) \rightarrow CO$; $M \bar{\oplus} (AC \wedge \wedge K) \rightarrow AT$
	3	$AT + (I \wedge K) + CI \rightarrow AT$		3	$CI\{V\}(AT \wedge I \wedge K) \rightarrow CO$; $AT \bar{\oplus} (B \wedge K) \rightarrow \rightarrow AT$

Таблица 9.3

K=00	M	K=11	M	$F_{гр}$
$R_n+CI\to R_n$, AC $M+CI\to AT$ $ATO\to RO$, $AT1\to ATO$, $R1\to AT1$	ILR ACM SRA	$AC+R_n+CI\to R_n$, AC $M+AC+CI\to AT$	ALR AMA	0
$R_n\to RA$ $R_n+CI\to R_n$ $M\to RA$ $M+CI\to R_n$ $AT+CI\to AT$	LMI LMM CIA	$11\to RA$ $R_n-1+CI\to R_n$ $11\to RA$ $M-1+CI\to AT$ $AT-1+CI\to AT$	DSM LDM DCA	1
$CI-1\to R_n$ $CI-1\to AT$ Смотри CSA	CSR CSA	$AC-1+CI\to R_n$ $AC-1+CI\to AT$ $B-1+CI\to AT$	SDR SDA LDI	2
$R_n+CI\to R_n$ Смотри ACM $AT+CI\to AT$	INR INA	$AC+R_n+CI\to R_n$ Смотри AMA $B+AT+CI\to AT$	ADR AIA	3
$CI\to CO$ $0\to R_n$ $CI\to CO$ $0\to AT$ Смотри CLA	CLR CLA	$CI\vee(R_n\wedge AC)\to CO$, $R_n\wedge AC\to R_n$ $CI\vee(M\wedge AC)\to CO$, $M\wedge AC\to AT$ $CI\vee(AT\wedge B)\to CO$, $AT\wedge B\to AT$	ANR ANM ANI	4
Смотри CLR Смотри CLA Смотри CLA		$CI\vee R_n\to CO$, $R_n\to R_n$ $CI\vee M\to CO$, $M\to AT$ $CI\vee AT\to CO$, $AT\to AT$	TZR LTM TZA	5
$CI\to CO$, $R_n\to R_n$ $CI\to CO$, $M\to AT$ Смотри NOP	NOP LMF	$CI\vee AC\to CO$, $R_n\vee AC\to R_n$ $CI\vee AC\to CO$, $M\vee AC\to AT$ $CI\vee B\to CO$ $B\vee AT\to AT$	ORR ORM ORI	6
$CI\to CO$ $\overline{R_n}\to R_n$ $CI\to CO$ $M\to AT$ $CI\to CO$ $AT\to AT$	CMR LCM CMA	$CI\vee(R_n\wedge AC)\to CO$, $R_n\bar{\oplus}AC\to R_n$ $CI\vee(M\wedge AC)\to CO$, $M\bar{\oplus}AC\to AT$ $CI\vee(AT\wedge B)\to CO$, $B\oplus AT\to AT$	XNR XNM XNI	7

шины K равны 0 и 1. Мнемоника каждой микроинструкции приведена для справки и используется для краткой записи при составлении микропрограмм.

При описании микрооперации применяют следующие символы: B, K, M — данные соответственно на шинах B, K, M; CI, RI — данные на входах переноса и сдвига вправо; C0, R0 — данные на выходах переноса и сдвига вправо; R_n — содержимое регистра; номер n задается в соответствии с табл. 9.1 (для R-группы 1); AC — содержимое аккумулятора; AT — содержимое аккумулятора или регистра T (для R-групп 2 и 3); RA — содержимое регистра адреса; «+», «—» — сложение и вычитание; $\wedge$ — логическое И; $\vee$ — логическое ИЛИ; $\overline{\oplus}$ — инверсия суммы по модулю 2.

В качестве примера рассмотрим F-группу 3 и R-группу 1. Из табл. 9.2 следует, что эта микроинструкция производит логическое умножение содержимого AC на данные шины K и результат складывает с содержимым регистра R_n и CI. Сумма заносится в регистр R_n. Из табл. 9.3 видно, что при значении всех разрядов шины K:

K = 0 — выполняется микроинструкция INR, т. е. сложение CI с содержимым R_n и запись результата в R_n. Эта микроинструкция используется для увеличения содержимого R_n на 1;

K = 1 — выполняется микроинструкция ADR, т. е. сложение содержимого AC, R_n и CI и запись результата в R_n. Эта микроинструкция используется для сложения AC и R_n.

Особенность данной системы микроинструкций заключается в отсутствии непосредственной пересылки содержимого одного регистра СОЗУ в другой. Эту операцию можно выполнить в два приема, используя AC для промежуточного хранения информации регистра-источника.

Схема ускоренного переноса К589ИК03 (СУП) (рис. 9.4). Данная схема предназначена для формирования групповых переносов при совместном использовании с ЦПЭ, имеет 17 информационных входов, восемь информационных выходов и один управляющий вход, переводящий выход самого старшего разряда в «третье» состояние. Одна схема СУП позволяет организовать 16-разрядный ОБ, т. е. может работать совместно с восемью ЦПЭ.

Схема ускоренного переноса представляет собой совокупность восьми комбинационных схем типа И-ИЛИ-НЕ и реализует следующие соотношения:

$\overline{C1} = X0 \cdot Y0 + Y0 \cdot \overline{C1}$

$\overline{C2} = X1 \cdot Y1 + Y1 \cdot Y0 \cdot Y0 + Y1 \cdot Y0 \cdot \overline{C1}$

$\overline{C3} = X2 \cdot Y2 + Y2 \cdot Y1 \cdot X1 + Y2 \cdot Y1 \cdot Y0 \cdot X0 + Y2 \cdot Y1 \cdot Y0 \cdot \overline{C1}$ и т. д.

Схема операционного блока (рис. 9.5). Содержит четыре микросхемы ЦПЭ и одну микросхему СУП, образующую восьмиразрядные информационные магистрали B, M, A, D. Использование СУП позволило уменьшить время распространения переносов, т. е. повысить быстродействие ОБ.

Особенность данной схемы — последовательное распространение переноса через последнюю секцию ЦПЭ, т. е. для выхода переноса из ОБ используется выход C0 последней секции ЦПЭ вместо выхода C(N+4) ↔СУП. Выходы C0 и R0 объединены, так как эти выходы ЦПЭ имеют три состояния, а выход C(N+4) СУП не имеет третьего состояния. Второй особенностью данной схемы является способ организации шины K для маскирования разрядов ЦПЭ, который заключается в следующем: шина K состоит из четырех разрядов, причем разряды K0 и K3 маскируют младший и старший разряды ЦПЭ, а разряды K1 и K2 маскируют соответственно 2, 3, 4 и 5, 6, 7-й разряды. Ниже выделены отдельно младший и старший разряды байта, а также младший и старший полубайты информации (табл. 9.4).

Устройство управления. В работе [30] подробно рассмотрена работа микро-ЭВМ, использующей микросхему БМУ К589ИК01, и указаны недостатки таких микро-ЭВМ. Устройство управления (УУ) микро-ЭВМ может быть реализовано на регистрах с динамическим управлением записью, при этом расширяются возможности по выбору адреса следующей микрокоманды. Устройство управления микро-ЭВМ обеспечивает последовательность выборки команд в соответствии с алгоритмом решаемой задачи, формирует управляющие коды для операционного блока

Таблица 9.4

K3	K2	K1	K0	$K_{(16)}$	Функции маскирования
0	0	0	0	0	Константа «Все нули»
0	0	0	1	1	Выбор младшего разряда
0	0	1	1	3	Выбор младшего полубайта
1	1	0	0	C	Выбор старшего полубайта
1	0	0	0	8	Выбор старшего разряда
1	1	1	1	F	Выбор всего байта

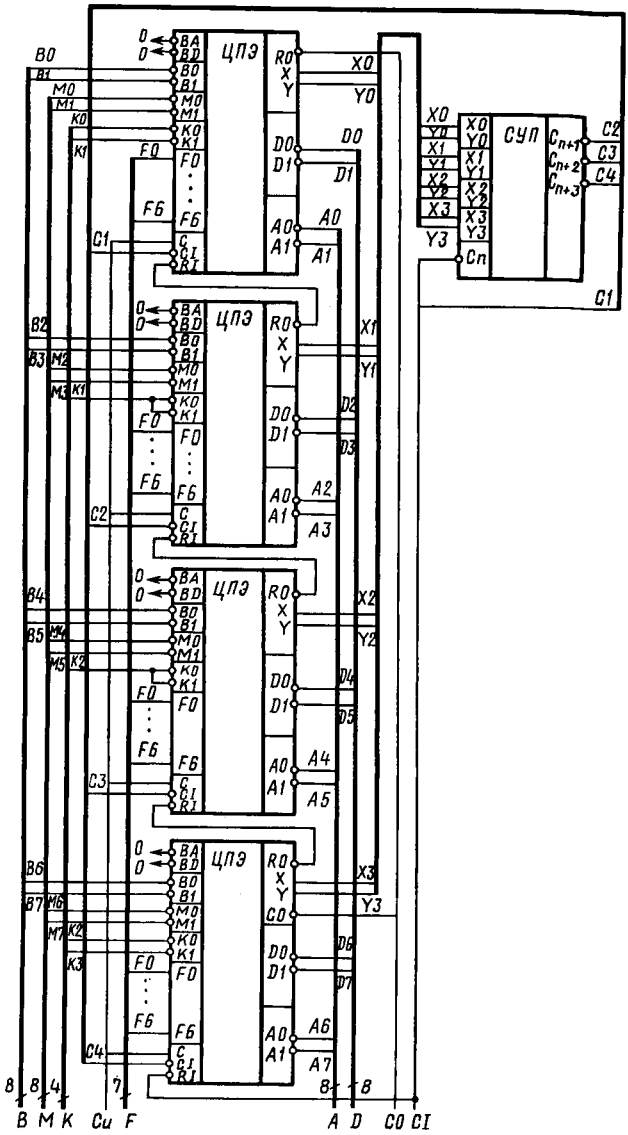


Рис. 9.5. Принципиальная схема операционного блока

и совокупность управляющих сигналов, обеспечивающих совместную работу блоков при выполнении текущей команды. На рис. 9.6 приведена функциональная схема УУ, реализующего микрокомандный принцип.

Запись микрокоманд ведет ПЗУ емкостью 512×32. Адрес выполняемой микрокоманды определяется выходным кодом регистра RGA и одноразрядным кодом мультиплексора условий MS (К155КП1). Изменение адреса микрокоманды осуществляется по фронту синхроимпульса CLK, поступающего на управляющий вход регистра RGA

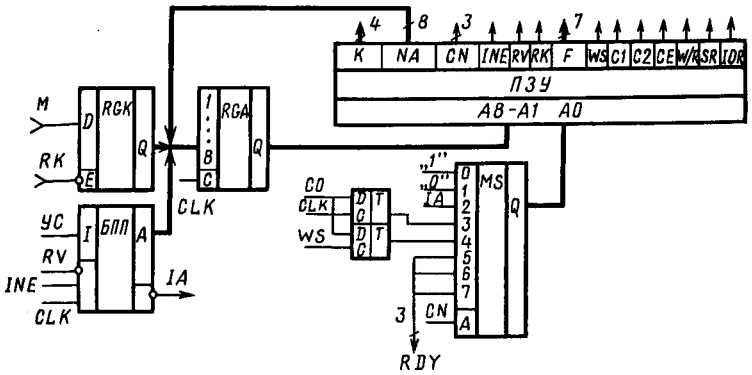


Рис. 9.6. Функциональная схема устройства управления

(2×К155 ИР1). Адрес следующей микрокоманды в общем случае может формироваться по содержимому адресной части микрокоманды (NA1 — NA9), по содержимому регистра RGK или по выходному коду блока приоритетного прерывания (БПП) (К589 ИК14).

Устройства, определяющие адрес следующей микрокоманды, объединены выходами в общую магистраль, подключенную ко входам RGA. Объединение в одну магистраль происходит на основе использования «третьего» состояния схем, подключенных ко входу RGA. В каждой микрокоманде активным может быть только один из источников следующего адреса. Для выполнения этого условия в состав микрокоманды включены микроинструкции RV, RK, воздействующие на управляющие входы RGK (К589 ИР12), БПП, ПЗУ (4×556РТ5), формирующие

код NA. Источник следующего адреса определяется в соответствии с приведенными данными:

Выход в третьем состоянии.., Активный выход	БПП, ПЗУ RGK	RGK, ПЗУ БПП	RGK, БПП ПЗУ
RV	1	0	1
RK	0	1	1

Мультиплексор условий MS К155КП7 формирует код младшего адреса микрокоманды, позволяя организовать условные переходы в зависимости от значения одного из восьми входных сигналов. Адрес входа, формирующего выходной сигнал мультиплексора, определяется кодом CN0, CN1, CN2 текущей микрокоманды. Микроинструкция CN поступает на управляющие входы мультиплексора MS.

В качестве входных сигналов MS используются следующие: напряжение логической единицы ($CN=0_8$); напряжение логического нуля ($CN=1_8$); выходной сигнал IA, формируемый БПП ($CN=2_8$); значение C0 ($CN=3_8$, 4_8), записанное в D-триггеры, управляемые тактирующим синхроимпульсом CLK и микроинструкцией WS; значения трех внешних управляющих сигналов RDY ($CN=5\div7_8$).

Приведенная структура устройства формирования следующего адреса позволяет в каждой микрокоманде осуществлять переходы указанных типов: безусловный переход к микрокоманде с адресом, определяемым кодом NA текущей микрокоманды и заданным значением A0; условный переход к микрокоманде с адресом, определяемым кодом NA текущей микрокоманды и кодом A0, формируемым мультиплексором по значению одного из источников его входных сигналов; переход к микрокоманде, адрес которой определяется выходными кодами БПП и MS; переход к микрокоманде, адрес которой определяется кодами RGK и MS.

Кроме микроинструкций, формирующих адрес следующей микрокоманды (МК), в состав последней входят микроинструкции, обеспечивающие согласованную работу всех блоков, входящих в микро-ЭВМ.

Для управления памятью в состав МК включены: микроинструкция CE, разрешающая обращение к устройствам памяти; микроинструкция W/R, задающая режим работы ОЗУ.

Для управления операционным блоком в МК входят микроинструкция F, которая задает 7-разрядным кодом выполняемую процессором операцию, микроинструкция K, выполняющая функцию маскирования разрядов ОБ, и

микроинструкция С, состоящая из двух разрядов и задающая код CI в ОБ. Значение сигнала CI в ОБ представлены ниже:

C1	0	1	0
C2	0	*	1
CI	0	1	C0

Для обработки сигналов от КУ используется микроинструкция INE, формирующая строб разрешения прерывания для БПП. На информационные входы БПП подаются сигналы от управляющей клавиатуры, задающие восемь режимов работы микро-ЭВМ. Состояния входов БПП опрашиваются только после выполнения последовательности микрокоманд, относящихся к данной команде. Для обеспечения этого условия опрос БПП осуществляется по микроинструкции INE лишь после выполнения команды, и поэтому она размещается в последней микрокоманде.

Возникновение запроса на одном из входов БПП обусловливает появление сигнала IA (низкий уровень) на выходе БПП в момент опроса его микроинструкцией INE. По возникшему сигналу IA устройство управления переходит к программе анализа сигнала, вызвавшего прерывание исполняемой программы. В результате выполнения программы анализа происходит переход к программе, начальный адрес которой определяется выходным кодом БПП, соответствующим появившемуся входному сигналу с управляющей клавиатуры.

Работой внешних устройств управляют микроинструкции IOR — запрос обмена с внешним устройством и SR — запрос обмена с платой звуковой индикации.

Блок оперативного запоминающего устройства. Оперативное запоминающее устройство микро-ЭВМ имеет объем 256 слов по 8 разрядов, или 256 байт, и выполняет функцию хранения рабочих программ, данных и адресов. Реализовано ОЗУ на восьми микросхемах К561РУ2 объемом 256 бит, причем каждый кристалл ОЗУ обеспечивает хранение одного разряда слова. Выбор ОЗУ К561РУ2 обоснован следующим: низкая потребляемая мощность (50 мкВт); возможность сохранения информации в ОЗУ при выключении микро-ЭВМ при условии питания ОЗУ от аккумулятора или батарей; соответствие объема ОЗУ (256 бит) 8-разрядной шине адреса.

Микросхемы К561РУ2 выполнены по К-МОП-технологии, поэтому для согласования выходов микросхем с ТТЛ-

входами в ОБ применены буферные усилители К561ПУ4. На схеме (рис. 9.7) выходы микросхем ОЗУ DM0—DM7 поступают на входы буферных усилителей, а их выходы M0—M7 образуют шину данных из ОЗУ (магистраль М).

Для управления работой ОЗУ используются поступающие из УУ два сигнала: W/R — запись/считывание (определяет режим) и CE — выбор кристалла (разрешает работу ОЗУ).

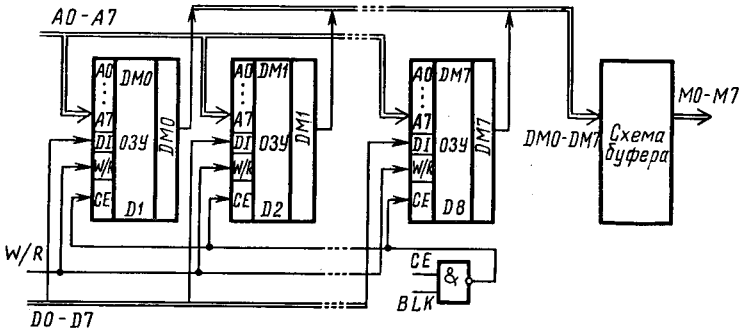


Рис. 9.7. Принципиальная схема ОЗУ

Работа ОЗУ в режиме считывания информации. Для считывания информации из ОЗУ необходимо последовательно выполнить следующие процедуры (рис. 9.8, *а*): выставить на шину адреса код ячейки ОЗУ, с которой считывается информация; установить в «0» сигналы W/R и CE.

Длительность управляющего сигнала CE должна превышать 1,2 мкс, поэтому считывание информации из ОЗУ требует в данной микро-ЭВМ три такта ($T = 0{,}5$ мкс).

Работа ОЗУ в режиме записи информации. Для записи информации в ОЗУ необходимо выполнить следующее: выставить на шину адреса код адреса, в который необходимо занести информацию; выставить на шину данных записываемую информацию; установить в «1» сигнал W/R, переводящий ОЗУ в режим записи, и в «0» сигнал CE, разрешающий запись информации.

Длительность сигнала записи равна 1,2 мкс, после этого сигнал CE может быть снят, т. е. переведен в «1» (рис. 9.8, *б*). Запись информации в ОЗУ, так же как и при считывании, производится за три такта.

Для расширения функциональных возможностей в схему ОЗУ введен элемент 2И—НЕ, позволяющий производить стробирование CE:

Вид работы	W/R	CE
Хранение	*	0
Считывание	0	1
Запись	1	1

Здесь * — произвольное состояние.

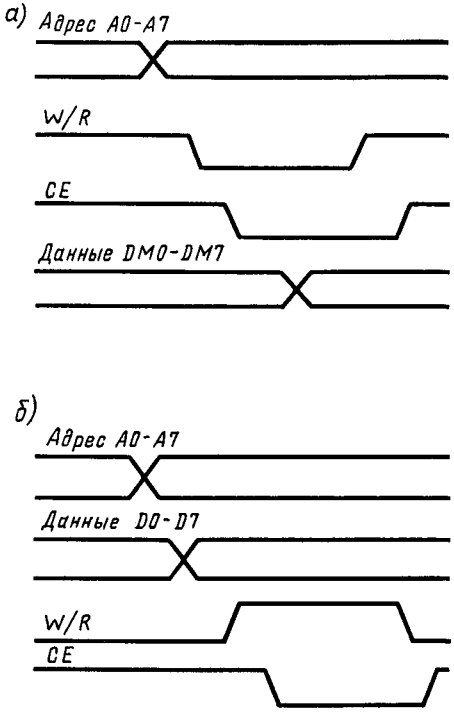


Рис. 9.8. Временны́е диаграммы режимов считывания (*а*) и записи (*б*)

Таким образом, при подаче сигнала низкого уровня на вход BLK сигнал CE переводится в «1» и работа ОЗУ блокируется, при этом выходы DM ОЗУ переводятся в высокоимпедансное состояние. Этот режим используется при подключении к шине внешнего ПЗУ. Управляющий сигнал CE поступает на входы ОЗУ через инвертирующий элемент.

Клавиатура КД и КУ. Процедура загрузки в память микро-ЭВМ рабочей программы и исходных данных может быть выполнена с помощью клавиатуры, сигналы с которой считываются и анализируются самой микро-ЭВМ. Широкое распространение получила шестнадцатеричная система счисления, формирующая двоичные коды вводимых данных.

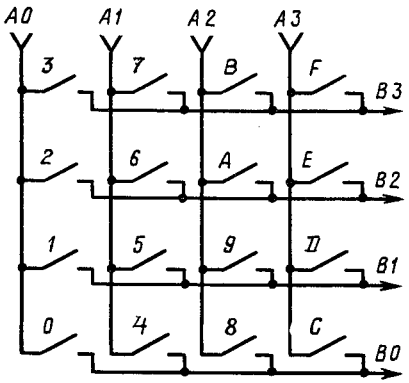


Рис. 9.9. Схема подключения клавиатуры

Формирование двоичных кодов, соответствующих замыканию одной из 16 кнопок, может выполняться микро-ЭВМ при подключении кнопок в соответствии со схемой рис. 9.9.

Для формирования двоичных кодов последовательно формируются нулевые уровни на шинах A0—A3. Замыкание одного из контактов *0—3* обеспечивает прохождение нулевого уровня на один из выходов B0—B3 клавиатуры в момент формирования сигнала на шине A0. Код на шинах B0—B3 однозначно определяет нажатую кнопку. При замыкании одной из кнопок *4—7* выходной сигнал на шинах B0—B3 возникает во время формирования сигнала на шине A1 и т. д.

Формируется двоичный код специальной программой, учитывающей время появления сигнала на шинах B0—B3 и устраняющей возможные импульсные помехи на шинах B и влияние дребезга контактов клавиатуры. Более подробно программа считывания сигналов клавиатуры будет рассмотрена дальше.

Кроме клавиатуры шестнадцатеричного кода в микро-ЭВМ предусмотрены клавиши для задания режима работы: сброс, пошаговое RS(runstep) и циклическое RS(run cycle) исполнение команд, установка адреса ОЗУ SA(set address), запись в ОЗУ WI(write increment); чтение содержимого ОЗУ с уменьшением и увеличением следующего адреса RD, RI(read decrement, increment). Информация с клавиатуры задания режимов работы поступает на входы БПП в УУ.

§ 9.2. Математическое обеспечение микро-ЭВМ

Операционная система микро-ЭВМ. Структура операционной системы микро-ЭВМ приведена на рис. 9.10 (блок *1* — переход в общую системную программу). Работа микро-ЭВМ начинается с установки значений во всех

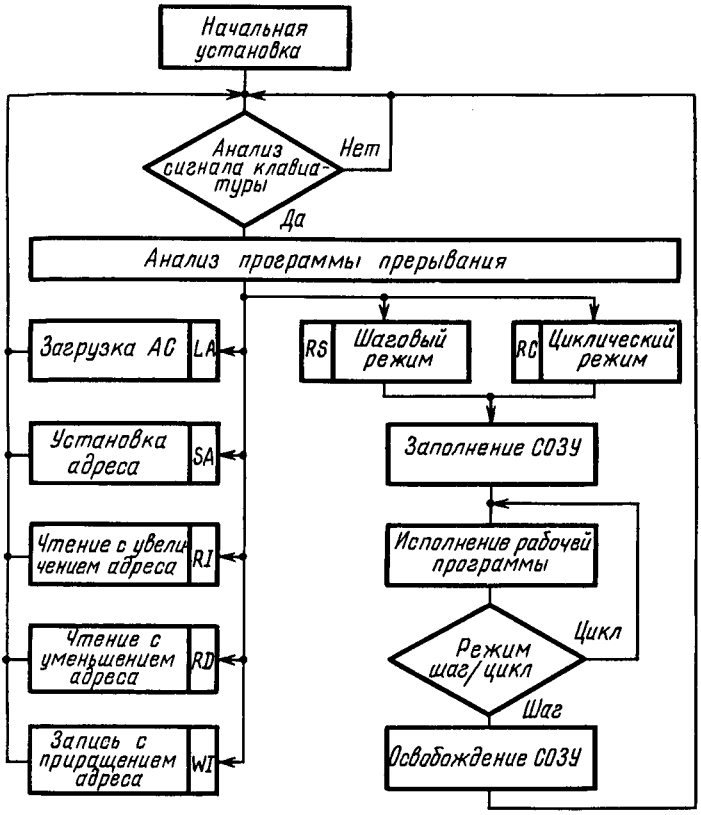


Рис. 9.10. Операционная система микро-ЭВМ

регистрах ЦПЭ. Вызывается начальная установка нажатием клавиши R(reset) или при первоначальном включении питающего напряжения. В процессе начальной установки обнуляются регистры ЦПЭ АС, RA, PC и программно формируется звуковой сигнал, представляющий собой нарастающий звуковой тон.

После начальной установки системных регистров микро-ЭВМ переходит к программе анализа сигналов клавиатуры, задающих режим работы. Схема программы представлена на рис. 9.11. Начинается программа блоком анализа выхода IA БПП, который примет уровень логического нуля при нажатии одной из клавиш, задающих режим. При IA = 0 управление передается блоку временной задержки, позволяющему устранить влияние случайного сигнала на выходе IA. При повторном опросе сигнала IA через время τ управление передается первому блоку при исчезновении запроса прерывания (IA = 1), а в случае повторного подтверждения сигнала IA = 0 формируется код прерывающей программы на выходах K0—K2 БПП и после подпрограммы формирования звукового сигнала управление передается программе, вызванной нажатой кнопкой.

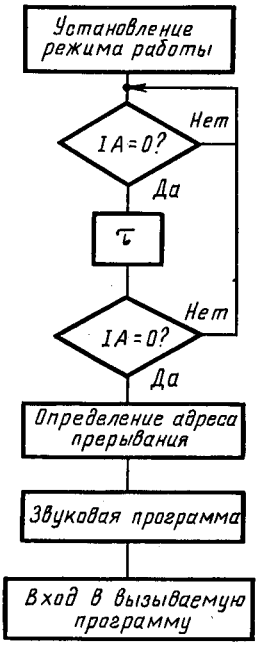


Рис. 9.11. Схема анализа сигналов клавиатуры

Во многих режимах микро-ЭВМ используется шестнадцатеричная клавиатура для ввода кодов исходных данных и программ. Формирование 8-разрядного кода, соответствующего нажатым клавишам, осуществляется в соответствии с программой BTN, схема которой представлена на рис. 9.12. Начинается программа с анализа сигнала по информационной шине B, который появится в случае нажатия одной из шестнадцати клавиш. Формирование двоичного кода, соответствующего нажатой клавише, осуществляется последовательной генерацией единичных кодов на одной из вертикальных шин клавиатуры (шины A0—A3 ЦПЭ) и анализом сигналов на горизонтальных шинах клавиатуры, поступающих на входы B0—B3 ЦПЭ (см. рис. 9.9). Единичный сигнал на шинах B0—B3 определяет код нажатой клавиши, при этом код шин A0—A3 с единичным сигналом соответствует младшим разрядам, а код шин B0—B3, в котором был сформирован единичный сигнал,— старшим разрядам вводимого числа.

Для устранения случайного сигнала при анализе шины B клавиатура опрашивается повторно через время τ после появления первого сигнала. Время τ выбирается бо́льшим, чем время переходных процессов сигнала кла-

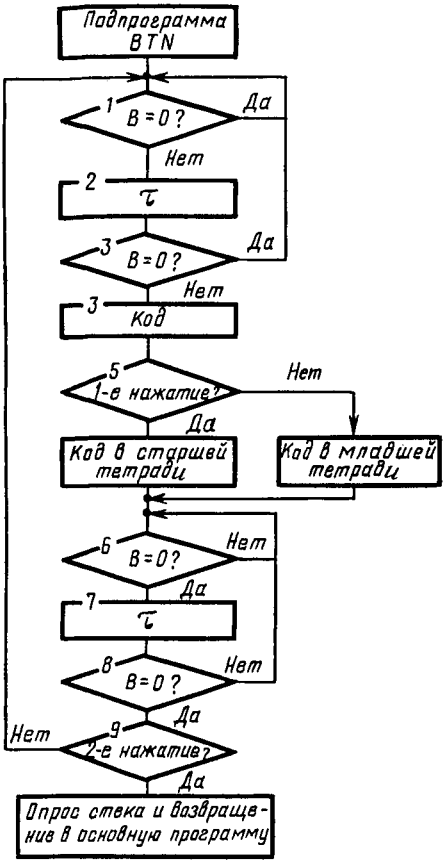


Рис. 9.12. Схема программы формирования кода

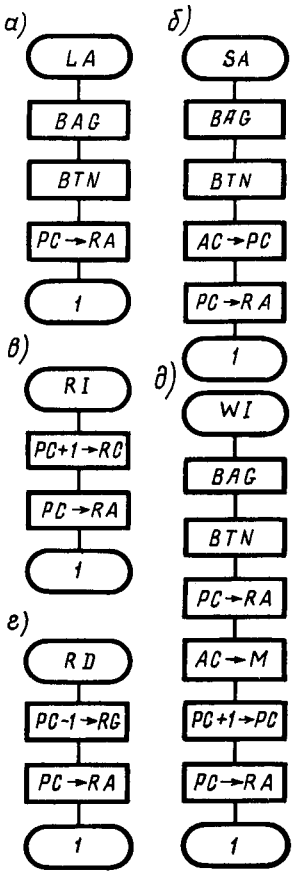


Рис. 9.13. Структура системных программ

виатуры. При подтверждении сигнала на шине B программно формируется четырехразрядный код, соответствующий нажатой клавише. При первом нажатии клавиши код заносится в старшие разряды формируемого слова. Блоки *6—8* фиксируют момент отпускания кнопки с учетом

переходных процессов. В блок 7 управление передается после первого момента исчезновения сигнала на шине B и после подтверждения через время τ исчезновения сигнала на шине. Процедура повторяется для записи младших разрядов слова. Завершается ввод кода после двух нажатий кнопок. Обращение к подпрограмме формирования кодов из системных программ осуществляется через стековую память.

К системным программам относятся программы загрузки данных в аккумулятор LA (load acc), установления адреса SA (set address), чтения содержимого памяти с увеличением адреса RI (read increment), чтения содержимого памяти с уменьшением адреса RD (read decrement), записи в память с увеличением адреса WI (write increment), пошагового исполнения рабочей программы RS (run step), циклического выполнения рабочей программы RC (run cycle).

Системная программа LA служит для записи в аккумулятор кода с клавиатуры. Схема программы приведена на рис. 9.13, *а*.

Начинается программа с заполнения стека адресом выхода из подпрограммы BAG и обращением к подпрограмме BTN. Затем подпрограммой BTN записывается в AC код нажатых клавиш и содержимое R8 (программный счетчик PC) записывается в RA. Программа SA обеспечивает запись в PC и RA кода с клавиатуры (рис. 9.13, *б*).

Программа RI предназначена для просмотра содержимого ОЗУ начиная с адреса, установленного в RA, с увеличением его на единицу после каждого нажатия клавиши RI (рис. 9.13, *в*).

Программа RD аналогична предыдущей, но отличается тем, что содержимое RA уменьшается на единицу (рис. 9.13, *г*).

Программа WI предназначена для загрузки ячейки ОЗУ с клавиатуры по адресу, установленному в RA, с последующим наращиванием адреса (рис. 9.13, *д*).

Программы циклического и пошагового исполнения рабочих программ (см. рис. 9.10) начинаются с установки триггера s устройства управления в «1» при циклическом исполнении и в «0» — при шаговом.

Затем в рабочие регистры процессора переписывается информация из специальной области ОЗУ (рис. 9.14), которая была занесена туда из регистров после выполнения последней команды перед остановом. Таким образом

восстанавливается состояние регистров. Эта операция необходима, так как системные программы операционной системы используют те же регистры ЦПЭ, что и рабочая программа. Затем выполняется очередная команда, номер которой определяется содержимым программного счетчика PC (R8). После этого анализируется состояние триггера s. При единичном коде триггера s последовательно исполняются все команды программы, при нулевом коде после каждой команды происходит останов рабочей программы. Информация из регистров ЦПЭ переписывается в ОЗУ для хранения, а системные команды операционной системы могут использовать регистры процессора. При выполнении очередной команды управление передается программе анализа сигналов клавиатуры. Для выполнения следующего шага рабочей программы необходимо нажать кнопку RS, после чего повторится описанная процедура.

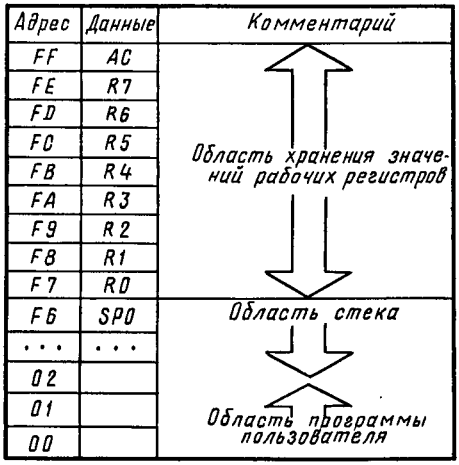


Рис. 9.14. Карта памяти

Микропрограммирование. Выполнение программ в микро-ЭВМ с микропрограммным управлением может осуществляться либо в соответствии с последовательностью микрокоманд, записанной в ПЗУ в виде рабочей программы, либо в соответствии с последовательностью команд, записанных в виде рабочей программы в ОЗУ. Во втором случае в ПЗУ содержатся микрокоманды, обеспечивающие выполнение системы команд. Каждая команда имеет начальный адрес, задающий адрес первой микрокоманды. Заканчиваются команды микрокомандами обращения к операционной системе.

Микропрограммная реализация команд на ПЗУ позволяет изменять систему команд простой заменой ПЗУ с записанной новой системой команд без изменения

остальных блоков микро-ЭВМ. Для каждого конкретного случая применения микро-ЭВМ разработчик имеет возможность использовать свою систему команд, наиболее полно соответствующую решаемой задаче [31].

Рассмотрим некоторые команды, реализованные в исследуемом лабораторном стенде. Команда 0→R6 обеспечивает обнуление регистра R6 ЦПЭ. Команда состоит из одной микрокоманды (табл. 9.5). Адрес команды 0→R6 соответствует коду этой команды, т. е. старшие разряды адреса $(A1—A8)_{16}$ образуют шестнадцатеричный код BD_{16}. При входе в команды младшему разряду адреса A0 присваивается код «1». Микроинструкция NA участвует в формировании адреса следующей микрокоманды и задает код адреса разрядов (A1—A8) следующей микрокоманды. Адрес A0 задается выходным сигналом мультиплексора MS. Рассматриваемая команда состоит из одной микрокоманды, поэтому должна завершаться обращением к системной программе. Согласно рис. 9.10, выполнение любой команды завершается обращением к блоку системной подпрограммы шаг/цикл, осуществляющему опрос режима работы микро-ЭВМ.

Первая микрокоманда подпрограммы опроса режима работы шаг/цикл записана в данной операционной системе по адресу ПЗУ $IE_{16}O_2(0001.1110.0)_2$.

Микроинструкция NA формирует разряды A1—A8 указанного адреса и в шестнадцатеричной системе счисления может быть записана как IE. Мультиплексор MS при управляющем коде $CN=000_2$ формирует на выходе код логического нуля, соответствующего адресу A0. В данной команде не устанавливается режим исполнения программы (пошаговый или циклический), поэтому управляющему сигналу WS присваивается уровень логического нуля.

Микроинструкция F задает код выполняемой микрооперации для ЦПЭ. В данной команде используется инструкция CLR_n ЦПЭ, по которой обнуляется регистр, номер которого указан в микрокоманде. Инструкция CLR_n относится к 4-й F-группе (см. табл. 9.2), а регистр R6 кодируется 6-й R-группой. Поэтому микроинструкция представлена в микрокоманде кодом 4_86_{16}, где первая цифра определяет код F-группы, записанный в восьмеричной системе счисления, а вторая цифра определяет код R-группы в шестнадцатеричной системе счисления. В ПЗУ микроинструкция записывается в двоичной системе счисления. В данной микрокоманде микроинст-

Таблица 9.5

Номер операции	$(A8—A1)_{16}$	AO_2	NA_{16}	WS_2	F F_8R_{16}	CN_8	$\overline{CE}_2$	C 1,2	RV_2	RK_2	IOR	SR_2	R/W_2	INE_2	$\overline{K}_{16}$	Вид операции
1	ВД	1	1*E*	0	46	0	0	11	1	1	0	0	0	0	F	0→R6
						GOTO										
1	0A	1	00	0	2A	1	0	00	1	1	0	0	0	0	0	AC→T
2	00	1	01	0	18	0	0	11	1	1	0	0	0	0	F	R8→RA; *RB* ± 1→*RB*
3	01	0	02	0	7B	0	1	11	1	1	0	0	0	0	F	$\underline{M}$→AC
4	02	0	08	0	7B	0	1	11	1	1	0	0	0	0	F	$\underline{M}$→AC
5	08	0	19	0	7B	0	1	11	1	1	0	0	0	0	F	$\underline{M}$→AC
6	19	0	1A	0	28	0	1	00	1	1	0	0	0	0	0	AC→R8
7	1A	0	1E	0	0C	0	1	11	1	1	0	0	0	0	F	T→AC
						TZR2										
1	8C	1	A1	0	52	3	1	11	1	1	0	0	0	0	0	TZR2
2	A1	0	0A	0	6C	1	1	11	1	1	0	0	0	0	F	NOP
3	A1	1	0A	0	38	1	1	00	1	1	0	0	0	0	F	PC + 1→PC
						CALL										
1	65	1	70	0	2*A*	0	0	00	1	1	0	0	0	0	0	AC→T
2	70	0	82	0	19	1	0	11	1	1	0	0	0	0	0	R9 — 1→R9
3	82	1	8*B*	0	19	0	0	11	1	1	0	0	0	0	F	R9→*RA*
4	8B	0	80	0	08	0	1	11	1	1	0	0	0	0	F	R8→AC
5	80	0	94	0	3D	0	1	00	1	1	0	0	0	0	F	AC + 1→AC
6	94	0	9A	0	6C	0	1	11	1	1	0	0	1	0	F	*AS*→*M*
7	9A	0	A6	0	6C	1	1	11	1	1	0	0	1	0	F	AC→*M*
8	A6	1	00	0	6C	1	1	11	1	1	0	0	1	0	F	AC→M
						RETURN										
1	A7	1	A4	0	19	0	0	11	1	1	0	0	0	0	F	R9→RA
2	A4	0	57	0	39	1	1	00	1	1	0	0	0	0	F	R9 + 1→R9
3	57	1	01	0	2A	0	1	00	1	1	0	0	0	0	0	AC→T

рукция $4_8 6_{16}$ в двоичной системе счисления имеет вид $1000\ 0110_2$.

Микроинструкция CE задает режим возможного обращения к ОЗУ. В рассматриваемой микро-ЭВМ ОЗУ построено на микросхемах К561РУ2, для которых сигнал CE должен принимать нулевое значение в момент изменения кода адреса на входе ОЗУ. При нулевом сигнале CE выходные каскады ОЗУ устанавливаются в третье состояние.

В данной микрокоманде ОЗУ не используется, поэтому принято значение CE = 0.

Микроинструкция C1, 2, состоящая из двух разрядов, формирует сигнал CI для ЦПЭ. В данной микрокоманде сигнал CI определяет сигнал C0 (см. табл. 9.2), который в выполняемой команде не используется, поэтому микроинструкция C1, 2 может принимать произвольное значение. В данной микрокоманде принято $C1, 2 = 11_2$.

Микроинструкция RV, RK определяет блок, формирующий адрес следующей микрокоманды. Нулевой уровень сигнала RV (RK) устанавливает в активное состояние выходы БПП (RGK) соответственно. При единичных значениях сигналов RV и RK выходы БПП и RGK устанавливаются в третье состояние, при котором адрес следующей микрокоманды задается микроинструкцией NA исполняемой микрокоманды. В рассматриваемой команде адрес следующей микрокоманды задается микроинструкцией NA, поэтому микроинструкция RV, RK имеет значение 11_2.

Микроинструкция IOR, SR состоит из двух разрядов и используется для управления внешними устройствами, которые могут быть подключены к микро-ЭВМ. В лабораторном стенде в качестве внешнего устройства применяется схема управления звуковой сигнализацией. В данной команде внешние устройства не нужны, поэтому микроинструкция IOR, SR принимает значение 11_2.

Микроинструкция R/W задает режим работы ОЗУ чтение/запись. При R/W = 1 ОЗУ работает в режиме записи, при R/W = 0 — в режиме считывания. В рассматриваемой команде ОЗУ не используется, поэтому микроинструкция R/W может принимать значение, при котором не меняется содержимое ОЗУ, т. е. R/W = 0.

Микроинструкция INE (строб разрешения прерывания) используется для опроса БПП. При наличии запроса прерывания БПП формирует сигнал IA в моменты прихода управляющего сигнала INE. Поэтому микроинструкция INE = 1 применяется только в подпрограммах опроса БПП и в рассматриваемой микрокоманде INE = 0.

Микроинструкция K необходима для выбора типа выполняемой в ЦПЭ функции (арифметической или логической), маскирования исходных данных и формирования констант (см. табл. 9.2). Для выполнения инструкции CLR в ЦПЭ на всех разрядах ЦПЭ формируется код $K_i = 0$. Так как по входам K ЦПЭ имеют активный инверсный (нулевой) уровень сигнала, необходимо в данной

микроинструкции сформировать уровни логических единиц: $K = F_{16} = 1111_2$. При этом старший разряд K формирует входной сигнал в 8-разряде ЦПЭ, третий разряд K формирует сигнал в 7, 6 и 5-м разрядах ЦПЭ, второй разряд K — в 4, 3 и 2-м разрядах ЦПЭ и первый разряд K — в 1-м разряде ЦПЭ.

В качестве второго примера рассмотрим команду безусловного перехода GO TO. Команда состоит из двух байтов, в первом записан код команды, во втором — адрес *n* команды в ОЗУ, которой передается управление.

Для выполнения команды GO TO необходимо осуществить считывание из ОЗУ второго байта команды и занести его в регистр R8 ЦПЭ, выполняющий функции программного счетчика PC. Информация в регистры ЦПЭ записывается через промежуточную запись информации в AC, поэтому при исполнении команды GO T0 необходимо информацию, находящуюся в AC, на время исполнения команды записать в служебный регистр T, а после выполнения команды восстановить содержимое AC.

Код команды определяет адрес первой микрокоманды, следовательно, в соответствии с табл. 9.5 первую микрокоманду следует написать по адресу $0A_{16}I_2$ ПЗУ. В команде не используется установка режима исполнения программы (шаг/цикл), поэтому во всех микрокомандах принимается $WS = 0$.

В микрокоманде 1 содержимое AC переписывается в служебный регистр T. Для выполнения этой процедуры используется микроинструкция ЦПЭ SDA, относящаяся ко 2-й F-группе и к A_{16}-й R-группе. В соответствии с выполняемой операцией микроинструкции F первой микрокоманды присваивается код 2_8A_{15}.

В соответствии с табл. 9.2 по микроинструкции SDA выполняется процедура $AC - 1 + CI \rightarrow AT$. Для того чтобы содержимое AC при переписывании не изменилось, необходимо входной переменной CI присвоить значение 1. В ЦПЭ это значение CI соответствует нулевому логическому уровню сигнала, поэтому микроинструкция C1, 2 выбирается такой, чтобы обеспечить CI, равное логическому нулю. Микроинструкции присвоено значение $C1,2 = 00_2$.

Для выполнения микрооперации SDA необходимо по всем разрядам магистрали K сформировать единичный уровень сигнала. С учетом инверсного характера управления по магистрали K микроинструкции K присваивается значение $K = 0_{16}$.

Во всех микрокомандах рассматриваемой команды адрес следующей микрокоманды формируется кодом NA и блоки БПП и RGK в формировании адресов не участвуют. В связи с этим микроинструкциям RV, RK, СРП присвоены значения RV, $RK=11_2$, INE=0. Внешние устройства в данной команде не используются, поэтому микроинструкция IOR=0 и SR=0.

В микрокоманде 1 информация ОЗУ не используется, поэтому принята микроинструкция CE=1, открывающая доступ к ОЗУ, а микроинструкция R/W=0 устанавливает режим считывания информации.

Адрес следующей микрокоманды формируется микроинструкциями NA и CN. В связи со значением $NA=00$ разряды A1—A8 следующей микрокоманды равны $00_{16}^{(16)}$, а разряд A0 равен единице, т. е. коду $CN=001_2$ соответствует логическая единица на выходе MS.

Микрокоманда 2 формирует на адресной шине ОЗУ адрес второго байта команды G0 T0, по которому записан адрес команды перехода. Для этого применяется инструкция LMI, по которой выполняется процедура $R_n \rightarrow RA$; $R_n + CI \rightarrow R_n$. Для выполнения инструкции LMI необходимо во всех разрядах ЦПЭ сформировать значение K=0. В качестве регистра R_n стоит программный счетчик PC (R8), в котором к моменту выполнения микрокоманды 2 содержится увеличенный на единицу адрес выполняемой команды в ОЗУ, совпадающий с адресом второго байта команды G0 T0. Заметим, что в этой же микрокоманде осуществляется увеличение на единицу содержимого PC (R8), в результате чего формируется адрес следующей команды при естественном порядке следования команд. Для увеличения содержимого PC переменной CI присваивается единица микроинструкцией $C1,2=(11)_2$. Таким образом, инструкция LMI микрокоманды 2 будет выполнена при значениях микроинструкций $F=(1,8)_{16}$; $C1,2=(11)_2$; $\overline{K}=(F)_{16}$.

Микроинструкции NA и CN задают адрес микрокоманды 3, в которой адреса $A1 - A8=(01)_{16}$; A0=0. После формирования адреса ОЗУ необходимо осуществить запись его содержимого в АС. Для выполнения этой процедуры следует учесть время формирования выходного сигнала ОЗУ. Для микросхем К561РУ2, на которых построено ОЗУ, время формирования выходного сигнала при считывании $t_{з/сч} \leqslant 1{,}2$ мкс для $U_{ип}=5$ В. Следовательно, считывание выходной информации можно осуществлять через время $t > t_{з/сч}$ после формирования сигнала

CE = 1. Учитывая, что каждая микрокоманда выполняется за один период задающего генератора, в данной команде организованы «холостые» микрокоманды 3 и 4 для формирования задержки считывания сигнала из ОЗУ. Микрокоманда 5 записывает выходной код ОЗУ в АС. Для этого используется инструкция LCM ЦПЭ, по которой выполняется процедура $\overline{M} \rightarrow$ AC. Для этой процедуры задаются следующие значения микроинструкций: $F = (7B)_{16}$; $C1,2 = (11)_2$; $\overline{K} = (F)_{16}$. Микроинструкции CE и R/W в микрокомандах 3; 4 и 5 соответствуют режиму считывания ОЗУ, так $CE = (1)_2$; $R/W = 0$.

В микрокоманде 6 содержимое АС переписывается в РС (R_8). Для этого нужна инструкция SDR ЦПЭ, в соответствии с которой выполняется процедура $AC - 1 + CI \rightarrow R_n$. Для того чтобы не изменить содержимое АС, необходимо переменной CI присвоить значение CI = 1, что выполняется при $C1,2 = (00)_2$ (аналогично микроинструкции C1,2 микрокоманды 1). Инструкция SDR будет выполняться при следующих значениях микроинструкций: $F = (2)_8(8)_{16}$; $K = (0)_{16}$; $C1,2 = (00)_2$·ОЗУ в микрокоманде 6 не участвует, поэтому микроинструкции CE и R/W принимают значения, соответствующие режиму считывания.

Микрокоманда 7 восстанавливает содержимое АС, которое на время выполнения команды было переписано в регистр Т. Микрокоманда 7 аналогична микрокоманде 2 и реализуется с помощью микроинструкций $F = (0)_8$, $(C)_{16}$; $\overline{K} = (F)_{16}$; $C1,2 = (11)_2$. Микроинструкция NA микрокоманды 7 совпадает с аналогичной микроинструкцией последних микрокоманд во всех командах: $NA = (1E)_{16}$ — и адресует управление к системной программе опроса режима работы микро-ЭВМ.

Адрес следующей команды будет определяться кодом РС (R8), в который была записана новая информация в результате выполненной команды G0 T0.

Рассмотрим трехбайтную команду условного перехода TZR2. Команда позволяет делать переход к команде, указанной во втором байте, если содержимое регистра R2 = 0. Если условие не соблюдается, то осуществляется переход к команде, адрес которой указан в третьем байте команды.

В микрокоманде 1 используется инструкция TZR2 ЦПЭ, по ней проверяют содержимое регистра R2 и формируют сигнал $C0 = CIVR_n$, который при R2 = 0 принимает значение C0 = 0, а при $R2 \neq 0$ — значение C0 = 1. Для

выполнения инструкции TZR2 в микрокоманде приняты следующие значения микроинструкций: $F=(52)_{16}$; $C1,2=(11)_2$; $\overline{K}=(0)_{16}$.

Значение сигнала C0 используется в микрокоманде 1 для формирования условного перехода. Для этого микроинструкция $CN=(011)_2$ формирует код, при котором выходной сигнал мультиплексора MS будет определяться сигналом C0, следовательно, сигнал C0 определяет значение адреса A0 следующей микрокоманды. Разряды A1 — A8 следующей микрокоманды определяются кодом $NA=(A1)_{16}$.

В зависимости от значения C0 после микрокоманды 1 будет исполняться либо 2-я (при $C0=0$), либо 3-я микрокоманда (при $C0=1$).

В микрокоманде 2 применяется «холостая» инструкция NOP, по которой содержимое регистра переписывается без изменения в свой же регистр. Операция NOP выполняется при $F=(6)_8(C)_{16}$; $\overline{K}=(F)_{16}$. В регистре PC(R8) сохраняется адрес следующего за исполняемой командой кода.

В микрокоманде 3 содержимое счетчика команд увеличивается на единицу с помощью операции INR ЦПЭ, по которой содержимое R_n при $CI=1$ увеличивается на единицу: $R_n+CI\rightarrow R_n$. Увеличение R8 на единицу выполняется при следующих значениях микроинструкций: $F=(3)_7(8)_{16}$; $C1,2=(00)_2$; $\overline{K}=(F)_{16}$.

Микроинструкции NA и CN в микрокомандах 2 и 3 совпадают и определяют адрес команды G0 T0; $NA=(0A)_{16}$; $CN=(001)_2$. Таким образом, в команде TZR2 используется как составная часть команда G0 T0, а рассмотренные три команды подготавливают, исходя из результатов сравнения, значение кода программного счетчика PC перед входом в программу G0 T0.

Рассмотрим команды, обеспечивающие обращение к стековой памяти CALL и RETURN. При выполнении команды CALL необходимо уменьшить на единицу содержимое указателя стека SP (R9) и по адресу, определяемому содержимым регистра R9, записать увеличенное на единицу содержимое PC(R8). Первая микрокоманда обеспечивает запись содержимого AC в регистр T аналогично микрокоманде 1 команды G0 T0.

В микрокоманде 2 используется инструкция DSM ЦПЭ для уменьшения содержимого R9 на единицу: $R_n-1+CI\rightarrow R_n$. Переменной CI микроинструкцией $C1,2=(11)_2$ присваивается нулевое значение. Инструкция DSM

выполняется при кодах микроинструкций $F=(1)_8(9)_{16}$; $C1,2=(11)_2$; $K=(0)_{16}$.

Микрокоманда 3 пересылает содержимое R9 в RA, для этого используется инструкция LMI, которая выполняется при следующих микроинструкциях: $F=(1)_8(9)_{16}$; $C1,2=(11)_2$; $\overline{K}=(F)_{16}$.

В микрокоманде 4 содержимое PC (R8) записывается в AC по инструкции ALR, а в микрокоманде 5 содержимое AC увеличивается на единицу по инструкции INA.

Микрокоманды 6, 7 и 8 организуют временну́ю задержку, определяемую динамикой работы ОЗУ. Микроинструкция R/W принимает значение «1», переводя ОЗУ в режим записи. После выполнения микрокоманды 8 в ячейку памяти с адресом, расположенным в RA, запишется содержимое AC, что соответствует записи в стековую память адреса команды, по которому будет осуществлен выход из подпрограммы.

В микрокоманде 8 формируется переход к микрокоманде 2 подпрограммы G0 T0, обеспечивающей передачу управления команде, указанной во втором байте команды CALL. Команда RETURN обеспечивает переход к основной программе после выполнения подпрограммы. Микрокоманда 1 записана по адресу $(A9)_{16}(1)_2$ и обеспечивает запись содержимого SP (R9) в регистр RA по инструкции LMI ЦПЭ, после чего микрокомандой 2 содержимое R9 увеличивается на единицу по инструкции INR. Микрокоманда 3 обеспечивает запись содержимого AC в регистр T по инструкции SDA и передает управление микрокоманде 3 программы G0 T0. При этом установленный в RA код указателя стека определяет адрес ячейки ОЗУ, где записан код очередной команды основной программы, из которой было произведено обращение к подпрограмме.

В изучаемой микро-ЭВМ разработка рабочих программ ведется на уровне команд. Записывают программы в ОЗУ с помощью управляющих программ, обращение к которым осуществляется с блока клавиатуры.

Система команд микро-ЭВМ. Система команд микро-ЭВМ предназначена для реализации на ее основе рабочих программ.

В систему команд входят следующие команды: арифметических и логических операций; переходов; работы с памятью; управления внешними устройствами; работы со стеком.

Команды арифметических и логических операций. *Команда* $0\rightarrow R_n$ очищает регистр, т. е. во

все разряды регистра заносятся логические 0. Шестнадцатеричный код этой команды и всех последующих приведены в табл. 9.6.

Команда $R_n + 1 \rightarrow R_n$ выполняет арифметическое сложение содержимого регистра с единицей и занесение полученного результата в тот же регистр.

Команда $R_n \rightarrow AC$ пересылает содержимое регистра в AC, при этом содержимое регистра R_n не изменяется.

Команда $AC \rightarrow R_n$ пересылает содержимое AC в регистр R_n, при этом содержимое AC не меняется.

Команда $R_n + AC \rightarrow R_n$, AC выполняет арифметическое сложение содержимого регистра R_n с содержимым AC, результат выполнения операции заносится в регистр R_n и в AC.

Таблица 9.6

Команда	Код	Команда	Код	Команда	Код
0→R0	B7	R0→AC	A8	R0+AC→R0, AC	77
0→R1	B8	R1→AC	A9	R1+AC→R1, AC	78
0→R2	B9	R2→AC	AA	R2+AC→R2, AC	79
0→R3	BA	R3→AC	AB	R3+AC→R3, AC	7A
0→R4	BB	R4→AC	AC	R4+AC→R4, AC	7B
0→R5	BC	R5→AC	AD	R5+AC→R5, AC	7C
0→R6	BD	R6→AC	AE	R6+AC→R6, AC	7D
0→R7	BE	R7→AC	AF	R7+AC→R7, AC	7E
0→AC	BF			AC+AC→AC	7F
R0+1→R0	C7	AC→R0	5A	R0→RA, AC→M	95
R1+1→R1	C8	AC→R1	5B	R1→RA, AC→M	96
R2+1→R2	C9	AC→R2	5C	R2→RA, AC→M	97
R3+1→R3	CA	AC→R3	5D	R3→RA, AC→M	98
R4+1→R4	CB	AC→R4	5E	R4→RA, AC→M	99
R5+1→R5	CC	AC→R5	5F	R5→RA, AC→M	9A
R6+1→R6	CD	AC→R6	60	R6→RA, AC→M	9B
R7+1→R7	CE	AC→R7	61	R7→RA, AC→M	9C
AC+1→AC	CF			AC→RA, AC→M	9D
R0−1→R0	66	$M_n \rightarrow AC$	9E	R0→RA, M→AC	E7
R1−1→R1	67	n	n	R1→RA, M→AC	E8
R2−1→R2	68	$AC \rightarrow M_n$	9F	R2→RA, M→AC	E9
R3−1→R3	69	n	n	R3→RA, M→AC	EA
R4−1→R4	6A			R4→RA, M→AC	EB
R5−1→R5	6B			R5→RA, M→AC	EC
R6−1→R6	6C	$\overline{AC} \rightarrow M_n$	72	R6→RA, M→AC	ED
R7−1→R7	6D	FF→AC	73	R7→RA, M→AC	EE
AC−1→AC	6E			AC→RA, M→AC	EF

Команда $R_n - 1 \rightarrow R_n$ выполняет арифметическую операцию вычитания единицы из содержимого регистра R_n; результат выполнения операции заносится в регистр R_n.

Команда $\overline{AC} \rightarrow AC$ осуществляет логическую операцию инверсии содержимого AC; результат выполнения операции заносится в AC.

Команда FF→AC заносит в AC шестнадцатеричный код FF, т. е. каждый разряд AC устанавливается в состояние «1».

Продолжение табл. 9.6.

Команда	Код	Команда	Код
TZR0	8A	ALR0	2D
TZR1	8B	ALR1	30
TZR2	8C	ALR2	33
TZR3	8D	ALR3	36
TZR4	8E	ALR4	39
TZR▱	8F	ALR5	3F
TZR6	90	ALR6	44
TZR7	91	ALR7	49
TZA	92	ALA	4C
FORMAT:		FORMAT:	
TZR_n A1, ifR_n=0 A2, if$R_n \neq 0$	XX A1 A2	ALR_n A1, ifC0=0 A2, ifC0=1	XX A1 A2
SRA0 (0→AC7, AC0→CO)	93	GOTO$_n$ *n*	0A *n*
SRA1 (1→AC7, AC0→C0)	16	CALL$_n$ *n*	65 *n*
FORMAT:		FORMAT:	
SRA_n A1, ifC0=0 A2, ifC0=1	XX A1 A2	RET END	A7 A0
JES A1, ifNOSIGN A2, ifSIGNAL	94 A1 A2	OVT$_n$ *n* IN$_n$ *n*	A4 *n* 4E *n*
JFF$_n$ *n* A1, ifRDY1=1 A2, ifRDY1=0	54 *n* A1 A2	MSC FRIQ TIME 00	75 n_1 n_2 00

Команда ALR_n трехбайтная складывает содержимое регистра R_n с содержимым АС и записывает результат в АС. После операции сложения в команде ALR_n реализуется условный переход по значению С0. При $C0=0$ управление передается команде с адресом, записанным во втором байте команды ALR_n при $C0=1$, адрес следующей команды определяется содержимым третьего байта команды ALR_n.

Команда ALA, аналогичная ALR_n, осуществляет удвоение содержимого АС, что эквивалентно сдвигу содержимого АС на один разряд влево.

Команды переходов. *Команда безусловного перехода* $GOTO_n$ определяет передачу управления к оператору программы с номером *n*. Команда двухбайтная; первым байтом является код команды, вторым — адрес (или номер оператора программы, которому необходимо передать управление):

1-й байт	код команды
2-й байт	адрес *n*

Команда условного перехода TZRN передает управление по условию: логические нули во всех разрядах регистра RN. Команда трехбайтная; первым байтом является код команды, вторым — адрес перехода, если условие выполняется, третьим — адрес перехода, если условие не выполняется:

1-й байт	код команды	
2-й байт	адрес A1	$R_n=0$
3-й байт	адрес A2	$R_n\neq 0$

Команда условного перехода SRA сдвигает содержимое АС вправо (т. е. в сторону младших разрядов) на один разряд и осуществляет условный переход в зависимости от содержимого младшего разряда АС.

Команда трехбайтная: первым байтом является код команды, вторым — адрес перехода, если младший разряд АС был равен «0», третьим — адрес перехода, если младший разряд АС был равен «1». Если используется команда SRA0, то в старший разряд АС заносится «0», если SRA1, то «1».

1-й байт	код команды	
2-й байт	адрес A0	AC0=0
3-й байт	адрес A1	AC0=1

Команда END является командой конца программы. По этой команде осуществляется переход к программе опроса управляющей клавиатуры, т. е. ОСТАНОВ.

Команда условного перехода JFS реализует условный переход в зависимости от сигнала RDY0, который является ответным сигналом платы звуковой сигнализации. В момент передачи звукового сигнала он имеет значение «0»; в режиме молчания — «1». Следует различать режим генерации паузы и режим молчания. Режим генерации пауз такой же активный режим, как и режим генерации звукового сигнала. Пауза (код паузы FF) представляет собой сигнал высокой частоты (около 30 кГц), не воспринимаемый органами слуха человека, поэтому он воспринимается как перерыв в звучании.

Структура команды JFS такая же, как и у команд TZR_n, SRA. Первым байтом команды служит код команды, вторым — адрес перехода для случая RDY0 = 1, третьим — адрес перехода для случая RDY0 = 0:

1-й байт	код команды	
2-й байт	адрес A0	молчание RDY0 = 1
3-й байт	адрес A1	сигнал RDY0 = 0

Команды работы с памятью. Команда $R_n \rightarrow RA$, $AC \rightarrow M$ записывает в память содержимое AC по адресу, хранящемуся в регистре R_n.

Команда $R_n \rightarrow RA$, $M \rightarrow AC$ считывает содержимое памяти в AC по адресу, хранящемуся в регистре R_n.

Команда $AC \rightarrow M_n$ обеспечивает запись в память содержимого AC по адресу, указанному в следующем байте:

1-й байт	код команды
2-й байт	адрес *n*

Команда $M_n \rightarrow AC$ считывает содержимое памяти в AC по адресу, указанному в следующем байте:

1-й байт	код команды
2-й байт	адрес *n*

Команды управления внешними устройствами. Команда вывода информации OUTn пересылает информацию из AC в регистр внешнего устройства, которому присвоен адрес *n*. Команда двухбайтная: первым байтом является код команды, вторым — адрес внешнего устройства:

1-й байт	код команды
2-й байт	*n* — адрес внешнего устройства

Адреса устройств ввода — вывода:

00 — счетчик длительности платы звуковой индикации;

01 — первый индикатор дисплея (счет ведется справа налево);

02 — второй индикатор;
03 — третий индикатор;
04 — четвертый индикатор;
05 — буферный регистр платы звуковой индикации;
06 — регистр внешнего признака (кнопка платы световой сигнализации);
07 — регистр переключателей платы световой сигнализации;
08 — регистр точечных светодиодов платы световой сигнализации.

Команда ввода IN_n пересылает информации из регистра внешнего устройства с адресом n в АС. Структура команды аналогична команде $0UT_n$.

Команда условного перехода по внешнему признаку JFF_n выполняет условный переход по сигналу RDY1. Этот сигнал формирует устройство, адрес которого задан во втором байте. Команда четырехбайтная:

1-й байт	код команды
2-й байт	адрес устройства
3-й байт	адрес перехода, если RDY1=1
4-й байт	адрес перехода, если RDY1=0

Команда MSC выдает связный звуковой текст (музыки). Организация команды MSC следующая:

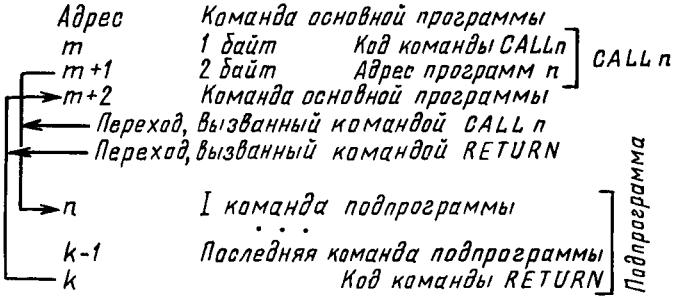
1-й байт	код команды
2-й-байт	код 1-й ноты
3-й байт	длительность 1-й ноты
4-й байт	код ноты
5-й байт	длительность 2-й ноты
$(2n)$-й байт	код m-й ноты
$(2n+1)$-й байт	длительность m-й ноты
$(2n+2)$-й байт	код конца 00000000

По этой команде коды нот и длительностей будут последовательно выдаваться на плату звуковой индикации, где они будут воспроизводиться в виде звуковых сигналов соответствующего тона и длительности. Конец музыки определяется кодом конца (0 во всех разрядах). По окончании музыки осуществляется безусловный переход к выполнению следующей команды программы.

Команды работы со стеком. Команда $CALL_n$ осуществляет переход к программе, начинающейся с адреса n, и предусматривает возвращение к основной программе после выполнения подпрограммы. Команда двухбайтная. Для организации перехода к основной программе содержимое регистра R9(SP) уменьшается на единицу и задает адрес ОЗУ. По указанному адресу

записывается номер команды, к которой следует перейти после выполнения подпрограммы. Адрес возврата определяется как $m+2$ (здесь m — адрес первого байта команды).

Команда RETURN используется для выхода из подпрограмм по содержимому стековой памяти. При выполнении этой команды в программный счетчик PC записывается код адреса, записанного при последнем обращении к стековой памяти в ячейку ОЗУ с адресом, хранящимся в R9(SP). После выполнения команды RETURN содержимое регистра R9 увеличивается на единицу и управление передается команде, адрес которой записан в PC. Команда однобайтная.



Количество вложений программ определяется областью ОЗУ, отведенной под стековую память.

§ 9.3. Работа микро-ЭВМ с внешними устройствами

Обмен информацией между микро-ЭВМ осуществляется по магистралям A, M и D, при этом в магистрали A формируется адресный код внешнего устройства, а в магистралях M и D — передаваемая информация. Для связи микро-ЭВМ с внешними устройствами используются команды OUTn и TNn. Такая процедура обмена информацией требует включения во внешние устройства схем, дешифрирующих присвоенный им адресный код и обеспечивающих обмен информацией по магистрали.

В качестве внешних устройств в микро-ЭВМ применяют платы световой индикации, звуковой сигнализации, плату таймера и любые другие внешние устройства, подключаемые к микро-ЭВМ через специальный разъем и выполненные в соответствии с требованиями по сопряжению магистралей.

Плата световой индикации. Схема платы световой индикации представлена на рис. 9.15, *а*. Четыре семисегментных индикатора отражают шестнадцатеричные коды магистралей A и D, а четыре других индикатора отражают информацию, записанную в регистрах RG1—RG4. Точечные светодиоды D1 отражают двоичный код, записанный в регистре RG5. Светодиоды D2 отражают двоичный код группы переключателей, формирующих входную информацию для регистра RG6, который поступает в магистраль B процессора. При наличии микроинструкции IOR, формируемой в командах IN и OUT, процессор обращается

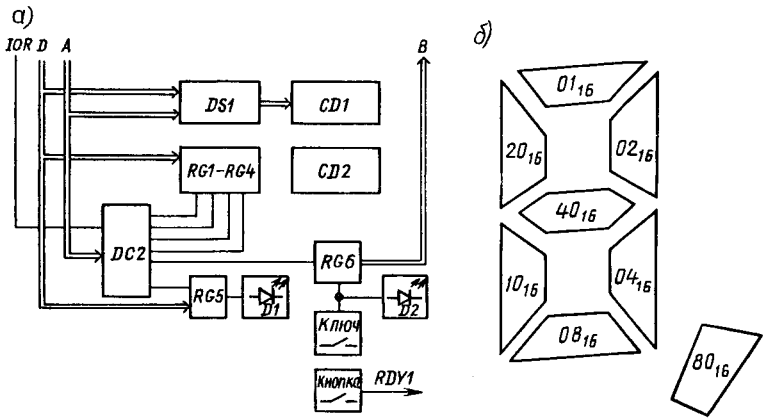


Рис. 9.15. Схема платы световой индикации (*а*) и кодировка ее сегментов (*б*)

к внешним устройствам. Узел, которому передается информация по магистрали, определяется дешифратором DC2 в соответствии с кодом магистрали A. Адресные коды внешних устройств определены в описании команды OUT в § 9.2. Дешифратор ДШ2 реализован на микросхемах 155ИД3, регистр RG5 — на микросхемах К155ИР1, регистр RG6 — на микросхеме К589ИР12, входящей в МПК серии 589. Двоичный код на входах регистра RG6 задается переключателями и отображается светодиодами красного цвета D2. Информацию, занесенную в регистр RG5, отображают светодиоды зеленого цвета D1. Регистры RG1—RG4 реализованы на микросхемах К589ИР12, их состояние отображается на семисегментных индикаторах

АЛС325Г. Для уменьшения числа резисторов используется схема динамической индикации. Для этого все выходы регистров объединены, и к ним через резисторы подключены объединенные катоды индикаторов. Устройство управления (УУ), реализованное на счетчике К155ИЕ5 (D10) и дешифраторе К155ИД4 (D11) и являющееся распределителем импульсов, вырабатывает сигналы управления, под действием которых происходит последовательная выдача информации с регистров на общую магистраль. Одновременно сигналы УУ поступают на транзисторные ключи, которые коммутируют соответствующие общие аноды индикаторов, тем самым производится последовательное высвечивание информации. Так как УУ синхронизируется тактовыми импульсами микроЭВМ, следующими с большой частотой, то мерцания индикаторов совершенно не заметны. Каждому сегменту индикатора соответствует один разряд регистра, поэтому путем задания определенной кодовой комбинации можно высветить на индикаторе произвольный символ (рис. 9.15, *б*). На рис. 9.15 цифры обозначают код, который необходимо сформировать в АС для включения сегмента. Код символа из нескольких сегментов равен арифметической сумме кодов отдельных сегментов (например, код символа $L=38_{16}$). Для шестнадцатеричного отображения информации на магистралях A и D служат семисегментные индикаторы CD1. Дешифрация двоичного кода в код семисегментного индикатора производится дешифратором, реализованным на ППЗУ К155РЕ3. Здесь также используется режим динамической индикации, при этом разряды магистралей A и D поступают на входы мультиплексоров К155КП2, чем обеспечивается поочередная подача информации этих шин на входы дешифратора.

Для выполнения условного перехода по внешнему признаку (команда JFF) введена кнопка, при опросе которой (ее код 06) на шину RDY1 выдается сигнал «0», если кнопка нажата, и «1», если отпущена.

Плата звуковой сигнализации. Схема платы звуковой сигнализации представлена на рис. 9.16. Счетчик *СТ1* вместе с компаратором СА образует делитель частоты с переменным коэффициентом деления. Коэффициент деления определяется кодом, хранимым в регистре RG, при этом коэффициент деления изменяется от 2 до 256. На вход делителя поступают импульсы с генератора *ГИ1* с частотой около 30 кГц, таким образом, на входе делителя формируются импульсы с частотой от 100 Гц до 15 кГц.

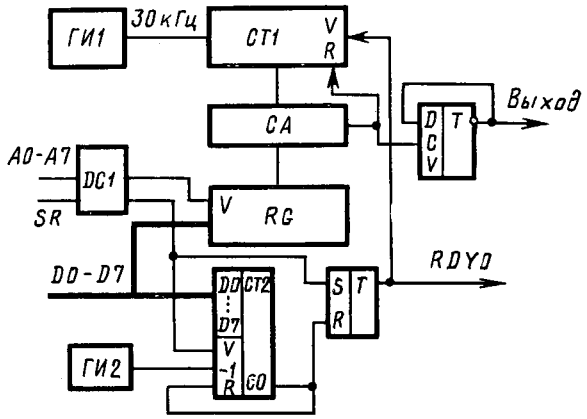


Рис. 9.16. Схема платы звуковой сигнализации

Ниже приведена таблица кодов музыкальных нот:

1. Коды нот

Ноты		Коды октав			
		первая	вторая	третья	четвертая
До	C	OD	86	C3	E1
До-диез	C#	1B	8D	C6	E2
Ре	D	28	93	C9	E4
Ре-диез	D#	34	99	CC	E6
Ми	E	3F	9F	CF	E7
Фа	F	4A	A4	D2	E8
Фа-диез	F#	54	AA	D4	EA
Соль	G	5E	AE	D7	EB
Соль-диез	G#	67	B3	D9	EC
Ля	A	6F	B7	DB	ED
Ля-диез	B#	77	BB	DD	EE
Си	H	7F	BF	DF	EF

2. Коды длительностей нот от 00_{16} до $0F_{16}$ эквивалентны абсолютной длительности.

3. Код паузы FF, длительность паузы устанавливается так же, как и длительность ноты.

4. Запись музыкальных текстов будет производиться в символах нот с верхней индексацией октавы и с обозначе-

нием абсолютной длительности в скобках. (Например, $G^3\#$ (4)-соль-диез третьей октавы с длительностью 04_{15}. Пауза обозначается P).

5. Музыкальные тексты.

5.1 $[G^1(2), A^2(4), G^2(2), A^2(4), E^2(4), D^2(F), P(1), D^2(2), E^2(4), D^2(2), E^2(4), G^2\#(4), A^2(F)]$ - *2 раза*
$A^2(1), P(1), A^2(4), P(2), A^2(4), G^2(2), A^2(2), E^2(1), P(1), E^1(4), P(2), E^2(4), D^2(2), E^2(2), A^1(1), P(1), A^1(4), P(2), C^2(4), A^2(4), G^2(7), P(1), G^2(2), A^2(2), G^2(4), A^2(4), G^2(2), A^2(6), D^2(1), P(1), D^2(6), P(1), D^2(8), G^2(1), A^2(1), B^2(1), P^2(2), P(2), G^2(1), A^2(1), B^2(2), G^2(1), P(1), G^2(1), A^2(1), B^2(2), G^2(14), P(2), G^2(1), A^2(1), B^2(2), C^2(2), P(2), G^2(1), A^2(1), B^2(2), G^2(1), P(1), G^2(1), A^2(1), B^2(2), G(F).$

5.2 $C^1(4), D^1\#(4), G^1(4), D^1\#(4), F^1(8), D^1\#(4), D^1(4), G^1(8), F^1(8), C^1(F), D^1\#(4), G^1(4), A^1\#(3), P(1), A^1\#(3), P(1), C^2(8), A^1\#(4), G^1\#(4), G^1(F), [A^1(8), B^1(8), D^2\#(4), C^2(4), G^1(12), D^1\#(4), D^1(4), C^1(4), G^1(4), F^1(4), G^1\#(F), A^1\#(4), G^1\#(4), G^1(8), F^1(4), D^1\#(4), G^1(8), F^1(8), C^1(16)]$ - *2 раза*

Код ноты загружается в регистр с магистрали D микро-ЭВМ при наличии соответствующего кода (05) на магистрали A и инструкции SR. Управляет записью дешифратор DC. Длительность звуковой посылки определяет второй делитель частоты с переменным коэффициентом деления на основе счетчика с предварительной установкой *CT2*. При наличии кода 00 на магистрали A и инструкции SR в счетчик заносится код длительности, одновременно устанавливается в «1» триггер *T*. Единичное состояние триггера разрешает работу счетчика *CT1*, т. е. вырабатывает звуковой сигнал. Затем счетчик *CT2* производит досчет занесенного кода до 15 с частотой импульсов, вырабатываемых генератором ГИ2 (около 0,1 Гц). Выработанный счетчиком сигнал переноса сбрасывает триггер *T* в «0», тем самым запрещает выдачу звукового сигнала. Выход триггера *T* является также сигналом RDY0 для микро-ЭВМ. Генераторы *ГИ1* и *ГИ2* реализованы на микросхемах 564ЛН2

Плата таймера. Плата таймера (рис. 9.17) используется в микро-ЭВМ для формирования временных интервалов произвольной длительности. Импульсы с тактового генератора микро-ЭВМ 2-МГц поступают на вход делителя, формирующего импульсы с частотой следования 1000, 100, 10, и 1 Гц — на четыре входа мультиплексора MS. По информационной магистрали D микро-ЭВМ

передает код, записываемый в регистр RG и определяющий через входы управления мультиплексором частоту импульсов на его выходе. Выходной сигнал мультиплексора устанавливает в единичное состояние триггер *T*, который своим выходным сигналом формирует управляющий сигнал RDY1. Установка в нулевое состояние триггера *T* осуществляется выходным сигналом дешифратора DC, который формируется кодом на адресной магистрали.

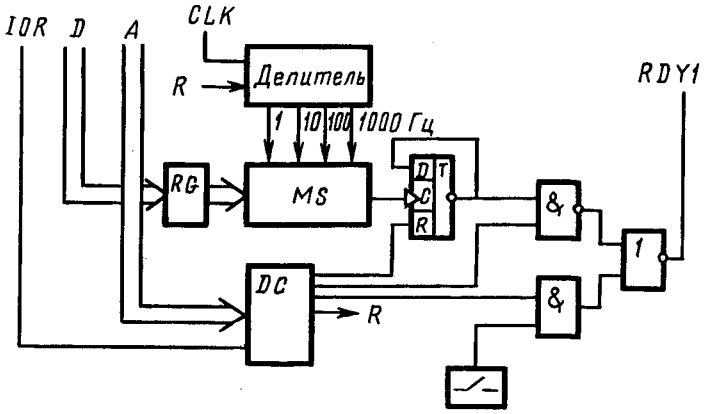


Рис. 9.17. Схема платы таймера

Плата таймера позволяет формировать сигнал RDY1 от внешних контактирующих устройств и устанавливать момент отсчета временны́х интервалов сбросом делителя в ноль.

§ 9.4. Лабораторные работы

В данном разделе приводятся задания для практического освоения микро-ЭВМ МП589 и использования ее для выполнения расчетов и управления внешними устройствами.

Задание режимов работы микро-ЭВМ и ввод исходных данных осуществляются с клавиатуры микро-ЭВМ. Исходные данные и рабочая программа вводятся в шестнадцатеричных кодах. Коды, передаваемые по магистралям D, A и M, индицируются точечными светодиодами,

при этом младшие разряды находятся справа. Для получения десятичного эквивалента двоичного кода необходимо просуммировать веса соответствующих разрядов, например состояние магистрали

128	64	32	16	8	4	2	1
●	0	●	0	0	●	●	●

определяет число $128+32+4+2+1=167$ (где 0 — выключенный светодиод).

При кодировании чисел в шестнадцатеричной системе счисления старшие четыре светодиода определяют код старшего разряда числа в шестнадцатеричном коде, младшие четыре светодиода — код младшего разряда числа. Перевод чисел из двоичного кода в шестнадцатеричный и обратно осуществляется в соответствии с табл. 9.7.

Таблица 9.7

Системы счисления					
Шестнадцатеричная	Двоичная	Десятичная	Шестнадцатеричная	Двоичная	Десятичная
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	A	1010	10
3	0011	3	Б	1011	11
4	0100	4	C	1100	12
5	0101	5	D	1101	13
6	0110	6	E	1110	14
7	0111	7	F	1111	15

Для перевода шестнадцатеричных кодов в десятичный необходимо умножить десятичное значение кода старшего разряда исходного числа на шестнадцать и сложить с десятичным значением младшего разряда переводимого числа.

Например: $01_{16}=0\cdot16+1=1_{10}$; $10_{16}=1\cdot16+0=16_{10}$; $21_{16}=2\cdot16+1=33_{10}$; $3F=3\cdot16+15=63_{10}$.

Лабораторная работа 9.1. Ввод исходных данных и рабочей программы в микро-ЭВМ. Выполнение простых программ

Подготовка микро-ЭВМ к работе. Перевести тумблер включения сетевого питания в положение *Сеть*. При этом загорится светодиодный индикатор *Сеть* и появится звуковой сигнал нарастающего тока. После окончания

звукового сигнала на индикаторах данных, адреса и памяти появятся нули (несвечение индикаторов). Микро-ЭВМ готова к работе.

Краткие сведения из теории

Функционирование системных программ. Начальная установка. Нажать кнопку R. Убедиться, что нажатие сопровождается звуковым сигналом нарастающего тона, а индикаторы данных, адреса и памяти отображают нулевые коды. В результате выполнения программы R обнуляется программный счетчик PC и устанавливается указатель стека SP в начальное положение. При этом изменяется содержимое нулевой и пятой ячеек ОЗУ, поэтому рабочие программы рекомендуется заносить в память микро-ЭВМ начиная с адреса 10_{16}.

Загрузка аккумулятора. Нажать кнопку LA. Услышав звуковой сигнал низкого тона, нажать последовательно две произвольные кнопки на клавиатуре данных. Каждое нажатие кнопки должно сопровождаться звуковым сигналом (первое нажатие — сигнал низкого тона, второе — сигнал высокого тона). Убедиться, что код на индикаторе данных соответствует набранному на клавиатуре.

Установка адреса. Нажать кнопку SA. Услышав звуковой сигнал низкого тона, набрать на клавиатуре данных произвольный код. Убедиться, что нажатие кнопок клавиатуры данных сопровождается звуковыми сигналами. Проверить код на индикаторе адреса. Убедиться, что он соответствует набранному на клавиатуре.

Запись с увеличением. Нажать кнопку WI. Услышав звуковой сигнал низкого тона, набрать на клавиатуре данных произвольный код с помощью последовательного нажатия кнопок клавиатуры данных. Убедиться, что каждое нажатие кнопки сопровождается звуковым сигналом, а после второго нажатия значение индикатора адреса увеличивается на единицу. Повторить процедуру записи в следующий адрес ЗУ.

Считывание с уменьшением. Нажать кнопку RD. Убедиться, что нажатие сопровождается звуковым сигналом и что значение индикатора адреса уменьшается на единицу. При этом на индикаторе памяти высвечивается код, записанный в ячейку ОЗУ при выполнении предыдущего пункта. Повторно нажать кнопку RD и с помощью индикатора памяти просмотреть содержимое

ячеек ОЗУ, сформированное при выполнении режима «Запись с увеличением».

Считывание с увеличением. Нажать кнопку RI. Убедиться, что нажатие сопровождается звуковым сигналом и индикатор адреса указывает на увеличение кода адреса, а индикатор памяти отражает содержимое соответствующей ячейки ОЗУ. Последовательно просмотреть содержимое ячеек ОЗУ, сформированное при выполнении процедуры «Запись с увеличением».

Загрузка активной программы. Установить начальный адрес программы, для этого нажать кнопку SA и набрать на клавиатуре данных начальный адрес программы.

Загрузить последовательно все команды программы, используя режим WI. Пример программы, обеспечивающей увеличение содержимого АС на единицу, представлен в виде табл. 9.8.

Таблица 9.8

Адрес ОЗУ	Команда	Код команды	Комментарий
11	0→AC	BF	Обнуление содержимого АС
12	AC+1→AC	CF	Увеличение на единицу содержимого АС
13	G0 T0	0A	Безусловный переход к команде, записанной в ячейке 12 ОЗУ
14	12	12	

Используя режим WI, ввести коды команд в ячейки ОЗУ и с помощью системных команд RI, RD проверить соответствие введенных кодов исходной программе.

Выполнение рабочей программы. С помощью кнопки SA установить начальный адрес программы (в рассматриваемом примере — 11). Исполнение рабочей программы возможно в пошаговом RS и циклическом RC режимах.

1. Пошаговый режим выполняется нажатием кнопки RS. Каждое нажатие приводит к выполнению одной команды. По состоянию светодиодов индикаторов убедиться в последовательном исполнении программы.

2. Циклический режим выполняется нажатием кнопки RC. При этом команды идут последовательно автоматически. Выход микро-ЭВМ из режима циклического исполнения программы обеспечивается командой END, завершающей рабочую программу, или нажатием одной из кнопок клавиатуры режима.

Для того чтобы снова запустить программу после сброса или окончания и для вызова другой программы, необходимо загрузить регистр адреса начальным адресом программы и перейти к ее выполнению.

Проверка и изменение содержимого регистров ЦПЭ. Пользователю предоставляется 8 регистров ЦПЭ (R0—R7). Каждому регистру пользователя соответствует ячейка памяти микро-ЭВМ (см. рис. 9.5). При переходе микро-ЭВМ в режим ожидания, т. е. после выполнения очередной команды в шаговом режиме RS или команды END в циклическом режиме RC, информация из внутренних регистров ЦПЭ переписывается в ОЗУ (см. описание операционной системы, рис. 9.10).

Для проверки содержимого внутреннего регистра необходимо загрузить в регистр адреса адрес ячейки ОЗУ, соответствующей выбранному регистру (см. рис. 9.5), и считать информацию с индикатора памяти. Изменение содержимого данной ячейки памяти приводит к изменению содержимого выбранного регистра. Этим способом рекомендуется пользоваться при отладке программ.

Задания для домашней подготовки

1. Разработать программу А, обеспечивающую запись во внутренние регистры ЦП R0—R7 последовательности чисел 0—7 и запись в АС числа 8. Проверить правильность введенной программы. Исполнить введенную программу по командам (в режиме RS).

Пользуясь картой памяти микро-ЭВМ, убедиться в правильности выполненной программы.

Разработать программу Б, обеспечивающую запись содержимого внутренних регистров R0—R7 и АС в ячейки памяти соответственно по адресам $40—48_{16}$. Программу разместить в памяти микро-ЭВМ таким образом, чтобы возможно было самостоятельное выполнение программ А и Б. В режиме RC выполнить последовательно программу А, а затем Б. Убедиться в правильности выполнения программ, просмотрев содержимое ячеек памяти с адресами $40—48_{16}$.

В режиме RC выполнить программу А. Изменить содержимое ячеек памяти, в которые записывается содержимое регистров R0—R7 и АС, после чего выполнить программу Б, проверить содержимое ячеек ОЗУ с адресами $40—48_{16}$.

Используя команду безусловного перехода, объеди-

нить программы А и Б в одну программу. Убедиться в правильности ее исполнения.

2. Составить программу, обеспечивающую формирование кода FF в AC, кода FE в R7, кода FD в R6 и запись содержимого регистров AC, R7, R6 в ячейки ОЗУ соответственно с адресами 30, 31, 32_{16}.

3. Составить программу, обеспечивающую запись в ОЗУ начиная с адреса 30_{16}, последовательности чисел с 5 до 15. Проверить правильность выполнения программы с помощью системных команд и с помощью контрольной программы.

4. Составить программу, обеспечивающую при пошаговом исполнении циклический сдвиг одного включенного светодиода на индикаторе данных.

Работа на учебной ЭВМ

1. Ввести подготовленные рабочие программы в ОЗУ. **2.** Проверить правильность введенной информации. **3.** Выполнить программы в режиме RS. Фиксируя и анализируя промежуточные результаты с помощью индикаторов магистралей, проверить изменение содержимого внутренних регистров ЦПЭ. **4.** Введенные программы выполнить в режиме RC. 5. Убедиться в правильности выполнения программ.

Содержание отчета

Отчет должен содержать: **1.** Схему выполненных заданий. **2.** Рабочие программы выполненных заданий, комментарии к изменению содержимого внутренних регистров ЦПЭ в процессе выполнения программы.

Задания для самопроверки

1. По каким магистралям возможен ввод и вывод информации в ЦПЭ К589 ИК02?

2. Нарисуйте структурную схему 6-разрядного процессора на базе ЦПЭ К589 ИК02 со схемой ускоренного переноса СУП и без нее.

3. Каково назначение выходной информации X, Y ЦПЭ?

4. Каким образом формируется двоичный код F0—F6 на входе ЦПЭ для выполнения конкретной микрокоманды (например, AC+R2+I→ $\rightarrow R_2$, AC; AC+I→AC; T+I→AC; R4+I→R4, AC; M→AC; M$\vee$AS→ →AC)?

5. Какой информации соответствует сигнал C0 при сложении двух чисел?

6. Какое устройство определяет адрес выполняемой микрокоманды?

7. Какими схемами формируется код выполняемой микрокоманды? Какова ее разрядность? Что необходимо сделать, чтобы увеличить разрядность микрокоманды?

8. Нарисуйте временну́ю диаграмму сигналов шины A при формировании шестнадцатеричных кодов.

9. Какие схемы могут формировать адрес следующей микрокоманды?

10. Каким образом задается источник формирования следующего адреса?

11. Какие ограничения в рассматриваемой микро-ЭВМ существуют при формировании адресов в командах условных переходов?

12. Каково назначение мультиплексора MS? Какими сигналами определяется его выходно́й код?

13. Назовите четыре микроинструкции, в которых используется сигнал CI.

14. Каким образом можно сформировать CI=0; CI=1?

15. Определите назначение задающего генератора ГСИ. Что изменится в работе микро-ЭВМ, если частота тактовых импульсов СИ удвоится? если СИ перестанут поступать на вход RG_a?

16. С какой целью и какими сигналами выход регистра RGK переводится в третье состояние?

17. Что произойдет с устройством управления, если исключить из него RG_a?

18. Каким образом информацию с магистрали K можно записать в ОЗУ? Какими микроинструкциями?

19. Что такое операционная система микро-ЭВМ и как она реализуется?

20. Определить назначение программного счетчика.

21. Каково назначение указателя стека? Как изменится содержимое указателя стека при последовательном обращении к трем подпрограммам?

22. Чем определяются ограничения по обращению к подпрограммам?

23. Укажите типы сигналов, формируемых клавиатурой. С помощью каких команд можно ознакомиться с содержимым ОЗУ?

24. Определите назначение подпрограммы временно́й задержки при считывании сигнала клавиатуры. Чем определяется длительность задержки?

25. Назовите последовательность формирования шестнадцатеричных кодов. Что произойдет при одновременном нажатии двух цифровых кнопок?

Лабораторная работа 9.2. Управление микро-ЭВМ на микрокомандном уровне. Составление команд

Цель работы: *изучить назначение микроинструкций и работу устройства управления. Ознакомить с последовательностью составления команд из микрокоманд.*

Краткие сведения из теории изложены в разделах 9.5 и 9.7.

Задания для домашней подготовки

Схема УУ для микро-ЭВМ соответствует рис. 9.6.

1. Составить на микрокомандном уровне команду R2+ +AC→R2, AC, по которой суммируются содержимое регистров R2 и AC, а результат записать в R2 и AC.

2. Составить микрокоманды, обеспечивающие выполнение следующих команд: 1) 0→R7; 2) R2→R2; 3) R4 + 1→→R4; 4) R5 — 1→R5; 5) R3→AC; 6) AC→R2. С помощью логического анализатора записать и провести анализ перечисленных команд лабораторной микро-ЭВМ.

3. Составить команду, обеспечивающую сложение содержимых регистров R0 и R1 с записью результата в AC.

4. Составить команду условного перехода по сигналу внешнего устройства (команда JFF).

5. Составить команду, обеспечивающую считывание информации с магистрали B и запись ее в регистр R2.

6. Составить команду записи содержимого AC в ячейку памяти, адрес которой определяется вторым байтом команды (команда AC→M_n).

7. Составить команду записи содержимого ячейки памяти с адресом, указанным во втором байте команды, в AC (команды M_n→AC).

Работа на учебной ЭВМ

1. Составить программу с циклическим исполнением команды, разработанной в процессе домашней подготовки. Записать программу в ОЗУ. Обеспечить циклическое исполнение программы.

2. Входы 1—8 логического анализатора подключить к контактам 09, 10, 11, 13, 14, 15, 16, 17 микросхемы D25, входы анализатора 9—16 — к таким же контактам микросхемы D28. Входы анализатора 17—25 подключить к контактам 08, 07, 06, 05, 04, 03, 02, 01 23 микросхемы D25. При таком подключении входы 1—16 отражают часть микрокоманды, записанной в ПЗУ D28, D25, а входы 17—25 — адрес микрокоманды (см. рис. 9.6). Установить на анализаторе запускающее слово, соответствующее коду исследуемой команды (использовать описание прибора). Перевести микро-ЭВМ в режим циклического исполнения программы. Установить на анализаторе циклический режим и нажать кнопку «Запуск». На экране анализатора отобразится фрагмент микропрограммы. Зафиксировать полученные диаграммы.

3. Переключить входы анализатора 1—16 на контакты 09, 10, 11, 13, 14, 15, 16, 17 микросхем D26, D27. Повторить операции, позволяющие фиксировать вторую часть микрокоманды, записанной в ПЗУ D26, D27. Полученную микропрограмму представить в виде таблицы, приведенной в описании микро-ЭВМ, перекодировав двоичные коды,

где это необходимо, в шестнадцатеричные и восьмеричные. Объясните роль и назначение микроинструкций в записанных микрокомандах.

Содержание отчета

Отчет должен содержать: 1. Программу, обеспечивающую выполнение заданной команды (выполняется при домашней подготовке). 2. Зафиксированную с помощью логического анализатора последовательность микрокоманд, обеспечивающую выполнение команды, подготовленной в домашней работе. Программы отчета должны содержать комментарии, поясняющие назначение выполняемых микрокоманд и управляющих микроинструкций.

Задания для самопроверки

1. Назовите обязательные типы микроинструкций для микропроцессоров серии 589.
2. Каково назначение микроинструкции K? Каким образом ввести в AC код 20_{16}?
3. Каково назначение микроинструкции W/R?
4. Каково назначение микроинструкции CN? Чем определяется ее разрядность? Каким образом формируется код A0=0; A0=1; A0=C0?
5. Какие микроинструкции определяют источник следующего адреса?
6. В какой ситуации микроинструкция NA может принимать произвольное значение?
7. Каким образом задается код команды?
8. Какова последовательность опроса БПП?
9. Составьте команду сложения содержимого двух внутренних регистров R_n и R_{n+1}, задав код команды кодом IE_{16}.
10. Составьте команду вычитания содержимого регистра R_n из регистра R_{n+1} с записью результата в регистр R_n. Код команды EE_{16}.
11. Составьте команду условного перехода по входному сигналу внешнего источника, подаваемому на 5-й вход мультиплексора MS. При единичном сигнале управление должно передаваться ячейке с адресом 15_{16}, при нулевом — 16_{16}. Код команды $A1_{16}$.

Лабораторная работа 9.3. Программы с условными переходами. Подпрограммы, работа со стековой памятью

Цель работы: *ознакомить с организацией условных переходов в программах с особенностями использования подпрограмм.*

Краткие сведения из теории

Условные переходы в лабораторном стенде МП 589 организованы на основе мультиплексора MS УУ, формирующего младший адрес микрокоманды (см. рис. 9.6). Код

младшего адреса микрокоманды может формироваться одним из восьми возможных устройств, выбираемых микроинструкцией CN. Команды передачи управления используют возможности УУ изменять номер выполняемой команды в зависимости от одного из анализируемых сигналов или заданного адреса. В качестве примера применения подпрограмм и организации условных переходов рассмотрим программу сложения двух чисел в дополнительном коде. Слагаемые в прямых кодах записаны в ячейки ЗУ с адресами 01 и 02, результат вычислений записать по адресу 03.

При записи чисел в прямом коде старший разряд отводится под запись знака числа. Ноль в старшем разряде соответствует положительным числам, единица — отрицательным. Для выполнения задания необходимо организовать преобразование чисел из прямого кода в дополнительный в виде специальной подпрограммы.

На рис. 9.18 подпрограмма выделена в виде самостоятельного блока программы. Начинается подпрограмма с команды 2, обеспечивающей сдвиг содержимого АС влево на один разряд для проведения анализа знакового разряда. Допустим, что команда 1 подпрограммы записана по адресу 30_{16}.

Команда 2 обеспечивает условный переход к команде 3 при равенстве нулю знакового разряда слагаемого и к команде 4 при равенстве единице знакового разряда. Команда 3 восстанавливает содержимое АС, так как для положительных чисел прямой и дополнительный коды чисел совпадают. Для преобразования отрицательных чисел в дополнительный код в команде 4 мантисса сдвигается на один разряд вправо и в знаковый разряд записывается код единицы, а в команде 5 формируется инверсия содержимого АС. После выполнения этой операции в АС записан обратный код числа. Для получения дополнительного кода к содержимому АС в команде 6 добавляется единица. Завершается подпрограмма командой RETURN, по которой организуются обращение к указателю стека и передача управления команде, записанной при последнем обращении к стеку.

В основной программе командой 1 в АС считывается содержимое ячейки 01 ЗУ. Команда 2 организует обращение к подпрограмме с адресом 30, в которой формируется дополнительный код числа, записанного в АС. Команда 3 переписывает дополнительный код второго слагаемого из АС в регистр R0. Команды 4 и 5 считывают из ЗУ второе

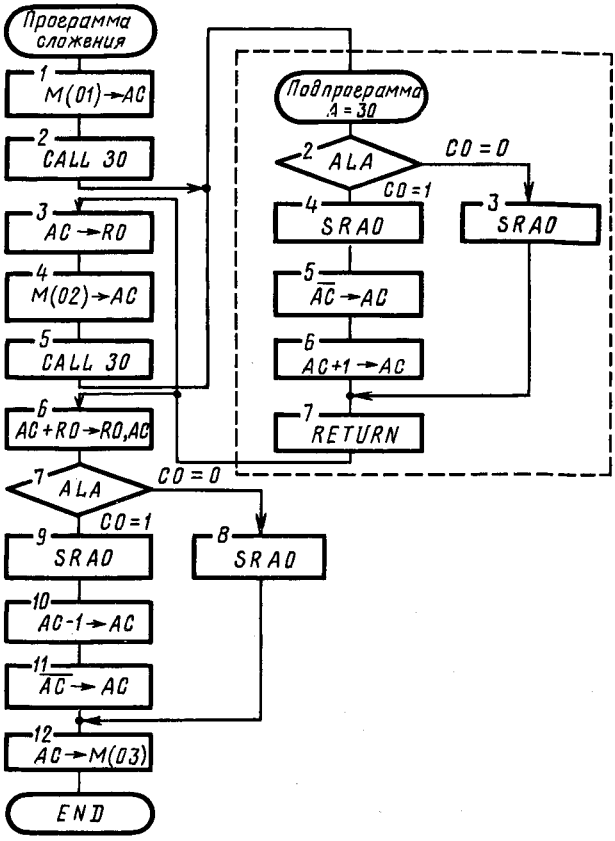


Рис. 9.18. Схема программы сложения в дополнительном коде

слагаемое и формируют его дополнительный код. Команда 6 обеспечивает сложение чисел в дополнительном коде, при этом возможные единицы переноса старшего разряда не учитываются при формировании окончательного результата. Команды 7—12 преобразуют дополнительный код числа в прямой и обеспечивают запись его в ячейку 03 ЗУ.

Задания для домашней подготовки

1. Составить программу сложения чисел в обратном коде.

2. Составить программу сложения чисел в модифицированном обратном коде.

3. Составить программу анализа «переполнения» при сложении чисел в обратном или дополнительном кодах. Переполнение фиксировать по отличию знака результата от совпадающих знаков слагаемых.

4. Составить программу умножения двух четырехразрядных чисел в соответствии со схемой, представленной на рис. 9.19.

5. Используя программу умножения как подпрограмму, реализовать вычисление функции $n!$ Вычислить значение функции при $n=4$; $n=5$.

6. Составить программу, обеспечивающую последовательное включение светодиодов магистрали D с временно́й задержкой, равной времени переполнения двух внутренних регистров ЦПЭ.

7. Это задание аналогично заданию 6, но в качестве временно́й задержки использовать музыкальную программу.

Работа на учебной мини-ЭВМ

1. Разработанные программы ввести в память микро-ЭВМ. 2. Проверить соответствие введенной и исходной программ. 3. Выполнить введенную программу в режиме покомандного исполнения (RS), фиксируя и анализируя промежуточные результаты и содержимое внутренних регистров ЦПЭ с помощью блока индикации. 4. Выполнить введенные программы в циклическом режиме. Повторить вычисления с измененными исходными данными.

Содержание отчета

Отчет должен содержать: **1.** Схему программ. **2.** Рабочие программы. **3.** Результаты промежуточных значений кодов магистралей A, D, M, зафиксированные в процессе покомандного исполнения программ.

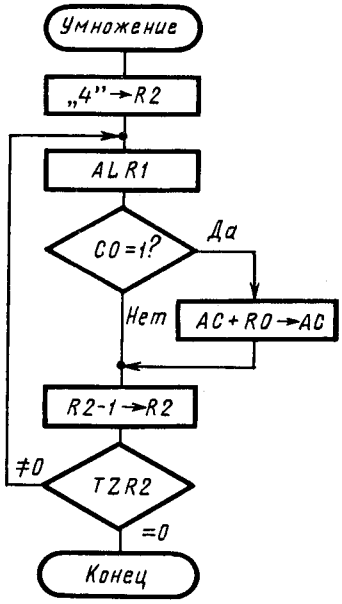


Рис. 9.19. Схема умножения чисел

Задания для самопроверки

1. Назовите основные типы команд.
2. Определите основные формы задания адресов в командах условного перехода.
3. Какие команды обеспечивают переход к заданной команде?
4. Опишите условия по команде TZR1.
5. Какой адрес в ПЗУ определяет шестнадцатеричный код команды?

Лабораторная работа 9.4. Работа микро-ЭВМ с внешними устройствами

Цель работы: *изучить организацию работы микро-ЭВМ с внешними устройствами.*

Краткие сведения из теории

Для выполнения работы необходимо освоить материал, приведенный в § 9.2 и 9.3. Лабораторная микро-ЭВМ имеет возможность подключения 256 внешних устройств. Выбирают внешнее устройство в соответствии с кодами второго байта команд OUT_n, IN_n, задающими адрес внешнего устройства. Связь микро-ЭВМ с внешними устройствами осуществляется по магистрали A, задающей код внешнего устройства, по магистрали B, передающей информацию от внешнего устройства к микро-ЭВМ, и по магистрали D, передающей информацию от микро-ЭВМ к внешним устройствам. Синхронизация по времени может происходить с помощью управляющих сигналов RDY1, формируемых внешним устройством и свидетельствующих о готовности его к процедуре обмена информации [32].

В лабораторном стенде в качестве внешнего устройства используют платы световой индикации, звуковой сигнализации и таймера, описание которых приведено ранее. При составлении программ, работающих в реальном масштабе времени, часто необходима задержка на определенное количество тактов (например, для устранения дребезга контактов). Разберем примеры построения таких программ.

Программа задержки на 256 циклов. Для организации задержки используется счетчик в одном из регистров или в АС. Выход из программы осуществляется по переполнению организованного счетчика. Программа будет иметь следующий вид:

```
Адрес  Код команды  Мнемоника     Комментарий
10     BF           0 →AC
11     CF           AC+1 →AC ←
12     92           TZA          Проверка переполнения
13     15*          =0 ——
14     11*          ≠0 ——
15     A0           END
```

Программа задержки на 65 536 циклов. Для организации задержки бóльшей, чем 256 циклов, длительности необходимо использовать два (и более) регистра, при этом в программе организуются вложенные циклы:

```
10   B7   0 →10
11   BF   0 →AC
12   C7   R0+1→10 ←   } Внутренний цикл
13   8A   TZR0        } Задержка на 256 циклов
14   16*  =0          }
15   12*  ≠0 ——       }
16   CF   AC+1→AC ←   } Внешний цикл
17   92   TZA         } Считается количество проходов внутреннего цикла
18   IA*  =0          }
19   12*  ≠0 ——       }
IA   A0   END ←       }
```

Рассмотрим пример управления точечными светодиодами. Необходимо организовать свечение только одного светодиода из линейки внешних индикаторов, обеспечив его циклический сдвиг.

Блок-схема программы, обеспечивающей заданный алгоритм, представлена на рис. 9.20. Для организации

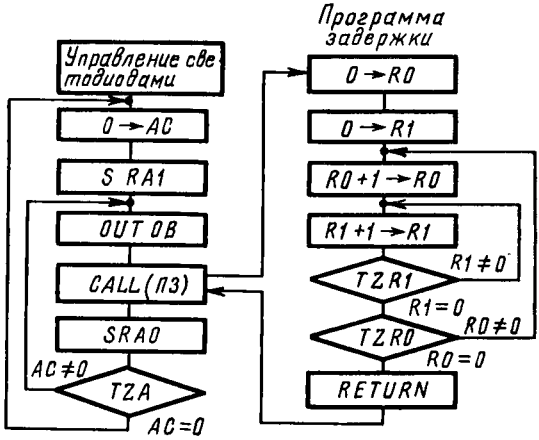


Рис. 9.20. Схема программы управления точечными светодиодами:
ПЗ — подпрограмма задержки

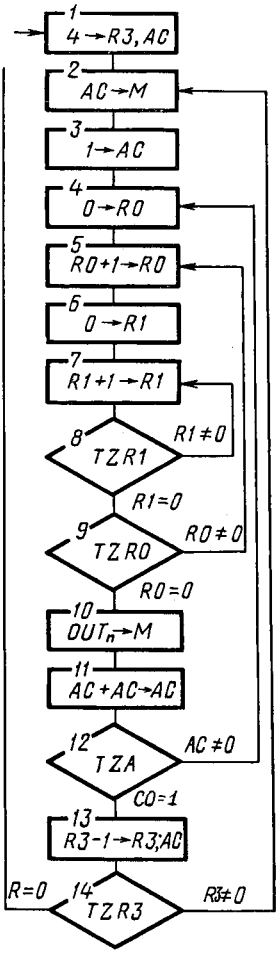


Рис. 9.21. Схема управления индикатором

пауз после переключения светодиодов в программе использована подпрограмма временно́й задержки. Обращение к подпрограмме обеспечивается командой CALL. Временна́я задержка переключения определяется временем переполнения регистров R0, R1 и составляет $256 \times 256 \times 2$ машинных цикла. В адресной части команды обращения к внешнему устройству должен быть задан адрес регистра, управляющего точечными светодиодами (08).

В качестве второго примера рассмотрим программу, обеспечивающую последовательное переключение включенного сегмента на четырех семисегментных индикаторах. Номера разрядов, включающие сегменты индикатора, приведены на рис. 9.15, *б*. Схема программы показана на рис. 9.21. Блок *1* обеспечивает загрузку константы *4* в R3, AC. Содержимое R3 определяет младший байт адреса внешнего устройства. Блок *2* записывает содержимое AC в ячейку памяти, относящуюся к команде OUT_n и определяющую младший байт адреса внешнего устройства. Блок *3* формирует единицу в AC. Блоки *4—9* образуют программу временно́й задержки, передающую управление блоку *10* после переполнения регистров R0, R1. Содержимое блока *10* AC передается на индикатор, блок *11* обеспечивает изменение содержимого AC, соответствующее включению следующего сегмента. Блок *12* организует переход к следующему индикатору (R3 = R3 — 1) после включения восьми сегментов четвертого индикатора.

Номер индикатора вычисляется в блоке *13*. Повторный переход к четвертому индикатору реализуется блоком *14*.

Задания для домашней подготовки

1. Составить программу, выводящую на точечные индикаторы определенную кодовую комбинацию.

2. Ввести в АС код с переключателей, проинвертировать его и вывести на точечные индикаторы.

3. Ввести в АС код с переключателей и вывести его на один из семисегментных индикаторов.

4. Это задание аналогично заданию 3, но код предварительно перевести из двоичного в код семисегментного индикатора для отображения шестнадцатеричных знаков.

5. Составить программу формирования звукового сигнала произвольного тона и длительности.

6. Сформировать перемещающуюся единицу, соответствующую горящему светодиоду, по регистру точечных светодиодов. Перемещение сопровождать звуковым сигналом (использовать музыкальную программу в качестве временно́й задержки).

7. Составить и выполнить программу вывода информации на семисегментный дисплей платы световой индикации следующих слов: ПОпЕ, FACE, PACE, PLUS. Для кодирования сегментов использовать рис. 9.15, *б*.

8. Реализуйте музыкальную программу, воспроизводящую нарастающую по частоте звуковую последовательность. Кодировка звуковой частоты выполняется в соответствии с приложением.

9. Реализуйте программу, воспроизводящую музыкальный текст (см. Приложение).

10. Реализуйте программу, чередующую смену слов задания 7 с музыкальными текстами.

Работа на учебной микро-ЭВМ

1. Ввести программу задержки на 256 тактов. 2. Выполнить программу в шаговом режиме. Убедиться, что в АС происходит счет. 3. Ввести программу задержки на 65 536 тактов. 4. Выполнить программу в шаговом режиме. Для выхода из внутреннего цикла во внешний использовать кнопку R1. Для этого после выполнения нескольких проходов внутреннего цикла с ячейки 13 перейти на ячейку 16, т. е. к внешнему циклу, три раза нажав кнопку R1. Убедиться в том, что после одного прохода внешнего цикла программа возвращается во внутренний цикл. Запустить программу в циклическом режиме. Заметить время выполнения программы. Рассчитать ориентировочное время выполнения одной команды. Ввести программу задержки

на 65 536 тактов, используя два регистра. Оформить ее в виде подпрограммы. Ввести основную программу, осуществляющую счет в АС и использующую для замедления счета подпрограмму задержки. Ввести разработанные подпрограмму и основную программу в микро-ЭВМ. Выполнить программу в циклическом режиме. Определить, во сколько раз замедлился счет в АС.

Ввести в микро-ЭВМ программы, подготовленные в процессе домашней подготовки.

Выполнить введенные рабочие программы, провести анализ промежуточных результатов.

Содержание отчета

Отчет должен содержать: 1. Рабочие программы с комментариями. 2. Результаты анализа промежуточных данных, полученных в процессе исполнения рабочих программ. 3. Функциональную схему внешнего устройства, имеющего двусторонний обмен информацией с микро-ЭВМ.

Задания для самопроверки

1. Какие адреса можно присваивать внешним устройствам, подключаемым в микро-ЭВМ?
2. Сколько внешних устройств одновременно можно подключать к лабораторному стенду?
3. Чем отличаются команды выдачи кодов внешним устройством от команд приема информации от них?
4. Какие блоки должно содержать внешнее устройство для подключения в микро-ЭВМ?
5. Нарисуйте функциональную схему произвольного внешнего устройства с блоками, обеспечивающими совместную работу с микро-ЭВМ.
6. Каким образом осуществляется асинхронный обмен информации внешнего устройства с микро-ЭВМ?
7. Чем отличается обращение к различным семисегментным индикаторам?
8. На основе лабораторной микро-ЭВМ предложите вариант обмена информацией по одной двунаправленной магистрали.

Раздел 3. ЗАДАЧНИК И СБОРНИК ПРИМЕРОВ

Проверить уровень собственной подготовки по освоению микропроцессоров можно с помощью задач и примеров, представленных в этом разделе. Тематика предлагаемых задач соответствует содержанию всех разделов учебников «Микропроцессоры». По каждому разделу приводится подробное решение первой из приведенных задач, для последующих задач дается ответ либо ссылка на главы учебника, которые необходимо изучить для решения предложенной задачи. При решении задач по проектированию узлов и блоков вычислительных устройств необходимо пользоваться справочным материалом по МП БИС, а при решении задач по составлению программного обеспечения — системой команд микропроцессорной БИС (например, системой команд БИС КР580ИК580) или системой команд, рекомендуемой ЭВМ.

Глава 10
ПРИМЕРЫ И ЗАДАЧИ

§ 10.1. Микропроцессоры, микроконтроллеры микро-ЭВМ — массовые средства цифровой вычислительной техники

Вопросы и задания данного раздела ориентированы на уточнение понимания существа микропроцессоров, развития и совершенствования возможностей микропроцессоров по мере развития схемно-технологических баз их производства.

1.1. Какие основные параметры обеспечивают массовость производства и применения микропроцессоров, микроконтроллеров и микро-ЭВМ?

1.2. Отличается ли микропроцессор от цифрового процессора ЭВМ? Если да, то по каким параметрам и характеристикам? Если нет, то почему?

1.3. Объяснить, зачем нужны микроконтроллеры.

1.4. Перечислить основные особенности микро-ЭВМ: однокристальной, многоплатной на основе секционного комплекта БИС микропроцессора. Для каких применений предпочтительны однокристальные микро-ЭВМ? Какие

микро-ЭВМ обеспечивают максимальную производительность сложных вычислений?

1.5. О чем свидетельствует приставка «микро» в терминах «микропроцессор» и «микро-ЭВМ»?

1.6. Потеряли ли свое значение большие и сверхбольшие ЭВМ в связи с созданием микро-ЭВМ?

1.7. Какие принципиально новые возможности возникают при построении многопроцессорной ЭВМ на основе микропроцессоров?

1.8. Будут ли применяться в будущем секционные микропроцессорные комплекты БИС в связи с созданием однокристальных микропроцессоров на основе сверхбольших интегральных схем (СБИС)?

1.9. Сравним ли аналоговый микропроцессор по производительности с аналоговыми вычислительными устройствами? Если да, то зачем создают аналоговые микропроцессоры? Если нет, то какие преимущества имеют аналоговые микропроцессоры перед аналоговыми вычислительными устройствами?

1.10. Какие возможности архитектуры микро-ЭВМ заложены в кристалл БИС микропроцессора?

1.11. Можно ли влиять на основные параметры микропрограммного, программного и аппаратурного уровня микропроцессорной микро-ЭВМ или системы или они жестко заданы архитектурой микропроцессора?

1.12. Какие особенности архитектуры имеют наибольшее значение, если необходимо создать микро-ЭВМ для многократного решения однажды запрограммированного потока задач? Какой микропроцессор обеспечивает экономичное решение множественного потока задач, каждая из которых решается однократно? Какой микропроцессор необходим, если микро-ЭВМ должна решать разнообразные задачи многих видов?

1.13. Какой микропроцессор обеспечивает применение минимального числа БИС в микро-ЭВМ?

1.14. Какими факторами определяется снижение удельной стоимости одного электронного элемента в ИС по мере развития научно-технического прогресса?

1.15. Какими физическими и технологическими факторами ограничивается предельное число электронных элементов в одном кристалле интегральной схемы? Достижимо ли создание микропроцессора с одним миллиардом электронных элементов в одном кристалле? Если да, то когда прогнозируется его создание? Если нет, то какие

принципиальные факторы ставят предел степени интеграции элементов в одном кристалле?

1.16. Позволяет ли развитие аппаратных ресурсов микропроцессора решать проблему снижения затрат на программирование работ микро-ЭВМ? Если да, то какие черты архитектуры микропроцессора должны быть развиты для этого в большей степени?

1.17. Какие преимущества имеет микропроцессорная реализация электронной системы перед ее реализацией на основе комбинационных логических цифровых интегральных схем? Получается ли выигрыш по производительности при этом?

Развитие архитектуры микропроцессоров

Ниже даны вопросы и задания, раскрывающие особенности и развитие архитектуры микропроцессоров.

1.18. Перечислить основные параметры микропроцессора и выделить из них те, которые определяют его характеристики как функционального устройства для обработки цифровой информации и управления этой обработкой.

1.19. Правильно ли утверждение о необходимости расширения системы команд микропроцессора? Сколько команд должна содержать система команд: 10, 100, 1000,..? Какой критерий применяется для рационализации числа команд в системе команд микропроцессора?

1.20. Зачем производить многокристальные комплекты БИС микропроцессора, если имеются однокристальные микропроцессоры?

1.21. Зачем производить однокристальные микро-ЭВМ, если можно создать на основе использования достигнутых технологией возможностей развитый однокристальный процессор?

1.22. Указать основные особенности архитектуры развитого микропроцессора.

1.23. Объяснить, с какой целью производят микропроцессоры с проблемно-ориентированной архитектурой.

1.24. Вследствие каких достоинств получили наибольшее распространение одномагистральные структуры?

1.25. Рассмотреть возможности повышения производительности системы за счет применения: а) более производительных вычислительных устройств; б) более производительного микропроцессора в структуре электронной системы с несколькими магистралями.

1.26. Какой из нижеприведенных видов микропроцессора может обеспечить построение высокопроизводитель-

ной (свыше 1 млн. операций/с) и универсальной системы обработки информации: К1800, АР1804, К1801, КР1802, К1883?

1.27. На основе какого комплекса БИС микропроцессора может быть построена вычислительная система, совместимая по системе команд с микро-ЭВМ типа «Электроника-60» и обеспечивающая максимальную производительность?

1.28. Возможно ли построение микро-ЭВМ, совместимой по системе команд с высокопроизводительной ЭВМ типа БЭСМ-6? Какой из приведенных в табл. 2.1 и 2.2 [57] микропроцессоров позволит получить наибольшую производительность микропроцессорной реализации ЭВМ?

1.29. Позволяют ли кросс-средства универсальных ЭВМ: а) отладить программу решения задачи для микро-ЭВМ; б) оценить время решения задачи; в) провести оптимизацию системы команд микро-ЭВМ?

1.30. Какие вычислительные возможности обеспечивают однокристальные и многокристальные эмуляторы?

1.31. Каким способом можно реализовать аппаратурно в микропроцессорной системе функции операционных систем. Что это дает а) пользователю; б) разработчику системы?

1.32. Какие особенности архитектуры должны быть развиты в микропроцессоре для того, чтобы обеспечить возможность аппаратурной реализации языков программирования высокого уровня?

Развитие логической структуры микропроцессоров

Вопросы и задания данного раздела раскрывают основные требования к структуре микропроцессоров, а также определяют взаимосвязь их архитектурных и структурных характеристик.

1.33. Объяснить, что такое структура микропроцессора; как она связана с архитектурой микропроцессора.

1.34. Перечислить основные элементы структуры микропроцессора. Всегда ли целесообразно увеличивать разрядность кода команд микропроцессора? Почему ограничена 2, 4, 8, 16, 32 бит разрядность обрабатываемых данных в микропроцессорах?

1.35. Объяснить, как связано между собой микропрограммное и программное управление в микропроцессоре. Возможно ли построение микро-ЭВМ только с одним видом управления. Какие недостатки и достоинства приобретает микро-ЭВМ при этом?

1.36. Какой выигрыш получается при реализации одновременного обращения по двум адресам к внутреннему сверхоперативному ОЗУ микропроцессора?

1.37. Какие задачи решает блок прерывания программ микропроцессора? Сколько уровней прерывания должен иметь микропроцессор?

1.38. Перечислить задачи, позволяющие решить стек в микропроцессорной системе. Какой должна быть глубина стека?

1.39. Какие аппаратурные средства микропроцессора позволяют организовать стек в оперативной памяти? Какие недостатки и достоинства свойственны такой реализации стека?

1.40. Перечислить достоинства и недостатки реализации системы обзорных и векторных прерываний. Какой вид прерываний может обеспечить наиболее производительный режим работы микропроцессорной системы, проводящей обработку сигналов большого количества источников информации в реальном масштабе времени?

1.41. За счет каких аппаратурных ресурсов микропроцессора происходит развитие системы микропроцессора с микропрограммным и схемным управлением?

1.42. Перечислить преимущества, обеспечиваемые вводом — выводом данных в канале прямого доступа в память? Какие аппаратурные средства микропроцессора обеспечивают возможность реализации канала прямого доступа в память?

1.43. Объяснить, почему широко используется магистральная структура микропроцессора.

1.44. Перечислить преимущества структуры процессора с распределенными функциями обработки данных и управления процессом обработки? Какая структура предпочтительна для двухкристального или однокристального варианта реализации микропроцессора?

1.45. Какое расширение возможностей микропроцессорной системы можно получить с помощью дополнительных проблемноориентированных микропроцессоров?

Организация и применение электронных микропроцессорных систем обработки данных и управления

Ниже приведены вопросы и задания, раскрывающие особенности организации и применения электронных микропроцессорных систем обработки данных и управления.

1.46. В какой структуре микропроцессорной системы возможно получение наибольших преимуществ за счет

стандартизации интерфейсных средств? В чем выражаются эти преимущества?

1.47. Перечислить функции, реализуемые микропроцессором в микропроцессорной системе. Сколько микропроцессоров может содержать микропроцессорная система?

1.48. Чем отличается микро-ЭВМ от микропроцессорной системы?

1.49. Перечислить функции, выполняемые универсальным программируемым контроллером. Может ли контроллер быть реализован на основе БИС или универсального микропроцессора?

1.50. Зачем нужны блоки программируемого интерфейса?

1.51. Дать определение унифицированного интерфейса.

1.52. Рассмотреть основные функции отдельных шин и их групп в информационной магистрали интерфейса.

1.53. Указать отличие однонаправленной информационной магистрали от двунаправленной.

1.54. Перечислить виды БИС, необходимые для построения высокопроизводительной микро-ЭВМ на основе секционного комплекта БИС микропроцессора.

1.55. Для каких целей применяются встроенные микропроцессорные вычислительные средства в оборудовании и приборах?

1.56. Объяснить достоинства распределенных микропроцессорных информационно-управляющих вычислительных систем.

1.57. Какие изменения в оборудовании или приборах необходимо провести для того, чтобы обеспечить выдачу и прием сигналов управления для микропроцессора?

1.58. Связаны ли между собой микропроцессоры и роботы?

1.59. Какие возможности открывают микропроцессоры для реализации параллельных вычислительных процессов?

1.60. Можно ли применять микропроцессорные вычислительные средства для решения задач управления?

Оценка возможностей и обоснование выбора архитектуры микропроцессора

Вопросы и задания данного раздела раскрывают методику оценки возможностей архитектуры микропроцессоров, а также особенности алгоритмов обоснования выбора микропроцессора.

1.61. Указать основные функции аппаратурных, программных и микропрограммных средств в микропроцес-

сорной системе; их развитие по мере развития микропроцессоров.

1.62. Перечислить основные возможности, представляемые регистровой и стековой архитектурой микропроцессора. Какими факторами объясняется возникновение микропроцессоров с архитектурой «память — память»?

1.63. Объяснить сущность понятия ортогональности архитектуры микропроцессора. Какие новые качества обеспечивает многоуровневая ортогональная архитектура микропроцессора?

1.64. Перечислить основные критерии выбора микропроцессора для конкретных применений по его назначению в системе обработки или управления данными.

1.65. Перечислить основные функции, выполняемые микропроцессором: а) во встроенной системе управления; б) в системе управления реального времени; в) в системе обработки данных; г) в системе для научно-технических расчетов.

1.66. Указать критерий, на основе которого можно дать обобщенную оценку эффективности архитектуры микропроцессора.

1.67. Какой из параметров микропроцессора оказывает наибольшее влияние на эффективность применения микропроцессора в системе: а) вычислительной; б) управления; в) коммутации?

1.68. Перечислить наиболее эффективные сферы применения 2, 4, 8, 16- и 32-разрядных микропроцессоров и их секций.

1.69. Объяснить, почему возможности расширения электронной микропроцессорной системы относятся к важным чертам выполненного проекта.

1.70. Как взаимосвязаны между собой функциональные требования к микропроцессорной системе с архитектурными особенностями микропроцессора, параметрами и интерфейсом всех системных устройств, используемых при реализации системы.

1.71. Объяснить, почему необходимо почти во всех случаях специализировать универсальные микропроцессорные средства при их применении для решения четко определенных задач.

1.72. Какое значение имеет сохранение программного (архитектурного) задела при проектировании новых систем? Можно ли ограничиться сохранением только конструктивно-технологической совместимости аппаратурных средств системы?

1.73. Какие условия определяют возможность отказа от уже отработанной архитектуры и позволяют считать технически и экономически выгодным переход к новой архитектуре микропроцессора?

1.74. В чем отличие моделирования возможностей архитектуры микропроцессора от процедур эталонного программирования? Какой из этих способов предпочтителен и на какой стадии разработки?

1.75. Как можно оценить эффективность выполненного проекта микропроцессорной системы? Как снизить затраты на проведение оценки эффективности выбранной архитектуры?

§ 10.2. Организация вычислительных процессов на микро-ЭВМ

Этот раздел содержит примеры и задачи по программированию, ориентированные на микро-ЭВМ «Электроника-60» и микропроцессорный диалоговый вычислительный комплекс (ДВК) «Электроника НЦ-8020» [57]. Так как обе микро-ЭВМ программно-совместимы друг с другом, то в дальнейшем, говоря о микро-ЭВМ «Электроника-60», будем подразумевать также и ДВК «Электроника НЦ-8020».

Раздел охватывает три темы: анализ команд и режимов адресации памяти микро-ЭВМ, программирование микро-ЭВМ на языке ассемблера и программирование на языке ПАСКАЛЬ.

Примеры и задачи данного раздела желательно проверить непосредственно на микро-ЭВМ. Для проверки примеров и задач первой из трех тем целесообразно воспользоваться режимом пультовых операций микро-ЭВМ. В этом режиме не требуется операционная система, а необходимые действия для ввода команд и данных, пуска программ и их фрагментов, проверки содержимого регистров и ячеек (слов) памяти осуществляются программистом с помощью специальных пультовых команд, вводимых с клавиатуры терминала.

Для трансляции, редактирования и выполнения программ, написанных на языках ассемблера и ПАСКАЛЬ, необходима операционная система. При использовании микро-ЭВМ «Электроника-60» такой операционной системой может быть, например, ФОДОС или РАФОС [57]. На ДВК «Электроника НЦ-8020» следует применять в этом случае ОС ДВК.

Анализ команд и режимов адресации памяти микро-ЭВМ

При решении задач данной темы предполагается знание материала § 10.3 и 10.4 из [57]. Полезно будет также изучение разделов, содержащих сведения о системах команд и режимах адресации памяти отечественных и зарубежных ЭВМ, с которыми программно совместима микро-ЭВМ «Электроника-60».

Примечание. В примерах и задачах данной темы будем использовать обозначения:

1. Запись Rm/XXXXXX говорит о том, что регистр Rm (где m = 0,1,..., 7) содержит восьмеричное число XXXXXX.

2. AAAAAA/XXXXXX означает, что слово памяти с адресом AAAAAA

хранит команду или данное XXXXXX. И адрес, и команда (или данное) выражаются в восьмеричной системе счисления.

3. Запись Rm/? или AAAAAA/? свидетельствует о том, что содержимое регистра Rm или соответственно слова памяти с адресом AAAAAA неизвестно и может быть произвольным.

● **2.1.** Записать машинный код команды

MOV 1024(R3), 562(R5)

вычислить адреса операндов (источника и приемника), а также определить содержимое регистров R3 и R5 и операндов в результате ее выполнения, если известно, что до выполнения этой команды в регистре R3 содержится число 2416, в регистре R5 — число 3124, а в слове памяти с адресом 3442 — число 12345, т. е. заданы R3/2416, R5/3124 и 3442/12345.

Решение. Так как для обоих операндов команды задан индексный режим адресации, то команда должна занимать три слова памяти. Обращаясь к таблице команд PDP-11 [37], находим, что общая запись первого слова команды MOV имеет вид 01SSDD, где SS представляет код режима адресации и номер регистра для операнда-источника, а DD — код той же информации для операнда-приемника (см. рис. 10.13 [37]). Так как код индексного режима имеет значение 6, то SS=63, а DD=65. Следовательно, искомый трехсловный машинный код команды имеет такой вид:

016365 (первое слово)
001024 (второе слово)
000562 (третье слово)

Адреса операнда-источника и операнда-приемника определятся выражениями 1024 + (R3) = 1024 + 2416 = 3442 и 562 + (R5) = 562 + 3124 = 3706 соответственно. Подчеркнем, что вычисления выполнены в восьмеричной системе. Это относится к вычислениям и в других примерах и задачах рассматриваемой темы.

Содержимое регистров и операндов после выполнения команды следующее: R3/2416, R5/3124, 3442/12345, 3706/12345.

Для проверки команды непосредственно на микро-ЭВМ необходимо:

1. Включить микро-ЭВМ и перевести ее в режим пультовых команд.

2. Используя пультовые команды, занести машинный код команды в три последовательные слова памяти; выбрав, например, адрес 1000 для первого слова. Следует,

кроме того, обнулить (очистить) четвертое слово, т. е. сформировать код команды HALT, чтобы обеспечить останов микро-ЭВМ после выполнения команды. В результате в памяти машины будет записано:

1 000/016365 1 002/001024	Код команды MOV
1 004/000562 1 006/000000	Код команды HALT

3. С помощью пультовых команд занести в регистры R3 и R5 заданные числа 2416 и 3124 соответственно.

4. Пультовой командой 1000C запустить команду MOV.

5. Используя пультовые команды, проверить содержимое регистров и слов памяти, участвовавших в выполненной команде MOV.

2.2. Определить код команды.

MOV −2(R3), R4

и результат ее выполнения при R3/1264, 1262/7651 и R4/?

Содержимое второго слова команды, т. е. смещение −2, должно быть представлено в дополнительном коде.

2.3. Записать код команды

MOV R0, * R1

Назвать режимы адресации ее операндов и определить результат ее выполнения, если заданы R0/123456 и R1/1340.

2.4. Определить код команды

ADD * 2534(R1), R3

и показать, что в результате ее выполнения при R1/52, R3/4122 2606/27342 и 27342/123 в регистре R3 получится число 4245.

2.5. Проанализировать байтовую команду с автоинкрементным режимом адресации второго операнда

MOVB R0, (R1) +

и при R0/123456, R1/1340, 1340/? показать, что в результате ее выполнения в байте памяти с адресом 1340 окажется число 56. Чему станет равно содержимое регистра R1?

2.6. Сформировать машинный код команды

SUB R2, * (R3)+

и при R2/125, R3/21346, 21346/3614, 3614/236 определить результат операции вычитания и адрес слова памяти, в которое будет помещен этот результат после выполнения команды. Назвать режим адресации второго операнда.

2.7. Записать машинный код команды

INC —(R3)

и при R3/2634, 2632/7 определить результат ее выполнения. Какой режим адресации использован в команде?

2.8. Сформировать машинный код команды

CMP R0, —(R1)

и при R0/123, R1/1266, 1264/124 определить значения признаков N, Z, V, C в слове (регистре) состояния процессора после ее выполнения. Изменится ли содержимое регистров R0 и R1?

2.9. Определить результат выполнения команды

XOR R1, * —(R4)

при R1/12746, R4/3652, 3650/13204, 13204/15023. Каков режим адресации второго операнда?

2.10. Записать машинный код команды

BISB# 1234, * (R1) +

и при R1/3572, 3572/3000, 3000/12345 определить результат ее выполнения. Как изменится результат, если вместо BISB использовать BIS (при прочих равных условиях)?

2.11. Определить значения признаков N, Z, V, C в слове состояния процессора после выполнения команды

CMPB R2, * # 12342

при R2/405 и 12342/5. Изменится ли результат, если вместо CMPB использовать CMP?

2.12. Определить значения признаков N, Z, V, C в слове состояния процессора в результате выполнения команды

BIT# 156432, * # 3000

при 3000/100432, если до выполнения команды N=Z=V=C=1. Сколько слов занимает команда в памяти микро-ЭВМ?

2.13. Записать машинный код команды

BIC# 12,2472

и определить содержимое слова памяти с адресом 2472 после выполнения этой команды, если первое слово команды имеет адрес 2450 и задано 2472/21773. Изменится ли результат выполнения команды, если вместо BIC записать BICB?

2.14. Определить результат команды занесения числа в стек

```
MOV R3, -(R6)
```

при R3/12356 и R6/364. Какой адрес имеет слово, в которое будет записано число?

2.15. Проанализировать команду извлечения числа из стека

```
MOV (R6)+, *#2400
```

при R6/346 и 346/4321. Какой адрес имеет слово памяти, в которое будет помещено извлеченное из стека число, и каково это число?

2.16. Сформировать код команды извлечения числа из байтового стека

```
MOVB (R5)+, *#3214
```

и объяснить, почему при R5/346, 346/14321 и 3214/174677 в результате выполнения команды содержимое слова с адресом 3214 станет равно 174721.

● **2.17.** Определить код команды

```
BR EXIT,
```

если эта команда имеет адрес 6544, а метка EXIT — адрес 6556.

Решение. При выполнении команды программный счетчик (счетчик команд) содержит адрес $6544+2=6546$. Так как $6556-6546=10$ байт, то младший байт кода команды должен хранить смещение $10:2=4$ слов (напомним, что арифметические действия выполняются в восьмеричной системе счисления). Поскольку при нулевом смещении код команды BR имеет значение 000400, с учетом смещения 4 получим искомый код $000400+4=000404$.

2.18. Определить машинный код и адрес ветвления команды

```
BR .-12,
```

если она располагается в слове памяти с адресом 6634.

2.19. Определить машинный код и адрес команды

```
BLT,+16.,
```

обеспечивающей ветвление по адресу 6636. Учесть при этом, что смещение задано в десятичной системе.

● **2.20.** Вычислить адрес ветвления, осуществляемого командой BR, имеющей код 000766 и расположенной в слове памяти с адресом 6302.

Р е ш е н и е. Младший байт слова памяти, в котором хранится код команды, содержит значение 366. Так как старший бит этого байта имеет единичное значение, то 366 представляет собой отрицательное смещение, заданное в дополнительном коде и выраженное в словах. Следовательно, значение смещения, выраженное в байтах, равно $2 \cdot 366 = 754$. Записывая это смещение так, чтобы оно заняло полное 16-разрядное слово, получим число 177754. Искомый адрес ветвления равен сумме адреса команды, увеличенного на 2, и вычисленного смещения, т. е. $6304 + 177754 = 6260$.

2.21. Вычислить адрес ветвления, осуществляемого командой, которая имеет машинный код 001442 и расположена в памяти по адресу 6254. Каков мнемонический код операции для данной команды?

2.22. Определить машинный код команды безусловного перехода

BR 2432,

расположенной в памяти по адресу 2544.

Р е ш е н и е. Вычислим сначала величину смещения, выраженную в словах:

$$(2432 - (2544 + 2)) : 2 = (-114) : 2 = -46.$$

Отрицательное восьмеричное число 46 в дополнительном коде, занимающее 1 байт, равно 332. Так как при нулевом смещении код команды с операцией BR равен 000400, то с учетом вычисленного смещения искомый код команды определится суммой $000400 + 332$, т. е. равен 000732.

2.23. Записать машинный код команды

JMP ABAC,

если команда имеет адрес 5120, а метка ABAC — адрес 6202.

Программирование для микро-ЭВМ на языке ассемблера

Перед решением задач данной темы следует изучить материал из § 11.2 [57]. Полезно будет также изучение разделов, содержащих подробные сведения о языке ассемблера отечественных и зарубежных ЭВМ, с которыми программно-совместима микро-ЭВМ «Электроника-60».

• **2.24.** Написать на языке ассемблера программу, которая должна обеспечивать ввод с клавиатуры терминала последовательности из 10 восьмеричных чисел без знака, отделенных друг от друга запятой, преобразование этих

чисел из символьной формы в форму неотрицательных целых чисел и их размещение в памяти микро-ЭВМ.

Проверку ввода знаков с клавиатуры осуществлять сканированием (циклическим опросом) состояния готовности клавиатуры. Программу оформить в виде абсолютной программной секции с начальным восьмеричным адресом 500.

Решение. Возможный вариант текста требуемой программы приведен на рис. 10.1. В самом начале программы объявлены имена используемых в ней регистров и заданы адреса регистров состояния и данных терминала: KSTAT и PSTAT — адреса регистров состояния клавиатуры и дисплея (печатающего устройства), а KBUF и PBUF — адреса регистров данных, т. е. буферных регистров, этих же двух устройств. Напомним, что дисплей (печатающее устройство) и его клавиатура считаются двумя различными внешними устройствами.

Директива .ASECT и следующий за ней оператор прямого присваивания определяют абсолютную программную секцию с начальным восьмеричным адресом 500. Смысл прочих операторов поясняется с помощью комментариев. Отметим, что если с клавиатуры будет введен знак, отличный от восьмеричной цифры или запятой, то он игнорируется программой. По мере ввода восьмеричных цифр очередного числа каждый из введенных знаков кода КОИ-7 преобразуется в трехразрядный двоичный код соответствующей цифры. Этот код заносится (упаковывается) в младшие разряды регистра R1, содержимое которого предварительно сдвигается влево на три разряда. Таким образом, результат ввода накапливается в регистре R1 в форме неотрицательного целого восьмеричного числа. Если будет введено столько цифр, что соответствующее восьмеричное число не помещается в 16 двоичных разрядах, то старшие разряды числа теряются. При вводе знака запятой число из регистра R1 переписывается в очередное слово массива MAS. После ввода 10 чисел программа завершает свою работу.

2.25. Модифицировать программу из предыдущей задачи таким образом, чтобы она представляла собой относительную программную секцию и завершалась не командой останова, а обеспечивала возврат управления операционной системе микро-ЭВМ.

2.26. Переписать программу из задачи 2.24 таким образом, чтобы ввод знаков с клавиатуры терминала осуществлялся по прерываниям. Какой адрес имеет вектор

```
;ВВОД И ПРЕОБРАЗОВАНИЕ ЦЕЛЫХ ЧИСЕЛ БЕЗ ЗНАКА
   R0=%0
   R1=%1
   R2=%2
   R3=%3
   R4=%4
   KSTAT=177560          ; АДРЕСА
   KBUF =KSTAT+2         ; РЕГИСТРОВ
   PSTAT=KSTAT+4         ; СОСТОЯНИЯ
   PBUF =KSTAT+6         ; И ДАННЫХ ТЕРМИНАЛА
         .ASECT
         .=500
START:   MOV   LIMIT,R3  ; LIMIT -> R3
         MOV   #MAS,R4   ; АДРЕС МАССИВА -> R4
NEW:     CLR   R1        ; ОЧИСТКА R1
INPUT:   CLR   R0        ; ОЧИСТКА R0
         MOV   #1,KSTAT  ; ИНИЦИИРОВАТЬ КЛАВИАТУРУ
TEST1:   TSTB  KSTAT     ; ГОТОВА КЛАВИАТУРА?
         BPL   TEST1     ; НЕТ
TEST2:   TSTB  PSTAT     ; ДИСПЛЕЙ ГОТОВ?
         BPL   TEST2     ; НЕТ
         MOVB  KBUF,R0   ; ВВОД ЗНАКА С КЛАВИАТУРЫ В R0
         MOVB  R0,PBUF   ; ОТОБРАЖЕНИЕ ЗНАКА
         CLR   R2        ; ОЧИСТКА R2
         MOVB  R0,R2     ; R0 -> R2
         CMP   #54,R2    ; ВВЕДЕН ЗНАК ЗАПЯТОЙ?
         BEQ   INMAS     ; ДА
         BIC   #177407,R2; ВЫДЕЛЕНИЕ ЧАСТИ ЗНАКА
         CMP   #60,R2    ; ВВЕДЕНА ВОСЬМЕРИЧНАЯ ЦИФРА?
         BNE   INPUT     ; НЕТ
         ASL   R1        ; ПОДГОТОВКА МЕСТА
         ASL   R1        ; В РЕГИСТРЕ R1
         ASL   R1        ; ДЛЯ НОВОЙ ЦИФРЫ
         BIC   #177770,R0; ВЫДЕЛЕНИЕ ЦИФРЫ В R0
         BIS   R0,R1     ; ЗАНЕСЕНИЕ ЦИФРЫ ИЗ R0 В R1
         BR    INPUT     ; НА ВВОД НОВОГО ЗНАКА
INMAS:   MOV   R1,(R4)+  ; ЗАПИСЬ ЧИСЛА ИЗ R1 В МАССИВ
         SUB   #1,R3     ; УМЕНЬШЕНИЕ СЧЕТЧИКА
         BNE   NEW       ; НА ВВОД НОВОГО ЧИСЛА
         HALT            ; ОСТАНОВ
LIMIT:   .WORD 10.       ; ЧИСЛО СЛОВ В МАССИВЕ
MAS:     .=.+20.         ; РЕЗЕРВИРОВАНИЕ ПАМЯТИ
         .END  START
```

Рис. 10.1. Программа ввода с клавиатуры последовательностей восьмеричных цифр и их преобразования в форму неотрицательных целых чисел

прерывания для клавиатуры? Каким должно быть содержимое двух слов этого вектора прерывания? В каком месте модифицированной программы должно быть сформировано содержимое слов вектора прерывания?

2.27. Написать программу, которая отыскивает максимальный элемент одномерного массива целых чисел. Размер массива — 100 элементов, каждый элемент занимает одно слово памяти. Элемент массива с максимальным значением занести в регистр R0, а индекс найденного максимального элемента — в регистр R1. Если массив содержит несколько элементов с максимальным значением, то выбрать первый из них при просмотре массива в порядке увеличения индекса.

2.28. Написать программу, которая попеременно суммирует и вычитает содержимое последовательных слов памяти с адресами в диапазоне от 0 до 17776 включительно, трактуя содержимое каждого слова как целое число. Иными словами, к исходному нулевому значению результата надо прибавить содержимое слова с адресом 0, затем из полученной суммы вычесть содержимое слова с адресом 2, потом к результату прибавить содержимое слова с адресом 4 и т. д. Программа должна остановиться, если после очередной операции суммирования или вычитания будет получен нулевой результат или если будут просмотрены все слова в заданном диапазоне адресов. Результат суммирования — вычитания следует хранить в регистре R0, счетчик общего числа слов — в регистре R1, а для адресации последовательных слов использовать регистр R2. Программа должна содержать минимально возможное число операторов машинных команд, включая и оператор останова.

2.29. Написать подпрограмму вывода на печатающее устройство целых восьмеричных чисел со знаком. Перед обращением к подпрограмме число, подлежащее выводу на печать, находится в регистре R0. Предполагается, что обращение к подпрограмме должно осуществляться оператором вида JSR PC OCTOUT, где OCTOUT — метка первого оператора машинной команды в подпрограмме. Вывод из подпрограммы осуществлять оператором RTS PC. Для печати знака и цифр числа воспользоваться вспомогательной подпрограммой TYPE, включив обращение к ней в требуемой подпрограмме. Перед обращением к подпрограмме TYPE код знака или цифры должен быть помещен в младший байт регистра R3.

2.30. Написать подпрограмму, осуществляющую умно-

жение двух целых чисел без знака, каждое из которых занимает одно слово памяти. Операцию умножения осуществлять путем последовательного сдвига множимого на один разряд и сложения [42]. Для результата умножения зарезервировать два последовательных слова памяти: слово с меньшим адресом — для младшей части результата и слово с большим адресом — для старшей части.

● **2.31.** Написать программу, обеспечивающую прием и передачу произвольных символов кода КОИ-7 между двумя микро-ЭВМ по линии связи, содержащей два асинхронных последовательных канала приема и передачи. Каждый из каналов трактуется как внешнее устройство с парой регистров, имеющих следующие адреса: 176560 и 176562 — для регистров состояния и данных канала приема; 176564 и 176566 — для регистров состояния и данных канала передачи. Требуется, чтобы посылка каждого символа в линию связи осуществлялась с предварительным опросом состояния готовности канала передачи, а прием каждого символа — по прерыванию из канала приема (адрес вектора прерывания 320).

Символы, подлежащие передаче, должны вводиться человеком-оператором с клавиатуры (адреса регистров состояния и данных — 177560 и 177562) и отображаться перед передачей на экране дисплея (адреса регистров состояния и данных — 177564 и 177566). Принимаемые символы выводятся на тот же дисплей, что и передаваемые данные. Программа должна обеспечивать возможность одновременной передачи и, следовательно, приема символов обеими микро-ЭВМ, при этом передаваемые и принимаемые символы будут при отображении на экране случайным образом чередоваться.

Решение. Вариант текста программы приведен на рис. 10.2. Текст подробно прокомментирован и не нуждается в особом описании. Подчеркнем лишь, что прием символов реализуется программой обработки прерываний, которая, вклиниваясь в работу основной программы (программы ввода с клавиатуры, отображения и передачи символов), выводит каждый принятый символ в текущую позицию курсора на экране дисплея. Следует также отметить применение стека для сохранения контекста основной программы (в том числе содержимого регистра R0) в момент ее прерывания. Читатель должен иметь в виду, что программа приема и передачи одновременно используется каждой из двух связанных микро-ЭВМ.

```
;*************************************************************
;*     ПРОГРАММА ПРИЕМА И ПЕРЕДАЧИ БАЙТОВ ПО ЛИНИИ СВЯЗИ      *
;*************************************************************
;    АДРЕСА РЕГИСТРОВ УСТРОЙСТВ И КАНАЛОВ ПРИЕМА И ПЕРЕДАЧИ
;       КЛАВИАТУРА ДИСПЛЕЯ:
            SCLAV = 177560          ; СОСТОЯНИЕ
            DCLAV = 177562          ; ДАННЫЕ
;       ЭКРАН ДИСПЛЕЯ:
            SDISP = 177564          ; СОСТОЯНИЕ
            DDISP = 177566          ; ДАННЫЕ
;       КАНАЛ ПРИЕМА БАЙТОВ:
            SRCV  = 176560          ; СОСТОЯНИЕ
            DRCV  = 176562          ; ДАННЫЕ
;       КАНАЛ ПЕРЕДАЧИ БАЙТОВ:
            SSND  = 176564          ; СОСТОЯНИЕ
            DSND  = 176566          ; ДАННЫЕ
;    АДРЕС ВЕКТОРА ПРЕРЫВАНИЯ ИЗ КАНАЛА ПРИЕМА
            AVECT = 320
;
            .CSECT
;*****************************************************
;*                  ОСНОВНАЯ ПРОГРАММА                *
;*****************************************************
START:  MOV     #.,SP           ; ИНИЦИАЛИЗИРОВАТЬ УКАЗАТЕЛЬ СТЕКА
        MOV     #RCVSYM,AVECT   ; ЗАНЕСТИ АДРЕС ПРОГРАММЫ ОБРАБОТКИ
                                ; ПРЕРЫВАНИЙ В ВЕКТОР ПРЕРЫВАНИЙ
        MOV     #100,SRCV       ; РАЗРЕШИТЬ ПРЕРЫВАНИЯ ИЗ КАНАЛА
;   ВВОД ОЧЕРЕДНОГО СИМВОЛА С КЛАВИАТУРЫ
CYCLE:  MOV     #1,SCLAV        ; РАЗБЛОКИРОВАТЬ КЛАВИАТУРУ
INSYM:  TSTB    SCLAV           ; НАЖАТА КЛАВИША ?
        BPL     INSYM           ; НЕТ ЕЩЕ, ПОВТОРИТЬ ТЕСТ
        MOVB    DCLAV,R0        ; СИМВОЛ С КЛАВИАТУРЫ - В РЕГИСТР R0
;   ОТОБРАЖЕНИЕ СИМВОЛА НА ЭКРАНЕ ДИСПЛЕЯ
DSPLAY: TSTB    SDISP           ; ЭКРАН ГОТОВ К ОТОБРАЖЕНИЮ ?
        BPL     DSPLAY          ; НЕТ ЕЩЕ, ПОВТОРИТЬ ТЕСТ
        MOVB    R0,DDISP        ; СИМВОЛ ИЗ R0 - НА ЭКРАН
;   ПОСЫЛКА СИМВОЛА В КАНАЛ ПЕРЕДАЧИ
SNDSYM: TSTB    SSND            ; КАНАЛ ГОТОВ К ПЕРЕДАЧЕ БАЙТА ?
        BPL     SNDSYM          ; НЕТ ЕЩЕ, ПОВТОРИТЬ ТЕСТ
        MOVB    R0,DSND         ; СИМВОЛ ИЗ R0 - В КАНАЛ
        BR      CYCLE           ; НА ВВОД НОВОГО СИМВОЛА
                                ; С КЛАВИАТУРЫ
;
;*****************************************************
;*  ПРОГРАММА ОБРАБОТКИ ПРЕРЫВАНИЙ ИЗ КАНАЛА ПРИЕМА  *
;*****************************************************
RCVSYM: BICB    #100,SRCV       ; ЗАПРЕТИТЬ ПРЕРЫВАНИЯ ИЗ КАНАЛА
        MOV     R0,-(SP)        ; СОХРАНИТЬ РЕГИСТР R0 В СТЕКЕ
        MOVB    DRCV,R0         ; СИМВОЛ ИЗ КАНАЛА - В РЕГИСТР R0
TS:     TSTB    SDISP           ; ЭКРАН ГОТОВ К ОТОБРАЖЕНИЮ ?
        BPL     TS              ; НЕТ ЕЩЕ, ПОВТОРИТЬ ТЕСТ
        MOVB    R0,DDISP        ; СИМВОЛ ИЗ R0 - НА ЭКРАН
        MOV     (SP)+,R0        ; ВОССТАНОВИТЬ РЕГИСТР R0 ИЗ СТЕКА
        BISB    #100,SRCV       ; РАЗРЕШИТЬ ПРЕРЫВАНИЯ ИЗ КАНАЛА
        RTI                     ; ВОЗВРАТ К ОСНОВНОЙ ПРОГРАММЕ
        .END    START
```

Рис. 10.2. Программа, обеспечивающая одновременную передачу и прием символов (байтов) по линии связи между двумя микро-ЭВМ

• **2.32.** Написать программу, обеспечивающую чтение заданного сектора на гибком магнитном диске накопителем на гибком магнитном диске (НГМД) типа «Электроника ГМД-7012», в основную память микро-ЭВМ. Контроллер накопителя имеет регистр состояния (РС) с адресом 177170 и регистр данных (РД) с адресом 177172. Кроме буферизации байтов данных РД на разных этапах выполнения заданной операции выполняет и другие функции. Так, РД используется для последовательной буферизации адресов сектора, дорожки, а также для хранения признаков состояния завершившейся операции.

Примечание. Состав и назначение разрядов регистра состояния и регистра данных (в разных его ролях) описаны ниже.

Разряды регистра состояния:

0 — флаг пуска операции;

1—3 — код требуемой операции (000 — заполнить внутренний буфер НГМД, имеющий размер 128 байт; 001 — прочитать внутренний буфер НГМД в оперативную память микро-ЭВМ; 010 — записать сектор, используя содержимое внутреннего буфера; 001 — прочитать сектор, поместив результат во внутренний буфер; 100 — не используется; 101 — чтение текущего состояния НГМД; 110 — запись сектора с меткой; 111 — чтение регистра ошибок НГМД);

4 — номер одного из двух дисководов (приводов);

5 — флаг завершения операции (при установленном флаге разрешения прерывания установка флага завершения операции генерирует сигнал прерывания процессора);

6 — флаг разрешения прерывания;

7 — флаг требования передачи (готовность к выполнению операции);

8—13 — не используются;

14 — начальная установка НГМД;

15 — признак наличия какой-нибудь ошибки в НГМД.

Разряды регистра данных:

0—7 — значение байта, передаваемого между основной памятью микро-ЭВМ и внутренним буфером НГМД;

8—15 — не используются.

Разряды регистра адреса дорожки:

0—6 — номер дорожки (восьмеричное число в диапазоне 0—114);

7 — всегда хранит нуль;

8—15 — не используются.

Разряды регистра адреса сектора:

0—4 — номер сектора (восьмеричное число в диапазоне 1—32);

5—7 — всегда содержат нули;

8—15 — не используются.

Разряды регистра ошибок и состояния операции:

0 — ошибка циклического контроля по избыточности;

1 — ошибка четности;

2 — признак завершения начальной установки НГМД;

3—5 — не используются;

6 — признак обнаружения специальной марки, предшествующей блоку данных;

7 — готовность дисковода (привода).

```
; ПРОГРАММА ЗАПИСИ НА НГМД 'ЭЛЕКТРОНИКА ГМД-7012'
        RXCS=177170         ; РЕГИСТР СОСТОЯНИЯ
        RXDB=177172         ; РЕГИСТР ДАННЫХ
        RXSA=177172         ; РЕГИСТР АДРЕСА СЕКТОРА
        RXTA=177172         ; РЕГИСТР АДРЕСА ДОРОЖКИ
        RXES=177172         ; РЕГИСТР ОШИБОК
        .CSECT
; ************************************************************
; *  ЗАПОЛНЕНИЕ БУФЕРА НГМД ДАННЫМИ ИЗ ПАМЯТИ МИКРО-ЭВМ     *
; ************************************************************
WRNGMD: MOV   #-10,PTRY     ; УСТАНОВИТЬ СЧЕТЧИК ПОВТОРЕНИЙ
                            ; ОПЕРАЦИИ
SETUP:  MOV   #DATA,R0      ; R0 <--- АДРЕС ОБЛАСТИ ДАННЫХ
        MOV   COUNT,R1      ; R1 <--- СЧЕТЧИК БАЙТОВ
        MOV   FILBUF,RXCS   ; КОМАНДА ЗАПОЛНЕНИЯ БУФЕРА НГМД
LOOP:   TSTB  RXCS          ; ЕСТЬ БИТ ЗАПРОСА ?
        BEQ   LOOP          ; НЕТ ЕЩЕ
        BMI   FILL          ; НА ПЕРЕДАЧУ БАЙТА
ERR1:   INC   PTRY          ; МОДИФИЦИРОВАТЬ СЧЕТЧИК ПОВТОРЕНИЙ
        BNE   SETUP         ; МОЖНО ПОВТОРИТЬ ОПЕРАЦИЮ
        HALT                ; НЕИСПРАВИМАЯ ОШИБКА
; ПЕРЕДАЧА ОЧЕРЕДНОГО БАЙТА ИЗ ПАМЯТИ МИКРО-ЭВМ В БУФЕР НГМД
FILL:   MOVB  (R0)+,RXDB
        DECB  R1            ; МОДИФИЦИРОВАТЬ СЧЕТЧИК БАЙТОВ
        BNE   LOOP          ; БАЙТЫ ПЕРЕДАНЫ ЕЩЕ НЕ ВСЕ
; ************************************************************
; *          ЗАПИСЬ СЕКТОРА ИЗ БУФЕРА НГМД                   *
; ************************************************************
WRTS:   MOV   #-10,PTRY     ; УСТАНОВИТЬ СЧЕТЧИК ПОВТОРЕНИЙ
                            ; ОПЕРАЦИИ
RTR:    MOV   WRSECT,RXCS   ; КОМАНДА ЗАПИСИ СЕКТОРА НГМД
W1:     TSTB  RXCS          ; ЕСТЬ БИТ ЗАПРОСА ?
        BEQ   W1            ; НЕТ ЕЩЕ
        BPL   ERR           ; ОШИБКА
        MOVB  SECTOR,RXSA   ; ЗАДАТЬ АДРЕС СЕКТОРА
W2:     TSTB  RXCS          ; ЕСТЬ БИТ ЗАПРОСА ?
        BEQ   W2            ; НЕТ ЕЩЕ
        BPL   ERR           ; ОШИБКА
        MOVB  TRACK,RXTA    ; ЗАДАТЬ АДРЕС ДОРОЖКИ
W3:     BIT   #DONE,RXCS    ; ОПЕРАЦИЯ ЗАВЕРШЕНА ?
        BEQ   W3            ; НЕТ ЕЩЕ
        TST   RXCS          ; УСПЕШНОЕ ЗАВЕРШЕНИЕ ?
        BMI   ERR           ; НЕТ
        HALT                ; ДА. ЗАПИСЬ НА НГМД ЗАКОНЧЕНА
ERR:    INC   PTRY          ; МОДИФИЦИРОВАТЬ СЧЕТЧИК ПОВТОРЕНИЙ
        BNE   RTR           ; МОЖНО ПОВТОРИТЬ ОПЕРАЦИЮ
        HALT                ; НЕИСПРАВИМАЯ ОШИБКА
; КОНСТАНТЫ
DONE=40                     ; МАСКА ДЛЯ ПРОВЕРКИ БИТА ЗАВЕРШЕНИЯ
PTRY:   0                   ; СЧЕТЧИК ПОВТОРЕНИЙ ОПЕРАЦИИ
SECTOR: 1                   ; АДРЕС СЕКТОРА НГМД
TRACK:  100                 ; АДРЕС ДОРОЖКИ НГМД
COUNT:  128.                ; ЧИСЛО БАЙТОВ В СЕКТОРЕ НГМД
; КОМАНДЫ ДЛЯ НГМД ( ДИСКОВОД 0 )
FILBUF: 1                   ; ЗАПОЛНИТЬ БУФЕР НГМД
                            ; ИЗ ПАМЯТИ МИКРО-ЭВМ
WRSECT: 5                   ; ЗАПИСАТЬ СЕКТОР ИЗ БУФЕРА НГМД
; БУФЕР ДАННЫХ
DATA:   .BLKB 128.          ; БУФЕР ДЛЯ ЗАПИСИ НА НГМД
        .END  WRNGMD
```

Рис. 10.3. Программа чтения данных с НГМД «Электроника ГМД-7012»

Р е ш е н и е. Текст требуемой программы представлен на рис. 10.3. Операторы программы снабжены комментариями и не нуждаются в более полном пояснении. Отметим лишь, что в регистр состояния заносится код операции чтения сектора НГМД (вместе с флагом пуска операции).

Затем, в случае готовности к выполнению операции, задаются адрес сектора и адрес дорожки. Эти два адреса представлены в программе константами с именами SECTOR и TRACK соответственно. Значения констант должны быть изменены при необходимости проверки других адресов сектора и дорожки.

При возникновении ошибки делается попытка повторить операцию, причем для простоты в программе используется общий для разных операций счетчик повторений. Содержимое этого счетчика формируется с отрицательным знаком, а при каждом повторе наращивается на единицу до тех пор, пока не станет нулевым. Программа не анализирует причину ошибки. При необходимости такой анализ можно провести на основе содержимого регистра ошибок НГМД. Следует обратить внимание на то, что в качестве признака завершения всех действий при чтении сектора в буфер НГМД используется появление единицы в пятом бите регистра состояния.

● **2.33.** Используя сведения о НГМД, приведенные в предыдущей задаче, написать программу записи сектора гибкого магнитного диска. Предполагается, что данные, подлежащие записи, заранее сформированы и находятся в основной памяти микро-ЭВМ. Объем этих данных равен 128 байт, т. е. числу байтов, помещающихся в одном секторе диска.

Р е ш е н и е. Текст программы приведен на рис. 10.4. Как и в предыдущей задаче, он разделен на две части: первая иллюстрирует заполнение внутреннего буфера НГМД данными из основной памяти микро-ЭВМ, вторая — запись сектора из буфера НГМД. Следует обратить внимание на то, что заполнение буфера НГМД данными из памяти микро-ЭВМ осуществляется по счетчику, так что после пересылки 128 байт эта операция считается законченной. При проверке программы на микро-ЭВМ желательно перед ее пуском занести в буфер DATA какие-нибудь данные. Отметим, что в этой и предыдущей программах коды команд НГМД справедливы для дисковода 0 (адрес дисковода необходим лишь в командах записи и чтения сектора).

```
; ПРОГРАММА ЧТЕНИЯ С НГМД 'ЭЛЕКТРОНИКА ГМД-7012'
        RXCS=177170         ; РЕГИСТР СОСТОЯНИЯ
        RXDB=177172         ; РЕГИСТР ДАННЫХ
        RXSA=177172         ; РЕГИСТР АДРЕСА СЕКТОРА
        RXTA=177172         ; РЕГИСТР АДРЕСА ДОРОЖКИ
        RXES=177172         ; РЕГИСТР ОШИБОК
        .CSECT
; ***************************************************************
; *          ПОИСК И ЧТЕНИЕ СЕКТОРА В БУФЕР НГМД                *
; ***************************************************************
RDNGMD: MOV   #-10,PTRY     ; УСТАНОВИТЬ СЧЕТЧИК ПОВТОРЕНИЙ
                            ; ОПЕРАЦИИ ПРИ ОШИБКАХ
RETRY:  MOV   RDSECT,RXCS   ; КОМАНДА ЧТЕНИЯ СЕКТОРА
S1:     TSTB  RXCS          ; ЕСТЬ БИТ ЗАПРОСА ?
        BEQ   S1            ; НЕТ ЕЩЕ
        BPL   ERROR         ; ОШИБКА
        MOVB  SECTOR,RXSA   ; ЗАДАТЬ АДРЕС СЕКТОРА
S2:     TSTB  RXCS          ; ЕСТЬ БИТ ЗАПРОСА ?
        BEQ   S2            ; НЕТ ЕЩЕ
        BPL   ERROR         ; ОШИБКА
        MOVB  TRACK,RXTA    ; ЗАДАТЬ АДРЕС ДОРОЖКИ
S3:     BIT   #DONE,RXCS    ; ОПЕРАЦИЯ ЗАВЕРШЕНА ?
        BEQ   S3            ; НЕТ ЕЩЕ
        TST   RXCS          ; УСПЕШНОЕ ЗАВЕРШЕНИЕ ?
        BMI   ERROR         ; НЕТ
        BR    OUTBUF        ; ДА. ПРИСТУПИТЬ К ВЫБОРКЕ
                            ; ДАННЫХ ИЗ БУФЕРА НГМД
ERROR:  INC   PTRY          ; МОДИФИЦИРОВАТЬ СЧЕТЧИК ПОВТОРЕНИЙ
        BNE   RETRY         ; МОЖНО ПОВТОРИТЬ ОПЕРАЦИЮ
        HALT                ; НЕИСПРАВИМАЯ ОШИБКА
; ***************************************************************
; *    ЧТЕНИЕ ДАННЫХ ИЗ БУФЕРА НГМД В ПАМЯТЬ МИКРО-ЭВМ          *
; ***************************************************************
OUTBUF: MOV   #-10,PTRY     ; УСТАНОВИТЬ СЧЕТЧИК ПОВТОРЕНИЯ
                            ; ОПЕРАЦИИ
RETUP:  MOV   #BUFFER,R0    ; R0 <--- АДРЕС ПАМЯТИ МИКРО-ЭВМ
        MOV   EMPBUF,RXCS   ; КОМАНДА ВЫБОРКИ ИЗ БУФЕРА НГМД
LP:     TSTB  RXCS          ; ЕСТЬ БИТ ЗАПРОСА ?
        BEQ   LP            ; НЕТ ЕЩЕ
        BMI   EMPTY         ; НА ВЫБОРКУ БАЙТА
        BIT   #DONE,RXCS    ; ОПЕРАЦИЯ ЗАВЕРШЕНА ?
        BEQ   LP            ; НЕТ ЕЩЕ
        TST   RXCS          ; УСПЕШНОЕ ЗАВЕРШЕНИЕ ?
        BMI   ERROR1        ; НЕТ
        HALT                ; ДА. ЧТЕНИЕ С НГМД ЗАКОНЧЕНО
ERROR1: INC   PTRY          ; МОДИФИЦИРОВАТЬ СЧЕТЧИК ПОВТОРЕНИЙ
        BNE   RETUP         ; МОЖНО ПОВТОРИТЬ ОПЕРАЦИЮ
        HALT                ; НЕИСПРАВИМАЯ ОШИБКА
; ВЫБОРКА ОЧЕРЕДНОГО БАЙТА ИЗ БУФЕРА НГМД В ПАМЯТЬ МИКРО-ЭВМ
EMPTY:  MOVB  RXDB,(R0)+
        BR    LP
;
; КОНСТАНТЫ
DONE=40                     ; МАСКА ДЛЯ ПРОВЕРКИ БИТА ЗАВЕРШЕНИЯ
PTRY:   0                   ; СЧЕТЧИК ПОВТОРЕНИЙ ОПЕРАЦИИ
SECTOR: 1                   ; АДРЕС СЕКТОРА НГМД
TRACK:  100                 ; АДРЕС ДОРОЖКИ НГМД
; КОМАНДЫ ДЛЯ НГМД ( ДИСКОВОД 0 )
RDSECT: 7                   ; ПРОЧИТАТЬ СЕКТОР В БУФЕР НГМД
EMPBUF: 3                   ; ПЕРЕДАТЬ ДАННЫЕ ИЗ БУФЕРА НГМД
                            ; В ПАМЯТЬ МИКРО-ЭВМ
; БУФЕР ДАННЫХ
BUFFER: .BLKB 128.          ; БУФЕР ДЛЯ ЧТЕНИЯ С НГМД
        .END  RDNGMD
```

Рис. 10.4. Программа записи данных на НГМД «Электроника ГМД-7012»

2.34. Объединив должным образом тексты программ из двух предыдущих задач, составить общую программу, которая последовательно обеспечивала бы заполнение буфера НГМД байтами из памяти микро-ЭВМ, запись сектора, чтение того же самого сектора и, наконец, передачу считанных байтов из внутреннего буфера НГМД в память микро-ЭВМ. Для проверки работы программы желательно, чтобы в памяти микро-ЭВМ исходные данные и данные, прочитанные с диска, запоминались в разных местах.

Программирование для микро-ЭВМ на языке Паскаль

Перед решением задач данной темы рекомендуется проработать материал § 12.4 из [57], а также [39—41]. Необходимо при этом учесть, что язык ПАСКАЛЬ, реализованный на микро-ЭВМ «Электроника-60», имеет некоторые отличия от версий, описанных в [39—41]. Эти отличия касаются главным образом организации файлов и работы с ними.

2.35. Написать программу, обеспечивающую создание файла вещественных чисел, запись заданного количества чисел в этот файл и, наконец, их чтение. Для файта зарезервировать два блока на гибком магнитном диске.

Решение. Текст программы приведен на рис. 10.5. В этой программе идентификатор F представляет собой файловую переменную, имеющую смысл только внутри данной программы и используемую для ссылки на реальный файл, организуемый на диске. Ассоциация файловой переменной F с именем реального файла обеспечивается с помощью оператора REWRITE и осуществляется на этапе выполнения программы.

При этом вслед за выводом на терминал сообщения ВЫХОДНОЙ ФАЙЛ программист должен ввести желаемое имя реального файла (не более 6 символов в соответствии с соглашениями, принятыми в ОС ФОДОС, РАФОС или ОС ДВК). Отметим, что тип реального файла явно указан в операторе REWRITE в виде параметра DAT. Таким образом, если программист введет, например, имя RAMON, то на диске будет создан файл RAMON.DAT. Конечно, если имя создаваемого файла известно на стадии написания программы, то его можно задать непосредственно в самой программе, как это сделано в данном примере для типа файла.

После того как файл создан на диске, программа цик-

```
PROGRAM REFILE;
  CONST SIZE=2;      (* ЧИСЛО БЛОКОВ В ФАЙЛЕ *)
  DLINA=10;          (* ЧИСЛО ЭЛЕМЕНТОВ ДЛЯ ЗАПИСИ-ЧТЕНИЯ *)
  X0=1.0;
  TYPE FINT=FILE OF REAL;
  VAR  I:INTEGER;
       F:FINT;
       X:REAL;
       NAME:ARRAY[1..8] OF CHAR;
  (* ОРГАНИЗАЦИЯ ФАЙЛА НА ДИСКЕ *)
  BEGIN
    REPEAT
     WRITELN ('ВЫХОДНОЙ ФАЙЛ:');
     READLN (NAME);
     I:=SIZE;
     REWRITE (F,NAME,'DAT',I);
    UNTIL I<>-1;
  (* ИНИЦИАЛИЗАЦИЯ ПЕРЕМЕННОЙ *)
  X:=X0;
  (* ЗАПИСЬ ПОСЛЕДОВАТЕЛЬНОСТИ ЧИСЕЛ В ФАЙЛ *)
  FOR I:=1 TO DLINA DO
     BEGIN
       F¬:=X;
       PUT(F); (* ЗАПИСЬ ЧИСЛА X В ФАЙЛ *)
       WRITELN( X); (* ВЫВОД ДЛЯ КОНТРОЛЯ *)
       X:=X+1.0  (* ПОДГОТОВКА ОЧЕРЕДНОГО ЧИСЛА *)
     END;
  CLOSE(F); (* ФАЙЛ ЗАКРЫТ ПОСЛЕ ЗАПИСИ *)
  WRITELN(' ЗАВЕРШИЛАСЬ ЗАПИСЬ ЧИСЕЛ В ФАЙЛ')
  RESET(F,NAME); (* ПОДГОТОВКА К ЧТЕНИЮ *)
  FOR I:=1 TO DLINA DO
     BEGIN
       X:=F¬;
       WRITELN( X); (* ВЫВОД НА ТЕРМИНАЛ *)
       GET(F)       (* ЧТЕНИЕ ОЧЕРЕДНОГО ЭЛЕМЕНТА *)
     END;
  CLOSE(F); (* ФАЙЛ ЗАКРЫТ ПОСЛЕ ЧТЕНИЯ *)
  WRITELN (' КОНЕЦ ЧТЕНИЯ')
END.
```

Рис. 10.5. Программа для создания файла, записи в него вещественных чисел и последующего их чтения из этого файла

лически записывает в него вещественные числа начиная с числа 1. Количество записываемых чисел задается константой DLINA. Буферная переменная, с помощью которой организуется «окно» для доступа к файлу, обозначена в программе F ┌, где F — файловая переменная. Подробное описание процедур работы с файлами приведено в [43].

2.36. Написать программу для последовательного чтения элементов существующего файла, имя которого на диске имеет вид NUMB.DAT. Элементами файла являются вещественные числа. Считываемые элементы файла следует выводить на системный терминал. Число элементов в файле должно быть определено в зависимости от числа блоков, занятых файлом на диске. Напомним, что размер блока равен 512 байт, причем одно вещественное число хранится в четырех последовательных байтах.

2.37. Написать программу, которая должна использовать существующий файл с именем INT1·DAT на диске и создать новый файл INT2. DAT. Элементы обоих файлов должны иметь целочисленный тип. Программа должна последовательно читать числа из первого файла, возводить их в квадрат и записывать во второй (т. е. вновь созданный) файл. Предельное число элементов в первом файле можно определить в зависимости от числа блоков, занятых файлом, с учетом того, что для хранения каждого целого числа в файле требуется 2 байт.

2.38. Составить программу, которая должна прочитать заданное число элементов файла целого типа в одномерный массив и затем обеспечить нахождение и вывод на терминал минимального и максимального чисел в полученном массиве. Для программирования можно выбрать произвольное имя реального файла.

Таблица 10.1

Действия программиста или реакция системы	Содержание действия или реакции
.R SCREEN	Вызов текстового редактора
ВХОДНОЙ ФАЙЛ>	Запрос редактором имени входного файла (так как программа создается впервые, то входной файл не существует)
ВЫХОДНОЙ ФАЙЛ> MXO:BANS PAS	Выходному файлу присвоено имя BANS.PAS, файл будет создан на устройство MXO

Продолжение табл. 10.1

Действия программиста или реакция системы	Содержание действия или реакции
Последовательность операторов программы на языке ПАСКАЛЬ, заканчивающаяся оператором END	
↓	Команда редактору запомнить исходную программу в файле BANS.PAS
.O	
.R PASCAL	Вызов транслятора с языка ПАСКАЛЬ
*BANS.MAC, BANS.LST=BANS.PAS	Команда транслятору: транслировать исходную программу, находящуюся в файле BANS.PAS, результат трансляции поместить в файл BANS.MAC, а листинг направить в файл BANS.LST
.MACRO BANS.MAC	Вызов макроассемблера для трансляции модуля, находящегося в файле BANS.MAC (в результате получится объектный модуль в файле BANS.OBJ, файл с листингом не создается)
ERRORS DETECTED:0	Сообщение транслятора об отсутствии ошибок
.LINK BANS.OBJ, PASCAL.OBJ	Вызов компоновщика для компоновки объектного модуля с объектными модулями из файла PASKAL.OBJ (в результате получится загрузочный модуль BANS.SAV)
.PRUN BANS.SAV	Запуск загрузочного модуля

2.39. Написать рекурсивную функцию вещественного типа для вычисления $X^n = X^{n-1} \cdot X$, где X — некоторая вещественная переменная, а n — целая степень. В теле рекурсивной функции должно быть по крайней мере одно обращение функции к самой себе.

2.40. Написать процедуру, реализующую суммирование компонентов вектора вещественного типа. В качестве параметров процедуры использовать вектор, компоненты которого подлежат суммированию, число компонент вектора и результат суммирования.

● **2.41.** Записать в виде последовательности мониторных команд этапы составления, трансляции, редактирования и запуска программы на языке ПАСКАЛЬ в операционной системе ОС ДВК.

Р е ш е н и е. Для составления исходной программы можно использовать любой из текстовых редакторов (например, реактор с именем SCREEN). В данном случае для решения поставленной задачи должны быть выполнены действия, представленные в табл. 10.1.

П р и м е ч а н и е. Программу, написанную на языке ПАСКАЛЬ, необходимо транслировать дважды. После первой трансляции, осуществляемой транслятором PASCAL, получается промежуточный текст программы на языке ассемблера. Вторая трансляция обеспечивается макроассемблером, в результате чего получается объектный модуль. И только после этого осуществляется компоновка объектного модуля, во время которой могут потребоваться стандартные объектные модули из файла PASCAL.OBJ.

§ 10.3. Элементная база и схемотехника средств сопряжения

Схемотехника цифровых и аналоговых ИС малой, средней и большой степени интеграции ИС, СИС, БИС

Опыт работы показывает, что без знания основ схемотехники ИС, СИС и БИС нельзя реализовать надежно функционирующую микро-ЭВМ или систему обработки данных.

● **3.1.** Объяснить, о чем говорят знаки «+» и «—» перед числовыми значениями параметров $I^1_{вх}$ и $I^0_{вх}$ схем ТТЛ-типа, например $I^1_{вх} = +0{,}04$ мА, $I^0_{вх} = -1{,}6$ мА.

Р е ш е н и е. Знак «+» перед числовым значением $I^1_{вх}$ схем ТТЛ-типа говорит о том, что этот ток выходит из управляющей схемы и входит в управляемую схему. Наличие тока $I^1_{вх}$ является причиной снижения уровня напряжения $U^1_{вых}$.

Знак «—» перед числовым значением тока $I^0_{вх}$ схем ТТЛ-типа говорит о том, что входной ток выходит из управляемой схемы и входит в управляющую схему. Наличие тока $I^0_{вх}$ является причиной повышения уровня напряжения $U^0_{вых}$.

3.2. Сравнить технологию и схемотехнику p-МДП-, n-МДП-, КМДП-, $И^2Л$-, ТТЛШ-, ЭСЛ-типа по следующим критериям: простота технологии, быстродействие элемента базовой схемы, степень интеграции, мощность рассеяния, тип и число источников питания, стыковка со стандартными уровнями сигналов схем ТТЛ-типа.

3.3. Объяснить, почему при разработке микро-ЭВМ еще широко применяются интегральные схемы малой и средней степени интеграции.

3.4. Объяснить, почему в схемах $И^2Л$-типа нагрузки развязываются друг от друга с помощью многоколлекторных транзисторов.

3.5. Указать, для каких целей разработаны логические схемы с тремя состояниями выхода. Что это за состояния?

3.6. Перечислите достоинства и недостатки схемы «монтажное **И**» (для положительной логики) для схем ТТЛ-типа, выполненных с открытым коллектором и с тремя состояниями выхода.

3.7. Какие виды нагрузок применяют для ключевых транзисторов, используемых при разработке схем *p*-МДП- и *n*-МДП-типа?

3.8. Изучить параметры операционных усилителей (ОУ), приводимые в справочниках или другой нормативной документации. Разработать схемы для измерения этих параметров. Оценить влияние данных параметров на работу различных схем, реализуемых с использованием ОУ.

3.9. Каким недостатком обладает открытый ключ, выполненный на биполярном транзисторе, по сравнению с открытым ключом, выполненным на униполярном транзисторе?

3.10. Какие параметры входного сигнала, ключа и ОУ определяют номинал конденсатора *C* в схеме выборки — хранения, показанной на рис. 10.6?

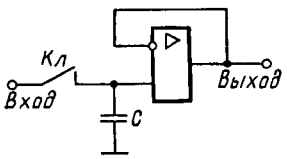


Рис. 10.6. Схема выборки — хранения

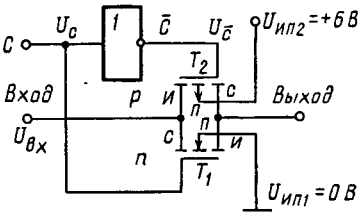


Рис. 10.7. Двунаправленный КМДП-ключ со схемой управления

● **3.11.** Проанализировать характер изменения сопротивлений транзисторов с каналами *p*- и *n*-типа и общее сопротивление двунаправленного ключа КМДП-типа от входного напряжения (рис. 10.7) для следующих числовых значений параметров: $U_{ип1}=0$ В; $U_{ип2}=+6$ В; $U_c=+6$ В (напряжение на управляющем входе); $U_{порп}=+2$ В, $U_{порр}=-1{,}5$ В (пороговые напряжения соответ-

ственно транзисторов T_1 с каналом n-типа и T_2 с каналом p-типа); $K_n = 2$ мА/В^2, $K_p = 1{,}7$ мА/В^2 (соответствующая удельная крутизна транзисторов T_1 и T_2). При анализе токами утечки закрытых транзисторов пренебречь, считать, что $R_н \gg R_о$, где $R_н$ — сопротивление нагрузки на выходе, $R_о$ — сопротивление открытого ключа.

Р е ш е н и е. В общем случае двунаправленный ключ управляется напряжениями, действующими на затворах транзисторов и принимающими значения $U_{\bar{С}} = U_{ип1}$, $U_{\bar{С}} = U_{ип2}$, когда ключ закрыт, и $U_{\bar{С}} = U_{ип2}$, $U_{\bar{С}} = U_{ип1}$, когда ключ открыт. В нашем случае $U_{\bar{С}} = +6$ В, $U_{\bar{С}} = 0$ В. Из условия $R_н \gg R_0$ следует, что транзисторов T_1 и T_2 справедливо соотношение $U_{СИ} = 0$, из которого, в свою очередь, следует вывод, что проводящий транзистор работает в крутой области выходных характеристик, описываемой уравнением Хофстайна [57]

$$I_c = K[(U_{ЗИ} - U_{пор})U_{СИ} - U^2_{СИ}/2] \text{ при } 0 \leqslant |U_{СИ}| \leqslant |U_{ЗИ} - U_{пор}|; \quad |U_{ЗИ}| > |U_{пор}|; \qquad (10.1)$$

Для транзисторов n- и p-типа уравнение (10.1) примет вид соответственно

$$I_{Cn} = K_n[(U_С - U_{вх} - U_{порn})U_{СИ} = U_{СИ}/2]; \qquad I_{Cp} = K_p[(U_{\bar{С}} - U_{вх} - U_{порp})\times U_{СИ} - U^2_{СИ}/2]. \qquad (10.2)$$

Продифференцировав (10.2) по $U_{СИ}$ в точке $U_{СИ} = 0$, получим

$$1/R_n = K_n(U_{СИ} - U_{вх} - U_{порn}); \; 1/R_p = K_p(U_{\bar{С}} - U_{вх} - U_{порp}). \qquad (10.3)$$

Так как для открытого КМДП-ключа $U_С = U_{ип2}$, $U_С = U_{ип1}$, из (10.3) следует

$$R_n = 1/[K_n(U_n(U_{ип2} - U_{вх} - U_{порn})]; \; R_p = 1/[K_p(U_{ип1} - U_{вх} - U_{порp})]. \qquad (10.4)$$

На рис. 10.8 приведены зависимости R_n, R_p и $R_{кл}$ от $U_{вх}$, откуда видно, что в диапазоне $U_{ип1} < U_{вх} < U_{ип1} - U_{порp}$ сопротивление КМДП-ключа определяется сопротивлением канала открытого транзистора n-типа; в диапазоне $U_{ип1} - U_{порp} < U_{вх} < U_{ип2} - U_{порn}$ — параллельно включенными сопротивлениями каналов открытых транзисторов n- и p-типа, а в диапазоне $U_{ип2} - U_{порn} < U_{вх} < U_{ип2}$ — сопротивлением канала открытого транзистора p-типа.

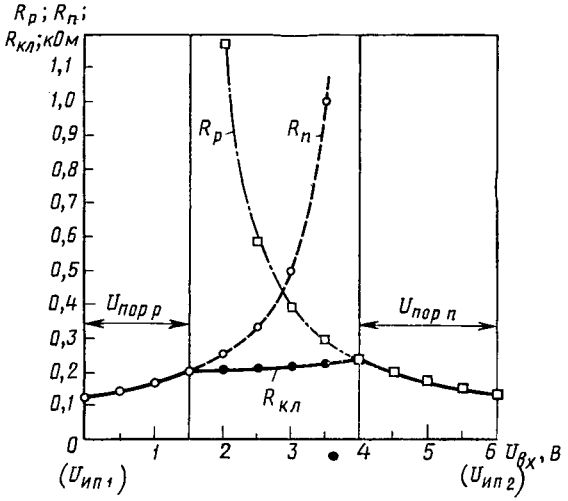


Рис. 10.8. Зависимости сопротивлений транзисторов с каналами *p*- и *n*-типа и общего сопротивления двунаправленного ключа КМДП-типа

Узлы цифровых и цифроаналоговых устройств микро-ЭВМ

Большинство узлов микро-ЭВМ являются многофункциональными устройствами, что иллюстрируется нижеследующими примерами, задачами и вопросами.

● **3.12.** Разработать двухфазный генератор, свободный от паразитных импульсов, возможных из-за перекоса синхронизации (показаны пунктиром на рис. 10.9, *б*) в схеме двухфазного генератора, приведенной на рис. 10.9, *а*.

Р е ш е н и е. Паразитные импульсы на выходах $Ф_1$ и $Ф_2$ схемы рис. 10.9, *а* возможны из-за логических го-

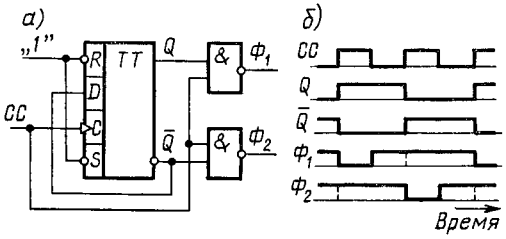


Рис. 10.9. Схема (*а*) и временные диаграммы (*б*) двухфазного генератора

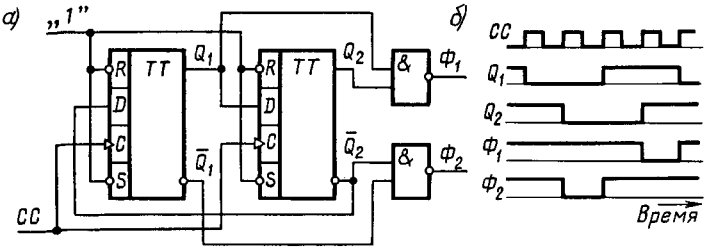


Рис. 10.10. Схема (*а*) и временны́е диаграммы (*б*) двухфазного генератора, свободного от паразитных импульсов

нок «одновременно» изменяющихся сигналов на входах соответствующих схем И — НЕ. На рис. 10.10, *а* приведена схема 2-разрядного синхронного счетчика, изменяющего свои состояния в коде Грея. Работа схемы поясняется временны́ми диаграммами (рис. 10.10, *б*), из которых видно, что меняет свое состояние с приходом синхросигнала только один триггер, что обеспечивает устранение логических гонок.

3.13. Объяснить, что такое система синхронизации, какие функции она выполняет.

3.14. Объяснить, что такое перекос синхронизации.

3.15. Используя динамические параметры микросхем микропроцессорного комплекта, определить максимальную частоту синхронизации для построенной на нем микро-ЭВМ.

3.16. На основе схемы, приведенной на рис. 10.10, *а*, разработать четырехфазный генератор синхросигналов, в котором отсутствуют паразитные импульсы.

3.17. Разработать схему, производящую операцию нормализации исходного двоичного 4-разрядного числа (старший разряд — знаковый), заданного в дополнительном коде. Получить: двоичный дополнительный код мантиссы, код характеристики (порядок, заданный в прямом двоичном коде), потенциальный сигнал, осведомляющий об окончании работы схемы. Запускается схема одиночным импульсом C_1, подаваемым во время низкого уровня синхросигнала *СС*, используемого для синхронизации работы схемы (рис. 10.11).

3.18. Разработать на основе арифметического расширителя (сдвигателя) КР1802ВР1 32-разрядный сдвига-

тель, предварительно ознакомившись с его функциональными возможностями.

3.19. Разработать дешифратор «1 из 4», пользуясь только инверторами и двухвходовыми вентилями И — НЕ. Какой активный уровень будет на выходах этого дешифратора?

3.20. Разработать дешифратор «1 из 8», пользуясь только инверторами и двухвходовыми вентилями И. Какой активный уровень будет на выходах этого дешифратора?

3.21. Разработать селектор «из 8 в 8», используя микросхемы К155КП7 и К155ИД4, работающий в режиме «из i-го входа на i-й выход». Что надо сделать для обеспечения режима «из i-го входа на j-й выход» ($i \neq j$ или $i = j$)?

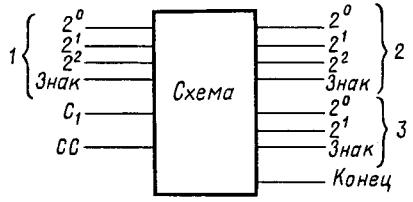


Рис. 10.11. Схема нормализации двоичного четырехразрядного числа

3.22. Реализовать мажоритарный элемент на мультиплексоре «4→1».

3.23. Как использовать микросхему К155ИД3 в режиме селектора «1→16»?

3.24. Разработать, используя микросхему 564КП2: а) дешифратор «1 из 8» с активным уровнем выхода «1»; б) дешифратор «1 из 8» с активным уровнем выхода «0»; в) мультиплексор «8→1»; г) селектор «1→8».

3.25. Разработать, используя микросхемы 564КП2, селектор «из 8 в 8» по требованиям, изложенным в задаче 3.21.

● **3.26.** Разработать преобразователь прямого кода целого числа со знаком в дополнительный, используя микросхему АЛУ К155ИП3. Коды 8-разрядные: восьмой (старший) разряд — знаковый.

Р е ш е н и е. Схема преобразователя прямого кода в дополнительный приведена на рис. 10.12. Через x_i и y_i обозначены соответственно преобразуемое и преобразованное числа. Восьмые разряды (x_7, y_7) являются знаковыми, а младший разряд соответствует первому разряду

числа (x_0, y_0). Логические сигналы на входах S_i, определяющих тип операции, имеют следующие значения: $S_0=0$, $S_1=S_2=S_3=1$. Значение логического сигнала на входе модификатора M равно $\overline{x_7}$.

Если преобразуемое число положительно ($x_7=0$), то АЛУ выполняет логическую функцию $F=A+B$ (см. табл. 2.5 [38]), а так как $A=0$, то окончательно $F=B$, т. е. дополнительным кодом положительного целого числа является само это число.

Если преобразуемое число отрицательно ($x_7=1$), то АЛУ выполняет функцию $F=(\mathrm{A}+\overline{\mathrm{B}})$ плюс A плюс 1, а так как $A=0$, то окончательно $F=\overline{\mathrm{B}}$ плюс 1, что и является определением дополнительного кода для отрицательного числа.

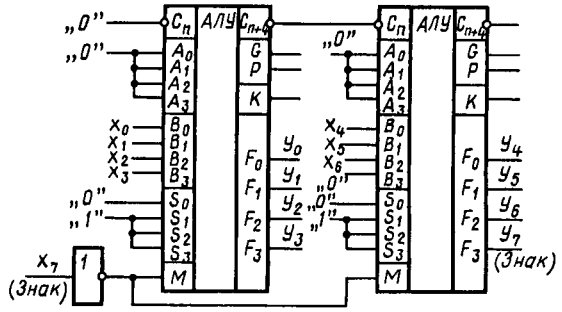


Рис. 10.12. Схема преобразователя прямого кода в дополнительный

3.27. Может ли схема, приведенная на рис. 10.12, выполнять функцию преобразователя из дополнительного кода в прямой?

3.28. Разработать преобразователи: а) из прямого кода в обратный; б) из обратного кода в прямой; в) из дополнительного кода в обратный; г) из обратного кода в дополнительный, используя микросхему АЛУ К155ИП3.

3.29. Объяснить, что такое элемент полупроводниковой памяти. Чем принципиально отличается элемент полупроводниковой памяти от простейших триггерных схем, выполненных на логических элементах?

3.30. Объяснить, что такое однокоординатная выборка ячейки полупроводникового ЗУ. До какого объема ЗУ целесообразно ее использовать?

3.31. Объяснить, что такое двухкоординатная выборка ячейки полупроводникового ЗУ.

3.32. Как организовать ЗУ с последовательной выборкой информации?

3.33. Изучить режимы программирования ППЗУ 155РЕ3, КР556РТ4, КР556РТ5.

3.34. Объяснить, что такое программируемые логические матрицы. Как они организованы, как программируются?

3.35. Спроектировать ОЗУ объемом 1024 слова по 16 бит в каждом из модулей на 256 слов по 4 бит.

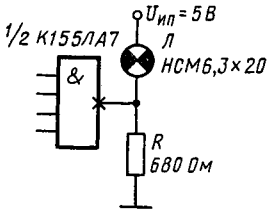


Рис. 10.13. Схема двоичного индикатора на лампе накаливания

3.36. Изучить временные соотношения при выполнении операций чтения из памяти и записи в память.

3.37. Объяснить, как организуются временные соотношения при выполнении операций чтения из памяти и записи в память при взаимодействии микропроцессора с ОЗУ.

3.38. Перечислить достоинства и недостатки полупроводникового ОЗУ статического и динамического типа.

3.39. Изучить организацию динамического ОЗУ типа 565РУ1.

3.40. Объяснить, что такое регенерация информации в динамических ОЗУ. Привести пример регенерации информации для ОЗУ, выполненного на микросхемах 565РУ1.

3.41. Перечислить с точки зрения разработчика достоинства и недостатки заказных БИС на основе нескоммутированных логических матриц.

3.42. Перечислить все известные вам типы двоичных индикаторов, часто используемых в виде простейших периферийных устройств микро-ЭВМ.

3.43. Объясните назначение резистора *R* в схеме, приведенной на рис. 10.13.

3.44. Объяснить, почему в качестве двоичных индикаторов широкое распространение получили полупроводниковые излучающие диоды.

3.45. Как организовать мультиплексное управление многоразрядными семисегментными индикаторами?

3.46. Разработать на основе 8-разрядного цифро-бук-

венного мозаичного индикатора с организацией одного разряда 5×7 точек индикатор типа «бегущая строка». Сделайте самостоятельно.

3.47. Разработать на основе линейчатого индикатора, выполненного из 64 изолированных друг от друга излучателей, дешифратор для реализации индикатора «шкала с заполнением».

3.48. Определить аналого-цифровые (АЦП) и цифро-аналоговые (ЦАП) преобразователи. Какие двоичные коды используют для построения ЦАП?

3.49. Сформулировать требования к источнику опорного напряжения, входящего в состав ЦАП.

3.50. Оценить достоинства и недостатки двоичновзвешенной матрицы резисторов и матрицы типа $R - 2R$, используемых при построении ЦАП и АЦП.

3.51. Перечислить методы аналого-цифрового преобразования.

3.52. Дать определения следующих составляющих погрешности ЦАП: а) погрешность масштабирования; б) погрешность линейности; в) погрешность монотонности.

3.53. Дать определение времени преобразования ЦАП и АЦП? Определить время преобразования для различных методов, используемых при построении АЦП.

3.54. Объяснить, как можно использовать ЦАП в качестве множительного устройства. В каком виде представляются сомножители и результат?

3.55. Объяснить, как можно расширить функциональные возможности ЦАП, если в вашем распоряжении имеются аналоговые коммутаторы и схемы выборки — хранения.

3.56. Воспроизвести операцию деления на основе схемы АЦП.

3.57. Совместить функции множительного и делительного устройств в рамках единого аналого-цифрового множительно-делительного устройства (АЦМДУ).

3.58. Разработать АЦМДУ на основе микросхемы 572ПВ1.

3.59. Можно ли АЦМДУ отнести к классу микропроцессоров?

Средства автоматического ввода данных в системы сбора и обработки информации

В этих системах широко используются как цифровые, так и линейные элементы.

3.60. Объяснить, что такое система сбора и обработки данных (СОД). Перечислить основные составные части и указать связи между ними.

3.61. В чем заключается принцип параллельности обработки аналоговых сигналов, поступающих с первичных датчиков?

3.62. Каково предельное число каналов СОД при параллельной обработке сигналов?

3.63. В чем заключается принцип последовательной обработки аналоговых сигналов, поступающих с первичных датчиков?

3.64. Каково предельное число каналов СОД при последовательной обработке сигналов?

3.65. Перечислить семь основных функциональных устройств СОД.

3.66. Классифицировать первичные датчики СОД по типу измеряемых параметров.

3.67. Как осуществляется согласование выходных цепей первичных датчиков с входными цепями СОД?

3.68. Перечислить методы линеаризации характеристики преобразования первичных датчиков (самостоятельно).

3.69. Разработать следующие функциональные преобразователи: 1) $y_1 = \sin x$; 2) $y_2 = \cos x$; 3) $y_3 = x^2$, используя обобщенную схему функционального преобразователя ПЗУ-ЦАП. Запрограммировать соответствующие ПЗУ.

Объяснить, чем определяется разрядность ПЗУ и разрядность ЦАП (самостоятельно).

3.70. Как можно уменьшить требуемый объем ПЗУ при решении задачи 3.69, если использовать свойство периодичности реализуемой функции? (самостоятельно).

3.71. Разработать устройство, вырабатывающее напряжение с частотой $f = 400$ Гц на выходе данного устройства с цифровой установкой амплитуды A выходного напряжения и его фазы φ, если имеется источник опорного напряжения $U_{оп}$, $f = 400$ Гц; $0 < A < \sqrt{2} U_{оп}$; $0 < \varphi < 360°$. Точность установки A не хуже 1 % от $\sqrt{2}\ U_{оп}$, φ не хуже 1 °.

3.72. Объяснить, для чего используется фильтрация сигналов, снимаемых с первичных датчиков СОД.

3.73. Спроектировать низкочастотный фильтр Баттерворта второго порядка с частотой среза 500 Гц и усилением 10.

3.74. В чем достоинство цифровых фильтров?

3.75. Каковы достоинства и недостатки рекурсивных и нерекурсивных фильтров?

3.76. Указать способы восстановления аналоговых сигналов.

3.77. Каким требованиям должен удовлетворять фильтр, сглаживающий восстановленный аналоговый сигнал?

Средства сопряжения устройств ввода с микро-ЭВМ

Приведенные задачи помогают в разработке средств интерфейса, необходимых при создании микро-ЭВМ с полезными потребителю свойствами.

● **3.78.** Определить функцию, получаемую на линии общего выхода двух элементов ТТЛ-типа с открытым коллектором, реализующих операцию И — НЕ (рис. 10.14, *а*).

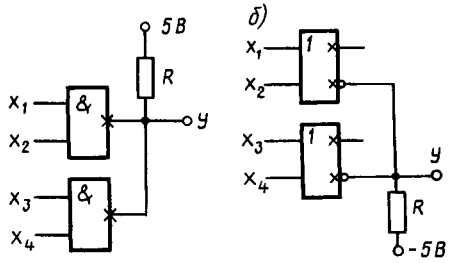


Рис. 10.14. Схема монтажного псевдоэлемента на логических элементах с открытым коллектором ТТЛ-типа (*а*) и с открытым эмиттером ЭСЛ-типа (*б*)

Решение. Схема реализует монтажную операцию И над переменными, в качестве которых рассматриваются сигналы на выходах двух вентилей. Обозначим эти сигналы y_1 и y_2; следовательно, $y=y_1y_2$, а так как $y_1=\overline{x_1x_2}$, $y_2=\overline{x_3x_4}$, то $y=\overline{x_1x_2}\cdot\overline{x_3x_4}=\overline{x_1x_2}+\overline{x_3x_4}$, т. е. реализуется операция 2И — 2ИЛИ — НЕ.

● **3.79.** Определить функцию, реализуемую на линии общего выхода двух элементов ЭСЛ-типа с открытыми эмиттерами, реализующих операцию ИЛИ — НЕ (рис. 10.14,*б*).

Решение. Схема реализует монтажную операцию ИЛИ над переменными, в качестве которых рассматриваются сигналы на выходах двух вентилей. Обозначим эти сигналы y_1 и y_2; следовательно, $y=y_1+y_2$, а так как $y_1=\overline{x_1+x_2}$, $y_1=\overline{x_3+x_4}$, то $y=\overline{x_1+x_2}+\overline{x_3+x_4}=\overline{(x_1+x_2)\,(x_3+x_4)}$, т. е. реализуется операция 2ИЛИ—2И—НЕ.

3.80. Дать определение линии связи.

3.81. Дать определение управляемой линии связи.

3.82. Дать определение подмагистрали.

3.83. Дать определение управляемой подмагистрали.

3.84. Дать определение магистрали. В чем заключается принцип обмена информацией по методу квитирования?

3.85. Какие аппаратурные средства используются для гальванической развязки передатчика и приемника информации?

3.86. Где используются двунаправленные буферные формирователи?

3.87. Дать определение «интерфейс». Какие преимущества обеспечиваются разработчику, применяющему стандартные интерфейсы?

3.88. Как классифицируются интерфейсы по способу передачи информации?

3.89. Чем отличаются синхронные интерфейсы от асинхронных?

3.90. Изучите техническую документацию на стандартные интерфейсы.

3.91. Что такое преобразователи уровней?

3.92. Перечислить основные базовые схемы построения преобразователей уровней.

3.93. Разработать (или выбрать из числа стандартных микросхем) преобразователи уровней: ЭСЛ→ТТЛ, ТТЛ→ЭСЛ, ЭСЛ→КМДП, КМДП→ЭСЛ, ТТЛ→КМДП, КМДП→ТТЛ.

3.94. Разработать универсальный преобразователь со следующими параметрами: входные уровни в диапазоне ± 12 В, выходные уровни соответствуют схемам ТТЛ-типа.

§ 10.4. Основы построения микро-ЭВМ

Ниже представлены задачи, раскрывающие особенности построения узлов и блоков микро-ЭВМ на основе различных микропроцессорных БИС. Задачи разбиты по разделам в зависимости от типа микропроцессорной БИС: однокристальная микропроцессорная БИС (серия К580), секционируемая микропроцессорная БИС (серия К589) и микропроцессорная БИС с асинхронным устройством управления, реализованным на основе программируемой логической матрицы ПАМ (серия К587, К588), на основе которой строится микро-ЭВМ.

Особенности построения микро-ЭВМ на микропроцессорном комплекте серии К580

Успешное овладение методами построения микро-ЭВМ на микропроцессорном комплекте серии К580 возможно лишь при детальном рассмотре-

нии как схемотехнических вопросов построения, так и программных методов организации обмена информацией для каждой ИС, входящей в микропроцессорный комплект. Приведенные в разделе примеры и задачи затрагивают основные особенности использования каждой ИС при решении практических задач. Многие из примеров и задач могут быть использованы при рассмотрении способов построения микро-ЭВМ и на других микропроцессорных комплектах.

● **4.1.** Изобразить схему формирования магистрали управления для микро-ЭВМ на основе БИС КР580ИК80, определить время формирования слова состояния после начала любого цикла.

Решение. Один из вариантов формирования магистрали управления представлен в [57]. Сигналы управления на выходе регистра слова состояния формируются через время t_c после начала цикла по положительному фронту второго синхроимпульса Φ_1 с задержкой на время записи информации в регистр слова состояния ($t_{рсс}$), т. е. $t_c = T_1 + t_{рсс}$.

4.2. Изобразить схемы формирования магистралей микро-ЭВМ, построенной на МП БИС КР580ИК80. При разработке схемы учесть необходимость перевода магистралей микро-ЭВМ в третье состояние при работе МП БИС в режиме ЗАХВАТ. Для формирования магистрали данных (МД) и магистрали управления (МУ) использовать ИС КР580ВК28.

4.3. Разработать схему, позволяющую выполнять МП БИС КР580ИК80 программу по командам, используя режим ожидания. Выполнение очередной команды программы инициируется нажатием на клавишу «Шаг команды».

4.4. Перечислить режимы работы МП БИС КР580ИК80, привести алгоритм их изменения, указать структурные схемы подключения устройств для осуществления различных режимов работы МП БИС.

4.5. Разработать схему для моделирования статических сигналов МП БИС КР580ИК80. Схема используется для отыскания неисправностей в микропроцессорных системах путем исключения из них МП БИС и формирования статических сигналов на магистралях микро-ЭВМ.

4.6. Изобразить схему для формирования импульса начальной установки МП БИС КР580ИК80.

4.7. Какие параметры питающих напряжений и в какой последовательности напряжения должны подаваться на МП БИС КР580ИК80 для правильного ее функционирования?

4.8. При работе микропроцессорных систем часто встает задача диагностики правильности функционирова-

ния отдельных их узлов. Такую проверку можно осуществить программными методами при начальном включении систем. Для этого в программы начальной установки включают подпрограммы диагностики их основных узлов. В число проверяемых узлов могут входить МП БИС, ОЗУ, ПЗУ, внешние устройства и т. д. Разработать программу проверки: 1) выполнения МП БИС КР580ИК80 основных арифметических и логических операций; 2) правильности осуществления операций записи/считывания информации МП БИС в каждую ячейку ОЗУ. По окончании проверки программа должна обнулить все ячейки ОЗУ (ОЗУ занимает адреса с 0800_{16} до OBFF); 3) информации, записанной в ПЗУ и занимающей адреса с 0000 до 07FF. Проверку осуществить по контрольной сумме, записанной по адресу 07FF.

4.9. Привести схему подключения клавиатуры, организованной в виде матрицы 8×3, к магистралям микро-ЭВМ. Разработать алгоритм и записать программу для определения и дешифрации нажатой клавиши, основанный на сканировании клавиатуры. Код нажатой клавиши должен быть записан в аккумуляторе МП БИС. Дешифрация нажатой клавиши осуществляется программно.

4.10. Привести схему подключения дисплея, состоящего из шести семисегментных индикаторов к магистралям микро-ЭВМ. Разработать алгоритм и записать программу для вывода информации на дисплей, основанный на сканировании ячеек дисплея. Коды цифр, выводимых на дисплей, записаны в шести определенных адресах памяти. При кодировании информации для вывода на дисплей используется специальная таблица.

4.11. Разработать программу вывода содержимого адреса памяти на дисплей микро-ЭВМ. В микро-ЭВМ применяется дисплей, состоящий из шести ячеек, четыре из которых используются для указания адреса памяти, а оставшиеся две для отображения его содержания. Программа вызывается нажатием на клавиатуре (рассмотренной в задаче 4.9) на клавишу «Отыскание адреса» (ОТА). После вызова программы микро-ЭВМ должна реагировать лишь на нажатие цифровых клавиш. Каждое нажатие должно сопровождаться записью цифры, соответствующей нажатой клавише, в младший разряд дисплея адреса и смещением цифр на дисплее адреса на 1 разряд влево. После введения с клавиатуры кода адреса микро-ЭВМ должна вывести число, записанное по адресу на дисплей данных.

При записи программы используйте результаты, полученные в задаче 4.10.

4.12. Разработать схему сопряжения микро-ЭВМ с устройствами ввода — вывода на базе БИС КР580ВВ55 и написать программу начальной установки БИС КР580ВВ55 для осуществления обмена информацией между устройствами ввода и вывода в синхронном режиме. В качестве устройства ввода — вывода использовать наборное поле из восьми двухпозиционных переключателей и две светодиодные линейки по восемь точечных светодиодов в каждой.

4.13. 1. Разработать на базе КР580ВВ55 схему подключения к микро-ЭВМ дисплея, состоящего из восьми светодиодных семисегментных матриц. 2. Написать программу вывода информации с использованием режима сканирования дисплея. Коды выводимых цифр расположены в восьми последовательных ячейках памяти.

4.14. 1. Разработать на базе БИС КР580ВВ55 схему сопряжения микро-ЭВМ с восьмиразрядными ЦАП и АЦП. Обмен осуществлять в асинхронном режиме. 2. Написать программу управления обменом, обеспечивающую формирование сигнала и строба ЦАП, с использованием режима установки/сброса отдельных разрядов канала СБИС КР580ВВ55. По переднему фронту сигнала производится запись информации в регистр ЦАП.

4.15. 1. Разработать на базе БИС КР580ВВ55 схему сопряжения микро-ЭВМ с алфавитно-цифровым дисплеем. 2. Написать программу, обеспечивающую асинхронный обмен данными.

Примечание. Сигналами интерфейса дисплея являются:
входные сигналы:
STBI (стробирование входа) — указывает на наличие данных на информационных входах дисплея при вводе данных;
DMDO (запрос на выдачу данных) — приемник информации готов принять данные;
выходные сигналы:
STBO (стробирование входа) — указывает на наличие данных на выходах дисплея при выводе из него информации;
DMDI (запрос на ввод данных) — дисплей готов принять данные.

4.16. 1. Разработать схему подключения БИС КР580ВВ55 к микро-ЭВМ, позволяющую наблюдать на экране осциллографа временные диаграммы данной БИС в режиме одно- и двунаправленного обмена. 2. Написать программу, обеспечивающую формирование сигналов СТБ и ПРИЕМ с использованием режима установки/

сброса отдельных разрядов канала С БИС КР580ВВ55. Эти сигналы необходимы для получения устойчивой картины на экране осциллографа и подаются на соответствующие разряды канала С.

● **4.17.** 1. Разработать схему обмена информацией в параллельном формате между двумя микро-ЭВМ с использованием БИС КР580ВВ55. Устройство сопряжения должно обеспечивать двунаправленный обмен информацией по каналу связи, в котором одна микро-ЭВМ является ведущей, а другая — ведомой. В качестве сигналов управления использовать разряды канала С. 2. Разработать про-

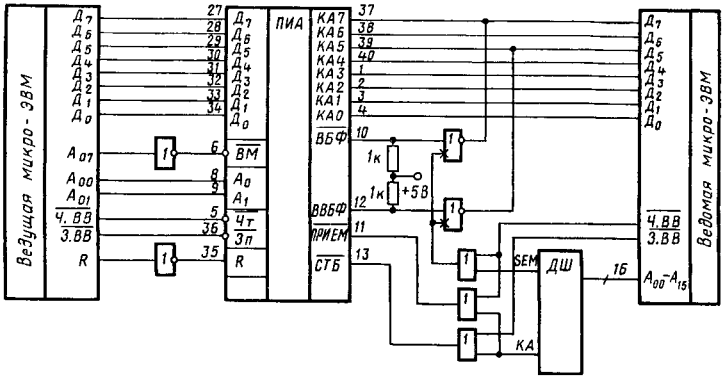


Рис. 10.15. Схема сопряжения двух микро-ЭВМ на основе БИС КР580ВВ55

грамму, обеспечивающую работу двупроцессорной системы в указанном режиме, по следующему алгоритму:

1. Передача массива из ведущей в ведомую.
2. Обработка массива в ведомой.
3. Передача обработанного массива из ведомой в ведущую.

Р е ш е н и е. Схема одного из вариантов решения данной задачи приведена на рис. 10.15.

Ведомая микро-ЭВМ связывается с буфером через схему управления посредством сигналов СТБ и ПРИЕМ. Импульс СТБ записывает информацию из ведомой микро-ЭВМ в буфер и одновременно устанавливает в «1» сигнал ВВБФ (ввода в буфер). Импульс ПРИЕМ разрешает выдачу данных из буфера и устанавливает в «1» сигнал ВБФ (вывод в буфер).

При передаче массива данных из ведущей в ведомую каждой микро-ЭВМ проверяется только состояние сигнала ВБФ. После обработки в ведомой массив передается назад, в ведущую микро-ЭВМ. При этом проверяется сигнал ВВБФ.

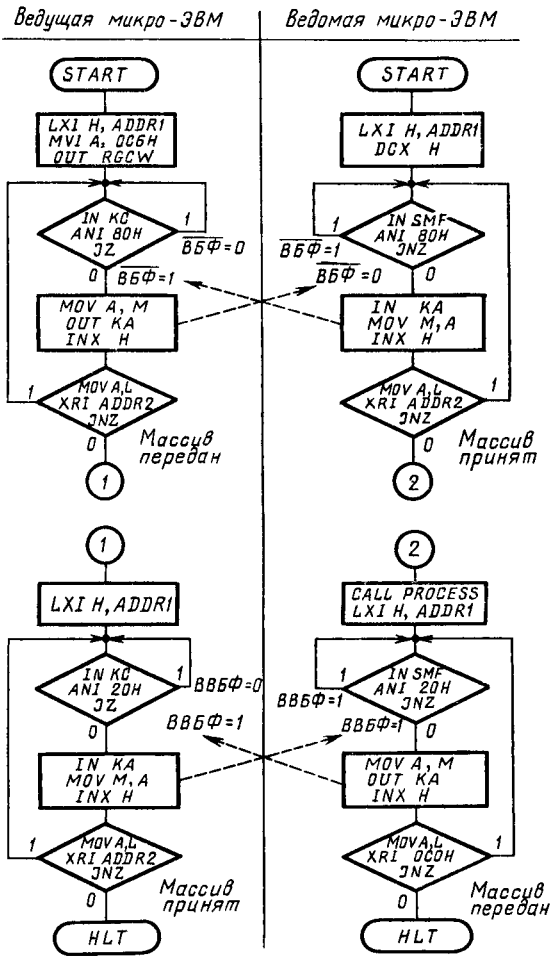


Рис. 10.16. Алгоритм организации обмена информацией между двумя микро-ЭВМ

Схемы программ, необходимых для реализации такого обмена, представлены на рис. 10.16.

В начале цикла вывода ведущая микро-ЭВМ проверяет состояние сигнала ВБФ, определяя таким образом, запол-

нен или нет регистр канала А. Если он пуст, выполняется команда OUT КА. При этом данные из аккумулятора ведущей микро-ЭВМ передаются в регистр КА и сигнал ВБФ устанавливается в «0». Ведомая микро-ЭВМ использует команду IN SEM для проверки состояния управляющих сигналов. Если установлен сигнал ВБФ, ведомая микро-ЭВМ по команде IN КА считывает данные из регистра КА. Таким образом весь массив передается в память ведомой микро-ЭВМ. Она производит их обработку (в подпрограмме PROCESS), после чего начинает циклически проверять состояние сигнала ВВБФ. Тем самым она определяет, свободен или нет регистр КА. Если он пуст, то по команде OUT КА ведомая передает данные из аккумулятора в регистр КА, устанавливая при этом сигнал ВВБФ в «1». Ведущая микро-ЭВМ в это время проверяет состояние данного сигнала и при его установке считывает информацию из регистра КА. После выполнения команды чтение IN КА сигнал ВВБФ сбрасывается и система начинает новый цикл обмена информацией.

Таким образом весь обработанный массив передается из ведомой в ведущую микро-ЭВМ, причем из алгоритма программы следует то, что он может быть записан по тем же адресам в ведущей микро-ЭВМ, с которых он был передан в ведомую.

Данная задача демонстрирует возможность создания мультиплексорных систем с использованием БИС КР580ВВ55.

4.18. 1. Разработать схему обмена информацией в параллельном формате между двумя микро-ЭВМ с использованием двух БИС КР580ВВ55. 2. Написать программу, обеспечивающую обмен данными в режиме, в котором одна микро-ЭВМ является ведущей, а другая — ведомой.

● **4.19.** Написать программу начальной установки и запрограммировать БИС КР580ВВ51 для работы в асинхронном режиме, двумя стоп-битами, контролем по признаку четности, длиной слова 8 бит и отношением частоты передачи данных к частоте сигнала синхронизации входа приемника и выхода передатчика 1 : 1.

Решение.

Подпрограмма загрузки:

```
        USA R 1 — адрес регистра режима и команды
LOAD: MVI A, 74H — записать команду начальной установки
      OUT USA R1   и загрузить в регистр команды
      MVI A,0FD H — записать инструкцию режима и загрузить
      OUT USA R1   в регистр режима
```

```
MVI   A, 37H — записать инструкцию команды
OUTUSARI — загрузить в регистр команды
RET — возврат
```

При выполнении команды начальной установки происходит программный сброс микросхемы в исходное состояние.

Запись инструкции FD в регистр режима обеспечивает работу БИС в асинхронном режиме с двумя стоп-битами, контролем по признаку четности, длиной слова 8 бит и отношением частоты передачи данных к частоте сигнала синхронизации входа приемника и выхода передатчика 1 : 1.

Запись инструкции команды 37 в регистр команды осуществляет сброс триггеров ошибок, разрешение передачи и приема, формирует запрос готовности приемника принять данные и запрос готовности передатчика терминала передать данные.

4.20. Написать программу начальной установки БИС КР580ВВ51 для работы в синхронном режиме обмена с внешней синхронизацией, контролем по признаку нечетности и длиной слова 7 бит.

4.21. Изобразить временные диаграммы синхронизации приема и передачи данных с использованием БИС КР580ВВ51.

4.22. Составить схему сопряжения микро-ЭВМ с модемом с использованием БИС КР580ВВ51 и записать программу приема 30 байт данных в асинхронном режиме и отношением частоты передачи данных к частоте сигнала синхронизации входа приемника и выхода передатчика 1 : 64.

4.23. Написать программу приема одного байта информации с клавиатуры дисплея и последующей выдачи эхосимвола в канал приема дисплея. Обмен информацией осуществить в дуплексном режиме. Интерфейс дисплея должен работать в асинхронном режиме с одним стоп-битом без контроля четности длиной слова 7 бит и отношением частоты передачи данных к частоте сигнала синхронизации входа приемника и выхода передатчика 1 : 64.

4.24. 1. Разработать схему для обмена информацией между двумя микро-ЭВМ в последовательном коде с помощью БИС КР580ВВ51. 2. Написать программу передачи массива данных длиной 256 байт в следующих режимах работы: а) симплексном асинхронном; б) дуплексном синхронном с различными видами синхронизации.

4.25. 1. Рассмотреть возможные варианты подключения магнитофона к микро-ЭВМ с использованием метода частотной модуляции и метода фазового кодирования. 2. Указать различия в схемотехническом и программном обеспечении устройств сопряжения. 3. Разработать схему сопряжения магнитофона с микро-ЭВМ на базе БИС КР580ВВ51 с использованием частотной модуляции сигнала. 4. Разработать схему сопряжения и написать программу, обеспечивающую обмен информацией с магнитофоном на базе БИС КР580ВВ51, с использованием метода фазового кодирования.

Примечание. Метод фазового кодирования характеризуется отсутствием постоянной составляющей сигнала. В этом методе сигнал «1» представляется переходом из «1» в «0», сигнал «0» представляется переходом из «0» в «1».

При использовании синхронного режима с двумя синхросимволами БИС КР580ВВ51 позволяет реализовать передачу методом фазового кодирования посредством программного представления передаваемых битов в виде 2 бит: «0»→«01», «1»→«10».

При этом скорость передачи будет в два раза меньше тактовой частоты синхроимпульсов передатчика и приемника, так как каждый байт будет передаваться в виде 2 байт.

4.26. Написать программу, обеспечивающую обмен информацией между ЗУ микро-ЭВМ и двумя внешними устройствами по каналу ПДП на базе БИС КР580ВТ57. Для связи с внешними устройствами использовать два канала ПДП: канал 0 — для передачи 32 слов из ЗУ микро-ЭВМ во внешнее устройство начиная с адреса 0000_{16}; канал 1 — для приема 1024 слов из внешнего устройства в ЗУ микро-ЭВМ начиная с адреса $E010_{16}$. При организации обмена использовать циклический режим задания приоритетов каналов. Рассмотреть возможности программного управления режимами обмена.

Решение. Программа загрузки БИС КР580ВТ57:

```
LOOP: MVI   A, 10H — запись младшего байта начального адреса
      OUT    DMA0 — загрузка в регистр L адреса канала 0
      MVI  A, 0EDH — запись старшего байта начального адреса
      OUT    DMA0 — загрузка в регистр H адреса канала 0
      MVI  A, 0FFH — запись младшего байта длины массива
      OUT    DMA1 — загрузка в регистр L конца счета канала
      MVI    A, 44H — запись старшего байта длины и режима работы
      OUT    DMA1 — загрузка в регистр H конца счета канала
      MVI    A, 00H — запись младшего байта начального адреса
      OUT    DMA2 — загрузка в регистр младшего адреса канала 1
      MVI    A, 00H — запись старшего байта начального адреса
      OUT    DMA2 — загрузка в регистр старшего адреса канала 1
      MVI    A, 1FH — запись младшего байта длины массива
```

```
OUT   DMA3 — загрузка в младший регистр конца счета канала 1
MVI   A, 80H — запись старшего байта длины и режима работы канала 1
OUT   DMA3 — загрузка в регистр режима канала 1 и регистр конца счета
MVI   A, 53H — запись управляющего слова
OUT   DMA3 — загрузка его в регистр управления DMA
HLT        — останов
```

4.27. Привести алгоритм работы контроллера ПДП КР580ВТ57.

4.28. Изобразить временны́е диаграммы импульсов на магистралях микро-ЭВМ в различных режимах обмена по каналу ПДП.

4.29. Разработать схему, позволяющую исследовать временны́е диаграммы работы БИС КР580ВТ57 с помощью осциллографа. Для обеспечения устойчивой картины на экране осциллографа следует использовать режим автозагрузки БИС. Схема должна обеспечивать периодическую посылку импульсов запроса ПДП.

4.30. Разработать схему и написать программу, позволяющую проводить обмен информацией с помощью БИС КР580ВТ57 между микро-ЭВМ и медленно действующими внешними устройствами.

4.31. Разработать схему для снятия запроса ПДП внешним устройством по импульсу МАРК на выводе БИС КР580ВТ57.

4.32. БИС КР580ВТ57 может функционировать в режиме контроля. В этом режиме блок управления каналами формирует все сигналы, за исключением сигналов $\overline{\text{Ч.память}}$, $\overline{\text{З.память}}$, $\overline{\text{Ч.ВВ}}$, $\overline{\text{З.ВВ}}$. Передачи информации при этом не происходит, МП БИС отключается от магистралей и контроллер ПДП выдает сигнал разрешения прямого доступа. С использованием описанного режима составить: 1) алгоритм работы контроллера для обеспечения возможности подсчета контрольной суммы внешним устройством; 2) алгоритм работы контроллера в режиме регенерации динамического ЗУ (последовательное обращение к младшим 128 адресам ЗУ в режиме чтения через определенные промежутки времени).

4.33. 1. Разработать схему подключения БИС КР580ВИ53 для реализации программного ждущего мультивибратора (МВ) с внешним запуском по переднему фронту входного импульса. Параметры выходного импульса задаются программно: задержка импульса изменяется от 0,5 мкс до 30 мс, длительность импульса — от 0,5 мкс до 30 мс.

2. Написать подпрограммы начальной установки и загрузки таймера. Задержка и длительность импульса задаются количеством машинных тактов МП БИС и записаны в четырех последовательных ячейках ОЗУ.

Р е ш е н и е. Схема программируемого ждущего мультивибратора (МВ) приведена на рис. 10.17, *а* и состоит из БИС таймера *У1*, триггера *У3* и схемы начальной установки триггера *У2*. Канал служит для получения необходимой задержки между передним фронтом импульса запуска МВ и передним фронтом выходного импульса. Канал 1 определяет длительность выходного импульса. При загрузке констант в канал 0 таймера производится началь-

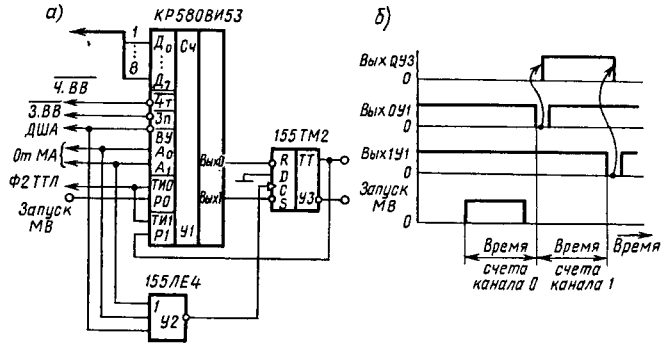


Рис. 10.17. Схема программируемого ждущего мультивибратора на основе КР580ВИ53 (*а*) и временна́я диаграмма его работы (*б*)

ная установка триггера *У3*, так как на выходе элемента триггера *У2* в этот момент появляется сигнал единичного уровня. Канал 0 запускается при наступлении на его вход *Р0* переднего фронта импульса запуска *МВ* (рис. 10.17, *б*), отсчитывает длительность задержки и устанавливает триггер *У3* в состояние «1». Триггер *У3*, в свою очередь, разрешает работу канала 1 (так как выход триггера подключен ко входу *Р1*). Канал 1 отсчитывает интервал времени, равный длительности выходного импульса, и сбрасывает триггер *У3*. После завершения цикла при наступлении переднего фронта импульса процессы в схеме повторяются. Каналы 0 и 1 работают как двоичные 16-разрядные счетчики в режиме 5. Программа для начальной установки и загрузки таймера будет иметь такой вид:

```
GMVO: MVI A, 3AH    — загрузить режим работы канала 0
      OUT TCW       — TCW — адрес регистра управления таймера
      LXI H, ADDR   — загрузить адрес области ОЗУ, в которой хранятся константы
      MOV A, M      — загрузить младший байт задержки по
      OUT T0          адресу 0 канала 0
      INX H         — загрузить старший байт задержки
      MOV A, M
      OUT T0
      INX H
      MVIA, 5AH     — загрузить режим работы канала 1
      OUT TCW       — загрузить младший байт длительности
      MOV A, M        по адресу канала 1
      OUT T1
      INX H
      MOV A, M      — загрузить старший байт длительности
      OUT T1
      RET           — возврат
```

4.34. Разработать схему программируемого генератора временны́х интервалов на основе БИС КР580ВИ53 с длительностью, изменяемой от 0,5 мкс до 0,5 ч. Написать подпрограммы начальной установки и загрузки таймера для случая, когда длительность временно́го интервала в машинных тактах записана в двух последовательных ячейках ОЗУ.

4.35. 1. Разработать схему подключения БИС КР580ВИ53 с микро-ЭВМ для определения времени выполнения (в машинных тактах) программы. 2. Написать подпрограмму начальной установки и загрузки таймера для случая, когда моменты начала и конца счета определяются записью соответствующих команд в его регистр управляющего слова.

4.36. 1. Разработать схему подключения БИС КР580ВИ53 к микро-ЭВМ, позволяющую определить число выполненных команд в программе. 2. Написать программу начальной установки и загрузки таймера для случая, когда моменты начала и конца счета определяются записью соответствующих команд в его регистр управляющего слова.

4.37. 1. Разработать схему генерации сигналов звуковой частоты на основе БИС КР580ВИ53. 2. Написать подпрограмму обслуживания таймера. Длительность сигнала равна 1 с. Частота звучания определяется числом в аккумуляторе, младший полубайт которого задает порядковый номер ноты в октаве (1—12), а старший — порядковый номер ноты в октаве (1—4). Абсолютное значение частоты определяется по таблице кодов, хранящейся в ЗУ. 3. Рас-

смотреть варианты задания длительности сигнала с использованием второго канала таймера и подпрограммы временно́й задержки.

● **4.38.** Разработать схему блока приоритетных прерываний с использованием ИС К589ИК14, позволяющую осуществлять сброс триггера запроса обслуживаемого прерывания после подтверждения его приема МП БИС.

Р е ш е н и е. Схема БПП со сбросом обслуживаемых прерываний приведена на рис. 10.18. Схема блока приоритетных прерываний на основе ИС К589ИК14 описана в [57]. К особенностям схемы, показанной на рис. 10.18, следует отнести наличие триггеров запросов прерывания

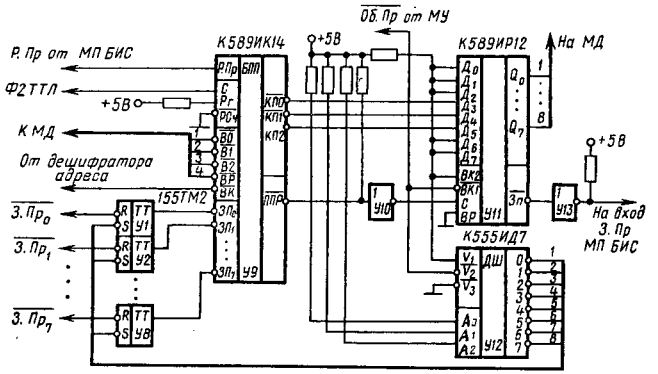


Рис. 10.18. Схема подключения блока приоритетного прерывания к магистралям микро-ЭВМ

У1—У8, позволяющих при наступлении короткого импульса запроса прерывания запомнить его и хранить до момента обслуживания. Сброс триггеров осуществляется импульсом сброса с выходов дешифратора *У12*. При разрешении прерываний БПП определяет прерывание с наивысшим разрешенным приоритетом и выдает его код на выходы A_0—A_2. При наступлении импульса Об.Пр (обслуживание прерывания), подтверждающего прием прерывания, дешифратор отпирается и выдает сигнал сброса на соответствующий триггер. Так как на выход БПП код прерывания выдается инверсный, то выход 7 дешифратора *У12* служит для сброса триггера *У1* запроса прерывания, выход 6 — триггера *У2* и т. д.

4.39. 1. Разработать схему, позволяющую реализовать 29-векторную систему прерываний с использованием БИС КР580ВН59. 2. Написать программу начальной загрузки для всех БИС КР580ВН59, входящих в состав схемы.

4.40. Указать команду, которую необходимо записать для запрета приема всех восьми прерываний одной из ведомых БИС в схеме для задачи 4.39. Указать последовательность команд для запрета приема прерывания с уровнем приоритета ниже 3 у всех ведомых БИС.

4.41. Разработать программу, позволяющую с использованием каскадного соединения и считываемого типа прерываний получить систему с числом векторов более 64.

Особенности построения микро-ЭВМ на микропроцессорном комплекте серии К589.

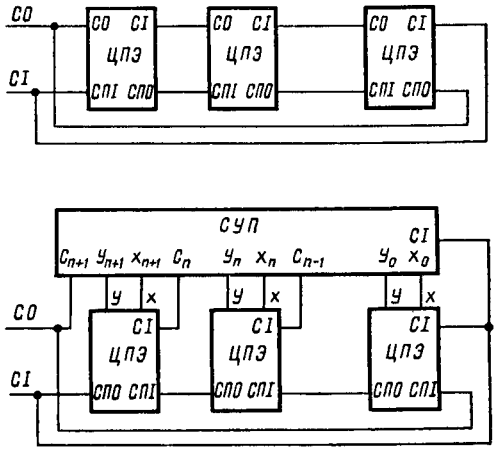


Рис. 10.19. Схема операционного блока

Особенности построения микро-ЭВМ на микропроцессорном комплекте серии К589

Примечание. Построение вычислительных устройств на основе микропроцессорных БИС серии 589 требует знаний по схемотехническому проектированию микропроцессорных БИС и организации микропрограммного управления на основе ПЗУ и ППЗУ [57].

● **4.42.** Изобразить схему 8-разрядного операционного блока на МП БИС К589ИК02 с использованием схемы ускоренного переноса К589ИК03 и без нее; оценить разницу в скорости выполнения операций в обоих случаях.

Решение. Схема операционного блока дана на рис. 10.19. Разница в скорости выполнения операции опре-

деляется различием во времени формирования сигнала *СО* ($T_{со}$). В первом случае $T_{со} = nt_{со}$, где n — число секций ЦПЭ; $t_{со}$ — время формирования сигнала СО в одном модуле ЦПЭ. Во втором случае $T_{со} = t_{суп}$, где $t_{суп}$ — время формирования сигнала схемой ускоренного переноса.

4.43. Какие сигналы и в какой последовательности необходимо подать на вход микросхемы при использовании МП БИС К589ИК01 в устройстве управления, чтобы сформировать на выходе адрес 00001101? По каким адресам возможно совершить условный переход из этого адреса?

4.44. Каким образом конвейерный режим работы устройства управления влияет на быстродействие микропроцессора? Чем отличаются программы микропроцессоров с конвейерной организацией устройства управления и без нее?

4.45. Какие микроинструкции в устройстве управления, построенном на МП БИС К589ИК01, являются обязательными? Определить их назначение и разрядность.

4.46. Составить последовательность микроинструкций, обеспечивающих начальную установку содержимого внутренних регистров ЦПЭ, в котором регистр 1 используется как счетчик команд, а регистр 2 — как указатель стека. При начальной установке в счетчике команд следует установить код 01_{16}, в указателе стека — код FE, остальные регистры ЦПЭ — обнулить.

4.47. Составить последовательность микроинструкций для микропроцессора, построенного на МП БИС К580ИК01, К589ИК02, обеспечивающих формирование обратного кода числа, записанного в регистре R1, и запись сформированного кода в регистр R2. Старший разряд микропроцессора отводится для записи знака чисел.

4.48. Составить последовательность микроинструкций для формирования дополнительного кода чисел в микропроцессоре, описанном в задаче 4.47.

4.49. Для микропроцессора, схема которого представлена на рис. 3.10 в [57], изобразить временны́е диаграммы, обозначающие последовательность работы блоков, и оценить минимальное время цикла выполнения операции.

4.50. Изобразить схему микро-ЭВМ на МП БИС серии К589 с 8-разрядным операционным блоком, позволяющую формировать 16-разрядное адресное поле.

4.51. Изобразить схему внешнего устройства, способного вести двусторонний обмен информацией с лабораторной микро-ЭВМ МП589. Какие команды обеспечивают об-

мен информацией между микро-ЭВМ и внешним устройством?

4.52. Составить схему и программы анализа запросов прерывания микропроцессора.

4.53. Составить принципиальную электрическую схему и схему программы, обеспечивающей считывание и дешифрацию сигналов с клавиатуры, имеющей 24 клавиши.

4.54. Каково назначение программ временно́й задержки при считывании сигналов клавиатуры и каким образом задается длительность временно́й задержки?

4.55. Изобразить схему подключения к микро-ЭВМ 6-разрядного семисегментного дисплея, отражающего содержимое определенных ячеек ОЗУ.

4.56. Составить схему программы управления дисплеем микро-ЭВМ.

4.57. Спроектировать принципиальную схему управления шестью разрядами семисегментных индикаторов, работающую в мультиплексном режиме.

4.58. Изобразить схему процесса многократного (не менее трех раз) обращения к стековой памяти, организованной на базе регистра-указателя стека в области оперативной памяти.

Особенности и специфические характеристики на микропроцессорном комплекте серии К589

Задачи и вопросы этого раздела связаны с особенностями работы асинхронных МКП серий К587 и К588, схожих по архитектуре построения, однако имеющих и отличительные особенности.

● **4.59.** Объяснить, в чем сложность построения микро-ЭВМ на МПК серии К587 с обрамлением на ИС серии К133.

Р е ш е н и е. Микро-ЭВМ, построенная на базе МПК К587, имеет напряжение питания $+9$ В. При этом уровни «0» и «1» управляющих и информационных сигналов равны соответственно $0+0{,}4$ и $+9\pm0{,}5$ В. Серия К133 имеет питающее напряжение $+5$ В и логические уровни «0» $—0^{+0{,}4}$ В и «1»$—2{,}4^{+2{,}1}$ В. Поэтому при привлечении серии К133 для указанной микро-ЭВМ необходимо применять схемы согласования логических уровней. Кроме того, при разработке необходимо учесть разницу в быстродействии схем. В качестве обрамления для микро-ЭВМ на базе МПК К587 рекомендуется применять серию К564, согласованную по логическим уровням и быстродействию.

4.60. Составить функциональную схему 16-разрядного операционного устройства на основе БИС АУ серии К587.

4.61. Возможна ли работа микро-ЭВМ, построенной на основе серий К587 или К588 без БИС АР?

4.62. Составить функциональную схему блока умножения 32-разрядных чисел на основе БИС АР серии К587.

4.63. Объяснить, в чем основной смысл применения БИС ОИ серии К587.

4.64. Как сказывается введение в схему БИС ОИ серии К587 на быстродействие микро-ЭВМ?

4.65. Для каких требований эксплуатации микро-ЭВМ на основе серии К587 наиболее выгодна?

4.66. Сколько микрокоманд должна выполнить БИС АР серии К587 для сдвига числа на 1 бит? на 3 бит? на 7 бит?

4.67. Какое устройство необходимо разработать в случае отказа от применения БИС УП при проектировании микро-ЭВМ на основе МПК серий К587 или К588?

4.68. Возможно ли наращивание управляющей памяти? Как подключить несколько БИС УП серии К587 к микро-ЭВМ?

4.69. Какая из схем ОЗУ наиболее удобна для сопряжения с БИС К587?

4.70. В чем основное преимущество асинхронного принципа работы операционного устройства?

4.71. Возможна ли работа микро-ЭВМ, построенной на основе МПК серии К588 только на микрокомандном уровне? Удобно ли это?

4.72. Какие изменения в составе БИС повлечет за собой модификация системы команд?

4.73. Чем отличается блок умножения 32-разрядных чисел, построенный на основе БИС АР серии К588, от аналогичного блока на основе БИС АР серии К587?

4.74. Возможна ли замена БИС АР серии К588 на БИС АР серии К587?

4.75. Каковы основные отличия БИС серии К588 от аналогичных БИС серии К587?

§ 10.5. Аппаратурно-программные средства отладки

Одна из основных функций средств отладки — сбор информации о поведении проектируемой системы и представление собранной информации в виде, удобном для того, кто ведет отладку: схемотехника, программиста, системотехника. Поэтому предлагаемые ниже примеры в основном направлены на усвоение форм представления информации о поведении системы на различных уровнях, на умение переводить из одной формы представления в другую и отличать один уровень представления от другого, показать информативность уровня.

Примеры подготовлены с помощью комплекса средств отладки «Электроника НЦ-803», который состоит из диалогового вычислительного комплекса «Электроника НЦ-8020» и блока логических устройств «Электроника Н МС 59401». Этот комплекс может выполнять функции: логического анализатора на 64 входа с объемом памяти 1024 бит на канал, с максимальной программируемой частотой 20 МГц и четырехуровневым логическим компаратором; генератора слов на 64 выхода с объемом памяти 1024 бит на канал, с максимальной программируемой частотой 10 МГц, с 12 микрокомандами; четырехлучевого цифрового осциллографа; частотомера; комплекса диагностирования на 64 входа/выхода (всего 128 каналов); эмулятора памяти емкостью 64К байт с временем выборки 300 нс, с внутрисхемным эмулятором микропроцессоров К1801ВМ1 и К1801ВМ2.

Основной материал главы базируется на [60]. Ссылки на литературу даются там, где необходим дополнительный источник.

● **5.1.** Определить значение сигнала на выходе компаратора уровней логического анализатора. Форма сигнала на входе логического анализатора показана на рис. 10.20, *а* (уровень компарации 3,5 В), где сигналы СС1—СС4 — синхросигналы логического анализатора, СС5 — сигналы сопровождения основного сигнала; * — значения сигналов в измеряемые моменты времени.

Р е ш е н и е. На рис. 10.20, *а* проведем горизонтальную линию, соответствующую уровню 3,5 В. Определим точки пересечения этой линии с заданной. Значения входного сигнала 3,5 В и выше на выходе дают значение сигнала, равное «1», значения входного сигнала ниже 3,5 В установят значение выходного сигнала, соответствующее «0». На рис. 10.20, *г* выходной сигнал обозначен К (3,5).

5.2. Определить значение сигнала на выходе компаратора уровней логического анализатора. Форма сигнала на входе логического анализатора показана на рис. 10.20, *а*; уровень компарации 0,5 В.

5.3. Определить значение сигнала на выходе компаратора уровней логического анализатора. Форма сигнала на входе логического анализатора показана на рис. 10.20, *а*; уровень компарации 1,5 В.

5.4. Определить значение сигнала на выходе компаратора уровней логического анализатора. Форма сигнала на входе логического анализатора показана на рис. 10.20, *б*; уровень компарации 0,7 В.

5.5. Определить значение сигнала на выходе компаратора уровней логического анализатора. Форма сигнала на входе логического анализатора показана на рис. 10.20, *б*; уровень компарации 2,8 В.

5.6. Определить значения сигналов на выходе компаратора уровней логического анализатора. Формы двух сиг-

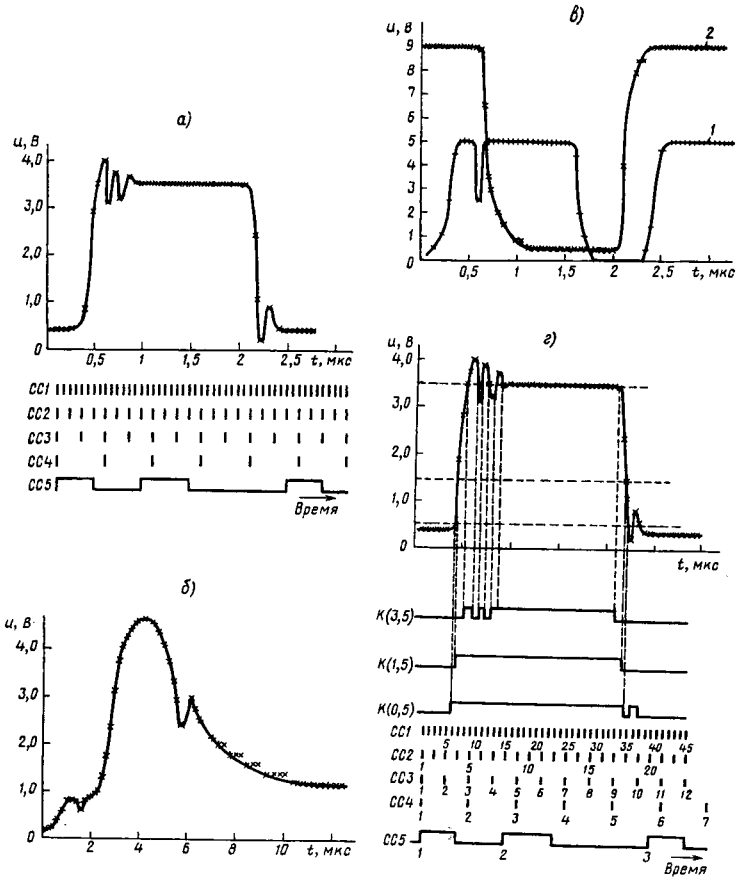


Рис. 10.20. Сигналы на входе логического анализатора, синхросигналы и сигналы на выходе компаратора уровней

налов на входе логического анализатора показаны на рис. 10.20, *в*; уровень компарации 4,0 В.

5.7. Определить значения сигналов на выходе компаратора уровней логического анализатора. Формы двух сигналов на входе логического анализатора показаны на рис. 10.20, *в*; уровень компарации 6,0 В.

● **5.8.** Определить данные, которые будут записаны в память логического анализатора. Задано: форма сигнала на входе логического анализатора (рис. 10.20, *а*); синхросигналы СС2; уровень компарации 1,5 В.

Р е ш е н и е. На рис. 10.20, *а* проведем горизонтальную линию, соответствующую уровню 1,5 В; определим точки пересечения этой линии с заданной. Значения сигнала 1,5 В и выше примем равным «1», значения сигнала ниже 1,5 В — «0». Пронумеруем синхросигналы, определим значение заданного сигнала в каждом такте и заполним табл. 10.2.

Таблица 10.2

А	Д	А	Д	А	Д	А	Д	А	Д	А	Д
01	0	05	1	09	1	13	1	17	1	21	0
02	0	06	1	10	1	14	1	18	0	22	0
03	0	07	1	11	1	15	1	19	0	23	0
04	1	08	1	12	1	16	1	20	0		

5.9. Определить данные, которые будут записаны в память логического анализатора. Задано: форма сигнала на входе логического анализатора (рис. 10.20, *а*); синхросигналы СС2; уровень компарации 3,5 В.

5.10. Определить данные, которые будут записаны в память логического анализатора. Задано: форма сигнала на входе логического анализатора (рис. 10.20, *а*); синхросигналы СС2; уровень компарации 0,5 В.

5.11. Определить данные, которые будут записаны в память логического анализатора. Задано: форма сигнала на входе логического анализатора (рис. 10.20, *а*); синхросигналы СС1; уровень компарации 3,5 В.

5.12. Определить данные, которые будут записаны в память логического анализатора. Задано: форма сигнала на входе логического анализатора (рис. 10.20, *а*); синхросигналы СС3; уровень компарации 3,5 В.

5.13. Определить данные, которые будут записаны в память логического анализатора. Задано: форма сигнала на входе логического анализатора (рис. 10.20, *а*); синхросигналы СС4; уровень компарации 3,5 В.

5.14. Определить данные, которые будут записаны в память логического анализатора. Задано: форма сигнала на входе логического анализатора (рис. 10.20, *а*); синхросигнал — передний фронт СС5; уровень компарации 3,5 В.

● **5.15.** Определить данные, которые будут записаны в память логического анализатора. Задано: форма сигнала на входе логического анализатора (рис. 10.20, *а*); синхросигналы СС2; уровень компарации 3,5 В; включены «ловушки».

Р е ш е н и е. Определим форму выходного сигнала компаратора уровней К (3,5) на рис. 10.20, *г*. Пронумеруем синхросигналы. Затем начиная с первого синхросигнала последовательно определяем данные, записываемые в память логического анализатора. По первым четырем сигналам в память будут записаны «0». В период между синхросигналами 4 и 5 выходной сигнал компаратора уровней изменит свое значение «0→1→0». Этот факт будет зафиксирован схемой «ловушка» и в память синхросигналом 5 будет записана «1». Несмотря на то что в такте 6 выходной сигнал компаратора уровней меняет свое состояние, схема «ловушка» этого факта не зафиксирует, так как для этой схемы сигнал меняется с «1» на «0». Синхросигнал 6 зафиксирует в память «0». С синхросигнала 7 по синхросигнал 16 будут записаны в память «1», в остальных тактах — «0». Содержимое памяти логического анализатора представлено в табл. 10.3 в столбце $Д_3$.

Т а б л и ц а 10.3

А	$Д_1$	$Д_2$	$Д_3$	$Д_4$	А	$Д_1$	$Д_2$	$Д_3$	$Д_4$	А	$Д_1$	$Д_2$	$Д_3$	$Д_4$
01	0	0	0	0	09	1	1	1	1	17	0	1	0	1
02	0	0	0	0	10	1	1	1	1	18	0	0	0	0
03	0	0	0	0	11	1	1	1	1	19	0	0	0	1
04	0	1	0	1	12	1	1	1	1	20	0	0	0	0
05	0	1	1	1	13	1	1	1	1	21	0	0	0	0
06	0	1	0	1	14	1	1	1	1	22	0	0	0	0
07	1	1	1	1	15	1	1	1	1	23	0	0	0	0
08	1	1	1	1	16	1	1	1	1					

5.16. Определить данные, которые будут записаны в память логического анализатора. Задано: форма сигнала на входе логического анализатора (рис. 10.20, *а*); синхросигналы СС2; уровень компарации 0,5 В; включены «ловушки».

5.17. Определить данные, которые будут записаны в память логического анализатора. Задано: формы сигналов на входах логического анализатора (рис. 10.20, *в*); синхросигналы; уровень компарации 4,0 В.

● **5.18.** Представить на экране логического анализатора в виде временно́й диаграммы собранные данные, хранящиеся в памяти логического анализатора (см. табл. 10.2).

Р е ш е н и е. Выберем масштаб представления, например в 5 мм один такт. Проведем горизонтальную линию, разметим ее по 5 мм, пронумеруем такты и поставим значение сигнала в каждом такте. Временна́я диаграмма

представлена на рис. 10.21. Сравнить ее с исходным сигналом и К (1,5) на рис. 10.20, *г*.

● **5.19.** Представить данные с адреса 0002 (момент запуска) по 0065 (табл. 10.4) в виде временно́й диаграммы на экране логического анализатора.

Данные (табл. 10.4) собраны 32-канальным логическим анализатором, который подключен к системной магистрали микро-ЭВМ «Электроника НЦ-8001Д» в соответствии с табл. 10.5. Максимальное число строк экрана анализатора — 24.

Рис. 10.21. Временна́я диаграмма на экране логического анализатора

Р е ш е н и е. Выберем масштаб изображения 1:1. Начертим 17 параллельных линий, присвоим каждой линии имя согласно табл. 10.5, отметим отрезки, соответствующие тактам. Пронумеруем такты. Поставим соответствие между номером каждого такта и адресом ячейки памяти логического анализатора, записанного в этом такте. Проставим значения сигналов на линиях в каждом такте. Значения сигналов на линии АД изобразим согласно принятой мнемонике (рис. 10.22).

Т а б л и ц а 10.4.

0	:	0000	1101	0000	0000	0000	0010	0011	0110
1	:	0000	0000	0000	0000	0000	0000	0000	0000
2	:	0000	0000	0000	0000	0000	0000	0000	0000
3	:	0000	0000	0000	0000	0000	0000	0000	0000
4	:	0000	0000	0000	0000	0000	0000	0000	0000
5	:	0000	0000	0000	0000	0000	0010	0010	0100
6	:	0000	0000	0000	0000	0000	0010	0010	0100
7	:	0000	0000	0000	0000	0000	0010	0010	0100
8	:	0000	1000	0000	0000	0000	0010	0010	0100
9	:	0000	1001	0000	0000	0000	0010	0010	0100
10	:	0000	1001	0000	0000	0000	0000	0000	0000
11	:	0000	1001	0000	0000	0000	0000	0000	0000
12	:	0000	1101	0000	0000	0000	0000	0000	0000
13	:	0000	1101	0000	0000	0000	0000	0000	1110
14	:	0000	1101	0000	0000	0000	0000	0000	1110
15	:	0000	1101	0000	0000	0000	0000	0000	1110
16	:	0000	1101	0000	0000	0000	0000	0000	1110
17	:	0000	1101	0000	0000	0000	0000	0000	1110

Продолжение табл. 10.4

```
   18 : 0000 1101 0000 0000 0000 0000 0000 1110
   19 : 0000 1101 0000 0000 0000 0000 0000 1110
   20 : 0000 1100 0000 0000 0000 0000 0000 1110
   21 : 0000 1000 0000 0000 0000 0000 0000 0000
   22 : 0000 1000 0000 0000 0000 0000 0000 0000
LA)
   23 : 0000 1000 0000 0000 0000 0000 0000 0000
   24 : 0000 1000 0000 0000 0000 0000 0000 0000
   25 : 0000 1000 0000 0000 0000 0000 0000 0000
   26 : 0000 0000 0000 0000 0000 0010 0011 0100
   27 : 0000 0000 0000 0000 0000 0010 0011 0100
   28 : 0000 1000 0000 0000 0000 0010 0011 0100
   29 : 0000 1000 0000 0000 0000 0010 0011 0100
   30 : 0000 1001 0000 0000 0000 0010 0011 0000
   31 : 0000 1001 0000 0000 0000 0000 0001 0000
   32 : 0000 1001 0000 0000 0000 0000 0000 0000
   33 : 0000 1101 0000 0000 0000 0000 0000 0000
   34 : 0000 1101 0000 0000 0000 0100 0000 1010
   35 : 0000 1101 0000 0000 0000 0100 0000 1010
   36 : 0000 1101 0000 0000 0000 0100 0000 1010
   37 : 0000 1101 0000 0000 0000 0100 0000 1010
   38 : 0000 1101 0000 0000 0000 0100 0000 1010
   39 : 0000 1101 0000 0000 0000 0100 0000 1010
   40 : 0000 1100 0000 0000 0000 0100 0000 1010
   41 : 0000 1100 0000 0000 0000 0000 0000 0000
   42 : 0000 1000 0000 0000 0000 0000 0000 0000
   43 : 0000 1000 0000 0000 0000 0000 0000 0000
   44 : 0000 1000 0000 0000 0000 0000 0000 0000
   45 : 0000 1000 0000 0000 0000 0000 0000 0000
LA)
   46 : 0000 1000 0000 0000 0000 0000 0000 0000
   47 : 0000 0000 0000 0000 0000 0000 0000 0000
   48 : 0000 0000 0000 0000 0000 0000 0000 0000
   49 : 0000 0000 0000 0000 0000 0000 0000 0000
   50 : 0000 0000 0000 0000 0000 0000 0000 0000
   51 : 0000 0000 0000 0000 0000 0000 0000 0010
   52 : 0000 0000 0000 0000 0000 0010 0010 0110
   53 : 0000 0000 0000 0000 0000 0010 0010 0110
   54 : 0000 1000 0000 0000 0000 0010 0010 0110
   55 : 0000 1001 0000 0000 0000 0010 0010 0110
   56 : 0000 1001 0000 0000 0000 0000 0000 0000
   57 : 0000 1001 0000 0000 0000 0000 0000 0000
   58 : 0000 1001 0000 0000 0000 0000 0000 0000
   59 : 0000 1101 0000 0000 0000 0010 0010 0111
   60 : 0000 1101 0000 0000 0000 0010 1111 0111
   61 : 0000 1101 0000 0000 0000 0010 1111 0111
   62 : 0000 1101 0000 0000 0000 0010 1111 0111
   63 : 0000 1101 0000 0000 0000 0010 1111 0111
   64 : 0000 1101 0000 0000 0000 0010 1111 0111
   65 : 0000 1101 0000 0000 0000 0010 1111 0111
   66 : 0000 1100 0000 0000 0000 0010 1111 0111
   67 : 0000 1100 0000 0000 0000 0000 0000 0000
   68 : 0000 1000 0000 0000 0000 0000 0000 0000
LA)
```

Таблица 10.5

Номер канала	Имя сигнала	Номер канала	Имя сигнала	Номер канала	Имя сигнала	Номер канала	Имя сигнала
1	ДА 00	9	ДА 08	17	ПОСТН	25	Ввод
2	ДА 01	10	ДА 09	18	ПИНТ	26	ПВ
3	ДА 02	11	ДА 10	19	Сброс	27	СИП
4	ДА 03	12	ДА 11	20	ОСТ	28	СИА
5	ДА 04	13	ДА 12	21	ВУ	29	ППР
6	ДА 05	14	ДА 13	22	Х	30	ТПР
7	ДА 06	15	ДА 14	23	БАЙТ	31	ППД
8	ДА 07	16	ДА 15	24	Вывод	32	ТПД

Принята следующая мнемоника изображения информации на магистрали данных и адреса:

— — значения сигналов на всех 16 линиях равны «0»;

> — момент перехода значений сигналов всех 16 линий в состояние «0»;

< — момент перехода от значений сигналов, когда они все равны «0», к состоянию, когда хотя бы на одной линии сигнал принимает значение «1»;

= — значения сигналов на линиях магистрали в данный момент равны значениям сигналов в предыдущий момент и не равны «0»;

X — переход от одного «ненулевого» состояния значений сигналов на линиях магистрали к другому «ненулевому» состоянию.

На линиях магистрали принято считать за «0» значение сигнала не менее 1,4 В; за «1» — в противном случае.

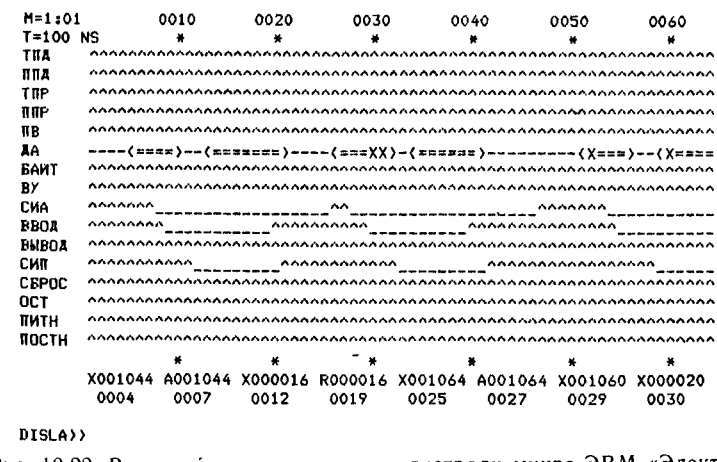


Рис. 10.22. Временна́я диаграмма на магистрали микро-ЭВМ «Электроника НЦ-8001Д»

```
001000 012700 001066
001004 016720 000042
001010 016701 000040
001014 010120
001016 105721
001020 020167 000032
001024 001373
001026 016700 000026
001032 110160 001066
001036 060067 000020
001042 020067 000016
001046 001367
001050 000753
001052 000012
001054 000000
001056 000004
001060 177777
```

Рис. 10.23. Листинг программы на машинном языке

5.20. Определить, какая процедура системной магистрали «Электроника НЦ-8001Д» (ввод, вывод) изображена на рис. 10.22.

5.21. Представить данные с адреса 0000 по 0010 табл. 10.4 в виде временно́й диаграммы на экране логического анализатора. Моменту запуска соответствует адрес 0002. Условия аналогичны условиям задачи 5.19.

5.22. Представить данные с адреса 0050 по 0068 табл. 10.4 в виде временно́й диаграммы на экране логического анализатора. Моменту запуска соответствует адрес 0000. Условия аналогичны условиям задачи 5.19.

5.23. Описать на языке ассемблера ЭВМ «Электроника-60» (PDP, LSI) программу, представленную в машинных кодах на рис. 10.23.

Примечание. Язык ассемблера ЭВМ PDP, LSI, «Электроника-60» описан в [44].

5.24. Составить граф программы, представленной на рис. 10.23.

5.25. Составить последовательность процедур системной магистрали ЭВМ «Электроника НЦ-8001Д» при выполнении программы, представленной на рис. 10.23, с адреса 1 по 7 памяти логического анализатора (табл. 10.6). В таблице дана распечатка памяти 32-канального логического анализатора, подключенного к системной магистрали ЭВМ «Электроника НЦ-8001Д» в соответствии с табл. 10.5. Данные собраны в синхронном режиме, синхросигна́лами являлись передние фронты сигналов СИА и СИП соответственно каналов 28 и 27 (табл. 10.6).

Решение. Примем следующие обозначения: INP — процедура ввода данных; OUW — процедура вывода слова; INP, ..., OUW — процедура ввода паузы и вывода слова; OUB — процедура вывода байта.

Примечание. Данные и адреса на магистрали принято обозначать восьмеричными числами, адрес логического анализатора — десятичными числами.

Таблица 10.6.

```
   0 : 0000 1101 0000 0000 0000 0010 0011 0110
   1 : 0000 1000 0000 0000 0000 0010 0001 1110
   2 : 0000 1101 0000 0000 0110 0000 0011 0111
   3 : 0000 1000 0000 0000 0000 0010 0010 0000
   4 : 0000 1101 0000 0000 0000 0000 0001 0000
   5 : 0000 1000 0000 0000 0000 0010 0011 0010
   6 : 0000 1101 0000 0000 1101 1111 1001 1110
   7 : 0000 1100 1000 0000 1101 1111 1001 1101
   8 : 0000 1000 0000 0000 0000 0010 0010 0010
   9 : 0000 1101 0000 0000 0010 0000 0011 0111
  10 : 0000 1000 0000 0000 0000 0010 0010 0100
  11 : 0000 1101 0000 0000 0000 0000 0000 1110
  12 : 0000 1000 0000 0000 0000 0010 0011 0100
  13 : 0000 1101 0000 0000 0000 0100 0000 1010
  14 : 0000 1000 0000 0000 0000 0010 0010 0110
  15 : 0000 1101 0000 0000 0000 0010 1111 0111
  16 : 0000 1000 0000 0000 0000 0010 0001 0110
  17 : 0000 1101 0000 0000 0001 1101 1100 0000
  18 : 0000 1000 0000 0000 0000 0010 0001 1000
  19 : 0000 1101 0000 0000 0000 0000 0001 0110
  20 : 0000 1000 0000 0000 0000 0010 0011 0000
  21 : 0000 1101 0001 0000 1111 1111 1111 1111
  22 : 0000 1000 0000 0000 0000 0010 0001 1010
LA)
  23 : 0000 1101 0000 0000 1001 0000 0111 0000
  24 : 0000 1000 0000 0000 0000 0010 0001 1100
  25 : 0000 1101 0000 0000 0000 0010 0011 0110
  26 : 0000 1000 0100 0000 0000 0010 0011 0101
  27 : 0000 1100 1100 0000 0000 0100 0000 0000
  28 : 0000 1000 0000 0000 0000 0010 0001 1110
  29 : 0000 1101 0000 0000 0110 0000 0011 0111
  30 : 0000 1000 0000 0000 0000 0010 0010 0000
  31 : 0000 1101 0000 0000 0000 0000 0001 0000
  32 : 0000 1000 0000 0000 0000 0010 0011 0010
  33 : 0000 1101 0000 0000 1101 1111 1001 1101
  34 : 0000 1100 1000 0000 1101 1111 1001 1100
```

Разбирая данные по адресу 1 логического анализатора, видим, что записан адрес 001036_8, который был установлен на магистрали (наличие сигнала СИА). В следующей ячейке памяти логического анализатора записаны данные (наличие сигналов СИА и СИП). Процедура на магистрали была INP, так как был сигнал ВВОД. В строке 3 таблицы имеется «1» в разряде 28 (СИА). Следовательно, в этой строке в разрядах 1—16 записан адрес 1040_8. В строке 4 имеются единицы в разрядах 28 (СИА), 27 (СИП) и 25 (ВВОД). Следовательно, вводились данные (INP). В разрядах 1—6 записано число 000020_8. В строке 5 имеется «1» только в разряде 28. Следовательно, в этой строке записан адрес 001062_8. В строке 6 имеются «1» в разрядах 28 (СИА), 27 (СИП) и 24 (ВЫВОД);

осуществлялась процедура ввода данных 157636_8. В следующей строке 7 «1» имеются в разрядах 28 (СИА), 27 (СИП) и 24 (ВЫВОД). Следовательно, в строках 5—7 записана процедура ввода паузы и вывода слова (рис. 10.24).

5.26. Составить последовательность процедур на системной магистрали ЭВМ «Электроника НЦ-8001Д» по данным с адреса 8 по 17 табл. 10.6. Условные обозначения описаны в задаче 5.25.

```
001036 -INP- 060067                    / 0    / /
001040 -INP- 000020                    / NUL DLE/
001062 -INP- 157636 -OUW- 157635       / _    GS/
```

Рис. 10.24. Последовательность процедур на магистрали микро-ЭВМ «Электроника НЦ-8001Д» по данным адреса табл. 10.4

5.27. Составить последовательность процедур на системной магистрали ЭВМ «Электроника НЦ-8001Д» по данным с адреса 18 по 63 табл. 10.6. Условные обозначения описаны в задаче 5.25.

5.28. Представить данные, записанные в табл.10.6 с адреса 8 по 17, в коде ASCII. Код ASCII описан в [61].

5.29. Представить данные, записанные в табл. 10.6 с адреса 18 по 63, в коде ASCII.

● **5.30.** Представить трассу участка программы, показанную на рис. 10.23, с адреса 8 по 13 табл. 10.6.

Р е ш е н и е. Составим последовательность процедур системной магистрали по данным, хранящимся в памяти логического анализатора с адреса 8 по адрес 13:

001042 — INP — 020067
001044 — INP — 000016
001064 — INP — 002012

Так как все процедуры INP, то это означает, что каждый раз из памяти могли быть считаны команды или данные. Делаем предположение, что по адресу 001042_8 читается команда. По адресу 001042_8 прочитан код 020067_8, где 020067_8 — команда (CMP) сравнения содержимого регистра R0 с содержимым ячейки памяти, адрес 1064_8 которой определяется как сумма содержимого второго слова команды 000016_8 с содержимым счетчика команд 001042_8 плюс константа 4. Таким образом, в памяти логического

анализатора с адреса 8 по 13 записана трасса выполнения команды:

```
001042 020067 000016 CMP RO, 001064
001044 — INP — 000016
001064 — INP — 002012
```

5.31. Представить трассу участка программы, показанную на рис. 10.23, с адреса 14 по 67 табл. 10.6.

5.32. Показать, что общего и в чем отличие между осциллографом и логическим анализатором с точки зрения пользователя этих приборов.

5.33. Продемонстрировать синхронный и асинхронный режимы записи данных логического анализатора на примере табл. 10.4 и 10.6. Указать адреса этих таблиц, в ячейках памяти которых записаны данные с одного и того же участка программы, представленной на рис. 10.23.

§ 10.6. Информационно-управляющие вычислительные системы

В данном разделе представлены контрольные вопросы и задачи, позволяющие более углубленно изучить основные положения [63] и приобрести практические навыки расчетов некоторых характеристик микропроцессорных информационно-управляющих вычислительных систем.

Основные требования, предъявляемые к микропроцессорным информационно-управляющим вычислительным системам (МП ИУВС)

Данный раздел содержит вопросы, позволяющие сконцентрировать внимание читателя на более важных моментах при изучении структуры, решаемых задач и параметров входной информации МП ИУВС.

6.1. Изобразить структуру микропроцессорной ИУВС, дать краткую характеристику устройств, входящих в ее состав.

6.2. Привести примеры задач, решаемых ИУВС, распределить задачи по классам.

6.3. Какие сведения о входной информации, представленной в цифровой форме, являются достаточными при разработке ИУВС: а) разрядность r_i входной информации по каждому каналу; б) разрядность r_i входной информации по каждому каналу, цена младшего разряда ЦМР$_i$ для каждого входного сигнала; в) диапазон представления сигналов x_{max} и x_{min}, первая и вторая производные сигналов; г) разрядность r_i входной информации по каждому каналу, цена младшего разряда ЦМР$_i$ для каждого

входного сигнала, количество достоверных разрядов $r_i^д$, закон изменения шумов и его параметры; д) число достоверных разрядов $r_i^д$, скорость передачи информации; е) скорость передачи информации; территориальная удаленность датчиков входной информации от ИУВС?

6.4. Какие сведения о входной информации, представленной в аналоговой форме, являются достаточными при разработке ИУВС: а) минимальное и максимальное значения величины x, кодируемой аналоговым сигналом A_x, коэффициент пропорциональности $K_x = A_x/x$; б) вид аналогового сигнала, количество достоверных разрядов; в) вид аналогового сигнала, полоса частот входного сигнала и полоса частот аддитивных шумов, маскирующих информационный сигнал; г) количество достоверных разрядов; д) закон изменения шумов и его параметры; е) закон изменения шумов и его параметры, точность представления входной информации; ж) максимальная скорость изменения сигналов; параметры, определяющие особенности подключения датчиков к ИУВС?

6.5. Перечислить необходимые при разработке ИУВС сведения о сигналах, поступающих с «релейных» датчиков входной информации.

6.6. Изобразить структуры параллельного и последовательного устройств сопряжения с объектом (УСО). Дать характеристику всех узлов, входящих в УСО.

6.7. Изобразить структуру УСО для ввода релейных сигналов, пояснить назначение всех узлов, входящих в состав устройства.

Примечание. Представленные ниже контрольные вопросы направлены на изучение составляющих технического задания, формы его представления, а также вытекающих отсюда основных требований, предъявляемых к техническим параметрам МП ИУВС.

6.8. Перечислить основные составляющие технического задания (ТЗ) на разработку ИУВС.

6.9. Перечислить основные технические параметры при разработке ИУВС.

6.10. Перечислить основные критерии выбора адресности микро-ЭВМ ИУВС.

6.11. В каких микро-ЭВМ (одноадресных или двухадресных) программы идентичных задач занимают меньший объем памяти?

6.12. В каком случае одноадресная ЭВМ обладает преимуществом во времени решения задач перед другими ЭВМ?

6.13. Объяснить, почему при реализации последовательного или параллельного алгоритма трехадресная машина более предпочтительна, чем одноадресная.

6.14. Определить разрядность адреса R_A микро-ЭВМ при использовании способа прямой адресации, если известно, что количество ячеек оперативной памяти $N_{озу} = 4096$; постоянного запоминающего устройства $N_{пзу} = 32768$; перепрограммируемой памяти $N_{ппзу} = 512$; количество источников информации $N_и = 120$; количество приемников информации $N_п = 30$.

6.15. Назвать типовой состав системы памяти ИУВС.

6.16. Объяснить, из чего складывается объем ОЗУ микро-ЭВМ ИУВС.

6.17. Рассчитать объем ОЗУ $N_{озу}$ микро-ЭВМ, если известно, что в микро-ЭВМ реализуется $K = 5$ алгоритмов; максимальные значения одновременно хранимых промежуточных величин для каждого алгоритма $q^1 = 16$; $q^2 = 24$; $q^3 = 4$; $q^4 = 36$; $q^5 = 19$; число уровней прерывания $n = 6$; ячеек памяти для хранения содержимого рабочих регистров микропроцессора $q_п = 15$; входных величин $a = 120$; выходных величин $b = 30$; рабочих ячеек, отводимых под тестовые программы, $N_т = 20$; рабочих ячеек, отводимых под стандартные программы, $N_{сп} = 24$.

6.18. Из чего в основном складывается объем ПЗУ микро-ЭВМ ИУВС?

6.19. Рассчитать объем ПЗУ микро-ЭВМ ИУВС, если известны объемы: рабочих программ для реализации $K = 5$ алгоритмов ($N_{рп1} = 310$; $N_{рп2} = 240$; $N_{рп3} = 740$; $N_{рп4} = 1311$; $N_{рп5} = 2037$); программы-диспетчера ($N_д = 270$); программ обработки прерываний ($N_п = 35$); тестовых программ ($N_т = 630$); стандартных подпрограмм ($N_{сп} = 160$); ячеек для хранения констант $N_с = 40$.

6.20. Объяснить, из чего складывается ошибка выработки управляющих воздействий.

6.21. Рассчитать объемы ППЗУ в случае: а) перезаписи невосстанавливаемых величин; б) постоянного хранения всех невосстанавливаемых величин в ППЗУ.

При этом известно, что число различных алгоритмов $K = 5$; невосстанавливаемых величин, используемых в алгоритмах, $N_{нв1} = 7$; $N_{нв2} = 15$; $N_{нв3} = 17$; $N_{нв4} = 19$; $N_{нв5} = 12$; невосстанавливаемых величин, являющихся общими для всех алгоритмов, $N_{нво} = 3$.

6.22. Перечислить способы, существующие для определения разрядности микро-ЭВМ. Какой из этих способов дает наиболее достоверный результат?

Архитектура микропроцессорных информационно-управляющих вычислительных систем

Ответы на вопросы этого раздела помогут читателю обратить внимание на ключевые моменты в изучении однопроцессорных и мультимикропроцессорных информационно-управляющих вычислительных систем, уяснить особенности построения магистральных, системных и внешних интерфейсов.

6.23. Изобразить структуру однопроцессорной ИУВС, пояснить назначение всех элементов, входящих в ее состав.

6.24. Перечислить основные этапы разработки ИУВС. Дать характеристику каждого этапа.

6.25. Указать причины, обусловливающие необходимость создания мультимикропроцессорных ИУВС.

6.26. По каким параметрам классифицируют особенности архитектуры вычислительной системы?

6.27. Объяснить, чем мультимикропроцессорная ИУВС отличается от многомашинной.

6.28. Назвать основные свойства архитектуры m МП ИУВС.

6.29. Перечислить структуры m МП ИУВС, удовлетворяющие одновременно свойствам параллельности, однозначности и программной изменяемости.

6.30. Изобразить схему m МП ИУВС со связями между модулями через общую память, организованную по принципу «почтового ящика».

6.31. Дать определение интерфейса и привести классификацию интерфейсов по функциональному назначению.

6.32. Объяснить, что понимается под симплексным, полудуплексным, дуплексным и мультиплексным режимами обмена информации.

6.33. Как получить код Манчестер II из кода без возвращения к нулю?

Обеспечение надежности и особенности конструирования микропроцессорных информационно-управляющих вычислительных систем

Настоящий раздел содержит задачи по расчету количественных показателей надежности резервированных систем с использованием теории марковских дискретных случайных процессов. Вопросы настоящего раздела помогут читателю акцентировать внимание на основных принципах конструирования микро-ЭВМ с учетом особенностей их эксплуатации.

6.34. Вычислить вероятность безотказной работы $P(t)$ и наработку на отказ T_0 невосстанавливаемого ОЗУ, состоящего из двух блоков: $ОЗУ_1$ и $ОЗУ_2$ (рис. 10.25, *а*). Невосстанавливаемое ОЗУ обеспечивает хранение, запись и

считывание информации, если в работоспособном состоянии находится по крайней мере один блок ОЗУ$_1$; второй блок *ОЗУ*$_2$ используется в режиме горячего (нагруженного) резерва и может заменить *ОЗУ*$_1$ на период его ремонта. Интенсивности отказов блоков ОЗУ постоянны во времени и равны: $\lambda_1 = \lambda_2$. В процессе эксплуатации ОЗУ обслуживается одной ремонтной бригадой. Оба блока ОЗУ восстанавливаемые. Параметры потоков восстановлений блоков ОЗУ постоянны во времени и равны: $\mu_1 = \mu_2$.

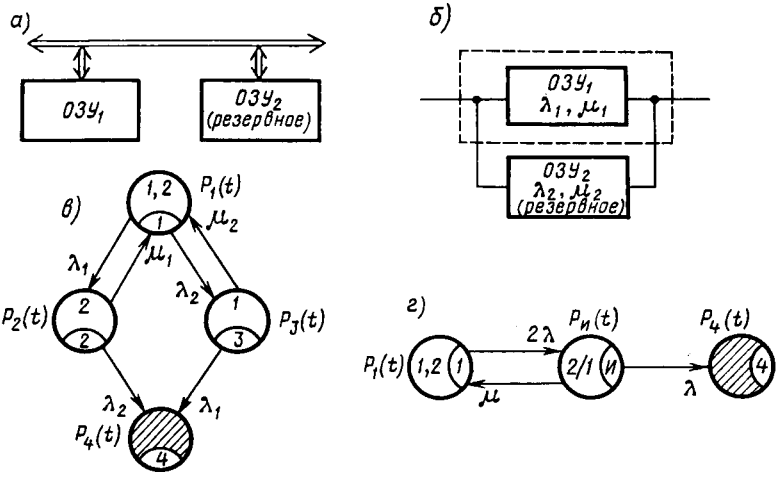


Рис. 10.25. Схемы структурная (*а*) и расчетная надежностная (*б*), полный граф состояний (*в*) и упрощенный граф состояний (*г*) невосстанавливаемого ОЗУ

Потоки отказов и восстановлений блоков ОЗУ считать простейшими. Принять, что в момент включения устройства оба блока ОЗУ исправны.

Р е ш е н и е. Построим расчетную надежностную схему. Описываемому в условии задачи устройству можно поставить в соответствие расчетную надежностную схему, приведенную на рис. 10.25, *б*, где штриховой линией обведена нерезервированная часть невосстанавливаемого в целом устройства, функционирование которого рассматривается лишь до первого отказа. Возможность замены первого (основного) блока *ОЗУ*$_1$ при его отказе вторым (резерв-

ным), работающим в нагруженном режиме, отражена параллельным соединением. Поскольку в расчетной надежностной схеме отдельные элементы могут восстанавливаться, а потоки отказов и восстановлений блоков ОЗУ по условию задачи простейшие, воспользуемся для расчета требуемых показателей надежности теорией марковских случайных процессов.

Для этого рассматриваем возможные состояния системы в период ее эксплуатации, сведя их в табл. 10.7.

Таблица 10.7

Состояние	Блоки ОЗУ			Состояние системы в целом
	работающие	резервные	отказавшие	
1	1	2	—	Рабочее
2	2	—	1	»
3	1	—	2	»
4				Отказовое

Учитывая условия задачи: а) устройство ОЗУ в целом невосстанавливаемое; б) обслуживается одной ремонтной бригадой; в) приоритет обслуживания: первый, затем второй блоки ОЗУ,— строим граф состояний (рис. 10.25, *в*). В изображении вершин графа указываем номер состояния устройства и работающие блоки ОЗУ в этом состоянии. Вершина *4* графа состояний, соответствующая отказовому состоянию, заштрихована. Вершина *2* графа состояния, например, соответствует рабочему состоянию устройства, когда работает $ОЗУ_2$, а отказавшее $ОЗУ_1$ восстанавливается ремонтной бригадой.

Записываем систему дифференциальных уравнений Колмогорова, связывающую вероятности нахождения невосстанавливаемого в целом устройства ОЗУ в любом из возможных его состояний в произвольный момент времени $P_i\ (t)$:

$$\left.\begin{aligned} P_1'(t) &= -(\lambda_1+\lambda_2)P_1(t)+\mu_1 P_2(t)+\mu_2 P_3(t);\\ P_2'(t) &= -(\mu_1+\lambda_2)P_2(t)+\lambda_1 P_1(t);\\ P_3'(t) &= -(\mu_2+\lambda_1)P_3(t)+\lambda_2 P_1(t);\\ P_4'(t) &= \lambda_2 P_2(t)+\lambda_1 P_3(t). \end{aligned}\right\} \qquad (10.5)$$

Напомним, что при записи системы дифференциальных уравнений Колмогорова левая часть каждого уравнения является производной вероятности нахождения устройства в рассматриваемой вершине графа (состоянии устройства), а правая часть содержит столько членов, сколько ребер графа состояний связано с данной вершиной. Если

ребро направлено из данной вершины, то соответствующий член уравнения имеет знак минус, если в вершину — знак плюс. Каждый член уравнения равен произведению параметра потока отказа (восстановления), связанного с данным ребром, на вероятность нахождения устройства в той вершине графа, из которой исходит ребро. Система уравнений Колмогорова включает столько уравнений, сколько вершин в графе состояний устройства.

Решаем систему дифференциальных уравнений (10.5) с учетом того, что в момент включения устройства все блоки ОЗУ исправны, т. е. $P_1(0)=1$, а $P_2(0)=P_3(0)=P_4(0)=0$, вычисляем функции вероятностей $P_i(t)$, а следовательно, и искомые вероятность безотказной работы $P(t)$ и наработку на отказ T_0.

Учитывая условия задачи о равенстве параметров потоков отказов и восстановлений блоков ОЗУ, т. е. $\lambda_1=\lambda_2=\lambda$; $\mu_1=\mu_2=\mu$, и ту особенность вычисления $P(t)$ и T_0, что в процессе их расчета не обязательно знать отдельно функции всех $P_i(t)$, а достаточно знать функции вероятности нахождения устройства по отдельности в рабочих и отказовых состояниях, т. е., в частности, обозначенную $P_2(t)+P_3(t)=P_и(t)$, систему уравнений (10.5) приводим к виду

$$\left.\begin{array}{l} P_1'(t)=-2\lambda P_1(t)+\mu P_и(t); \\ P_и'(t)=-(\mu+\lambda)P_и(t)+2\lambda P_1(t); \\ P_4'(t)=\lambda P_и(t). \end{array}\right\} \qquad (10.6)$$

Системе (10.6) соответствует графа на рис. 10.25, *г*. Функции вероятности $P_1(t)$, $P_и(t)$, $P_4(t)$ вычисляем, применяя преобразование Лапласа к системе (10.6) при начальных условиях $P_1(0)=1$, $P_и(0)=P_4(0)=0$:

$$\left.\begin{array}{l} sP_1(s)-1=-2\lambda P_1(s)+\mu P_и(s); \\ sP_и(s)=-(\mu+\lambda)P_и(s)+2\lambda P_1(s); \\ sP_4(s)=\lambda P_и(s). \end{array}\right\} \qquad (10.7)$$

Решая эту систему уравнений, находим изображения вероятностей:

$$P_1(s)=\Delta_1/\Delta;\ P_и(s)=\Delta_и/\Delta;\ P_4(s)=\Delta_4/\Delta, \qquad (10.8)$$

где

$$\Delta=\begin{vmatrix} (s+2\lambda) & -\mu & 0 \\ -2\lambda & (s+\mu+\lambda) & 0 \\ 0 & -\lambda & s \end{vmatrix}=s[s^2+s(\mu+3\lambda)+2\lambda^2];$$

$$\Delta_1=\begin{vmatrix} 1 & -\mu & 0 \\ 0 & (s+\mu+\lambda) & 0 \\ 0 & -\lambda & s \end{vmatrix}=s(s+\mu+\lambda);$$

$$\Delta_м = \begin{vmatrix} (s+2\lambda) & 1 & 0 \\ -2\lambda & 0 & 0 \\ 0 & 0 & s \end{vmatrix} = s2\lambda;\ \Delta_н = \begin{vmatrix} (s+2\lambda) & -\mu & 1 \\ -2\lambda & (s+\mu+\lambda) & 0 \\ 0 & -\lambda & 0 \end{vmatrix} = 2\lambda^2.$$

Вычисляем функцию вероятности безотказной работы как

$$P(t) = 1 - P_4(t). \tag{10.9}$$

Функцию $P_4(t)$ находим, применяя обратное преобразование Лапласа к выражению (10.8):

$$P_4(s) = \frac{2\lambda^2}{s[s^2 + s(\mu+3\lambda) + 2\lambda^2]} = 2\lambda^2 \frac{N(s)}{sM(s)},$$

где $N(s) = 1$; $M(s) = s^2 + s(\mu+3\lambda) + 2\lambda^2$.

В результате получаем

$$P_4(t) = 2\lambda^2 \left[\frac{N(0)}{M(0)} + \frac{N(s_1)}{s_1 M'(s_1)}\exp(s_1 t) + \frac{N(s_2)}{s_2 M'(s_2)}\exp(s_2 t)\right],$$

где $N(0) = N(s_1) = N(s_2) = 1$; $M(0) = 2\lambda^2$; $M'(s_1) = 2s + \mu + 3\lambda\,|_{s=s_1} = B$; $M'(s_2) = 2s + \mu + 3\lambda\,|_{s=s_2} = -B$; $s_{1,2} = -A/2 \pm B/2$ — корни выражения; $M(s) = 0$; $A = \mu + 3\lambda$; $B = \sqrt{\mu^2 + 6\mu\lambda + \lambda^2}$,

или функция вероятности нахождения устройства в состоянии отказа

$$P_4(t) = 1 + \frac{4\lambda^2}{B(A+B)}\exp\left[-\frac{(A+B)}{2}t\right] - \frac{4\lambda^2}{B(A-B)}\exp\left[-\frac{(A-B)}{2}t\right].$$

Подставив $P_4(t)$ в (10.9), находим вероятность безотказной работы устройства:

$$P(t) = \frac{4\lambda^2}{B(A-B)}\exp\left[-\frac{(A-B)}{2}t\right] - \frac{4\lambda^2}{B(A+B)}\exp\left[-\frac{(A+B)}{2}t\right].$$

Наработку на отказ T_0 устройства ОЗУ вычислим согласно (4.6) из [63]:

$$T_0 = \lim_{s\to 0}[P_1(s) + P_н(s)] = \lim_{s\to 0}\left[\frac{s+\mu+\lambda}{s^2 + s(\mu+3\lambda) + 2\lambda^2} + \right.$$

$$\left. + \frac{2\lambda}{s^2 + s(\mu+3\lambda) + 2\lambda^2}\right] = \frac{\mu+3\lambda}{2\lambda^2}.$$

6.35. Вычислить вероятность безотказной работы $P(t)$ и наработку на отказ T_0 невосстанавливаемого ПЗУ, состоящего из двух идентичных блоков ПЗУ. Невосстанавливаемое ПЗУ обеспечивает хранение и считывание информации, если в работоспособном состоянии находится по крайней мере один из его блоков. Второй блок ПЗУ используется в режиме холодного (ненагруженного) резерва и может заменить первый на период его ремонта. Интенсивности отказов блоков ПЗУ постоянны во времени и равны $\lambda_1 = \lambda_2$. В процессе эксплуатации ПЗУ обслуживается одной ремонтной бригадой. Оба блока ПЗУ восстанавливаемые. Параметры потоков восстановлений блоков ПЗУ постоянны во времени и численно равны $\mu_1 = \mu_2$. Потоки отказов и восстановлений блоков ПЗУ считать простейшими. Принять, что в момент включения устройства оба блока ПЗУ исправны.

6.36. Вычислить вероятность безотказной работы $P(t)$ и наработку на отказ T_0 невосстанавливаемой вычислительной системы, состоящей из трех микро-ЭВМ. Невосстанавливаемая вычислительная система обеспечивает выполнение всех возложенных на нее функций, если в работоспособном состоянии находятся по крайней мере две из них. Третья микро-ЭВМ используется в режиме горячего (нагруженного) резерва и может заменить любую из двух основных на период их ремонта (восстановления). Интенсивности отказов микро-ЭВМ постоянны во времени и равны: $\lambda_1 = \lambda_2 = \lambda_3$. В процессе эксплуатации система обслуживается двумя ремонтными бригадами; каждая бригада в любой момент времени ремонтирует лишь одну микро-ЭВМ. Приоритет обслуживания: первая, вторая, третья микро-ЭВМ. Параметры потоков восстановлений микро-ЭВМ постоянны во времени и равны: $\mu_1 = \mu_2 = \mu_3$. Потоки отказов и восстановлений микро-ЭВМ считать простейшими. Принять, что в момент включения системы все микро-ЭВМ исправны.

● 6.37. Вычислить функцию готовности $K_г(t)$, коэффициент готовности $K_г$, среднее время наработки между отказами $t_{ср}$, среднее время восстановления $t_в$ в вычислительной системе, состоящей из трех микро-ЭВМ (рис. 10.26, *а*). Восстанавливаемая вычислительная система обеспечивает выполнение всех возложенных на нее функций, если в работоспособном состоянии находятся по крайней мере две из них. Третья микро-ЭВМ используется в режиме горячего (нагруженного) резерва и может заменить любую из двух основных на период их ремонта (восстановления).

Интенсивности отказов микро-ЭВМ постоянны во времени и равны: $\lambda_1 = \lambda_2 = \lambda_3$. В процессе эксплуатации система обслуживается двумя ремонтными бригадами; каждая бригада в любой момент времени ремонтирует лишь одну микро-ЭВМ. Приоритет обслуживания: первая, вторая, третья микро-ЭВМ. Параметры потоков восстановлений микро-ЭВМ постоянны во времени и равны: $\mu_1 = \mu_2 = \mu_3$. Потоки отказов и восстановлений микро-ЭВМ считать простейшими. Принять, что в момент первоначального включения системы все микро-ЭВМ исправны.

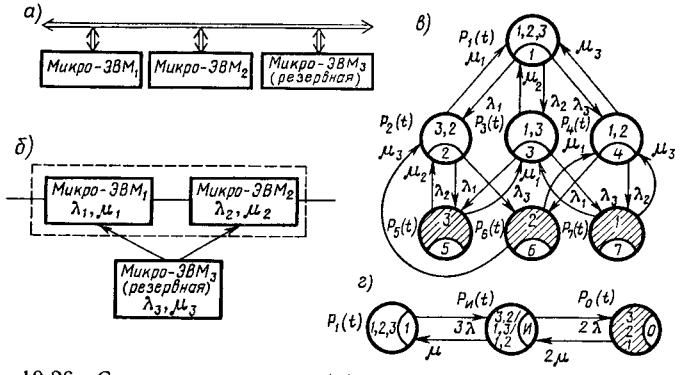


Рис. 10.26. Схемы структурная (*а*) и расчетная надежностная (*б*), полный граф состояний (*в*) и упрощенный граф состояний (*г*)

Р е ш е н и е. Построим расчетную надежностную схему. Описываемой в условии задачи восстанавливаемой вычислительной системе можно поставить в соответствие расчетную надежностную схему, приведенную на рис. 10.26, *б*, где пунктирной линией обведена нерезервированная часть восстанавливаемой системы, представляющая последовательное соединение двух основных микро-ЭВМ, отказ любой из которых приводит к отказу системы. Возможность замены какой-либо из основных микро-ЭВМ резервной отражена линиями со стрелками. Поскольку в расчетной надежностной схеме отдельные элементы могут восстанавливаться, а потоки отказов и восстановлений микро-ЭВМ по условию задачи простейшие, воспользуемся для расчета требуемых показателей надежности теорией марковских случайных процессов. Для этого рассматриваем возможные состояния системы в период ее эксплуатации, сведя их в табл. 10.8.

Таблица 10.8

Состояние	Микро-ЭВМ			Состояние системы в целом
	работающие	резервные	отказавшие	
1	1,2	3	—	Рабочее
2	3,2	—	1	»
3	1,3	—	2	»
4	1,2	—	3	»
5	3	—	1,2	Отказовое
6	2	—	1,3	»
7	1	—	2,3	»

Учитывая условия задачи: а) вычислительная система восстанавливаемая; б) обслуживается двумя ремонтными бригадами; в) в каждый момент времени одна микро-ЭВМ ремонтируется лишь одной бригадой; г) приоритет обслуживания: первая, вторая, третья микро-ЭВМ, — построим граф состояний (рис. 10.26, *в*), отражающий износ («старение») вычислительной системы (рис. 10.26, *б*). В изображении вершин графа указываются номер состояния системы и работающие микро-ЭВМ в этом состоянии. Вершины графа состояний, соответствующие отказовым состояниям, заштрихованы. Вершина *3* графа, например, соответствует рабочему состоянию системы, когда работают микро-ЭВМ$_1$ и микро-ЭВМ$_3$, а микро-ЭВМ$_2$ восстанавливается первой ремонтной бригадой; вершина *6* графа соответствует нерабочему (отказовому) состоянию системы, когда исправна микро-ЭВМ$_2$, микро-ЭВМ$_1$ восстанавливается первой ремонтной бригадой, микро-ЭВМ$_3$ — второй ремонтной бригадой.

Записываем систему дифференциальных уравнений Колмогорова, связывающую вероятности нахождения восстанавливаемой вычислительной системы в любом из возможных ее состояний в произвольный момент времени $P_i(t)$:

$$\left.\begin{aligned}
P_1'(t) &= -(\lambda_1+\lambda_2+\lambda_3)\,P_1(t)+\mu_1 P_2(t)+\mu_2 P_3(t)+\mu_3 P_4(t);\\
P_2'(t) &= -(\mu_1+\lambda_2+\lambda_3)\,P_2(t)+\lambda_1 P_1(t)+\mu_2 P_5(t)+\mu_3 P_6(t);\\
P_3'(t) &= -(\mu_2+\lambda_1+\lambda_3)\,P_3(t)+\lambda_2 P_1(t)+\mu_1 P_5(t)+\mu_3 P_7(t);\\
P_4'(t) &= -(\mu_3+\lambda_1+\lambda_2)\,P_4(t)+\lambda_3 P_1(t)+\mu_1 P_6(t)+\mu_2 P_7(t);\\
P_5'(t) &= -(\mu_2+\mu_1)\,P_5(t)+\lambda_2 P_2(t)+\lambda_1 P_3(t);\\
P_6'(t) &= -(\mu_1+\mu_3)\,P_6(t)+\lambda_3 P_2(t)+\lambda_1 P_4(t);\\
P_7'(t) &= -(\mu_2+\mu_3)\,P_7(t)+\lambda_3 P_3(t)+\lambda_2 P_4(t).
\end{aligned}\right\} \qquad (10.10)$$

Решаем систему дифференциальных уравнений (10.10) с учетом условия задачи, что в момент первоначального

включения системы все микро-ЭВМ исправны, т. е. $P_1(0)=1$, а $P_2(0)=P_3(0)=P_4(0)=P_5(0)=P_6(0)=P_7(0)=0$, найдем функции вероятностей $P_i(t)$, а следовательно, согласно [63], искомую функцию готовности $K_г(t)$, коэффициент готовности $K_г$, среднее время наработки между отказами $t_{ср}$, среднее время восстановления $t_в$. Для этого, учитывая условие о равенстве параметров потоков отказов и восстановлений микро-ЭВМ, т. е. $\lambda_1=\lambda_2=\lambda_3=\lambda$; $\mu_1=\mu_2=\mu_3=\mu$, и ту особенность нахождения $K_г(t)$, $K_г$, $t_{ср}$, $t_в$, что в процессе их вычисления не обязательно знать отдельно функции всех вероятностей $P_i(t)$, а достаточно знать функции сумм вероятностей нахождения системы в отдельных исправных и отказовых состояниях, в частности $P_2(t)+P_3(t)+P_4(t)=P_и(t)$; $P_5(t)+P_6(t)+P_7(t)=P_0(t)$, систему уравнений (10.10) приведем к виду

$$\left.\begin{aligned} P_1'(t)&=-3\lambda P_1(t)+\mu P_и(t);\\ P_и'(t)&=-(\mu+2\lambda)\,P_и(t)+3\lambda P_1(t)+2\mu P_0(t);\\ P_0'(t)&=-2\mu P_0(t)+2\lambda P_и(t). \end{aligned}\right\} \qquad (10.11)$$

Упрощенной системе (10.11) соответствует граф, приведенный на рис. 10.25, *г*.

Функции вероятности $P_1(t)$, $P_и(t)$, $P_0(t)$ определяем, применяя преобразования Лапласа к системе (10.11) при начальных условиях $P_1(0)=1$, $P_и(0)=P_0(0)=0$:

$$\left.\begin{aligned} sP_1(s)-1&=-3\lambda P_1(s)+\mu P_и(s);\\ sP_и(s)&=-(\mu+2\lambda)\,P_и(s)+3\lambda P_1(s)+2\mu P_0(s);\\ sP_0&=-2\mu P_0(s)+2\lambda P_и(s). \end{aligned}\right\} \qquad (10.12)$$

Решая систему алгебраических уравнений (10.12), находим изображения вероятностей:

$$P_1(s)=\Delta_1/\Delta;\ P_и(s)=\Delta_и/\Delta;\ P_0(s)=\Delta_0/\Delta, \qquad (10.13)$$

где

$$\Delta=\begin{vmatrix} (s+3\lambda) & -\mu & 0\\ -3\lambda & (s+\mu+2\lambda) & -2\mu\\ 0 & -2\lambda & (s+2\mu) \end{vmatrix}=s[s^2+s(3\mu+5\lambda)+2(\mu^2+3\lambda\mu+3\lambda^2)];$$

$$\Delta_1=\begin{vmatrix} 1 & -\mu & 0\\ 0 & (s+\mu+2\lambda) & -2\mu\\ 0 & -2\lambda & (s+2\mu) \end{vmatrix}=s^2+s(3\mu+2\lambda)+2\mu^2;$$

$$\Delta_н = \begin{vmatrix} (s+3\lambda) & 1 & 0 \\ -3\lambda & 0 & -2\mu \\ 0 & 0 & (s+2\mu) \end{vmatrix} = 3\lambda s + 6\lambda\mu;$$

$$\Delta_0 = \begin{vmatrix} (S+3\lambda) & -\mu & 1 \\ -3\lambda & (s+\mu+2\lambda) & 0 \\ 0 & -2\lambda & 0 \end{vmatrix} = 6\lambda^2.$$

Вычисляем функцию готовности, используя соотношение (4.9) из [63]:

$$K_г(t) = 1 - P_0(t). \tag{10.14}$$

Функцию $P_0(t)$ определяем, применяя обратное преобразование Лапласа к выражению (10.13):

$$P_0(s) = \frac{6\lambda^2}{s[s^2 + s(3\mu+5\lambda) + 2(\mu^2+3\lambda\mu+3\lambda^2)]} = 6\lambda^2 \frac{N(s)}{sM(s)},$$

где

$$N(s) = 1;\ M(s) = s^2 + s(3\mu+5\lambda) + 2(\mu^2+3\lambda\mu+3\lambda^2).$$

Тогда

$$P_0(t) = 6\lambda^2 \left[\frac{N(0)}{M(0)} + \frac{N(s_1)}{s_1 M'(s_1)} e^{s_1 t} + \frac{N(s_2)}{s_2 M'(s_2)} e^{s_2 t} \right],$$

где $N(0) = N(s_1) = N(s_2) = 1$; $M(0) = 2(\mu^2+3\lambda\mu+3\lambda^2)$;
$M'(s_1) = 2s+3\mu+5\lambda|_{s=s_1} = B$; $M'(s_2) = 2s+3\mu+5\lambda|_{s=s_2} = -B$;
$s_{1,2} = -A/2 \pm B/2$ — корни выражения; $M(s) = 0$;
$A = 3\mu+5\lambda$; $B = \sqrt{\mu^2+6\lambda\mu+\lambda^2}$,

т. е. функция вероятности нахождения системы в состоянии отказа равна

$$P_0(t) = \frac{3\lambda^2}{\mu^2+3\lambda\mu+3\lambda^2} + \frac{12\lambda^2}{B(A+B)} \exp -(A+B)t/2 - \\ - \frac{12\lambda^2}{B(A-B)} \exp -(A-B)t/2. \tag{10.15}$$

Подставив (10.15) в (10.14), найдем функцию готовности системы:

$$K_г(t) = \frac{\mu^2+3\lambda\mu}{\mu^2+3\lambda\mu+3\lambda^2} + \frac{12\lambda^2}{B(A-B)} \exp -(A-B)t/2 - \\ - \frac{12\lambda^2}{B(A+B)} \exp -(A+B)t/2.$$

Коэффициент готовности определяем согласно (4.10) и (4.11) из [63]:

$$K_г = \lim_{t\to\infty} K_г(t) = \lim_{s\to 0} s[P_1(s) + P_и(s)] = \mu(\mu + 3\lambda)/(\mu^2 + 3\lambda\mu + 3\lambda^2).$$

Среднее время наработки между отказами $t_{ср}$ находим, используя соотношение (4.13) в [63], которое в данном случае принимает вид

$$t_{ср} = (P_1^ф + P_и^ф)/(2\mu P_0^ф), \qquad (10.16)$$

где финальные вероятности $P_1^ф$, $P_и^ф$, $P_0^ф$ нахождения системы в состояниях «1», «И», «О» соответственно

$$P_1^ф = \lim_{s\to 0} sP_1(s) = \mu^2/(\mu^2 + 3\lambda\mu + 3\lambda^2);$$

$$P_и^ф = \lim_{s\to 0} sP_и(s) = 3\lambda\mu/(\mu^2 + 3\lambda\mu + 3\lambda^2);$$

$$P_0^ф = \lim_{s\to 0} sP_0(s) = 3\lambda^2/(\mu^2 + 3\lambda\mu + 3\lambda^2).$$

Подставив значения финальных вероятностей $P_1^ф$, $P_и^ф$, $P_0^ф$ в (10.16), получим $t_{ср} = (\mu + 3\lambda)/(6\lambda^2)$.

Среднее время восстановления системы $t_в$ находим, используя соотношение (4.14) из [63], которое в конкретном случае принимает вид $t_в = P_0^ф/(2\mu P_0^ф) = 1/(2\mu)$.

6.38. Вычислить функцию готовности $K_г(t)$, коэффициент готовности $K_г$, среднее время наработки между отказами $t_{ср}$, среднее время восстановления $t_в$ оперативного запоминающего устройства, состоящего из двух блоков ОЗУ. Восстанавливаемое ОЗУ обеспечивает хранение, запись и считывание информации, если в работоспособном состоянии находится по крайней мере один из блоков ОЗУ. Второй блок ОЗУ используется в режиме горячего (нагруженного) резерва и может заменить первый на период его ремонта. Интенсивности отказов блоков ОЗУ постоянны во времени и равны: $\lambda_1 = \lambda_2$. В процессе эксплуатации ОЗУ обслуживается одной ремонтной бригадой. Оба блока ОЗУ восстанавливаемые. Параметры потоков восстановлений блоков ОЗУ постоянны во времени и равны: $\mu_1 = \mu_2$.

Потоки отказов и восстановлений блоков ОЗУ считать простейшими. Принять, что в момент первоначального включения устройства оба блока ОЗУ исправны.

6.39. Вычислить функцию готовности $K_г(t)$, коэффициент готовности $K_г$, среднее время наработки между отказами $t_{ср}$, среднее время восстановления $t_в$ ПЗУ, состоящего из двух идентичных блоков. Восстанавливаемое ПЗУ обеспечивает хранение и считывание информации, если в работоспособном состоянии находится по крайней мере

один из блоков ПЗУ. Второй блок ПЗУ используется в режиме холодного (ненагруженного) резерва и может заменить первый на период его ремонта. Интенсивности отказов блоков ПЗУ постоянны во времени и равны: $\lambda_1 = \lambda_2$. В процессе эксплуатации ПЗУ обслуживается одной ремонтной бригадой. Оба блока ПЗУ восстанавливаемые. Параметры потоков восстановлений блоков ПЗУ постоянны во времени и равны: $\mu_1 = \mu_2$. Потоки отказов и восстановлений блоков ПЗУ считать простейшими. Принять, что в момент первоначального включения устройства оба блока ПЗУ исправны.

6.40. Повышается ли плотность компоновки с использованием микрокорпусов?

6.41. Какими мерами можно увеличить теплопроводность конструкции ИУВС?

6.42. Объяснить, каковы должны быть соотношения между собственными резонансными частотами РЭИ, ячеек, блока при соблюдении правила «октавы».

6.43. Целесообразна ли установка блока с использованием амортизаторов при больших линейных перегрузках?

6.44. Можно ли считать линию связи длиной 20 см, соединяющую логические элементы ЭСЛ-типа, электрически короткой?

6.45. Необходимо ли учитывать отражения в линии связи длиной 1 м, соединяющей логические элементы ТТЛ-типа?

6.46. Емкость или индуктивность межсоединений необходимо уменьшать, конструируя устройства на логических элементах КМДП-типа?

6.47. Индуктивность какого проводника меньше: одиночного или проводника, расположенного вблизи заземленной плоскости?

6.48. Согласована ли кабельная линия связи с волновым сопротивлением 50 Ом, если к ней подключен логический элемент с входным сопротивлением, равным 2 кОм? Чему равен коэффициент отражения?

6.49. Какими мерами достигается уменьшение импульсных помех по цепям питания?

6.50. Для уменьшения внешних электромагнитных помех необходимо:

1) экран не заземлять;

2) заземлять в одной любой точке;

3) заземлять в двух крайних точках. Выберите правильный ответ.

§ 10.7. Автоматизированные системы контроля объектов

Реализация принципа программного управления в АСК обеспечивает их гибкость и возможность широких практических применений. Настройка АСК на выполнение определенных функций происходит путем составления программ. Задача программирования существенно упрощается при использовании диалоговых языков высокого уровня. В [65] описано применение алгоритмического языка ФОКАЛ в качестве одного из возможных вариантов. Приведенные ниже задачи и решения весьма близки к вопросам, с которыми встречается разработчик программного обеспечения на практике.

● 7.1. Написать на языке ФОКАЛ подпрограмму измерения напряжения по одному из каналов с усреднением 10 измерений. Имя функции аналого-цифрового преобразователя FADC (I), где I — номер канала.

Р е ш е н и е: Один из вариантов подпрограммы измерения напряжения с усреднением следующий:

4.1. SET S=0

4.2. FOR K=1, 10; SET S=S+FADC(I)

4.3. SET U(I)=S/10

Для измерения напряжения, например по пятому каналу, и вывода его значения на печать достаточно записать программную строку:

20.1. SET I=5; DO 4; TYPE U(I)

7.2. Составить программу на языке ФОКАЛ, определяющую, находится ли измеренное напряжение по каналу 7 внутри интервала $\pm 10\%$ от номинального значения. В случае выхода измеренного значения напряжения за указанные пределы выдать на печать сообщение: «напряжение по каналу 7 не в допуске». Воспользоваться подпрограммой измерения напряжения, составленной для предыдущей задачи.

7.3. Установить состояние «0» в каналах от 0 до 15. Использовать подпрограмму-функцию FCLR.

7.4. Установить состояние «1» в каналах с нечетными номерами. Использовать подпрограмму-функцию FSET.

7.5. В АСК после установки состояния «1» в канале выдачи 10 ожидается установка состояния «1» в приемном канале 14. Определить с помощью системного таймера промежуток времени (в секундах) между выдачей команды и приемом сигнала.

7.6. С помощью функции управления магистралью FX установить на выходе канала 1 напряжение 2,54 В, на выходе канала 2 напряжение 3,48 В, на выходе канала 3 напряжение 6,59 В, на выходе четвертого — 10,23 В, если модуль аналогового вывода АСК содержит в своем составе

четыре аналого-цифровых преобразователя. Адреса выходных регистров модуля на общей магистрали следующие: канал 1 = 160 100; канал 2 = 160 102; канал 3 = 160 104; канал 4 = 160 106.

7.7. Проанализировать с помощью функции опроса FTST состояние каналов 0—3 и в случае, если все они имеют состояние «1», передать управление на строку 31.5.

7.8. Решить задачу 7.7 с использованием функции FX. Адрес регистра устройства параллельного обмена 176 764.

7.9. Составить программу обслуживания сетевого таймера на языке ассемблера при следующих исходных данных: 1) вектор прерывания находится по адресу 100; 2) сигнал прерывания приходит с частотой 50 Гц; 3) для накопления импульсов таймера используют две соседние ячейки оперативной памяти; 4) содержимое таймера должно быть представлено в виде целого положительного числа.

7.10. Определить максимальную емкость (в часах). Исходные данные те же, что и в задаче 7.9.

7.11. Написать подпрограмму приведения таймера в исходное состояние (состояние «0») на языке ассемблера. Вычислить смешанный код функции.

7.12. Чем определяется точность представления чисел на языке ФОКАЛ?

7.13. От чего зависит диапазон представления чисел в языке ФОКАЛ?

7.14. Назвать способы обмена данными между устройствами, подсоединенными к общей магистрали.

7.15. Назвать рабочие циклы общей магистрали при взаимодействии процессора с внешними устройствами.

7.16. Какие адреса и какой объем адресного пространства выделяется для регистров процессора и регистров внешних устройств?

7.17. Объяснить назначение битов N, Z, V и C регистра состояния процессора.

7.18. Каким образом в команде задаются способы адресации?

7.19. Объяснить, почему на практике применяются только четыре способа адресации с использованием счетчика команд.

7.20. Почему в регистре состояния внешних устройств для битов готовности и ошибки удобно выделять биты 7 и 15?

ОТВЕТЫ

2.2. Команда — двухсловная: первое слово 016304, а второе — 177776. После ее выполнения в регистр R4 будет занесено число 7651; содержимое регистра R3 и слова памяти с адресом 1262 не изменяются. **2.3.** Код команды — 010011, режим адресации первого операнда — регистровый, второго операнда — регистровый косвенный. В слово памяти с адресом 1340 будет занесено число 123456. **2.4.** Команда — двухсловная: первое слово — 067103, второе — 002534. Сумма индекса 2534 и содержимого регистра R1 представляет собой адрес 2606. Так как для первого операнда задан индексный косвенный режим адресации, то содержимое слова с адресом 2606 трактуется как адрес другого слова, в котором и располагается первый операнд, равный 123. После суммирования числа 123 с содержимым регистра R3 в этом же регистре получается результат 4245. **2.5.** Содержимое регистра R1 увеличится на 1 и станет 1341. **2.6.** Код команды — 160233. Результат вычитания, равный 111, будет помещен в слово памяти с адресом 3614. Для второго операнда задан автоинкрементный двойной косвенный режим адресации. **2.7.** Код команды — 005243, в результате ее выполнения содержимое регистра R3 станет 2632, а содержимое слова памяти с адресом 2632 станет 10. В команде указан автодекрементный косвенный режим адресации операнда. **2.8.** Код команды — 020041, после ее выполнения в регистре состояния процессора установятся признаки $N=1$, $Z=0$, $V=0$, $C=1$. Содержимое регистра R0 не изменится, а содержимое регистра R1 станет 1264. **2.9.** Уменьшится на 2 содержимое регистра R4, а в слово памяти с адресом 13204 будет помещено число 7765. Задан автодекрементный двойной косвенный режим адресации второго операнда. **2.10.** Команда — двухсловная: первое слово — 152731, а второе слово — 001234. После ее выполнения в слове памяти с адресом 3000 будет число 12375. Если использовать операцию BIS (вместо BISB), то в слове памяти с адресом 3000 будет число 13375. **2.11.** $Z=1$, $N=V=C=0$. Если вместо CMPB использовать операцию CMP, то $N=Z=V=C=0$. **2.12.** $N=1$, $Z=V=C=0$. Команда занимает три слова памяти. **2.13.** Команда — трехсловная: первое слово — 042767, второе слово — 000012, третье слово — 000014. После ее выполнения в слове памяти с адресом 2472 будет число 21761. Результат не изменится, если вместо операции BIC применить операцию BICB. **2.14.** Число (содержимое регистра R3) помещено в слово памяти с адресом 362. **2.15.** Из стека будет извлечено число 4321 и занесено в слово памяти с адресом 2400. **2.16.** Команда осуществляет выборку содержимого младшего байта из слова памяти с адресом 346 (т. е. числа 321) и занесение этого содержимого в младший байт слова памяти с адресом 3214, в результате чего содержимое этого слова станет 174721. **2.18.** Код команды — 000772, адрес ветвления — 6622. **2.19.** Код команды — 002407, адрес команды — 6616. **2.21.** Адрес ветвления — 6362, код команды — 001442, мнемонический код операции — BEQ. **2.23.** Команда — двухсловная: первое слово содержит код 000167, второе слово — 001056. **2.25.** Заменить директиву .ASECT и следующий за ней оператор прямого присваивания директивой .CSECT. Перед началом программной секции записать директиву .MCALL .EXIT, а оператор останова заменить директивой .EXIT. **2.26.** Рекомендуется сначала изучить текст программы, представлен-

ной в [57] и осуществляющей дублирование перфоленты. Адрес вектора прерывания для клавиатуры 60. Требуемая программа должна в самом начале своей работы занести в первое слово вектора прерывания адрес оператора, с которого начинается подпрограмма обработки прерывания с клавиатуры, а во второе слово — значение 200 (восьмеричное). **2.27.** Вариант текста требуемой программы имеется в [36]. Следует дополнить этот текст командами или директивами формирования значений элементов массива, командой останова, а также директивами начала и конца программной секции. **2.30.** Пример подпрограммы умножения двух целых чисел без знака приведен в [37]. Следует, однако, учесть, что в этом примере допущена ошибка, вследствие которой не обеспечивается возврат в нужное место вызывающей программы. Читателю предлагается выявить и устранить эту ошибку. **2.36.** В требуемой программе необходимо установить ассоциацию между файловой переменной, выбранной программистом для ссылки на файл, и именем реального файла на диске. Для этой цели следует воспользоваться стандартной процедурой RESET с помощью, например, оператора RESET (F, NAME, 'DAT', LEN), в котором F — идентификатор файловой переменной, NAME — текстовая строка, которой предварительно в программе надо присвоить значение 'NUMB', а LEN — целочисленная переменная, которой в результате обращения к процедуре присваивается значение размера файла NUMB.DAT, выраженное в блоках. Рекомендутся обеспечить в программе проверку значения переменной LEN, а если оно равно — 1, то это свидетельствует об отсутствии требуемого файла на диске. **2.37.** Программу целесообразно писать, используя подходящие фрагменты текстов из задач 2.35 и 2.36. Для контроля возникновения ситуации «конец файла» необходимо перед каждой операцией записи в файл проверять значение предиката EOF (F), где F — идентификатор файловой переменной для записываемого файла. **2.38.** Для написания той части программы, которая реализует чтение файла, рекомендуется воспользоваться соответствующим фрагментом программы из задачи 2.36, изменив в нем тип элементов файла. Вариант программы поиска максимального и минимального элемента массива имеется в [41].

2.39. Вариант текста:

```
FUNCTION POWER (X: REAL; N: INTEGER):REAL;
  BEGIN
  IF X=0.0 THEN POWER:=1.0
      ELSE IF N=0 THEN POWER:=1.0
        ELSE IF N<0 THEN POWER:=POWER (X, N=1)/X
          ELSE POWER:=POWER (X, N-1)*X
  END
```

3.2. [45, § 1.1, 1.2]. **3.3.** [46]. **3.4.** [47, § 3.5]. **3.5.** [47, § 3.1]. **3.6.** [46]. **3.7.** [47, § 3.6]. **3.8.** [48, § 2.1]. **3.9.** Открытый ключ, выполненный на биполярном транзисторе, имеет остаточное напряжение $U_{кэ,ост}$, которое отсутствует в ключе, выполненном на униполярном транзисторе. **3.10.** [48, § 6.3]. **3.13.** [45, § 2.1]. **3.14.** Перенос синхронизации — временной сдвиг синхросигналов, приводящий к неправильной передаче информации между регистровыми структурами. **3.15.** [46]. **3.16.** См. решение к 3.12. **3.17.** На рис. 10.27 показан один из возможных вариантов аппаратурного решения поставленной задачи. **3.18.** См. паспортный листок КР1802ВР1. **3.19, 3.20.** [45, § 2.3]. **3.21.** [45, § 2.4]. **3.22 — 3.25.** [45]. **3.27.** Может. **3.28.** [49]. **3.29. — 3.35.** [45, § 2.6]. **3.36, 3.37.** [50, § 7.3]. **3.38, 3.39.** [51]. **3.40.** На рис. 10.28 приведена типичная вольт-амперная характеристика лампы НСМ 6,3×20, из которой видно, что в момент включения она имеет очень низкое сопротивление (участок ВАХ вблизи

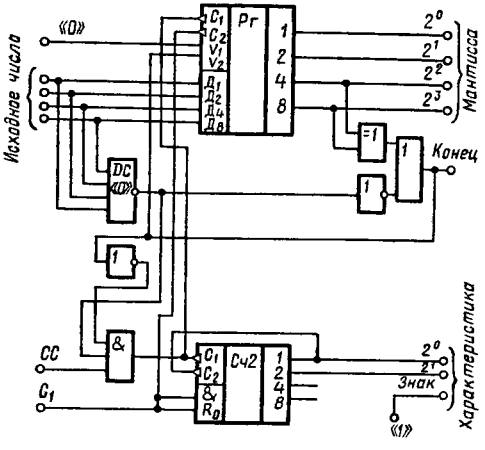
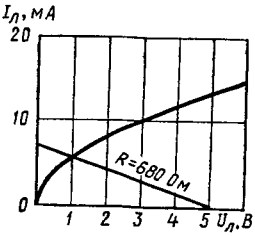


Рис. 10.27. Аппаратурное решение схемы, представленной на рис. 10.10

Рис. 10.28. Вольт-амперная характеристика лампы накаливания НСМ 6,3 × 20



0 В), благодаря чему через транзистор элемента индикации К155ЛА7 может протекать аварийный ток. Включение резистора $R=680$ Ом выводит лампу в режим, при котором $U_л \approx 15\%$ от $U_{л.\ ном}$, и лампа не светится, а динамическое сопротивление лампы возрастает до 350—400 Ом, благодаря чему резко снижается бросок тока при включении транзистора в элементе индикации К155ЛА7. **3.44, 3.45.** [45, § 2.8]. **3.47.** [45]. **3.48.** [45, § 2.9; 48, § 6.4; 6.6]. **3.49.** [45, § 2.9; 3.4]. **3.50, 3.51.** [45, § 2.9] **3.52.** [48, § 6.4]. **3.53.** [45, § 2.9; 48, § 6.4, 6.6]. **3.54.** [45, § 2.10]. **3.55 — 3.58.** [45, § 2.10]. **3.59.** Можно. **3.60.** [45, § 3.1; 48, § 6.1]. **3.61 — 3.65.** [45, § 3.1]. **3.66, 3.67.** [45, § 3.2]. **3.71.** [45]. **3.72.** [45, § 3.3; 48, § 3.0]. **3.73.** [45, § 3.3; 48, § 3.3]. **3.74, 3.75.** [52]. **3.76.** [45, § 3.4; 48, § 6.8]. **3.77.** [45, § 3.4; 48, § 6.8]. **3.80 — 3.86.** [45, § 4.1]. **3.87 — 3.89.** [45, § 4.2; 53]. **3.90.** [53]. **3.91 — 3.93.** [45, § 4.3; 47, § 4.3]. **3.94.** На рис. 10.29 (где *БК* — буферный каскад; *ФУ* — формирователь ТТЛ-уровней) приведена функциональная схема универсального преобразователя уровней.

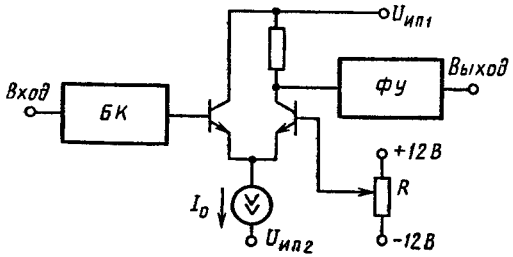


Рис. 10.29. Схема универсального преобразователя уровней

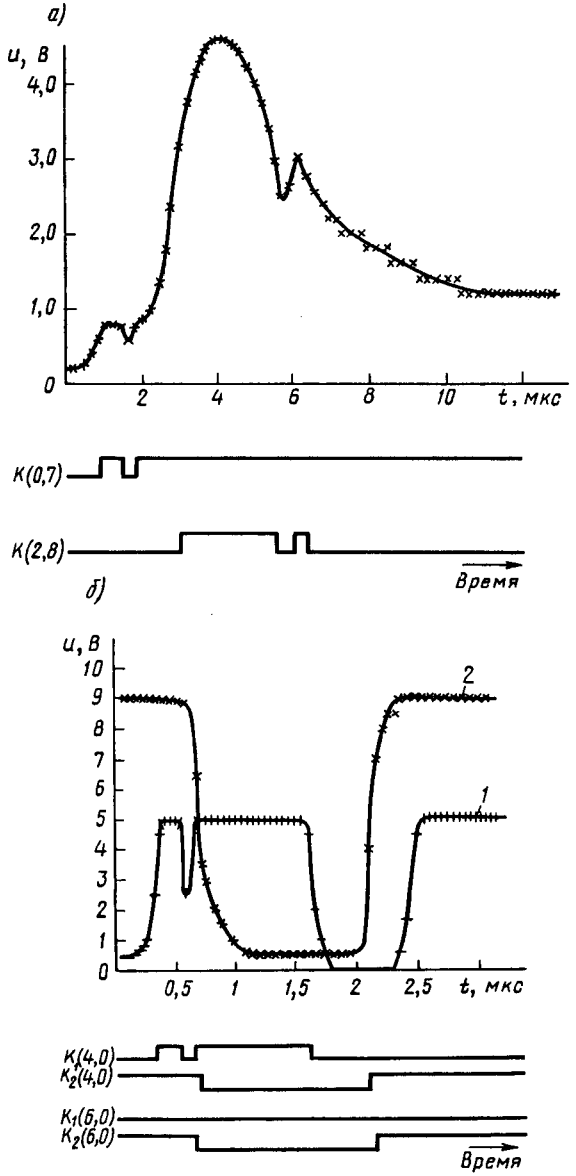


Рис. 10.30. Различные уровни компарации входного сигнала (*а*), представленного на рис. 10.20, *а*, и входного сигнала (*б*), представленного на рис. 10.20, *г*

4.2. [59, § 2.14]. **4.3.** [54, § 3.1]. **4.4.** [57, § 7.2]. **4.5.** [59, гл. 5]. **4.6.** [54, § 3.1]. **4.7.** [54, § 1.1]. **4.9.** [58, § 1.2]. **4.10.** [58, § 1.3]. **4.11.** [58, § 1.4]. **4.12.** [54, § 3.3]. **4.13.** [55, § 6.52]. **4.15.** [54]. **4.18.** [56, гл. 9]. **4.21.** [57, § 7.2]. **4.22.** [54, § 1.4]. **4.25.** [54, § 3.4]. **4.27.** [54, § 1.5]. **4.28.** [55, § 3.12]. **4.35 — 4.37.** [54, § 1.6]. **4.39.** [54, § 1.4.]. **4.40.** [58, § 3.4]. **4.53.** [58, § 3.4]. **4.54.** Программа временной задержки предназначена для устранения дребезга контактов клавиатуры. Длительность задержки задается программно [58]. **4.55.** [58, § 3.4]. **4.56, 4.57.** [59, § 3.4]. **4.59.** [57, § 8.3]. **4.60 — 4.63.** [57, § 8.6]. **4.64, 4.65.** [57, § 8.7]. **4.66.** [57, § 8.6]. **4.67.** [57, § 8.7]. **4.68.** [57, § 8.6]. **4.69.** [57, § 8.7]. **4.70.** [57, § 8.7]. **4.71.** [57, § 8.6]. **4.72.** [57, § 8.7]. **4.73 — 4.75.** [57, § 8.6].

5.2. K (0,5) на рис. 10.20,*г*. **5.3.** K (1,5) на рис. 10.20,*г*. **5.4.** K (0,7) на рис. 10.30, *а*. **5.5.** K (2,8) на рис. 10.30,*а*. **5.6.** K_1 (4,0) и K_2 (4,0) на рис. 10.30,*б*. **5.7.** K_1(6,0) и K_2 (6,0) на рис. 10.30,*б*. **5.9.** Столбец $Д_1$ — табл. 10.3. **5.10.** Столбец $Д_2$ — табл. 10.3. **5.11.** Табл. 10.9. **5.12.** Табл. 10.10. **5.13.** Табл. 10.11. **5.14.** Табл. 10.12. **5.16.** Столбец $Д_4$ в табл. 10.3. **5.17.** Табл. 10.13.

Таблица 10.9

А	Д	А	Д	А	Д	А	Д
01	0	12	1	23	1	34	0
02	0	13	1	24	1	35	0
03	0	14	1	25	1	36	0
04	0	15	1	26	1	37	0
05	0	16	1	27	1	38	0
06	0	17	1	28	1	39	0
07	0	18	1	29	1	40	0
08	1	19	1	30	1	41	0
09	0	20	1	31	1	42	0
10	1	21	1	32	1	43	0
11	0	22	1	33	1	44	0

Таблица 10.10

А	Д	А	Д	А	Д	А	Д
01	0	04	1	07	1	10	0
02	0	05	1	08	1	11	0
03	0	06	1	09	0	12	0

Таблица 10.11

АД	Д	А	Д
01	0	04	1
02	0	05	0
03	1	06	0
		07	0

Таблица 10.12

А	Д
01	0
02	1
03	0

Таблица 10.13

A	2	1	A	2	1	A	2	1	A	2	1	A	2	1	A	2	1
01	1	0	21	0	1	41	0	0	11	1	1	31	0	1	51	1	1
02	1	0	22	0	1	42	1	0	12	1	0	32	0	1	52	1	1
03	1	0	23	0	1	43	1	0	13	1	1	33	0	0	53	1	1
04	1	0	24	0	1	44	1	0	14	0	1	34	0	0	54	1	1
05	1	0	25	0	1	45	1	0	15	0	1	35	0	0	55	1	1
06	1	0	26	0	1	46	1	0	16	0	1	36	0	0	56	1	1
07	1	1	27	0	1	47	1	0	17	0	1	37	0	0	57	1	1
08	1	1	28	0	1	48	1	0	18	0	1	38	0	0	58	1	1
09	1	1	29	0	1	49	1	0	19	0	1	39	0	0	59	1	1
10	1	1	30	0	1	50	1	1	20	0	1	40	0	0	60	1	1

5.20. Процедура ввода данных на магистрали. **5.23.** На рис. 10.31. **5.24.** На рис. 10.32. **5.26.** На рис. 10.33, *а*. **5.27.** На рис. 10.34,*а*. **5.28.** На рис. 10.33,*б*. **5.29.** На рис. 10.34, *б*. **5.31.** Рис. 10.35. **5.33.** С адреса 10 по адрес 17 табл. 10.6 и с адреса 1 по 68 табл. 10.4. Для некоторых задач существует несколько решений. Поэтому приведенные ниже решения не следует рассматривать как единственно возможные.

6.1. [62, § 2.2, рис. 2.7]. **6.2 — 6.5.** [62, § 2.2]. **6.6.** [62, § 2.2, рис. 2.9 и 2.10]. **6.7.** [62, § 2.2, рис. 2.11]. **6.8—6.22.** [62, § 2.2]. **6.23—6.25.** [62, § 2.3]. **6.26.** [62, § 2.3, табл. 2.5]. **6.27—6.29.** [62, § 2.3]. **6.30.** [62, § 2.3, рис. 2.17,*а*]. **6.31—6.33.** [62, § 1.3]. **6.35.** При $\lambda_1=\lambda_2=\lambda$ и $\mu_1=\mu_2=\mu$ 9

$$P(t)=\frac{A+B}{2B}e^{-(A-B)t/2}-\frac{A-B}{2B}e^{-(A+B)t/2};$$

$$A=2\lambda+\mu;\ B=\sqrt{4\lambda\mu+\mu^2};\ T_0=(\mu+2\lambda)/\lambda^2.$$

6.36. При $\lambda_1=\lambda_2=\lambda_3=\lambda$ и $\mu_1=\mu_2=\mu_3=\mu$

```
MOV     #001066     ,R0
MOV     001052      ,(R0)+
MOV     001054      ,R1
MOV     R1          ,(R0)+
TSTB    (R1)+
CMP     R1          ,001056
BNE     001014
MOV     001060      ,R0
MOVB    R1          ,001066(R0)
ADD     R0          ,001062
CMP     R0          ,001064
BNE     001026
BR      001000
000012
HALT
IOT
177777
```

Рис. 10.31. Листинг программы на языке ассемблера ЭВМ «Электроника-60»

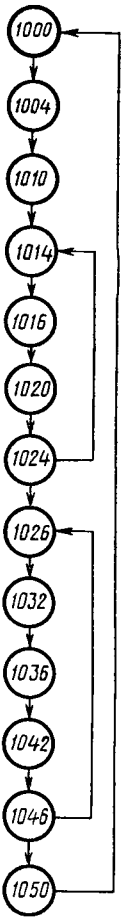


Рис. 10.32. Граф программы, представленной на рис. 10.23

а)
```
001042 -INP- 020067
001044 -INP- 000016
001064 -INP- 002012
001046 -INP- 001367
001026 -INP- 016700
BISLA>>
```

б)
```
/      7 /
/ NUL  SO/
/ EOT  LF/
/ STX  B /
/  GS  @ /
```

Рис. 10.33. Последовательность процедур на магистрали микро-ЭВМ «Электроника НЦ-8001Д» по данным с адреса 8 по 17 (см. табл. 10.7) (*а*) и представление данных в коде ASCII (*б*)

а)
```
001030 -INP- 000026
001060 -INP- 177777
001032 -INP- 110160
001034 -INP- 001066
001065 -OUB- 002000
001036 -INP- 060067
001040 -INP- 000020
001062 -INP- 157634 -OUW- 157633
```

б)
```
/ NUL SYN/
/       /
/ DLE  1 /
/ STX  6 /
/ EOT NUL/
/  0   7 /
/ NUL DLE/
/  _  ESC/
```

Рис. 10.34. Последовательность процедур на магистрали микро-ЭВМ «Электроника НЦ-8001Д» по данным с адреса 18 по 63 (см. табл. 10.7) (*а*) и представление данных в коде ASCII (*б*)

$$P(t)=\frac{12\lambda^2}{B(A-B)}\,e^{-(A-B)t/2}-\frac{12\lambda^2}{B(A+B)}\,e^{-(A+B)t/2};$$

$$A=\mu+5\lambda;\ B=\sqrt{\mu^2+10\lambda\mu+\lambda^2};\ T_0=(\mu+2\lambda)/(6\lambda^2).$$

6.40. [62, § 4.2]. **6.41.** [62, § 4.2]. **6.42.** $f_0^{\text{рэн}}\geqslant 2f_0^{\text{яч}}$; $f_0^{\text{яч}}\geqslant 2f_0^{\text{св}}$. **6.43.** Нет, следует применять «жесткий» монтаж. **6.44.** Нет, такая линия должна рассматриваться как электрически длинная. **6.45.** Да, так как такая линия электрически длинная. **6.46.** Следует выбирать варианты межсоединений с минимальной удельной емкостью. **6.47.** Проводника, расположенного вблизи заземленной плоскости (см. [63, табл. 7.2]). **6.48.** Нет, так как $K_{\text{отр}}=0{,}951$. **6.49.** [62, § 4.2]. **6.50.** Вариант 3.

```
001046 001367                          BNE     001026
001026 016700 000026                   MOV     001060          ;R0
       001030 -INP- 000026                       / NUL SYN/
       001060 -INP- 177777                       /       /
001032 110160 001066                   MOVB    R1              ;001066(R0)
       001034 -INP- 001066                       / STX  6 /
       001065 -OUB- 002000                       / EOT NUL/
001036 060067 000020                   ADD     R0              ;001062
       001040 -INP- 000020                       / NUL DLE/
       001062 -INP- 157634 -OUW- 157633          /  _  ESC/
```

Рис. 10.35. Трасса участка программы по данным с адреса 14 по 67 (см. табл. 10.6)

7.2.

6.1. SET I=7; DO4

6.2. IF(U(I)—1.1*TU(I)6.3, 6.3, 6.4

6.3. IF(U(I)—0,9*TU(I)6.4, 6.5, 6.5

6.4. TYPE «НАПРЯЖЕНИЕ ПО КАНАЛУ», I, «НЕ В ДОПУСКЕ», !; QUIT

6.5. COMMENT ПРОДОЛЖЕНИЕ ПРОГРАММЫ

Пояснение. В программе предварительно должны быть определены номинальные значения напряжений TU(I) по соответствующим каналам с помощью операторов присваивания. Например, если номинальное значение напряжения по каналу 7 должно быть 5 В, то в программе должен присутствовать оператор

SET TU(7)=5

7.3. 15.1. FOR I=0,15; XECUTE ESET(10)

Пояснение. Подпрограмма-функция, вызывающая установку состояния «О» в соответствующем канале, имеет вид FCLR (I), где I — номер канала. 7.4. 30.5 FOR I=1,2,15; XECUTE ESET (I)

7.5.

1.1. SET TO=FCLK(I); XECUTE ESET(10)

1.2. IF (FTST(14) 1.3, 1.2, 1.3

1.3. TYRE «ВРЕМЯ ЗАДЕРЖКИ», (FCLK(I)—TO/50; QUIT

где (FCLK(I), ESET(I), FTST (I) — подпрограммы-функции считывания таймера; установки канала и опроса канала.

Пояснение. Строка 1.1 фиксирует начало отсчета времени и устанавливает «1» в начале выдачи 10; строка 1.2 реализирует ожидание установки «1» в приемном канале с номером 14 (до тех пор, пока состояние канала равно нулю, происходит передача управления этой же строке); строка 1.3 печатает время задержки (в секундах).

7.7.

*XECUTE FX (—1,160100,254)

*XECUTE FX (—1,160102,348)

*XECUTE FX (—1,160104,659)

*XECUTE FX (—1,160106,1023)

Пояснение. В языке ФОКАЛ имеется функция управления общей магистралью FX, которая может быть использована для выполнения операции с нестандартными внешними устройствами или для обращения к ячейкам оперативной памяти. Формат функции FX (операция, адрес общей магистрали, данные).

Операция может иметь значения — 1, 0, +1 для выбора одного из последующих действий: (+1) — чтение, (0) — чтение с маской, (—1) — запись.

Адрес общей магистрали должен быть восьмеричным числом, указывающим адрес слова на общей магистрали.

Данные есть либо переменная, либо десятичное число.

В рассматриваемой версии языка ФОКАЛ все действия функции FX совершаются над словами.

Одновременный сброс всех четырех выходных регистров модуля аналогового вывода можно выполнить с помощью оператора KILL. **7.7.**

```
3.1. IF (FTST(0)*FTST(1)FTST(2)*FTST(3) 3.2, 3.3, 31,5
3.2. TYPE «ОШИБКА», !; QUIT
3.3. COMMENT УСЛОВИЕ НЕ ВЫПОЛНЕНО
  .
  .
  .
31.5 COMMENT КАНАЛЫ 0, 1, 2, 3 в состоянии «1»
```

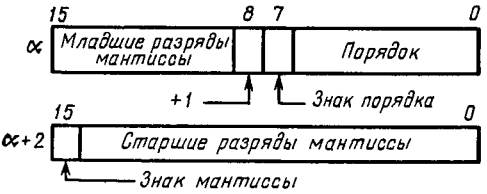
7.8.

```
2.5. IF (FX(0,176764,13)—13) 2.7, 31.5, 2.6
2.6. TYPE «ОШИБКА», !; QUIT
2.7. COMMENT УСЛОВИЕ НЕ ВЫПОЛНЕНО
  .
  .
  .
31.5. COMMENT КАНАЛЫ 0, 1, 2, 3 в состоянии «1»
```

Пояснение. Чтобы прочитать только четыре младших разряда из регистра ввода, воспользуемся маской 1111_2, восьмеричное значение которой 15_8, а десятичное — 13_{10}. Функция управления общей магистралью с аргументами FX(0,176764,13) вызовет чтение слова из регистра с адресом 176764_8, выполнение операции «логическое И» над содержимым регистра и маской и присвоение полученного значения имени функции. **7.9.**

```
                    . =100
              GINT  ; вектор таймера
               200  ; приоритет процессора
                 :  ; подпрограмма обслуживания таймера
GINT:ADD #400, CLKT ; увеличить содержимое таймера на
                      единицу
       ADC CLKT+2   ; прибавить перенос
               RTI  ; возврат из прерывания
CLKT:.WORD27,0      ; рабочие ячейки таймера
```

Пояснение. В языке ФОКАЛ во внутреннем коде число с плавающей точкой представляется в двухслойном формате:



Мантисса содержит 23 двоичных разряда для значащих цифр и плюс один разряд для кодирования знака мантиссы. Подсчет импульсов от сетевого таймера ведется прибавлением единицы к самому младшему раз-

ряду мантиссы. Весовой эквивалент младшего разряда мантиссы равен 2^{-23}, а так как удобнее каждое элементарное приращение представить равным единице, то порядок запишется как +23 или 27_8. Например, счетчик таймера, содержащий 1, будет выглядеть так:

α 000427
α+2 000000

Чтобы каждый раз при поступлении импульса от сетевого таймера содержимое последнего увеличивалось на 1, используют команду

ADD # 400, α

Перенос из младших разрядов мантиссы в старшие разряды совершают по команде

ADC α+2

7.10. $T_{max}=(2^{23}-1)\cdot 0{,}02/3600=8388607\cdot 0{,}02/3600=46{,}6$ ч.

Так как для записи значащих цифр мантиссы отводится 23 двоичных разряда, то максимальное положительное число будет равно $2^{23}-1$. Единичное приращение содержимого таймера соответствует промежутку времени 0,02 с.

7.11.

```
XZCLK:MOV #27,CLKT      ; занести порядок и обнулить
                          младшие разряды мантиссы
                        ; обнулить старшие разряды
CLK CLKT + 2              мантиссы выход из подпрограм-
RTS PC                    мы
```

Для приведения таймера в исходное состояние в разряды порядка нужно записать код 27_8 и нули во все разряды мантиссы. Имея точки входа в подпрограмму, выберем XZCLK.

Пояснение. Методика вычисления смешанного кода имени функции была рассмотрена в [63]. Для нашего случая: F=106; Z=132; C=103; L=114; K=113; $113\times 4^0=113\times 1=113$; $114\times 4^1=114\times 4=460$; $103\times 4^2=103\times 20=2060$; $132\times 4^3=132\times 100=13\,200$; $106\times 4^4=106\times 400=43\,000$. Суммируем полученные значения: 113+460+2060+13 200+43 000=61 053.

Обратить внимание на то, что все вычисления ведутся в восьмеричной системе счисления. Теперь в таблицу функций добавляем две новые строки:

```
XZCLK     ; адрес входа в подпрограмму
          ; обнуление таймера
061053    ; смешанный код имени
          ; функции FZCLK
```

7.12. Точность представления числа зависит от числа двоичных разрядов, отводимых под мантиссу. В языке ФОКАЛ мантисса содержит 23 двоичных разряда (не считая знакового разряда). Известно, что соотношение между числом разрядов десятичной и двоичной систем счисления выражается формулой $n_{дес}\approx 0{,}3010 m_{дв}$. Отсюда точность представления чисел соответствует 6—7 значащим десятичным цифрам. **7.13.** Диапазон представления чисел практически определяется числом двоичных разрядов, отводимых под порядок. В языке ФОКАЛ во внутреннем представлении чисел с плавающей точкой под порядок отводится 1 байт. С учетом знака порядка максимальный порядок равен 2^{+127}, а минимальный -2^{-128}. В десятичной системе счисления диапазон чисел в языке ФОКАЛ заключен в пределах от 10^{-38} до 10^{+38}. **7.14.** [64]. **7.15.** [64]. **7.16.** 160000—177777, 4К слов или 8К байт. **7.17.** [64]. **7.18.** [64]. **7.19.** [64]. **7.20.** При таком выделении битов их опрос упрощается, так как бит 7 определяет знак байта, а бит 15 — знак слова. Проверка состояний битов выполняется с помощью команд TSTB и TST.

СПИСОК ЛИТЕРАТУРЫ

1. Микро-ЭВМ /Под ред. *А. Дирксена.* — М.: Энергоиздат, 1982.
2. *Siewiorek D.P.* and *Lai L. K.* Testing of digital system Proceedings of the IEEE, 1981, vol, 69, № 10.
3. *Northcutt I. D.* The design and implementation of fault insertion capabilities for ISPL — Proceeding 17th IEEE annu design automation conf, 1980.
4. *Шива С.* Языки описания аппаратуры: Методологический обзор — ТИИЭР, 1979, т. 67, № 12.
5. *Monachino M.* Design verification system for large scale LSIdesions — IBM J.Res. and Develoo, 1982, vol, 26, № 1.
6. *Авиженис А.* Отказоустойчивость — свойство, обеспечивающее постоянную работоспособность цифровых систем. — ТИИЭР, 1978, т. 66, № 10.
7. *Хауз Ч.* Особенности проектирования и испытания информационных систем. — Электроника, 1975, т. 48, № 9.
8. *Фарли Б.* Логические анализаторы или системы проектирования? — Электроника, 1979, т. 52, № 19.
9. *Липаев В. В.* Надежность программного обеспечения АСУ. — М.: Энергоиздат, 1981.
10. *Mc. Cann I. and Findlay D.* Microprocessor product design: the roll of the development system. The Radio and Electronie Engineers, 1982, vol. 52, № 2.
11. *Dolch V.* Looikanalyse — Geratetechnik und Anwendungen — Elektronik, 1978, vol 27, № 3.
12. Логические анализаторы — новое средство метрологического обеспечения производства и эксплуатации современной цифровой РЭА. — Радиоэлектроника за рубежом, 1982, вып. 20.
13. *Маршалл М.* Возможности новейших анализаторов временных последовательностей. — Электроника, 1979, т. 52, № 7.
14. *Фарнбах В.* Поиск неисправностей в цифровых схемах с помощью логических анализаторов. — Электроника, 1975, т. 48, № 10.
15. Каталог фирмы «Hewlett—Packard», США, 1981.
16. *Гош Д.* Генератор слов для цифровой отладки, дополняющий любой анализатор. — Электроника, 1982, № 1.
17. Комплекс диагностирования «Электроника НЦ-603».//Тезисы докладов на V Всесоюзной конференции по технической диагностике/. *В. Р. Горовой, Н. П. Васильев, А. В. Бокарев* и др. Суздаль, 1982.
18. *Палмквист С., Херен Г.* Логический анализатор, генерирующий испытательные сигналы. — Электроника, 1981, т. 54, № 18.
19. Генератор и анализатор в одной системе. — Электроника, 1982, т. 55, № 17.
20. *Бейли К., Кахл Т.* Универсальное устройство проектирования систем. — Электроника, 1979, т. 52, № 18.
21. *Иванников А. Д., Старих В. А.* Проектирование микропроцессорных систем. — Зарубежная электронная техника, 1980, № 11.
22. *Мун Дж.* Микрокомпьютер для эмуляции. — Электроника, 1980, т. 53, № 16.

23. Универсальная отладочная система автоматизации проектирования, микропроцессорных устройств: Микропроцессорные средства и системы/*Е. А. Иванов, Л. Л. Муренко, Ю. Ф. Широков.* — Электроника, 1984, № 3.

24. *Ч. Хауз.* Система проектирования, закладывающая основу комплексного центра проектирования СБИС. — Электроника, 1980, т. 53, № 5.

25. *Saponas T.* Development system: soft keys+four buses—easy use and full—speand emulation—Electronic Design, 1979, vol. 27, № 20.

26. Схемный подход к автоматизированному проектированию и отладке микромашинных устройств/*Б. Клайн, М. Мейрз, П. Розенфельд.* — ТИИЭР, 1976, т. 64, № 6.

27. *Васильев Н. П., Горовой В. Р.* Средства проектирования микропроцессорных систем. — Электронная промышленность, 1981, вып. 4 (100).

28. Badogliacca L. New tools build microsystems—Computer Design, 1984, vol. 23, № 1.

29. Микропроцессоры: Основы построения микро-ЭВМ/*В. Л. Горбунов, Д. И. Панфилов, Д. Л. Преснухин.* — М.: Высшая школа, 1984.

30. Микропроцессорные комплекты повышенного быстродействия/*А. И. Березенко, Л. Н. Корягин, А. Р. Назарьян.* — М.: Радио и связь, 1981.

31. *Балашов Е. П., Пузанков Д. В.* Микропроцессоры и микропроцессорные системы. — М.: Радио и связь, 1981.

32. *Горбунов В. Л., Панфилов Д. И.* Микропроцессоры: Лабораторный практикум. — М.: Высшая школа, 1984.

33. *Нестеров П. В.* Микропроцессоры: Архитектура и ее оценка/Под ред. *Л. Н. Преснухина.* — М.: Высшая школа, 1984.

34. *Шаньгин В. Ф., Костин А. Е.* Микропроцессоры: Организация вычислительных процессов на микро-ЭВМ/Под ред. *Л. Н. Преснухина.* — М.: Высшая школа, 1984.

35. Основы программирования на Ассемблере для СМ ЭВМ/*Г. В. Вигдорчик, А. Ю. Воробьев, В. Д. Праченко,* — М.: Финансы и статистика, 1983.

36. *Стоун Г. С., Сиворек Д. П.* Введение в организацию ЭВМ и структуры данных: Пер. с англ. М. А. Зайцева/Под ред. *А. Ф. Волкова.* — М.: Машиностроение, 1980.

37. *Соучек Б.* Микропроцессоры и микро-ЭВМ: Пер. с англ./Под ред. *А. И. Петренко.* — М.: Советское радио, 1979.

38. *Экхауз Р., Моррис Л.* Мини-ЭВМ. Организация и программирование: Пер. с англ. А. Ф. Кондратюка и Л. С. Черняка/Под ред. *Г. П. Васильева.* — М.: Финансы и статистика, 1983.

39. *Перминов О. Н.* Язык программирования ПАСКАЛЬ. — М.: Радио и связь, 1983.

40. *Грогоно П.* Программирование на языке ПАСКАЛЬ: Пер. с англ./Под ред. *Д. Б. Подшивалова.* — М.: Мир, 1982.

41. *Йенсен К., Вирт Н.* ПАСКАЛЬ: Руководство для пользователя и описание языка: Пер. с англ./Под ред. *Д. Б. Подшивалова.*— М.: Финансы и статистика, 1982.

42. *Майоров С. А., Новиков Г. И.* Принципы организации цифровых машин. — Л.: Машиностроение, 1974.

43. Операционная система ОС ДВК ПАСКАЛЬ: Описание языка, 1982.

44. *Гилл А.* Программирование на языке Ассемблера для РДР-11: Пер. с англ. — М.: Радио и связь, 1983.

45. *Воробьев Н. В., Вернер В. Д.* Микропроцессоры: Элементная база

и схемотехника средств сопряжения/Под ред. *Л. Н. Преснухина.* — М.: Высшая школа, 1984.

46. *Коффрон Дж.* Технические средства микропроцессорных систем. Практический курс: Пер. с англ. — М.: Мир, 1983.

47. Расчет элементов цифровых устройств/*Л. Н. Преснухин, Н. В. Воробьев, А. А. Шишкевич;*/Под ред. *Л. Н. Преснухина.* — М.: Высшая школа, 1982.

48. *Гаррет П.* Аналоговые устройства для микропроцессоров и мини-ЭВМ: Пер. с англ./Под ред. *М. В. Гальперина.* — М.: Мир, 1981.

49. *Хофмен.* Использование логических СИС для преобразования двоичных чисел. — Электроника, 1974, № 8.

50. *Гивоне Д., Россер Р.* Микропроцессоры и микрокомпьютеры: Вводный курс: Пер. с англ. — М.: Мир, 1983.

51. Микросхемы и их применение: Справочное пособие. 2-е изд. — М.: Радио и связь, 1983.

52. *Голд Б., Рэйдер Ч.* Цифровая обработка сигналов: Пер. с англ./ Под ред. *А. М. Трахтмана.* — М.: Советское радио, 1973.

53. Стандартные интерфейсы для измерительной техники: Пер. с нем./*Г. Науман, В. Майлинг, А. Щербина.* — М.: Мир, 1982.

54. Проектирование радиоэлектронной аппаратуры на микропроцессорах/*А. Г. Алексенко, А. А. Голицын, А. Д. Иванников.* — М.: Радио и связь, 1984.

55. *Гибсон Г., Лю Ю. Ч.* Аппаратные и программные средства микро-ЭВМ: Пер. с англ. — М.: Финансы и статистика, 1983.

56. 8080A microcomruter interfacing and programming second edition howard w. SAMS & CO., Inc. 1982.

57. Микропроцессоры. В 3-х кн. Кн. 1./*Н. В. Воробьев, В. Л. Горбунов, А. В. Горячев* и др.; Под ред. *Л. Н. Преснухина.* — М.: Высшая школа, 1986.

58. *Горбунов В. Л., Панфилов Д. И.* Применение микропроцессорных устройств и микро-ЭВМ. — М.: Машиностроение, 1983.

59. *Коффрон Дж.* Технические средства микропроцессорных систем: Пер. с англ. — М.: Мир, 1983.

60. *Васильев Н. П., Горовой В. Р.* Микропроцессоры: Аппаратурно-программные средства отладки/Под ред. проф. *Л. Н. Преснухина.* — М.: Высшая школа, 1984.

61. *Тицце У., Шенк К.* Полупроводниковая схемотехника: Справочник: Пер. с нем./Под ред. *А. Г. Алексеенко.*— М.: Мир, 1982.

62. Микропроцессоры. В 3-х кн. Кн. 2./*Н. В. Воробьев, В. Л. Горбунов, А. В. Горячев* и др.; Под ред. *Л. Н. Преснухина.* — М.: Высшая школа, 1986.

63. Расчет цифровых устройств/*Л. Н. Преснухин, Н. В. Воробьев, А. А. Шишкевич;* Под ред. *Л. Н. Преснухина.* — М.: Высшая школа, 1982.

64. *Фролов Г. И., Гембицкий Р. А.* Микропроцессоры. Автоматизированные системы контроля объектов/Под ред. *Л. Н. Преснухина.* — М.: Высшая школа, 1984.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

ОГЛАВЛЕНИЕ

Учебное издание

Николай Васильевич Воробьев,
Владимир Леонидович Горбунов,
Александр Васильевич Горячев,
Владимир Родионович Горовой,
Александр Егорович Костин,
Петр Владимирович Нестеров,
Дмитрий Иванович Панфилов,
Дмитрий Леонидович Преснухин,
Владимир Дмитриевич Вернер,
Владимир Федорович Шаньгин,
Георгий Иванович Фролов,
Александр Адамович Шишкевич

Под редакцией
Леонида Николаевича
Преснухина

МИКРОПРОЦЕССОРЫ
Книга 3.
Средства отладки, лабораторный практикум и задачник

Зав. редакцией *Н. И. Хрусталева*
Редактор *И. Е. Якушина*
Мл. редактор Е. В. Растегаева
Художник И. Н. Павлова
Художественный редактор М. И. Чуринов
Технический редактор А. К. Нестерова
Корректор Г. И. Кострикова

ИБ № 6455

Изд. № СТД—519. Сдано в набор 19.03.86. Подп. в печать 13.08.86. Т-14530. Формат 84×108/32. Бум. тип. № 1. Гарнитура литературная. Печать высокая. Объем 18,48 усл. п. л.+0,21 усл. п. л. форз. 18,90 усл. кр.-отт. 19,28 уч.-изд. л.+0,40 уч.-изд. л. форз. Тираж 90 000 экз. Зак. № 317. Цена 95 коп. Издательство «Высшая школа», 101430, Москва, ГСП-4, Неглинная ул., д. 29/14.

Ордена Октябрьской Революции, ордена Трудового Красного Знамени Ленинградское производственно-техническое объединение «Печатный Двор» имени А. М. Горького Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли. 197136, Ленинград, П-136, Чкаловский пр., 15.