



Г. В. Зеленко
В. В. Панов
С. Н. Попов

Домашний компьютер

Издательство «Радио и связь»



Основана в 1947 году

Выпуск 1139

Г. В. Зеленко

В. В. Панов

С. Н. Попов

Домашний компьютер



Москва
«Радио и связь» 1989



ББК 32.973
З-48
УДК 681.322-181.4

Редакционная коллегия:

В. Г. Белкин, С. А. Бирюков, В. Г. Борисов, В. М. Бондаренко, Е. Н. Геншта, А. В. Гороховский, С. А. Ельяшkevич, И. П. Жеребцов, В. Т. Поляков, А. Д. Смирнов, Ф. И. Тарасов, О. П. Фролов, Ю. Л. Хотунцев, Н. И. Чистяков

Рецензенты: *А. С. Долгий, канд. техн. наук В. И. Мазнев*

Зеленко Г. В. и др.

З-48 Домашний компьютер/Зеленко Г. В., Панов В. В., Попов С. Н.— М.: Радио и связь, 1989.—144 с.: ил.— (Массовая радиобиблиотека; Вып. 1139).
ISBN 5-256-00312-7.

Описываются структура, принцип работы, принципиальные электрические схемы и программное обеспечение персональной микроЭВМ радиолюбителя, построенной на микропроцессоре типа КР580ИК80А. В качестве внешних устройств служат телевизор и кассетный магнитофон. МикроЭВМ может быть использована как универсальная ЭВМ для вычислений, управления разнообразной радиолюбительской и бытовой аппаратурой и создания телеигр, а также может найти применение в качестве инструмента для разработки других микропроцессорных устройств.

Для подготовленных радиолюбителей.

3 $\frac{2404020000-068}{046(01)-89}$ 146-89

ББК.32.973

ISBN 5-256-00312-7

© Издательство «Радио и связь», 1989

ПРЕДИСЛОВИЕ

В нашей стране выпускается более десятка комплектов различных типов микропроцессорных больших интегральных микросхем (БИС), предназначенных для применения в самой разнообразной аппаратуре — от сложных высокопроизводительных вычислительных систем до детских игрушек. Появились стандартные универсальные программируемые элементы — микропроцессорные БИС, структуры которых аналогичны структуре ЭВМ. На основе микропроцессорных БИС строят микропроцессорные системы микроЭВМ.

Все ЭВМ делят на большие, мини- и микроЭВМ. Это деление основывается на таких признаках, как скорость выполнения операций, габаритные размеры и стоимость ЭВМ. Так, большие ЭВМ занимают значительные площади и стоят миллионы рублей, современные мини-ЭВМ размещаются в небольшой комнате и стоят несколько десятков тысяч рублей, а микроЭВМ может быть размещена на одной плате и стоит всего несколько сот рублей. Основные принципы работы всех ЭВМ одинаковы, но область применения микроЭВМ из-за их малой стоимости шире, их можно встраивать в различную аппаратуру, повышая тем самым ее потребительские качества.

Так как специализация микроЭВМ с учетом конкретных функций, выполняемых аппаратурой, достигается с помощью записи в ее память соответствующих программ, то различная по назначению микропроцессорная аппаратура может иметь похожие электрические схемы. Это позволяет унифицировать многие узлы, сократить сроки проектирования и снизить производственные расходы на изготовление микроЭВМ.

Приборы, станки с числовым управлением, роботы-манипуляторы, бытовая и профессиональная радиоаппаратура, системы управления на транспорте, телефонная связь, обучающие системы, домашние ЭВМ, детские игрушки — вот далеко не полный перечень областей применения новой элементной базы.

Применение микропроцессорных БИС в измерительной технике позволяет существенно повысить точность и автоматизировать процессы измерений. Измерительные приборы, выполненные с применением микропроцессорных БИС, обладают способностью самокалибровки и самопроверки, а также могут проводить математическую обработку результатов измерений. Во многих приборах встроенная микроЭВМ позволяет избавиться от панелей с множеством ручек управления. В магнитофоне микроЭВМ выполняет функции управления скоростью движения и натяжения ленты во всех режимах, позволяет автоматически устанавливать ток подмагничивания применительно к конкретному типу ленты, находить нужные записи, программировать последовательность смены режимов работы.

Точно так же микроЭВМ может быть встроена и в любую бытовую и радиолюбительскую аппаратуру, повышая качество ее работы и придавая ей новые функциональные свойства. Области применения микроЭВМ в радиолюбительской практике ограничиваются только фантазией и квалификацией радиолюбителя.

Развитие микропроцессорной техники изменило технологию проектирования цифровых устройств и повысило требования к знаниям их разработчиков. Радиолюбитель, занимающийся проектированием микропроцессорной аппаратуры, должен знать как методы проектирования и отладки электронной аппаратуры, так и программирование. Накопленный опыт показывает, что разработчику электронных схем легче и быстрее освоить методы программирования, чем программисту методы проектирования и отладки аппаратуры. Но и программисты после изучения основ микропроцессорной техники могут с успехом проектировать аппаратуру.

1. ПРИНЦИПЫ РАБОТЫ МИКРОЭВМ

1.1. Структура микроЭВМ и ее система команд

Знакомство с принципами работы микроЭВМ начнем с рассмотрения ее упрощенной обобщенной структурной схемы, представленной на рис. 1.1 и содержащей минимальный набор функциональных элементов, в том или ином виде входящих в состав любой микроЭВМ. Предположим, что элементом, производящим обработку данных, является микропроцессор КР580ИК80А, выполненный в виде БИС, содержащей около шести тысяч транзисторов. Микропроцессор и ряд вспомогательных устройств, обеспечивающих его работу и работу всей микроЭВМ в целом, образуют процессорный модуль, к которому с помощью системных шин подключают периферийные модули микроЭВМ. Разделение микроЭВМ на указанные модули носит функциональный характер. Конструктивно все модули могут быть выполнены, например, на одной плате, более того, имеются однокристалльные микроЭВМ, где все рассматриваемые модули размещены в одной БИС.

Системные шины представляют собой набор соединительных проводников — линий, объединяющих одноименные выводы всех периферийных модулей. По каждой линии может быть передано значение одного разряда двоичного кода в виде уровней напряжения $+0,4$ или $+2,4$ В, соответствующих логическому 0 или логической 1. По роду передаваемой информации все линии разделены на три группы, образующие шины данных, адреса и управления.

Периферийными модулями в микроЭВМ могут быть различные *запоминающие устройства* (ЗУ) и регистры для подключения внешних устройств (например, клавиатуры, устройства визуального отображения информации, различных датчиков и исполнительных механизмов), называемые портами ввода или вывода. Так как микропроцессор КР580ИК80А предназначен для обработки 8-разрядных двоичных чисел (далее будем называть их словами или байтами), то порты ввода или вывода также должны быть 8-разрядными. Запоминающее устройство микроЭВМ состоит из набора 8-разрядных ячеек памяти. Обмен данными между процессорным и периферийными модулями микроЭВМ происходит по шине данных, состоящей также из восьми линий D0—D7. По линии D0 передается младший, а по линии D7 — старший разряды байта.

Характерной особенностью шины данных является ее двунаправленность. Под *двунаправленностью* понимается возможность передачи данных в разные моменты в различных направлениях. Например, сначала по шине данных можно передавать данные от процессорного модуля к периферийному, а затем в обратном направлении. Двунаправленность шины данных обеспечивается трехстабильными буферными регистрами, через которые периферийные модули подклю-

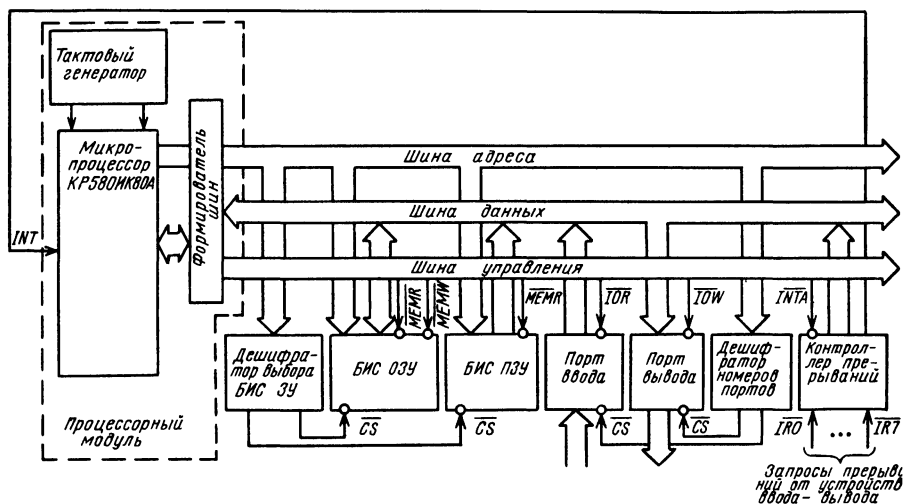


Рис. 1.1. Обобщенная структурная схема микроЭВМ

чаются к шине. Выходы трехстабильных регистров, кроме состояний, соответствующих напряжению высокого и низкого уровней, могут принимать третье пассивное или так называемое высокоимпедансное состояние, благодаря чему они оказываются как бы отключенными от соответствующих линий шины данных.

Каждый периферийный модуль микроЭВМ имеет вход для приема сигнала «Выбор модуля». В процессе работы микроЭВМ с помощью этого сигнала одновременно может активизироваться только один из периферийных модулей. Это означает, что возможен обмен данными между ним и процессорным модулем. Выходы остальных модулей при этом остаются в высокоимпедансном состоянии (отключенном) и на работу микроЭВМ не влияют.

При работе процессорный модуль должен обмениваться данными с определенными ячейками памяти или портами. Для того чтобы иметь возможность обращаться (адресоваться) к ним, каждая ячейка памяти и каждый порт ввода или вывода имеют свои индивидуальные номера — *адреса*. При обмене данными процессорный модуль устанавливает на адресной шине микроЭВМ двоичный код, соответствующий адресу ячейки памяти. Число линий адресной шины микроЭВМ определяется разрядностью адресной шины микропроцессора КР580ИК80А и равно 16. Это позволяет обращаться к $2^{16} = 64$ К ячейкам памяти, где $K = 1024$ байт — единица измерения объема памяти.

Напомним, что в двоичных числах, так же как и в десятичных, значение каждой цифры зависит от того, в каком разряде числа она находится. В двоичной системе счисления только две цифры: 0 и 1. Цифра 0, в каком бы она разряде двоичного числа не находилась, будет эквивалентна нулю в десятичной системе счисления, а цифра 1 в самом младшем (нулевом) разряде двоичного числа эквивалентна $2^0 = 1$ в десятичной системе счисления, в следующем (первом) разряде — $2^1 = 2$, во втором разряде — $2^2 = 4$, в третьем — $2^3 = 8$ и т. д.

Таким образом, 16-разрядное двоичное число, если во всех его разрядах будут единицы, эквивалентно десятичному числу 65535:

$$65535 = 2^{15} + 2^{14} + 2^{13} + \dots + 2^2 + 2^1 + 2^0.$$

Однако, если в 14-м разряде двоичного числа появится 0 (т. е. оно станет равно 101111111111111), его десятичный эквивалент станет меньше на $2^{14} = 16384$ и будет равен 49151.

Конструктивно ЗУ микроЭВМ состоит из одной или нескольких БИС памяти, каждая из которых имеет вход для приема сигнала «Выбор модуля». Дешифрация кода на адресной шине позволяет выбрать определенную БИС ЗУ с помощью соответствующего сигнала. На рис. 1.1 сигнал «Выбор модуля» обозначен CS (chip select). Обращение же к определенной ячейке памяти внутри БИС ЗУ происходит по сигналу с выхода внутреннего дешифратора, входы которого (адресные входы БИС ЗУ) подключаются к соответствующим линиям шины адресов.

Микропроцессор КР580ИК80А позволяет подключить к линиям шины адресов А0—А7 до 256 портов ввода и до 256 портов вывода. Линии шины А8—А15 для подключения портов не используются. Все входы «Выбор модуля» портов ввода или вывода подключаются через дешифраторы номеров портов к восьми младшим разрядам адресной шины микроЭВМ. Порты активизируются при появлении на шине адресов кодов, соответствующих их номерам.

Дополнительным условием активизации любого периферийного модуля является наличие соответствующего сигнала на шине управления, поступающего от процессорного модуля к периферийным. Им является один из сигналов выбора группы модулей (порты или модули памяти) и направления обмена данными: «Чтение памяти» MEMR, «Запись в память» MEMW, «Запись в порт» IOW, «Чтение из порта» IOR. Обычно эти и другие сигналы управления в микроЭВМ формируются в инверсном виде, т. е. активными они являются тогда, когда их уровень соответствует напряжению низкого уровня.

Работа микроЭВМ, как и любого цифрового устройства, заключается в обработке исходных данных по заданному алгоритму. Под *алгоритмом работы* цифрового устройства понимается набор последовательно выполняемых действий по обработке исходных данных с целью получения требуемого результата. В микроЭВМ алгоритм реализуется при выполнении программы, находящейся в ЗУ в виде последовательности команд. При этом исходными для программы являются данные, вводимые через порты ввода, промежуточные данные хранятся в ЗУ микроЭВМ или во внутренних регистрах микропроцессора, а полученные результаты выводятся через порты вывода.

Каждый тип микропроцессора характеризуется определенной системой команд. Система команд — это полный перечень элементарных действий, который способен производить микропроцессор. Управляемый этими командами микропроцессор выполняет очень простые действия, такие как элементарные арифметические и логические операции, операции пересылки данных, сравнения двух величин и др. Составив программу из последовательности таких команд, можно запрограммировать выполнение алгоритма любой сложности.

По формату (числу отведенных для них разрядов) команды микропроцессора КР580ИК80А делятся на одно-, двух- и трех-байтовые. Байты команды

последовательно располагаются соответственно в одной, двух или трех ячейках ЗУ микроЭВМ. Первый байт любой команды содержит код операции. Он определяет формат команды и те действия, которые должны быть произведены микропроцессором над данными в процессе ее выполнения. Эти данные называют *операндами*.

Программа работы микроЭВМ, встроенной в какое-либо устройство, хранится в постоянном запоминающем устройстве (ПЗУ) модуля ЗУ. *Постоянное запоминающее устройство* — это БИС памяти, в которую необходимая информация (программа, константы) заносится в процессе ее изготовления или непосредственно перед установкой в микроЭВМ. Информация в ПЗУ сохраняется независимо от того, включен или выключен источник питания. Во время работы микроЭВМ информацию из ПЗУ можно только считывать. Промежуточные данные в микроЭВМ хранятся в *оперативном запоминающем устройстве (ОЗУ)*, в которое они записываются и из которого считываются в процессе работы. При снятии напряжения питания данные в ОЗУ теряются. При отладке программ, а также когда микроЭВМ используется в качестве универсальной (т. е. выполняющей в разное время различные программы), ОЗУ служит и для хранения программ. В этом случае микроЭВМ обычно имеет ПЗУ с малым количеством ячеек (с малым объемом) памяти, куда записывается небольшая программа-загрузчик, под управлением которой в начале работы в ОЗУ с какого-либо внешнего устройства загружается рабочая программа.

Выполнение любой команды микропроцессора начинается с чтения ее кода операции из ЗУ. Для этого процессорный модуль устанавливает на адресных шинах код адреса ячейки памяти, в которой записан код операции команды, а на соответствующей линии шины управления — сигнал «Чтение памяти». В результате код операции команды выдается из ячейки памяти на шину данных и считывается процессорным модулем.

Микропроцессор декодирует код операции, определяет, какие действия ему необходимо выполнить в соответствии с ним, и переходит к выполнению команды. В зависимости от команды микропроцессор может производить либо только внутренние операции, либо неоднократно обращаться к памяти для чтения или записи данных. В частности, при выполнении двух- и трехбайтовых команд после считывания кода операции процессор обращается к памяти для считывания второго и третьего байта команды. После выполнения текущей команды микропроцессор переходит к выполнению очередной команды, т. е. обращается к ячейке ЗУ, где хранится код операции следующей команды.

Микропроцессор имеет сложную внутреннюю структуру, но с точки зрения программирования он состоит только из семи 8-разрядных регистров А, В, С, D, Е, H, L, регистра признаков результата выполнения операции и двух 16-разрядных регистров SP и PC.

Рассмотрим назначение внутренних регистров микропроцессора. Регистр А (аккумулятор) используется для хранения операнда, с которым работает *арифметико-логическое устройство (АЛУ)* микропроцессора. Результат работы АЛУ по окончании обработки данных вновь помещается в регистр А¹.

¹ При проектировании аппаратуры на базе микропроцессора КР580ИК80А знание принципов организации и работы АЛУ и других недоступных для про-

Шесть регистров В, С, D, E, H, L предназначены для хранения промежуточных данных. При исполнении некоторых команд регистры В и С, D и E, H и L объединяются в регистровые пары для хранения 16-разрядных данных.

Для изучения системы команд и написания программ важно знать способы адресации, которые заложены в микропроцессоре, т. е. знать, как происходит формирование кода на шине адресов.

При обращении к памяти для чтения кода очередной команды из микропроцессора на шину адресов поступает содержимое 16-разрядного регистра РС, называемого счетчиком команд. В этом регистре к моменту окончания выполнения текущей команды всегда подготавливается адрес очередной команды программы. Во время выполнения программы микропроцессору необходимо обращаться к определенным ячейкам памяти для чтения и записи промежуточных данных. В системе команд имеются команды, с помощью которых можно задать адрес обращения непосредственно к памяти (команды с непосредственной адресацией). Они имеют трехбайтовый формат, т. е. каждая команда занимает три ячейки памяти, расположенные друг за другом. В первом байте команды хранится код операции, а во втором и третьем байтах записан 16-разрядный адрес обращения к памяти. При выполнении такой команды микропроцессор последовательно считывает значения второго и третьего байтов во внутренние буферные регистры и затем при обращении к памяти для записи или чтения данных передает из этих регистров на шину адресов 16-разрядный адрес. Команды с непосредственной адресацией выполняются медленно, так как микропроцессору при их выполнении приходится дополнительно дважды обращаться к памяти для побайтного чтения кода адреса.

В системе команд есть также одно- и двухбайтовые команды, в которых используется косвенная регистровая адресация. При их выполнении адресация осуществляется по содержимому одной из регистровых пар BC, DE или HL, куда предварительно помещается адрес требуемой ячейки памяти и откуда он поступает на шину адресов.

Кроме описанных двух способов адресации возможна адресация к ячейкам памяти по содержимому 16-разрядного регистра SP, называемого указателем стека.

Под *стеком* в микроЭВМ на базе микропроцессора КР580ИК80А понимается любая область ОЗУ, служащая для хранения адресов, констант и промежуточных данных, адресация к ячейкам памяти которой осуществляется с помощью команд, работающих со стеком. При обращении к ячейке памяти ЗУ, расположенной в стековой области, на шину адресов выдается содержимое регистра SP. Перед выполнением команд, использующих регистр SP, в него должен быть предварительно записан код начала стековой области ОЗУ (код «верхушки» стека). С помощью команд, использующих стековую адресацию, в стек можно переслать 16-разрядное число из любой регистровой пары или регистра счетчика команд РС. Запись числа в память происходит побайтно: сначала записывается старший байт в ячейку памяти с адресом, на единицу меньшим содержимого указателя стека (т. е. в ячейку с адресом SP—1),

граммиста элементов его внутренней структуры необязательно, поэтому их работа здесь не освещается.

затем — младший байт в ячейку с адресом $SP-2$. Таким образом, по окончании записи содержимое (или положение) указателя стека становится равным $SP-2$. При занесении в стек содержимого регистровых пар или счетчика команд РС указатель стека автоматически каждый раз смещается вниз (т. е. в сторону младших адресов памяти) на две ячейки.

В системе команд микропроцессора есть и такие, которые позволяют осуществлять обратную операцию, т. е. побайтно пересылать содержимое пары ячеек стека в любую регистровую пару или в счетчик команд РС. При этом во внутренний регистр микропроцессора сначала переписывается младший байт из ячейки памяти, адрес которого определяется текущим положением указателя стека, затем в другой регистр регистровой пары переписывается старший байт из ячейки памяти с адресом $SP+1$. После выполнения команды положение указателя стека принимает значение $SP+2$, т. е. указатель стека оказывается автоматически смещенным вверх на две ячейки в сторону старших адресов памяти.

Достоинством команд с адресацией по указателю стека является то, что программист может не заботиться каждый раз о конкретных адресах ячеек памяти, куда записывают и откуда считывают данные. Ему необходимо только соблюдать определенную последовательность при записи данных в стек и их извлечении, т. е. читать данные из стека в последовательности, обратной записи.

Рассмотрим порядок записи 16-разрядных чисел в память и внутренние регистры микропроцессора. Для хранения таких чисел в микропроцессоре можно использовать три регистровые пары — BC, DE, HL, указатель стека SP и счетчик команд PC. При этом в регистрах B, D и H регистровых пар хранятся старшие байты чисел, а в регистрах C, E и L — их младшие байты. В операциях со стеком как 16-разрядное число рассматривается также совокупность регистра A (старший байт) и регистра признаков F (младший байт), именуемая PSW.

Для 16-разрядного числа всегда отводятся две смежные ячейки памяти. Запись чисел в эти ячейки происходит побайтно, причем в ячейку с меньшим адресом записывается младший байт, а в ячейку с большим адресом — старший байт числа. Это правило выполняется при любых способах адресации, а также при записи в память трехбайтовых команд, где второй и третий байты являются соответственно младшим и старшим байтами 16-разрядного числа.

При написании программы для микроЭВМ на уровне машинных кодов программисту необходимо знать ее систему команд. Это означает, что программист должен помнить весь перечень команд, хорошо представлять себе действие микропроцессора при их выполнении.

Код операции любой команды (для однобайтовой команды — это просто код команды) в ЗУ микроЭВМ представляется двоичным 8-разрядным числом. Например, код команды пересылки из регистра C в регистр A будет иметь вид 0111 1001, код операции команды непосредственной записи 8-разрядного операнда в память записывается так: 0011 0110, а команды загрузки аккумулятора с непосредственной адресацией — 0011 1010. Всего двоичным кодом можно представить $2^8=256$ различных комбинаций. Почти столько же команд имеет и микропроцессор (некоторые комбинации двоичных 8-разрядных чисел не используются, и поэтому команд меньше, чем 256).

Естественно, что запомнить более 200 кодов команд, представленных в виде двоичных 8-разрядных чисел, т. е. в виде набора единиц и нулей, почти невозможно. Поэтому каждому коду команды соответствует mnemonicическое ее название, которое является сокращением от английских слов, описывающих ее действие. *Мнемоника* (греч. *mnemōnikā* — искусство запоминания), совокупность приемов и способов, облегчающих запоминание путем образования искусственных ассоциаций. Мнемоника команд позволяет легче запомнить их функции и значительно упрощает написание программ.

После мнемоники двухбайтовых команд записывается их 8-разрядный операнд, который при описании системы команд обозначается D8, а для трехбайтовых команд — 16-разрядный адрес или операнд, обозначаемые соответственно через ADR и D16. Ячейка памяти, адресуемая в соответствии с описанием команды, обозначается M. Так, первая из упомянутых выше команд в mnemonicическом коде будет иметь вид MOV A, C (MOVE REGISTER), вторая — MVI M, D8 (MOVE TO MEMORY IMMEDIATE), а третья — как LDA ADR (LOAD DIRECT).

Названия регистровых пар в мнемонике команд даются в сокращенном виде с помощью первых букв их названия: так, вместо BC, DE и HL записывается соответственно B, D и H. Например, команда увеличения на единицу содержимого регистровой пары HL записывается как INX H.

Все команды микропроцессора КР580ИК80А представлены в табл. 1.1. С ее помощью можно легко и быстро сопоставить мнемонику команды с ее кодом операции. Код операции каждой команды в таблице приведен в верхней и нижней горизонтальных строках (младшие разряды) и в крайних левом и правом столбцах (старшие разряды) в шестнадцатиричном виде.

Далее для наглядности и упрощения записи все двоичные коды будем представлять в шестнадцатиричном виде. Для этого двоичный код числа делится на группы по четыре разряда. Для 8-разрядного кода операции или операнда таких групп будет две, а для 16-разрядного адреса или операнда — четыре. Четырехразрядным двоичным кодом можно представить любое десятичное число от 0 до 15. Обозначив эти величины цифрами от 0 до 9 и далее буквами латинского алфавита от A до F, получим шестнадцатиричные цифры. В табл. 1.2 приведено соответствие между десятичными, двоичными и шестнадцатиричными значениями величин. Например, двоичный код 1100 0011 можно представить в виде шестнадцатиричного числа C3, операнду или коду команды B8, записанному в шестнадцатиричной форме, будет соответствовать код 1011 1000, а адресу F204 — код 1111 0010 0000 0100.

Таким образом, зная мнемонику команды, например ORA C, из табл. 1.1 можно определить ее шестнадцатиричный код операции B1, что будет соответствовать двоичному коду 1011 00001.

При выполнении микропроцессором некоторых команд в регистре признаков F вырабатываются признаки состояния, при этом устанавливаются в 1 следующие биты регистра F.

Бит Z — признак нуля, устанавливается, если результат выполнения команды равен 0.

Бит S — признак знака, устанавливается, если результат выполнения команды отрицателен. При выполнении арифметических команд каждый двоич-

Таблица 1.1.

!	!	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	!	!	
0		NOP	LXI	STAX	INX	INR	DCR	MVI	RLC	-	DAD	LDAX	DCX	INR	DCR	MVI	RRC	0		
1		B, &	B	B	B	B	B	B, &	B	B	B	B	B	B	B	C, &	C, &			
2		-	LXI	STAX	INX	INR	DCR	MVI	RAL	-	DAD	LDAX	DCX	INR	DCR	MVI	RAR	1		
3		D, &	D	D	D	D	D	D, &	D	D	D	D	D	D	D	E	E			
4		-	LXI	SHLD	INX	INR	DCR	MVI	DAA	-	DAD	LHLD	DCX	INR	DCR	MVI	CMA	2		
5		H, &	H	H	H	H	H	H, &	H	H	H	H	H	L	L	L	L			
6		-	LXI	STA	INX	INR	DCR	MVI	STC	-	DAD	LDA	DCX	INR	DCR	MVI	CMC	3		
7		SP, &	SP	M	M	M	M	M, &	SP		SP	SP	A	A	A	A	A			
8		MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	4		
9		B, B	B, C	B, D	B, E	B, H	B, L	B, M	B, A	C, B	C, C	C, D	C, E	C, H	C, L	C, M	C, A	5		
0		MOV	MOV	C, D	D, E	D, H	D, L	D, M	D, A	E, B	E, C	E, D	E, E	E, H	E, L	E, M	E, A			
1		D, B	D, C	D, D	D, E	D, H	D, L	D, M	D, A	E, B	E, C	E, D	E, E	E, H	E, L	E, M	E, A	6		
2		MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV			
3		H, B	H, C	H, D	H, E	H, H	H, L	H, M	H, A	L, B	L, C	L, D	L, E	L, H	L, L	L, M	L, A	7		
4		MOV	MOV	MOV	MOV	MOV	MOV	HLT	M, A	A, B	A, C	A, D	A, E	A, H	A, L	A, M	A, A			
5		M, B	M, C	M, D	M, E	M, H	M, L	M, M	A, B	A, C	A, D	A, E	A, H	A, L	A, M	A, A	A, A			
6		ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADC	ADC	ADC	ADC	ADC	ADC	ADC	ADC	ADC	8		
7		B	C	D	E	H	L	M	A	B	C	D	E	H	L	M	A			
8		SUB	SUB	SUB	SUB	SUB	SUB	SUB	SUB	SBB	SBB	SBB	SBB	SBB	SBB	SBB	SBB	9		
9		B	C	D	E	H	L	M	A	B	C	D	E	H	L	M	A			
A		ANA	ANA	ANA	ANA	ANA	ANA	ANA	ANA	XRA	XRA	XRA	XRA	XRA	XRA	XRA	XRA	A		
B		B	C	D	E	H	L	M	A	B	C	D	E	H	L	M	A			
C		ORA	ORA	ORA	ORA	ORA	ORA	ORA	ORA	CHP	CHP	CHP	CHP	CHP	CHP	CHP	CHP	B		
D		B	C	D	E	H	L	M	A	B	C	D	E	H	L	M	A			
E		RNZ	POP	B	JNZ	JMP	CNZ	PUSH	ADI	RST	RZ	RET	JZ	-	CZ	CALL	ACI	C		
F		B	*	*	*	*	B	*	*	*	*	*	*	*	*	*	*	*		
0		RNC	POP	D	JNC	OUT	CNC	PUSH	SUI	RST	RC	-	JC	IN	CC	-	SBI	D		
1		D	*	*	*	D	*	*	*	*	*	*	*	*	*	*	*	*		
2		RPO	POP	H	JPO	XTLH	CPO	PUSH	ANI	RST	RPE	PCHL	JPE	XCHG	CPE	-	XRI	E		
3		H	*	*	*	H	*	*	*	*	*	*	*	*	*	*	*	*		
4		RP	POP	JP	DI	CP	PUSH	ORI	RST	RM	SPHL	JM	EI	CM	-	CPI	RST	F		
5		PSW	*	*	*	PSW	*	*	*	*	*	*	*	*	*	*	*	*		
6		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F			

N - НОМЕР ПОРТА ВВОДА-ВЫВОДА
 & - ДВУХБАЙТОВЫЙ ОПЕРАНД - D16
 * - ДВУХБАЙТОВЫЙ ОПЕРАНД - ADR
 # - ОДНОБАЙТОВЫЙ ОПЕРАНД - DB

ПРИМЕР: КОМАНДА STAX D ИМЕЕТ
 КОД ОПЕРАЦИИ 12
 КОД ОПЕРАЦИИ CA СООТВЕТСТВУЕТ
 КОМАНДЕ JZ ADR

Таблица 1.2

Значение					
десятичное	двоичное	шестнадцатичное	десятичное	двоичное	шестнадцатичное
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

ный операнд представляется как 7-разрядное двоичное число со знаком, записанным в старшем разряде. Единица в восьмом разряде соответствует отрицательному числу в дополнительном коде.

Бит P — признак четности, устанавливается, если количество единиц в двоичном коде результата четное.

Бит C — признак переноса, устанавливается, если в результате сложения

двух 8-разрядных чисел возникает перенос из старшего разряда или заем — в результате вычитания.

Бит АС — признак вспомогательного переноса, устанавливается, когда перенос возникает из четвертого разряда двоичного числа (из разряда 3). Этот признак используется при различных операциях с 4-разрядными операндами.

Функциональное описание системы команд микропроцессора КР5801К80А приведено в табл. 1.3. В ней рассмотрены действия, которые совершает микропроцессор при их выполнении, показано расположение битов в регистре признаков *F*. Таблица компактна, так как имеется много однотипных команд, выполняющих одинаковые операции над операндами, хранящимися в различных регистрах. Вместо нескольких однотипных команд в таблице помещена одна обобщенная команда (см. условные обозначения). В такой команде вместо обозначения конкретного регистра или регистровой пары используется обобщенное обозначение нескольких регистровых пар. Рядом с командой помещены числа, характеризующие количество тактов, необходимых для выполнения данной команды (см. п. 2.1), через точку с запятой дано условное описание действия команды. Подставляя вместо обобщенного обозначения названия конкретных, допустимых для этой команды регистров или регистровых пар, получаем мнемоничную нужной команды и описание ее работы. Направление пересылки данных при выполнении команды обозначается «→», а через *M* обозначается ячейка памяти, адресуемая по содержимому, приведенному в скобках. Это может быть содержимое одной из регистровых пар, регистра *SP* (указателя стека) или адрес *ADR*, записанный во втором и третьем байтах команды с непосредственной адресацией. Для понимания действия команд необходимо помнить способы адресации, о которых было рассказано ранее. Кратко охарактеризуем группы команд микропроцессора.

Группа команд одnobайтовых пересылок данных. С их помощью производится обмен данными между внутренними регистрами микропроцессора, а также между внутренними регистрами и ячейками памяти.

При выполнении этой команды содержимое регистра *R* пересылается в регистр *R1*, причем в регистре *R* сохраняется прежнее значение данных.

Для пересылок данных между аккумулятором и ячейкой памяти в качестве адреса ячейки памяти может быть использовано также содержимое регистровых пар *BC* или *DE*. Тогда для записи в память данных из аккумулятора используются одnobайтовые команды *STAX B* или *STAX D*, а при обратной пересылке — *LDAX B* или *LDAX D*. Адрес ячейки памяти для обмена с аккумулятором можно задать также с помощью трехбайтовых команд с непосредственной адресацией. В этом случае для записи данных из аккумулятора в память служит команда *STA ADR*, для обратной пересылки — *LDA ADR*.

С помощью двухбайтовой команды *MVI R, D8* (*R* — буквенное наименование регистра или ячейки памяти, адресуемой по содержимому *HL*) можно записать операнд в любой 8-разрядный регистр микропроцессора или ячейку памяти. Операндом здесь будет содержимое второго байта команды.

Группа команд двухбайтовых пересылок данных. Трехбайтовые команды *LXI B, D16*, *LXI D, D16*, *LXI H, D16* служат для непосредственной записи в соответствующие регистровые пары 16-разрядного операнда *D16*.

Используя команды *SHLD ADR*, *LHLD ADR*, можно организовать пересылки данных между регистровой парой и ячейкой памяти, непосредственно

Таблица 1.3

ОДНОБАЙТОВЫЕ ПЕРЕСЫЛКИ

```

MOV R1,R      15/7 1   R ----> R1.
MVI R,DB      17/101   DB ----> R.
STAX YZ       1 7 1   A ----> M(YZ).
LDAX YZ       1 7 1   M(YZ) ----> A.
STA ADR       1 13 1   A ----> M(ADR).
LDA ADR       1 13 1   M(ADR)----> A.

```

КОМАНДЫ ВВОДА И ВЫВОДА

```

IN N          1101 (N) ----> A.
OUT N         1101 A ---->(N).

```

АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ ОПЕРАЦИИ С ОДНИ

```

CMC''         141 C ----> C.
STC''         141 1 ----> C.
CMA          141 A ----> A.
DAA'          141 ДЕСЯТИЧН. КОРРЕКЦИЯ.

```

АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ ОПЕРАЦИИ С ДВУМ
8-БИТОВЫЕ ОПЕРАЦИИ

```

ADD' R 14/71 A+R ----> A.   ADI' DB 171
ADC' R 14/71 A+R+C ----> A.  ACI' DB 171
SUB' R 14/71 A-R ----> A.   SUI' DB 171
SBB' R 14/71 A-R-C ----> A.  SBI' DB 171
ANA' R 14/71 A^R ----> A.   ANI' DB 171
ORA' R 14/71 A^V R ----> A.  ORI' DB 171
XRA' R 14/71 A ⊕ R ----> A.  XRI' DB 171

```

КОМАНДЫ СДВИГА СОДЕРЖ. АККУМУЛЯТОРА

```

RLC'' 141 СДВИГ ВЛЕВО.
RAL'' 141 СДВИГ ВЛЕВО ЧЕРЕЗ БИТ ПРИЗНАКА С.
RRC'' 141 СДВИГ ВПРАВО.
RAR'' 141 СДВИГ ВПРАВО ЧЕРЕЗ БИТ ПРИЗНАКА С

```

СПЕЦИАЛЬНЫЕ КОМАНДЫ

```

EI 141 РАЗРЕШЕНИЕ ПЕРЕРЫВАНИЯ.   КОМАНДЫ
DI 141 ЗАПРЕЩЕНИЕ ПЕРЕРЫВАНИЯ.   CALL
HLT 171 ОСТАНОВ.                  C-CON
NOP 141 ХОЛОСТАЯ ОПЕРАЦИЯ.        RST

```

ФОРМАТ РЕГИСТРА F

```

D7 D6 D5 D4 D3 D2 D1 D0          RET
S Z O AC O P 1 C                  R-CON

```

УСЛОВНЫЕ ОБОЗНАЧЕНИЯ

```

'      - КОМАНДА ОКАЗЫВАЕТ ВОЗДЕЙСТВИЕ
''     - КОМАНДА ОКАЗЫВАЕТ ВОЗДЕЙСТВИЕ
'''    - КОМАНДА ОКАЗЫВАЕТ ВОЗДЕЙСТВИЕ
R,R1   - СОДЕРЖИМОЕ РЕГИСТРОВ А, В, С,
        БОЛЬШЕЕ ЧИСЛО ТАКТОВ КОМАНДЫ С
        ХРАНЯЩИМСЯ В ПАМЯТИ
YZ *    - СОДЕРЖИМОЕ РЕГИСТРОВОЙ ПАРЫ ВС
YZ **   - СОДЕРЖИМОЕ РЕГИСТРОВОЙ ПАРЫ ВС
YZ      - СОДЕРЖИМОЕ РЕГИСТРОВОЙ ПАРЫ ВС
        (YZ В МНЕМОНИКЕ СООТВЕТСТВУЮЩИ
SP      - СОДЕРЖИМОЕ УКАЗАТЕЛЯ СТЕКА ПЕР
DB      - 8-РАЗРЯДНЫЙ ОПЕРАНД (СОДЕРЖИМО
(N)     - СОДЕРЖИМОЕ ПОРТА ВВОДА ИЛИ ВЫВ
D16     - 16-РАЗРЯДНЫЙ ОПЕРАНД (СОДЕРЖИМО
ADR     - 16-РАЗРЯДНЫЙ АДРЕС В ТРЕХБАЙТО
M( )    - СОДЕРЖИМОЕ ЯЧЕЙКИ ПАМЯТИ (АДРЕ
-CON    - ЧАСТЬ МНЕМОНИКИ КОМАНДЫ, ОПРЕД
        ВЫЗОВА И ВОЗВРАТА ИЗ ПОДПРОГРА
        NZ, Z, NC, C, PO, PE, P ИЛИ M)
        КОМАНДЫ ПРИ ВЫПОЛНЕНИИ УСЛОВИЯ

```

ДВУХБАЙТОВЫЕ ПЕРЕСЫЛКИ

```
LXI YZ,D16 ;10; D16 ----> YZ.
SHLD ADR ;16; H ----> M(ADR+1), L ----> M(ADR).
LHLD ADR ;16; M(ADR) ----> L, M(ADR+1) ----> H.
PUSH YZ** ;11; YZ ----> M(SP-1) M(SP-2),
          SP-2 ----> SP.
POP YZ** ;10; M(SP) M(SP+1) ----> YZ,
(POP' PSW) SP+2 ----> SP.
SPHL ; 5; HL ----> SP.
```

ОБМЕН БАЙТАМИ

```
XCHG ; 4; HL <--> DE.
XTHL ;18; H <--> M(SP+1), L <--> M(SP).
```

М ОПЕРАНДОМ

```
INR''' R ;5/10; R+1 ----> R.
DCR''' R ;5/10; R-1 ----> R.
INX YZ ; 5 ; YZ+1 ----> YZ.
DCX YZ ; 5 ; YZ-1 ----> YZ.
```

Я ОПЕРАНДАМИ

```
A+DB ----> A. CPI' DB ; 7 ; УСТАНОВКА ПРИЗ-
A+DB+C ----> A. CMP' R ;4/7; НАКОВ В СООТВ.
A-DB ----> A. C A-DB ИЛИ A-R.
A-DB-C ----> A.
A^DB ----> A. 16-БИТОВЫЕ ОПЕРАЦИИ
A^DB ----> A. DAD'' YZ ; 10; HL+YZ ----> HL.
A @ DB ----> A.
```

КОМАНДЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ

```
PCHL ; 5; HL ----> PC.
JMP ADR ;10; ADR ----> PC.
J-CON ADR ;10; ADR ----> PC.
```

ВЫЗОВА И ВОЗВРАТА ИЗ ПОДПРОГРАММЫ

```
ADR ; 17 ; PC ----> M(SP-1) M(SP-2),ADR ----> PC.
ADR ;11(17); PC ----> M(SP-1) M(SP-2),ADR ----> PC.
X ; 11 ; PC ----> M(SP-1) M(SP-2),
ADD ----> PC, ГДЕ X=0,1,...,7,
ADD СООТВ. РАВЕН 0H,0H,10H,
18H,20H,28H,30H,38H.
; 10 ; M(SP) M(SP+1) ----> PC,
; 5(11); SP+2 ----> SP.
```

НА ВСЕ ПРИЗНАКИ

НА ПРИЗНАК C

НА ВСЕ ПРИЗНАКИ, КРОМЕ ПРИЗНАКА C
D, E, H, L ИЛИ ЯЧЕЙКИ ПАМЯТИ M(HL).
ОТВЕТСТВУЕТ РАБОТЕ С ОПЕРАНДОМ,

, DE, HL ИЛИ РЕГИСТРА SP

ИЛИ DE

, DE, HL ИЛИ PSW

X КОМАНД ЗАМЕНЯЕТСЯ НА B,D,H,SP ИЛИ PSW)

ЕД ВЫПОЛНЕНИЕМ КОМАНДЫ

Е ВТОРОГО БАЙТА ДВУХБАЙТОВОЙ КОМАНДЫ)

ОДА С НОМЕРОМ N (N=0,1,...,255)

ОЕ ВТОРОГО И ТРЕТЬЕГО БАЙТА КОМАНДЫ)

ВОЯ КОМАНДЕ

С ЯЧЕЙКИ УКАЗАН В СКОБКАХ)

ЕЛЯЩАЯ УСЛОВИЕ ПЕРЕДАЧИ УПРАВЛЕНИЯ,

ММЫ (-CON В МНЕМОНИКЕ ЗАМЕНЯЕТСЯ НА

. В СКОБКАХ УКАЗАНО ЧИСЛО ТАКТОВ

ПЕРЕДАЧИ УПРАВЛЕНИЯ

адресуемой по содержимому второго и третьего байтов команды. Остальные команды этой группы осуществляют пересылки с адресацией по указателю стека SP. С помощью команд PUSH B, PUSH D и PUSH H содержимое регистровых пар BC, DE и HL записывается в стек. По команде PUSH RSW в стек записываются данные из аккумулятора и регистра признаков. Команды POP B, POP D, POP H служат для пересылки 16-разрядного слова из ячеек памяти, адресуемых указателем стека SP в соответствующую пару регистров. Командой PSW данные из стека пересылаются в аккумулятор и регистр признаков F. Таким образом, команда POP PSW может изменять все биты регистра признаков F.

Команда SPHL позволяет занести адрес из регистровой пары HL в указатель стека SP.

Группа команд ввода-вывода. Микропроцессор имеет всего две команды для ввода-вывода данных.

С помощью команды ввода IN N можно переписать байт данных в регистр A микропроцессора из одного порта ввода. Номер порта (от 0 до 255) определяется вторым байтом команды.

Аналогично по команде вывода OUT N байт данных из регистра A микропроцессора будет переписан в любой из 256 портов вывода, адресуемых вторым байтом команды.

Группа команд обмена. В этой группе также всего две команды: XCHG — для обмена содержимым между регистровыми парами HL и DE; XTHL — для обмена содержимым между регистровой парой HL и ячейками памяти, адресуемыми по указателю стека SP (при ее выполнении состояние указателя стека не изменяется).

Группа команд арифметических и логических операций с одним операндом. С помощью команды CMC можно изменить значение бита признака переноса на противоположное, т. е. инвертировать признак переноса. Команда STC позволяет установить значение признака переноса в 1. Значения всех битов в регистре A можно инвертировать, применив команду CMA. Команда DAA предназначена для выполнения двоично-десятичного сложения.

Часто при написании программ используются команды INR R, DCR R, INX YZ, DCX YZ, служащие для увеличения или уменьшения значения содержимого регистра, ячейки памяти или регистровой пары на 1. Многие команды этой группы воздействуют на различные биты регистра признаков (табл. 1.3).

Группа команд арифметических и логических операций с двумя операндами. Перед началом выполнения любой команды из этой группы один из операндов должен быть помещен в регистр A, а другой (если команда однобайтовая) — в один из 8-разрядных внутренних регистров микропроцессора или ячейку памяти, адресуемую содержимым регистровой пары HL. В двухбайтовой команде значение второго операнда задается непосредственно во втором байте команды. Результат выполнения команды помещается в регистр A.

Команды ADD R или ADI D8 позволяют сложить два операнда. Сложение двух операндов со значением бита переноса C происходит по команде ADC R или ACI D8. Вычитание из аккумулятора второго операнда и учет значения бита заема C производится соответственно командами SUB R, SUI D8, SBB R или SBI D8.

Операция поразрядного логического умножения (операция И, обозначается знаком \wedge) содержимого аккумулятора со вторым операндом происходит при выполнении команды ANA R или ANI D8. При этом результатом выполнения команды является 8-разрядное двоичное число, отдельные разряды которого равны 1 только тогда, когда соответствующие разряды у обоих операндов также равны 1.

При выполнении поразрядного логического сложения (операция ИЛИ, обозначается знаком \vee) с помощью команды ORA R или ORA D8 образуется двоичное число, отдельные разряды которого равны 1 в случае, когда соответствующие разряды какого-либо одного или обоих операндов также равны 1.

Результатом выполнения операции Исключающее ИЛИ (обозначается знаком \oplus) командами XRA R или XRI D8 является байт, отдельные разряды которого равны 1 только тогда, когда соответствующие разряды операндов имеют противоположные значения.

Рассмотрим пример различных логических операций над двумя операндами.

Первый операнд	1 0 0 1 0 0 1 1
Второй операнд	1 1 0 0 1 1 1 0
<hr/>	
Результат операций:	\wedge 1 0 0 0 0 0 1 0
	\vee 1 1 0 1 1 1 1 1
	\oplus 0 1 0 1 1 1 0 1

После выполнения рассмотренных команд логической обработки двух операндов значения признаков C и AC регистра признаков F всегда равны 0.

Команды CMP R и CPI D8 позволяют сравнивать два операнда. Сравнение происходит вычитанием из первого операнда, хранящегося в аккумуляторе, второго. Если в результате операции вычитания окажется, что операнды равны, то признак нуля устанавливается в 1, если же значение операнда, хранимого в аккумуляторе, меньше значения второго операнда, то устанавливается в 1 признак переноса C.

Есть в системе команд микропроцессора команды DAD B, DAD D, DAD H, DAD SP, позволяющие сложить два 16-разрядных числа. Одно из этих чисел должно быть записано в регистровую пару HL, а другое — в регистровую пару BC, DE, HL или SP. Результат сложения помещается в пару HL.

Группа команд сдвигов содержимого аккумулятора. На рис. 1.2 схематически показано, как происходит сдвиг содержимого аккумулятора влево или вправо командами сдвига RAL и RAR и командами циклического сдвига RLC и RRC. В операциях сдвига участвует бит переноса C регистра признаков F. Под воздействием каждой из этих команд происходит сдвиг содержимого аккумулятора только на один разряд. Если необходимо сдвинуть содержимое аккумулятора на большее число разрядов, то команду сдвига следует повторить требуемое число раз.

Группа команд передачи управления и работы с подпрограммами. Эти команды играют особую роль в организации выполнения программ микроЭВМ. Пока в программе не встречаются команды этой группы, счетчик команд PC постоянно увеличивает свое значение и микропроцессор выполняет команду за командой в порядке их расположения в памяти.

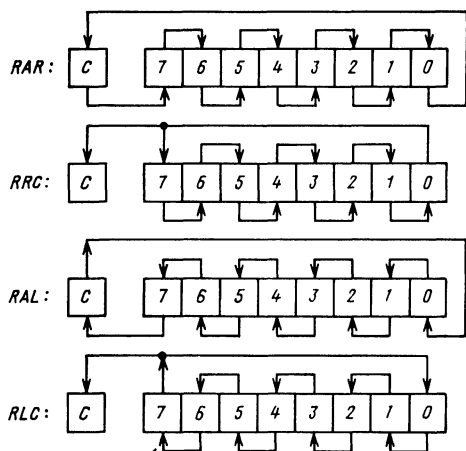


Рис. 1.2. Схема выполнения команд сдвига содержимого аккумулятора

Порядок выполнения программы может быть изменен, если занести в регистр счетчика команд микропроцессора код адреса, отличающийся от адреса очередной команды. Это вызовет передачу управления работой микропроцессора другой части программы. Такая передача управления (или переход в программе) может быть выполнена с помощью трехбайтовой команды безусловного перехода — `JMP ADR`. Как только такая команда встретится в программе, в регистр счетчика команд PC микропроцессора запишется значение ADR. Таким образом, следующей командой,

которую выполнит микропроцессор вслед за командой `JMP ADR`, будет та, код операции которой записан в ячейке с адресом, равным значению ADR.

Безусловную передачу управления можно произвести также по команде `PCHL`, в результате выполнения которой произойдет передача управления по адресу, хранящемуся в регистровой паре HL.

Кроме команды безусловного перехода микропроцессор имеет восемь трехбайтовых команд условного перехода. При появлении команды условного перехода передача управления по адресу, указанному в команде, происходит только в случае выполнения определенного условия. Если условие не удовлетворяется, то выполняется команда, непосредственно следующая за командой условного перехода.

Условия, с которыми оперируют команды условной передачи управления, определяются состоянием битов (разрядов) регистра признаков F:

NZ (NOT ZERO)	— ненулевой результат $Z=0$;
Z (ZERO)	— нулевой результат $Z=1$;
NC (NO CARRY)	— отсутствие переноса, $C=0$;
C (CARRY)	— перенос, $C=1$;
PO (PARITY ODD)	— нечетный результат, $P=0$;
PE (PARITY EVEN)	— четный результат, $P=1$;
P (PLUS)	— число неотрицательное, $S=0$;
M (MINUS)	— число отрицательное, $S=1$.

Эти условия проверяются соответствующими командами условного перехода: `JNZ ADR`, `JZ ADR`, `JNC ADR`, `JC ADR`, `JPO ADR`, `JPE ADR`, `JP ADR`, `JM ADR`.

При написании программ можно выделить одинаковые последовательности команд, нередко встречающиеся в разных частях. Для того чтобы многократно не переписывать такие последовательности команд, их объединяют в подпрограммы. В любой части основной программы программист может поставить трехбайтовую команду безусловного вызова подпрограммы `CALL ADR`, во вто-

ром и третьем байтах которой указывается адрес вызываемой подпрограммы. Выполнение команды CALL ADR начинается с побайтовой засылки в стек адреса следующей после этой команды ячейки памяти. Этот адрес называется адресом возврата из подпрограммы. При уходе на подпрограмму его необходимо запомнить для того, чтобы по окончании выполнения подпрограммы вернуться к продолжению выполнения основной программы.

После записи в стек адреса возврата из подпрограммы в счетчик команд PC микропроцессора загружается значение ADR, т. е. адрес первой команды вызываемой подпрограммы. Таким образом, управление из основной программы передается на вызываемую подпрограмму.

Выполнение подпрограммы всегда заканчивается командой возврата из подпрограммы, например, однобайтовой командой безусловного возврата из подпрограммы RET. При этом содержимое стека, т. е. адрес возврата из подпрограммы, пересылается из стека в регистр PC микропроцессора и управление вновь передается основной программе.

Кроме трехбайтовой команды CALL ADR безусловного вызова подпрограммы, располагаемой по произвольному адресу, в системе команд микропроцессора имеется восемь однобайтовых команд RST 0—RST 7 вызова подпрограмм, расположенных по фиксированным адресам. Появление в основной программе любой из этих команд вызывает запись в стек адреса возврата из подпрограммы и передачу управления на соответствующую ячейку памяти, где расположена первая команда подпрограммы. Соотношение между командами RST 0—RST 7 и шестнадцатиричными адресами ячеек памяти, куда передается управление при их выполнении:

<i>Команда</i>	<i>Адрес начала подпрограммы</i>	<i>Команда</i>	<i>Адрес начала подпрограммы</i>
RST 0	0000H	RST 4	0020H
RST 1	0008H	RST 5	0028H
RST 2	0010H	RST 6	0030H
RST 3	0018H	RST 7	0038H

Кроме команды безусловного вызова и возврата из подпрограмм, в системе команд имеется восемь команд условного вызова подпрограмм и восемь команд условного возврата из подпрограмм, действие которых определяется, так же как у команд условной передачи управления, состоянием регистра признаков F. Если условие для выполнения команды отсутствует, то вызов подпрограммы или возврат из нее не выполняется.

Группа специальных команд. Команда NOP этой группы не производит никаких операций, однако так как она выполняется за определенное время, ее можно использовать в программах для задания временных интервалов. Программисты используют эту команду также при отладке программ, записывая ее на место удаленных. Появление в программе команды HLT ведет к останову выполнения программы. Продолжить выполнение программы можно только подачей сигнала «Сброс» или «Прерывание» на соответствующие входы микропроцессора. Действие команд EI (Разрешение прерывания) и DI (Запрет прерывания) будет рассмотрено в приложении.

1.2. Программирование в машинных кодах

Микропроцессорные устройства и микроЭВМ выполняют свои функции в соответствии с программами, составленными и отлаженными разработчиками и записанными затем в память машины. При разработке и отладке микроЭВМ необходимо уметь писать программы в машинных кодах.

Каким же образом составляют такие программы? Рассмотрим простой пример. Требуется составить программу работы специализированного микропроцессорного устройства. Пусть в нем имеются два порта: ввода D2 и вывода D4. К порту ввода D2 по линии, связанной с его младшим разрядом, подключена кнопка S1, а к порту вывода D4 по линиям, связанным с его двумя младшими разрядами, подключены два светодиода. Функциональная схема части этого устройства приведена на рис. 1.3. Дешифраторы D1 и D3 формируют сигналы «Выбор модуля 1» $\overline{CS1}$ и «Выбор модуля 2» $\overline{CS2}$, служащие для активизации соответствующих портов. При этом дешифраторы включены таким образом, что эти сигналы появляются только тогда, когда на младших восьми разрядах шины адресов возникают коды 0000 0001 и 0000 0010 соответственно. В таком случае говорят, что порт D2 включен как устройство с номером 1, а порт D4 — с номером 2. Далее порт D2 будем называть портом 1, а порт D4 — портом 2.

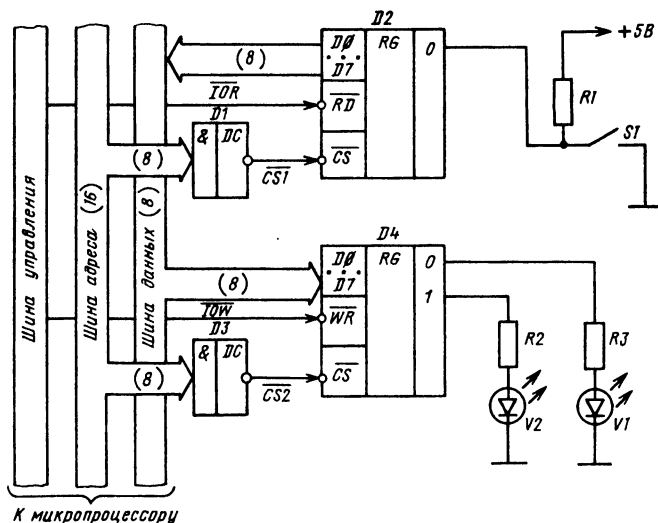


Рис. 1.3. Фрагмент функциональной схемы специализированного устройства

Допустим, что от устройства требуется выполнение следующей задачи. В исходном состоянии, пока кнопка S1 не нажата, светодиод V1 должен гореть, а светодиод V2 нет. Если кратковременно нажать на кнопку, то светодиод V1 должен на 0,25 с погаснуть, а светодиод V2 загореться. После этого в течение 0,5 с устройство не должно реагировать на нажатие кнопки.

Конечно, использовать микроЭВМ для решения такой простой задачи нецелесообразно, но для знакомства с принципами составления программ именно такая задача нам и нужна.

Итак, программа должна начинаться с записи в порт 2 комбинации битов 00000001. Именно при этом условии V1 горит, а V2 погашен. Затем по команде ввода содержимое порта 1 пересылается в аккумулятор и анализируется, не появился ли нуль в младшем разряде, что может быть только при нажатии на кнопку. Если кнопка не нажата, то операция повторяется. В противном случае в порт 2 посылается комбинация 00000010 (светодиод V2 горит, V1 погашен). После этого микропроцессор в течение 0,25 с не должен производить никаких операций с портами, а затем после задержки, равной 0,5 с, вновь перейти к началу нашей программы. Эти задержки можно будет выполнить программно с помощью специальной подпрограммы.

Занесем все перечисленные действия в табл. 1.4 и пронумеруем каждый шаг. Таблица — это, по существу, алгоритм решения поставленной задачи. Представление алгоритма в виде таблицы не очень удобно. Более наглядны структурные схемы алгоритмов. Они могут состоять всего из четырех основных элементов (рис. 1.4). Структурная схема нашего алгоритма приведена на рис. 1.5. Сплошными линиями со стрелками показан ход его выполнения. Справа дана программа, под управлением которой микропроцессор выполняет алгоритм. Штриховые линии со стрелками лишь указывают на соответствие элементов алгоритма и команд программы. Эти линии не являются принадлежностью структурной схемы и включены в данном случае только для наглядности.

Вновь обратимся к табл. 1.4 и увидим, что при выполнении алгоритма нам неоднократно придется возвращаться к шагам 1 и 3. Поэтому эти шаги должны быть особо обозначены метками — произвольно выбираемыми именами. На рис. 1.5 в штриховых прямоугольниках в соответствующих местах схем алгоритма показаны эти метки. Они обычно содержат знак двоеточия после своего имени, а имена меток выбираются таким образом, чтобы было ясно их назначение (в рассматриваемом примере будем использовать метки НАЧАЛО, ВВОД, ВРЕМЯ).

Таблица 1.4

Шаг	Выполняемое действие
1	Записать в аккумулятор код 00000001 и переслать его в порт вывода 2
2	Выполнить программу, реализующую задержку в 0,5 с
3	Переслать содержимое порта ввода 1 в аккумулятор
4	Проверить состояние нулевого разряда аккумулятора
5	Если нулевой разряд аккумулятора равен 0, то перейти к шагу 6, в противном случае — к шагу 3
6	Записать в аккумулятор код 00000010 и переслать его в порт вывода 2
7	Выполнить программу, реализующую задержку в 0,25 с
8	Перейти к шагу 1

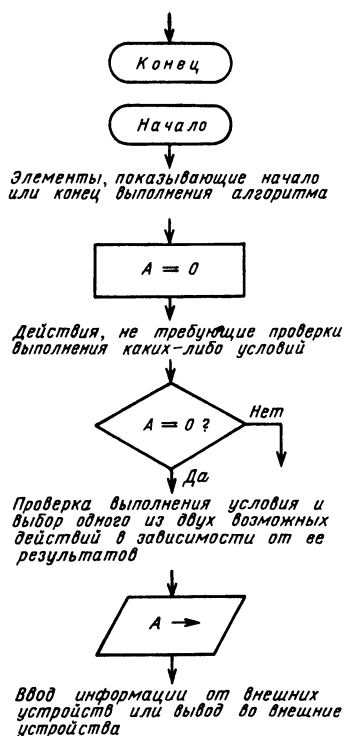


Рис. 1.4. Основные элементы схем алгоритмов

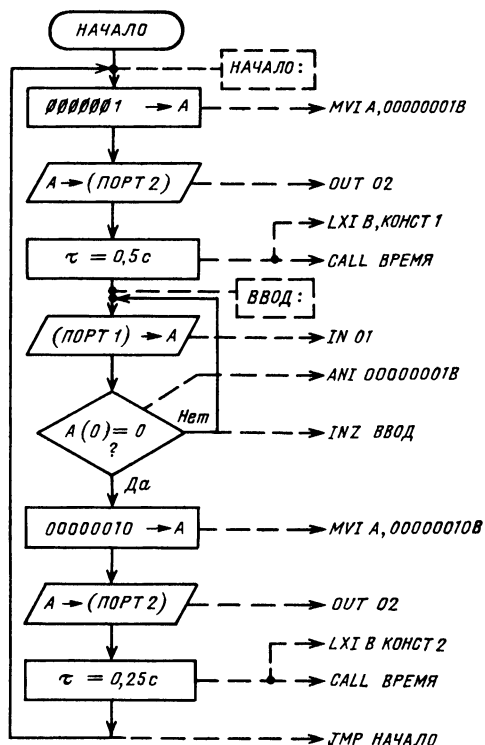


Рис. 1.5. Алгоритмы функционирования специализированного устройства

Теперь, когда структурная схема алгоритма составлена, можно перейти к написанию программы.

Пользуясь описанием системы команд и табл. 1.3, можно привести устройство в начальное состояние с помощью последовательности команд загрузки операнда в аккумулятор MVI A, 00000001 и вывода байта данных из аккумулятора в порт 2 — OUT 02. Затем следуют две команды, необходимые для организации временной задержки (их действие мы разберем несколько позже), и далее команда опроса состояния кнопки S1 (т. е. значения младшего разряда порта 1). С помощью команды ввода IN 01 перепишем содержимое порта в аккумулятор и проверим состояние его младшего бита A(0). Для этого воспользуемся командой логического умножения содержимого аккумулятора на непосредственный операнд (команда ANI 00000001), содержащий 1 только в младшем разряде. В результате выполнения этой команды в аккумулятор запишется код, семь старших разрядов которого будут равны нулю. Значение нулевого разряда аккумулятора будет зависеть от значения младшего разряда кода, прочитанного из порта 1, т. е. от того, была ли нажата кнопка в момент действия команды IN 01.

Команда логического умножения воздействует на все биты признаков ре-

гистра F. В данном случае нас интересует состояние бита Z — признака нуля, который устанавливается в 1 при нулевом содержимом аккумулятора, что в данном случае соответствует нажатой кнопке. Таким образом, команда JNZ ВВОД будет передавать управление команде IN 01, помеченной меткой ВВОД, до тех пор, пока признак Z остается равным нулю, нажатие на кнопку приведет к установке признака Z в 1 и выполнению последующих команд программы.

При написании программы мы пользовались mnemonicскими обозначениями команд микропроцессора и символическими именами для меток. Конечно, программа в таком виде не может быть непосредственно выполнена на микроЭВМ, ее нужно перевести на машинный язык — язык двоичных (или объектных) кодов. Такой перевод мы будем делать, пользуясь специальным бланком. В табл. 1.5 приведена форма такого бланка с записанной на нем программой. Этот документ программисты называют *распечаткой программы*. Каждая строка

Адрес	Код	Метка	Мнемоника	Операнд	Комментарии
1	2	3	4	5	6
0000 0002	3E01 D302	НАЧАЛО: ВВОД:	MVI OUT	A, 01H 02H	00000001 → A Зажечь V1, погасить V2
0004	0170CB		LXI	B, КОНСТ1	Задать время задержки 0,5 с
0007	CD1E00		CALL	ВРЕМЯ	Вызов подпрограммы задержки
000A 000C	DB01 E601		IN ANI	01H 01H	Ввод байта из порта 1 Маскирование неиспользуемых разрядов
000E	C20A00		JNZ	ВВОД	Повторить ввод, если кнопка не нажата
0011 0013	3E02 D302		MVI OUT	A,02H 02H	000000010 → A Зажечь V2, погасить V1
0015	01B865		LXT	B,КОНСТ2	Задать время задержки 0,25 с
0018	CD1E00		CALL	ВРЕМЯ	Вызов подпрограммы задержки
001B	C30000		JMP	НАЧАЛО	Возврат на «НАЧАЛО:»
001E	OB		DCX	B	Подпрограмма задержки
001F	78	ВРЕМЯ: КОНСТ1 КОНСТ2	MOV	A,B	Уменьшить на 1 содержимое BC
0020 0021	B1 C21E00		ORA JNZ	C ВРЕМЯ	Переслать в A содержимое B и C равны нулю?
0024	C9		RET		Если нет, то повторить цикл
			EQU	52080D	Возврат в основную программу
			FQU	26040D	Присвоение числовых значений символическим операндам

бланка разделена на шесть полей. При написании программы первоначально используют поля 4—6. В поле 4 записывают мнемоническое обозначение выполняемой команды, а в 5 — необходимый операнд (адрес перехода, номер порта, наименование регистра и т. д.). Операнд может быть задан непосредственно в виде конкретного числа или неявно, т. е. ему может быть присвоено символическое имя, а конкретное значение, соответствующее этому имени, определяется в дальнейшем при переводе текста программы в машинные коды. Когда операнд задается непосредственно, то после него проставляется буква D, H или B, если число представлено в десятичной, шестнадцатиричной или двоичной форме соответственно. Поле 6 для комментариев (пояснения действий, выполняемых программой). Наличие этого поля необязательно, но если вы хотите, чтобы программа была понятна другим, комментарии необходимы.

Поле 3 заполняют в тех случаях, когда необходимо отметить начало фрагментов программы, к которым осуществляется переход из других частей программы по команде передачи управления и вызова подпрограмм.

Поля 1 и 2 можно заполнить с помощью таблицы кодов команд микропроцессора КР580ИК80А (см. табл. 1.1). Эти поля всегда заполняются числами в шестнадцатиричной форме, и буква H в конце числа при этом опускается. Если мы при составлении программы задаем операнды неявно, то в начале или в конце текста программы определяем, чему же эти операнды равны на самом деле. Для этого в поле 3 записываем символическое имя операнда, в поле 4 — сокращенное английское слово EQU (равно), а в поле 5 — действительное значение операнда. В поле 2 записываем коды и соответствующие операнды команд. При этом символические имена заменяются их действительными значениями. При записи 16-разрядных (двухбайтовых) операндов и адресов в поле 5 мы придерживаемся естественной формы, т. е. слева записываем два старших, а правее два младших разряда числа. При записи кодов операндов в поле 2 — порядок обратный, что объясняется порядком записи байтов 16-разрядных чисел в память микроЭВМ.

Первое значение адреса в поле 1 (адрес первой команды программы) выбирается программистом в зависимости от свободного места в памяти микроЭВМ. Поле заполняют так, чтобы значения, указанные здесь, соответствовали адресам ячеек памяти, содержащих коды команд.

При рассмотрении системы команд Вы узнали о том, что команды бывают одно-, двух- и трехбайтовые. Естественно, это нашло отражение и в формате поля 2: оно может содержать 1, 2 или 3 байта. Поэтому при расчете очередного адреса в поле 1 в зависимости от длины предшествующей команды к предыдущему значению прибавляется 1, 2 или 3 (следует учитывать, что здесь используется шестнадцатиричная арифметика: например, $8 + 3 = B$, $B + 2 = D$, $F + 3 = 12$ и т. д.). Коды второго и третьего байтов в командах передачи управления и вызова подпрограммы в поле 2 заносим в таблицу в последнюю очередь, после заполнения всех строк поля 1. При этом в поле 2 получаем запись программы в машинных кодах. Если теперь коды из поля 2 поместить в память микроЭВМ по адресам, указанным в поле 1, то при пуске микроЭВМ с начального адреса (в данном случае 0000) она начнет выполнять наш алгоритм.

Рассмотрим, как в микроЭВМ создают временные задержки. В данном случае для этого используется подпрограмма ВРЕМЯ. В виде подпрограмм, как пра-

вило, оформляют часто повторяемые однотипные действия, встречающиеся в различных частях реализуемого алгоритма. Такой подход позволяет значительно сэкономить требуемый объем памяти и сделать основную программу более понятной. Вместо того чтобы неоднократно писать в разных местах программы одинаковые последовательности команд, программист использует там только команды вызова подпрограмм, а нужную последовательность команд оформляет однократно в виде подпрограммы.

Обратиться к подпрограмме просто: в основной программе для этого нужно подготовить исходные данные, поместив их в определенные регистры микропроцессора или ячейки памяти, и затем записать команду вызова подпрограммы.

Подпрограммы можно располагать в любом месте памяти, но обычно их помещают сразу после основной программы. В тексте основной программы нашего примера используется команда вызова подпрограммы `CALL ВРЕМЯ`, которая обладает определенной универсальностью. Изменяя значение всего лишь одного операнда, используемого в ней, можно в широких пределах изменять время ее выполнения и следовательно, на любое заданное время задержать работу основной программы. В рассматриваемом примере при первом вызове подпрограммы используем операнд `КОНСТ1`, а при втором вызове подпрограммы — `КОНСТ2`. При обращении к подпрограмме значения этих операндов заносятся в регистровую пару `BC` микропроцессора по команде `LXI B, КОНСТ1` или `LXI B, КОНСТ2` основной программы. Подпрограмму `ВРЕМЯ` можно сравнить с одновибратором с перестраиваемой длительностью импульса. Действительно, можно светодиод подключить не прямо к выводу порта `I`, а через одновибратор, запускаемый положительным фронтом сигнала на выходе младшего разряда порта `I`. При этом длительность импульса $0,25$ с на выходе одновибратора задается `RC`-цепью. Используемый нами программный одновибратор по стабильности и диапазону перестройки длительности выходного импульса значительно превосходит своего электронного «двойника».

Теперь рассмотрим, как работает подпрограмма `ВРЕМЯ`. По команде `DCX B` содержимое регистровой пары `BC` микропроцессора уменьшается на `1`, а затем в аккумулятор пересылается содержимое регистра `B` и производится операция логического сложения с содержимым регистра `C` этой регистровой пары. Если в регистровой паре `BC` код еще не стал равным `0`, то после выполнения этой команды в аккумуляторе окажется число, также отличное от нуля, и выполнится команда условного перехода `JNZ ВРЕМЯ` к началу подпрограммы, все действия повторяются вновь. При этом программисты говорят, что в программе организован цикл. Выход из него возможен только тогда, когда в результате выполнения команды `DCX B` в регистровой паре `BC` окажутся все нули. Тогда работа подпрограммы закончится выполнением команды `RET` и произойдет возврат к выполнению основной программы. Любая подпрограмма всегда должна оканчиваться командой возврата из подпрограммы.

Временная задержка, обеспечиваемая подпрограммой `ВРЕМЯ`, определяется, во-первых, временем, необходимым для однократного выполнения всех команд этой подпрограммы, и, во-вторых, содержимым регистровой пары `BC`. Последнее и определяет количество программных циклов.

Как же определить число, которое надо поместить в регистровую пару `BC` для задания временной задержки $0,5$ с? Выполнение любой команды микропро-

цессором занимает строго определенное время. Поэтому, зная длительность выполнения каждой команды, можно вычислить общее время однократного выполнения подпрограммы ВРЕМЯ. В табл. 1.3 приведено число машинных тактов, в течение которых выполняются команды микропроцессора (подробно о машинных тактах рассказано в приложении), однако здесь будем считать, что длительность одного такта равна 500 нс.

Таким образом время однократного выполнения программы составляет 9,6 мкс. Следовательно, для задания временной задержки в 0,5 с подпрограмма ВРЕМЯ должна быть выполнена $0,5 / (9,6 \times 10^{-6}) = 55080$ раз. Полученный результат необходимо присвоить операнду КОНСТ2. Для организации задержки в 0,25 с операнду КОНСТ1 необходимо присвоить значение, уменьшенное вдвое. При переводе программы в машинные коды и заполнении поля 2 распечатки программы значения второго и третьего байтов команд загрузки регистровой пары ВС должны быть записаны шестнадцатиричными числами.

При переводе десятичного числа в шестнадцатиричное необходимо разделить исходное число на 16, затем полученное частное вновь разделить на 16. Этот процесс повторяют, пока последнее частное не станет меньше 16. Если теперь его представить в виде шестнадцатиричной цифры, то это и будет цифра старшего разряда искомого числа. Следующими цифрами шестнадцатиричного числа будут остатки операций деления, записываемые в последовательности обратной их получению. Естественно, все остатки должны быть также представлены в виде шестнадцатиричных цифр. Пример перевода значения КОНСТ1—55080 в шестнадцатиричную форму:

$$\begin{array}{r}
 52080 \quad | 16 \\
 \hline
 52080 \quad 3255 \quad | 16 \\
 \hline
 0 \quad 3248 \quad 203 \quad | 16 \\
 \hline
 \quad \quad 7 \quad 192 \quad 12 \\
 \hline
 \quad \quad \quad 11
 \end{array}$$

Учтите, что 12D=CH, 11D=BH, 7D=7H, 0D=0H. Таким образом 52080D=CB70H.

Для перевода шестнадцатиричных чисел в десятичные можно использовать табл. 1.6 В соответствии с ней каждому значению шестнадцатиричной цифры в зависимости от того, какой разряд в числе она занимает, соответствует определенное десятичное число. Сумма всех десятичных чисел, соответствующих цифрам переводимого числа, и будет равна искомому числу. Так, например, шестнадцатиричному числу 65B8 соответствует десятичное

$$24576 + 1280 + 176 + 8 = 26040.$$

В рассмотренном примере есть одна, часто встречающаяся у начинающих программистов ошибка, из-за которой программа может не заработать. В чем же ошибка? Мы «забыли» перед началом работы программы настроить регистр SP указателя стека микропроцессора. Настроить — означает поместить в регистровую пару SP адрес памяти (ОЗУ), не содержащий кодов команд. Стек используется в нашей программе (команды CALL и RET), и поэтому мы должны были позаботиться о содержимом регистра указателя стека. Первая

Таблица 1.6

Байт 2				Байт 1			
полубайт 4		полубайт 3		полубайт 2		полубайт 1	
Число							
шестнадцатиричное	десятичное	шестнадцатиричное	десятичное	шестнадцатиричное	десятичное	шестнадцатиричное	десятичное
1	4 096	1	256	1	16	1	1
2	8 192	2	512	2	32	2	2
3	12 288	3	768	3	48	3	3
4	16 384	4	1 024	4	64	4	4
5	20 480	5	1 280	5	80	5	5
6	24 576	6	1 536	6	96	6	6
7	28 672	7	1 792	7	112	7	7
8	32 768	8	2 048	8	128	8	8
9	36 864	9	2 304	9	144	9	9
A	40 960	A	2 560	A	160	A	10
B	45 056	B	2 816	B	176	B	11
C	49 152	C	3 072	C	192	C	12
D	53 248	D	3 328	D	208	D	13
E	57 344	E	3 584	E	224	E	14
F	61 440	F	3 840	F	240	F	15

команда программы должна была быть командой настройки указателя стека — LXI SP, СТЕК. Для стека отведем три ячейки памяти после команды RET. Читателю предлагается самостоятельно внести изменения в программу. При этом следует обратить особое внимание на то, как происходит адресация при работе со стеком.

Рассмотрим еще один пример программы, обнуляющей область памяти, начиная с ячейки 0100H по 02FFH включительно. Вариант такой программы представлен в табл. 1.7. Так как нам предстоит обнуление последовательности ячеек, то организуем циклическую работу программы. В каждом цикле будем обнулять одну ячейку и затем подготавливать адрес очередной ячейки памяти для ее обнуления в следующем цикле. Для этого в цикле необходимо выполнять команду INX H, увеличивающую каждый раз на 1 содержимое регистров HL.

Работа программы должна прекратиться после обнуления последней ячейки памяти заданной области. В нашем случае это будет ячейка с адресом 02FFH, загружаемым в регистровую пару DE по команде LXI D, 02FFH. В ходе выполнения каждого цикла программы необходимо следить, чтобы постоянно увеличивающееся значение адреса в регистровой паре HL не превысило значения конечного адреса области памяти в регистровой паре DE. Для этого в каждом цикле программы необходимо сравнивать старшие байты адресов текущей и конечной ячеек памяти, т. е. коды в регистрах H и D. Это можно сделать вычитанием первого кода из второго (команды MOV A, H, SUB D и JNZ НАЧАЛО). При равенстве этих кодов проводится аналогичная проверка на равенство значений младших байтов адресов текущей и конечной ячеек обнуляемой области памяти. Достижение такого равенства означает, что в HL уже находится адрес конечной ячейки памяти, поэтому необходимо обнулить эту ячейку (предпоследняя команда программы) и прекратить выполнение программы.

Таблица 1.7

Адрес	Код	Метка	Мнемоника	Операнд	Комментарий
1000	210001	НАЧАЛО:	LXI	H, 0100H	Загрузка адреса начала обнуляемой области памяти
1003	11FF02		LXI	D, 02FFH	Загрузка адреса конца обнуляемой области памяти
1006	3600		MVI	M, 00H	Обнуление ячейки памяти
1008	23		INX	H	Подготовка адреса очередной обнуляемой ячейки
1009	7C		MOV	A, H	Сравнение старших байтов адресов текущей и конечной ячеек памяти
100A	92		SUB	D	Сравнение младших байтов адресов текущей и конечной ячеек памяти
100B	C20610		JNZ	НАЧАЛО	
100E	7B		MOV	A, E	
100F	95		SUB	L	
1010	C20610		JNZ	НАЧАЛО	
1013	3600		MVI	M, 0	Обнуление последней ячейки
1015	76		HLT		Окончание программы

У начинающих программистов обычно вызывает трудность использование команд условной передачи управления, например, при сравнении значений двух байтов. Напомним, что перед командой условной передачи управления всегда располагается команда, воздействующая на соответствующий бит регистра признаков. В предыдущем примере для сравнения двух байтов в качестве такой команды использовалась команда вычитания, а передача управления осуществлялась по команде JNZ ADR, контролирующей состояние бита Z. Вместо команд вычитания можно использовать и другие команды, например команды сравнения CPI D8 или CMP M. Действия, оказываемые этими командами на биты регистра F, зависят от результата операции A—D8 или A—M, но, в отличие от других команд, они не изменяют предшествующего содержимого аккумулятора.

Команды JC ADR и JNC ADR осуществляют передачу управления соответственно в случаях, когда M (или D8) $> A$ и M (или D8) $\leq A$.

Так, например, последовательность команд

CPI 10D

JNC ADR

осуществляет передачу управления, если $A \geq 10D$, а последовательность команд

CPI 10D

JC ADR,

если $A < 10D$.

Предположим, что содержимое аккумулятора A к моменту выполнения этих команд будет равно 10D. Тогда в первом случае будет осуществлена

передача управления на команду, расположенную в ячейке с адресом ADR, а во втором случае передача управления не произойдет и будет выполняться следующая по порядку команда. В табл. 1.8 даны примеры использования команд условной передачи управления, часто встречающиеся на практике. Пользуясь таблицей, помните, что если сравниваются операнды со знаком, то вместо команд JC и JNC следует использовать команды JM и JP соответственно.

Таблица 1.8

Условие	Команда, устанавливающая бит регистра признаков F	Команда передачи управления
Любой бит аккумулятора.-0	ANI D8 (1 в соответств. разряде D8 выбирает бит)	JZ ADR
Любой бит аккумулятора.-1	ANI D8 (1 в соответств. разряде D8 выбирает бит)	JNZ ADR
бит 7 аккумулятора.-0	RAL, RLC или ADD A	JNC ADR
бит 7 аккумулятора.-1	RAL, RLC или ADD	JC ADR
бит 6 аккумулятора.-0	ADD A	JP ADR
бит 6 аккумулятора.-1	ADD A	JM ADR
бит 0 аккумулятора.-0	RAR или RRC	JNC ADR
бит 0 аккумулятора.-1	RAR или RRC	JC ADR
Все биты аккумулятора.-0	ANA A или ORA A	JC ADR
Содержимое аккумулятора. \neq 0	ANA A или ORA A	JNZ ADR
Содержимое аккумулятора. положительно (ст. бит-0)	ANA A или ORA A	JP ADR
Содержимое аккумулятора. отрицательно (ст. бит-1)	ANA A или ORA A	JM ADR
Содержимое аккумулятора.-D8	CPI D8	JZ ADR
Содержимое аккумулятора. \neq D8	CPI D8	JNZ ADR
Содержимое аккумулятора. \geq D8	CPI D8	JNC ADR
Содержимое аккумулятора. $<$ D8	CPI D8	JC ADR
Содержимое аккумулятора.-R	CMP R	JZ ADR
Содержимое аккумулятора. \neq R	CMP R	JNZ ADR
Содержимое аккумулятора. \geq R	CMP R	JNC ADR
Содержимое аккумулятора. $<$ R	CMP R	JC ADR

Следующий пример посвящен программной реализации такого распространенного цифрового элемента, как дешифратор для семисегментного индикатора.

Будем считать, что в микроЭВМ имеется входной порт 0, к линиям D0, D1, D2, D3 которого подключены четыре тумблера, образующие тумблерный регистр (рис. 1.6). Оператор может набирать на нем различные кодовые комбинации. К линии D7, связанной со старшим разрядом порта, подключена кнопка S, на которую необходимо нажать, когда на тумблерном регистре будет набран правильный код. К выходному порту 1 подключается семисегментный индикатор. В табл. 1.9 показано соотношение между кодовыми комбинациями, набираемыми на тумблерном регистре, байтами на выходе порта 1 (семисегментными кодами) и десятичной цифрой, отображаемой на семисегментном индикаторе.

Таблица 1.9

Кодовая комбинация	Семисегментный код	Десятичная цифра	Кодовая комбинация	Семисегментный код	Десятичная цифра
0000	3F	0	0110	7D	6
0001	06	1	0111	07	7
0010	5B	2	1000	7F	8
0011	4F	3	1001	6F	9
0100	66	4	1010	Запрещенные комбинации	—
0101	6D	5	... 1111		

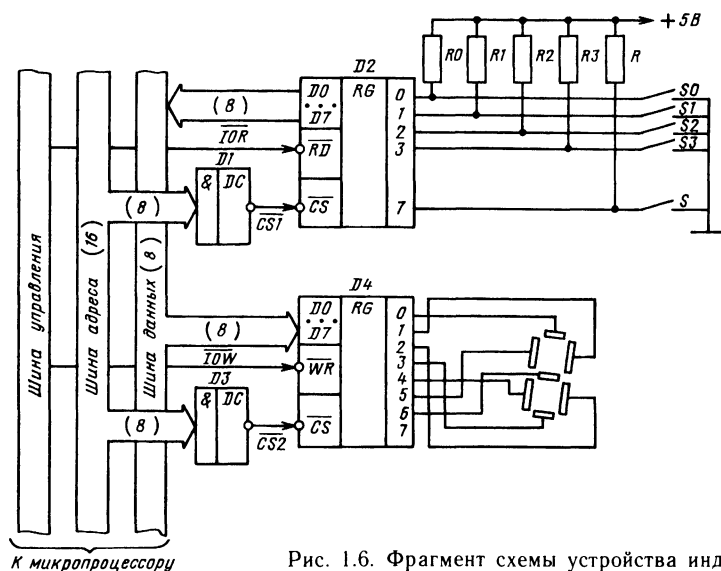


Рис. 1.6. Фрагмент схемы устройства индикации

В табл. 1.10 приведена распечатка программы. Программа должна проверить, не была ли нажата кнопка, считывать информацию с тумблерного регистра и отображать соответствующую десятичную цифру на семисегментном индикаторе. Поставленная задача наиболее просто может быть решена, если мы воспользуемся следующим приемом. Поместим в десять последовательно расположенных вслед за программой ячеек памяти семисегментные коды, приведенные в табл. 1.9. На то, что в ячейках памяти находятся не коды команд, а используемые в программе константы или операнды, указывают символы DB (сокращение английского выражения «определит байт»), помещаемые в поле 4 распечатки. В поле 5, напротив этих символов, заносятся числа, которые должны быть записаны в память до начала выполнения программы. Адресу первой из этих ячеек присвоим метку TABL. Начало области памяти или отдельные ячейки памяти, содержащие различные операнды, используемые в программах, могут быть отмечены метками точно так же, как и команды программы.

Таблица 1.10

Адрес	Код	Метка	Мнемоника	Операнд	Комментарий
0000	DB00	КНОПКА:	IN	0	Опрос состояния кнопки
0002	17		RAL	КНОПКА	Кнопка не нажата Восстановить регистр А
0003	D A0000		JC		
0006	1F		RAR		
0007	E60F		ANA	00001111B	Маскировать биты
0009	FE0A		CPI	10D	Код допустим?
0008	D20000		JNC	КНОПКА	Нет → снова опрос
000E	211C00		LXI	Н, табл	начало таблицы кодов
0011	110000		LXI	D, 0000	Обнулить DE
0014	5F		MOV	E, A	DE-смещению
0015	19	ТАБЛ.:	DAD	D	HL-адресу кода
0016	7E		MOV	A, M	A = коду
0017	D301		OUT	1	Вывод на индикацию
0019	C30000		JMP	КНОПКА	Переход на опрос
001C	3F		DB	00111111B	Код цифры 0
001D	06		DB	00000110B	Код цифры 1
001E	5B		DB	01011011B	Код цифры 2
001F	4F		DB	01001111B	Код цифры 3
0020	66		DB	01100110B	Код цифры 4
0021	6D		DB	01101101B	Код цифры 5
0022	7D		DB	01111101B	Код цифры 6
0023	07		DB	00000111B	Код цифры 7
0024	7F		DB	01111111B	Код цифры 8
0025	6F		DB	01101111B	Код цифры 9

Наша программа будет работать следующим образом. Сначала в аккумулятор вводится содержимое порта 0. Для проверки состояния кнопки, т. е. значения старшего разряда введенного байта, с помощью команды RAL осуществляется сдвиг содержимого аккумулятора влево. После этого значение бита С признака F (признака переноса) будет зависеть от того, была ли нажата кнопка S в момент чтения содержимого порта 0. Если кнопка не была нажата, то признак переноса окажется равным 1 и в программе произойдет передача управления вновь на команду ввода содержимого порта 0. При нажатой кнопке, т. е. когда оператор уже набрал очередную цифру и хочет вывести ее на индикацию, управление в программе передается на команду сдвига содержимого аккумулятора вправо (в исходное состояние).

Так как далее значения четырех старших разрядов порта не используются, то они «маскируются» выполнением команды логического умножения содержимого аккумулятора на операнд, в котором в старших четырех разрядах записаны нули, а в младших четырех разрядах — единицы. Затем проводится проверка на допустимость числа, считанного с тумблерного регистра (не превышает ли оно девяти). Обратите внимание, что операнд команды CPI 10D на единицу больше, чем допустимое вводимое число. Если введенный код недопустим, то вновь производится считывание содержимого тумблерного регистра. Каждому допусти-

тому коду ставится в соответствие семисегментный код. Для этого в регистровую пару HL помещается адрес метки ТАБЛ, обнуляется содержимое регистровой пары DE и затем двоичный код из аккумулятора пересылается в регистр E. Если теперь сложить содержимое HL и DE, то HL будет заключать адрес ячейки памяти, в которой хранится соответствующий семисегментный код. Этот код по команде MOV A, M пересылается в порт I, к которому подключен семисегментный индикатор. После этого программа вновь возвращается на считывание содержимого тумблерного регистра.

Описанный прием использования таблицы, хранящейся в памяти, может быть применен при дешифрации и преобразовании кодов.

Рассмотренная программа в компактной форме в виде содержимого области памяти:

0100	DB	00	E6	0F	FE	0A	D2	00	01	21	17	01	11	00	00	5F
0110	19	7E	D3	01	C3	00	01	3F	06	5B	4F	66	6D	7D	07	7F
0120	6F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Каждая строка начинается с 4-разрядного шестнадцатиричного числа — адреса ячейки памяти, в которой записан первый из 16 последовательно расположенных в памяти байтов, представленных далее в строке двухразрядными шестнадцатиричными числами. По этой таблице можно восстановить текст программы, если воспользоваться таблицей кодов команд микропроцессора и при этом точно знать, в каких ячейках записаны собственно команды программы, а в каких — константы или промежуточные данные. Иначе последние будут расшифрованы как команды.

Рассмотренный пример имеет важную особенность — в нем использован программный опрос готовности внешнего устройства — тумблерного регистра, с которого считывалась информация. Однако считывать информацию с него программа могла только при наличии сигнала готовности — нажатой кнопки. Благодаря этому можно быть уверенным, что на семисегментном индикаторе всегда отображается только та цифра, которая необходима. Все промежуточные манипуляции с тумблерами не оказывают влияния на работу программы, и неверные данные не вводятся. Конечно, в данном случае сигнал готовности выдается оператором, но в большинстве устройств ввода-вывода он вырабатывается автоматически после того, как устройство будет готово к обмену с микроЭВМ. Для того чтобы представить как это делается, рассмотрим процесс ввода информации с перфоленты.

На рис. 1.7 изображен отрезок перфоленты. В устройстве считывания перфоленты протягивается между источником света и фотодиодами, расположенными вдоль прямой, перпендикулярной направлению движения (на рисунке она показана пунктиром). Число диодов равно числу дорожек (продольных рядов отверстий) на перфоленте. Обычно имеются восемь информационных дорожек и одна синхродорожка, расположенная несимметрично относительно информационных и имеющая отверстия меньшего диаметра. Несимметричность позволяет правильно (нужной стороной) заправлять перфоленту в считывающее устройство, а меньший диаметр синхроотверстия гарантирует, что во время движения перфоленты фотодиод, расположенный под ним, будет засвечен позже, чем фотодиоды под

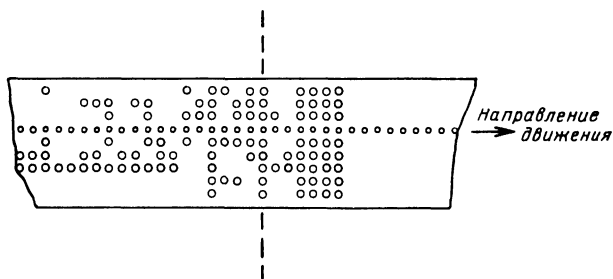


Рис. 1.7. Перфолента

информационными отверстиями, т. е. синхросигнал появится только тогда, когда гарантировано правильно установятся сигналы от других фотодиодов. Если бы синхроотверстие было бы такого же размера, как и информационные, то разброс в диаметрах отверстий, перекося ленты относительно фотодиодов влекли бы за собой ошибочное считывание данных.

Теперь ясно, что синхросигнал можно использовать как сигнал готовности в считывающем устройстве. При этом для подключения считывающего устройства в микроЭВМ необходимо иметь два порта: через один будут передаваться байты данных, а через другой — сигнал готовности. Наряду с сигналом готовности по другим линиям этого порта можно передавать разные служебные сигналы. Например, сигнал от датчика, расположенного в крышке прижима перфоленты к блоку считывания, сигнал о включении электродвигателя протяжки ленты и др. Байт, считываемый через такой порт, называется словом состояния. Анализируя программно этот байт, можно определить, в каком состоянии находится устройство. Следует заметить, что в микроЭВМ одному внешнему устройству соответствуют несколько портов ввода-вывода.

В последнем примере процессор микроЭВМ постоянно опрашивает готовность внешнего устройства и только иногда выполняет другую часть программы. Такая особенность программного опроса готовности внешних устройств совершенно не подходит для микропроцессорных устройств и микроЭВМ, предназначенных для работы в реальном масштабе времени. Однако они часто работают именно в таком режиме. Он характерен тем, что микропроцессор своевременно реагирует на события во внешнем по отношению к микроЭВМ мире, происходящие в различные, неизвестные заранее моменты. Это означает, что микропроцессор должен после обмена данными с внешними устройствами и датчиками обрабатывать их за время, не превышающее заданное. Конечно, в таких случаях у микропроцессора нет времени заниматься программным опросом готовности внешних устройств.

Для работы в реальном масштабе времени у микропроцессора имеется возможность прерывания — временного прекращения выполнения текущей программы и перехода к выполнению подпрограммы обслуживания устройства, вызвавшего прерывание, т. е. непосредственно к обмену информацией с внешним устройством. Обычно устройство, готовое к обмену данными с микроЭВМ, посылает один из сигналов «Запрос прерывания» IR0-IR7 на соответствующий вход специального контроллера прерываний, с выхода которого запрос в виде сигнала INT поступает в микропроцессор. При его появлении микропроцессор должен перейти к

выполнению соответствующей подпрограммы обработки прерывания, т. е. среагировать на внешнее событие. При этом микропроцессор после выполнения очередной команды текущей программы считает не как обычно код операции следующей команды из памяти, а код команды вызова подпрограммы, формируемый на шине данных контроллером прерываний (см. рис. 1.1).

Это происходит потому, что микропроцессор вместо обычно формируемого в начале выполнения команды сигнала «Чтение памяти» MEMR, вырабатывает и передает по шине управления сигнал «Подтверждение прерывания» INTA. Этот сигнал активизирует контроллер прерывания, который обычно формирует на шине данных одну из однобайтовых команд вызова подпрограмм по фиксированным адресам (команды RST0-RST7). Каждому сигналу «Запрос прерывания» от устройств, подключенных к определенным входам контроллера, соответствует «своя» команда RST и, следовательно, определенная подпрограмма обслуживания. В зависимости от номера команды ее выполнение ведет к передаче управления на одну из ячеек в начальной области памяти (см. табл. 1.3). Именно с команды в этой ячейке и должна начинаться подпрограмма обслуживания прерывания.

В простейшем случае, когда внешнее устройство только одно, сигнал «Запрос прерывания» может быть непосредственно заведен на соответствующий вход микропроцессора, а команда RST7 (ее код 11111111) будет сформирована с помощью подключения всех линий шины данных через резисторы сопротивлением в несколько килоом к источнику напряжения +5 В. Тогда микропроцессор в ответ на сигнал «Запрос прерывания» по окончании текущей команды считает с шины данных код команды RST7. Сигнал «Подтверждение прерывания» будет выдаваться на шину управления, но в данном случае использоваться не будет.

В качестве контроллера прерывания в микроЭВМ может быть использована специальная БИС KP580BH59, формирующая на шине данных в ответ на сигнал «Подтверждение прерывания» код операции трехбайтовой команды CALL ADR, а также младшие и старшие байты адресов вызова подпрограмм.

Рассмотрим, как оформляются подпрограммы, обрабатывающие прерывания. Представим себе высококачественный магнитофон, в котором управление его работой, а также автоматическая стабилизация натяжения ленты и индикация числа оборотов производятся микропроцессорным устройством. В режиме воспроизведения устройство должно определять степень натяжения ленты, производить расчет управляющего воздействия, выдачу его на исполнительное устройство и в то же время подсчитывать число импульсов от датчика оборотов и отображать его на соответствующем индикаторе.

Таким образом, основная программа в этом режиме — замкнутый цикл считывания состояния датчика натяжения ленты, расчет управляющего воздействия и выдача его к исполнительному механизму. Так как сигналы от датчика оборотов приходят довольно редко и моменты их появления не связаны с работой основной программы, то целесообразно использовать их в качестве источников запросов прерывания. Предположим, что по каждому запросу прерывания на шине данных контроллер прерываний формирует команду RST7. Следовательно, в соответствии с табл. 1.4 первая команда обслуживания запроса прерывания должна располагаться в ячейке памяти с адресом 0038H. Эта программа начинается следующими командами:

```
PUSH PSW
PUSH B
PUSH D
PUSH H.
```

Четыре первые команды позволяют сохранить в стеке содержимое всех регистров микропроцессора для того, чтобы после возврата к основной программе можно было восстановить их содержимое. Текст подпрограммы здесь не приводится, так как в данном случае нам важны только ее фрагменты, специфичные для обработки прерывания. В конце подпрограммы выполняются следующие команды:

```
POP H
POP D
POP B
POP PSW
EI
RET.
```

С помощью четырех команд считывания из стека восстанавливается содержимое регистров микропроцессора, а затем выполняется команда разрешения прерывания EI. Последнее необходимо, так как после возникновения прерывания в микропроцессоре всегда автоматически запрещается прием запросов прерываний.

Конечно, если вызываемая подпрограмма не использует ни один из регистров какой-либо регистровой пары, то сохранять ее содержимое в стеке не нужно. Например, если подпрограмма не использует ни регистр D, ни регистр E, то команды PUSH D и POP D в программе необязательны. Порядок засылки в стек и извлечения из него содержимого регистровых пар имеет ту особенность, что данные, записываемые в стек последними, считываются из него первыми. Последняя команда RET производит возврат в основную программу — программу стабилизации натяжения ленты. Возврат происходит в то место этой программы и с тем состоянием внутренних регистров, которые были до момента возникновения прерывания.

Вы ознакомились с программными методами организации обмена информацией между внешними устройствами и микроЭВМ. Методы различаются способами проверки готовности внешних устройств, а общим у них является то, что ввод или вывод данных происходит по командам программы под управлением микропроцессора. При обмене информацией между внешними устройствами и памятью связующим звеном служит аккумулятор микропроцессора. Для осуществления такого обмена в программе наряду с командами ввода или вывода приходится использовать и команды пересылки байтов между аккумулятором и ячейками памяти, а также некоторые команды для подготовки адресов тех ячеек памяти, с которыми осуществляется обмен. Таким образом, для пересылки одного байта между портом ввода или вывода и ячейкой памяти, микропроцессором выполняется последовательность из нескольких команд, на что затрачивается определенное время.

Учитывая, что темп обмена информацией между памятью микроЭВМ и внешним устройством часто определяется последним, это время может оказаться

недопустимо большим. Например, при работе с накопителем на гибких магнитных дисках, когда темп обмена информацией определяется скоростью вращения магнитного диска в накопителе и плотностью записи, микропроцессор не успевает программным путем осуществлять пересылку каждого нового байта информации.

В этом и других аналогичных случаях используют метод *прямого доступа к памяти* (ПДП), при котором обмен байтами происходит непосредственно между портами и ячейками памяти. Тогда скорость обмена может лимитироваться только временем обращения к памяти. Для реализации ПДП в состав микроЭВМ вводят специальный контроллер ПДП, например БИС КР580ИК57.

Контроллер передает запрос прямого доступа от внешнего устройства к микропроцессору. Получив такой запрос, последний отключается от шин микроЭВМ с помощью перевода своих шин в высокоимпедансное состояние, и далее управление обменом выполняет контроллер ПДП. При этом он выдает адреса ячеек памяти и сигналы управления для памяти и портов на шины адреса и управления микроЭВМ. Под их воздействием через шину данных осуществляется непосредственный обмен байтами между портом и адресуемыми ячейками памяти. Адреса ячеек памяти и объем передаваемой информации предварительно заносятся в контроллер ПДП микропроцессором.

Метод ПДП используется также и при формировании видеосигнала для вывода информации на экран телевизора.

2. ПЕРСОНАЛЬНАЯ РАДИОЛЮБИТЕЛЬСКАЯ МИКРОЭВМ «РАДИО-86РК»

Структурная схема персональной радилюбительской микроЭВМ «Радио-86РК»¹ (далее для краткости будем называть РК) изображена на рис. 2.1. Основой РК является микропроцессор КР580ИК80А. Для синхронизации работы микропроцессора и всех остальных узлов использован тактовый генератор на микросхеме КР580ГФ24. Память образована ПЗУ объемом 2 Кбайт (микросхема К573РФ5) и ОЗУ объемом 16 или 32 Кбайт (соответственно на 8 или 16 микросхемах К565РУ3А). В ПЗУ хранится управляющая программа — Монитор, а ОЗУ служит для хранения кодов символов, отображаемых на экране дисплея, программ пользователя и данных. Информацию вводят в РК с бытового кассетного магнитофона и клавиатуры, результаты работы отображаются на экране телевизора и могут быть сохранены на магнитной ленте.

Клавиатуру и магнитофон подключают к РК через *программируемый периферийный адаптер* (ППА) КР580ИК55. Через дополнительный ППА могут быть подключены различные радилюбительские конструкции с цифровым управлением режимами работы, например блок RTTY [9], устройства бытового радиокомплекса, различные датчики, исполнительные узлы и т. п.

Видеосигнал формируется контроллером дисплея, собранным на БИС КР580ВГ75. Содержимое области ОЗУ, в которой хранятся коды отображаемых

¹ Принципиальная электрическая схема разработана Ю. В. Озеровым (см. Радио, 1986, № 4—6).

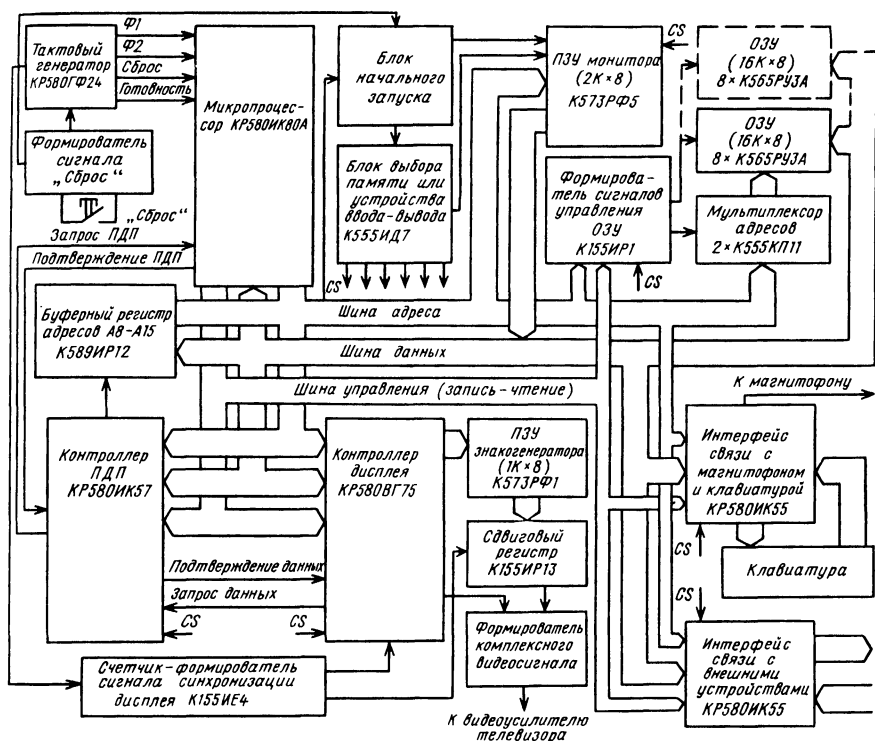
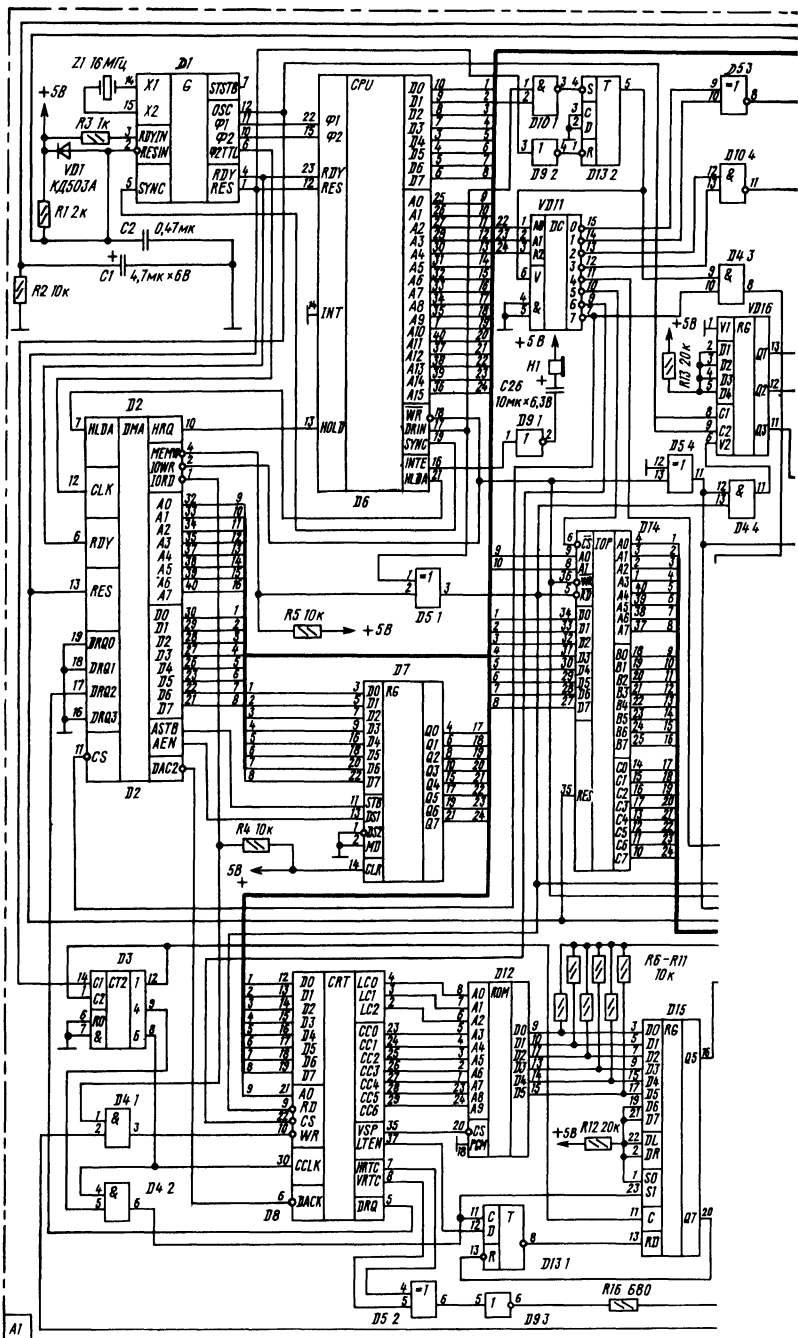


Рис. 2.1. Структурная схема радилюбительского персонального компьютера «Радио-86РК»

символов, передается во внутренние регистры контроллера методом ПДП. Для управления процессом ПДП использована БИС КР580ИК57. Одновременно с формированием видеосигнала в процессе ПДП проводится регенерация содержимого ОЗУ.

Чтобы лучше понять принципы работы РК, необходимо подробно ознакомиться с устройством микропроцессора КР580ИК80А и программируемых БИС КР580ИК55, КР580ВГ75, КР580ИК57. Программируемые БИС могут работать в одном из нескольких режимов, выбираемых загрузкой в их внутренние регистры по командам программы определенных управляющих слов.

Обычно в конкретной аппаратуре используются только некоторые из возможных режимов работы таких БИС. В этом отношении рассматриваемый компьютер не представляет исключения. Однако в описании программируемых БИС будут даны все возможные режимы их работы (см. приложение).



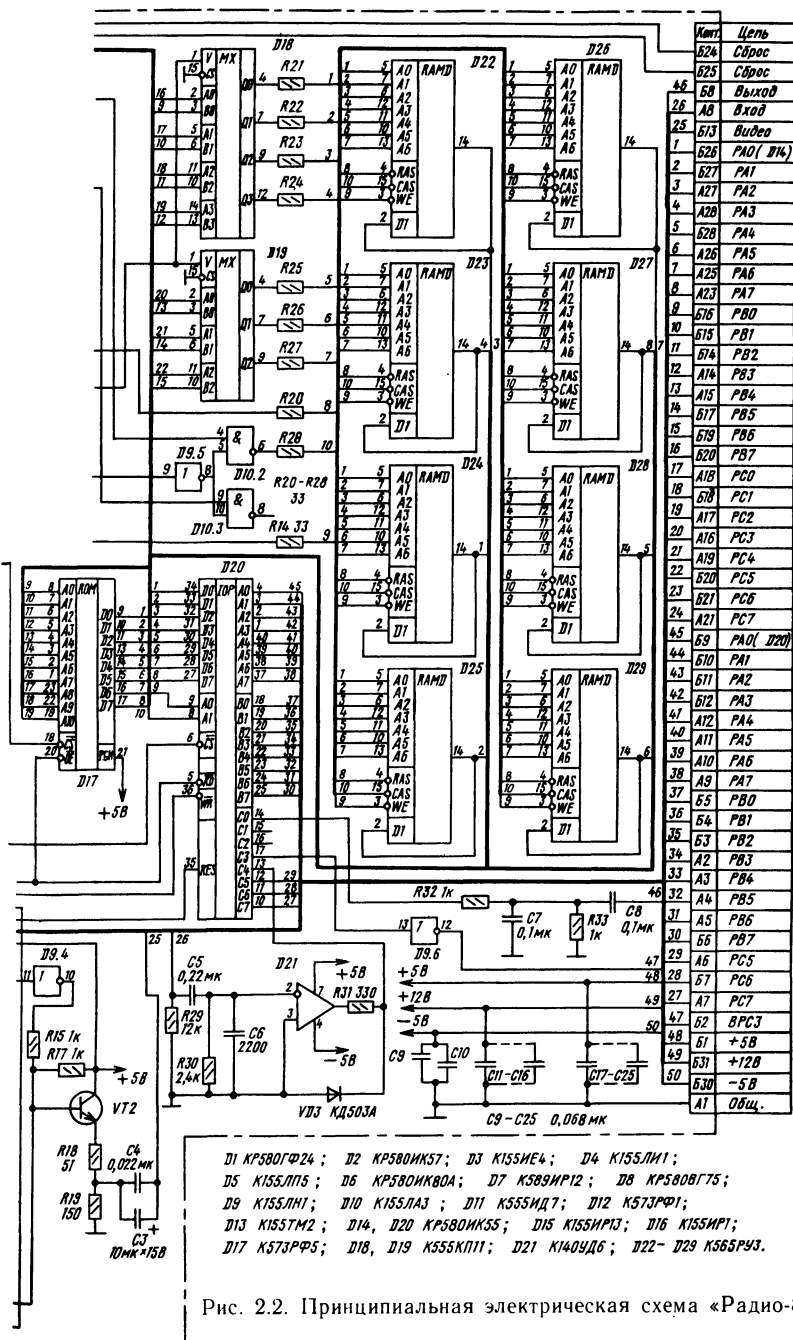


Рис. 2.2. Принципиальная электрическая схема «Радио-86РК»

2.1. Принципиальная электрическая схема РК

Центральный процессор микрокомпьютера

Принципиальная электрическая схема РК представлена на рис. 2.2. Тактовый генератор РК выполнен на микросхеме КР580ГФ24 (D1), предназначенной для работы с микропроцессором КР580ИК80А (D6). Частота тактовых импульсов генератора определяется кварцевым резонатором, подключенным к его выводам X1 и X2. Резонансная частота кварцевого резонатора должна быть в 9 раз больше выбранной тактовой частоты микропроцессора. Так как в РК тактовый генератор служит также и для синхронизации работы контроллера дисплея, в качестве которого использован обыкновенный телевизор, частота кварцевого резонатора выбрана равной 16 МГц. При этом тактовая частота микропроцессора $16/9 = 1,78$ МГц (ниже номинальной, которая равна 2,5 МГц). Именно при такой частоте удастся обеспечить параметры сигналов кадровой и строчной синхронизации в РК, соответствующие телевизионному стандарту.

Необходимые для синхронизации микропроцессора импульсы на выводах Ф1 и Ф2 имеют амплитуду 12 В. На остальных выводах микросхемы формируются сигналы с уровнем ТТЛ. Синхронизация работы периферийных БИС осуществляется последовательностью импульсов, получаемых на выводе Ф2ТТЛ. Последовательность импульсов с частотой кварцевого резонатора и скважностью около двух, формируемая на выводе OSC, используется для синхронизации контроллера дисплея и формирователя сигналов управления БИС динамической памяти. Выводы RDYIN и RESIN предназначены для подачи сигналов «Готовность» и «Сброс» в произвольные моменты. Эти сигналы запоминаются во внутренних триггерах микросхемы КР580ГФ24 и передаются на выводы RDY и RES по фронту импульса Ф2. Для устранения дребезга входной сигнал RESET формируется триггером Шмитта, находящимся в тактовом генераторе. В РК на вход REST N сигнал приходит от устройства формирования и ограничения длительности одноименного сигнала, собранного на элементах C1, C2, R1 — R3, VD1 и кнопке «Сброс».

Так как в РК нет медленно работающих устройств, которые при взаимодействии с микропроцессором требовали бы перевода его в состояние ожидания, на вывод RDYIN постоянно подано напряжение высокого уровня.

На выводе STSTB в момент действия сигнала Ф1 формируется импульс, стробирующий байт состояния микропроцессора. Обычно этот сигнал подают на одноименный вход системного контроллера. В описываемом компьютере этот сигнал не используется, однако он необходим для организации поциклового режима при отладке РК.

Рассмотрим формирование сигналов «Запись» и «Чтение» шины управления. Первый из них формируется на выводе \overline{WR} микропроцессора D6. На входы микросхем D2, D14, D17, D20 этот сигнал поступает непосредственно, на входы WE микросхем памяти D22 — D29 и формирователь сигнала управления ОЗУ подается через повторитель D5.4, а на вывод \overline{WR} контроллера дисплея D8 — еще и через элемент D4.1.

Сигнал «Чтение» формируется не только микропроцессором D6, но и контрол-

лером ПДП D2 при передаче кодов символов из экранной области ОЗУ в контроллер дисплея. При этом используются некоторые особенности работы микропроцессора и контроллера ПДП. Сигнал на выводе DBIN микропроцессора активен (напряжение высокого уровня) только при чтении данных, а на выводе MEMW контроллера (напряжение низкого уровня) — в момент считывания байта из экранной области ОЗУ в контроллер дисплея. Из этих сигналов элементом D5.1 и формируется сигнал «Чтение». Резистор R5 служит для формирования напряжения высокого уровня на выводе 2 элемента D5.1 в то время, когда выход MEMW контроллера находится в высокоимпедансном состоянии.

Блок выбора памяти или устройства ввода-вывода

Для упрощения схемы РК было решено использовать шину управления, состоящую только из линий передачи сигналов «Чтение» и «Запись», при этом обращение к портам контроллеров РК происходит так же, как и к ячейкам памяти, т. е. адреса портов и ячеек памяти располагаются в едином адресном пространстве. Поэтому в РК все обращения к внешним устройствам выполняются по командам обращения к памяти, а команды IN N и OUT N использовать нельзя. Максимально допустимый объем непосредственно адресуемой памяти в этом случае, естественно, менее 64 Кбайт.

Дешифратор адреса выполнен на микросхемах D11, D5.3, D10.4 и D4.3. В зависимости от состояния линий A13—A15 шины адреса на одном из выходов микросхемы D11 формируется напряжение низкого уровня, позволяющее определить, к какой группе ячеек памяти происходит обращение. Таким образом, все адресное пространство микроЭВМ (64 Кбайт) оказывается разделенным на восемь блоков по 8 Кбайт каждый. На выходах элементов D5.3 и D10.4 при обращении к ячейкам ОЗУ с адресами соответственно 0000H—3FFFH и 4000H—7FFFH формируется напряжение высокого уровня. На рис. 2.3 представлено распределение адресного пространства.

Сигналы с выходов 4—7 дешифратора D11 используются для выбора одной из периферийных БИС: D20, D14, D8 или D2. Следует заметить, что сигнал с выхода 7 использован также и для выбора микросхемы ПЗУ D17, т. е. один и тот же сигнал служит как для выбора БИС контроллера ПДП, так и для ПЗУ. Такое решение оказалось возможным благодаря тому, что из ПЗУ информация только считывается, а в контроллер ПДП ее только записывают при инициализации последнего.

Так как после сброса микропроцессор начинает выполнять программу с команды, расположенной по адресу 0000H, а ПЗУ, хранящему управляющую программу Монитор, отведены адреса, начиная с F800H, в компьютер введен блок запуска. На выходе триггера D13.2 в момент прихода сигнала «Сброс» появляется напряжение низкого уровня, которое запрещает работу дешифратора D11 и через элемент D4.3 поступает на вход \overline{CS} микросхемы ПЗУ D17, что и обеспечивает чтение первой команды из ПЗУ — команды безусловного перехода на начало Монитора. После выполнения этой команды на шине адресов появляется код адреса следующей команды, старший разряд которого равен 1. Появление высокого уровня на линии A15 переводит триггер D13.2 в исходное состояние, в результате чего в дальнейшем дешифрация адресов происходит обычным образом.

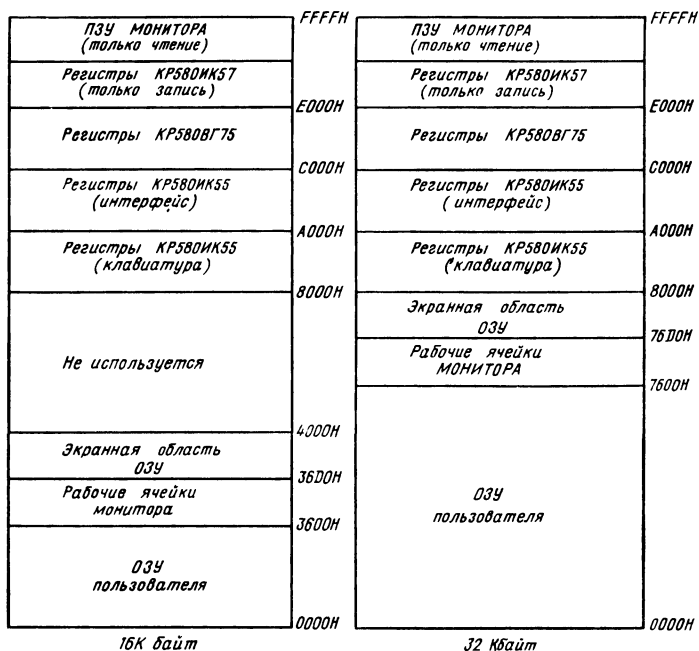


Рис. 2.3. Распределение адресного пространства микроЭВМ

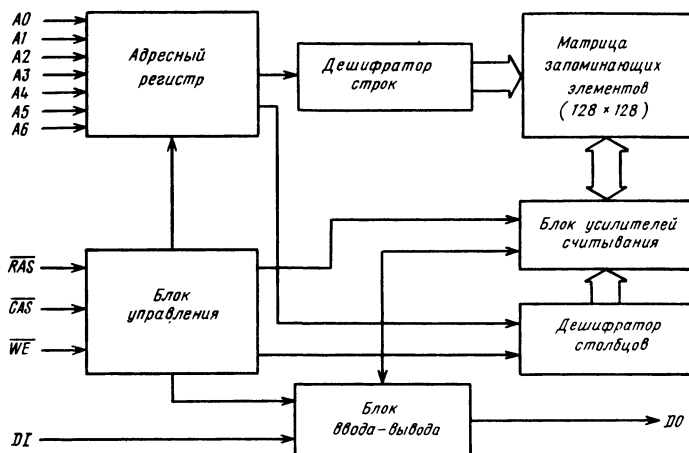


Рис. 2.4. Структурная схема БИС K565PY3A

Оперативное запоминающее устройство

Оперативное запоминающее устройство РК выполнено на микросхемах памяти динамического типа К565РУЗ (D22—D29). Эти микросхемы выполнены по *n*-МОП-технологии и имеют информационную емкость 16 384 бит с организацией 16384×1 разряд. Входные и выходные сигналы микросхемы совместимы по уровням напряжения с ТТЛ-микросхемами. Структурная схема БИС К565РУЗА показана на рис. 2.4.

Основой микросхемы является матрица запоминающих элементов, способных хранить информацию в виде заряда. Для доступа к какому-либо запоминающему элементу матрицы необходимо выбрать соответствующую строку и столбец. Выбор происходит по сигналам дешифраторов строк и столбцов, которые подключены к семи младшим и семи старшим разрядам адресного регистра микросхемы.

Микросхема К565РУЗА имеет всего 16 выводов — один вывод общий, три вывода — для подключения напряжений питания, два информационных: D1 и D0 — для ввода и вывода бита данных, и вывод \overline{WE} — для управляющего сигнала записи бита данных в ячейку памяти. Оставшихся выводов не хватает для передачи на адресный регистр 14-разрядного кода адреса для выбора ячейки памяти (именно $2^{14} = 16\,384$ бит). Поэтому код адреса заносится в адресный регистр последовательно — сначала через адресные входы A0—A6 микросхемы поступают коды семи младших, а затем — семи старших разрядов адреса, сопровождаемые сигналами \overline{RAS} (сигнал выборки строки) и \overline{CAS} (сигнал выборки столбца) соответственно.

Такой режим передачи кода называется мультиплексированным по времени. Его часто применяют в БИС из-за несоответствия количества выводов у корпусов БИС и количества сигналов, которые необходимо обработать. Например, шина данных в микропроцессоре также используется в мультиплексном режиме.

Считывание бита из ячейки БИС ОЗУ происходит в момент действия сигнала \overline{CAS} , если предварительно уже был установлен сигнал \overline{RAS} . На время действия сигнала \overline{CAS} информационный выход D0 микросхемы переходит из высокоимпедансного состояния в режим выдачи сигнала нулевого или единичного уровня, в зависимости от значения хранимого бита в ячейке, адресуемой содержимым адресного регистра.

Если одновременно с сигналом \overline{CAS} при предварительно установленном сигнале \overline{RAS} действует сигнал \overline{WE} , то бит данных с входа D1 будет записан в ячейку памяти. При этом выход D0 микросхемы остается в высокоимпедансном состоянии в течение всего цикла записи.

Обращение к матрице запоминающих элементов для записи или чтения бита данных вызывает подключение к усилителям считывания одной строки матрицы запоминающих элементов, содержащей 128 ячеек памяти. При этом автоматически происходит подзаряд запоминающих конденсаторов всех ячеек памяти выбранной строки до исходного уровня. Этот процесс называется *процессом регенерации памяти*.

Для предотвращения разряда запоминающих конденсаторов ячеек памяти необходимо обращаться к каждой строке матрицы запоминающих элементов не реже чем через 2 мс. При выполнении микропроцессором реальной программы

это условие не соблюдается, так как обращение к одним ячейкам происходит часто, а к другим очень редко. Поэтому необходимо специальное устройство, ответственное за регенерацию памяти. Оно должно (естественно только в те моменты, когда в БИС ОЗУ нет обращений со стороны микропроцессора) циклически формировать на входах A0—A6 значения всех адресов от 00H до 7FH, сопровождая каждое из них одним управляющим сигналом RAS (т. е. формировать адреса строк матрицы запоминающих элементов) с периодом не более 2 мс.

Так как в РК процесс передачи данных из экранной области ОЗУ в контроллер дисплея регулярен, а период обращения к ОЗУ не превышает 2 мс, то циклы ПДП оказались возможным использовать для регенерации содержимого ОЗУ и отказаться от применения специального контроллера регенерации памяти. Однако при этом необходимо ограничить длительность сигнала «Сброс». Если длительность импульса на выводе RESET не будет превышать 1...1,5 мс, то микропроцессор после действия этого сигнала, начав выполнять управляющую программу Монитор, успеет вновь настроить контроллер дисплея и ПДП, возобновив тем самым процесс регенерации памяти. В противном случае информация, находящаяся в ОЗУ, будет потеряна. Адреса в РК мультиплексируются микросхемами D18 и D19, на входы которых с адресной шины поступают разряды A0—A13 кода адреса. В зависимости от уровня сигнала на входах V этих мультиплексоров, на входы A0—A6 микросхем D22—D29 поступают сигналы либо с линий A0—A6, либо с линий A7—A13 шины адреса.

Для формирования сигналов, управляющих работой динамической памяти, служит узел, собранный на микросхеме D16 и элементах D4.4, D9.5, D10.2 и D10.3. На входы C1 и C2 сдвигающего регистра D16 поступает сигнал OSC тактового генератора. При отсутствии на шине управления сигналов «Чтение» и «Запись» на вход V2 регистра D16 с элемента D4.4 поступает напряжение высокого уровня, обуславливающее его работу в режиме параллельного приема данных.

При необходимости считывания из ОЗУ или записи в него на выходе элемента D4.4 формируется низкий уровень сигнала, в результате чего регистр переходит в режим сдвига информации и на его выходах поочередно с задержкой 62,5 нс устанавливаются низкие уровни напряжений. Сигналы Q1 и Q2 поступают соответственно на входы RAS микросхем памяти и входы V мультиплексоров D18, D19. Такой же уровень напряжения при наличии сигнала выбора ОЗУ поступает через элемент D10.2 с выхода Q3 регистра D16 и на входы \overline{CAS} микросхем памяти.

Резисторы R14, R20—R28 служат для улучшения формы сигналов, подаваемых на адресные входы микросхем ОЗУ.

Как уже отмечалось, объем ОЗУ РК может быть увеличен до 32 Кбайт. Для этого в него вводят восемь дополнительных микросхем памяти, выводы которых соединяют с одноименными выводами микросхем D22—D29. Исключение составляют входы \overline{CAS} : их объединяют между собой и подключают к выходу элемента D10.3, назначение которого аналогично назначению элемента D10.2. Наиболее простое конструктивное решение — установить дополнительные микросхемы ОЗУ на уже имеющиеся на печатной плате и припаять их выводы к выводам последних.

Организация в РК режима ПДП

Как уже отмечалось, пересылка данных из ОЗУ в контроллер дисплея осуществляется методом прямого доступа к памяти контроллером КР580ИК57 (D2). Упрощенная шина управления потребовала нестандартного включения БИС контроллера КР580ИК57.

Взаимодействие контроллера дисплея, ПДП и микропроцессора происходит следующим образом. Для вывода очередной строки на экран первый из них формирует сигнал «Запрос данных» на выводе DRQ. По этому сигналу контроллер КР580ИК57 подготавливает микропроцессор к работе в режиме ПДП, выдавая сигнал «Запрос ПДП» на вход HOLD микропроцессора, который в ответ на это переводит свои шины в высокоимпедансное состояние и уведомляет контроллер сигналом «Подтверждение ПДП» на выводе HLDA. Получив этот сигнал, контроллер ПДП инициирует выдачу данных из ОЗУ на шину данных, т. е. устанавливает на шине адреса коды адресов ячеек экранной области, на шину управления выдает через вывод MEMW сигнал «Чтение» для БИС памяти и формирует на выводе DACK контроллера дисплея сигнал «Подтверждение данных». По этим сигналам и при появлении напряжения высокого уровня на выводе IORD контроллера ПДП (сигнал WR для БИС КР580ВГ75) байт из ОЗУ по шине данных переписывается во внутренний буфер контроллера дисплея.

Многорегимный буферный регистр D7 работает совместно с контроллером ПДП D2 и предназначен для временного хранения восьми старших разрядов кода адреса. Это необходимо потому, что в контроллере выходы D0—D7 используются в мультиплексном режиме — как для приема информации с шины данных при его инициализации, так и для выдачи на адресную шину старших разрядов кода адреса в режиме ПДП. В этом режиме на входе DS1 микросхемы D7 от контроллера поступает сигнал высокого уровня, переводящий ее выходы в высокоимпедансного состояния в активное. В первом такте каждого цикла ПДП на входы D0—D7 регистра поступают восемь старших разрядов кода адреса, которые фиксируются сигналом STB и поступают через выходы Q1—Q7 на шину адреса. После этого выходы Q0—Q7 контроллера переводятся в высокоимпедансное состояние, освобождая шину данных для передачи кодов символов из экранной области ОЗУ в контроллер дисплея.

Формирование видеосигнала РК

Рассмотрим теперь, как РК формирует изображение на экране телевизора. Для упрощения узла формирования видеосигнала кадровые и строчные синхронимпульсы формируются непосредственно на выходах HRTC и VRTC контроллера дисплея D8 благодаря его соответствующей настройке. Поскольку изображение на краях экрана телевизионных приемников менее резко и нередко выходит за его границы, оно в этих зонах затемняется программно, т. е. записью в соответствующие ячейки экранной области ОЗУ кодов символа «Пробел», что равносильно формированию в видеосигнале бланкирующих интервалов (гасящих импульсов). На экране алфавитно-цифровая информация отображается 25 строками по 64 знакоместа в каждой. Под каждое знакоместо отводится

матрица точек 6×8 . Строки символов разделены двумя затемненными строками телевизионного раstra.

Таким образом, в одной строке раstra, время отображения которой в соответствии с телевизионным стандартом равно 48 мкс, могут быть засвечены $6 \times 64 = 384$ точки. Следовательно, частота повторения импульсов, подаваемых на вход С сдвигающего регистра D15, должна быть равна 8 МГц. Она получается делением частоты тактового генератора D1 на 2. В качестве делителя частоты использован счетчик D3. Одновременно он формирует импульсы символьной синхронизации, подаваемые на вход CCLK контроллера дисплея D8 ($f_{CCLK} = f_{OSC}/12$). Период следования этих импульсов, равный времени прохождения луча кинескопа в пределах одного знакоместа, и определяет частоту смены кодов символов на выходах CC0—CC6 контроллера.

После того как информация о графическом представлении текущего символа последовательно выдана на выход Q5 сдвигающего регистра D15, последний под действием выходного сигнала элемента D4.2 переходит из режима сдвига в режим приема информации об очередном символе. Этот же сигнал используется для формирования курсора — мигающей черточки, расположенной под отображаемым символом. Такой вид курсора определяется записью соответствующего кодового слова во внутренний регистр контроллера дисплея D8.

При прохождении лучом помеченного знакоместа на выводе LTEN контроллера дисплея периодически появляется высокий логический уровень. Он подготавливает триггер D13.1 к переключению выходным сигналом элемента D4.2, которое происходит в момент начала отображения помеченного знакоместа, что и обеспечивает формирование курсора.

Элемент D4.1 формирует сигнал для записи информации в контроллер дисплея. Низкий уровень напряжения появляется на его выходе при таком же уровне на выходе микропроцессора при инициализации контроллера дисплея или низком уровне напряжения на выводе IORD контроллера ПДП при передаче байта из экранной области ОЗУ. Резистор R4 выполняет те же функции, что и резистор R5.

Формирование комплексного телевизионного видеосигнала осуществляется матрицей резисторов R15—R17. Эмитерный повторитель на транзисторе VT2 служит для согласования формирователя телевизионного сигнала с низкоомной нагрузкой. Через резисторы R15 и R16 поступают сигналы соответственно с выхода сдвигающего регистра D15 и узла формирования синхросмеси, выполненного на элементах D5.2 и D9.3.

Формирование звуковых сигналов

Выход INTE микропроцессора использован в ПК нетрадиционно — как одноканальный порт вывода: командами EI и DI, и подпрограммами временной задержки на нем можно формировать сигналы звуковой частоты. Через элемент D9.1 они поступают на микротелефонный капсюль H1. Таким образом, в ПК имеется возможность программной реализации различных звуковых эффектов.

Возможности используемого в ПК звукоинтегратора ограничены, а качество синтезированного звука невелико. Звукоинтегратор ориентирован прежде всего на «озвучивание» клавиатуры, т. е. на воспроизведение коротких щелчков при

Таблица 2.1

0	1	2	3	4	5	6	7
0	F1	ПРОБЕЛ	0	Q	P	Ю	П
1	F2		1	A	Q	A	Я
2	F3	"	2	B	R	Б	Р
3	F4	#	3	C	S	Ц	С
4		\$	4	D	T	Д	Т
5		%	5	E	U	Е	У
6		&	6	F	V	Ф	Ж
7		'	7	G	W	Г	В
8	<-	->	(8	H	X	Х
9	ТАБ)	9	I	Y	И	Ы
A	ПС	*	:	J	Z	Й	З
B	AP2	+	;	K	[К	Ш
C		,	<	L	\	Л	Э
D	ВК	-	=	M]	М	Щ
E		.	>	N	^	Н	Ч
F	СТР	/	?	O		О	ЗБ

**КОД СИМВОЛА ОБРАЗУЕТСЯ ИЗ НОМЕРА СТРОКИ И НОМЕРА СТОЛБЦА ,
НАПРИМЕР: КОД СИМОЛА "F" = 46**

нажатию на клавиши. Конечно, такой синтезатор способен в соответствии с программой воспроизводить и музыкальные мелодии.

Однако при этом имеются следующие два недостатка. Во-первых, из-за того, что формирование звука осуществляется программным путем, в момент воспроизведения звука микропроцессор не сможет выполнять каких-либо других действий.

Во-вторых, воспроизводимый звук будет несколько искажен из-за того, что процесс его программного формирования будет прерываться циклами прямого доступа к памяти, необходимыми для регенерации изображения на экране.

Возможности формирователя звуковых сигналов в домашних компьютерах можно значительно расширить, применив для этой цели какую-либо специальную микросхему. Однако, отечественной промышленностью не выпускаются периферийные БИС для звуковых синтезаторов. В качестве компромисса можно использовать микросхему КР580ВИ53, выполняющую роль таймера.

Микросхема содержит в своем составе три независимых 16-разрядных счетчика, работающих на вычитание в двоичном или двоично-десятичном коде. При

Таблица 2.2

0000	FF	FF	FF	FF	FF	FF	FF	FF	C7	C7	C7	C7	FF	FF	FF	FF
0010	F8	F8	F8	F8	FF	FF	FF	FF	C0	C0	C0	C0	FF	FF	FF	FF
0020	FF	FF	FF	FF	F8	F8	F8	F8	C7	C7	C7	C7	F8	F8	F8	F8
0030	F8	F8	F8	F8	F8	F8	F8	F8	C0	C0	C0	C0	F8	F8	F8	F8
0040	FF	FF	FF	FF	FF	FF	FF	FF	F3	F3	C0	D2	F3	F3	ED	DE
0050	FF	FF	FF	FF	FF	FF	FF	FF	F3	E1	C0	F3	F3	F3	F3	F3
0060	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0070	F7	F3	D1	C0	C0	D1	F3	F7	F3	F3	F3	F3	F3	FF	EE	CC
0080	FF	FF	FF	FF	C7	C7	C7	C7	C7	C7	C7	C7	C7	C7	C7	C7
0090	F8	F8	F8	F8	C7	C7	C7	C7	C0	C0	C0	C0	C7	C7	C7	C7
00A0	FF	FF	FF	FF	C0	C0	C0	C0	C7	C7	C7	C7	C0	C0	C0	C0
00B0	F8	F8	F8	F8	C0	C0	C0	C0	C0	C0	C0	C0	C0	C0	C0	C0
00C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00D0	FF	FF	FF	FF	FF	FF	FF	FF	F3	F3	F3	F3	F3	F3	F3	F3
00E0	FF	FF	FF	C0	C0	FF	FF	FF	F8	F3	E2	C0	C0	E2	F3	FB
00F0	C7	DF	DF	D8	C2	FA	FA	FA	FA	FF	FF	FF	FF	FF	FF	FF
0100	FF	FF	FF	FF	FF	FF	FF	FF	FB	FB	FB	FB	FB	FF	FB	FF
0110	F5	F5	F5	FF	FF	FF	FF	FF	F5	F5	E0	F5	E0	F5	F5	FF
0120	FB	F0	EB	F1	FA	E1	FB	FF	E7	E6	FD	FB	F7	EC	FC	FF
0130	FB	F5	F5	F3	EA	ED	F2	FF	F9	F9	FD	FB	FF	FF	FF	FF
0140	FD	FB	F7	F7	F7	FB	FD	FF	F7	FB	FD	FD	FD	FB	F7	FF
0150	FF	FB	EA	F1	EA	FB	FF	FF	FF	FB	FB	FB	E0	FB	FB	FF
0160	FF	FF	FF	F3	F3	FB	F7	FF	FF	FF	FF	E0	FF	FF	FF	FF
0170	FF	FF	FF	FF	FF	F3	F3	FF	FF	FE	FD	FB	F7	EF	FF	FF
0180	F1	EE	EC	EA	E6	EE	F1	FF	FB	F3	FB	FB	FB	FB	F1	FF
0190	F1	EE	FE	F9	F7	EF	E0	FF	E0	FE	FD	F9	FE	EE	F1	FF
01A0	FD	F9	F5	ED	E0	FD	FD	FF	E0	EF	E1	FE	FE	EE	F1	FF
01B0	F8	F7	EF	E1	EE	EE	F1	FF	E0	FE	FD	FB	F7	F7	F7	FF
01C0	F1	EE	EE	F1	EE	EE	F1	FF	F1	EE	EE	F0	FE	FD	E3	FF
01D0	FF	F3	F3	FF	FF	F3	F3	FF	F3	F3	FF	F3	F3	F3	F7	FF
01E0	FD	FB	F7	EF	F7	FB	FD	FF	FF	FF	E0	FF	E0	FF	FF	FF
01F0	F7	FB	FD	FE	FD	FB	F7	FF	F1	EE	FE	FD	FB	FB	FB	FF
0200	F1	EE	EC	EA	E8	EF	F1	FF	FB	F5	EE	E0	EE	EE	EE	FF
0210	E1	EE	EE	E1	EE	EE	E1	FF	F1	EE	EF	EF	EF	EE	F1	FF
0220	E1	F6	F6	F6	F6	F6	E1	FF	E0	EF	EF	E1	EF	EF	E0	FF
0230	E0	EF	EF	E1	EF	EF	EF	FF	F1	EE	EF	EF	EC	EE	F0	FF
0240	EE	EE	EE	E0	EE	EE	EE	FF	F1	FB	FB	FB	FB	FB	F1	FF
0250	FE	FE	FE	FE	EE	EE	F1	FF	EE	ED	EB	E7	EB	ED	EE	FF
0260	EF	EF	EF	EF	EE	E0	FF	EE	E4	EA	EA	EA	EE	EE	EE	FF
0270	EE	EE	E6	EA	AC	EE	EE	FF	F1	EE	EE	EE	EE	EE	F1	FF
0280	E1	EE	EE	E1	EF	EF	EF	FF	F1	EE	EE	EE	EA	ED	F2	FF
0290	E1	EE	EE	E1	EB	ED	EE	FF	F1	EE	EF	F1	FE	EE	F1	FF
02A0	E0	FB	FB	FB	FB	FB	FB	FF	EE	EE	EE	EE	EE	EE	F1	FF
02B0	EE	EE	EE	F5	F5	FB	FB	FF	EE	EE	EE	EA	EA	EA	F5	FF
02C0	EE	EE	F5	FB	F5	EE	EE	FF	EE	EE	F5	FB	FB	FB	FB	FF
02D0	E0	FE	FD	F1	F7	EF	E0	FF	F1	F7	F7	F7	F7	F7	F1	FF
02E0	FF	EF	F7	FB	FD	FE	FF	FF	F1	FD	FD	FD	FD	FD	F1	FF
02F0	F1	EE	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	E0	FF
0300	ED	EA	EA	E2	EA	EA	ED	FF	FB	F5	EE	EE	E0	EE	EE	FF
0310	E0	EF	EF	E1	EE	EE	E1	FF	ED	ED	ED	ED	E0	FE	FF	FF
0320	F9	F5	F5	F5	F5	E0	EE	FF	E0	EF	EF	E1	EF	EF	E0	FF
0330	FB	E0	EA	EA	E0	FB	FB	FF	E0	EE	EF	EF	EF	EF	EF	FF
0340	EE	EE	F5	FB	F5	EE	EE	FF	EE	EE	EC	EA	E6	EE	EE	FF
0350	EA	EE	EC	EA	E6	EE	EE	FF	EE	ED	EB	E7	EB	ED	EE	FF
0360	F8	F6	F6	F6	F6	F6	E6	FF	EE	E4	EA	EA	EE	EE	EE	FF
0370	EE	EE	EE	E0	EE	EE	EE	FF	F1	EE	EE	EE	EE	EE	F1	FF
0380	E0	EE	EE	EE	EE	EE	EE	FF	F0	EE	EE	F0	FA	F6	EE	FF
0390	E1	EE	EE	E1	EF	EF	EF	FF	F1	EE	EF	EF	EF	EE	F1	FF
03A0	E0	FB	FB	FB	FB	FB	FB	FF	EE	EE	EF	F5	FB	F7	EF	FF
03B0	EE	EA	EA	F1	EA	EA	EE	FF	E1	EE	EE	E1	EE	EE	E1	FF
03C0	EF	EF	EF	E1	EE	EE	E1	FF	EE	EE	EE	E6	EA	EA	E6	FF
03D0	F1	EE	FE	F9	FE	EE	F1	FF	EE	EA	EA	EA	EA	E0	FF	FF
03E0	F1	EE	FE	F8	FE	EE	F1	FF	EA	EA	EA	EA	EA	E0	FE	FF
03F0	EE	EE	EE	E0	FE	FE	FE	FF	C0	C0	C0	C0	C0	C0	C0	FF

программировании в каждый из счетчиков может быть загружен произвольный код, значение которого уменьшается в процессе работы под воздействием тактовой частоты, поступающей на входы тактирования счетчиков. Когда содержимое счетчиков становится равным нулю, на их выходах генерируются выходные сигналы.

Каждый счетчик имеет вход для приема управляющего сигнала. С помощью этого сигнала можно приостановить и возобновить счет импульсов, поступающих на тактовые входы счетчиков, возобновить работу счетчиков сначала без их программной перезагрузки.

Счетчики могут быть запрограммированы на работу в нескольких различных режимах: как программируемые делители частоты, как программно и аппаратно запускаемые одновибраторы, генерирующие импульсы программно заданной длительности. В частности, в режиме делителя частоты с программно заданным коэффициентом деления, зависящим от значения предварительно загружаемого кода. Наличие такого счетчика в микропроцессорной системе позволяет построить схему, которая параллельно с работой микропроцессора будет непрерывно воспроизводить звуковой тон.

Клавиатура РК

При разработке клавиатуры РК ставилась задача создать максимально простой узел, не критичный к параметрам применяемых коммутационных устройств.

Каким же требованиям должна удовлетворять клавиатура РК? Прежде всего, она должна формировать коды всех символов, приведенных в табл. 2.1 (коды для программирования ПЗУ знакогенератора даны в табл. 2.2).

Кроме того, необходимо предусмотреть защиту от одновременного нажатия на несколько клавиш и дребезга контактов. Желательно также иметь набор функциональных клавиш, предназначенных для перемещения курсора по экрану дисплея, и несколько клавиш, назначение которых программируется пользователем. Для удобства работы должен быть предусмотрен режим автоповтора, т. е. непрерывной выдачи кода символа при длительном (более 1 с) нажатии на клавишу и звуковая индикация в момент замыкания контактов. Выполнение большинства этих требований в РК возложено на подпрограмму обслуживания клавиатуры.

На рис. 2.5 показана принципиальная схема клавиатуры, которая подключается к РК через ППА D20. Основные клавиши, объединенные в блок A2, связаны с матрицей нормально разомкнутых контактов и отдельной группой из трех таких же контактов. Расположение клавиш клавиатуры, принятое в большинстве промышленных дисплеев, показано на рис. 2.6.

Через линии канала A, настроенного на вывод информации в режиме 0, на диоды VD5—VD11 (рис. 2.5) последовательно поступают сканирующие импульсы. Диоды защищают линии порта от повреждения при одновременном нажатии на несколько клавиш.

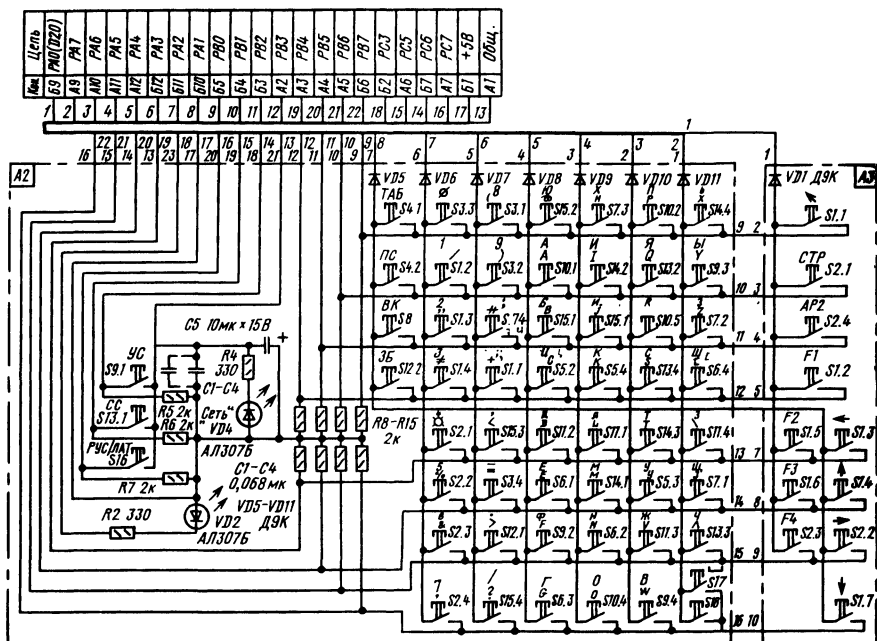


Рис. 2.5. Принципиальная электрическая схема клавиатуры

В процессе опроса контактов клавиатуры подпрограмма обслуживания последовательно формирует напряжение низкого уровня на каждой из линий порта А (на других семи линиях уровни остаются высокими). Сразу после этого подпрограмма считывает и анализирует содержимое порта В. Если ни одна из клавиш не нажата, то на все разряды этого порта через резисторы R8—R15 подано напряжение +5 В. При нажатии на какую-либо клавишу низкий уровень напряжения с соответствующей линии канала А поступает на одну из линий порта В. Подпрограмма обслуживания определяет номер нажатой клавиши и формирует соответствующий ей семиразрядный код.

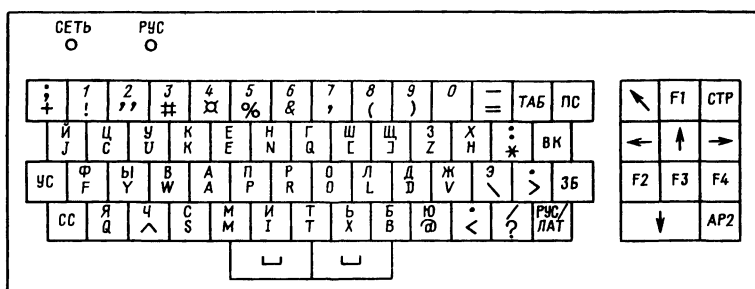


Рис. 2.6. Расположение клавиш клавиатуры

При нажатии на каждую из основных клавиш могут формироваться три различных кода в зависимости от того, была ли нажата вместе с основной (или немного раньше) одна из клавиш модификации кода СС или УС. При этом формируют специальные, управляющие или графические символы. Клавиша «РУС/ЛАТ» определяет, какой из двух алфавитов (русский или латинский) будет отображаться на экране (для перехода с одного на другой достаточно нажать на нее один раз). Замыкание контактов этих трех клавиш приводит к формированию низкого уровня напряжения на линиях С5—С7, работающих в режиме ввода, и иной интерпретации основных клавиш, что позволяет сократить их число. Дребезг контактов устраняется программно.

На рис. 2.7 показана печатная плата, предназначенная для монтажа основных клавиш (минимально необходимый набор). Для удобства работы с РК клавиатуру целесообразно дополнить еще несколькими клавишами (рис. 2.5, блок А3). Печатную плату (рис. 2.8) с этими клавишами устанавливают справа от основной. Дополнительная клавиатура создает удобство в работе, позволяя формировать коды некоторых управляющих символов нажатием только одной клавиши, хотя эти же коды можно получать и с помощью основных клавиш при предварительно нажатой клавише УС.

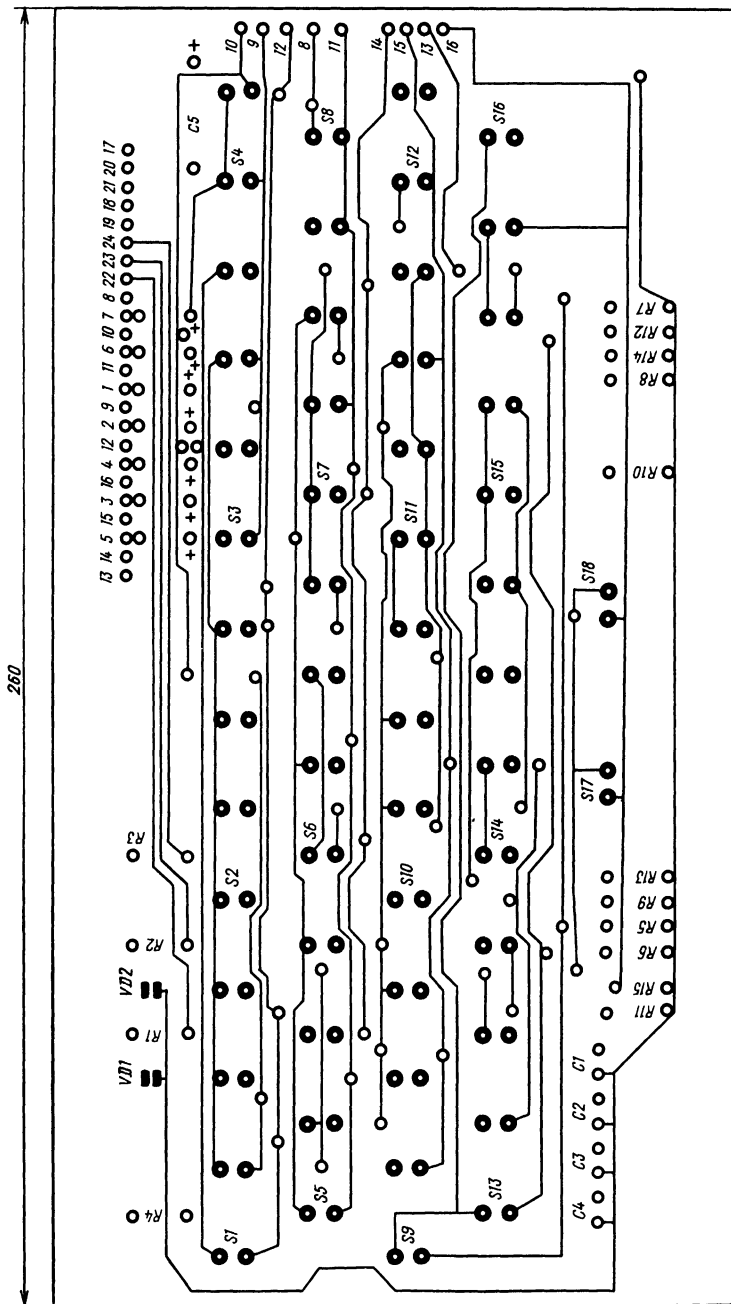
На плате основной клавиатуры установлены светодиоды VD2 и VD4. Первый из них сигнализирует о включении РК, второй (он подключен через элемент D9.6 к линии С3 ППА, настроенной на вывод), — служит для индикации состояния клавиши «РУС/ЛАТ» (как и остальные, она не имеет фиксации при нажатии).

Интерфейсы связи с магнитофоном и дополнительными устройствами

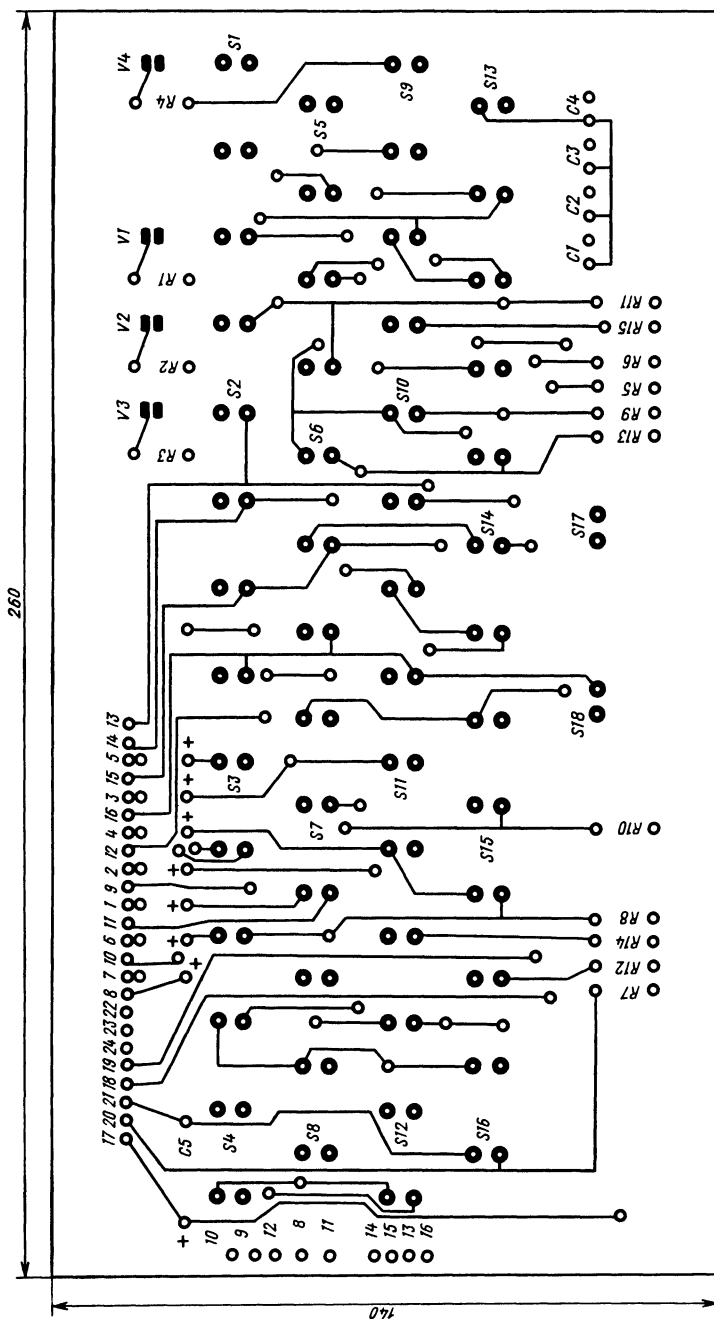
Запись информации на ленту производится последовательно по методу двухфазного кодирования. На рис. 2.9 приведены временные диаграммы, поясняющие этот метод. На диаграмме А показан байт Е6Н (его двоичное представление имеет вид 11100110), преобразованный в последовательную форму. Отдельные разряды байта следуют с периодом $T_{\text{след}}$, причем запись байта начинается со старшего разряда, т. е. сначала должен быть записан разряд D7, затем D6, D5 и т. д. Однако непосредственно записать такой сигнал на магнитную ленту нельзя, так как частотная характеристика магнитофона не соответствует спектру записываемого сигнала. Это происходит потому, что в потоке данных неравномерно чередуются нули и единицы, а следовательно, имеется постоянная составляющая, которая не может быть записана на обычный магнитофон. Для того чтобы записать такой поток данных, обычно применяют один из известных способов модуляции несущей частоты — по амплитуде, частоте или фазе.

Используя метод двухфазного кодирования, можно так преобразовать (закодировать) поток данных, что он не будет содержать постоянной составляющей. Это позволит записывать данные (двухфазные коды) на магнитную ленту непосредственно без предварительной модуляции.

На диаграмме В показан двухфазный код байта данных Е6Н, записываемый на ленту. Этот код формируется следующим образом. Всегда в середине пере-



a)



б)

Рис. 2.7. Печатная плата для монтажа основных клавиш (а, б)

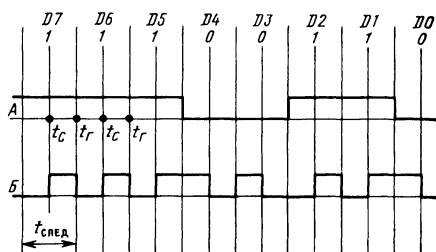
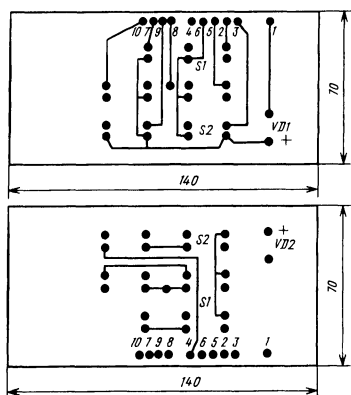


Рис. 2.9. Временные диаграммы сигналов при двухфазном кодировании

← Рис. 2.8. Печатная плата для монтажа дополнительных клавиш

даваемого бита (t_c) происходит изменение его значения на противоположное, причем изменение с 1 на 0 означает, что передан бит, равный 0, а обратное изменение, с 0 на 1 — бит, равный 1.

На границе двух одинаковых по значению смежных битов также всегда происходит изменение значения двухфазного кода. На границе разных по значению смежных битов изменение двухфазного кода не происходит.

Подобным образом должны быть закодированы все биты информации, записываемые на ленту. Период $T_{\text{след}}$ выбран равным 0,9 мс. При этом скорость записи-считывания равна 1100 бит/с. Опыт показал, что при такой скорости можно обеспечить надежное, практически безошибочное считывание информации.

Рассмотрим теперь, каким образом при чтении происходит декодирование двухфазных кодов. Предположим, что считывание данных началось в момент t_0 (рис. 2.10). Подпрограмма чтения позволяет считывать и распознавать информацию примерно один раз в 15 мкс. Начиная с t_0 , подпрограмма считывания производит чтение информации и ее анализ (t_A) до тех пор, пока не произойдет изменение уровня сигнала по сравнению с предыдущим считанным значением. На рисунке эти моменты обозначены t_B . Уровень сигнала, считанный в момент t_B , рассматривается как полезная информация и поэтому запоминается.

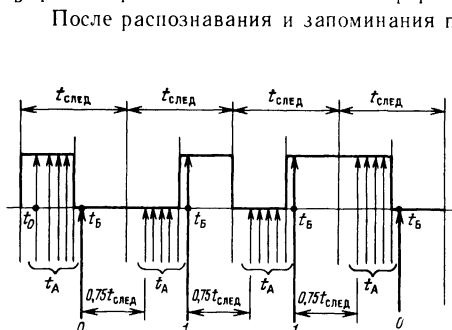


Рис. 2.10. Временные диаграммы сигналов при чтении двухфазного кода

После распознавания и запоминания принятого бита происходит задержка в работе программы, равная 0,75 $T_{\text{след}}$, и процесс считывания информации начинается вновь. В РК описанный алгоритм кодирования реализован программно.

Программное обеспечение при этом выполняет следующие функции:

при записи данных сначала производит преобразование записываемого байта из параллельного в последовательный вид;

затем каждый записываемый бит кодирует в соответствии с методом двух-
фазного кодирования, который был описан выше;

и наконец, формирует соответствующие временные интервалы.

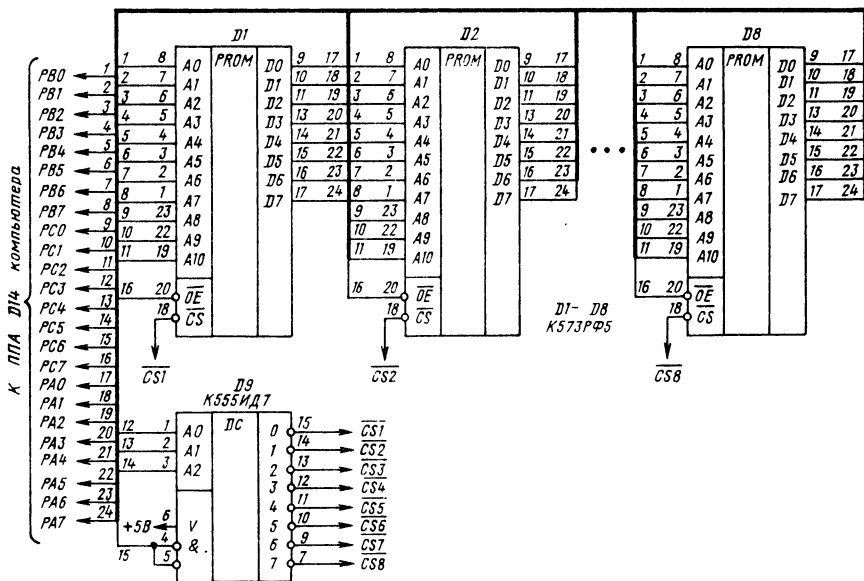
В РК через линии C0 и C5 канала C к ППА D20 подключены узлы форми-
рования сигналов для обмена информацией с бытовым кассетным магнитофоном.

Кроме того, в РК имеется дополнительный ППА D14, не настраиваемый в
программе Монитор. После прихода сигнала «Сброс» все три его канала сво-
бодны и работают на ввод информации. Свободными остаются также линии
C1 и C2 ППА D20, запрограммированные в режим вывода информации.

Использование дополнительного ППА D14

В РК имеется дополнительный ППА D14, который может быть ис-
пользован для подключения различных внешних устройств ввода-вывода. Монит-
ор в начале работы не настраивает ППА, но после прихода сигнала «Сброс»
все три его канала оказываются настроенными на ввод информации в режиме 0.
Кроме того, в РК свободными также остаются линии C1 и C2 ППА D20,
запрограммированные на режим ввода информации.

В качестве примера использования ППА D14 можно привести устройство
организации внешней программно-управляемой шины для подсоединения допол-
нительных периферийных БИС и внешнего ПЗУ с объемом памяти до 32 Кбайт.
Принципиальная схема такого устройства приведена на рис. 2.11. Канал
A ППА D14 служит для ввода данных, каналы B и C - для выдачи адреса
ПЗУ и формирования сигналов чтения. Кроме K573PФ5 (K573PФ2), в устройстве
можно применить микросхемы ПЗУ K573PФ4, K556PT5, K556PT7 (естественно,



с учетом их особенностей и цоколевки). В любом случае следует учесть, что при подключении внешнего ПЗУ ток, потребляемый от источника напряжения питания $+5$ В, возрастает до 1,5...2 А.

Внешнее ПЗУ целесообразно выполнить в виде небольшого самостоятельного блока (кассеты), снабженного соединителем. Ответную часть последнего устанавливают в корпусе компьютера. Очень удобно иметь несколько таких кассет — это позволяет оперативно готовить компьютер для решения тех или иных задач.

В кассетах ПЗУ можно хранить самые различные программы: интерпретатор Бейсика, редактор текста, языка Ассемблера и т. д. Прежде чем начать работу с любой из этих программ, ее необходимо перегрузить из кассеты в ОЗУ компьютера. Делают это с помощью директивы R Монитора.

Блок питания ПК

Микрокомпьютер отличается малым энергопотреблением (0,8 А, 150 и 30 мА от источников напряжения $+5$, $+12$ и -5 В соответственно), что позволяет сделать блок питания простым и компактным. Авторы разместили такой блок в отдельном корпусе, но возможно смонтировать его и в корпусе ПК.

На рис. 2.12 приведена принципиальная схема блока питания. Напряжения $+12$ и -5 В устанавливают соответственно подстроечным резистором R4 и стабилитроном VD14. В качестве сетевого можно использовать стандартный трансформатор ТПП260-127/220-50 или любой другой мощностью 20...30 Вт. Микросхему D1 устанавливают на теплоотводе с суммарной площадью охлаждения не менее 50 см^2 , для охлаждения транзистора VT1 используют простейший пластинчатый теплоотвод площадью около 20 см^2 .

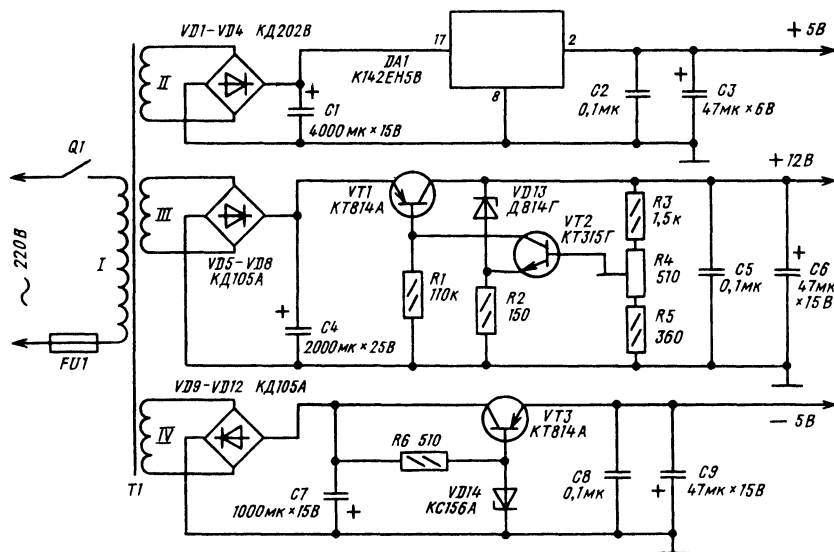


Рис. 2.12. Принципиальная электрическая схема блока питания

При проектировании ПК перед разработчиками встала задача уменьшить нагрузку на линии шины адреса и данных микропроцессора, так как их максимальная нагрузочная способность — один ТТЛ-вход (1,9 мА). С этой целью в ПК использованы микросхемы серий с малым потреблением по входу.

Печатные платы ПК (рис. 2.13) и клавиатуры рассчитаны на установку резисторов МДТ-0,125, конденсаторов КМ-6Б, К53-1, штыревой части соединителя ГРПМ1-61ШУ2 и блоков клавиатуры ВМ16-1, ВМ16-4.

2.2. Отладка ПК

Краткое описание процесса отладки ПК поможет Вам выявить причину той или иной неисправности компьютера как сразу после сборки, так и при его эксплуатации.

Несмотря на простоту устройства, компьютер является сложным объектом для диагностики неисправностей. Это объясняется тесной связью в нем программного обеспечения и аппаратных средств.

При отладке и контроле микропроцессорных устройств в условиях исследовательской лаборатории или производства обычно применяют специальные технические и программные средства, например логические анализаторы и моделирующие программы. В любительской практике приходится искать другие методы и средства решения этой задачи.

Для ПК разработана методика отладки, ориентированная на использование только обычного осциллографа и омметра. Следуя этой методике, Вы сможете убедиться в работоспособности отдельных блоков и компьютера в целом или обнаружить и локализовать имеющиеся неисправности.

Рассмотрим последовательность отладки ПК.

1. Начните отладку ПК с проверки омметром монтажа печатной платы и устраните выявленные дефекты. С особой тщательностью следует проверить наличие электрических связей между выводами источника напряжения питания микросхем (особенно D22—D29) и соответствующими контактами соединителя. Щупами омметра необходимо касаться непосредственно выводов микросхем — это позволит обнаружить некачественные пайки. Отсутствие на микросхемах D6, D12, D22—D29 одного из напряжений питания может привести к выходу их из строя.

Затем необходимо убедиться, что блок питания обеспечивает требуемые напряжения: $+5\text{ В} \pm 10\%$ при токе до 1 А, $+12\text{ В} \pm 10\%$ при токе до 200 мА и $-5\text{ В} \pm 10\%$ при токе до 100 мА.

2. Выньте из панели микросхему D17 и установите в нее ПЗУ с тест-программой. Соедините вывод 3 микросхемы D1 с общим проводом и, подав напряжение питания, убедитесь в наличии напряжений на соответствующих выводах всех микросхем.

3. Проверьте осциллографом наличие и параметры сигналов, вырабатываемых микросхемой D1: на выводах 10 и 11 — импульсов амплитудой 12 В с периодом следования 562 нс, на выводе 6 — амплитудой 5 В с периодом следования 562 нс, на выводе 12 — амплитудой 5 В с периодом следования 62,5 нс. Отсутствие этих сигналов обычно свидетельствует о неисправности микросхемы или кварцевого резонатора.

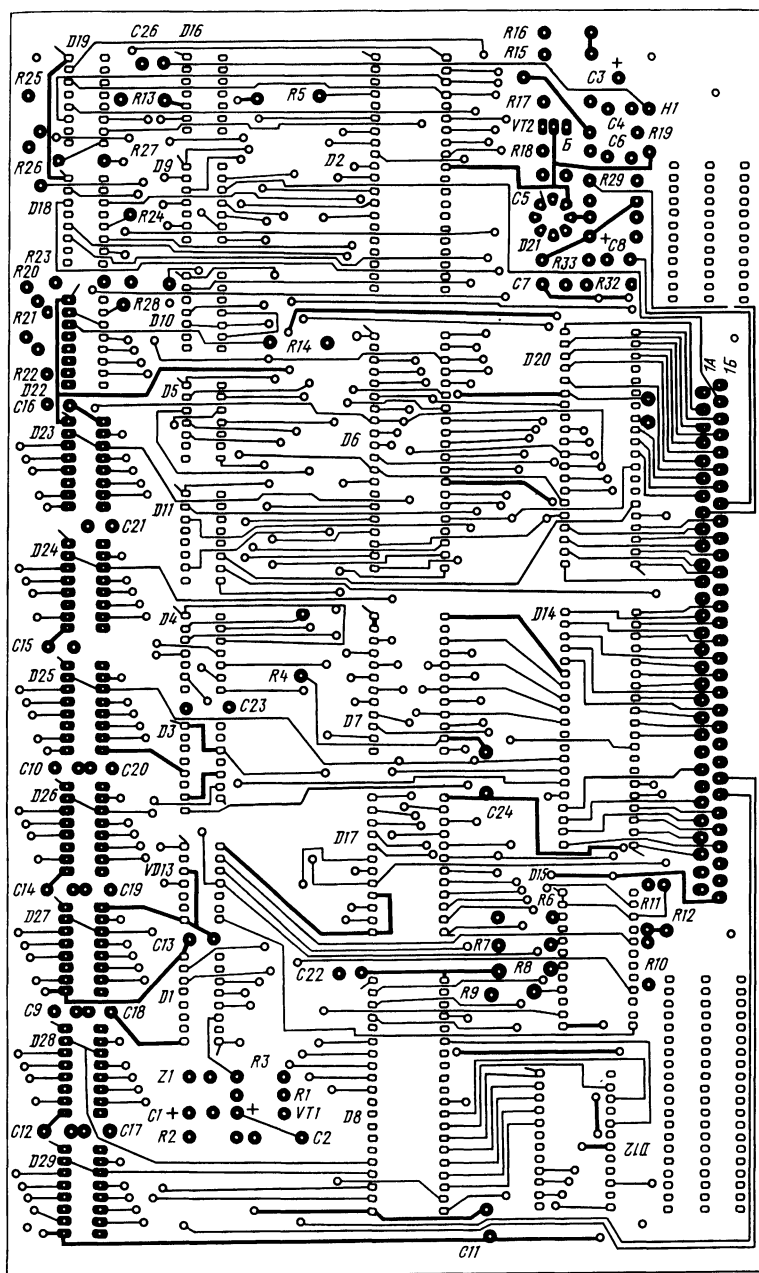


Рис. 2.13. Печатная плата РК

Для выявления причины отсутствия указанных сигналов поступают следующим образом. Отсоединяют вывод OSC микросхемы D1 от монтажной платы и снова контролируют этот сигнал при помощи осциллографа. Появление сигнала OSC свидетельствует о дефектах монтажа или микросхем K155IE4 (D3) или K155IP1 (D16).

Отсутствие сигнала OSC в этом случае однозначно означает неисправность кварцевого резонатора или микросхемы тактового генератора.

Для проверки работоспособности генератора вместо кварцевого резонатора к его выводам X1 X2 подключают керамический конденсатор емкостью 10...15 пФ. Наличие генерации в этом случае говорит о неисправности кварцевого резонатора.

В РК можно использовать кварцевые резонаторы на частоты 15...17 МГц. При этом синхронизация телевизионных приемников остается достаточно устойчивой и надежность считывания данных с магнитофона не снижается. Если найти подходящий кварцевый резонатор не удастся, то его можно заменить подстроечным конденсатором, или, что еще лучше, LC-контуром. Между выводами X1 и X2 подключается цепь, состоящая из последовательно включенных конденсатора емкостью 5,6 пФ и катушки индуктивности. Параллельно катушке индуктивности следует подключить конденсатор емкостью 20 пФ. Катушку наматывают на стандартном каркасе от фильтра ПЧ ЧМ приемника с подстроечником M100HH-CC2, 8X15. Катушка содержит 16 витков провода ПЭВ-1 0,2.

Такая замена не ухудшит работу компьютера. Частоту тактовых импульсов следует установить по частотомеру или по наилучшей синхронизации изображения на экране телевизора.

При отсутствии какого-либо из других сигналов, формируемых тактовым генератором, необходимо отсоединить соответствующий вывод микросхемы D1 от монтажа, и если сигнал при этом появляется, то искать неисправность следует либо в монтаже, либо в микросхемах, на входы которых этот сигнал поступает. Полезно также проверить наличие высокого уровня на выходе RDY микросхемы тактового генератора.

Если при проведении описанных проверок выяснится, что причина дефектов — монтаж, то следует внимательно осмотреть печатную плату и независимо от способа, каким она изготовлена, пропаять хотя бы те переходы с одной стороны на другую, которые находятся под микросхемами.

Большинство применяемых в РК микросхем изготовлено по МОП технологии и боится воздействия статического электричества, поэтому для пайки следует применять низковольтный паяльник с жалом, соединенным с браслетом на руке оператора и желательно заземленным.

Особенно осторожно следует обращаться с микросхемами ОЗУ, в которых из-за несоблюдения правил монтажа могут выйти из строя несколько ячеек памяти, а микросхемы с таким дефектом уже непригодны.

Опыт эксплуатации нескольких РК показал, что тактовый генератор достаточно надежен. Целесообразно установить микросхему D1 на панель. Для дальнейшего расширения возможностей РК на панель следует установить и микросхему D14.

4. Проверьте работу узла формирования сигнала «Сброс». При каждом нажатии на одноименную кнопку на выводе 1 микросхемы D1 должен формироваться импульс амплитудой 5В и длительностью около 1 мс.

5. Убедитесь в наличии высокого уровня напряжения на выводе 24 микропроцессора D6, что свидетельствует о нахождении его в состоянии ожидания, вызванном установкой перемычки в соответствии с п.2. Проверьте состояние шин адреса и данных РК, касаясь последовательно щупом осциллографа соответствующих выводов микросхем. На всех линиях шины адреса должен присутствовать низкий уровень напряжения, а на линиях шины данных — двоичный код 10101010, записанный в нулевой ячейке тест-программы.

При обнаружении несоответствий проверьте, нет ли замыканий между линиями шины адреса и какими-либо другими сигнальными линиями или проводниками источника напряжения питания, убедитесь в работоспособности ПЗУ D17, узла начального сброса (микросхемы D10.1, D9.2, D13.2) и дешифратора D11. Правильной работе дешифратора соответствует низкий уровень напряжения на выводе 18 микросхемы D17. После устранения обнаруженных неисправностей следует вновь провести отладку по пп. 4 и 5.

6. Выключив источник напряжения питания, снимите установленную ранее перемычку, подсоедините узел поциклового выполнения программы микропроцессором и вновь включите РК (рис. 2.14).

При однократном нажатии на кнопку «Шаг» микропроцессор после выполнения такта T2 каждого машинного цикла команды переходит в состояние ожидания, что позволяет контролировать все сигналы в статическом режиме.

7. Подав сигнал «Сброс», проконтролируйте выполнение первых 11 команд тест-программы в поцикловом режиме. При выполнении действий этого пункта ОЗУ РК не используется, что позволяет проводить отладку даже при наличии в нем неисправных микросхем.

При очередном нажатии на кнопку «Шаг» контролируйте состояние шин адреса, данных и управления в соответствии с табл. 2.3. Как было указано, при выборке из ПЗУ первой команды (однobaйтовой) на шине данных появляется код 10101010. В следующей ячейке ПЗУ помещен код 01010101. Эти два кода являются инверсией друг друга, и их использование необходимо для того, чтобы убедиться, что все линии шины данных могут быть переключены в оба состояния. Третья команда (строки 3...5) — команда безусловного перехода по адресу F805H. После трехкратного нажатия на кнопку «Шаг» (эта команда выполняется за три машинных цикла) должен сработать узел начального запуска и на выводе 5 триггера D13.2 появиться высокий уровень напряжения. Следующие четыре команды (6...17) предназначены для настройки БИС ППА D14 и D20. Для вывода результатов работы тест-программы будем использовать ППА D14. После выполнения еще двух команд (18...23) на выводах канала в ППА D14 должна

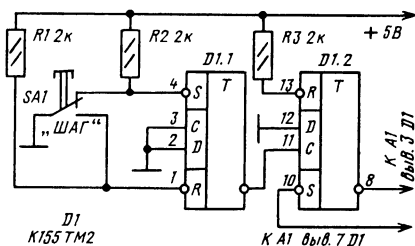


Рис. 2.14. Принципиальная электрическая схема узла поциклового выполнения команд

появиться кодовая комбинация 01010101. Выполнение следующих двух команд (24...30) приводит к зажиганию светодиода «РУС/ЛАТ», подтверждая тем самым работоспособность БИС D20.

Т а б л и ц а 2.3.

	Адрес	Данные	RD	WR	Команда	Примечание
1	0000	AA	0	1		Проверка шины данных
2	0001	55	0	1		
3	0002	C3	0	1	JMP	Переход на начало программы
4	0003	05	0	1	05	
5	0004	F8	0	1	F8	
6	F805	3E	0	1	MVI	Настройка порта
7	F806	8A	0	1	A,8AH	
8	F807	32	0	1	STA	Клавиатуры
9	F808	03	0	1	03	(D20)
10	F809	80	0	1	80	
11	8003	8A	1	0		Запись в порт
12	F80A	3E	0	1	MVI	Настройка дополнительного порта
13	F808	80	0	1	A, 80H	
14	F80C	32	0	1	STA	
15	F80D	03	0	1	03	(D14)
16	F80E	AO	0	1	A0	
17	A003	80	1	0	MVI	Запись в порт
18	F80F	3E	0	1		Выдача тестового сигнала
19	F810	55	0	1	A,55H	
20	F811	32	0	1	STA	в канал
21	FB12	01	0	1	01	в микросхемы
22	F813	A0	0	1	A0	D14
23	A001	55	1	0		Запись в порт
24	F814	3E	0	1	MVI	Зажечь светодиод
25	F815	08	0	1	A,08H	«РУС/ЛАТ»
26	F816	32	0	1	STA	
27	F817	02	0	1	02	
28	F818	80	0	1	80	
29	8002	08	1	0		Запись в порт
30	F819	AF	0	1	XRA A	

Выключите источник напряжения питания и отключите узел цикловой работы. Дальнейшая отладка и проверка функционирования клавиатуры, ОЗУ и дисплейного блока будет проходить при автоматической работе ПК по тест-программе.

8. Включите источник напряжения питания. Нажмите на кнопку «Сброс» и после окончания звукового сигнала — на клавишу «РУС/ЛАТ». Эти действия приводят к запуску программы проверки ОЗУ. Если микросхемы D22—D29 исправны, то по окончании работы тест-программы раздастся звуковой сигнал и загорится светодиод VD2. О наличии неисправных микросхем ПК сообщит двумя звуковыми сигналами (светодиод VD2 в этом случае гореть не должен) и сформирует высокие уровни напряжения на линиях канала В ППА D14, соответствующих разрядам шины данных, к которым подключены неисправные микросхемы ОЗУ.

Причиной неисправности могут быть как дефекты микросхем, так и невер-

ная работа формирователя сигналов RAS и CAS (D16) или мультиплексоров D18, D19.

Сначала убедитесь в работоспособности формирователя. Для этого проверьте наличие и форму сигналов на входах RAS, CAS, WE микросхем памяти и на входах V адресных мультиплексоров D18 и D19 (рис. 2.15). Следует учесть, что сигнал $\overline{\text{CAS}}$ переходит в активное состояние (0) только при наличии сигнала выбора данного блока ОЗУ. Повторно тест-программу запускают нажатием кнопки «Сброс» и клавиши «РУС/ЛАТ».

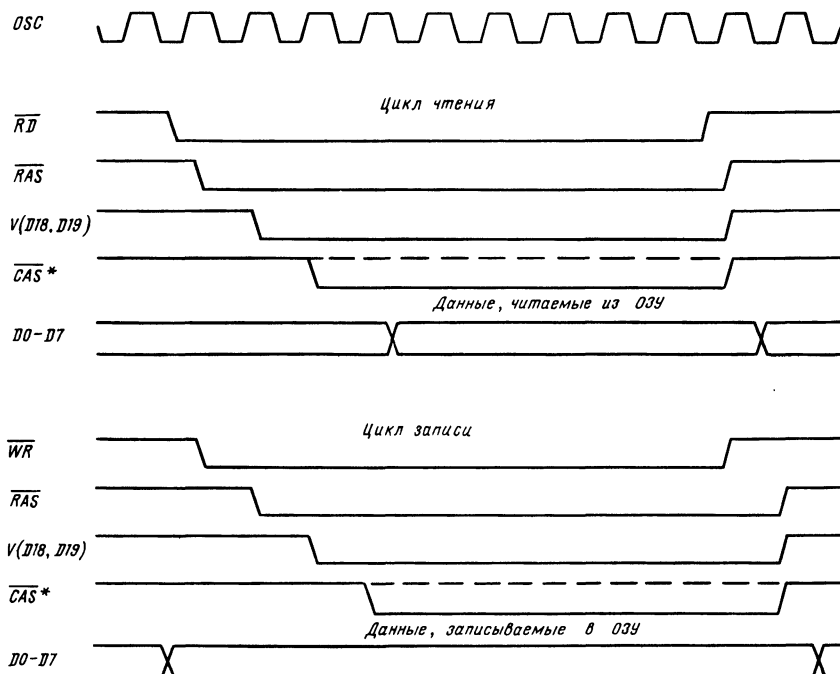


Рис. 2.15. Временная диаграмма сигналов при чтении и записи в ОЗУ

Следует учесть, что тест-программа проверяет только ОЗУ, расположенное по адресам 0000H—3FFFH. Для проверки дополнительного ОЗУ сигнал с выхода D10.2 необходимо подать на соответствующие выводы его микросхем. Проверка остальных узлов РК возможна только при исправном ОЗУ.

9. Убедитесь в работоспособности дисплейного блока и контроллера ПДП, нажав кнопку «Сброс» и (после окончания звукового сигнала) клавишу «УС». По этой команде тест-программа очищает экранную область ОЗУ, инициализирует контроллеры ПДП и дисплея и помещает курсор в левый верхний угол экрана. Если этого не происходит, то необходимо осциллографом проверить способность контроллера дисплея формировать сигнал «Запрос ПДП» на выводе 17 БИС D2 и наличие сигналов «Запрос» и «Подтверждение ПДП» на выходах

13 и 21 микропроцессора D6. Затем следует убедиться, что контроллер дисплея формирует импульсы строчной и кадровой синхронизации с периодами 64 мкс (HRTC) и 20 мс (VRTC) соответственно. Если все эти сигналы вырабатываются, то причиной неисправности могут быть отсутствие сигнала «Запрет отображения» (VSP), переходящего в активное состояние на время действия кадровых и строчных синхроимпульсов и межстрочных интервалов, или дефекты элементов узла формирования видеосигнала.

В исправном дисплейном блоке при каждом нажатии на клавишу «УС» начинается вывод на экран алфавитно-цифровых символов. По правильному отображению этих символов убеждаются в работоспособности ПЗУ знакогенератора и сдвигового регистра.

10. В заключение проверьте работу клавиатуры. Для этого нажмите на кнопку «Сброс» и клавишу «СС», а затем — поочередно на все остальные и убедитесь, что отображаемые символы соответствуют клавишам, на которые вы нажимаете.

3. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПК

3.1. Управляющая программа Монитор

В ПЗУ объемом 2 Кбайт записана управляющая программа Монитор (табл. 3.1), инициализирующая все программируемые БИС и обеспечивающая работу клавиатуры, дисплея и интерфейса с кассетным магнитофоном. Кроме того, Монитор поддерживает диалог с пользователем, который вводит с клавиатуры определенные директивы и на экране дисплея читает сообщения о результате их выполнения. Имеющиеся директивы позволяют просматривать и изменять содержимое памяти, вводить программы вручную или с магнитофона, выполнять записанные в ОЗУ программы или их части, контролируя при этом содержимое внутренних регистров микропроцессора, а также выводить программы и массивы данных на внешний накопитель — магнитную ленту. Важной функцией Монитора является обеспечение работы других программ (интерпретатора Бейсика, редактора текста и др.), для чего в него включен набор стандартных подпрограмм ввода-вывода информации.

Начальная фаза работы Монитора

После включения источника напряжения питания и нажатия на кнопку «Сброс» управление передается Монитору, работа которого начинается с инициализации ППА. Все каналы ППА программируются в режим нестробируемого ввода-вывода (0). Канал А настраивается в режим вывода для выдачи сканирующих импульсов на клавиатуру, а канал В — на ввод сигналов с нее. Линии С0—С3 канала С программируются на вывод и используются для управления светодиодом, отображающим состояние регистра «РУС/ЛАТ», и блоком вывода на

¹ Управляющая программа Монитор разработана Д. В. Горшковым (см. Радио, 1986, № 8, 9).

Таблица 3.1

F800:	C3	36	F8	C3	63	FE	C3	98	F8	C3	BA	FC	C3	46	FC	C3
F810:	BA	FC	C3	01	FE	C3	A5	FC	C3	22	F9	C3	72	FE	C3	7B
F820:	FA	C3	7F	FA	C3	B6	FA	C3	49	F8	C3	16	F8	C3	CE	FA
F830:	C3	52	FF	C3	56	FF	3E	8A	32	03	80	31	CF	36	CD	CE
F840:	FA	21	00	36	11	5F	36	0E	00	CD	ED	F9	21	CF	36	22
F850:	1C	36	21	5A	FF	CD	22	F9	CD	CE	FA	21	FF	35	22	31
F860:	36	21	2A	1D	22	2F	36	3E	C3	32	26	36	31	CF	36	21
F870:	66	FF	CD	22	F9	32	02	80	3D	32	02	A0	CD	EE	F8	21
F880:	6C	F8	E5	21	33	36	7E	FE	58	CA	D3	FF	FE	55	CA	00
F890:	F0	F5	CD	2C	F9	2A	2B	36	4D	44	2A	29	36	EB	2A	27
F8A0:	36	F1	FE	44	CA	C5	F9	FE	43	CA	D7	F9	FE	46	CA	ED
F8B0:	F9	FE	53	CA	F4	F9	FE	54	CA	FF	F9	FE	4D	CA	26	FA
F8C0:	FE	47	CA	3F	FA	FE	49	CA	86	FA	FE	4F	CA	2D	FB	FE
F8D0:	4C	CA	08	FA	FE	52	CA	68	FA	C3	00	F0	3E	33	BD	CA
F8E0:	F1	F8	E5	21	9E	FF	CD	22	F9	E1	2B	C3	F8	21	33	
F8F0:	36	06	00	CD	63	FE	FE	08	CA	DC	F8	FE	7F	CA	DC	F8
F900:	C4	B9	FC	77	FE	0D	CA	1A	F9	FE	2E	CA	6C	F8	06	FF
F910:	3E	52	BD	CA	AE	FA	23	C3	F3	F8	78	17	11	33	36	06
F920:	00	C9	7E	A7	C8	CD	B9	FC	23	C3	22	F9	21	27	36	11
F930:	2D	36	0E	00	CD	ED	F9	11	34	36	CD	5A	F9	22	27	36
F940:	22	29	36	D8	3E	FF	32	2D	36	CD	5A	F9	22	29	36	D8
F950:	CD	5A	F9	22	2B	36	D8	C3	AE	FA	21	00	00	1A	13	FE
F960:	0D	CA	8E	F9	FE	2C	C8	FE	20	CA	5D	F9	D6	30	FA	AE
F970:	FA	FE	0A	FA	82	F9	FE	11	FA	AE	FA	FE	17	F2	AE	FA
F980:	D6	07	4F	29	29	29	29	DA	AE	FA	09	C3	5D	F9	37	C9
F990:	7C	BA	C0	7D	BB	C9	CD	A4	F9	CD	90	F9	C2	A2	F9	33
F9A0:	33	C9	23	C9	CD	72	FE	FE	03	C0	CD	CE	FA	C3	AE	FA
F9B0:	E5	21	6C	FF	CD	22	F9	E1	C9	7E	C5	CD	A5	FC	3E	20
F9C0:	CD	B9	FC	C1	C9	CD	78	FB	CD	B9	F9	CD	96	F9	7D	E6
F9D0:	0F	CA	C5	F9	C3	C8	F9	0A	BE	CA	E6	F9	CD	78	FB	CD
F9E0:	B9	F9	0A	CD	BA	F9	03	CD	96	F9	C3	D7	F9	71	CD	99
F9F0:	F9	C3	ED	F9	79	BE	CC	78	FB	CD	96	F9	C3	F4	F9	7E
FA00:	02	03	CD	99	F9	C3	FF	F9	CD	78	FB	7E	B7	FA	15	FA
FA10:	FE	20	D2	17	FA	3E	2E	CD	B9	FC	CD	96	F9	7D	E6	0F
FA20:	CA	08	FA	C3	08	FA	CD	78	FB	CD	B9	F9	E5	CD	EE	F8
FA30:	E1	D2	3B	FA	E5	CD	5A	F9	7D	E1	77	23	C3	26	FA	CD
FA40:	90	F9	CA	5A	FA	EB	22	23	36	7E	32	25	36	36	F7	3E
FA50:	C3	32	30	00	21	A2	FF	22	31	00	31	18	36	C1	D1	E1
FA60:	F1	F9	2A	16	36	C3	26	36	3E	90	32	03	A0	22	01	A0
FA70:	3A	00	A0	02	03	CD	99	F9	C3	6D	FA	2A	02	36	C9	E5
FA80:	2A	00	36	7E	E1	C9	3A	2D	36	B7	CA	91	FA	7B	32	2F
FA90:	36	CD	B6	FA	CD	78	FB	EB	CD	78	FB	EB	C5	CD	16	FB
FAA0:	60	69	CD	78	FB	D1	CD	90	F9	C8	EB	CD	78	FB	3E	3F
FAB0:	CD	B9	FC	C3	6C	F8	3E	FF	CD	FF	FA	E5	09	EB	CD	FD
FAC0:	FA	E1	09	EB	E5	CD	0A	FB	3E	FF	CD	FF	FA	E1	E5	21
FAD0:	01	C0	36	00	2B	36	4D	36	1D	36	99	36	93	23	36	27
FAE0:	7E	7E	E6	20	CA	E1	FA	21	08	E0	36	80	2E	04	36	D0
FAF0:	36	36	2C	36	23	36	49	2E	08	36	A4	E1	C9	3E	08	CD
FB00:	98	FB	47	3E	08	CD	98	FB	4F	C9	3E	08	CD	98	FB	77
FB10:	CD	99	F9	C3	0A	FB	01	00	00	7E	81	4F	F5	CD	90	F9
FB20:	CA	9F	F9	F1	78	8E	47	CD	99	F9	C3	19	FB	79	B7	CA
FB30:	35	FB	32	30	36	E5	CD	16	FB	E1	CD	78	FB	EB	CD	78
FB40:	FB	EB	E5	60	69	CD	78	FB	E1	C5	01	00	00	CD	46	FC
FB50:	05	E3	E3	C2	4D	FB	0E	E6	CD	46	FC	CD	90	FB	EB	CD
FB60:	90	FB	EB	CD	86	FB	21	00	00	CD	90	FB	0E	E6	CD	46
FB70:	FC	E1	CD	90	FB	C3	CE	FA	C5	CD	B0	F9	7C	CD	A5	FC
FB80:	7D	CD	BA	F9	C1	C9	4E	CD	46	FC	CD	99	F9	C3	86	FB
FB90:	4C	CD	46	FC	4D	C3	46	FC	E5	C3	D5	57	3E	80	32	08
FBA0:	EO	21	00	00	39	31	00	00	22	0D	36	0E	00	3A	02	80
FBB0:	0F	0F	0F	0F	E6	01	5F	F1	79	E6	7F	07	4F	26	00	25
FBC0:	CA	34	FC	F1	3A	02	80	0F	0F	0F	0F	E6	01	BB	CA	BF
FBD0:	FB	B1	4F	15	3A	2F	36	C2	DC	FB	D6	12	47	F1	05	C2
FBE0:	DD	FB	14	3A	02	80	0F	0F	0F	0F	E6	01	5F	7A	B7	F2
FBF0:	0B	FC	79	FE	E6	C2	FF	FB	AF	32	2E	36	C3	09	FC	FE

FC00:	19	C2	B7	FB	3E	FF	32	2E	36	16	09	15	C2	B7	FB	21
FC10:	04	E0	36	D0	36	36	23	36	23	36	49	3E	27	32	01	C0
FC20:	3E	E0	32	01	C0	2E	08	36	A4	2A	0D	36	F9	3A	2E	36
FC30:	A9	C3	A1	FC	2A	0D	36	F9	CD	CE	FA	7A	B7	F2	AE	FA
FC40:	CD	A4	F9	C3	9C	FB	E5	C5	D5	F5	3E	80	32	08	E0	21
FC50:	00	00	39	31	00	00	16	08	F1	79	07	4F	3E	01	A9	32
FC60:	02	80	3A	30	36	47	F1	05	C2	66	FC	3E	00	A9	32	02
FC70:	80	15	3A	30	36	C2	7A	FC	D6	0E	47	F1	05	C2	7B	FC
FC80:	14	15	C2	58	FC	F9	21	04	E0	36	D0	36	36	23	36	23
FC90:	36	49	3E	27	32	01	C0	3E	E0	32	01	C0	2E	08	36	A4
FCA0:	F1	D1	C1	E1	C9	F5	0F	0F	0F	0F	CD	AE	FC	F1	E6	0F
FCB0:	FE	0A	FA	B7	FC	C6	07	C6	30	4F	F5	C5	D5	E5	CD	01
FCC0:	FE	21	85	FD	E5	2A	02	36	EB	2A	00	36	3A	04	36	3D
FCD0:	FA	EE	FC	CA	65	FD	E2	73	FD	79	D6	20	4F	0D	FA	E9
FCE0:	FC	C5	CD	B9	FD	C1	C3	DD	FC	AF	32	04	36	C9	79	E6
FCF0:	7F	4F	FE	1F	CA	A3	FD	FE	0C	CA	B2	FD	FE	0D	CA	F3
FD00:	FD	FE	0A	CA	47	FD	FE	08	CA	D6	FD	FE	18	CA	B9	FD
FD10:	FE	19	CA	E2	FD	FE	1A	CA	C5	FD	FE	1B	CA	9E	FD	FE
FD20:	07	C2	38	FD	01	F0	05	78	FB	3D	C2	28	FD	78	F3	3D
FD30:	C2	2E	FD	0D	C2	27	FD	C9	71	CD	B9	FD	7A	FE	03	C0
FD40:	7B	FE	08	C0	CD	E2	FD	7A	FE	1B	C2	C5	FD	E5	D5	21
FD50:	C2	37	11	10	38	01	9E	07	1A	77	23	13	08	79	B0	C2
FD60:	58	FD	D1	E1	C9	79	FE	59	C2	E9	FC	CD	B2	FD	3E	02
FD70:	C3	EA	FC	79	D6	20	4F	0D	3E	04	FA	EA	FC	C5	CD	C5
FD80:	FD	C1	C3	77	FD	22	00	36	EB	22	02	36	3E	80	32	01
FD90:	C0	7D	32	00	C0	7C	32	00	C0	E1	D1	C1	F1	C9	3E	01
FDA0:	C3	EA	FC	21	F4	3F	11	25	09	AF	77	2B	1B	7B	B2	C2
FDB0:	A9	FD	11	08	03	21	C2	37	C9	7B	23	1C	FE	47	C0	1E
FDC0:	08	01	C0	FF	09	7A	FE	1B	01	4E	00	C2	D3	FD	16	02
FDD0:	01	B0	F8	14	09	C9	7B	2B	1D	FE	08	C0	1E	47	01	40
FDE0:	00	09	7A	FE	03	01	B2	FF	C2	F0	FD	16	1C	01	50	07
FDFO:	15	09	C9	7D	93	D2	F9	FD	25	6F	1E	08	01	08	00	09
FE00:	C9	3A	02	80	E6	80	CA	0E	FE	3A	05	36	B7	C0	E5	2A
FE10:	09	36	CD	72	FE	BD	6F	CA	2A	FE	3E	01	32	08	36	26
FE20:	15	AF	22	09	36	E1	32	05	36	C9	25	C2	21	FE	3C	CA
FE30:	22	FE	3C	CA	51	FE	C5	01	03	50	CD	27	FD	C1	3A	0B
FE40:	36	26	E0	3D	32	08	36	CA	4C	FE	26	40	3E	FF	C3	22
FE50:	FE	3A	02	80	E6	80	CA	51	FE	3A	06	36	2F	32	06	36
FE60:	C3	1A	FE	CD	01	FE	B7	CA	63	FE	AF	32	05	36	3A	09
FE70:	36	C9	3A	02	80	E6	80	C2	7D	FE	3E	FE	C9	AF	32	00
FE80:	80	32	02	80	3A	06	36	E6	01	F6	06	32	03	80	3A	01
FE90:	80	3C	C2	97	FE	3D	C9	E5	2E	01	26	07	7D	0F	6F	2F
FEA0:	32	00	80	3A	01	80	2F	B7	C2	B3	FE	25	F2	9C	FE	3E
FEB0:	FF	E1	C9	2E	20	3A	01	80	2F	B7	CA	AF	FE	2D	C2	B5
FEC0:	FE	2E	08	2D	07	D2	C3	FE	7C	65	6F	FE	01	CA	FA	FE
FED0:	DA	F3	FE	07	07	07	C6	20	B4	FE	5F	C2	06	FF	3E	20
FEE0:	E1	C9	09	0A	0D	7F	08	19	18	1A	0C	1F	1B	00	01	02
FEF0:	03	04	05	7C	21	EA	FE	C3	FE	FE	7C	21	E2	FE	85	6F
FF00:	7E	FE	40	E1	D8	E5	6F	3A	02	80	67	E6	40	C2	1A	FF
FF10:	7D	FE	40	FA	3F	FF	E6	1F	E1	C9	3A	06	36	B7	CA	2A
FF20:	FF	7D	FE	40	FA	2A	FF	F6	20	6F	7C	E6	20	C2	3F	FF
FF30:	7D	FE	40	FA	3B	FF	7D	EE	20	E1	C9	7D	E6	2F	6F	7D
FF40:	FE	40	E1	F0	E5	6F	E6	0F	FE	0C	7D	FA	50	FF	EE	10
FF50:	E1	C9	2A	31	36	C9	22	31	36	C9	1F	72	61	64	69	6F
FF60:	2D	38	36	72	6B	00	0D	0A	2D	2D	3E	00	0D	0A	18	18
FF70:	18	18	00	0D	0A	20	50	43	2D	0D	0A	20	48	4C	2D	0D
FF80:	0A	20	42	43	2D	0D	0A	20	44	45	2D	0D	0A	20	53	50
FF90:	2D	0D	0A	20	41	46	2D	19	19	19	19	19	19	00	08	20
FFA0:	08	00	22	16	36	F5	E1	22	1E	36	E1	2B	22	14	36	21
FFB0:	00	00	39	31	1E	36	E5	D5	C5	2A	14	36	31	CF	36	CD
FFC0:	78	FB	EB	2A	23	36	CD	90	F9	C2	6C	F8	3A	25	36	77
FFD0:	C3	6C	F8	21	73	FF	CD	22	F9	21	14	36	06	06	5E	23
FFE0:	56	C5	E5	EB	CD	78	FB	CD	EE	F8	D2	F6	FF	CD	5A	F9
FFF0:	D1	D5	EB	72	2B	73	E1	C1	05	23	C2	DE	FF	C9	BB	89

магнитофон, а линии С4—С7 — в режим ввода информации о нажатии специальных клавиш и ввода сигнала с магнитофона.

Далее Монитор настраивает контроллер ПДП. Для этого в его внутренние регистры заносятся адрес начала экранной области ОЗУ и количество передаваемых байтов. При передаче каждого байта в режиме ПДП значение адреса увеличивается, а содержимое счетчика байтов уменьшается на единицу. Так как контроллер всегда управляет передачей кодов символов из одной и той же области ОЗУ, устанавливается режим работы с автозагрузкой, характерный тем, что после завершения передачи из экранной области ОЗУ всех кодов символов в контроллере ПДП происходит автоматическая перезагрузка внутренних регистров исходными параметрами и процесс формирования телевизионного кадра начинается сначала.

Инициализация контроллера дисплея сводится к следующему: в его регистры заносится информация о формате знакоместа, экрана, курсора, а также о длительности импульсов HRTC и VRTC. Значения всех этих параметров тесно связаны между собой и зависят от частоты сигнала, подаваемого на вход CCLK контроллера, поэтому параметры настройки контроллера не могут быть выбраны произвольно. После окончания настройки контроллеров ПДП и дисплея происходит их запуск.

Ввод директив и анализ результатов

После запуска Монитора экран телевизора очищается, в левом верхнем углу появляется надпись «Радио-86РК», а под ней — стрелка «→», уведомляющая пользователя о том, что Монитор готов к вводу очередной директивы. Неверно набранные символы стирают нажатием на клавишу ЗБ («Забой») или «←» («Курсор влево»). Для выполнения директивы нажимают на клавишу возврата каретки ВК. Если директива задана правильно, то начнется ее выполнение, если нет — на экране появится знак вопроса — признак того, что Монитор «не понимает» вашу директиву.

Выполнение директив D, L, S может быть прервано. Для этого, удерживая клавишу «УС», надо нажать еще и на «С»: выполнение директивы прервется, и Монитор будет готов выполнять следующую.

Имена всех директив состоят из одной латинской буквы, непосредственно за которой могут следовать не более трех параметров, представляющих собой шестнадцатиричные числа. Один параметр от другого отделяют запятой. Ее ставят и в тех случаях, когда один из параметров (в том числе и первый) отсутствует.

Директивы работы с памятью

Содержимое области памяти может быть выведено на экран дисплея либо в виде шестнадцатиричных чисел (директива D) либо в виде алфавитно-цифровых символов, соответствующих этим кодам (директива L). Если при выполнении последней встретятся коды, не соответствующие ни одному алфавитно-цифровому символу, то они отобразятся в виде точек. Содержимое памяти выводится в виде таблицы из 16 колонок. Слева от каждой строки указывается шестнадцатиричный адрес первой в строке ячейки. Форматы этих и других директив Монитора приведены в табл. 3.2.

Директивы работы с памятью

D (НАЧАЛЬНЫЙ АДРЕС), (КОНЕЧНЫЙ АДРЕС)
 L (НАЧАЛЬНЫЙ АДРЕС), (КОНЕЧНЫЙ АДРЕС)
 F (НАЧАЛЬНЫЙ АДРЕС), (КОНЕЧНЫЙ АДРЕС), (ЗАПИСЫВАЕМЫЙ КОД)
 M (АДРЕС)
 T (НАЧАЛЬНЫЙ АДРЕС), (КОНЕЧНЫЙ АДРЕС), (АДРЕС ОБЛАСТИ ПЕРЕСЫЛКИ)
 C (НАЧАЛЬНЫЙ АДРЕС), (КОНЕЧНЫЙ АДРЕС), (АДРЕС ОБЛАСТИ СРАВНЕНИЯ)
 S (НАЧАЛЬНЫЙ АДРЕС), (КОНЕЧНЫЙ АДРЕС), (ИСКОМЫЙ КОД)

Директивы запуска и отладки

G (АДРЕС ЗАПУСКА),/(АДРЕС ОСТАНОВА)/
 X ОТОБРАЖЕНИЕ СОДЕРЖИМОГО ВНУТРЕННИХ РЕГИСТРОВ

Директивы ввода-вывода

O (НАЧАЛЬНЫЙ АДРЕС), (КОНЕЧНЫЙ АДРЕС),/(СКОРОСТЬ)/
 I (СМЕЩЕНИЕ),/(СКОРОСТЬ)/

Дополнительные директивы

R (НАЧАЛЬНЫЙ АДРЕС ПЗУ), (КОНЕЧНЫЙ АДРЕС ПЗУ), (АДРЕС ЗАГРУЗКИ)

Примечание. Наклонными чертами выделены необязательные параметры.

Директива M предназначена для просмотра и изменения содержимого одной или нескольких ячеек памяти. После ее ввода на экране высвечивается адрес ячейки и ее содержимое, курсор останавливается справа от этого значения, и Монитор ожидает ввода с клавиатуры. Если необходимо изменить содержимое ячейки, набирают новое значение и нажимают клавишу ВК, если изменений не требуется, ее нажимают сразу же. При каждом нажатии на клавишу ВК значение адреса автоматически увеличивается на единицу. Выполнение директивы продолжается до тех пор, пока не будет нажата клавиша «.» (точка).

Если во все ячейки области памяти необходимо записать одинаковые коды, удобно воспользоваться директивой F.

Директива T предназначена для копирования (пересылки) содержимого одной области памяти в другую. Копирование осуществляется побайтно, начиная с младшего адреса.

Для сравнения двух областей памяти необходимо пользоваться директивой C. Если содержимое соответствующих ячеек памяти не совпадает, на экран выводится адрес ячейки из первой области, ее содержимое и содержимое соответствующей ячейки из второй области. Программа Монитор предоставляет пользователю возможность поиска кода в заданной области памяти (директива S).

При обнаружении искомого кода на экране появляются адреса ячеек, в которых он был обнаружен.

Директивы запуска и отладки программ

Для запуска программы служит директива G. Второй параметр этой директивы, задающий адрес останова, используется только при отладке программ и может быть опущен. Кроме того, пользователь может сам назначать в своей программе контрольные адреса останова, записав по этим адресам код команды RST 6 (OF7H). Если при выполнении программы встретится эта команда, управление будет передано Монитору, который сообщит адрес, в котором было прервано выполнение программы, после чего пользователь может воспользоваться любыми директивами Монитора для контроля результатов работы и модификации программы.

Просмотреть и изменить содержимое внутренних регистров микропроцессора поможет директива X (не имеющая параметров). В результате ее выполнения на экран выводятся символические имена и содержимое регистров, которое можно изменять так же, как и содержимое ячеек памяти по директиве M. Регистр признаков результата операции обозначен F, другие внутренние регистры микропроцессора обозначены стандартно.

Директивы ввода-вывода

Первые два параметра директивы вывода на магнитную ленту O задают область памяти, содержимое которой подлежит выводу, третий — шестнадцатичный код, определяет скорость вывода. Если скорость не указана, то будет использовано либо значение, заданное в предыдущей директиве вывода, либо стандартное — 1DH (около 1100 бит/с), записываемое (при нажатии на кнопку «Сброс») в рабочую ячейку Монитора 03630H (использование константы вывода менее 10H недопустимо!). Скорость выбрана с учетом использования магнитного фона и магнитной ленты невысокого качества. Кроме того, такая скорость облегчит обмен программами. После завершения вывода на экране отобразятся начальный и конечный адреса и четырехзначная контрольная сумма выведенной информации.

Ввод с ленты осуществляется по директиве I, которая может иметь два параметра. Первым параметром — необязательным — задают смещение. Если он есть, вводимая информация будет загружена по адресу, являющемуся суммой указанного в записи на ленте адреса и смещения. Второй параметр определяет временную задержку при чтении с ленты. Он также может отсутствовать, но в этом случае будет взята задержка, использовавшаяся в предыдущей команде ввода (если вы не нажимали на кнопку «Сброс») или установленная (по умолчанию) при начальной настройке рабочих ячеек Монитора (стандартное значение, записанное по адресу 0362FH-2AH).

После окончания ввода Монитор сообщит начальный и конечный адреса загрузки и контрольную сумму, подсчитанную при вводе информации. Если она не совпадает с введенной с ленты, то на следующей строке будет выведено значение, записанное на ленте, и вопросительный знак. Это свидетельствует об ошибках при чтении информации с ленты.

Прервать программу ввода с магнитной ленты можно либо выключив магнитофон, либо введя с клавиатуры код $УС + С$ (здесь и далее латинский регистр).

В табл. 3.2 приведены форматы еще одной директивы R — чтения информации из ПЗУ, подключаемого к ППА D14.

Подпрограммы Монитора

Программа Монитор содержит набор подпрограмм ввода-вывода (табл. 3.3), к которым программа пользователя может обращаться, соблюдая соглашения об обмене информацией. Этот набор обеспечивает совместимость программного обеспечения аналогичных компьютеров при условии, что в программах соблюдаются все правила вызова подпрограмм.

Следует заметить, что при использовании подпрограмм ввода и вывода байта на ленту возможно нарушение процесса отображения на экране телевизора, поэтому после завершения работы с этими подпрограммами рекомендуется вызвать подпрограмму запуска отображения экранного буфера. Кроме того, при написании программ с использованием подпрограмм ввода-вывода необходимо учитывать, что для надежной работы время между обращениями к этим подпрограммам должно быть около 55 мкс (110 тактов работы микропроцессора).

В набор подпрограмм входят также подпрограммы ввода с магнитной ленты и вывода на нее блоков памяти, а также подпрограмма вычисления контрольной суммы такого блока. При использовании этих подпрограмм запуск отображения на экране производится автоматически, так же как и при вводе-выводе по директивам I и O. При выводе на ленту необходимо указать в определенных регистрах (табл. 3.3) адреса начала и конца блока, а также его контрольную сумму, подсчитанную подпрограммой Монитора или собственной подпрограммой пользователя. При вводе блока с ленты можно указать смещение, с которым блок данных должен быть загружен в ОЗУ. Подпрограмма чтения блока информации возвращает (в регистрах микропроцессора) адреса загрузки и считанную с ленты контрольную сумму, которые затем могут быть проверены в программах пользователя.

Подпрограмма опроса кода нажатой клавиши позволяет ускорить опрос клавиатуры и более эффективно использовать ее в программах, работающих в реальном масштабе времени.

С помощью подпрограммы запроса положения курсора можно определить его местонахождение на экране телевизора, а с помощью подпрограммы запроса экранного байта — узнать код символа из экранной области памяти, находящегося в позиции курсора. Для считывания произвольного байта из экранного буфера следует предварительно установить курсор в нужную позицию экрана.

В Мониторе предусмотрены также две подпрограммы для определения верхней границы свободной оперативной памяти. Первая из них возвращает программе пользователя в регистровой паре HL установленный адрес верхней границы доступного ОЗУ (по умолчанию — 035FFH для РК с объемом ОЗУ 16 Кбайт), вторая позволяет установить новую границу. Недопустимо устанавливать верхнюю границу свободной памяти выше ее положения по умолчанию, так как это может привести к неправильной работе вашей программы и искажению данных в рабочих ячейках Монитора!

Таблица 3.3

Назначение	Адрес вызова		Параметры
Ввод символа с клавиатуры	0F803H —2045D	Входные Выходные	— А — код символа
Вывод символа на экран	0F809H —2039D	Входные Выходные	С — код символа —
Опрос состояния клавиатуры	0F812H —2030D	Входные Выходные	— А-00 — не нажата А-0FFH — нажата А — выводимый байт
Вывод байта на экран в шестнадцатичном виде	0F815H —2027D	Входные Выходные	—
Вывод на экран сообщения	0F818H —2024D	Входные Выходные	HL — адрес сообщения —
Опрос кода нажатой клавиши	0F81BH —2021D	Входные Выходные	— А-0FFH — не нажата А-0FEN — «РУС/ЛАТ» Иначе — код клавиши
Запрос положения курсора на экране	0F81EH —2018D	Входные Выходные	— Н — номер строки +3 L — номер позиции +8
Запрос байта из экранного буфера	0F821H —2015D	Входные Выходные	— А — код из буфера
Ввод байта с магнитофона	0F806H —2042D	Входные	А-0FFH — с поиском синхробайта А-08 — без поиска синхробайта
Запись байта на магнитофон	0F80CH —2036D	Выходные- Входные	А — введенный байт С — выводимый байт
Ввод блока с магнитофона	0F824H —2012D	Входные Выходные	— HL — смещение HL — адрес начала DE — адрес конца BC — контрольная сумма
Вывод блока на магнитофон	0F827H —2009D	Входные	HL — адрес начала DE — адрес конца BC — контрольная сумма
Подсчет контрольной суммы блока	0F82AH— —2006D	Выходные Входные	— HL — адрес начала DE — адрес конца BC — контрольная сумма
Разрешение отображения информации на экране	0F82DH —2003D	Выходные Входные	— —
Определение адреса верхней границы	0F830H —2000D	Входные Выходные	— HL — адрес границы
Установка адреса верхней границы свободной памяти	0F833H —1997D	Входные Выходные	HL — адрес —

Распределение оперативной памяти при работе Монитора

Как было уже сказано, ОЗУ ПК имеет информационную емкость 16 Кбайт и располагается, начиная с адреса 0000H по 3FFFH (табл. 3.4,а) включительно. Следует иметь в виду, что часть памяти (3600H—3FFFH) отве-

Таблица 3.4а

ПЗУ МОНИТОРА (2К)	FFFFH
РЕГИСТРЫ ПЕРИФЕРИЙНЫХ БИС	F800H
БУФЕР ЭКРАНА	8000H
РАБОЧИЕ ЯЧЕЙКИ МОНИТОРА	37C0H (77C0H)
СТЕК БЕЙСИКА	3600H (7600H)
ПЕРЕМЕННЫЕ И МАССИВЫ	
ТЕКСТ ПРОГРАММЫ НА БЕЙСИКЕ	
ОБЛАСТЬ ПОДПРОГРАММ ПОЛЬЗОВАТЕЛЯ	1C00H
ИНТЕРПРЕТАТОР БЕЙСИКА	1A00H
	0000H

Таблица 3.4б

Необходимо заменить код 36H на 76H в ячейках ПЗУ:

F83D	F92E	FA88	FCC7	FE6D
F843	F931	FA90	FCCB	FE70
F846	F939	FAF1	FCCE	FE86
F84E	F93F	FB34	FCEC	FF1C
F851	F942	FBAA	FD87	FF54
F860	F948	FBD6	FD8B	FF5B
F866	F94E	FBFB	FE0B	FFA4
F86B	F955	FC08	FE11	FFA9
F86E	FA48	FC15	FE1E	FFAE
F885	FA4C	FC2B	FE24	FF85
F897	FA5C	FC2F	FE28	FFBB
F89C	FA64	FC36	FE40	FFBE
F8A0	FA67	FC64	FE46	FFC5
F8FO	FA7D	FC74	FE5B	FFCE
F91E	FA82	FC8C	FE5F	FFDB

Необходимо заменить в ячейках ПЗУ:

F85D:35H на 75H	FDA5:3FH на 7FH
FD51: 37H на 77H	FDB7: 37H на 77H
FD54: 38H на 78H	

дена под рабочие ячейки Монитора и экранную область ОЗУ, поэтому использовать ее при разработке и отладке программ нельзя. Остальная часть ОЗУ (0000H—35FFH) предназначена для программ и данных пользователя. Исключение составляют лишь три ячейки памяти с адресами 0030H, 0031H и 0032H, которые необходимы для организации отладочного режима запуска программ с использованием адресов останова, поэтому использовать эти ячейки в Вашей программе не рекомендуется.

При необходимости объем ОЗУ ПК может быть увеличен до 32 Кбайт.

Это потребует внесения изменений и в программное обеспечение. Рабочие ячейки Монитора и экранная область ОЗУ в этом случае будут находиться в области памяти с адресами 7600H—7FFFH. В табл. 3.4,б приведены изменения, которые необходимо внести в Монитор для работы с ОЗУ объемом 32 Кбайт.

Особенности клавиатуры РК

О назначении клавиши «РУС/ЛАТ» уже говорилось ранее. В отличие от остальных клавиш, она срабатывает при отпускании. Этот эффект можно использовать для приостановки вывода сообщений на экран телевизора. После отпускания клавиши вывод будет продолжен.

Для кратковременного переключения регистров можно пользоваться и другой клавишей — СС.

Еще одним полезным свойством клавиатуры является автоматическое повторение выдачи кода символа при длительном нажатии на клавишу.

Управляющие коды дисплея

Управляющие коды, в отличие от остальных, не отображаются в виде алфавитно-цифрового или псевдографического символа, а вызывают выполнение какой-либо специфичной функции, связанной с управлением форматом выводимых на экран сообщений (табл. 3.5).

Функция «←» («Курсор влево») вызывает перемещение курсора на одну позицию влево. Если курсор находился в самой левой позиции строки, он переместится в последнюю позицию предыдущей строки, а если он находился в нулевой позиции экрана — в последнюю позицию последней строки.

Функция «→» («Курсор вправо») перемещает курсор в противоположном направлении.

Таблица 3.5

Функция	Коды	Ввод с клавиатуры
Курсор влево	08H	«←» или УС+H
Курсор вправо	18H	«→» или УС+X
Курсор вверх	19H	«↑» или УС+Y
Курсор вниз	1AH	«↓» или УС+Z
Возврат каретки	0DH	«BK» или УС+M
Перевод строки	0AH	«PC» или УС+J
Очистка экрана	1FH	«СТР» или УС+3B
Курсор в начало экрана	0CH	«\» или УС+L
Прямая адресация курсора	1BH, 59H 20H + (Номер строки), 20H + (номер позиции)	

Примечание. Символ «+» между обозначениями клавиш означает, что надо нажать на вторую клавишу, держа первую уже в нажатом состоянии.

Здесь и далее символ «\» означает «курсор в начало экрана».

Функции «↑» («Курсор вверх») и «↓» («Курсор вниз») вызывают перемещение курсора на одну строку вверх или вниз соответственно. Если курсор находился в самой нижней строке экрана, то при выполнении функции «Курсор вниз» он переместится в ту же позицию верхней строки, функция «Курсор вверх» из самой верхней строки переместит его в самую нижнюю.

Функция ВК («Возврат каретки», не путайте с клавишей!) переведет курсор в первую позицию той же строки экрана, в которой он и находился. Если курсор уже находится в самой левой позиции, его положение не изменится.

Функция ПС («Перевод строки») действует так же, как и «Курсор вниз», в том случае, если курсор не находится в последней строке экрана, в противном случае курсор остается в прежней позиции, и на экране происходят следующие изменения: на месте первой строки появляется вторая, на месте второй — третья и т. д., последняя строка экрана очищается. Таким образом, текст на экране дисплея передвигается на одну строку вверх, вся информация, высвечиваемая в первой строке, теряется, а последняя строка освобождается для вывода новой строки символов.

Функция СТР («Стирание экрана») полностью стирает весь текст на экране и устанавливает курсор в нулевую позицию (левый верхний угол).

Функция «\» («Курсор в начало экрана») перемещает курсор в левый верхний угол экрана. Информация на экране остается неизменной.

Если обратиться к подпрограмме вывода символа на экран (по адресу 0F809H), записав предварительно в регистр С микропроцессора код 07H, то РК выдаст звуковой сигнал продолжительностью примерно 0,25 с. Такой сигнал подается и при одновременном нажатии клавиш УС и G.

В заключение — о функции прямой адресации курсора. Чтобы установить курсор в требуемую позицию на экране, необходимо выдать на дисплей последовательность кодов: $IVH + 59H + (\text{НОМЕР СТРОКИ} + 20H) + (\text{НОМЕР ПОЗИЦИИ} + 20H)$. Строки и позиции на экране отсчитываются от 0, причем нулевой строкой экрана считается самая верхняя строка, а нулевой позицией — самая левая. Сложность данной управляющей последовательности объясняется стремлением обеспечить совместимость с наборами кодов дисплеев промышленного производства.

3.2. Интерпретатор языка Бейсик

Язык программирования высокого уровня Бейсик был разработан еще в 1964 г., но широко стал применяться только после появления микропроцессоров и микроЭВМ. Бейсик, в отличие от других языков программирования, очень прост для освоения его начинающими программистами. Однако, несмотря на простоту, он позволяет создавать сложные программы, решающие самые разнообразные задачи: от игровых до программ статической обработки информации, автоматизации проектирования и управления различными объектами.

К недостаткам Бейсика следует отнести отсутствие общепринятого стандарта на основные элементы языка и отсутствие средств для написания структурированных программ. Несмотря на эти недостатки, мы все же считаем этот язык наиболее подходящим для простых микроЭВМ и предлагаем читателям вариант транслятора, который при малом объеме (7 Кбайт) имеет достаточно большие

возможности и позволяет при незначительных переделках исполнять на микро-ЭВМ практически любые программы, написанные на Бейсике, но разработанные для других ЭВМ.

Описываемый транслятор является *интерпретатором*. Это означает, что в памяти микроЭВМ должны одновременно находиться как сама программа-интерпретатор, так и программа, написанная на языке Бейсик. Такой подход позволяет возложить на интерпретатор, кроме трансляции программы, и некоторые дополнительные функции, в частности редактора для подготовки текстов программ и внесения в них изменений и функции отладчика для эффективного поиска ошибок. Таким образом, интерпретатор, загруженный в ОЗУ микроЭВМ и запущенный в работу, позволяет как разрабатывать новые программы, так и загружать (с магнитной ленты), запускать в работу программы, составленные ранее. Такой интерпретатор требует наличия оперативной памяти объемом не менее 12 Кбайт (начиная с адреса 0000H). Интерпретатор хранится на магнитной ленте и загружается в ОЗУ с помощью директивы монитора I. Для ввода-вывода информации интерпретатор использует подпрограммы монитора.

В табл. 3.6 приведены коды интерпретатора. Вы можете набрать программу с помощью директивы M монитора сразу или по частям, записывая набранную информацию на магнитную ленту с помощью директивы O. При наборе такой относительно большой программы легко допустить ошибки, поэтому ниже приведены коды программы, предназначенной для подсчета контрольных сумм:

2800	31	F0	28	21	43	28	CD	18	F8	21	00	00	11	B7	28	AF
2810	86	23	47	7D	B7	78	C2	10	28	1A	B8	C4	31	28	13	7C
2820	FE	19	C2	0F	28	21	64	28	CD	18	F8	CD	03	E8	C3	00
2830	F8	E5	21	A3	28	CD	18	F8	E1	E5	D5	25	7C	CD	15	F8
2840	D1	E1	C9	1F	2A	6D	69	6B	72	6F	2F	38	80	2A	20	70
2850	72	6F	67	72	61	6D	6D	61	20	70	72	6F	77	65	72	6B
2860	69	0D	0A	00	0D	0A	6B	6F	6E	65	63	20	72	61	62	6F
2870	74	79	2E	0D	0A	64	6C	71	20	77	6F	7A	77	72	61	74
2880	61	20	77	20	6D	6F	6E	69	74	6F	72	20	6E	61	76	6D
2890	69	74	65	20	6C	60	62	75	60	20	6B	6C	61	77	69	7B
28A0	75	2E	00	0D	0A	6F	7B	69	62	6B	61	20	77	20	62	6C
28B0	6F	6B	65	20	2D	20	00	95	3A	14	4B	58	61	E4	90	3B
28C0	D6	F4	6A	74	CB	33	93	65	70	BF	D6	A2	B4	3D	1A	6A

Эту программу запускают в работу при директиве G 2400. Контрольная сумма подсчитывается для каждого из блоков объемом 256 байт. При обнаружении ошибки на экране дисплея высвечивается номер ошибочно введенного блока. Номера блоков выдаются в шестнадцатиричном виде. Например, если ошибка обнаружена в блоке номер 0A, то искать ее надо в области памяти с адреса 0AF0H по 0AFFH. Запускать программу проверки целесообразно только после набора всей программы интерпретатора.

Запускают интерпретатор в работу по директиве монитора G0.

На экране дисплея при этом появляется сообщение:

* РАДИО-86PK*BASIC

⇒

Это значит, что интерпретатор готов к приему директив, задаваемых оператором.

Таблица 3.6

0000:	31	FF	3F	C3	42	17	23	E3	7E	E3	BE	23	E3	C2	D0	02
0010:	23	7E	FE	3A	D0	C9	D0	05	F5	3A	17	02	B7	C3	B7	04
0020:	7C	92	CC	7D	93	92	01	00	3A	50	02	B7	C2	CA	02	C9
0030:	E3	22	41	00	E1	C3	3B	00	00	00	00	00	C2	46	23	C3
0040:	C3	F9	04	D4	12	92	13	00	C9	36	17	0A	0C	75	0F	A8
0050:	0C	15	13	2A	14	7E	11	99	15	60	16	0E	16	14	0F	D8
0060:	16	24	17	E7	0E	1F	0D	C8	0F	7B	8A	11	7B	C3	14	44
0070:	0F	54	0F	79	4C	14	79	7D	10	4C	D3	44	44	D2	12	7F
0080:	50	15	50	77	0A	46	49	4E	43	55	D4	49	49	08	4E	45
0090:	58	44	44	41	54	C1	4F	54	50	52	D4	44	44	55	52	45
00A0:	41	C4	43	55	D2	47	4F	53	55	55	CE	4E	4E	C6	4F	45
00B0:	53	54	4F	52	C5	47	4F	53	55	C2	52	45	45	55	4F	CE
00C0:	52	45	CD	53	54	4F	4B	C5	50	4C	4C	4E	4E	44	4C	C6
00D0:	4C	4F	4E	C5	50	4F	48	C5	50	4C	4C	4E	4E	44	4C	4F
00E0:	43	49	4E	D4	4C	49	4F	53	55	4C	4E	4E	4E	44	4C	4F
00F0:	41	C4	44	53	41	56	41	4E	45	D7	54	41	42	A8	54	CF
0100:	53	50	43	A8	46	CE	D3	55	45	CE	4E	4F	D4	53	54	45
0110:	D0	AB	AA	AA	AF	DE	41	4E	C4	4F	D2	46	BD	BC	47	D0
0120:	C0	49	4E	D4	41	42	52	4E	C4	4C	4F	50	45	45	4E	D0
0130:	50	4F	D3	53	51	D2	52	4E	C4	4C	4F	50	45	58	08	43
0140:	4F	D3	53	49	CE	54	41	CE	41	54	CE	43	49	48	44	4C
0150:	45	C5	53	54	52	A4	56	41	CC	41	53	C3	49	44	A4	00
0160:	4C	4E	46	54	A4	52	49	47	48	08	15	06	79	08	6A	17
0170:	B3	17	35	05	1D	09	07	06	52	B7	E3	06	06	06	EF	05
0180:	C7	06	AB	06	78	07	17	18	2C	17	91	07	B0	08	17	06
0190:	EE	04	82	06	05	19	47	18	9D	03	30	B1	30	B2	30	B3
01A0:	80	0A	80	06	05	B4	30	B7	30	B8	30	B9	31	B0	31	B1
01B0:	30	31	B2	31	B3	B4	31	B5	31	B6	31	B7	31	B8	3C	9C
01C0:	31	B2	31	B3	31	B4	31	B5	31	B6	31	B7	31	B8	3C	9C
01D0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
01E0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
01F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0200:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0210:	00	32	32	37	30	00	00	00	00	01	00	FF	3F	1F	02	06
0220:	00	6B	02	01	01	89	00	00	00	00	00	06	00	6B	02	FF
0230:	3F	0D	01	00	00	00	00	00	22	03	01	FF	FF	6E	0A	00
0240:	00	CD	3F	01	22	03	03	22	17	D5	01	00	FF	FF	02	84
0250:	87	C2	20	32	35	36	00	30	30	30	00	00	00	00	06	FD
0260:	7B	69	62	6B	E1	00	20	20	77	A0	00	0D	0A	BD	3E	0D
0270:	0A	00	0D	0A	73	74	6F	70	A0	00	21	04	00	39	92	23
0280:	FE	81	CO	4E	23	44	23	E3	69	60	7A	B3	EB	CA	7E	02
0290:	EB	E7	01	0D	08	E1	C8	09	C3	7E	02	B3	EB	CA	7E	02
02A0:	C1	E7	7E	02	C8	0B	2B	C3	A1	02	E3	4E	2B	EB	C3	E3
02B0:	49	02	06	00	09	09	CD	B3	02	E1	C9	D5	EB	21	DA	02
02C0:	39	E7	D1	D0	09	1E	0C	C3	D8	02	2A	33	02	2A	3B	02
02D0:	1E	02	01	1E	14	01	1E	00	CD	C2	03	AF	32	17	02	CD
02E0:	DC	02	21	AA	01	57	3E	3F	DF	19	7E	DF	17	DF	02	5F
02F0:	02	02	93	0D	2A	3B	7C	7C	AS	3C	C4	3D	14	AF	32	17
0300:	02	21	F3	FF	22	3B	02	21	6B	02	CD	93	0D	CD	80	04
0310:	D7	3D	3D	CA	0D	03	03	CD	61	06	D5	CD	E6	03	03	D1
0320:	F2	3C	AD	C5	D5	03	FF	CD	61	06	D5	CD	E6	03	03	D1
0330:	F2	3C	AD	C5	D5	03	FF	CD	61	06	D5	CD	E6	03	03	D1
0340:	F2	3C	AD	C5	D5	03	FF	CD	61	06	D5	CD	E6	03	03	D1
0350:	F2	3C	AD	C5	D5	03	FF	CD	61	06	D5	CD	E6	03	03	D1
0360:	F2	3C	AD	C5	D5	03	FF	CD	61	06	D5	CD	E6	03	03	D1
0370:	B6	F7	23	13	B7	C2	23	23	CD	23	23	23	23	23	23	23
0380:	F2	EB	C3	6C	03	2A	43	02	44	4D	7E	23	23	23	23	23
0390:	F7	F7	E7	E7	E1	C1	3F	C8	3F	D0	C3	B8	03	CO	3A	43
03A0:	02	AF	77	23	77	23	22	45	02	2A	43	02	22	22	22	22
03B0:	2A	1B	02	22	2F	02	CD	DB	05	2A	45	02	22	47	02	22
03C0:	49	02	C1	2A	41	02	F9	21	1F	02	22	1D	02	21	00	00
03D0:	E5	22	3F	02	2A	37	02	AF	32	35	02	C5	C9	3F	DF	DF
03E0:	FE	20	DF	C3	80	04	AF	32	1A	02	0E	05	11	CF	01	7E
03F0:	3E	20	CA	39	04	47	FE	22	CA	59	04	95	CA	39	04	9A
0400:	1E	02	B7	47	7E	C2	39	04	FE	3F	3E	95	CA	39	04	9A
0410:	FA	30	DA	1A	04	FE	3C	DA	39	04	D5	11	B7	00	E3	3E
0420:	D7	F3	1A	E6	7F	CA	36	04	BE	C2	60	04	1A	B7	F2	20
0430:	04	13	78	F6	80	F2	E1	7E	D1	23	12	13	0C	D6	3A	CA
0440:	47	04	FE	49	C2	4A	04	32	1A	02	D6	54	CF	EF	03	47
0450:	7E	7E	CA	6D	04	B8	CA	39	04	23	12	C3	C3	C3	50	04
0460:	E1	E5	04	EB	86	23	F2	64	04	EB	C3	02	04	DF	CD	0E
0470:	12	13	12	13	12	C9	05	2B	DF	C2	85	04	DF	CD	07	CA
0480:	21	07	01	06	01	CD	D8	04	FE	08	CA	76	04	FE	0D	CA
0490:	D7	0F	FE	18	CA	7C	04	FE	7F	D2	85	04	FE	04	CA	85
04A0:	04	00	00	00	00	00	4F	78	FE	48	3E	07	D2	B3	04	79
04B0:	71	23	04	DF	C3	85	04	C2	C4	0D	F1	F3	FE	F1	C5	4F
04C0:	04	3A	27	00	FE	48	CC	DC	07	3C	32	27	0F	F1	C5	4F
04D0:	F5	CD	09	F8	F1	C1	00	C9	CD	03	09	FE	1F	CD	02	F8
04E0:	06	E6	7F	FE	0F	C0	03	3A	17	02	2F	32	17	02	CD	61
04F0:	00	C0	C1	CD	85	03	F2	E3	CD	65	14	3E	20	11	88	05
0500:	E5	05	C5	CD	DC	07	F2	E3	CD	65	14	3E	20	11	88	05
0510:	B7	23	CA	F7	04	05	0D	E1	0E	05	05	05	05	05	05	05
0520:	1A	13	B7	F2	20	05	0D	E1	0E	05	05	05	05	05	05	05
0530:	DF	23	B7	F2	20	05	0D	E1	0E	05	05	05	05	05	05	05
0540:	02	E3	C2	47	05	09	F9	CD	69	09	09	09	09	09	09	09
0550:	E3	E5	2A	3B	02	E3	E5	2A	3B	02	E3	E5	2A	3B	02	E3

Продолжение
табл. 3.6

0560:	OD	13	E1	C5	D5	01	00	81	51	5A	7E	FE	A3	3E	01	C2
0570:	0D	05	D7	CD	46	09	06	CD	OD	13	E1	FE	C5	03	05	05
0580:	7C	2A	2A	02	FE	05	3A	05	03	CD	B7	02	00	02	7E	23
0590:	25	37	02	7E	F6	05	5E	05	05	0B	22	3B	02	07	11	89
05A0:	B6	D3	CA	C8	D6	70	01	07	07	05	C5	EB	02	FA	06	DO
05B0:	05	05	EB	21	70	01	09	23	44	00	C5	EB	02	FE	02	DO
05C0:	00	D5	C8	0A	0A	05	00	3F	3C	00	C8	CD	02	43	03	CB
05D0:	FE	20	CA	C2	C8	09	FE	3D	2A	02	7D	0A	02	0A	06	06
05E0:	22	4B	02	EB	0A	02	02	3F	00	02	AF	32	0C	0F	21	72
05F0:	F6	3D	00	22	37	02	02	00	0E	02	3F	CD	02	B5	03	CA
0600:	02	C2	02	F1	02	00	00	00	09	0A	20	C9	08	0C	3C	3C
0610:	0B	02	02	02	02	00	00	00	00	00	20	C9	08	0C	3C	3C
0620:	02	02	02	02	02	00	00	00	00	00	20	C9	08	0C	3C	3C
0630:	FE	4E	D2	5C	66	09	00	00	00	00	20	C9	08	0C	3C	3C
0640:	C9	D7	CD	CD	66	09	00	00	00	00	20	C9	08	0C	3C	3C
0650:	13	01	80	90	90	11	00	00	00	00	20	C9	08	0C	3C	3C
0660:	62	06	11	29	29	03	00	00	00	00	20	C9	08	0C	3C	3C
0670:	65	6B	19	29	29	03	00	00	00	00	20	C9	08	0C	3C	3C
0680:	62	06	11	29	29	03	00	00	00	00	20	C9	08	0C	3C	3C
0690:	93	05	7C	57	57	03	00	00	00	00	20	C9	08	0C	3C	3C
06A0:	D2	2B	02	22	22	03	00	00	00	00	20	C9	08	0C	3C	3C
06B0:	03	03	01	03	03	03	00	00	00	00	20	C9	08	0C	3C	3C
06C0:	02	02	02	02	02	03	00	00	00	00	20	C9	08	0C	3C	3C
06D0:	02	02	02	02	02	03	00	00	00	00	20	C9	08	0C	3C	3C
06E0:	02	02	02	02	02	03	00	00	00	00	20	C9	08	0C	3C	3C
06F0:	02	02	02	02	02	03	00	00	00	00	20	C9	08	0C	3C	3C
0700:	02	02	02	02	02	03	00	00	00	00	20	C9	08	0C	3C	3C
0710:	02	02	02	02	02	03	00	00	00	00	20	C9	08	0C	3C	3C
0720:	02	02	02	02	02	03	00	00	00	00	20	C9	08	0C	3C	3C
0730:	23	23	D1	D1	2A	2A	41	02	02	02	02	02	02	02	02	02
0740:	6B	62	F7	F7	CD	CD	1A	02	02	02	02	02	02	02	02	02
0750:	1C	1C	1C	1C	1C	1C	1C	02	02	02	02	02	02	02	02	02
0760:	47	06	06	06	06	06	06	02	02	02	02	02	02	02	02	02
0770:	62	06	06	06	06	06	06	02	02	02	02	02	02	02	02	02
0780:	07	07	07	07	07	07	07	02	02	02	02	02	02	02	02	02
0790:	D7	CA	DC	CA	07	07	07	02	02	02	02	02	02	02	02	02
07A0:	FE	2C	2C	2C	2C	2C	2C	02	02	02	02	02	02	02	02	02
07B0:	3A	19	02	02	02	02	02	02	02	02	02	02	02	02	02	02
07C0:	3A	19	02	02	02	02	02	02	02	02	02	02	02	02	02	02
07D0:	CA	0A	0A	0A	0A	0A	0A	02	02	02	02	02	02	02	02	02
07E0:	01	01	01	01	01	01	01	02	02	02	02	02	02	02	02	02
07F0:	01	01	01	01	01	01	01	02	02	02	02	02	02	02	02	02
0800:	01	01	01	01	01	01	01	02	02	02	02	02	02	02	02	02
0810:	01	01	01	01	01	01	01	02	02	02	02	02	02	02	02	02
0820:	47	06	06	06	06	06	06	02	02	02	02	02	02	02	02	02
0830:	77	77	77	77	77	77	77	02	02	02	02	02	02	02	02	02
0840:	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0850:	3A	19	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0860:	3B	0B	0B	0B	0B	0B	0B	02	02	02	02	02	02	02	02	02
0870:	28	0B	0B	0B	0B	0B	0B	02	02	02	02	02	02	02	02	02
0880:	36	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0890:	08	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
08A0:	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
08B0:	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
08C0:	06	13	13	13	13	13	13	02	02	02	02	02	02	02	02	02
08D0:	06	13	13	13	13	13	13	02	02	02	02	02	02	02	02	02
08E0:	06	13	13	13	13	13	13	02	02	02	02	02	02	02	02	02
08F0:	45	20	20	20	20	20	20	02	02	02	02	02	02	02	02	02
0900:	14	09	23	23	23	23	23	02	02	02	02	02	02	02	02	02
0910:	22	03	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0920:	C4	1A	0B	0B	0B	0B	0B	02	02	02	02	02	02	02	02	02
0930:	C4	1A	0B	0B	0B	0B	0B	02	02	02	02	02	02	02	02	02
0940:	12	13	0B	0B	0B	0B	0B	02	02	02	02	02	02	02	02	02
0950:	20	13	0B	0B	0B	0B	0B	02	02	02	02	02	02	02	02	02
0960:	89	05	07	07	07	07	07	02	02	02	02	02	02	02	02	02
0970:	1E	39	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0980:	1E	39	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0990:	D6	AB	DA	AA	AA	09	FE	03	03	03	03	03	03	03	03	03
09A0:	DA	DA	02	02	02	02	02	02	02	02	02	02	02	02	02	02
09B0:	DA	DA	02	02	02	02	02	02	02	02	02	02	02	02	02	02
09C0:	DA	DA	02	02	02	02	02	02	02	02	02	02	02	02	02	02
09D0:	CA	69	01	01	01	01	01	02	02	02	02	02	02	02	02	02
09E0:	01	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
09F0:	D1	0E	0A	FE	22	40	CA	50	0A	0A	0A	0A	0A	0A	0A	0A
0A00:	1E	0A	0A	FE	22	40	CA	50	0A	0A	0A	0A	0A	0A	0A	0A
0A10:	0D	06	AE	D2	D2	39	02	02	02	02	02	02	02	02	02	02
0A20:	CD	0A	0A	0A	0A	0A	0A	02	02	02	02	02	02	02	02	02
0A30:	0A	0A	0A	0A	0A	0A	0A	02	02	02	02	02	02	02	02	02
0A40:	06	06	06	06	06	06	06	02	02	02	02	02	02	02	02	02
0A50:	06	06	06	06	06	06	06	02	02	02	02	02	02	02	02	02
0A60:	06	06	06	06	06	06	06	02	02	02	02	02	02	02	02	02
0A70:	06	06	06	06	06	06	06	02	02	02	02	02	02	02	02	02
0A80:	06	06	06	06	06	06	06	02	02	02	02	02	02	02	02	02
0A90:	06	06	06	06	06	06	06	02	02	02	02	02	02	02	02	02
0AA0:	06	06	06	06	06	06	06	02	02	02	02	02	02	02	02	02
0AB0:	06	06	06	06	06	06	06	02	02	02	02	02	02	02	02	02

Продолжение
табл. 3.6

OACO:	3C	13	AF	32	19	02	D5	CD	C1	OE	D1	F7	F7	CD	C5	OE
OADO:	CD	DO	E1	E1	E3	55	E1	7B	B2	C8	7A	7A	2F	CD	12	BB
OAE0:	3C	DO	1D	1D	0A	9F	23	03	CA	D7	0A	0A	3F	CD	AF	3C
OAF0:	8F	C1	AO	C6	FF	2F	4F	05	1F	16	5A	5A	3F	CD	69	09
OB00:	02	D7	DA	06	7B	02	32	7A	2F	CD	9B	9B	3F	CD	83	09
OB10:	09	D7	C8	CF	2C	01	10	32	C5	02	06	06	3F	CD	46	CD
OB20:	39	04	DA	DO	02	AF	4F	19	0B	02	D7	DA	06	CD	CD	39
OB30:	04	DA	3F	0B	4F	D7	DA	35	0B	02	06	06	3F	CD	0B	06
OB40:	26	C2	4C	0B	3C	02	4F	32	0F	02	81	4F	D7	CD	35	02
OB50:	FE	E2	28	CA	9E	AF	32	35	02	E5	2A	47	78	CD	2A	45
OB60:	02	E7	7A	78	0B	0B	23	23	C2	6D	0B	06	69	CD	9B	0B
OB70:	09	C1	23	23	23	C3	61	0B	C5	01	06	06	00	CD	49	02
OB80:	0B	E3	E3	CD	9B	0B	D1	73	23	72	23	22	47	CD	2A	E5
OB90:	36	00	E7	C2	8F	0B	D5	0B	23	02	06	06	69	CD	2B	0B
OBA0:	09	C1	E7	16	00	00	D5	0B	41	06	C1	E3	E3	CD	2A	E5
OBBO:	3C	57	7E	FE	2C	02	CA	0B	CF	22	22	39	39	CD	18	0B
OBC0:	02	D5	2A	47	02	3E	A5	0B	2A	49	02	E7	E7	CD	0B	0B
OBDO:	7E	B9	23	C2	D8	0B	7E	B8	02	5E	23	56	56	CD	06	10
OBEO:	3A	16	02	B7	1E	12	C2	D8	02	F1	BE	CA	CA	CD	1E	10
OBFO:	02	E9	11	02	04	00	71	23	70	23	F1	32	32	CD	AA	01
OC00:	C3	D8	22	31	02	23	03	41	23	23	3A	16	16	CD	78	0B
OC10:	0B	00	CA	17	0C	01	03	71	23	70	23	F5	F5	CD	AB	13
OC20:	0B	E1	C1	05	C2	0A	0C	42	4B	EB	19	03	03	CD	16	23
OC30:	02	22	49	02	2B	36	00	E7	C2	34	31	02	73	CD	02	C2
OC40:	B7	3A	01	0C	6F	29	00	13	5E	23	56	23	23	CD	E7	0C
OC50:	7E	0C	23	01	00	00	16	E1	5E	23	56	23	23	CD	21	00
OC60:	EE	0B	E3	CD	AB	13	02	D7	0C	C9	2A	49	49	CD	2A	2A
OC70:	29	C1	09	EB	2A	39	02	B7	D7	C9	9A	41	41	CD	21	00
OC80:	00	39	3A	19	02	B7	02	CA	0C	C1	0E	41	41	CD	0C	0C
OC90:	41	02	EB	2A	2F	02	7D	93	4F	7C	9A	41	41	CD	0C	0C
OCA0:	19	02	73	06	90	C3	06	12	3A	27	00	00	00	CD	1A	0B
OCBO:	CD	10	0D	01	F9	06	CD	D5	CD	02	0B	CF	CF	CD	10	0B
OCC0:	CD	69	09	CF	29	CF	09	E3	44	02	E3	0B	0B	CD	0D	0D
OCDO:	D5	CD	E3	0A	CD	69	09	E3	F7	D1	F7	E1	E1	CD	2B	2B
OCE0:	D8	2B	E7	E7	D5	1E	02	D8	02	02	CD	F7	F7	CD	66	66
OCFO:	09	2B	D7	C2	D0	02	E1	D1	C1	71	E1	1E	1E	CD	73	73
OD00:	E1	C9	E5	2A	3B	02	E3	D1	E5	E1	CO	1E	1E	CD	08	02
OD10:	CF	AO	3E	80	32	35	02	B6	47	CD	0B	0B	0B	CD	09	09
OD20:	09	02	70	14	CD	CD	4F	CD	0D	0D	0E	01	01	CD	7E	7E
OD30:	23	23	CD	AA	0D	0D	E1	F7	02	C1	CD	46	46	CD	84	84
OD40:	0E	D1	C9	AA	0D	0D	21	2B	02	E5	77	23	23	CD	2B	2B
OD50:	06	23	CD	0E	0E	0E	22	22	0C	B7	05	63	63	CD	65	65
OD60:	0D	23	CD	0E	0D	0D	22	22	0C	B7	05	63	63	CD	46	46
OD70:	0E	E7	D4	2F	11	13	E7	02	2A	02	02	22	22	CD	01	01
OD80:	32	E7	02	CD	1C	1D	0B	0B	1D	0B	02	22	22	CD	02	02
OD90:	7E	C9	23	CD	4F	19	0B	CD	CD	CD	10	13	13	CD	2A	2A
ODAO:	DE	0F	0B	CC	E3	E3	07	03	C1	0E	0B	0E	0E	CD	0A	41
ODBO:	02	2F	2A	2F	02	2F	4F	F1	06	1A	23	02	02	CD	22	22
ODCO:	0F	02	2A	2F	F1	02	09	09	09	09	09	02	02	CD	01	01
ODDO:	2D	02	2A	1B	02	2A	02	02	02	02	02	02	02	CD	02	02
ODE0:	0D	1E	02	2A	47	12	02	02	02	02	02	02	02	CD	41	41
ODEO:	02	02	02	2A	47	02	02	02	02	02	02	02	02	CD	02	02
OE00:	32	0E	13	78	0B	0B	02	02	02	02	02	02	02	CD	02	02
OE10:	09	09	09	78	0B	0B	02	02	02	02	02	02	02	CD	02	02
OE20:	09	09	09	78	0B	0B	02	02	02	02	02	02	02	CD	02	02
OE30:	09	09	09	78	0B	0B	02	02	02	02	02	02	02	CD	02	02
OE40:	09	09	09	78	0B	0B	02	02	02	02	02	02	02	CD	02	02
OE50:	09	09	09	78	0B	0B	02	02	02	02	02	02	02	CD	02	02
OE60:	09	09	09	78	0B	0B	02	02	02	02	02	02	02	CD	02	02
OE70:	09	09	09	78	0B	0B	02	02	02	02	02	02	02	CD	02	02
OE80:	09	09	09	78	0B	0B	02	02	02	02	02	02	02	CD	02	02
OE90:	09	09	09	78	0B	0B	02	02	02	02	02	02	02	CD	02	02
OEA0:	09	09	09	78	0B	0B	02	02	02	02	02	02	02	CD	02	02
OEB0:	09	09	09	78	0B	0B	02	02	02	02	02	02	02	CD	02	02
OEC0:	09	09	09	78	0B	0B	02	02	02	02	02	02	02	CD	02	02
OED0:	09	09	09	78	0B	0B	02	02	02	02	02	02	02	CD	02	02
OEEO:	47	09	22	2F	02	02	E1	C9	01	AB	OC	C5	06	CD	57	57
OF00:	7E	C3	AB	0C	3E	01	CD	43	0D	4F	03	0F	0F	CD	02	02
OF10:	C1	C3	73	0D	CD	9F	0F	0F	0F	0F	0F	0F	0F	CD	0F	0F
OF20:	78	11	0E	00	C5	CD	AA	0D	0D	0D	0D	0D	0D	CD	66	66
OF30:	68	06	00	09	44	0D	0D	0D	0D	0D	0D	0D	0D	CD	C5	C5
OF40:	0E	C3	73	0D	CD	9F	0F	0F	0F	0F	0F	0F	0F	CD	7E	7E
OF50:	CD	A2	0F	C5	1E	0F	FE	FE	FE	FE	FE	FE	FE	CD	0F	0F
OF60:	CB	29	F1	E3	01	1A	0F	0F	C5	3D	BE	0F	0F	CD	91	91
OF70:	BB	47	D8	43	C9	0F	0F	0F	0F	0F	0F	0F	0F	CD	0C	0C
OF80:	CD	AC	0F	D3	00	C9	0F	0F	0F	0F	0F	0F	0F	CD	CA	96
OF90:	0F	2C	2C	CD	B9	0F	0F	0F	0F	0F	0F	0F	0F	CD	C9	0F
OFA0:	CF	29	C1	D1	C5	43	0F	0F	0F	0F	0F	0F	0F	CD	0F	32
OFBO:	9F	32	84	06	2B	D7	78	09	09	09	09	09	09	CD	06	7A
OFDO:	B7	C2	5C	06	19	44	72	09	09	09	09	09	09	CD	23	23
OFEO:	23	CD	F8	06	00	00	0F	0F	0F	0F	0F	0F	0F	CD	F1	F1
OFF0:	C9	4F	F8	CD	0C	0C	0F	0F	0F	0F	0F	0F	0F	CD	0F	0F
1000:	CD	EE	0F	E7	C2	0E	0F	10	0F	0F	0F	0F	0F	CD	1A	4D
1010:	CD	EE	0F	E7	C2	0E	0F	10	0F	0F	0F	0F	0F	CD	1A	4D

[illegible]

1580:	1F	E1	22	4F	02	E1	22	4D	02	DC	4F	15	CC	EA	12	D5
1590:	C3	CD	7E	11	C1	D1	CD	BC	02	CD	F2	12	01	38	81	D5
15A0:	3B	AD	CD	BC	11	BC	50	02	FE	88	D2	AB	01	CD	92	13
15B0:	C6	80	C6	02	DA	F5	12	F5	05	6D	11	CD	12	10	CD	B3
15C0:	11	F1	C1	D1	F5	CD	7F	CD	10	EA	12	21	73	15	CD	09
15D0:	16	77	00	00	C1	88	00	BC	00	08	40	2E	94	74	70	4F
15E0:	1E	11	0E	02	88	7A	00	AO	00	7C	50	2A	AA	7E	FF	11
15F0:	7F	7F	00	00	80	81	00	00	00	81	CD	F2	12	BA	11	11
1600:	D5	E5	CD	OD	13	CD	13	CD	11	CD	F2	12	CD	23	CD	FF
1610:	12	03	CD	C1	D1	3D	10	BC	D5	F5	E5	FA	BC	11	E1	CD
1620:	10	10	E5	CD	82	10	10	E1	C3	16	EF	16	7E	11	21	5C
1630:	16	CD	FF	12	C8	01	35	98	11	7A	44	4C	BC	01	01	28
1640:	16	11	46	B1	CD	82	10	10	OD	13	7B	19	4F	11	80	2B
1650:	46	36	80	CD	D3	10	21	SC	16	C3	19	13	52	C7	4F	80
1660:	21	A6	16	CD	73	10	CD	F2	12	01	49	83	DB	0B	0F	CD
1670:	02	13	C1	D1	AA	1A	12	CD	F2	12	CD	92	C1	CD	D1	10
1680:	7F	10	10	AA	FA	16	79	EF	16	49	73	10	7F	05	70	CD
1690:	EF	B7	F5	F4	C3	FA	12	AA	16	81	CD	00	00	05	12	BA
16A0:	21	1E	86	64	C3	FA	99	56	34	23	87	EO	5D	A5	86	DA
16B0:	D7	1E	86	64	C3	FA	12	56	34	23	87	EO	5D	A5	86	DA
16C0:	0F	4E	83	CD	F2	12	12	66	16	C1	E1	4F	FC	1A	12	CD
16D0:	02	13	CD	FE	81	C3	18	16	01	FC	81	51	FC	1A	12	CD
16E0:	05	02	FE	81	FD	F3	16	CD	15	FC	81	51	FC	1A	12	CD
16F0:	79	10	10	02	FD	16	78	FE	15	A6	74	16	C9	09	D7	3D
1700:	3B	78	05	6E	84	7B	78	FE	15	A6	74	16	C9	09	D7	3D
1710:	5A	7D	C8	00	91	7E	7E	FE	4C	7E	6C	AA	AA	66	00	0F
1720:	00	81	00	00	EF	CD	00	06	1A	C3	AB	CD	41	72	C3	AB
1730:	CD	49	06	C3	67	10	22	49	06	06	EB	CD	41	72	C3	AB
1740:	0C	0F	AF	32	00	00	22	49	06	06	EB	CD	41	72	C3	AB
1750:	CD	09	F8	C3	49	17	21	0D	0A	2A	4D	69	6B	32	2F	19
1760:	38	30	2A	20	42	41	32	58	5A	20	D2	5C	06	36	00	99
1770:	CF	2C	CD	B9	0F	32	58	5A	20	D2	5C	06	36	00	99	99
1780:	CF	2C	CD	B9	0F	32	58	5A	20	D2	5C	06	36	00	99	99
1790:	C0	EF	16	00	FF	3A	57	19	B7	CA	A1	F7	19	38	19	36
17A0:	17	16	00	3A	57	19	00	00	00	1A	AF	77	23	12	2C	CD
17B0:	80	E2	C9	E5	21	00	00	00	00	0F	AF	77	23	12	2C	CD
17C0:	FE	80	C2	BA	17	E1	00	00	00	0F	AF	77	23	12	2C	CD
17D0:	B9	FE	32	5C	19	CF	2C	55	3A	0F	0F	32	5C	19	3F	19
17E0:	FE	80	32	5C	19	CF	2C	55	3A	0F	0F	32	5C	19	3F	19
17F0:	92	32	5C	19	E3	AF	1A	79	34	0F	32	5C	19	3F	19	36
1800:	16	07	57	21	02	1A	79	34	0F	32	5C	19	3F	19	36	36
1810:	E6	FE	01	06	00	02	1A	79	34	0F	32	5C	19	3F	19	36
1820:	04	0F	32	5C	19	7F	1A	79	34	0F	32	5C	19	3F	19	36
1830:	04	0F	32	5C	19	7F	1A	79	34	0F	32	5C	19	3F	19	36
1840:	21	0F	32	5C	19	7F	1A	79	34	0F	32	5C	19	3F	19	36
1850:	B9	0F	32	5C	19	7F	1A	79	34	0F	32	5C	19	3F	19	36
1860:	B9	0F	32	5C	19	7F	1A	79	34	0F	32	5C	19	3F	19	36
1870:	B7	F2	8D	18	01	CF	C6	01	57	3E	FE	32	4F	19	BA	51
1880:	A8	18	43	5A	50	3A	50	19	32	4E	19	3A	4F	19	BA	51
1890:	A8	18	43	5A	50	3A	50	19	32	4E	19	3A	4F	19	BA	51
18A0:	19	AF	32	50	19	32	4F	19	78	1F	4F	06	06	19	B8	54
18B0:	45	18	2A	54	19	3E	3F	19	67	3A	50	19	85	F2	4E	19
18C0:	3A	51	19	84	32	55	19	7A	81	4F	04	78	B9	F2	4E	19
18D0:	79	93	4F	2A	54	19	3A	4E	19	19	04	50	19	3A	4E	19
18E0:	84	0F	32	55	19	D5	CD	DD	17	D1	C1	C3	03	18	E5	OF
18F0:	3E	AF	CD	EE	OF	2D	C2	F2	18	E1	E5	E6	CD	77	FE	CD
1900:	0E	D3	C3	FD	OF	32	4D	02	02	9E	03	06	03	0F	FE	CD
1910:	3E	FE	D3	CA	23	19	06	04	04	03	03	03	03	0F	FE	D3
1920:	C2	17	19	05	C2	19	06	21	4D	03	02	03	02	E1	OF	BE
1930:	C2	17	19	2A	43	02	06	03	3E	08	08	08	08	CD	FF	BB
1940:	02	7E	B7	23	C2	36	19	05	C2	38	19	C3	51	10	00	00
1950:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1960:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1970:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1980:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1990:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
19A0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
19B0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
19C0:	72	61	7A	72	61	62	6F	74	61	6E	6F	20	64	6C	71	20
19D0:	76	75	72	6E	61	6C	61	20	72	61	64	6F	64	6C	6D	6F
19E0:	73	6B	77	61	20	31	39	38	34	20	67	6F	64	6C	22	00
19F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1A00:	0F	0A	FE	22	CA	50	0D	FE	A2	CA	F9	09	0A	CA	16	CD
1A10:	0D	D6	AE	D2	40	0A	0F	FE	28	CA	F9	09	0A	CA	16	CD
1A20:	CD	78	0B	2A	39	02	E5	CD	EA	12	CD	69	CC	E1	12	C9
1A30:	0A	09	E5	EB	22	4D	02	FE	19	02	B7	09	FF	CD	C9	75
1A40:	06	00	07	4F	C5	D7	79	FE	29	DA	E5	0A	EB	28	B9	00
1A50:	06	00	07	4F	C5	D7	79	FE	29	DA	E5	0A	EB	28	B9	00
1A60:	EB	E3	C3	6D	0A	0A	CD	16	0A	E3	11	2A	0A	01	43	00

Интерпретатор имеет два основных режима работы: непосредственной интерпретации и интерпретации программы, хранящейся в памяти (далее для краткости будем называть режимы: первый — непосредственным, а второй — программным).

Непосредственный режим работы позволяет использовать микроЭВМ для выполнения вычислений без написания программы, т. е. примерно так же, как это делается с помощью калькулятора для научно-технических расчетов. Например, необходимо рассчитать мощность, рассеиваемую на коллекторе транзистора выходного каскада. Для этого справедливо соотношение:

$$P_k = E_k^2 / 4\pi^2 R_n,$$

где E_k — напряжение на коллекторе, R_n — сопротивление нагрузки.

Пусть, например, $E_k = 12\text{В}$, $R_n = 8\text{ Ом}$. Если теперь набрать на клавиатуре следующее выражение:

```
PRINT 12^2/(4*(3.14)^2*8),
```

то после нажатия на клавишу ВК на экране дисплея прочтем ответ: 0,456408.

Обратите внимание, что в Бейсике для обозначения операции возведения в степень используется символ «^», вместо запятой в десятичных дробях — точка, а знак умножения — «*».

Для указания последовательности вычислений используются круглые скобки. Если эти вычисления необходимо произвести несколько раз (для разных значений E_k и R_n), то можно написать простую программу (табл. 3.7 пример 1). Но прежде чем подробно обсуждать работу этой программы, познакомимся с некоторыми терминами и понятиями.

Программа на Бейсике состоит из последовательности пронумерованных строк. Строкам принято присваивать номера с интервалом, равным 10. В дальнейшем это может оказаться полезным, если понадобится вставить в программу несколько дополнительных строк. Номера строк могут быть любыми — от 0 до 65529. Каждая строка программы может состоять из одного или нескольких операторов, выполняющих над различными операндами строго определенные действия. Если операторов несколько, то они отделяются друг от друга символом «:» (двоеточие). В качестве операндов служат выражения, составленные из констант и переменных. В Бейсике существует два типа констант: числовые и символьные. Числовые константы — это любые десятичные числа от $-1,7 \cdot 10^{38}$ до $+1,7 \cdot 10^{38}$, символьные — последовательность любых отображаемых символов, заключенная в кавычки, например:

«МОСКВА»; «КП350»; «РЕЗУЛЬТАТ-» — символьные константы;

220; —3801; 3.14; —3.003E — 03; 8E12 — числовые константы. Две последние константы заданы в экспоненциальной форме, на что указывают буква E, за которой следует знак, и величина порядка (для положительных чисел знак «+» можно опускать). Точность задания констант — шесть десятичных цифр.

Соответственно типам переменных в программах на Бейсике используются переменные также двух типов: числовые и символьные. Обращение к переменным производится по имени, которое состоит из одного или двух символов, первый из них обязательно должен быть буквой латинского алфавита, а второй — буквой латинского алфавита или цифрой. Символьные переменные после имени содер-

Таблица 3.7

ПРИМЕР 1

```

-----
10 REM РАСЧЕТ МОЩНОСТИ
20 PI=3.14159
30 PRINT "ЗНАЧЕНИЕ-НАПРЯЖЕНИЯ?"
40 INPUT EK
50 PRINT "СОПРОТИВЛЕНИЕ НАГРУЗКИ?"
60 INPUT RN
70 PK=EK^2/(4*(PI)^2*RN)
80 PRINT "РАСSEИВАЕМАЯ МОЩНОСТЬ =" ; PK
90 INPUT "ПРОДОЛЖИМ (Д ИЛИ Н) " ; X$
100 IF X$="Д" THEN 330
110 STOP

```

ПРИМЕР 2

```

-----
ПУСТЬ А$="ТЕЛЕ", В$="ФОН", С$="ВИЗОР",
ТОГДА: М$=А$+В$ = "ТЕЛЕФОН"
        Л$=А$+С$ = "ТЕЛЕВИЗОР"
        В$(>)С$

```

ПРИМЕР 3

```

-----
10 DIM A(30), B(15,15), C2(3,3,3)
20 DIM P$(5), R$(7,4)

```

ПРИМЕР 4

```

-----
10 DATA 12,865,"КП103М","К1810ВМ86"
20 DATA "ТРАНЗИСТОР", "МИКРОСХЕМА"

```

ПРИМЕР 5

```

-----
10 INPUT A1
20 INPUT A2,A3,A4
30 INPUT B$, E2

```

ПРИМЕР 6

```

-----
40 INPUT "ВАШЕ ИМЯ " ; I$

```

ПРИМЕР 7

```

-----
10 PRINT "ПРОГРАММА РАСЧЕТА УСИЛИТЕЛЯ"
20 PRINT "КОЛИЧЕСТВО = " ; A6
30 PRINT A,B,S4
40 PRINT 45*A2+S3
50 PRINT A1$,A2$;N7

```

ПРИМЕР 8

```

-----
10 CUR 25,15
20 PRINT "ГРАФИК #5"

```

ПРИМЕР 9

```

-----
10 FOR I=0 TO 10 STEP 2
20 PRINT I;
30 NEXT I

```

ПРИМЕР 10

```

-----
10 FOR I=10 TO 0 STEP -2
20 PRINT I;
30 NEXT I

```

ПРИМЕР 11

```

-----
IF X=0 THEN 1000
IF A=0 THEN B=1
IF A=5 AND C=7 THEN GOSUB 200
IF Z=0 OR F=0 THEN L=4:GOTO 22
IF B$="НЕТ" THEN PRINT "ВЕРНО"
IF K$="ДА" OR K$="Д" THEN STOP

```

ПРИМЕР 12

```

-----
10 CLS
20 FOR I=0 TO 60
30 PLOT 50+I,I,1
40 NEXT I

```

ПРИМЕР 13

```

-----
10 CLS
20 PLOT 0,0,1
30 LINE 40,40
40 STOP

```

ПРИМЕР 14

```

-----
10 CLS
20 PLOT 20,20,1
30 LINE 20,50:LINE 50,50
40 LINE 50,20:LINE 20,20
50 STOP

```

жат знак «\$», например:

A; A8; K — допустимые имена числовых переменных;

A\$; A1\$, C3\$ — допустимые имена символьных переменных.

Группе переменных одного типа может быть присвоено общее имя. Такие переменные называют переменными с индексами или массивами переменных (мы будем пользоваться термином «массив»).

Для обращения к каждой отдельной переменной в массиве используют один или несколько индексов. Наименьшее значение индекса равно 0, а наибольшее определяется размером массива. Если индекс один, то говорят, что массив одномерный, два — двумерный и т. д. Имена массивов подчиняются тем же правилам, что и имена переменных. Индексы необходимо указывать в круглых скобках после имени массива. Разрешено использование как массивов числовых переменных, так и массивов символьных переменных.

Наименьшее значение индекса массива равно 0, например:

AB(4) — пятый элемент одномерного массива AB;

LS(3,8) — элемент, стоящий на пересечении четвертой строки и девятого столбца двумерного массива LS;

K\$ (5) — шестой элемент символьного одномерного массива K\$.

При работе программы для каждого массива в памяти ЭВМ резервируется соответствующая область. Массив перед использованием должен быть описан с помощью оператора DIM.

Переменные и константы служат операндами в выражениях языка Бейсик. Кроме них в выражения входят также знаки операций, скобки и имена функций. Обращение к функциям происходит по их имени, аргумент при этом указывается в круглых скобках после имени. Все выражения можно разделить на четыре типа: арифметические, символьные, отношения и логические. В табл. 3.8 приведены знаки операций, принятые в Бейсике. Числовые переменные и константы могут принимать участие в выражениях любого типа. Для символьных переменных и констант разрешены только операции отношения и конкатенации (слияния), обозначаемые знаком «+» (см. пример 2 табл. 3.7).

Таблица 3.8

Знак операции	Операция
	<i>Арифметическая</i>
+	Сложение
-	Вычитание
*	Умножение
/	Деление
^	Возведение в степень
	<i>Отношения</i>
>	Больше
<	Меньше
=	Равно
<>	Не равно
>=	Больше или равно
<=	Меньше или равно
	<i>Логическая</i>
NOT	Отрицание (НЕ)
AND	Логическое умножение (И)
OR	Логическое сложение (ИЛИ)

При вычислении результата выражения все операции производятся в определенном порядке. Порядок вычисления зависит от приоритета операций. Использование круглых скобок позволяет изменить порядок вычисления выражений интерпретатором.

Во всех алгоритмических языках одним из основных операторов является оператор присваивания. В Бейсике таковым является оператор LET. Например, запись LET B1=12 означает, что переменной (или константе) B1 присваивается значение 12. В описываемом интерпретаторе слово LET не используется, и поэтому необходимо присвоение значений записывать так: B1=12.

Вернемся к примеру 1 табл. 3.7. В строке 20 константе PI (число π) присваивается значение 3.14156. В строках 30, 50 и 80 записаны операторы печати, поэтому на экран будут выведены соответствующие сообщения. В строках 40, 60 и 90 записаны операторы ввода, в результате выполнения которых в первых двух случаях организуется ввод числовых значений, в последнем — символьного значения.

Введенное с клавиатуры слово присваивается символьной переменной X\$. В строке 100 использован условный оператор, который в случае выполнения условия передает управление на начало программы (строка 20). В противном случае будет выполнен оператор STOP, прекращающий выполнение программы и переводящий интерпретатор в непосредственный режим работы.

Мы уже говорили, что интерпретатор реализует также функции редактора текстов и отладчика. Для ввода текста новой программы необходимо сначала задавать ему соответствующий режим работы — директивой NEW, а затем с клавиатуры набрать программу (не забывая набирать вначале каждой строки ее номер). Для исправления неверно набранных символов служит клавиша «←» (так же, как и в Мониторе). Ввод каждой строки заканчивают нажатием на клавишу BK. Если нажать на клавишу CTR, то управление передается программе Монитор и для повторного запуска интерпретатора в работу необходимо воспользоваться директивой Монитора G0. Перезапуск интерпретатора не приводит к потере текста ранее набранной программы. Клавиша «→» служит для отмены всей набранной строки.

Просмотреть текст программы можно с помощью директивы LIST. Если необходимо удалить какую-либо строку программы, достаточно просто набрать ее номер и нажать клавишу BK, а чтобы вставить строку в середину программы — набрать любой номер строки, попадающий в интервал между двумя соседними, затем саму строку и нажать клавишу BK. Интерпретатор не реагирует на пробелы, если только они не стоят внутри символьных констант. Это создает определенные удобства, так как при использовании пробелов для выделения операторов улучшается читаемость программы (хотя несколько увеличивается ее объем). Для исправления строки программы, содержащей ошибки, набирают эту строку заново. После нажатия на клавишу BK вновь набранная строка встанет на место старой.

Отладка программы на Бейсике производится также с помощью интерпретатора. Для этой цели использована возможность перевода интерпретатора из программного режима в непосредственный при выполнении оператора STOP или одновременном нажатии на клавиши UC и C. В непосредственном режиме можно просмотреть и при необходимости модифицировать (изменить) значения переменных, снова просмотреть текст программы по директиве LIST и затем продолжить выполнение программы с того места, где она была прервана.

Перейдем теперь к подробному описанию используемого варианта интерпретатора языка Бейсик. Прежде всего рассмотрим системные директивы интерпретатора. Отметим только, что далее по тексту фигурные скобки будут означать, что данные параметры в конкретной директиве могут отсутствовать. В этом случае используются так называемые *соглашения по умолчанию*.

Директивы языка Бейсик

Директива NEW подготавливает интерпретатор для ввода новой программы с клавиатуры дисплея, при этом всем числовым переменным присваивается значение 0, а символьным — «пустая строка». Текст программы, набранный ранее, стирается.

Директива RUN[N] служит для запуска программы со строки с номером N. Если номер строки отсутствует, то работа программы начинается со строки с наименьшим номером. Всем переменным присваивается значение 0 или «пустая строка».

Директива LIST[N] инициирует распечатку текста программы, находящейся в ОЗУ, начиная со строки с номером N. Если номер отсутствует, то распечатка начинается со строки с наименьшим номером.

Директива CONT позволяет продолжить выполнение программы с того места, где она была прервана нажатием на клавиши УС и С или при выполнении оператора STOP. Эта директива — одно из основных средств отладки программ на Бейсике. Если после останова текст программы изменялся или было выдано сообщение интерпретатора о синтаксической ошибке, то продолжение выполнения программы по директиве CONT невозможно, о чем интерпретатор выдает на экран дисплея соответствующее сообщение.

Директива MSAVE [имя] позволяет переписать программу на Бейсике из ОЗУ на магнитную ленту. Имя программы, если оно указано, может содержать один символ латинского алфавита или одну цифру. Допускается не указывать имя программы. В этом случае и при последующем вводе программы имя указывать также не надо.

Директива MLOAD [имя] предназначена для загрузки программ на Бейсике с магнитной ленты в ОЗУ. Если имя программы указано, то происходит поиск данной программы, если нет — в ОЗУ загружается первая же встретившаяся на ленте программа на Бейсике, записанная без указания имени.

Операторы языка Бейсик

В Бейсике имеется ряд операторов, которые можно разделить на две группы — выполняемые и невыполняемые. Невыполняемые операторы — это операторы описания, к ним относятся следующие.

Оператор REM служит для вставки в текст программы комментариев. Он не влияет на выполнение программы, так как на текст, стоящий в строке за этим оператором, интерпретатор не реагирует (см. пример 1, табл. 3.7).

Оператор DIM предназначен для описания массивов, используемых в программе. Массив можно не описывать, если его размерность не превышает 10. Одним оператором DIM можно описать сразу несколько массивов (см. пример 3, табл. 3.7). Наименьшее значение индекса равно 0. Массив должен быть описан до его использования в программе, иначе вступает в силу описание по умолчанию.

Оператором DATA можно описывать данные непосредственно в программе. Значения данных присваиваются переменным программы с помощью оператора READ. Программа может содержать любое число операторов DATA и распо-

лагаться они могут в любом месте программы независимо от положения операторов READ. Оператором DATA могут быть описаны любые данные, как числовые, так и символьные (см. пример 4, табл. 3.7).

Все данные, описанные с помощью оператора DATA, образуют блок данных. Начало блока — данные, описанные самым первым в программе оператором DATA, конец — данные, описанные последним оператором DATA. Данные из блока можно считать, воспользовавшись оператором READ. Они будут считаны последовательно, начиная с первого. После каждого обращения к блоку данных происходит перемещение на одну позицию внутреннего (для интерпретатора) указателя данных.

Существует еще один оператор описания, но с ним мы познакомимся в разделе, посвященном встроенным функциям Бейсика. В зависимости от назначения выполняемые операторы также могут быть разделены на ряд групп: ввода-вывода; управления ходом выполнения программы; организации циклов; условный; графические и связи с машинными ресурсами.

Оператор READ предназначен для чтения данных из блока данных и присвоения значений переменным программы. Запись этого оператора в программе выглядит так: READ X1, X2, ... XN, где XN — имена числовых или текстовых переменных. При описании данных программист обязан строго соблюдать соответствие между типом данных и переменными. При каждом выполнении оператора указатель данных смещается на одну позицию.

Оператор RESTORE служит для перемещения указателя данных в первую позицию, обеспечивая тем самым возможность повторного считывания данных из блока.

Оператор INPUT позволяет вводить данные с клавиатуры дисплея непосредственно при выполнении программы на Бейсике. Значения введенных данных присваиваются переменным, имена которых указывают вслед за оператором INPUT. Это могут быть как числовые, так и символьные переменные (см. пример 5, табл. 3.7).

При выполнении оператора INPUT на экране дисплея возникает символ «?». В ответ на этот вопрос с клавиатуры вводят данные, которые распечатываются на экране в строку сразу после этого символа.

Если оператором INPUT необходимо ввести несколько переменных, то после ввода очередного значения необходимо нажать клавишу «,». Ввод данных заканчивается нажатием клавиши ВК. Если после появления символа «?» сразу нажать клавишу ВК, то интерпретатор переходит из программного в непосредственный режим.

Перед списком переменных в операторе INPUT может стоять строка символов, заключенная в кавычки. В этом случае при выполнении оператора на экран дисплея сначала будет выведено указанное сообщение, а затем символ «?».

При вводе данных, пока еще не была нажата клавиша конца ввода ВК, для внесения исправлений можно пользоваться клавишей «←».

Оператор PRINT предназначен для вывода на экран дисплея значений переменных, сообщений и результатов вычислений. Если оператор использован без операнда, то это приводит к печати одной пустой строки. При наборе программы и в непосредственном режиме вместо слова PRINT, можно набрать символ «?», тогда при последующем просмотре текста программы по директиве

LIST вы увидите, что в тексте символ «?» автоматически заменен на слово PRINT.

Операндов, стоящих вслед за оператором PRINT, может быть несколько, и тогда их отделяют друг от друга разделителями. В качестве разделителей используют символы «,» и «;», причем при использовании первого символа под каждое выводимое значение отводится 14 позиций в строке, второй служит для компактной печати результатов.

Для печати чисел используются три формы записи: в виде целого числа, числа с десятичной точкой и в экспоненциальном виде. В любой форме печатается не более шести цифр.

Если после последнего операнда в операторе PRINT стоит разделитель, то при выполнении следующего оператора PRINT печать продолжается в той же строке. Если же разделитель не стоит, то печать начинается с новой строки. Для задания формы печати данных с помощью оператора PRINT в языке Бейсик предусмотрены специальные функции, рассматриваемые ниже (см. пример 7 табл. 3.7).

Оператор *CUR X, Y* служит для перемещения курсора в позицию с координатой X по горизонтали и координатой Y по вертикали. Начало отсчета — левый нижний угол экрана. Диапазон изменения координат курсора по горизонтали 0...63, по вертикали 0...24. Если после оператора CUR должен быть выполнен оператор печати PRINT, то вывод информации на экран начнется с позиции с координатами X и Y. После выполнения программы в центре экрана появится сообщение «график N5» (см. пример 8, табл. 3.7).

Оператор *CUR* позволяет создавать программы, полностью использующие возможности нашего дисплея. К таким программам можно отнести разнообразные игровые программы, экранные редакторы текстов, программы обработки информации, представленной в табличной форме, и многие другие.

Программа на языке Бейсик выполняется строка за строкой, в соответствии с их номерами. Однако имеется ряд операторов, меняющих естественный ход выполнения программы.

Оператор *GOTO N* — это оператор безусловной передачи управления на строку с номером N.

Операторы *GOSUB N* и *RETURN* служат для организации подпрограмм. Оператор *GOSUB N* служит для вызова подпрограммы, начинающейся со строки с номером N. Заканчиваться подпрограмма должна обязательно оператором *RETURN*. После его выполнения происходит возврат в то место основной программы, откуда произошел вызов подпрограммы. Допускается многократная вложенность подпрограмм, степень которой ограничена только объемом свободной памяти.

Операторы *ON X GOTO N1, N2,...* и *ON X GOSUB N1, N2,...* реализуют условную передачу управления на одну из строк программы, номер которой указан в списке, следующем за оператором.

При выполнении оператора сначала вычисляется значение выражения X, от результата берется целая часть, которая и служит указателем на номер строки в списке. Например, если результат выражения равен 1, то управление будет передано на строку № 1, двум — на строку № 2 и т. д. Если же результат выражения меньше 1 или больше, чем число номеров строк в списке, то выполняется

оператор, непосредственно следующий за оператором ON — GOTO или ON — GOSUB. Рассмотренный оператор часто называют переключателем. Действительно, его работа похожа на работу многопозиционного переключателя, коммутирующего прохождение сигнала по одному из возможных направлений. Оператор позволяет сократить текст программы в тех случаях, когда необходим анализ множества вариантов, программисту же необходимо так подобрать выражение X, чтобы при всех возможных значениях переменных, входящих в него, происходило переключение на строку с нужным номером.

Оператор STOP предназначен для прекращения выполнения программы и перевода интерпретатора в непосредственный режим. Этот оператор очень удобен при отладке программы, так как позволяет в программе создать контрольный останов. Выполнение прерванной программы можно продолжить, выдав интерпретатору директиву CONT. После выполнения оператора STOP на экран дисплея выводится сообщение: «СТОП в XX», где XX — номер строки, в которой произошел останов.

Программы, написанные на любых языках программирования, обычно содержат многократно повторяющиеся фрагменты — циклы. Для организации циклов в языке Бейсик имеются специальные операторы.

Оператор FOR X TO Y (STEP Z) — это оператор инициализации цикла, а оператор NEXT имя — оператор конца цикла. Все, что находится между ними, называют телом цикла. В операторе инициализации цикла X — выражение, задающее имя переменной цикла и присваивающее ей начальное значение; Y — выражение, определяющее конечное значение переменной цикла. Выражение Z определяет значение, на которое должна измениться переменная цикла (шаг цикла) после каждого выполнения оператора NEXT. Это выражение приводится только в том случае, если шаг цикла отличен от +1. После оператора NEXT помещается список имен переменных — параметров цикла. В частном случае этот список может быть пустым. Имена переменных в списке указываются через запятую. Проиллюстрируем сказанное примером 9 — табл. 3.7.

После выполнения этой программы на экране дисплея появится шесть значений: 0 2 4 6 8 10. В данном примере оператор, стоящий в строке 20, является телом цикла. Таким образом, работа оператора цикла заключается в следующем:

- задается начальное значение переменной цикла;
- выполняются операторы, входящие в тело цикла;
- приводится проверка достижения переменной цикла конечного значения;
- если конечное значение не достигнуто, то значение переменной цикла изменяется на величину, равную шагу цикла, и все перечисленные выше операции повторяются вновь;

если конечное значение достигнуто, то выполняется оператор, следующий непосредственно за оператором NEXT. Шаг цикла может принимать и отрицательное значение (см. пример 10, табл. 3.7). В этом случае последовательность выведенных на экран значений будет обратной: 10 8 6 4 2 0.

Заметьте! Операторы, входящие в тело цикла, в любом случае выполняются хотя бы один раз, так как проверка условия окончания производится в конце цикла.

Организовать циклическую работу программы можно и не пользуясь специальным оператором цикла (например, с помощью оператора GOTO и услов-

ного оператора, описываемого ниже), однако наличие такого оператора значительно упрощает разработку программ, освобождая программиста от необходимости проведения изменения переменной цикла и проверки его окончания.

Условный оператор IF X THEN — один из фундаментальных операторов, имеющих почти в любом алгоритмическом языке программирования высокого уровня. Работа его заключается в следующем:

проверяется выполнение условия X;

если X — истина, то выполняются операторы, стоящие в строке после слова THEN;

если X — ложь, то управление будет передано следующей строке программы.

Выражение X может включать проверку самых различных условий с использованием как операций отношения, так и логических операций. Операторы, стоящие после слова THEN, также могут быть различными. В частности, если необходимо выполнить оператор GOTO, то его название можно опустить и только указать номер строки программы, которой следует передать управление.

В условном операторе выражение X может быть арифметическим. В этом случае условие считается выполненным, если $X=0$. Пример 11, табл. 3. 7. иллюстрирует использование условного оператора.

Особенностью описываемого варианта интерпретатора языка Бейсик является наличие операторов, позволяющих формировать графические изображения на экране дисплея.

Оператор CLS предназначен для стирания информации с экрана и всегда должен выполняться прежде, чем начнется формирование графических изображений. В графическом режиме на экране дисплея возможно отображение 128 точек по горизонтали и 50 точек по вертикали. На рис. 3.1 показан формат экрана в графическом режиме. Точки адресуются в прямоугольной системе координат с началом в левом нижнем углу экрана.

Оператор *PLOT X, Y, Z* позволяет погасить (или высветить) точку с координатами X по горизонтали и Y по вертикали. Если последний операнд равен 1, то соответствующая точка засвечивается, 0 — гаснет. Программа «рисует» на экране прямую линию с начальными координатами $X=50, Y=0$ и конечными $X=110, Y=60$ (см. пример 12, табл. 3.7). (Обратите внимание на то, что диапазон изменения координат X в операторе CUR вдвое меньше, чем соответствующий диапазон в операторе PLOT.)

Оператор LINE X, Y позволяет вычерчивать или стирать отрезки прямых линий. Операнды X и Y задают координату конечной точки отрезка. Для задания координат начала отрезка, а также его вида (засветка или гашение) служит оператор PLOT (см. пример 13, табл. 3.7).

После выполнения программы на экране появится отрезок прямой линии с координатами начала 0,0 и координатами конца 40, 40.

Если следующий отображаемый отрезок

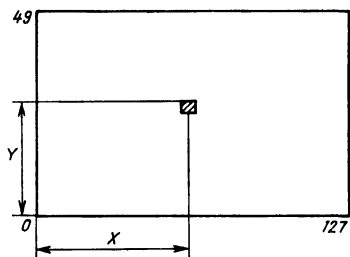


Рис. 3.1. Формат экрана в графическом режиме

зок должен начинаться там, где закончился предыдущий, то оператор PLOT необходим для задания координат начала только самого первого отрезка. Так, программа примера 14, табл. 3.7 выводит на экран дисплея изображение квадрата.

Используя описанные операторы, на экране можно создавать разнообразные изображения, в том числе и динамически изменяющиеся.

Среди операторов Бейсика особое место занимают операторы, позволяющие получить доступ к машинным ресурсам.

Оператор POKE X, Y позволяет записать в ячейку памяти, адрес которой задан выражением X, значение, равное результату выражения Y (оно должно находиться в диапазоне 0... 255 (0H — FFH)).

Так как в РК регистры периферийных БИС адресуются как ячейки памяти, то для записи информации в них может быть использован оператор POKE.

Оператор BEL X, Y предназначен для формирования различных звуковых сигналов. Параметр X (0... 255) задает высоту тона, а параметр Y (0... 255) — длительность звучания.

И наконец, еще один оператор, имеющийся в языке Бейсик, — CLEAR X предназначен для очистки памяти. Если параметр X не указан, то после выполнения оператора всем числовым переменным присваивается значение 0, а всем символьным — значение «пустая строка». Если же параметр X указывается, то в памяти выделяется дополнительный буфер размером X байт, предназначенный для хранения символьных переменных. В этом случае числовые переменные остаются без изменений. Выделение дополнительного буфера необходимо в том случае, если символьные переменные программы не помещаются в буфер, выделенный для них по умолчанию. Первоначальный размер этого буфера равен 50 байтам.

Функции в языке Бейсик

В языке Бейсик имеется ряд встроенных функций, которые позволяют значительно упростить написание некоторых программ. Название «встроенная» означает, что в интерпретаторе есть программа обработки данной функции. Существуют функции, работающие с числовыми и символьными данными, функции преобразования типов данных, а также функции обращения к таким машинным ресурсам, как содержимое ячеек памяти. В этом порядке и будем их рассматривать.

В выражениях обращение к функции всегда должно находиться в правой части, например: $A = \sin(X)$.

Таким образом, для вызова функции достаточно просто указать ее имя в выражении. В круглых скобках указывают аргумент. Результат работы функции (в данном случае значение синуса X) используется в дальнейшем в выражении для вычисления окончательного результата и присвоения полученного значения левой части выражения. Во всех описываемых ниже функциях в качестве аргумента могут выступать константы, имена переменных и выражения, содержащие, в свою очередь, обращения к встроенным функциям.

В табл. 3.9 перечислены встроенные функции языка Бейсик. Почти все хорошо знакомы читателю по курсу математики средней школы и только две последние функции требуют дополнительных разъяснений.

Таблица 3.9

Функция	Результат работы
SQR (X)	Корень квадратный из X
EXP (X)	Экспонента от X
LOG (X)	Натуральный логарифм от X
ABS (X)	Абсолютная величина X
	$ABS(X) = \begin{cases} X, & \text{если } X > 0 \\ 0, & \text{если } X = 0 \\ -X, & \text{если } X < 0 \end{cases}$
SGN (X)	Знак X
	$SGN(X) = \begin{cases} 1, & \text{если } X > 0 \\ 0, & \text{если } X = 0 \\ -1, & \text{если } X < 0 \end{cases}$
SIN (X)	Синус от X (X в радианах)
COS (X)	Косинус от X (X в радианах)
TAN (X)	Тангенс от X (X в радианах)
ATN (X)	Арктангенс от X (в радианах)
INT (X)	Целая часть от X
RND (X)	Случайное число от 0 до 1

Функция *INT* (X) возвращает в качестве результата наибольшее целое число, которое меньше или равно X. Например: *INT*(7.6) = 7; *INT*(-5,6) = -6; *INT*(3) = 3.

Функция *RND* (1) занимает особое место среди других функций. Прежде всего отметим, что аргумент этой функции всегда должен быть равен 1. Результат работы функции — случайное число больше 0, но меньше 1. С помощью этой функции можно писать на языке Бейсик разнообразные программы моделирования и как частный случай таких программ — игровые.

Если необходимо случайное число в другом диапазоне, например от 0 до 100, то достаточно в программе написать: $A = RND(1) * 100$, а чтобы сделать его целым — $A = INT(RND(1) * 101)$.

Тригонометрические функции требуют, чтобы аргумент был указан в радианах. Если по каким-либо причинам нужно указать угол в градусах, то перевод из одной меры угла в другую легко произвести и в программе, воспользовавшись формулой: $Y = X \cdot 3.14 / 180$, где Y, X — значения угла в радианах и градусах соответственно.

Как вы, наверное, уже заметили, в языке реализованы не все тригонометрические функции, однако это не должно вызывать затруднений, так как всегда можно воспользоваться известными тригонометрическими выражениями для определения одних функций через другие.

Язык Бейсик имеет развитые встроенные средства для обработки текстов, что выгодно отличает его от многих других языков программирования высокого уровня. Рассмотрим встроенные функции для работы с символьной информацией.

Результатом работы функции *LEN* (X\$) является число, равное количеству символов (длине) символьной переменной X\$, например, если X\$ = «Пароход», то *LEN* (X\$) = 7.

Функция $LEFT\$ (X\$,N)$ позволяет выделить строку символов длиной N начиная с крайнего левого символа, например: $LEFT\$ (X\$,3) = \text{«Пар»}$.

Функция $RIGHT\$ (X\$,N)$ делает то же самое, но начинается с крайнего правого символа, например: $RIGHT\$ (X\$, 3) = \text{«Ход»}$.

Функция $MID\$ (X\$,M,N)$ позволяет получить строку символов длиной N , начиная с позиции M . Отсчет позиций ведется слева направо, например: $MID\$ (X\$, 3,4) = \text{«РОХО»}$.

В программах на Бейсике часто возникает необходимость перевода данных из числового вида в символьный и наоборот. Для этого имеются две специальные встроенные функции.

В функции $STR\$ (X\$)$ аргументом является число или арифметическое выражение, а результат работы функции — строка, являющаяся символьным представлением данного числа. Таким образом, эта функция служит для преобразования числовых величин в символьный вид. При переводе учитываются знак числа (минус), десятичная точка и символ E , например: пусть $X=1$, тогда $STR\$ (X)$ равно «1».

Функция $VAL (X\$)$ предназначена для обратного преобразования данных из символьного вида в числовой, начиная с крайнего левого символа символьной переменной $X\$$. Если в строке встречается недопустимый символ, преобразование заканчивается. Если уже первый символ является недопустимым, то результат работы равен 0, например:

если $X = \text{«13 штук»}$, то $VAL (X\$) = 13$;

если $X = \text{«штук 13»}$, то $VAL (X\$) = 0$.

Мы знаем, что каждому символу соответствует определенный семиразрядный код. В Бейсике есть две встроенные функции для работы с кодами символов.

Функция $ASC (X)$ переводит код первого символа X в десятичное число, например:

если $X\$ = \text{«D»}$, то $ASC (X\$) = 68$.

Функция $CHR\$ (X)$ позволяет получить строку, состоящую из единственного символа, код которого равен X (аргумент функции не должен превышать 255), например: $CHR\$ (69) = \text{«Е»}$. Эту функцию удобно использовать для выдачи на экран дисплея различных управляющих символов, например: $PRINT CHR\$ (12)$ приведет к перемещению курсора в левый верхний угол экрана.

Для управления форматом печати результатов на экране дисплея служат следующие три функции.

Функция $POS (I)$, результатом работы которой является целое число, равное номеру позиции последнего отпечатанного символа в текущей строке, например: $PRINT \text{«ABCDE»}$, $POS (1) = 5$.

Функция $SPC (X)$ позволяет вставлять в печатаемую строку X пробелов (аргумент X не должен превышать 255). Например, в результате обработки строки:

$PRINT \text{«Транзистор»}; SPC (5); \text{«КТ315Г»}$,

будет напечатано:

ТРАНЗИСТОР КТ315Г (— пробел).

Функция TAB (X) — это функция горизонтальной табуляции, которая позволяет переместить курсор в заданную позицию в строке. Аргумент X и в этом случае не должен превышать 255. Он указывает, сколько позиций необходимо отступить от левого края строки. Например, после выполнения операторов

PRINT TAB (5); «А»; TAB (10); «В»

символ «А» будет напечатан в шестом знакоместе, а «В» — в 11-м.

Функция FRE (X) предусмотрена для определения объема свободной памяти, не занятой текстом программы и переменными при разработке больших программ, а также программы обработки текстов. Если аргумент этой функции число, то результатом будет количество свободных байтов в памяти, если аргумент символьное выражение, то результат — количество свободных байтов в буфере для символьных переменных. Например, после выполнения оператора PRINT FRE (0) на экран будет выведено количество свободных байтов в памяти.

Выше уже отмечалось, что интерпретатор позволяет разрабатывать программы управления различными объектами. Управляющие программы обычно содержат критичные по времени выполнения фрагменты и требуют также возможности непосредственного манипулирования с содержимым ячеек памяти и портов ввода-вывода. Следующие встроенные функции Бейсика и служат для этих целей.

Функция PEEK (X) — результатом ее работы будет десятичное число, равное содержимому ячейки памяти, адрес которой определен аргументом X.

Функция KEY (0) возвращает в качестве результата код символа, если на клавиатуре была нажата какая-либо клавиша или число 255 в противном случае. Использование этой функции позволяет на Бейсике создавать программы, в которых осуществляется динамическое управление поведением объектов на экране.

Функция USR (X) предназначена для организации связи программ, написанных на Бейсике, с подпрограммами в машинных кодах. Аргумент X — это адрес ячейки памяти, начиная с которой записана программа в машинных кодах. Поэтому если в выражении встретится обращение функции USR (X), то управление будет передано подпрограмме, расположенной по адресу X.

В конце подпрограммы обязательно должна стоять команда RET, после выполнения которой управление возвращается к программе на Бейсике. Результат работы подпрограммы (в виде одного байта) перед возвратом из подпрограммы помещается в аккумулятор. Функция USR (X) позволяет критичные по времени и специфике работы фрагменты алгоритма реализовывать на языке Ассемблера, а основную программу писать на языке Бейсик. Для передачи параметров и результатов в программы на Бейсике можно воспользоваться оператором POKE X и функцией PEEK(X).

Подпрограммы целесообразно размещать в защищенной области памяти, т. е. в той области, которую заведомо не использует интерпретатор. Если в функциях и операторах, работающих с адресами памяти, адрес превышает значение 32767 (7FFFH), то он должен указываться в виде отрицательного числа. Адресу FFFFH будет соответствовать —1, адресу FFFEH — —2 и т. д.

Кроме перечисленных встроенных функций в Бейсике имеется возможность вводить в программе определение новых функций и в дальнейшем обращаться

к ним в различных выражениях по имени. Определить функцию можно в любом месте программы с помощью оператора DEF, важно только, чтобы строка с этим определением была исполнена до обращения к функции. Имена всех функций должны начинаться с символов FN, за ними следуют один или два символа, ограничения на которые такие же, как и на имена переменных, например, FNA, FNХ допустимые имена, ES2, EKA — недопустимые имена.

Определим функцию FNCT(X) следующим образом:

10 DEF FNCT(X) = COS(X)/SIN(X).

Функция *FNCT* в соответствии с нашим определением — это тригонометрическая функция котангенса, не реализованная в виде встроенной в интерпретаторе.

Оператор DEF можно использовать только в программном режиме, и, кроме того, допускается определение функций только от одного числового аргумента. Параметр X, стоящий в операторе определения, является формальным, необходимым для обозначения функциональной зависимости. При обращении к функции вместо него указывается фактический аргумент, который заменяет формальный параметр, стоящий в правой части оператора присваивания. Например, после выполнения строки программы: 20 PRINT FNCT(2) на экране будет напечатано значение котангенса для угла, равного 2 рад.

Возможность определения функции программистом позволяет сократить текст программ и значительно улучшить их читаемость.

В интерпретаторе предусмотрен вывод информации на печатающее устройство. При нажатии клавиши JC+O вся выводимая на экран информация дублируется на печатающем устройстве. Повторное нажатие на эти клавиши запрещает вывод на печатающее устройство. Подпрограмма вывода на печатающее устройство получает код символа в регистре C микропроцессора. Она может быть размещена в области для подпрограмм в машинных кодах. В ячейках памяти 1854H, 1855H, 1856H необходимо поместить команду безусловного перехода (JMP ADR) на начало подпрограммы печати. В подпрограмме печати должно быть предусмотрено сохранение регистров в стеке и их восстановление после возврата из подпрограммы. Описанный способ выдачи данных на печатающее устройство был выбран из-за невозможности в универсальной программе учесть все варианты характеристик печатающих устройств и способов их подключения.

Сообщения об ошибках

Интерпретатор языка Бейсик позволяет в процессе выполнения программы обнаруживать и анализировать ошибки. Интерпретатор, естественно, не может обнаружить логические ошибки. Это может сделать только программист, так как только он знает, что должна делать программа. О каких же ошибках тогда идет речь?

Если ошибка обнаружена в непосредственном режиме, то на экран выводится сообщение:

XX ОШИБКА,

где XX — двузначный код ошибки.

Если ошибка обнаружена в программном режиме, то на экран выводится сообщение:

XX ОШИБКА В N,

где XX — код ошибки, а N — номер строки, где ошибка была обнаружена. После сообщения об ошибке появляется знак « \Rightarrow », означающий, что интерпретатор готов к приему директив и программист может внести изменения в программу и продолжить отладку.

Рассмотрим, какие ошибки обнаруживает интерпретатор.

ОШИБКА 01. В программе встретился оператор NEXT, для которого не был выполнен соответствующий оператор FOR.

ОШИБКА 02. Неверный синтаксис.

ОШИБКА 03. В программе встретился оператор RETURN без предварительного выполнения оператора GOSUB.

ОШИБКА 04. При выполнении программы не хватает данных для оператора READ, т. е. данных, описанных операторами DATA, меньше, чем переменных в операторах READ.

ОШИБКА 05. Аргумент функции не соответствует области определения данной функции. Например, отрицательный или нулевой аргумент функции $\text{SQR}(X)$, отрицательный аргумент у функции $\text{LOG}(X)$ и т. д.

ОШИБКА 06. Переполнение при выполнении арифметических операций. Результат любой операции не может быть больше $+1,7 \cdot 10^{38}$ или меньше $-1,7 \cdot 10^{-38}$.

ОШИБКА 07. Недостаточен объем памяти. Возможные причины: велик текст программы, слишком длинны массивы данных, вложенность подпрограмм и циклов больше нормы, слишком много переменных.

ОШИБКА 08. Нет строки с данным номером. Возникает при выполнении операторов перехода.

ОШИБКА 09. Индекс не соответствует размерности массива.

ОШИБКА 10. Повторное выполнение операторов DIM или DEF, описывающих массив или функцию, которые уже были описаны ранее.

ОШИБКА 11. Деление на нуль.

ОШИБКА 12. Попытка выполнить оператор DIM или DEF в непосредственном режиме.

ОШИБКА 13. Несовпадение типов данных. Возникает при попытке символической переменной присвоить числовое значение и наоборот.

ОШИБКА 14. Переполнение буферной области памяти, отведенной для хранения символьных переменных. Для расширения объема буфера служит директива CLEAR.

ОШИБКА 15. Длина символьной переменной превышает 255.

ОШИБКА 16. Выражение, содержащее символьные переменные, слишком сложно.

ОШИБКА 17. Невозможность продолжения выполнения программы по директиве CONT.

ОШИБКА 18. Попытка обратиться к функции, не описанной оператором.

Кроме сведений об этих ошибках, интерпретатор выдает еще три сообщения в случае неверного набора данных при выполнении оператора INPUT. ? ПОВТОРИТЕ ВВОД — указывает на ошибку в наборе данных.

Вместо числа Вы набрали строку символов.

? ЛИШНИЕ ДАННЫЕ — данных набрано больше, чем переменных в операторе INPUT. Лишние данные просто игнорируются.

?? — данных набрано меньше, чем переменных в операторе INPUT. Необходимо ввести недостающие данные.

3.3. Программирование на Бейсике

В предыдущем разделе мы познакомились с описанием интерпретатора языка Бейсик и разобрали ряд простейших программ. Опытный программист из простого перечисления набора операторов и функций языка может сделать вывод о его возможностях и представить себе круг задач, решаемых с его помощью. Этому способствует практика разработки программы на других языках и известные алгоритмы решения стандартных задач.

Для овладения техникой составления программ начинающему программисту целесообразно разбирать и анализировать программы, написанные другими, аналогично шахматисту, разбирающему и анализирующему партии, сыгранные мастерами. С основными приемами составления программ на Бейсике ознакомимся на конкретных примерах. Рассматриваемые небольшие программы иллюстрируют особенности интерпретатора Бейсика для ПК.

Программирование на Бейсике

Одно из основных достоинств языка Бейсик — возможность разработки программ в диалоговом режиме. Вы, например, можете начать отладку программы, набрав всего одну строку. Убедившись в том, что эта строка программы работает так, как Вы и предполагали, можно продолжить набор программы, если нет, то после анализа полученного результата можно быстро исправить обнаруженные ошибки. Кроме того, непосредственный режим работы интерпретатора предоставляет возможность проведения экспериментов по выявлению особенностей применения того или иного оператора или встроенной функции, что также способствует быстрому освоению языка.

При разработке программ Вы оцените преимущества диалогового режима, а пока попробуем сами составить программу, реализующую диалог с оператором. Начнем с постановки задачи. Допустим, необходимо разработать программу для проверки знаний математики у учащихся начальных классов. Сразу оговоримся, что описываемая программа нужна прежде всего для ознакомления с возможностями языка Бейсик. Реальные программы проверки знаний требуют более тщательной проработки с привлечением специалистов по обучению и психологии.

Начнем с составления сценария работы программы (умышленно пользуемся словом «сценарий», чтобы подчеркнуть диалоговый характер нашей будущей программы). Одним из возможных вариантов может быть такой. На экране появляется вопрос: КАК ТЕБЯ ЗОВУТ?

Ученик набирает на клавиатуре свое имя (например, АНТОН), и программа выводит на экран текст:

АНТОН, Я ХОЧУ ПРОВЕРИТЬ ТВОЕ ЗНАНИЕ МАТЕМАТИКИ. Я БУДУ ПРЕДЛАГАТЬ ПРИМЕРЫ, А ТЫ ПОСТАРАЙСЯ ИХ РЕШИТЬ.
ЕСЛИ ГОТОВ, НАЖМИ НА ЛЮБУЮ КЛАВИШУ.

После этого компьютер очищает экран и выводит в его верхней части следующую справочную информацию:

ПОМНИ!

ЕСЛИ ПРИ ОТВЕТЕ ОШИБЕШЬСЯ И СЛУЧАЙНО НАЖМЕШЬ НЕ ТУ КЛАВИШУ, ТО НАЖМИ КЛАВИШУ «←», А ЗАТЕМ НУЖНУЮ. ПОСЛЕ НАБОРА ОТВЕТА НАЖМИ КЛАВИШУ «ВК».

Этот текст остается на экране до конца работы программы, а в центре экрана появляется вопрос:

АНТОН, СКОЛЬКО БУДЕТ 5 ПРИБАВИТЬ 2 = ?

При верном ответе на экран выводится:

МОЛОДЕЦ, АНТОН!

РЕШИ ЕЩЕ ПРИМЕР.

ЕСЛИ ГОТОВ, НАЖМИ ЛЮБУЮ КЛАВИШУ.

Неверный ответ приводит к появлению на экране сообщения:
НЕПРАВИЛЬНО!

ТЕБЕ, АНТОН, НАДО ПОДУМАТЬ.

Если за две попытки учащийся так и не смог решить пример, то на экране появляется текст:

ПЛОХО, АНТОН! ТЫ НЕ СМОГ РЕШИТЬ ЭТОТ ПРИМЕР.

ВОТ ПРАВИЛЬНОЕ РЕШЕНИЕ:

5 ПРИБАВИТЬ 2 = 7.

РЕШИ ЕЩЕ ПРИМЕР.

ЕСЛИ ГОТОВ, НАЖМИ ЛЮБУЮ КЛАВИШУ.

Ученик должен решить десять примеров. В зависимости от результатов в конце работы на экране высвечивается одно из следующих сообщений:

МОЛОДЕЦ, АНТОН, ТЫ ЗНАЕШЬ МАТЕМАТИКУ НА 5!

МОЛОДЕЦ, АНТОН, ТЫ ЗНАЕШЬ МАТЕМАТИКУ НА 4!

АНТОН, ТЫ ЗНАЕШЬ МАТЕМАТИКУ НА 3!

АНТОН, ТЫ ПЛОХО ЗНАЕШЬ МАТЕМАТИКУ!

Полный текст программы, выполняющей описанные действия, приведен в табл. 3.10. Рассмотрим ее особенности. Прежде всего обратим внимание на то, как формируются текстовые сообщения, появляющиеся на экране дисплея. В соответствии с нашим сценарием возникает необходимость вывода на экран сообщений, в которых встречается имя человека, отвечающего на вопросы (символьная переменная I\$). Для этой цели используется разделить «;» (точка с запятой).

В строках 100 и 430 происходит вызов стандартной подпрограммы Монитора «ввод кода символа с клавиатуры», адрес которой равен F803H в шестнадцатиричной системе счисления. Это вызвано необходимостью приостановки выполнения программы до тех пор, пока не будет нажата любая клавиша. Значение, возвращаемое подпрограммой, в данном случае не используется. В строках программы 180 и 200 формируются случайные числа, предназна-

Таблица 3.10

```

10 REM *****
20 REM * ПРОГРАММА ПРОВЕРКИ ЗНАНИИ МАТЕМАТИКИ *
30 REM * У УЧЕНИКОВ МЛАДШИХ КЛАССОВ *
40 REM *****
50 CLS:PRINT:PRINT
60 INPUT "КАК ТЕБЯ ЗОВУТ ";I$:PRINT:PRINT
70 PRINT I$," , Я ХОЧУ ПРОВЕРИТЬ ТВОЕ ЗНАНИЕ МАТЕМАТИКИ."
80 PRINT "Я БУДУ ПРЕДЛАГАТЬ ПРИМЕРЫ, А ТЫ ПОСТАРАЯСЯ ИХ РЕШИТЬ."
90 PRINT " ЕСЛИ ГОТОВ, НАЖМИ НА ЛЮБУЮ КЛАВИШУ."
100 A1=USR(-2045)
110 X=1:W=0
120 CLS:PRINT:PRINT
130 PRINT TAB(15);"ПОМНИ !"
140 PRINT "ЕСЛИ ПРИ ОТВЕТЕ ОШИБЕШЬСЯ И СЛУЧАЙНО НАЖМЕШЬ"
150 PRINT "НЕ НА ТУ КЛАВИШУ, ТО НАЖМИ КЛАВИШУ <-,А ЗАТЕМ НУЖНУЮ."
170 PRINT "ПОСЛЕ НАБОРА ОТВЕТА НАЖМИ КЛАВИШУ 'BK':PRINT
180 R=ABS(INT(RND(1)*20-10))
190 IF R=0 OR R=1 THEN 180
200 G=ABS(INT(RND(1)*20-10))
210 IF G=0 OR G=1 THEN 200
220 KL=INT(RND(1)*5)
230 IF KL=0 THEN 220
240 ON KL GOSUB 510,550,600,630
250 PRINT:PRINT I$;" СКОЛЬКО БУДЕТ ";P;U$;R;" = ";
260 INPUT D
270 IF D=Q THEN 370
280 PRINT:PRINT " НЕПРАВИЛЬНО !"
290 PRINT "ТЕБЕ, ";I$," , НАДО ПОДУМАТЬ.":PRINT
300 PRINT I$;" СКОЛЬКО БУДЕТ ";P;U$;R;" = ";
310 INPUT D:PRINT
320 IF D=Q THEN 370
330 PRINT "ПЛОХО, ";I$;"! ТЫ НЕ СМОГ РЕШИТЬ ЭТОТ ПРИМЕР."
340 PRINT " ВОТ ПРАВИЛЬНОЕ РЕШЕНИЕ:"
350 PRINT TAB(8);P;U$;R;" = ";Q:PRINT
360 GOTO 390
370 W=W+1
380 PRINT "МОЛОДЕЦ ";I$;"! ПРАВИЛЬНО.":PRINT
390 IF X=10 THEN 450
400 X=X+1
410 PRINT "РЕШИ ЕЩЕ ПРИМЕР."
420 PRINT "ЕСЛИ ГОТОВ, ТО НАЖМИ ЛЮБУЮ КЛАВИШУ."
430 A1=USR(-2045)
440 GOTO 120
450 PRINT:PRINT
460 IF W=10 THEN PRINT "МОЛОДЕЦ, ";I$," , ТЫ ЗНАЕШЬ МАТЕМАТИКУ НА 5!"
470 IF W=8 OR W=9 THEN PRINT "МОЛОДЕЦ, ";I$," , ТЫ ЗНАЕШЬ МАТЕМАТИКУ НА 4!"
480 IF W=6 OR W=7 THEN PRINT I$," , ТЫ ЗНАЕШЬ МАТЕМАТИКУ НА 3!"
490 IF W<6 THEN PRINT I$," , ТЫ ПЛОХО ЗНАЕШЬ МАТЕМАТИКУ!"
500 STOP
510 U$=" ПРИБАВИТЬ "
520 G=G*3:R=R*3
530 P=G:Q=G+R
540 RETURN
550 U$=" ОТНЯТЬ "
560 IF G=R THEN G=G+3
570 G=G*3:R=R*3
580 IF G<R THEN Q=G:G=R:R=Q
590 P=G:Q=G-R:RETURN
600 U$=" УМНОЖИТЬ НА "
610 P=G:Q=G*R
620 RETURN
630 U$=" РАЗДЕЛИТЬ НА "
640 P=G/R:Q=G
650 RETURN

```

ченные для генерации операндов очередной задачи. Вид действия (сложение, вычитание, умножение или деление) определяется следующим образом. В строке 220 переменной KL присваивается значение 1, 2, 3 или 4. В строке 240 в за-

зависимости от значений этой переменной происходит вызов соответствующей подпрограммы. В этих подпрограммах формируются операнды предлагаемого примера. Символьной переменной U\$ присваивается определенное значение в зависимости от вида арифметической операции.

Переменная X хранит значение, равное числу заданных примеров, а значение переменной W равно числу верных ответов.

При разработке программ и при работе с ЭВМ часто возникает необходимость перевода чисел из одной системы счисления в другую. Пользуясь известными правилами, это можно сделать и вручную. В табл. 3.11 приведен пример программы, позволяющей возложить эту трудоемкую работу на микро-ЭВМ. Программа переводит числа из шестнадцатичной системы счисления в десятичную и наоборот, широко используя встроенные функции Бейсика для работы с символьными данными.

Таблица 3.11

```

10 REM *****
20 REM * ПРОГРАММА ПЕРЕВОДА ЧИСЕЛ ИЗ ШЕСТ- *
30 REM * ШЕСТИНАДЦАТИРИЧНОЙ СИСТЕМЫ СЧИСЛЕНИЯ В *
40 REM * ДЕСЯТИЧНУЮ И НАОБОРОТ *
50 REM *****
100 INPUT "ВВЕДИТЕ (Д)ЕСЯТИЧНОЕ ИЛИ (Ш)ЕШТИНАДЦАТИРИЧНОЕ ";S$
110 IF S$="Ш" OR S$="Д" THEN 130
120 GOTO 100
130 IF S$="Д" THEN 160
140 INPUT "ШЕСТИНАДЦАТИРИЧНОЕ ЧИСЛО ";N$
150 N=0:GOSUB 1000:GOTO 100
160 INPUT "ДЕСЯТИЧНОЕ ЧИСЛО ";N
170 GOSUB 1100: GOTO 100
1000 REM ПЕРЕВОД ШЕСТИНАДЦАТИРИЧНОЕ -> ДЕСЯТИЧНОЕ
1010 FOR I=1 TO LEN(N$)
1020 D=ASC(MID$(N$,I,1))-48
1030 IF D<10 THEN 1050
1040 D=D-7
1050 IF D<0 OR D>=16 THEN N=0: GOTO 1080
1060 N=N*16+D
1070 NEXT I
1080 PRINT "ДЕСЯТИЧНОЕ ЧИСЛО = ";N
1090 RETURN
1100 REM ПЕРЕВОД ДЕСЯТИЧНОЕ -> ШЕСТИНАДЦАТИРИЧНОЕ
1110 A$=""
1120 L=INT(N/16)
1130 M=N-16*L
1140 IF M<10 THEN 1160
1150 M=M+7
1160 N=L:A$=CHR$(M+48)+A$
1170 IF N>=1 THEN 1120
1180 PRINT "ШЕСТИНАДЦАТИРИЧНОЕ ЧИСЛО = ";A$
1190 RETURN

```

Начинается программа с запроса ввода с клавиатуры одной из букв Д или Ш. При ошибочном вводе запрос повторяется. Далее в зависимости от выбранного режима оператор должен ввести десятичное или шестнадцатичное целое число. При переводе чисел из шестнадцатичной системы счисления в десятичную в программе используется то обстоятельство, что коды символов, отображающих шестнадцатичные числа (0...9, A—F), равны соответственно 48...57, 65...70. В подпрограмме преобразования проводится проверка на кор-

ректность набранного числа (оно должно содержать только допустимые символы). При обнаружении ошибки ввода результат преобразования равен нулю.

В подпрограмме перевода из десятичной системы счисления в шестнадцатеричную вычисляется код очередной цифры и все число «накапливается» в символьной переменной A\$. Здесь каких-либо специальных проверок не предусмотрено, так как сам интерпретатор осуществляет проверку корректности ввода числовых данных.

В тексте описываемой программы процедуры преобразования оформлены в виде подпрограммы. Это позволяет использовать их в программах на Бейсике, требующих ввод и отображение данных в шестнадцатеричной системе счисления. К таким программам относятся Монитор, Ассемблер и др.

Описанный интерпретатор имеет средства, позволяющие программам на Бейсике работать с подпрограммами, написанными на языке Ассемблера или в машинных кодах. Для вызова таких подпрограмм служит встроенная функция `USR (X)`. Рассмотрим пример, иллюстрирующий ее использование.

Алфавитно-цифровая клавиатура является в настоящее время самым распространенным устройством ввода данных в ЭВМ. Однако чтобы быстро и безошибочно вводить данные с ее помощью, нужен немалый опыт. Кроме того, если использование такой клавиатуры для ввода символьной информации вполне естественно, то ввод графических данных или указывание на какой-либо объект на экране с ее помощью весьма затруднен. В современных персональных микроЭВМ для выполнения этих действий используют ряд специальных устройств, таких как «джойстик» и «мышь». Первое представляет собой электромеханическое устройство, в котором два или три переменных резистора управляются одной рукояткой, выполненной в виде рычажка. Иногда вместо переменных резисторов используют емкостные датчики. Недостатком этого устройства является довольно сложная конструкция.

Манипулятор «мышь» представляет собой небольшую коробочку с клавишами, соединяемую с ЭВМ кабелем — «хвостом» (определенное внешнее сходство и породило название манипулятора — «мышь»). Перемещение манипулятора по поверхности стола приводит к взаимно однозначному перемещению курсора по экрану дисплея. Конечно, выдаваемую манипулятором информацию о положении программное обеспечение может трактовать и иначе. Например, в графическом режиме с помощью манипулятора можно рисовать на экране различные фигуры, а в игровых программах управлять перемещением каких-либо объектов по экрану. Однако чаще всего манипулятор используют именно для указывания.

Для ПК разработано простое устройство ввода данных, функционально аналогичное описанному выше. Его принципиальная схема изображена на рис. 3.2, а, возможный вариант внешнего оформления — на рис. 3.3. Устройство состоит из двух одновибраторов (D1, D2), во времязадающие цепи которых включены переменные резисторы R4 и R5. Устройство подключается к порту D14 процессорной платы. Длительность выходных импульсов пропорциональна углу поворота движка соответствующего резистора. При вводе данных резистором R4 (ось X) управляют большим пальцем правой руки, резистором R5 (ось Y) — средним, а кнопкой B1 — указательным.

Для подключения устройства к микроЭВМ используют свободные порты

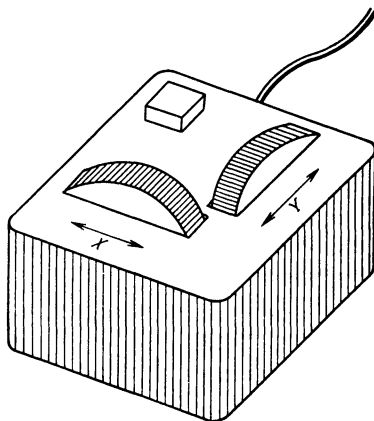
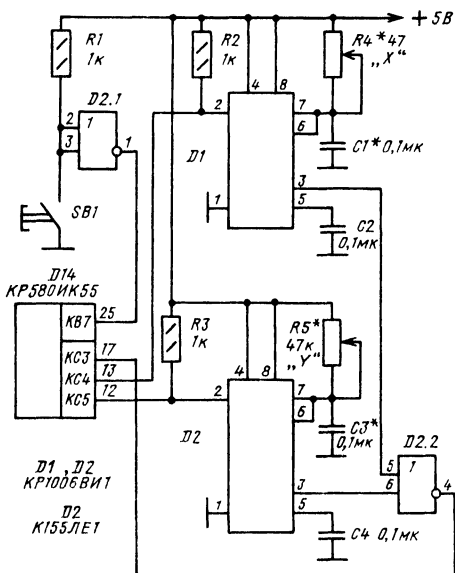


Рис. 3.3. Эскиз устройства управления перемещением курсора

Рис. 3.2. Принципиальная схема устройства управления курсором

ППА КР580ИК55. Можно подключить устройство иначе, но в этом случае необходимо соответственно изменить подпрограммы его обслуживания.

Запуск одновибраторов (по спаду импульса на входе 2) и определение длительности импульса производится подпрограммой, приведенной в табл. 3.12. После выдачи импульса запуска она считывает состояние порта ввода С, к разряду 3 которого через элемент D2.2 подключены выходы одновибраторов. При выходе из подпрограммы содержимое аккумулятора соответствует углу поворота движка переменного резистора. Конденсаторы C1 и C3 подбирают

Таблица 3.12

ПОДПРОГРАММА ROTX - Ось X			
1A80 1600	ROTX:	MVI	D,0
ИСПОЛЬЗУЕМ ОПЕРАЦИЮ УСТАНОВКИ/СБРОСА			
БИТОВ КАНАЛА С			
1A82 3E08	MVI	A,8	И ФОРМИРОВАНИЕ
1A84 320000	STA	0A003H	ИМПУЛЬСА
1A87 3E09	MVI	A,9	
1A89 320000	STA	0A003H	
1A8C C39B1A	JMP	ROTXY	
ПОДПРОГРАММА ROTY - Ось Y			
1A8F 1600	ROTY:	MVI	D,0
1A91 3E0A	MVI	A,0AH	
1A93 320000	STA	0A003H	
1A96 3E0B	MVI	A,0BH	
1A98 320000	STA	0A003H	
1A9B 3A0000	ROTX:	LDA	0A002H
1A9E 14	INR	D	
1A9F E608	ANI	8H	
1AA1 CA9B1A	JZ	ROTXY	
1AA4 7A	MOV	A,D	
1AA5 C9	RET		

таким образом, чтобы при полном повороте движков переменных резисторов R4 и R5 подпрограммы ROTX и ROTY возвращали числа 127 (размер по горизонтали) и 49 (размер по вертикали).

Для хранения подпрограмм в машинных кодах, используемых совместно с интерпретатором, специально выделена небольшая область памяти с адресами с IB00H по IC00H. Кроме приведенных подпрограмм, Вы можете разместить здесь (если, конечно, хватит места) и свои. После этого на магнитофон целесообразно записать эту новую версию интерпретатора, что позволит в дальнейшем пользоваться библиотекой подпрограмм, расширяющих возможности интерпретатора.

В табл. 3.13 приведен текст программы на Бейсике, которая предполагает использование описанного выше устройства управления курсором. Программа позволяет рисовать на экране дисплея разнообразные фигуры. В начале программы производится настройка порта (строка 80). Состояние кнопки, расположенной в устройстве управления курсором, определяется непосредственно в программе на Бейсике с помощью функции PEEK(X). В строках 90 и 110 происходит вызов подпрограмм в машинных кодах (функция USR(X)). Чтобы избежать в процессе работы программы появления ошибки 08, в ней предусмотрена коррекция значений, возвращаемых подпрограммами ROTX и ROTY.

Таблица 3.13

```

10 REM *****
20 REM * ПРОГРАММА РИСОВАНИЯ НА ЭКРАНЕ *
30 REM * С ПОМОЩЬЮ УСТРОЙСТВА УПРАВЛЕНИЯ *
40 REM * КУРСОРОМ. ИСПОЛЬЗУЕТ ПОДПРОГРАММЫ *
50 REM * В МАШИННЫХ КОДАХ ДЛЯ ОПРЕДЕЛЕНИЯ *
60 REM * КООРДИНАТ. *
70 REM *****
80 POKE -24573,131:REM НАСТРОЙКА ПОРТА
90 X=USR(6784):REM ВЫЗОВ ПОДПРОГРАММЫ ROTX
100 IF X>127 THEN X=127
110 Y=USR(6799):REM ВЫЗОВ ПОДПРОГРАММЫ ROTY
120 IF Y>49 THEN Y=49
130 REM ПРОВЕРКА СОСТОЯНИЯ КНОПКИ
140 IF PEEK(-24573) AND 128 THEN S=1:GOTO 160
150 S=0
160 PLOT X,Y,S
170 GOTO 90

```

Последний пример, который мы рассмотрим, посвящен разработке программ, с помощью которых на экран дисплея выводятся графические изображения. Формирование рисунков, состоящих из отрезков прямых линий и отдельных точек, обычно не вызывает каких-либо осложнений — для этого служат графические операторы Бейсика (CLS, PLOT, LINE). Необходимо только постоянно следить за тем, чтобы координаты, задаваемые этим операторам, не выходили за допустимые пределы, иначе интерпретатор выдаст сообщение об ошибке номер 08.

Формирование изображений окружностей и эллипсов — более сложная задача. Приводимая в табл. 3.14 универсальная подпрограмма позволяет нарисовать окружность произвольного радиуса в любом месте экрана. Эту подпрограмму можно включить в состав вашей программы и обращаться к ней с помощью оператора GOSUB 9000. Можно создать наборы подпрограмм

Таблица 3.14

```

9000 REM *****
9010 REM * ПОДПРОГРАММА ВЫЧЕРЧИВАНИЯ *
9020 REM * ИЗОБРАЖЕНИЙ ОКРУЖНОСТЕЙ И *
9030 REM * ЭЛЛИПСОВ. *
9040 REM *****
9050 P2=3.14159*2: REM УГОЛ = 2 ПИ
9060 REM ДЛЯ ВЫЧЕРЧИВАНИЯ ДУГ НАДО
9070 REM ИЗМЕНИТЬ КОНСТАНТЫ В ОПЕРАТОРЕ
9080 REM ИНИЦИАЛИЗАЦИИ ЦИКЛА
9100 FOR F9=0 TO P2 STEP 0.1
9110 X9 = XC + RC*KX*SIN(F9)
9120 Y9 = YC + RC*KY*COS(F9)
9130 IF X9<0 THEN X9=0
9140 IF X9>127 THEN X9=127
9150 IF Y9<0 THEN Y9=0
9160 IF Y9>49 THEN Y9=49
9170 PLOT X9,Y9,FL
9180 NEXT F9
9190 RETURN

```

(библиотеки) на Бейсике. Например, к описываемой ниже подпрограмме можно добавить подпрограммы для формирования геометрических фигур. В этом случае основная программа будет состоять из операторов присваивания, подготавливающих параметры для подпрограмм и операторов вызова подпрограмм. В Бейсике передача параметров к подпрограммам происходит присвоением значений соответствующим переменным. Все переменные в программе на Бейсике глобальные. Это значит, что если значение переменной изменилось при выполнении подпрограммы, то это отразится и на выполнении основной программы. Поэтому программист должен следить за сохранением значения переменных, необходимых для работы основной программы.

Для подпрограмм, входящих в библиотеку, принято назначать большие номера строк. Поэтому при написании новой программы сначала загрузите с магнитной ленты библиотеку, а основную программу набирайте со строки с номером 10. В качестве подпрограмм целесообразно оформлять только часто выполняемые операции, с остальными этого лучше не делать, так как вызов подпрограммы и возврат из нее требуют определенных затрат времени.

Перед обращением к подпрограмме (табл. 3.14) необходимо задать значения переменным, с которыми она работает. Переменные XC и YC определяют координаты центра окружности, а RC — ее радиус. Переменными KX и KY задают степень сжатия или растяжения по осям X и Y соответственно. Изменяя их значения, можно получать изображения окружности, вытянутые вдоль оси X или Y. При KX и KY, равных единице, на экране формируется эллипс, вытянутый вдоль оси Y. Это объясняется особенностями телевизионного раstra и дисплейного модуля. Поэтому для получения изображения окружности параметр KY должен быть равен $\approx 0,8$ KX.

В подпрограмме для вычисления координат точек, лежащих на окружности заданного радиуса R, использованы известные соотношения:

$$X = XC + RC * \sin(\varphi);$$

$$Y = YC + RC * \cos(\varphi),$$

где XC и YC — координаты центра окружности.

В подпрограмме предусмотрены контроль и коррекция координат центра окружности при условии:

если $XC < 0$, то $XC = 0$;
если $XC > 127$, то $XC = 127$;
если $YC < 0$, то $YC = 0$;
если $YC > 49$, то $YC = 49$.

Поэтому при выполнении подпрограммы сообщение об ошибке 08 не возникает.

Таблица 3.15

```
10 REM *****
20 REM * ПРОГРАММА ФОРМИРОВАНИЯ *
30 REM * ИЗОБРАЖЕНИЯ ОЛИМПИЙСКОГО *
40 REM * СИМВОЛА. ИСПОЛЬЗУЕТ ПОД- *
50 REM * ПРОГРАММУ "КРУГ" *
60 REM *****
70 CLS
80 KX=1:KY=0.8:REM КОЭФФИЦИЕНТЫ СЖАТИЯ
90 RC=10: REM РАДИУС ОКРУЖНОСТИ
100 YC=30
110 XC=40:GOSUB 9000:REM 1 КРУГ
120 XC=56:GOSUB 9000:REM 2 КРУГ
130 XC=72:GOSUB 9000:REM 3 КРУГ
140 YC=15
150 XC=48:GOSUB 9000:REM 4 КРУГ
160 XC=64:GOSUB 9000:REM 5 КРУГ
170 STOP
```

В табл. 3.15 приведена программа, формирующая на экране изображение олимпийского символа — пяти пересекающихся колец, в которой использована описанная подпрограмма.

Программа и память ЭВМ

При разработке программ на любом языке программирования практически всегда приходится учитывать такие факторы, как время выполнения программы и необходимый для этого объем памяти. Оба параметра следует по возможности уменьшать, это позволит и результат получить быстрее, и учесть ограничения, связанные с небольшим объемом памяти ЭВМ. Однако добиться одновременно и того, и другого сложно.

К программам на Бейсике сказанное имеет самое прямое отношение, особенно если транслятор с языка высокого уровня (как в нашем случае) реализован в виде интерпретатора и объем памяти для хранения программ и переменных мал. Рассмотрим несколько приемов, позволяющих уменьшить требуемый для работы программы объем памяти.

При работе интерпретатора в ОЗУ одновременно находятся сам интерпретатор, текст программы на Бейсике и различные переменные и константы, встречающиеся в программе. Появление сообщения об ошибке 07, как при наборе текста программы, так и при ее выполнении, указывает на то, что для данной программы объем памяти недостаточен. Как следует поступать в таких случаях? Прежде всего можно сократить в программе комментарии, уменьшить

длину текстовых сообщений, использовать однобуквенные имена для переменных. Объем памяти, занимаемый текстом программы, можно также уменьшить, набирая в каждой строке не по одному, а по несколько операторов, отделяя их друг от друга символом «:» (почему происходит такое сокращение Вы поймете, ознакомившись с дополнительными сведениями об интерпретаторе).

Если после этих переделок объем памяти все равно недостаточен, то следует критически проанализировать необходимость использования тех или иных переменных и особенно переменных с индексами — массивов. Для массивов следует резервировать ровно столько ячеек памяти, сколько для них в действительности потребуется. Например, если массив A(I) состоит из пяти элементов, а вы его не описали оператором DIM, то по умолчанию интерпретатор выделит в памяти место для десяти элементов массива, и это приведет к потере 20 байт памяти. Для того чтобы оценить, к каким затратам памяти приводит тот или иной вариант реализации алгоритма, удобно пользоваться встроенной функцией FRE (0). Попробуйте посмотреть, чему равно значение этой функции до загрузки программы в память, после загрузки и после выполнения программы. Для этого достаточно в непосредственном режиме набрать PRINT FRE (0).

Таким образом, вы сможете определить, сколько места в памяти занимает текст программы и сколько ячеек отводится для хранения переменных и констант.

Программа и время ее выполнения

Сократить время выполнения программы несколько сложнее, чем ее объем. Для этого прежде всего необходимо определить ту часть программы, на выполнение которой затрачивается наибольшее время.

Известно, что циклическая часть программы, занимающая всего 5% исходного текста, требует для своего выполнения обычно около 95% времени работы всей программы. Вы должны обнаружить эти критические циклы в программе и попытаться уменьшить время их выполнения. Вначале надо убедиться, что все операторы, входящие в этот цикл, действительно должны выполняться внутри него. Например, если значение переменной не изменяется в цикле (табл. 3.16, пример 1), то ее инициализацию следует производить вне цикла

Таблица 3.16

10 REM *****	10 REM *****
20 REM * ПРИМЕР 1 *	20 REM * ПРИМЕР 2 *
30 REM *****	30 REM *****
40 FOR I=0 TO 340	40 A=SIN(1.45)*30
50 A=SIN(1.45)*30	50 FOR I=0 TO 340
60 B(I)=A*I/4	60 B(I)=A*I/4
70 NEXT I	70 NEXT I
80 STOP	80 STOP
10 REM *****	10 REM *****
20 REM * ПРИМЕР 3 *	20 REM * ПРИМЕР 4 *
30 REM *****	30 REM *****
40 FOR I=1 TO 50	40 FOR I=1 TO 30
50 S(I)=0	50 S(I)=0:S(I+30)=0
60 NEXT I	60 NEXT I
70 STOP	70 STOP

(пример 2), что сократит время его выполнения. Уменьшить время выполнения программы можно также некоторым видоизменением операторов, входящих в тело цикла. В примере 3 обнуление элементов массива проводится в цикле. Если переписать эту программу, как показано в примере 4, то время выполнения операторов FOR...NEXT значительно сократится.

Во многих случаях, если число элементов массива невелико, имеет смысл вообще отказаться от оператора цикла и проводить инициализацию элементов массива следующим образом:

$$10 \ A(1) = 0: A(2) = 0: A(3) = 0: A(4) = 0: A(5) = 0.$$

При вычислении результатов выражений имеется возможность сократить время вычислений благодаря использованию дополнительных переменных. На пример, при выполнении строки IF $K * R/T - D > 0$ THEN $E = K * R/T + 1$ дважды определяется величина $K * R/T$. Если переписать эту строку следующим образом:

$$Q = K * R/T: \text{IF } Q - D > 0 \text{ THEN } E = Q + 1,$$

то вычисление значения новой переменной Q будет проведено только один раз. Для сокращения времени работы программы необходимо резервировать несколько ячеек памяти для хранения переменной Q .

Для обращения к элементу массива — интерпретатору, как правило, требуется больше времени, чем для обращения к простой переменной. Это связано с необходимостью обработки индексов. Поэтому рекомендуется, если это возможно, использовать простые переменные, которые будут хранить значения часто используемых элементов массива, напримр. элементы массива

$$Z1 = Y(A) + 10: Z2 = Y(A) - 5: Z3 = Y(A) + 20$$

целесообразнее определять так:

$$Z1 = Y(A) + 10: Z2 = Z1 - 15: Z3 = Z1 + 10.$$

Особенно большой выигрыш во времени может быть получен, если подобные вычисления проводятся в теле цикла или в программе используются многомерные массивы.

Мы рассмотрели ряд конкретных примеров, позволяющих сократить время выполнения программ на Бейсике, но, конечно, наиболее ощутимые результаты можно получить, если воспользоваться функцией USR(X) и запрограммировать критичный по времени выполнения фрагмент алгоритма на языке Ассемблера.

Таким образом, перед программистом всегда стоит проблема выбора наилучшего варианта реализации программы с точки зрения времени ее выполнения, занимаемого объема памяти, простоты и ясности.

Дополнительные сведения об интерпретаторе

Посмотрим теперь, как хранится программа на Бейсике в памяти ЭВМ. Это может оказаться полезным в случае какого-либо сбоя в работе микроЭВМ, который может привести к потере набранной программы. Кроме того, приводимые ниже сведения помогут восстановить программу, введенную с ошибками

с магнитной ленты, и разобраться в организации структуры данных в ваших программах.

Интерпретатор Бейсика занимает начальную область оперативной памяти, описываемой микроЭВМ. Текст программы хранится в памяти сразу же после кодов интерпретатора, за ним следует область памяти, выделенная для хранения переменных. Вблизи верхней границы ОЗУ имеется специальная область (стек), отведенная для «внутренних» нужд интерпретатора.

Каким образом строка программы представлена в микроЭВМ? Просматривая содержимое памяти с помощью директив D и L Монитора, вы можете обнаружить, что хранящаяся в памяти информация не похожа на то, что было набрано на клавиатуре. Это вполне объяснимо, так как микроЭВМ «понимает» только двоичные коды, т. е. способна понять код той или иной буквы, а не саму букву. Поэтому строки программы хранятся в памяти в виде двоичных кодов и переводятся в символьный вид только в случае просмотра текста программы по директиве интерпретатора. При этом каждый вводимый символ программы занимает в памяти одну ячейку (1 байт), что потребовало бы, во-первых, очень большого объема памяти, и, во-вторых, программа выполнялась бы медленно.

Вместо того чтобы хранить в памяти коды всех символов исходного текста программы, можно закодировать каждое ключевое слово всего одним байтом. Это вполне возможно, так как из 256 возможных двоичных кодов ($2^8 = 256$), которые можно записать в одну ячейку памяти, для кодирования алфавитно-цифровых символов используется только 128. Двоичные коды, у которых старший бит равен 1, и использованы для кодирования ключевых слов языка Бейсик (табл. 3.17). Что же дает такое «сжатие» информации? Очень многое. Например, если в программе 100 раз встречается оператор GOSUB, то кодирование позволяет сэкономить 400 байт памяти!

На рис. 3.4 показан формат строки программы на Бейсике в том виде, в каком она хранится в памяти микроЭВМ. В начале каждой строки два байта отведены для хранения указателя адреса начала следующей строки программы, следующие два байта хранят номер строки, а заканчивается она байтом, заполненным одними нулями. Таким образом, текст программы хранится в памяти в виде специальной структуры данных, называемой односвязным списком.

Заканчивая обработку очередной строки программы, интерпретатор по-

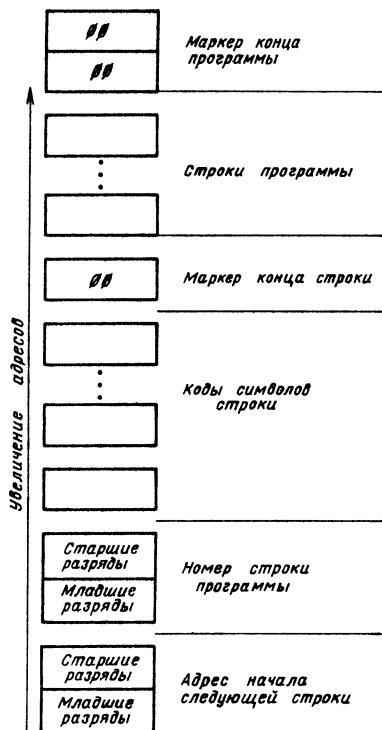


Рис. 3.4. Формат хранения в памяти микроЭВМ фрагмента программы на Бейсике

Таблица 3.17

Слово	Код		Слово	Код		Слово	Код	
	шест- над- цати- рич- ный	деся- тич- ный		шест- над- цати- рич- ный	деся- тич- ный		шест- над- цати- рич- ный	деся- тич- ный
CLS	80	128	CONT	97	151	SGN	AD	174
FOR	81	129	LIST	98	152	INT	AE	175
NEXT	82	130	CLEAR	99	153	ABS	AF	176
DATA	83	131	MLOAD	9A	154	USR	BO	177
INPUT	84	132	MSAVE	9B	155	FRE	B1	178
DIM	85	133	NEW	9C	156	KEY	B2	179
READ	86	134	TAB (9D	157	POS	B3	180
CUR	87	135	TO	9E	158	SQR	B4	181
GOTO	88	136	SPC (9F	159	RND	B5	182
RUN	89	137	FN	A0	160	LOG	B6	183
IF	8A	138	THEN	A1	161	EXP	B7	184
RESTORE	8B	139	NOT	A2	162	COS	B8	185
COSUB	8C	140	STEP	A3	163	SIN	B9	186
RETURN	8D	141	+	A4	164	TAN	BA	187
REM	8E	142	-	A5	165	ATN	BB	188
STOP	8F	143	*	A6	166	PEEK	BC	189
BEL	90	144	/	A7	167	LEN	BD	190
ON	91	145	^	A8	168	STR\$	BE	191
PLOT	92	146	AND	A9	169	VAL	BF	192
LINE	93	147	OR	AA	170	ASC	CO	193
POKE	94	148	>	AB	171	CHR\$	C1	194
PRINT	95	149	=	AC	172	LEFT\$	C2	195
DEF	96	150	<	AD	173	RIGHT\$	C3	196
						MID\$	C4	197

следовательно просматривает указатели списка до тех пор, пока не будет найдена строка с требуемым номером. Конец списка помечается двумя байтами, заполненными нулями. Вы таким же образом можете вручную (с помощью директив Монитора) определить, где заканчивается программа, просматривая указатели списка до тех пор, пока не обнаружите три смежных байта, заполненных нулями. Во многих практических случаях, воспользовавшись рассмотренными рекомендациями, можно восстановить программу, в которой в результате сбоя была нарушена целостность списка. После восстановления структуры списка необходимо изменить значения, хранящиеся в ячейках памяти 0245H и 0246H. В этих ячейках хранятся значения соответственно младшего и старшего байтов конечного адреса программы. Этот адрес на два превосходит адрес первого байта маркера конца списка.

Сфера использования интерпретатора Бейсик и языков высокого уровня

Все языки программирования можно разделить на две группы. К первой группе относится язык, непосредственно понимаемый микропроцессором —

язык машинных кодов. Для каждого типа ЭВМ существует свой машинный язык, поэтому язык машинных кодов называют машинно-ориентированным. Ко второй группе относятся более легкие для программиста языки высокого уровня — машинно-независимые языки.

Приемы составления программ на языке машинных кодов и на языках высокого уровня существенно отличаются. Отдельную команду программы, составленной в машинных кодах, можно сравнить со словом человеческого языка, которое приобретает определенный смысл только в предложении. По-другому обстоит дело с языками высокого уровня. В них каждую команду можно сравнить уже со словосочетанием, имеющим свой смысл.

Таким образом, языки высокого уровня предоставляют программисту как бы набор штампов — готовых стандартных средств, используемых для решения определенных задач. А это существенно упрощает решение задач, оберегает от ошибок, но ограничивает творческую инициативу. Поэтому появление языков высокого уровня существенно облегчило программирование, но во многих случаях программы стали менее эффективны.

Как же оценивается эффективность программ? Рассмотрим только основные ее параметры: объем памяти, занимаемый программой, и время ее работы. Значение этих параметров стараются максимально уменьшить. Это необходимо для того, чтобы программа умещалась в памяти компьютера или занимала меньше в ней места, чтобы быстрее осуществлялся ее ввод или, если вы хотите записать ее в ПЗУ, чтобы для этого требовались меньшие по емкости и более дешевые микросхемы.

Программа должна работать быстро, чтобы не приходилось ожидать у экрана результатов и компьютер мог управлять какими-либо подключенными к нему устройствами в реальном масштабе времени.

Практически между моментом прихода информации от объекта и подачи на него от ЭВМ управляющего сигнала всегда проходит некоторое время. Это время обычно называют временем реакции управляющей программы или микропроцессорной системы в целом. Оно не должно превосходить некоторое заданное значение, выбор которого зависит от объекта управления. Например, чтобы управлять моделью железной дороги, программа должна реагировать на изменение состояния модели с запаздыванием на десятые доли или единицы секунд. Если же гребутся принимать и дешифровать сигналы, переданные кодом Морзе, то скорость работы программы должна быть на порядок выше.

Языки высокого уровня ориентированы на различные области применения (например, Фортран — для решения математических задач, Форт — для программирования алгоритмов управления, Кобол — для решения экономических задач). Они имеют соответствующие «выразительные» средства, позволяющие получать эффективные программы. Поэтому важен правильный выбор языка программирования применительно к каждой конкретной задаче.

Эффективность программ существенно зависит от трансляторов — программ, осуществляющих перевод (трансляцию) исходного текста программы с языка высокого уровня в машинные коды. Именно трансляторы переводят языковые штампы в последовательность машинных команд.

Трансляторы относят к системным программам, которые как бы являются обязательными и неотъемлемыми частями компьютеров. Системные программы

необходимы пользователям компьютера как инструменты при написании и отладке своих прикладных программ. Наличие трансляторов с одинаковых языков высокого уровня для компьютеров разных типов, в том числе для компьютеров с различной системой команд, позволяет писать прикладные программы, работающие на любой из этих машин. Это означает, что языки высокого уровня обеспечивают возможность работы одной программы на разных машинах.

Есть два типа трансляторов — *компиляторы* и *интерпретаторы*. Первые целиком считывают текст программы и переводят его в машинные (их иногда называют объектными) коды. Программу в кодах, как правило, записывают на какой-либо внешний носитель информации. Затем для выполнения программы ее с этого носителя загружают в память компьютера.

Интерпретатор поочередно просматривает строки программы в том порядке, в котором они должны выполняться, и вызывает нужные подпрограммы, входящие в состав интерпретатора. Подпрограммы тут же выполняются компьютером. Таким образом, в памяти компьютера постоянно должны находиться интерпретатор и исходный текст прикладной программы.

При использовании компиляторов скорость работы прикладных программ во много раз больше, чем при использовании интерпретаторов. Действительно, компилятор транслирует исходный текст прикладной программы только один раз — при «переводе» в машинные коды. Интерпретатору же приходится транслировать программу построчно при каждом ее выполнении, и время этой трансляции входит во время работы прикладной программы.

Рассмотрим, как соотносятся объемы памяти, необходимые для работы прикладных программ при использовании компиляторов и интерпретаторов. Учтем, что объем памяти для хранения текста строки программы на языке высокого уровня в среднем меньше того, который необходим для хранения соответствующих машинных кодов, полученных в результате трансляции. Очевидно, что при использовании компилятора объем программы в машинных кодах будет больше объема исходного текста, и чем больше программа, тем больше будет эта разница.

Так как при работе с интерпретатором в памяти компьютера всегда находится текст прикладной программы и транслятор, то при малых исходных программах (меньших или сравнимых с объемом интерпретатора) объем требуемой памяти компьютера будет больше, чем для аналогичной программы в машинных кодах, полученной с помощью компилятора. Для прикладных программ, текст которых занимает больший, чем интерпретатор, объем памяти, их суммарный объем может оказаться меньше, чем объем кодов предварительно скомпилированной программы.

Следует заметить, что, используя компилятор, можно выполнить трансляцию не на том компьютере, на котором программа будет работать, а на хорошо оснащенной инструментальной ЭВМ, которая имеет больший объем внутренней памяти, внешнюю память в виде накопителей с произвольным доступом к информации (обычно это накопители на гибких магнитных дисках — НГМД). Все это необходимо для того, чтобы можно было использовать сложное системное программное обеспечение (различные трансляторы, редакторы текстов и программ, отладчики программ), что повышает производительность труда программистов.

В качестве инструментального компьютера для ПК может быть использована ЭВМ СМ-1800, выполненная на том же микропроцессоре КР580ИК80А. Могут быть использованы и ЭВМ с другой системой команд, например СМ-4. Эта машина должна быть оснащена специальными программами — кроссобееспечением, позволяющим транслировать программы в машинные коды микропроцессора КР580ИК80А.

Чтобы наглядно представить выигрыш, который дают инструментальные машины, отметим, что компилятор с языка Паскаль для ЭВМ СМ-1800 занимает на гибком магнитном диске более 200 Кбайт и требует для своей работы не менее 48 Кбайт ОЗУ машины, куда отдельные части компилятора загружаются по мере надобности. Таким образом, работать на языке Паскаль на ПК без инструментальной ЭВМ практически нельзя.

Полученные на инструментальных ЭВМ программы в машинных кодах могут работать на таких простых компьютерах, как, например, ПК или какие-либо микропроцессорные устройства управления, память которых рассчитана на выполнение только одной программы.

Несмотря на повсеместное распространение языков высокого уровня, по занимаемому объему памяти и скорости работы во многих случаях более выгодны программы, написанные на языке Ассемблера. Это машинно-ориентированный язык, программирование на нем, по сути, не отличается от программирования в машинных кодах. Но при этом язык Ассемблера позволяет работать программисту, используя мнемонику команд, символьные переменные и автоматическое распределение памяти, избавляет его от ручной трансляции исходных текстов программ в машинные коды.

С чем же связано то, что ассемблерные программы более эффективны? Поясним это на примере. Компилятор, встретив в исходном тексте программы какую-либо арифметическую операцию над целыми числами, включает в состав объектного (машинного) кода соответствующие программы для выполнения арифметических операций. Но предположим, что эти программы могут выполнять действия не только с целыми числами. Следовательно, в объектном коде появляются «лишние», неиспользуемые коды. Эта избыточность приводит к нерациональному увеличению объема программы после трансляции, увеличивается время ее выполнения. А программы на языке Ассемблера в принципе могут быть свободны от такой избыточности, так как сам язык очень близок к машинным кодам.

ПРИЛОЖЕНИЕ

1. Микропроцессор КР580ИК80А

Микропроцессор КР580ИК80А конструктивно выполнен в пластмассовом корпусе с 40 выводами. Для микропроцессора необходимы три источника напряжения питания: $+5$, -5 и $+12$ В.

Выводы микропроцессора и назначение передаваемых через них сигналов:

$\Phi 1$, $\Phi 2$ — входные периодические сигналы для тактирования микропроцессора, формируемые внешним генератором. Они должны иметь амплитуду $+12$ В,

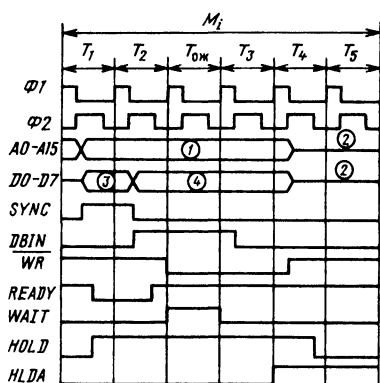


Рис. П.1. Временная диаграмма машинного цикла микропроцессора: 1 — адрес; 2 — высокий импеданс; 3 — состояние микропроцессора; 4 — данные

в то время как все другие сигналы микропроцессора имеют ТТЛ-уровни. Тактирующие импульсы определяют моменты всех действий микропроцессора. Выполнение каждой команды происходит за один или несколько (до 5) машинных циклов, в каждом из которых происходит обмен одним байтом с памятью или портами ввода-вывода. В свою очередь, машинный цикл разделяется на 3...5 тактов, длительность которых равна периоду следования тактирующих импульсов. Временная диаграмма выполнения машинного цикла приведена на рис. П. 1. В первом машинном цикле M_1 любой команды микропроцессор в такте T_3 считывает из памяти микроЭВМ код операции команды и, если не требуются дополни-

тельные действия по обмену данными с памятью или портами ввода-вывода, заканчивает ее выполнение в такте T_4 (или T_4 и T_5). В противном случае для выполнения команды требуются дополнительные машинные циклы $M_2—M_5$.

$A0—A15$ — выходные сигналы шины адреса для адресации памяти и портов ввода-вывода. Причем при адресации портов ввода-вывода информация, передаваемая по линиям $A0—A7$ шины адресов, дублируется на линиях $A8—A15$. Это позволяет более равномерно распределить нагрузку на шину адреса при использовании в микроЭВМ большого количества портов ввода-вывода. Код адреса появляется на шине адреса в такте T_1 с задержкой 200...270 нс относительно фронта импульса Φ_2 и остается стабильным до момента возникновения фронта импульса Φ_2 в такте, следующем за тактом T_3 текущего цикла. Линии шины адреса могут принимать высокоимпедансное состояние.

$D0—D7$ — двунаправленные линии шины данных. Являются входными при вводе данных из памяти или портов ввода-вывода, выходными — при выводе данных из микропроцессора. Используются также для вывода в такте T_1 по фронту Φ_2 дополнительной информации о характере операций обмена в текущем машинном цикле — байта состояния. Эта информация снимается с линий шины данных через 220...280 нс после прихода импульса Φ_2 в такте T_2 . Линии шины данных могут принимать высокоимпедансное состояние.

$DBIN$ — выходной сигнал, инициирующий выдачу данных из ячейки памяти или порта на шину данных. Во время действия этого сигнала микропроцессор считывает байт с шины данных. Сигнал генерируется с задержкой 130...200 нс относительно фронта Φ_2 в такте T_2 машинных циклов считывания данных в микропроцессор. Байт данных должен быть установлен на шине данных не менее чем за 30 нс до появления заднего фронта Φ_1 и оставаться стабильным не менее 130 нс после появления фронта Φ_2 такта T_3 . Сигнал $DBIN$ снимается по положительному фронту Φ_2 такта T_3 с максимальной задержкой 200 нс.

WR — выходной сигнал, сопровождающий выдачу микропроцессором байта данных на шину данных для записи в ячейку памяти или порт. Формируется в

тактах T3 или T_{ож} (такт ожидания) с задержкой 70...120 нс относительно положительного фронта сигнала Ф1. Байт данных микропроцессор формирует на шине данных в такте T2 с задержкой 220...280 нс относительно фронта Ф2. Сигнал \overline{WR} снимается в такте, следующем за тактом T3 с задержкой около 100 нс относительно фронта Ф1. Следует заметить, что на рис. П.1 сигналы DBIN и \overline{WR} изображены активными условно. В течение реального машинного цикла активен только один из них. Эти сигналы являются общими как для модулей памяти, так и для портов ввода-вывода, что приемлемо только в простейших микроЭВМ. В более сложных микроЭВМ используют дополнительные сигналы управления, выдаваемые микропроцессором на шину данных в тактах T1/T2 в виде 8-разрядного кода — байта состояния, который запоминается (фиксируется) во внешнем регистре и определяет действие микропроцессора в следующих тактах текущего машинного цикла. Наличие единицы в разрядах D0, D2—D7 или нуля в разряде D1 байта состояния является признаком выполнения микропроцессором в текущем машинном цикле следующих действий:

- D0(INTA) — обслуживание запроса прерывания;
- D1(WO) — запись в память или вывод данных в порт;
- D2(STACK) — обращение к области памяти, используемой в качестве стека;
- D3(HLT) — останов микропроцессора по команде HLT;
- D4(OUT) — вывод данных в порт;
- D5(MI) — чтение кода операции команды;
- D6(INP) — ввод данных из порта;
- D7(MEMR) — чтение данных из памяти.

SYNC — выходной сигнал синхронизации, вырабатываемый по фронту Ф2 в такте T1 каждого машинного цикла и указывающий, что по шине данных передается байт состояния микропроцессора. Сигнал снимается через 130...200 нс после появления фронта Ф2 в такте T2. По сигналу синхронизации байт состояния фиксируется во внешнем регистре. Таким образом, через шину данных из микропроцессора поступают не только данные, а в отдельные моменты и управляющие сигналы. Такое использование шин для передачи в разное время различных сигналов называется временным мультиплексированием. Оно необходимо из-за ограниченного числа выводов микропроцессора. Увеличивать же число выводов у микропроцессора нецелесообразно, так как это повышает его стоимость и снижает надежность.

READY — входной сигнал от модулей памяти или портов ввода-вывода, указывающий на их готовность к обмену данными с микропроцессором. Если сигнал на этом входе имеет низкий уровень, то после такта T2 микропроцессором выполняются вспомогательные такты ожидания T_{ож} (т. е. он переходит в состояние ожидания). Это позволяет использовать в микроЭВМ память и периферийное оборудование с малым быстродействием. Число вспомогательных тактов ожидания определяется длительностью поддержания входного сигнала на выводе READY в состоянии 0. Как только память или порт будут готовы к обмену, они установят на этом входе высокий уровень напряжения, что позволит микропроцессору перейти к выполнению такта T3. Контроль состояния сигнала на входе READY осуществляется в момент действия импульса Ф2 в тактах T2 и T_{ож}. Для правильной работы микропроцессора необходимо, чтобы этот сигнал устанавливался и оставался стабильным минимум за 180 нс до оконча-

ния импульса $\Phi 2$. Необходимо отметить, что при выполнении микропроцессором тактов ожидания выходные сигналы на его выводах остаются неизменными.

WAIT — выходной сигнал, подтверждающий переход микропроцессора в состояние ожидания. Устанавливается и снимается по фронту $\Phi 1$.

HOLD — входной сигнал, инициирующий перевод шин адресов и данных микропроцессора в высокоимпедансное состояние. Это позволяет производить непосредственный обмен данными между ячейками памяти и портами микроЭВМ, минуя микропроцессор (режим ПДП). Контроль состояния сигнала на входе *HOLD* осуществляется в момент действия импульса $\Phi 2$ в тактах T_2 , аналогично как и при контроле сигнала на входе *READY*. Для правильной работы микропроцессора необходимо, чтобы этот сигнал на входе *HOLD* устанавливался и оставался стабильным минимум за 180 нс до окончания импульса $\Phi 2$. Рекомендуется синхронизировать сигнал *HOLD* сигналом $\Phi 1$. Так как перед выполнением циклов ПДП микропроцессор должен закончить обмен в текущем машинном цикле, то сигнал *READY* имеет более высокий приоритет, чем *HOLD*.

H LDA — выходной сигнал подтверждения перевода шин адресов и данных микропроцессора в высокоимпедансное состояние. Формируется при обнаружении на входе *HOLD* напряжения высокого уровня. Формирование сигнала происходит с задержкой в 70...120 нс относительно положительного фронта $\Phi 1$ в такте T_3 или следующем за ним такте (T_4 или T_1 следующего цикла) в зависимости от того, выполнялся ли цикл чтения или записи соответственно. Затем по положительному фронту $\Phi 2$ с задержкой не более 200 нс линии шин адресов и данных переводятся в высокоимпедансное состояние.

INT — входной сигнал запроса прерывания. При его появлении микропроцессор, окончив выполнение текущей команды, устанавливает в первом такте цикла $M1$ очередной команды на шине данных байт состояния, где вместо признака чтения данных из памяти *MEMR* присутствует признак подтверждения прерывания *INTA*. В результате этого микропроцессор вместо кода очередной команды текущей программы считывает с шины данных микроЭВМ код одной из команд вызова подпрограмм, формируемый дополнительной БИС — контроллером прерывания. Таким образом происходят прерывание выполнения текущей программы и переход к выполнению подпрограммы обработки запроса прерывания.

INTE — выходной сигнал разрешения прерывания выполнения текущей программы. Отражает состояние внутреннего триггера микропроцессора, разрешающего прохождение запроса прерывания. Он устанавливается, если ранее в программе была выполнена команда разрешения прерывания *EI*. При обработке запроса прерывания триггер разрешения прохождения запроса прерывания автоматически сбрасывается и снимает выходной сигнал *INTE*, что не позволяет микропроцессору в дальнейшем реагировать на новые запросы прерывания вплоть до выполнения следующей команды *EI*. Запретить прерывание текущей программы можно также командой запрета прерывания *DI* в любом месте выполняемой программы.

RESET — входной сигнал установки микропроцессора в исходное состояние. После снятия этого сигнала микропроцессор начинает выполнение программы с команды, содержащейся в ячейке памяти с адресом 0000H. При этом

Таблица П.1

Машинный		Действия
цикл	такт	
M1 чтение	T1	Вывод на шину адреса содержимого указателя адреса
	T1/T2 T2	Вывод на шину данных байта состояния Опрос состояния входов READY и HOLD; увеличение на 1 содержимого указателя адреса
	T2/T3 T4	Чтение из памяти кода операции команды Подготовка к выполнению команды
M2 чтение	T1	Вывод на шину адреса содержимого указателя адреса
	T1/T2 T2	Вывод на шину данных байта состояния Опрос состояния входов READY и HOLD; увеличение на 1 содержимого указателя адреса
	T2/T3	Чтение из памяти второго байта команды — младшего байта адреса памяти для хранения содержимого регистра L
M3 чтение	T1	Вывод на шину адреса содержимого указателя адреса
	T1/T2 T2	Вывод на шину данных байта состояния Опрос состояния входов READY и HOLD; увеличение на 1 содержимого указателя адреса
	T2/T3	Чтение из памяти третьего байта команды — старшего байта адреса памяти для хранения содержимого регистра L
M4 запись	T1	Вывод адреса памяти для хранения содержимого регистра L на шину адреса
	T1/T2 T2	Вывод на шину данных байта состояния Опрос состояния входов READY и HOLD
	T2/T3 T3	Запись в память содержимого регистра L Увеличение на 1 адреса памяти для хранения содержимого регистра L
M5 запись	T1	Вывод адреса памяти для хранения содержимого регистра H на шину адреса
	T1/T2 T2	Вывод на шину данных байта состояния Опрос состояния входов READY и HOLD
	T2/T3 T3	Запись в память содержимого регистра H Опрос состояния входа INT

автоматически запрещается прием запросов прерывания. Сигнал RESET должен иметь длительность не менее трех периодов тактовой частоты.

Из анализа временной диаграммы на рис. П.1 следует, что в каждом такте любого машинного цикла микропроцессор выполняет определенные действия:

T1 — устанавливает код адреса периферийного модуля на шине адресов;

T1/T2 — выводит информацию о своем состоянии по шине данных;

T2 — проверяет состояние сигналов на входах READY и HOLD;

T3 — реализует обмен одним байтом информации с памятью или портами;

T4/T5 — выполняет внутренние межрегистровые передачи и обработку данных в соответствии с командой.

Для примера в табл. П.1 приведена последовательность действий, совершаемых микропроцессором при выполнении команды SHLD. Эта команда предписывает запомнить содержимое внутренних регистров H и L микропроцессора в ячейках памяти. Адрес ячейки памяти, в которой необходимо разместить содержимое регистра L, задается вторым и третьим байтами этой трехбайтовой команды. Содержимое регистра H размещается в ячейке памяти с адресом, на единицу большим. Выполнение команды требует пяти машинных циклов: в трех первых считывается сама трехбайтовая команда, а два другие служат для записи содержимого регистров H и L в ячейки памяти.

2. Программируемый периферийный адаптер КР580ВВ55

Для упрощения подключения внешних устройств к шинам микроЭВМ применяют программируемые периферийные микропроцессорные БИС разных типов.

Внешние устройства подключаются к микроЭВМ с помощью соединительных линий, по которым передаются данные и управляющие сигналы. Перечень управляющих сигналов и данных, их электрические и временные параметры, порядок следования и взаимодействия определяют интерфейс внешних устройств. Интерфейс обуславливает также и конструкцию электрических соединителей, назначение их контактов.

С помощью программируемых периферийных БИС взаимодействие сигналов шин микроЭВМ и сигналов многих внешних устройств можно организовать программно, не разрабатывая для этого специальные схемы. Кроме того, программируемые периферийные БИС позволяют формировать программным путем произвольные последовательности импульсов (временные диаграммы), что и используется в РК при подключении к ней клавиатуры и магнитофона.

Здесь применяется БИС КР580ВВ55 (ППА), предназначенная для ввода или вывода данных в параллельной форме из микроЭВМ. Эта микросхема выполнена в 40-выводном пластмассовом корпусе, питается от одного источника напряжения +5 В.

Внешние устройства подключают к линиям ввода или вывода ППА, образующим каналы А, В и С по 8 линий в каждом. Канал С может быть разделен на младший и старший подканалы. Линии каналов связаны с соответствующими буферными регистрами ППА — портами А, В и С.

Обмен информацией между микропроцессором и портами происходит в соответствии с командами ввода или вывода программы, обслуживающей ППА, по шине данных, в которой ППА подключен через двунаправленные линии D0—D7. Эти линии находятся в высокоимпедансном состоянии при отсутствии сигнала «Выбор микросхемы» на выводе CS и одного из сигналов «Чтение» или «Запись» (на выводах RD и WR). При выполнении команд ввода или вывода микропроцессор устанавливает на линиях A0—A7 шины адресов номер порта. В результате дешифрации внешним дешифратором кода на линиях A2—A7 вырабатывается сигнал на выводе CS, определяющий ППА, с которым будет осуществляться обмен. Выводы A0 и A1 ППА подключаются обычно к одноименным линиям шины адреса для выбора конкретного порта внутри ППА.

Направление обмена задают управляющие сигналы, поступающие с шины

Т а б л и ц а П.2

A1	A0	RD	WR	CS	Направление передачи
0	0	0	1	0	Порт А → шина данных
0	1	0	1	0	Порт В → шина данных
1	0	0	1	0	Порт С → шина данных
0	0	1	0	0	Шина данных → порт А
0	1	1	0	0	Шина данных → порт В
1	0	1	0	0	Шина данных → порт С
1	1	1	0	0	Шина данных → РУС
X	X	X	X	1	Шина данных → 3 состояние
X	X	1	1	X	Шина данных → 3 состояние
1	1	0	1	0	Недопустимая комбинация

управления на входы \overline{WR} или \overline{RD} ППА, в зависимости от того, выполняется ли команда ввода или вывода (табл. П.2).

Наряду с портами А, В и С ППА содержит также порт регистра управляющего слова РУС, куда для задания режимов работы каналов по команде вывода

Таблица П. 3.

[illegible]

предварительно заносится управляющее слово. Формат этого слова представлен в табл. П.3. Отметим, что считывать информацию из порта РУС нельзя.

Возможны три режима (0, 1 и 2) работы каналов ППА. Режим 0 предусматривает обмен данными с внешними устройствами через каналы А, В и два подканала С без управляющих сигналов о готовности к работе и подтверждения обмена. Это означает, что данные, выводимые из микроЭВМ по командам вывода, фиксируются в соответствующих портах ППА и хранятся там до поступления очередных команд вывода, т. е. до записи в порты новых данных. По линиям каналов они поступают во внешние устройства.

При вводе из внешних устройств данные не фиксируются в портах, а считываются непосредственно в аккумулятор микропроцессора из соответствующего порта при выполнении команды ввода. Поэтому изменения входных данных в интервалах времени между обращениями микропроцессора к портам никак не отражаются на работе микроЭВМ.

Обычно режим 0 используют для ввода медленно меняющихся данных или каких-либо постоянных значений. При выводе в этом режиме на линиях каналов можно программно формировать сигналы, соответствующие заданным временным диаграммам. Иными словами, этот режим позволяет программно управлять разнообразными внешними устройствами. Именно в режиме 0 работают каналы ППА, через которые в РК подключены контакты клавиатуры и входы и выходы магнитофона.

В режиме 1 обменом данными между внешними устройствами и каналами А и В (или одним из них) управляют сигналы, передаваемые по линиям канала С (эти сигналы называют сигналами квитирования).

Работу ППА в этом режиме рассмотрим на примере совместной работы микроЭВМ с широко распространенным в вычислительной технике внешним устройством — алфавитно-цифровым дисплеем. Как правило, дисплей состоит из двух функционально независимых частей: блоков клавиатуры и вывода данных на экран дисплея.

Обратимся к рис. П.2, на котором показана схема подключения этих блоков к ППА. Необходимо отметить, что эта схема здесь приводится только как пример, так как в РК дисплей реализован по-другому и является его составной частью. В нашем примере порты А и В запрограммированы в режиме 1 соответственно на ввод данных с клавиатуры и их вывод из микропроцессора на экран дисплея. На рис. П.3 приведены временные диаграммы управляющих сигналов при вводе байта данных в ППА. После нажатия на клавишу на линиях канала А появляется код соответствующего символа, а на линии РС4 канала С — сигнал «Строб приема А» (\overline{STB}_A). По этому сигналу данные с входных линий переписываются в порт А, что подтверждается сигналом «Подтверждение приема А» (IBF_A) на выводе РС5, поступающим из ППА в дисплей. По сигналу IBF_A дисплей снимает сигнал \overline{STB}_A . При этом ППА формирует на выводе РС3 сигнал «Запрос прерывания А» ($INTR_A$), свидетельствующий о том, что данные в порту А подготовлены для последующего их считывания микропроцессором по команде ввода. Ее выполнение вызывает появление сигнала «Чтение» на выводе RD, фронт которого гасит сигнал «Запрос прерывания А», а спад — сигнал «Подтверждение приема А», после чего с клавиатуры дисплея по линиям канала А можно передать в ППА новый код символа.

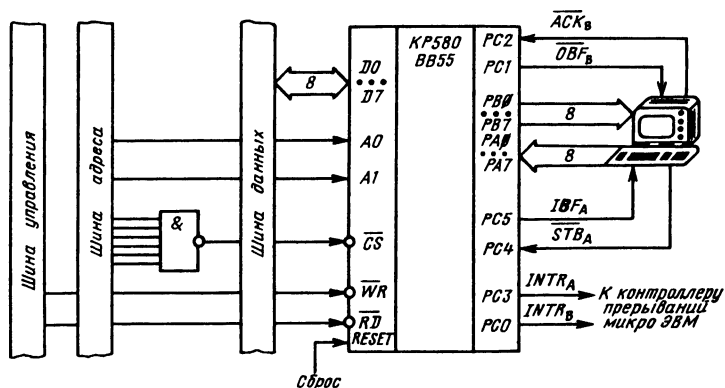


Рис. П.2. Схема подключения дисплея к микроЭВМ

При выводе на экран дисплея какого-либо символа его код переписывается из микропроцессора в порт В по команде вывода, вызывающей появление сигнала «Запись» на входе \overline{WR} ППА (рис. П.4). По спаду этого сигнала на выводе PC1 формируется сигнал «Строб записи В» (\overline{OBF}_B) для дисплея, который разрешает считывание данных с линий канала В. После считывания кода символа дисплей формирует на выводе PC2 сигнал «Подтверждение записи В» (ACK_B), который сбрасывает сигнал «Строб записи В» и инициирует в ППА на линии PC0 сигнал «Запрос прерывания В» ($INTR_B$). Теперь микропроцессор может записать в буфер ППА код очередного символа для его последующего вывода на экран.

В режиме 2 линии канала А приобретают свойство двунаправленности. Они могут подключаться к внешнему устройству, также использующему для ввода-вывода двунаправленные линии. В качестве такого внешнего устройства может, например, использоваться другая микроЭВМ с аналогичным ППА. Возможно также подключить линии канала А ППА одной микроЭВМ непосредственно к шинам другой. При работе в режиме 2 управляющие сигналы, аналогичные сигналам в режиме 1, передаются по линиям канала С. Канал В может при этом использоваться в режиме 0 или 1. Канал А отличается от других каналов тем, что ему в ППА соответствуют два регистра временного хранения данных, один из которых используется при вводе, а другой — при выводе данных. Бла-

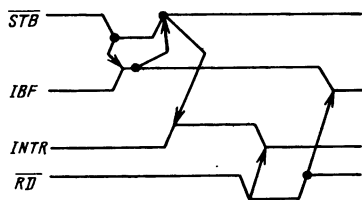


Рис. П.3. Временные диаграммы управляющих сигналов ППА при вводе данных из микроЭВМ в режиме 1

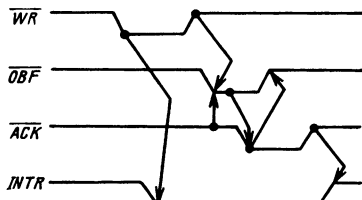


Рис. П.4. Временная диаграмма управляющих сигналов ППА при выводе данных в микроЭВМ в режиме 1

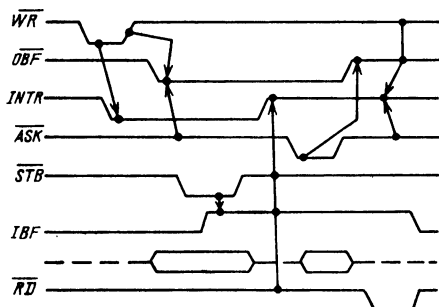


Рис. П.5. Временные диаграммы работы ППА в режиме 2

годаря этому при обмене в режиме 2 возможно одновременное хранение в ППА вводимого и выводимого байтов.

На рис. П.5 показаны временные диаграммы работы ППА в режиме 2 для случая, когда микропроцессор записывает в выводной буферный регистр порта А байт для последующего вывода. Это происходит по сигналу «Запись», поступающему по шине управления на вывод \overline{WR} ППА. По окончании его действия появляется сигнал «Строб

записи А» (\overline{OBF}_A) на выводе PC7, извещающий внешнее устройство, что ППА готов выдать байт данных. Однако в рассматриваемом на рисунке случае к моменту появления этого сигнала линии порта А оказываются заняты вводом байта данных под управлением сигналов «Строб приема А» (\overline{STB}_A), поступающего от внешнего устройства на линию PC4, и «Подтверждение приема А» (\overline{IBF}_A), вырабатываемого ППА на линии PC5. Только после ввода байта и снятия сигнала подтверждения приема внешнее устройство устанавливает на выводе PC6 сигнал «Подтверждение записи А» (\overline{ASK}_A), по которому байт, хранящийся в выводном буферном регистре порта А, переписывается на внешнее устройство.

Таким образом, состоянием линий порта А в режиме 2 управляют сигналы \overline{STB}_A и \overline{ASK}_A , поступающие от внешнего устройства. Если оба эти сигнала имеют уровень 1, то линии порта А находятся в высокоимпедансном состоянии. Совместное нахождение обоих сигналов в нулевом состоянии не допускается.

Одновременно с пересылкой байта из ППА во внешнее устройство по команде ввода микропроцессор может считать ранее введенный в порт А байт.

Сигнал «Запрос прерывания А» (\overline{INTR}_A) на выводе PC3 в режиме 2 возникает при выводе данных, если процесс обмена с внешним устройством закончен, т. е. выводной регистр порта А пуст. При вводе этот сигнал появляется, если данные уже введены из внешнего устройства в ППА, а команда ввода данных от микропроцессора из порта А отсутствует.

В табл. П.4 показано назначение линий канала С при разных режимах работы каналов ППА. Эта таблица необходима при составлении принципиальных электрических схем, так как определяет назначение выводов ППА, используемых для передачи управляющих сигналов в режимах 1 и 2. Свободные от управляющих сигналов линии канала С (в таблице они обозначены «Ввод-вывод») могут быть использованы как линии ввода или вывода данных. Для вывода данных по отдельным линиям канала С используют специальное управляющее слово, записываемое в порт PUC и служащее для индивидуальной установки в 1 или 0 любого разряда порта С (табл. П. 5). В зависимости от кода, записанного в разрядах D1 — D3, выбирается разряд порта С, значение которого (следовательно, и сигнала на соответствующей линии) будет изменено при записи этого слова

Линия порта	Режим 1		Режим 2
	ввод	вывод	
PC0	INTR B	INTR B	Ввод-вывод
PC1	IBF B	OBF B	Ввод-вывод
PC2	STB B	ACK B	Ввод-вывод
PC3	INTR A	INTR A	INTR A
PC4	STB A	Ввод-вывод	STR A
PC5	IBF A	Ввод-вывод	IBF A
PC6	Ввод-вывод	ACK A	ASK A
PC7	Ввод-вывод	OBF A	OBF A

Управляющие сигналы, передаваемые по линиям канала С, фиксируются в соответствующих разрядах порта С (табл. П. 6). Байт, считанный микропроцессором из порта С, отражает текущее состояние ППА и может быть затем программно проанализирован. Одновременно могут быть считаны и данные, вводимые или выводимые по свободным линиям канала С.

Таблица П.5

[illegible]

выполнению программы обмена байтом с соответствующим внешним устройством. Во-вторых, сигналы запросов прерывания можно подать на контроллер прерывания микроЭВМ. В этом случае при их появлении микропроцессор прерывает выполнение основной программы и начинает выполнять программу обслуживания внешнего устройства.

Т а б л и ц а П.6

D7	D6	D5	D4	D3	D2	D1	D0
<i>Ввод в режиме 1</i>							
Ввод-вывод	Ввод-вывод	IBF A	INTE A	INTR A	INTE B	IBF B	INTR B
<i>Вывод в режиме 1</i>							
$\overline{OBF} A$	INTE A	Ввод-вывод	Ввод-вывод	INTR A	INTE B	OBF B	INTR B
<i>Режим 2</i>							
$\overline{OBF} A$	INTE 1	IBF A	INTE 2	INTR A	Используются в зависимости от режима канала B		

Для разрешения или запрещения формирования запросов прерывания в ППА необходимо предварительно устанавливать определенные разряды порта C (в табл. П.6 они названы INTE_A и INTE_B для режима 1 и INTE₁ или INTE₂ при вводе или выводе в режиме 2) соответственно в 1 или 0. Это позволяет программным путем разрешить или запретить отдельным внешним устройствам работать с микропроцессором.

В табл. П.7 представлена программа ввода данных с клавиатуры в аккумулятор микропроцессора и последующего их вывода на экран дисплея. В ней

Т а б л и ц а П.7

Адрес	Код	Метка	Мнемоника	Операнд	Комментарий
1000	3EB4		MVI	A,B4H	Настройка портов
1002	D303		OUT	PYC	Порт A — ввод в режиме 1
1004	3E09		MVI	A,09H	Порт B — вывод в режиме 1
1006	D303		OUT	PYC	Установка бита D4 порта C для разрешения сигнала INTR A
1008	3E05		MVI	A,08H	Установка бита D2 порта C для разрешения сигнала INTR B
100A	D303		OUT	PYC	Ввод символа с клавиатуры:
100C	DB02	ВВОД:	IN	ПОРТ-C	Проверка наличия символа клавиатуры. Ввод байта состояния
100E	E604		ANI	04H	Выделение и анализ разряда D3 для обнаружения сигнала INTR A;
1010	CA0C10		JZ	ВВОД	Ввод кода символа с клавиатуры
1013	DB00	IN	IN	ПОРТ-A	Сохранить код в регистре C
1015	4F		MOV	C,A	

Окончание табл. П.7

Адрес	Код	Метка	Мнемоника	Операнд	Комментарий
1016	DB02	ВЫВОД:	IN	ПОРТ-С	Вывод символа на экран:
1018	E601		ANI	01H	Проверка готовности дисплея к выводу символа на экран
101A	CA1610		JZ	ВЫВОД	Ввод байта состояния, выделение и анализ разряда D0 для обнаружения сигнала INTR В
101D	79	ПОРТ-А ПОРТ-В ПОРТ-С РУС	MOV	A,C	Вернуть код символа в регистр А
101E	D301		OUT	ПОРТ-В	Вывести символ на экран
1020	C30C10		JMP	ВВОД	Присвоение номеров портам
				00H	в соответствии с рисунком
				01H	П.2
				02H	
				03H	

программно проверяется, была ли нажата клавиша на клавиатуре и готовы ли дисплей к приему очередного байта. В данном случае последнюю проверку можно было бы и не выполнять, так как заведомо известно, что дисплей бывает готов к отображению очередного знака на экране раньше, чем оператор успевает нажать клавишу. Однако в общем случае, например при выводе на экран данных, подготовленных в памяти (это весьма быстрый процесс), такая проверка необходима.

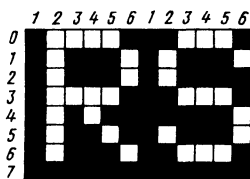
При начальном запуске из микроЭВМ на соответствующий вход ППА должен быть подан сигнал «Сброс» (RESET). По этому сигналу все порты ППА настраиваются на выполнение операций ввода в режиме 0 и их разряды обнуляются. Только после этого можно задавать нужные режимы работы. Обнуление происходит также и при операциях смены режимов работы ППА.

При конструировании различных устройств с ППА необходимо помнить, что нагрузочная способность его линий позволяет подключать к ним только по одному входу ТТЛ микросхем.

3. Контроллер электронно-лучевого прибора КР580ВГ75

Прежде чем перейти к описанию контроллера электронно-лучевого прибора (ЭЛП), рассмотрим особенности формирования алфавитно-цифровых символов на экране растровых дисплеев.

Символы на экране ЭЛП формируются путем засветки отдельных точек телевизионного раstra и располагаются в фиксированных позициях — знако-местах. Под каждое знакоместо отводится определенное число строк телевизионного раstra, а в пределах одного знакоместа на каждой строке в зависимости от отображаемого символа может быть засвечено несколько точек. Максимальное число таких точек, а также число телевизионных строк, отведенных на одно знакоместо, определяют формат отображаемых символов. По горизонтали знако-места обычно расположены вплотную друг к другу, образуя ряды знакомест.



Режим 0
 0 0 0
 0 0 1
 0 1 0
 0 1 1
 1 0 0
 1 0 1
 1 1 0
 1 1 1

Режим 1
 1 1 1
 0 0 0
 0 0 1
 0 1 0
 0 1 1
 1 0 0
 1 0 1
 1 1 0

Например, для формирования символов на экране можно использовать матрицу точек форматом 8×6 (рис. П.6). Это означает, что каждое знакоместо занимает на экране восемь строк телевизионного растра, а в

Рис. П.6. Представление сигналов на экране в виде матрицы

пределах знакоместа в строке растра может быть засвечено до шести точек. В приведенном примере точки в строке 7 и столбцах 1 в отображении алфавитно-цифровых символов не участвуют, благодаря чему на экране образуются промежутки между словами.

Для формирования символов на экране ЭЛП в схемах алфавитно-цифровых дисплеев обычно присутствует так называемое ПЗУ знакогенератора. В нем хранится информация о графическом представлении отображаемых символов. Засветка точек (модуляция луча ЭЛП) в алфавитно-цифровых дисплеях обычно происходит по сигналам с выхода сдвигового регистра, в который предварительно в параллельной форме заносится код из ПЗУ знакогенератора.

Частота, с которой осуществляется сдвиг информации в сдвиговом регистре, определяется числом точек в строке в пределах знакоместа, числом знакомест в ряду и временем развертки одной строки телевизионного растра. Интервал времени, за который луч на экране ЭЛП проходит через одно знакоместо, определяет период сигнала «Символьная синхронизация» $T_{\text{ССЛК}}$.

Применительно к примеру на рис. П.6 (т. е. к формату матрицы 8×6) каждому отображаемому символу в ПЗУ знакогенератора соответствует группа из восьми последовательно расположенных ячеек памяти, в которых записана информация о том, какие шесть точек в каждой из восьми строк растра в пределах знакоместа должны засвечиваться при отображении соответствующего символа. Нулевое значение бита в ячейке памяти знакогенератора определяет местоположение засвечиваемой точки на экране в пределах знакоместа. Фрагмент содержимого ПЗУ знакогенератора приведен в табл. П.8. Если сигнал с выхода сдвигового регистра проинвертировать, то на экране будут отображаться символы в негативном виде — темные точки на светлом фоне.

Выбор группы ячеек знакогенератора, соответствующей текущему отображаемому символу, определяется кодом последнего. Семиразрядный код символа подается на адресные линии АЗ—А9 ПЗУ знакогенератора. На линии А0—А2 подается код с двоичного счетчика номеров строк в пределах знакомест, определяющий, из какой ячейки знакогенератора в выбранной группе будет считан код в сдвиговый регистр. В течение развертки одной строки растра состояние счетчика номеров строк в пределах знакомест остается неизменным, в то время как коды символов на адресных линиях АЗ—А9 ПЗУ знакогенератора могут меняться в зависимости от отображаемых символов при достижении лучом ЭЛТ каждого нового знакоместа. Изменение кодов символов происходит под воздействием сигналов «Символьная синхронизация».

При развертке следующей строки телевизионного растра код с выхода счетчика увеличивается на 1, а на адресных линиях АЗ—А9 вновь повторяется та

Таблица П.8

Адреса ячеек ПЗУ знакогенератора		Коды с выхода ПЗУ	
А	Б		
01010010	000	100001	Группа ячеек с информацией о графическом представлении символа R
01010010	001	101110	
01010010	010	101110	
01010010	011	100001	
01010010	100	101011	
01010010	101	101101	
01010010	110	101110	
01010010	111	111111	Группа ячеек с информацией о графическом представлении символа S
01010011	000	110001	
01010011	001	101110	
01010011	010	101111	
01010011	011	110001	
01010011	100	111110	
01010011	101	101110	
01010011	110	110001	
01010011	111	111111	

Примечание. А — коды символов; Б — номера строк раstra в пределах знакомест.

же последовательность кодов, что и при развертке предыдущей строки. Этот процесс повторяется, пока не завершится развертка всех строк телевизионного раstra для ряда знакомест, после чего счетчик номеров строк раstra в пределах знакомест будет установлен в состояние нуль, а на адресные линии А3–А9 начнут поступать коды для отображения символов в следующем ряду знакомест. В табл. П.8 в качестве примера показано соответствие адресов групп ячеек ПЗУ знакогенератора, где хранится информация для отображения символов R и S, кодам этих символов. Символы R и S кодируются семиразрядными кодами 1010010 и 1010011 соответственно.

В микропроцессорных дисплеях каждому знакоместу на экране ЭЛП соответствует определенная ячейка ОЗУ в экранной области ОЗУ. Микропроцессор может работать с экранной областью так же, как и с любой другой областью памяти. Для вывода символа на экран на определенное знакоместо микропроцессор должен записать его код в ячейку экранной области ОЗУ.

Чтобы изображение постоянно присутствовало на экране ЭЛП, необходимо периодически в течение развертки каждого телевизионного раstra выдавать последовательно все коды экранной области на адресные входы ПЗУ знакогенератора. Эту и ряд других функций и выполняет БИС контроллера ЭЛП КР580ВГ75.

В составе БИС КР580ВГ75 содержится два 8-разрядных регистра, в каждом из которых может храниться по 80 кодов символов для последующего их вывода

на экран. В процессе работы, в то время как символы, хранимые в одном из буферных регистров, последовательно отображаются на экране дисплея, в другой буферный регистр переписываются из экранной области ОЗУ следующие символы. По окончании отображения символов из одного буферного регистра начинается процесс отображения из другого, т. е. регистры функционально меняются местами. Символы из экранной области в буферные регистры контроллера ЭЛП пересылаются методом ПДП. Поэтому работа контроллера ЭЛП, как правило, осуществляется совместно с контроллером ПДП КР580BT57.

Назначение выводов микросхем показано в табл. П.9. Контроллер ЭЛП формирует сигнал «Запрос ПДП» на выводе DRQ в течение обратного хода кадровой развертки раньше окончания кадрового гасящего импульса на период, равный длительности отображения одного ряда знакомест. В соответствии с этим запросом контроллер ПДП подготавливает систему к работе в режиме ПДП путем подачи сигнала на вход HOLD микропроцессора, который в ответ на него переводит свои шины в высокоимпедансное состояние, о чем и уведомляет контроллер ПДП сигналом HLDA. После этого контроллер ПДП устанавливает на выводе DACK контроллера ЭЛП сигнал «Подтверждение ПДП». При наличии этого сигнала и при появлении сигнала «Запись ввода-вывода» \overline{IOW} от контроллера ПДП на выводе \overline{WR} контроллера ЭЛП байт с шины данных переписывается в буферный регистр. Сигнал «Запись ввода-вывода» вырабатывается контроллером ПДП для каждого очередного байта. Предварительно контроллер ПДП выдает на шину адреса адрес очередной ячейки экранной области ОЗУ и на шину

Таблица П.9

Номер вывода	Обозначение вывода		Наименование сигнала	Функциональное назначение сигнала
40	V_{CC}	Вход	Напряжение питания +5 В	Примечание. Максимальный потребляемый ток 160 мА
20	GND		Общий	

Сигналы подключения к системным шинам

12...19	DB0—DB7	Двухнаправленная шина данных	Линии шины данных	Прием команд, параметров, кодов и признаков в режиме ПДП. Выдача содержимого внутренних регистров контроллера. Переходит в высокоимпедансное состояние, если $WR \& RD = 1$ или $CS = 1$
22	\overline{CS}	Вход	Выбор микросхемы	Разрешает считывание или запись байта
21	A0	»	Адрес порта	Если A0=0, то разрешен обмен с регистрами параметров; если A0=1, то разрешено считывание из регистра состояния или запись в регистр команд
10	\overline{WR}	»	Запись	Запись команд, параметров в программном режиме. В режиме

Продолжение табл. П.9

Номер вывода	Обозначение вывода	Наименование сигнала	Функциональное назначение сигнала
9	\overline{RD}	Вход	Считывание
<i>Сигналы управления ПДП</i>			
31	IRQ	Выход	Запрос прерывания
5	DRQ	»	Запрос ПДП
6	DACK	Вход	Подтверждение ПДП
<i>Сигналы управления ПЗУ знакогенератора</i>			
23...29	CC0—CC6	Выходы	Код отображаемого символа
1...4	C3—C0	»	Счетчик строк растра в пределах знакоместа
<i>Сигналы управления разверткой и видеосигналом</i>			
38, 39	LA1—LA0	Выходы	Признаки псевдографических символов

Окончание табл. П.9

Номер вывода	Обозначение вывода		Наименование сигнала	Функциональное назначение сигнала
7	HRTC	Выход	Обратный ход по строке	Используется для генерации строчных синхросигналов
8	VRTC	»	Обратный ход по кадру	Используется для генерации кадровых синхросигналов
35	VSP	»	Гашение луча	Сигнал активен 1) если HRTC=1 или VRTC=1, 2) при затемнении верхней и нижней строк раstra в пределах знакоместа, если запрограммировано подчеркивание строки с номером больше 8, 3) при появлении кода «Конец строки символов» или «Конец экрана», 4) при недогрузке ПДП, 5) при отображении мигающих курсора или символа с частотами, равными соответственно 1/16 и 1/32 частоты кадровой развертки
37	LTEN	Выход	Разрешение видеосигнала	Активен в момент отображения курсора в виде подчеркивания, элементов псевдографики и символов с соответствующими признаками
36	RVV	»	Инвертирование видеосигнала	Активен при отображении алфавитно-цифровых и псевдографических символов с соответствующими признаками
33, 34	GPA0—GPA1	Выходы	Универсальные признаки	Активны при появлении соответствующих признаков. Служат для реализации каких-либо дополнительных функций, например, отображения многоцветных символов на экране ЭЛП цветного изображения
32	HLGT	Выход	Подсветка	Используется для увеличения яркости свечения символов при соответствующем признаке
<i>Прочие сигналы</i>				
30	CCLK	Вход	Символьная синхронизация	Период следования сигналов, определяется временем прохождения луча ЭЛП в пределах знакоместа. Минимальный период следования 320 нс
11	LPEN	То же	Световое перо	Активизирует запоминание в регистрах светового пера контроллера номера знакоместа в ряду и номера ряда. Используется для определения положения светового пера на экране

управления сигнал «Чтение памяти», инициирующий вывод данных из ОЗУ на шину данных.

Последовательность символов, переписываемую в буферный регистр контроллера дисплея за время действия сигнала «Запрос ПДП», называют пакетом. Длительность этого сигнала зависит от числа символов в пакете, которое задается программно при начальной инициализации контроллера ЭЛП и может быть равна 1, 2, 4 или 8. Пакеты следуют друг за другом с интервалом времени, который может быть запрограммирован от 0 до $55 T_{CLK}$. По истечении каждого такого интервала контроллер ЭЛП вновь выдает сигнал «Запрос ПДП». Сигналы «Запрос ПДП» будут следовать, пока буферный регистр контроллера дисплея не заполнится (заполнение буфера может произойти и до окончания передачи пакета). Число символов в пакете и интервалы времени между ними задаются из системных соображений. Например, если процесс загрузки кодов символов в контроллер ЭЛП одновременно используется в системе для регенерации динамического ОЗУ, то временные параметры ПДП выбираются исходя из условий регенерации памяти.

Контроллер ПДП при работе с контроллером ЭЛП обычно программируется на работу в режиме с автозагрузкой. При этом используется канал 2, параметры которого каждый раз автоматически восстанавливаются после окончания процесса пересылки всего содержимого экранной области ОЗУ. Однако бывают особые случаи, когда по окончании отображения кадра требуется перезагрузка контроллера ПДП новыми параметрами, например при необходимости быстрого сдвига всего текста на экране на несколько строк вверх или вниз. В этом случае контроллер ЭЛП может быть запрограммирован на формирование в конце каждого кадра сигнала «Запрос прерывания». По возникновении этого сигнала программа обслуживания прерывания может изменить параметры в контроллере ПДП.

Необходимо отметить, что использование режима ПДП для обслуживания контроллера ЭЛП снижает общую производительность микропроцессорной системы. Оценку доли времени, используемой для обслуживания контроллера, нужно проводить, исходя из числа знакомест на экране ЭЛП, частоты смены кадров и времени, затрачиваемого на передачу одного байта в режиме ПДП (при использовании контроллера ПДП KP580BT57 передача первого байта в пакете осуществляется за пять-шесть машинных тактов, а последующих — за четыре).

Как уже указывалось, в буферные регистры контроллера ЭЛП заносятся 8-разрядные коды. Однако контроллер рассчитан только на отображение символов, кодируемых 7-разрядными кодами, поэтому восьмой разряд кодов отображаемых символов должен быть равен нулю. Наличие единицы в восьмом разряде свидетельствует, что данный код является либо специальным кодом, либо визуальным признаком алфавитно-цифровых символов, либо признаком псевдографических символов.

Специальные коды (табл.П.10, П.11) позволяют более экономно использовать временные ресурсы и память микропроцессора, а также в некоторых случаях упростить программирование. Так, коды «Конец строки символов» и «Конец экрана» позволяют программисту не заботиться о стирании остатков прежнего текста на конце строки и экрана. Эти коды активизируют сигнал на выходе VSP

Таблица П.10

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	0	0	S	S

Примечание. 00 — «Конец строки символов»; 01 — «Конец строки символов — стоп ПДП»; 10 — «Конец экрана»; 11 — «Конец экрана — стоп ПДП».

контроллера. С помощью внешних микросхем этот сигнал используется для затемнения изображения, после того как в экранной области ОЗУ встретятся специальные коды.

Специальные коды «Конец строки символов — стоп ПДП» и «Конец экрана — стоп ПДП» приостанавливают режим ПДП, чтобы не считывать из экранного ОЗУ коды символов, которые не будут затем отображены. Если эти коды не являются последними в строке символов или в пакете, то дополнительно в буфер контроллера ЭЛП считывается следующий за специальным кодом символ.

Часто на экране необходимо визуальнo выделять один или несколько символов подчеркиванием или путем отображения их с повышенной яркостью, в негативном виде или мигающими. Для этого в схеме дисплея необходимо иметь дополнительные элементы, которые реализуют эти функции и управляются с помощью сигналов от контроллера ЭЛП.

Таблица П.11

D7	D6	D5	D4	D3	D2	D1	D0
1	0	U	R	G1	G0	B	H

Примечание. H=1 — подсветка (сигнал на выходе HLGT); B=1 — мигание символа (периодически появляется сигнал на выходе VSP); G₀=1 — универсальный (сигнал на выходе GPA0); G₁=1 — универсальный (сигнал на выходе GPA); P=1 — инвертирование видеосигнала (сигнал на выходе RVV); U=1 — подчеркивание (сигнал на выходе LTEN).

Перед символом или группой символов, которые необходимо визуальнo выделить на экране, в экранной области ОЗУ записывают соответствующий код визуального признака. Действие этого признака распространяется на все символы, следующие вслед за ним до конца экрана или до появления в экранной области ОЗУ нового кода признака. Возможно использование следующих признаков: мигания, подсветки, отображения символа в негативном виде, подчеркивания. Эти признаки активизируют выходные сигналы соответственно на выходах VSP, HLGT, RVV и LTEN контроллера. Имеется также возможность с помощью еще двух универсальных признаков выдать сигналы на выходе GPA0 и GPA1. Назначение этих сигналов произвольное и может дополнительно определяться разработчиками дисплея.

В зависимости от того, как запрограммирован контроллер ЭЛП, признаки, встречающиеся в экранной области ОЗУ, могут быть не отображены вообще или отображены в виде пробела. Выбор одного из этих режимов определяет, будет ли всегда соблюдаться однозначное соответствие адресов экранной области ОЗУ положению знакомест на экране ЭЛП (когда признаки отображаются в виде пробелов) или же такое соответствие будет динамически изменяться в соответствии с числом и расположением признаков в экранной области ОЗУ (в случае, когда признаки не отображаются).

При передаче из экранной области ОЗУ в контроллер код признака записывается в один из его 80-символьных буферных регистров. Но при этом следующий за признаком алфавитно-цифровой символ будет записан не в этот буферный регистр, а в специальный 7-разрядный стековый регистр. Таких регистров в контроллере два — по числу буферных регистров. В каждый из стековых регистров может быть записано не более 16 кодов символов. Стек организован по принципу «Первый вошел — первый вышел», поэтому при выводе символов на экран из буферного регистра в случае появления среди них кода признака последний не отображается, а вместо него отображается очередной код символа из стекового регистра или пробел.

Такая организация контроллера накладывает следующие ограничения на использование кодов признаков. Во-первых, их не должно быть более 16 в одном ряде знакомест. Во-вторых, нельзя допускать, чтобы два признака непосредственно следовали друг за другом — в этом случае 8-разрядный код второго признака будет отправлен в стековый регистр, где он «потеряет» старший бит и в дальнейшем будет ошибочно отображаться как символ.

Контроллер ЭЛП позволяет под управлением кодов псевдографических символов (табл. П.12) отображать на экране комбинации вертикальных и горизонтальных линий. В правой части табл. П.12 показаны элементы, из которых состоят такие изображения. Каждый элемент — псевдографический символ, так же как и алфавитно-цифровой символ, располагается в пределах знакоместа.

Для отображения псевдографических символов вместо ПЗУ знакогенератора используется специальная дополнительная схема (рис. П.7). При появлении хотя бы на одном из выходов LA1, LA0 контроллера напряжения высокого уровня выходы ПЗУ знакогенератора отключаются от входов сдвигового регистра D7. В зависимости от комбинации на выходах LA1, LA0, VSP и LTEN отображается один из псевдографических символов. Каждое знакоместо при его отображении можно рассматривать как состоящее из трех частей — средней части в виде линии подчеркивания и частей, расположенных над и под линией подчеркивания. Для отображения каждой из этих частей знакоместа на выходы LA1, LA0, VSP и LTEN выдается соответствующий код, поэтому одному коду псевдографических символов соответствуют три кодовых комбинации. У выходов элементов D4.1 — D4.3 на рис. П.7 условно показаны те части символов, которые формируются на экране при их активизации. Изображение полной горизонтальной линии в середине знакоместа происходит вследствие появления сигнала на выходе LTEN.

Каждый псевдографический символ может быть визуально выделен на экране повышенной яркостью или миганием с помощью задания значения двух младших битов в коде псевдографического символа. На визуальное выделение псевдографических символов влияют также визуальные признаки алфавитно-цифровых

Таблица П.12

D7	D6	D5	D4	D3	D2	D1	D0
I	I	C	C	C	C	B	H

Биты CCCC	Сигналы на выходах				Отобра- жаемый символ	Биты CCCC	Сигналы на выходах				Отобра- жаемый символ	
	LA ₁	LA ₀	VSP	LTEN			LA ₁	LA ₀	VSP	LTEN		
0000	0	0	1	0		0110	0	1	0	0		
	1	0	0	0			1	0	0	0		
	0	1	0	0			0	1	0	0		
	0	0	1	0			0	1	0	0		
0001	1	1	0	0		0111	0	0	0	1		
	0	1	0	0			0	0	1	0		
	0	1	0	0			0	0	1	0		
	0	1	0	0			0	0	1	0		
0010	1	0	0	0		1000	0	0	0	1		
	0	0	1	0			0	0	1	0		
	0	0	1	0			0	0	1	0		
	0	0	1	0			0	0	1	0		
0011	1	1	0	0		1001	0	0	1	0		
	0	1	0	0			0	1	0	0		
	0	1	0	0			0	1	0	0		
	0	1	0	0			0	1	0	0		
0100	0	0	0	1		1010	0	1	0	0		
	0	1	0	0			0	1	0	0		
	0	1	0	0			0	1	0	0		
	0	1	0	0			0	1	0	0		
0101	1	1	0	0								
	0	1	0	0								
	0	1	0	0								
	0	1	0	0								

Примечание. H-1 — подсветка (сигнал на выходе HLGT); B=i — мигание символа (периодически появляется сигнал на выходе VSP); CCCC — код псевдографического символа. Остальные комбинации для псевдографических символов не используются.

символов — признак отображения символа в негативном виде и универсальные признаки.

При работе с дисплеем необходимо помечать то знакоместо, куда будет введен очередной символ. Это осуществляется с помощью специального знака — курсора. БИС контроллера ЭЛП позволяет запрограммировать вывод курсора на экран в одном из четырех видов: мигающей линии подчеркивания; мигающего символа (или знакоместа) в негативном виде; немигающей линии подчеркивания; немигающего алфавитно-цифрового символа (или знакоместа) в негативном виде. Формирование курсора осуществляется с помощью дополнительных схемных элементов, управляемых сигналами VSP, RVV и LTEN.

Контроллер ЭЛП позволяет подключать к дисплею световое перо для непосредственного указания на экране какого-либо знакоместа. Световое перо обычно

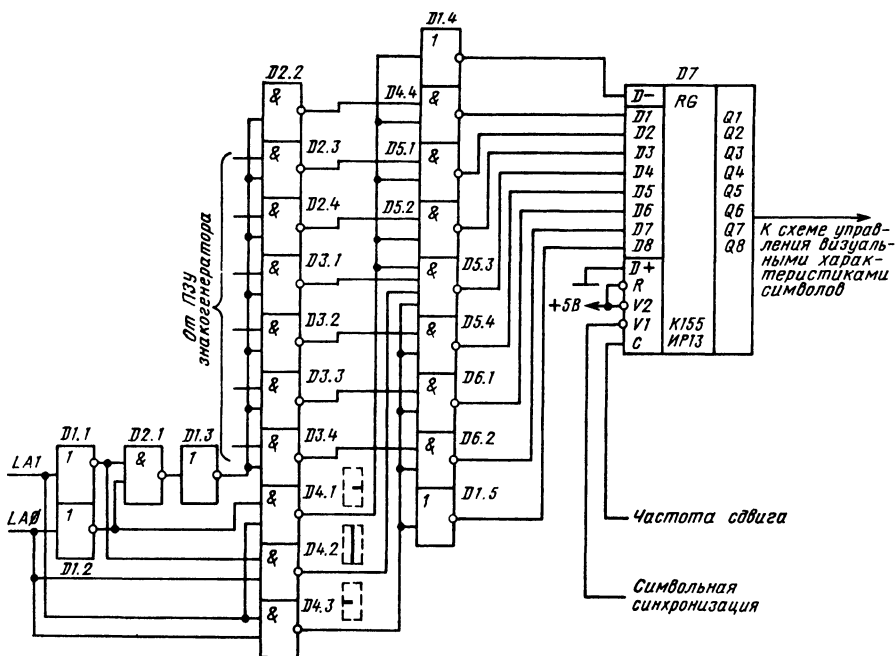


Рис. П.7. Схема отображения псевдографических символов

представляет из себя небольшой фотодатчик, который прижимается к поверхности экрана. При этом микропереключатель, вмонтированный в корпус датчика, замыкает цепь и фотодатчик через устройство формирования сигнала оказывается подключенным к входу LPEN контроллера. Когда луч ЭЛП при развертке попадает на фотодатчик, то последний вырабатывает сигнал, по которому в регистрах светового пера контроллера запоминаются координаты положения засветившего датчик символа. Состояние этих регистров может быть считано по команде «Считывание положения светового пера». Из-за задержек прохождения сигнала от светового пера в контроллере и внешних схемах координаты фотодатчика считываются с ошибкой (более чем на три знакоместа). Это должно быть скорректировано программным обеспечением.

Программирование осуществляется записью команд и параметров во внутренние регистры команд и параметров. Всего имеется восемь команд, некоторые из них требуют ввода дополнительных параметров (табл. П.13). Назначение параметров команд «Формат экрана» и «Разрешение отображения»:

Таблица П.13

Команда, параметр	A0	D7	D6	D5	D4	D3	D2	D1	D0
Формат экрана	1	0	0	0	0	0	0	0	0
Параметр 1	0	S	H	H	H	H	H	H	H
Параметр 2	0	V	V	P	P	P	P	P	P

Продолжение табл. П.13

Команда, параметр	A0	D7	D6	D5	D4	D3	D2	D1	D0
Параметр 3	0	U	U	U	U	L	L	L	L
Параметр 4	0	M	Г	C	C	Z	Z	Z	Z
Разрешение отображе- ния	1	0	0	1	S	S	S	B	B
Запрет отображения	1	0	1	0	0	0	0	0	0
Управление курсором	1	1	0	0	0	0	0	0	0
Параметр 1	0	Номер знакоместа в ряду							
Параметр 2	0	Номер ряда знакомест							
Разрешение прерыва- ния	1	1	0	1	0	0	0	0	0
Запрещение прерыва- ния	1	1	1	0	0	0	0	0	0
Предварительная уста- новка счетчиков	1	1	1	1	0	0	0	0	0
Считывание положе- ния светового пера	1	0	1	1	0	0	0	0	0
Параметр 1	0	Номер знакоместа в ряду							
Параметр 2	0	Номер ряда знакомест							

Примечание. Параметры всех команд загружаются в контроллер непосредственно вслед за загрузкой соответствующих команд. Исключение составляет команда «Считывание положения светового пера», параметры 1 и 2 которой считываются из контроллера непосредственно вслед за загрузкой команды.

Отображение рядов знаков мест — бит S

0 Отображаются все ряды

1 Отображаются только четные ряды

Число знакомест в ряду — биты NNNNNN

0000000 1

0000001 2

0000010 3

... ..

1001111 80

Остальные кодовые комбинации не используются

Длительность обратного хода кадровой развертки — биты

00 1T

01 2T

10 3T

11 4T

T — время отображения одного ряда знакомест

Число рядов знакомест в кадре — биты RRRRRR

000000 1

000001 2

000010 3

... ..

111111 64

Номер линии подчеркивания — биты UUUU

0000 1

0001 2

0010 3

... ..

1111 16

Число строк раstra в знакоместе — биты LLLL		
0000	1	
0001	2	
0010	3	
...	...	
1111	16	
Режим счета строк раstra в знакоместе — бит M		
0	Режим без смещения	
1	Режим со смещением на 1	
Режим отображения признаков — бит F		
0	В виде пробела	
1	Не отображаются	
Формат курсора — биты CC		
00	Мигающий символ	
01	Мигающая линия подчеркивания	
10	Немигающий символ в негативном виде	
11	Немигающая линия подчеркивания	
Длительность обратного хода строчной развертки — биты ZZZZ		
0000	2 T _{CCLK}	
0001	4 T _{CCLK}	
0010	6 T _{CCLK}	
...	...	
1111	32 T _{CCLK}	
Время между запросами ПДП — биты SSS		
000	0 T _{CCLK}	
001	7 T _{CCLK}	
010	15 T _{CCLK}	
011	23 T _{CCLK}	
100	31 T _{CCLK}	
101	39 T _{CCLK}	
110	47 T _{CCLK}	
111	55 T _{CCLK}	
ПДП — биты BB		
Число байт в пакете ПДП — биты BB		
00	1	
01	2	
10	4	
11	8	

Вслед за командой «Считывание положения светового пера» требуется считывать (не записывать!) два параметра, следующих непосредственно друг за другом.

Поясним режим счета строк раstra в знакоместе. Режим без смещения применяется для ПЗУ знакогенераторов, в которых начальные ячейки в группах с информацией о графическом представлении символов используются для затемнения строки. Если же в этих ячейках записана информация об отображаемом символе, то используется режим счета со смещением на единицу (см. рис. П.6). Рассмотрим действия, выполняемые командами.

Команда «Формат экрана» прекращает процесс ПДП (прохождение сигнала «Запрос прерывания»), сигнал на выходе VSP используется для затемнения экрана. Сигналы на выходах HRTC и VRTC находятся в активном состоянии.

Команда «Разрешение отображения» разрешает прохождение сигналов «Запрос прерывания» и «Запрос ПДП». При этом устанавливаются соответствующие биты в слове состояния и разрешается появление изображения на экране.

Команда *«Конец отображения»* запрещает прохождение видеосигнала. Прохождение сигналов *«Запрос прерывания»* и *«Запрос ПДП»* продолжает оставаться разрешенным, сигналы обратного хода луча по строкам и кадрам продолжают проходить.

По команде *«Считывание сигнала светового пера»* из внутренних регистров светового пера контроллера считываются ранее записанные координаты пера на экране — номер знакоместа и номер ряда.

Команда *«Загрузка положения указателя курсора»* позволяет загрузить два байта параметров — положение курсора. На регистр состояния команда не влияет.

Команды *«Разрешение прерывания»* и *«Запрещение прерывания»* соответственно устанавливают режим, разрешающий и запрещающий возникновение запросов прерывания, и управляют соответствующим битом слова состояния.

Команда *«Предустановка счетчиков»* позволяет установить счетчики знакомест в состояние, соответствующее отображению символа в верхнем углу дисплея. Для этой операции требуются два символьных синхронимпульса. Счетчики будут оставаться в таком состоянии до тех пор, пока не будет подана любая следующая команда. Команда необходима, когда в системе работают несколько контроллеров ЭЛП, работу которых необходимо синхронизировать.

Для программного анализа состояния, в котором находится контроллер ЭЛП, в процессе работы можно считывать из регистра состояния слово состояния контроллера. Назначение отдельных битов регистра состояния контроллера следующее:

D6 — разрешение прерывания. Устанавливается с помощью команд *«Разрешение прерывания»* и *«Разрешение отображения»*, а также при включении источника питания. Сбрасывается командами *«Запрещение прерывания»* и *«Формат экрана»*.

D5 — запрос на прерывание. Устанавливается в начале отображения последнего ряда знакомест кадра, если было разрешено прерывание. Сбрасывается при считывании слова состояния.

D4 — сигнал от светового пера. Сбрасывается при считывании слова состояния.

D3 — неверная команда. Устанавливается в случае несоответствия команде числа загруженных параметров или недопустимого кода. Сбрасывается считыванием слова состояния.

D2 — разрешение отображения. Устанавливается по команде *«Разрешение отображения»* и сбрасывается по командам *«Запрет отображения»* и *«Формат экрана»*.

D1 — недогрузка ПДП. Устанавливается при недогрузке данных в ходе отображения информации на экране. При появлении бита режим ПДП прекращается и экран затемняется до начала отображения нового кадра изображения. Сбрасывается при считывании слова состояния.

D0 — переполнение стека символов. Устанавливается, если число признаков в строке символов превысило 16. Сбрасывается при считывании слова состояния.

Бит D7 в регистре всегда равен нулю.

4. Контроллер ПДП КР580ИК57

В состав контроллера входят буфер шины данных, блок управления записью/чтением, блок управления ПДП, четыре канала ПДП и блок управления приоритетами каналов ПДП. Пересылка данных и управляющей информации в контроллере осуществляется по внутренней шине.

Буфер шины данных представляет собой 8-разрядный регистр, обеспечивающий двунаправленный обмен данными по восьми линиям шины данных D0—D7. По этим линиям микропроцессор загружает управляющую информацию и данные в блоки и каналы контроллера ПДП, а также считывает информацию о его состоянии. Эти же линии используются для выдачи восьми старших разрядов адреса при адресации ячейки памяти, участвующей в пересылке данных в цикле ПДП.

Загрузка управляющей информации и данных в контроллер ПДП и чтение информации о его состоянии осуществляются с помощью сигналов, поступающих от микропроцессора к блоку управления записью/чтением. В этом случае сигналы на выводах четырех младших разрядов адреса A0—A3 определяют один из внутренних регистров контроллера при наличии сигнала «Выбор микросхемы» \overline{CS} . Запись или чтение информации в/из выбранного регистра обеспечивается при поступлении сигналов «Запись во внешнее устройство» или «Чтение из внешнего устройства» на соответствующие выводы \overline{IOW} и \overline{IOR} контроллера, которые в этом случае используются как входы. Адресация внутренних регистров контроллера приведена в табл. П.14.

В режиме ПДП адресные выводы A0—A3 используются для адресации ячейки памяти, участвующей в обмене данными с внешним устройством. Выводы \overline{IOW} и \overline{IOR} при этом служат в качестве выходов для управления соответственно записью или чтением информации в/из внешнего устройства. Вход

Таблица П. 14

Регистр	Состояние входа				
	\overline{CS}	A3	A2	A1	A0
РА 0	0	0	0	0	0
РУ 0	0	0	0	0	1
РА 1	0	0	0	1	0
РУ 1	0	0	0	1	1
РА 2	0	0	1	0	0
РУ 2	0	0	1	0	1
РА 3	0	0	1	1	0
РУ 3	0	0	1	1	1
РР	0	1	0	0	0
РС	0	1	0	0	0
Нет доступа	1	*	*	*	*

Примечание. РА X — регистр адреса канала X; РУ X — регистр управления канала X; РР — регистр режима (доступен только для записи); РС — регистр состояния (доступен только для чтения); * — произвольное состояние.

\overline{CS} в этом случае автоматически блокируется для исключения ошибочной вы-
борки контроллера при выполнении пересылки данных в режиме ПДП.

Блок управления ПДП содержит узлы, обеспечивающие работу контроллера
в режиме ПДП, а также регистр режима (PP) и регистр состояния (PC).

При поступлении сигнала «Сброс» на вход RES внутренние регистры
контроллера устанавливаются в исходное состояние и тем самым запрещают
работу каналов ПДП. Для тактирования работы контроллера ПДП исполь-
зуется тактовый синхриимпульс Ф2, поступающий на вход «Синхросигнал»
(CLK). При поступлении в контроллер запроса по одному из четырех каналов
на пересылку данных в режиме ПДП блок управления ПДП формирует сигнал
«Захват» (HRQ) на одноименном выходе контроллера. Этот сигнал обеспе-
чивает перевод микропроцессора в режим «Захват», о чем свидетельствует
сигнал «Подтверждение захвата», поступающий на вход HLDA контроллера.
При наличии сигнала HLDA контроллер обеспечивает пересылку данных между
памятью и внешним устройством, выставившим запрос ПДП. При этом по
сигналу «Строб адреса» (ADD STB) старший байт адреса, выставляемый
контроллером на шине данных D0—D7, загружается во внешний буферный
регистр, откуда он по сигналу «Разрешение адреса» (AEN) передается на во-
семь старших разрядных линий шины адресов для адресации ячейки памяти,
участвующей в цикле ПДП. Младший байт адреса ячейки памяти устанавли-
вается на адресных линиях A0—A3 и A4—A7.

Помимо сигналов \overline{IOW} или \overline{IOR} соответственно для записи или чтения
байта данных из внешнего устройства контроллер формирует сигналы «За-
пись в память» \overline{MEMW} или «Чтение из памяти» \overline{MEMR} . Если быстроедействие
памяти или внешнего устройства, участвующего в цикле ПДП, ниже номи-
нального, блок управления ПДП обеспечивает увеличение длительности цикла
ПДП. Это обеспечивается с помощью сигнала «Готовность» на входе RDY. Если
на входе RDY контроллера устанавливается сигнал низкого уровня, то контроллер
переводится в состояние «Ожидание» и остается в нем так долго, пока не появит-
ся положительный сигнал на этом входе.

При пересылке блока данных на выводе MARK контроллера формируется
сигнал «Маркер» каждый раз, когда до окончания пересылки блока данных
остается выполнить число циклов ПДП, кратное 128. При пересылке последнего
байта данных контроллер извещает об окончании передачи сигналом «Конец
блока» на выводе TC. Одновременно с этим снимается сигнал «Захват», обес-
печивая окончание режима ПДП. Этот сигнал может быть использован для
прерывания микропроцессора и изменения режима работы контроллера.

Для задания режима работы контроллера ПДП служит 8-разрядный ре-
гистр режима, каждый разряд которого имеет самостоятельное значение. Управ-
ляющая информация записывается в регистр режима микропроцессором при
обращении к внешнему устройству с номером, определяемым по табл. П.14, но не
может быть считана из него. При инициализации работы микроЭВМ все разряды
регистра режима контроллера сбрасываются по сигналу RES, блокируя тем са-
мым работу каналов.

Для разрешения работы каналов контроллера в режиме ПДП разряды
PP0—PP3 регистра режима, соответствующие номерам каналов, устанавли-
ваются в состояние 1. Запретить работу того или другого канала можно с по-

мощью установки соответствующего разряда регистра режима в состояние 0. Разряд PP4 регистра режима определяет работу блока управления приоритетами. В состоянии 0 разряд PP4 обеспечивает работу контроллера с фиксированными приоритетами каналов ПДП. При этом высшим приоритетом обладает канал 0 контроллера, а низшим — канал 3.

Для того чтобы исключить возможность блокировки работы менее приоритетных внешних устройств в том случае, когда ведется пересылка данных более высокоприоритетным каналом, в контроллере предусмотрен режим работы с циклически изменяющимися приоритетами. Задание этого режима обеспечивается установкой в состояние 1 разряда PP4 регистра режима. При этом после каждого цикла ПДП приоритет каналов изменяется в последовательности... $\rightarrow 0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 0 \dots$ и т. д., т. е. канал, только что переславший байт данных между памятью и внешним устройством, приобретает низший приоритет, а следующий за ним по номеру канал — высший.

Разряд PP5 регистра режима, установленный в состояние 0, обеспечивает формирование сигналов записи номинальной длительности. Однако длительность сигналов записи может быть увеличена на период синхросигнала CLK, если разряд PP5 установлен в состояние 1. В этом случае устройства, которые формируют сигнал готовности по фронту сигнала записи, вырабатывают сигнал RDY раньше, что позволяет избежать перехода контроллера в состояние ожидания и связанных с этим задержек.

Разряд PP6 регистра режима в состоянии 1 блокирует работу каналов, завершивших пересылку всего блока данных. В этом случае по сигналу «Конец блока» соответствующие разряды PP0 — PP3 сбрасываются в состояние 0. При этом каналы выключаются и запросы ПДП от внешних устройств не воспринимаются до тех пор, пока работа каналов не будет повторно разрешена. Если в разряде PP6 установлен 0, то после пересылки всего блока данных внешнее устройство само должно блокировать формирование сигнала запроса ПДП для избежания возможности появления ошибок при пересылке данных до повторного перепрограммирования канала.

Контроллер ПДП обеспечивает возможность автоматического аппаратного перепрограммирования канала 2 при завершении пересылки всего блока данных по сигналу «Конец блока». При этом параметры режима пересылки в канал 2 без изменения переписываются из канала 3, а установленный в состояние 1 разряд PP6 не останавливает работу канала 2. Это позволяет сократить потери времени на перепрограммирование канала по сравнению с программным методом. Для того чтобы параметры режима пересылки, хранимые в канале 3, оставались неизменными, его работа должна быть запрещена установкой в состояние 0 разряда PP3 регистра режима.

Регистр состояния контроллера содержит пять разрядов, используемых для индикации выполнения контроллером определенных действий. Адресация регистра состояния обеспечивается сигналами на адресных линиях A0—A3 в соответствии с табл. П.14. При этом допускается только считывание содержимого регистра состояния. Сигнал RES сбрасывает в состояние 0 все разряды регистра.

Разряды PC0 — PC3 устанавливаются в состояние 1, когда канал с номером, равным номеру разряда, заканчивает передачу всего блока данных и возникает сигнал «Конец блока». При считывании информации из регистра состояния эти

разряды сбрасываются в исходное состояние по заднему фронту сигнала «Чтение из внешнего устройства».

Разряд PC4 устанавливается в состояние 1 в момент автоматической перезагрузки параметров из канала 3 в канал 2. В случае, если необходимо организовать по каналу 2 последовательность пересылок блоков данных с различными параметрами, загрузку новых значений параметров в канал 3 нельзя производить до тех пор, пока разряд PC4 не примет значение 0. Это свидетельствует о завершении автоматической перезагрузки параметров в канал 3. Считывание содержимого регистра состояния не оказывает влияния на разряд PC4. Однако сброс в состояние 0 разряда PP7 регистра режима сбрасывает и разряд PC4 регистра состояния.

Контроллер обеспечивает пересылку данных по четырем каналам ПДП, каждому из которых присвоен свой номер. Все четыре канала содержат 16-разрядные регистры адреса (РА) и регистры управления (РУ), адресация которых осуществляется в соответствии с табл. П.14. При инициализации работы канала в его регистр адреса загружается адрес первой ячейки памяти, с которой начинается пересылка данных в первом цикле ПДП. После каждой пересылки по каналу ПДП содержимое регистра адреса соответствующего канала увеличивается на 1. Размер пересылаемого блока данных, уменьшенный на единицу, указывают 14 младших разрядов регистра управления канала, обеспечивая пересылку максимально до 16 384 байтов. Два старших разряда регистра управления определяют тип операции пересылки в соответствии с табл. П.15.

Таблица П.15

Разряды		Операция	Формируемые контроллером сигналы
РУ 15	РУ 14		
0	0	Верификация в цикле ПДП	Нет
0	1	Чтение из памяти	<u>MEMR, IOW</u>
1	0	Чтение из внешнего устройства	<u>IOR, MEMW</u>
1	1	Запрещенная комбинация	—

Загрузка регистров адреса и управления каналов ПДП осуществляется побайтно с помощью выполнения последовательно двух команд ввода во внешнее устройство по одному и тому же адресу соответствующего регистра. Первая команда вывода обеспечивает запись в восемь младших разрядов регистра, а вторая — в старшие разряды регистра. Если порядок записи в 16-разрядные регистры контроллера по каким-либо причинам нарушен, для его восстановления достаточно выполнить запись в регистр режима, после чего можно вновь загружать регистры адреса и управления.

Для избежания неверной загрузки каналов ПДП при их программировании следует запретить возникновение прерываний в микроЭВМ. После этого сначала загружаются регистры адреса и управления каналов и в последнюю очередь загружается регистр режима контроллера. Нельзя разрешать работу каналов ПДП, если в каналные регистры предварительно не были загружены действительные значения параметров пересылки. Иначе случайные сигналы запроса ПДП от внешних устройств могут привести к изменению содержимого памяти.

Для пересылки данных в режиме ПДП внешнее устройство формирует сигнал «Запрос ПДП», поступающий на один из входов DRQ0 — DRQ3 соответствующего канала. Если работа канала разрешена, то в ответ на запрос канал формирует сигнал «Разрешение ПДП» DAC0 — DAC3. Этот сигнал служит подтверждением того, что очередной цикл ПДП отводится внешнему устройству, выставившему сигнал «Запрос ПДП». Пересылка данных между памятью и выбранным внешним устройством осуществляется до тех пор, пока поддерживается сигнал «Запрос ПДП» или не появится запрос от более высокоприоритетного внешнего устройства. Если сигнал «Запрос ПДП» поддерживается внешним устройством в течение пересылки всего блока данных, то скорость обмена в режиме ПДП максимальна. Однако в этом случае микропроцессор полностью блокирован и лишен возможности реагировать на сигналы прерывания или осуществлять программную загрузку каналов ПДП контроллера. Для исключения этого внешнее устройство, осуществляющее пересылки в режиме ПДП, может снимать на некоторое время сигнал «Запрос ПДП» после каждого цикла ПДП. При этом микропроцессор периодически продолжает выполнение программы до поступления очередного запроса ПДП от внешнего устройства.

Список литературы

1. Долгий А. Бейсик для «Радио-86РК» // Радио.—1987.— № 1.— С. 31—32.
2. Долгий А. Компьютерные игры // Радио.— 1987.— № 2.— С. 23—26, 38.
3. «Радио-86РК». Справочные таблицы (цветная вкладка) // Радио.— 1987.— № 5.
4. Долгий А. Если нет КР580ВГ75... // Радио.—1987.— № 5.— С. 22—24; 1987.— № 6.— С. 33—34.
5. Сергеев А. Еще о замене микросхем в «Радио-86РК» // Радио.—1987.— № 6.
6. Барчуков В., Зеленко Г., Фадеев Е. Редактор и Ассемблер для «Радио-86РК» // Радио.—1987.— № 7.— С. 22—26.
7. Лукьянов Д., Богдан А. «Радио-86РК» — программатор ПЗУ // Радио.— 1987.— № 8.— С. 21—23; 1987.— № 9.— С. 24—26.
8. Барчуков В., Фадеев Е. Программа-модификатор // Радио.—1987.— № 8.— С. 24.
9. Покладов А., Константинов Ю. Принимаем RTTY // Радио.— 1987.— № 10.— С. 17—20.
10. Иванов Г. Программа «Морзе-тренажер» // Радио.—1987.— № 10.— С. 21—23.

Оглавление

Предисловие	3
1. ПРИНЦИПЫ РАБОТЫ МИКРОЭВМ	5
1.1. Структура микроЭВМ и ее система команд	5
1.2. Программирование в машинных кодах	20
2. ПЕРСОНАЛЬНАЯ РАДИОЛЮБИТЕЛЬСКАЯ МИКРОЭВМ «РАДИО-86РК»	36
2.1. Принципиальная электрическая схема РК	40
2.2. Отладка РК	57
3. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РК	64
3.1. Управляющая программа Монитор	64
3.2. Интерпретатор языка Бейсик	74
3.3. Программирование на Бейсике	96
Приложение	111
Список литературы	141

Научно-популярное издание

Массовая радиобиблиотека. Вып. 1139

**Зеленко Геннадий Вадимович, Панов Виктор Васильевич,
Попов Сергей Николаевич**

ДОМАШНИЙ КОМПЬЮТЕР

Руководитель группы МРБ И. Н. Сусл ова

Редактор О. В. Воробьева

Художественный редактор Н. С. Шенн

Технический редактор Г. З. Кузнецова

Корректор Т. В. Демидович

ИБ № 1261

Сдано в набор 03.10.88. Подписано в печать 24.01.89. Т. 05019. Формат 60×88¹/₁₆. Бумага офсетная № 2. Гарнитура литературная. Печать офсетная. Усл. печ. л. 8,82. Усл. кр.-отт. 9,07. Уч.-изд. л. 11,28. Тираж 100 000 экз. (1-й завод 1—25 000 экз.). Изд. № 21246. Зак. №18
Цена 80 к.

Издательство «Радио и связь». 101000 Москва, Почтамт, а/я 693

Ордена Октябрьской Революции и ордена Трудового Красного Знамени МПО «Первая Об-
разцовая типография» Союзполиграфпрома при Государственном комитете СССР по делам
издательств, полиграфии и книжной торговли. 113054, Москва, Валовая, 28

НАШИМ ЧИТАТЕЛЯМ

Издательство «Радио и связь» книги не высылает. Литературу по вопросам радиоэлектроники и радиолюбительства можно приобрести в магазинах научно-технической книги.

Для сведения сообщаем, что по вопросам переделки и усовершенствования конструкций издательство и авторы консультацию не дают. По этим вопросам следует обращаться в письменную радиотехническую консультацию Центрального радиоклуба СССР по адресу: 103012, Москва, К-12, ул. Куйбышева, д. 4/2, пом. 12.

Издательство не имеет возможности оказать помощь в приобретении нужных вам радиотоваров и не располагает сведениями о наличии их в торгующих организациях.

Радиотовары по почте высылают Центральная торговая база Посылторга (111126, Москва, Е-126, Авиамоторная, 50) и Московская межреспубликанская база Центросоюза (127471, Москва, Г-471, ул. Рябиновая, 45).

80 к.

Мрб

Домашний
компьютер

Издательство «Радио и связь»