Микро-ЭВМ

КНИГА 8

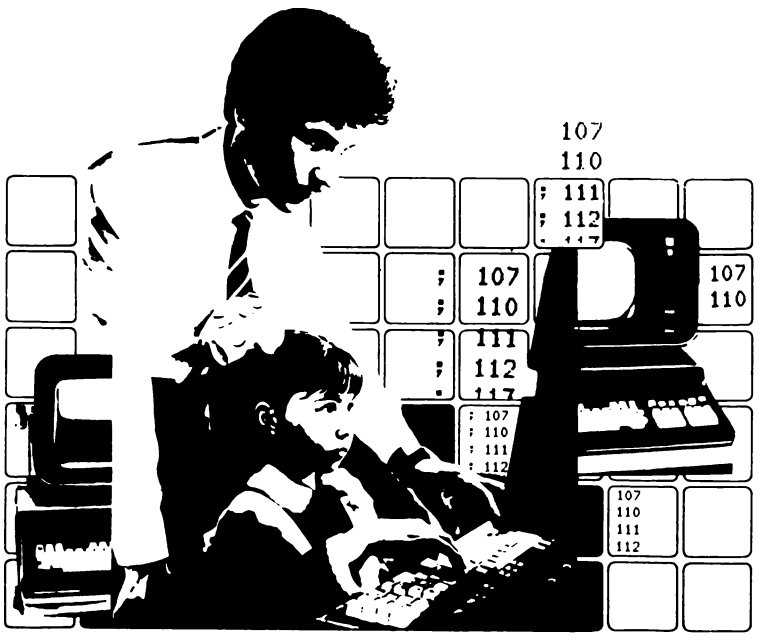
МикроЭВМ в учебных заведениях

МикроЭВМ в учебных заведениях

МикроЭВМ

В ВОСЬМИ КНИГАХ

Под редакцией
лауреата Государственной премии СССР
члена-корреспондента АН СССР
Л. Н. ПРЕСНУХИНА



МикроЭВМ в учебных заведениях

КНИГА 8

Москва «Высшая школа» 1988

ББК 32.973.2
М 59
УДК 681.322

Рекомендовано Министерством высшего и среднего специального образования СССР для использования в учебном процессе

Г. И. Фролов, В. А. Шахнов, Н. А. Смирнов

Рецензенты: кафедра «Автоматизированные системы управления» Московского высшего технического училища им. Н. Э. Баумана (зав. кафедрой — проф. В. Н. Четвериков); лауреат Государственной премии СССР канд. техн. наук Б. И. Ермолаев (Научно-исследовательский центр электронной вычислительной техники)

МикроЭВМ. В 8 кн.: Практ. пособие/Под ред.
М 59 Л. Н. Преснухина. Кн. 8. МикроЭВМ в учебных заведениях/Г. И. Фролов, В. А. Шахнов, Н. А. Смирнов. — М.: Высш. шк., 1988. — 160 с.: ил.

В пособии описаны структура, устройство, системное и прикладное программное обеспечение микроЭВМ, предназначенных для использования в учебных заведениях; рассмотрены требования к микроЭВМ; освещен первый опыт использования кабинетов вычислительной техники на основе микроЭВМ типа ДВК и бытовых компьютеров типа «Электроника БК-0010»; приведены сведения о перспективных моделях разрабатываемых микроЭВМ.

М $\frac{2405000000-524}{001(01)-88}$ **152—88**

ББК 32.973.2
6Ф7.3

Введение

Среди важнейших направлений научно-технического прогресса решающую роль играет широкое применение вычислительной техники. Высокими темпами растет численность парка ЭВМ, особенно персональных. В этих условиях знание специалистами в различных областях промышленности основных принципов работы ЭВМ и программирования приобретает определяющее значение для всей экономики.

Существует два основных аспекта применения ЭВМ в учебных целях: в качестве средства обучения, когда ЭВМ используется при изучении различных дисциплин, и в качестве предмета обучения, когда объектом изучения являются сама ЭВМ и методы программирования. ЭВМ используется также для управления учебным процессом, в исследовательских целях, для автоматизации лабораторного эксперимента, выполнения научно-технических расчетов и т. д.

Возможности использования ЭВМ в качестве средства обучения велики. ЭВМ может стать важным вспомогательным средством для изучения многих дисциплин в вузе, техникуме, профессионально-техническом училище, школе. Однако внедрение методов обучения с применением ЭВМ в широких масштабах во многом зависит от качества аппаратуры и системных программных средств, наличия подходящих пакетов обучающих программ и уровня подготовки преподавателей.

Основной целью применения ЭВМ в качестве предмета обучения является получение учащимися определенного уровня знаний о вычислительной технике и программировании, позволяющего им обращаться к ЭВМ для решения задач, с которыми они сталкиваются в учебе и повседневной жизни.

Создание технических средств и программного обеспечения, отвечающих требованиям учебного процесса, — одна из важных задач.

При написании книги авторы стремились сделать ее максимально полезной для получения практических навыков работы с персональной ЭВМ. С этой целью дается достаточно подробное описание простого входного языка программирования, сопровождаемое множеством примеров в виде машинных распечаток, полученных с помощью реального компьютера. Для чтения книги не требуется специальных знаний или других источников, хотя было бы весьма желательным выполнение единственного условия — иметь доступ к одной из микроЭВМ типа «Электроника БК-0010», «Электроника УК НЦ», ДВК (диалоговый вычислительный комплекс).

Принцип «обучения действием» особенно важен для новичков, впервые приступающих к изучению программирования. Именно на этой стадии их интересуют конкретные особенности входного языка программирования и ЭВМ, за которой они работают. Авторы считают, что первое знакомство с вычислительными машинами следует начинать с определенной ЭВМ. Опыт показывает, что после овладения одним из языков программирования переход к изучению другого не вызывает принципиальных затруднений.

Введение, гл. 1—3 написаны канд. техн. наук Г. И. Фроловым, гл. 4 — канд. техн. наук В. А. Шахновым, гл. 5 — канд. техн. наук Н. А. Смирновым.

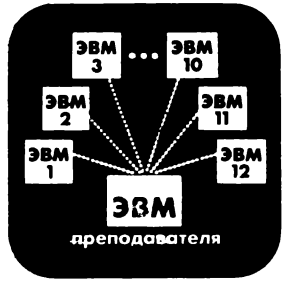
Авторы сознают сложность проблемы компьютеризации образования, а также сложность создания учебного компьютера, к которому предъявляется ряд противоречивых требований, и вполне допускают, что по ряду вопросов, изложенных в настоящей книге, могут быть и другие точки зрения. Все критические замечания и высказывания будут нами с благодарностью приняты и учтены.

Отзывы о книге просим направлять по адресу: 101430, Москва, ГСП-4, Неглинная ул., д. 29/14, издательство «Высшая школа».

Авторы

глава 1

Персональные ЭВМ в учебном заведении



Во все времена главной задачей педагога была передача знаний, накопленных человеком в ходе исторического развития. Поскольку объем знаний в наши дни растет экспоненциально, а время обучения относительно ограничено, во всем мире наблюдается тенденция к преобразованию системы образования как по содержанию, так и с точки зрения совершенствования процесса обучения, его методов, структурных форм, а также вспомогательных средств. Ускоряющееся развитие науки, обновление техники, появление новых технологий предъявляют все более строгие требования к системе образования, удовлетворить которым, используя только традиционные средства обучения, становится все труднее. Сегодня благодаря массовому появлению и быстрому совершенствованию персональных компьютеров появилась реальная возможность обучения с использованием средств вычислительной техники. Потребность в высокоэффективном машинном обучении очень велика и растет с каждым годом все быстрее.

Во многих промышленно развитых странах разработаны и осуществляются специальные программы и проекты по внедрению компьютеров в учебные заведения, включая школы. Умение обращаться с ЭВМ все в большей степени рассматривается как один из элементарных навыков, которым обязан владеть каждый человек.

Компьютеризация учебного процесса поставила перед преподавателями не только естественно-научных, но и гуманитарных дисциплин много новых задач и ранее неизвестных проблем.

1.1. Некоторые общие вопросы использования ЭВМ в образовании

Работа преподавателя, связанная с раскрытием сущности предмета, логической организацией учебного материала, подбором примеров и упражнений, коррекцией материала в зависимости от результатов усвоения, а также пробуждением устойчивого интереса к изучаемому предмету, носит явно выраженный творческий характер. Такую работу справедливо оценивают как педагогическое искусство. Сегодняшним компьютерам, даже самым мощным, подобная деятельность пока не под силу. Ученые связывают свои надежды с решением в недалеком бу-

дущем проблем искусственного интеллекта в ЭВМ пятого поколения.

Какие же функции можно поручить современному компьютеру, используя его для процесса обучения? Что умеет делать компьютер?

Возможность запоминать множество вещей является одним из самых замечательных свойств компьютера, в его памяти может храниться большое количество самой разнообразной информации, причем поиск ее осуществляется очень быстро.

Компьютеры работают с высокой скоростью, выполняя миллионы арифметических или логических операций в секунду. В отличие от человека они не испытывают утомления от долгой работы и не совершают ошибок.

Универсальность является фундаментальным свойством компьютеров. Они могут выполнять множество самых различных операций. Научно-технические расчеты, банковские операции, резервирование мест в гостиницах и на транспорте, начисление зарплаты, управление станками и роботами, обучение, тренажеры, игры — вот далеко не полный перечень работ, выполняемых ЭВМ.

Не менее важным свойством ЭВМ является определенность и однозначность их работы. Результаты вычислений на ЭВМ не зависят от температуры окружающей среды, давления, влажности, времени. Определенность работы ЭВМ является гарантией множества ее приложений.

Вместе с тем компьютеру присущи и недостатки, о которых никак нельзя забывать в любых сферах его использования, особенно в такой деликатной, как образование:

негибкость, очень бедная импровизация; ЭВМ фиксируют малейшие ошибки в программе и данных;

отсутствие воображения, способности к творчеству. Компьютеры «мыслят» прямолинейно, а не разносторонне;

машины полностью лишены эмоций, они не могут любить или чувствовать. Бессмысленно вымещать злость за свои ошибки в программе на клавишах компьютера;

ЭВМ не способны к принятию обобщающих решений, они не могут сами решить, что правильно, а что нет. Только человек способен назначать цели в сложных противоречивых ситуациях;

при неисправности аппаратуры ЭВМ возможны отказы и сбои. Плохая работа вычислительной машины мо-

жет быть вызвана низким качеством питающей сети и наводками различного характера.

Что нового несет внедрение ЭВМ в процессе обучения? Как повлияет компьютер на развитие учащегося, становление его интеллекта? Не изменят ли ЭВМ природу человеческого мышления? И если да, то в лучшую или в худшую сторону?

Поставленные вопросы еще ждут своего разрешения. Насколько сложна общая проблема, можно судить по такому частному вопросу.

Во многих случаях компьютер может сделать ненужным механическое заучивание, например, таблицы умножения. Какой смысл также учить правописанию, если компьютер с помощью словаря; заложенного в его память, сам может исправлять грамматические ошибки? Станет ли мозг, освобожденный от нудной умственной работы, искать более трудные задачи? Стимулирует ли ЭВМ работу головного мозга или, наоборот, выполняя часть его задач, дает ему возможность частично атрофироваться?

Последние 10—15 лет ведутся активные дискуссии: стоит ли зубрить? Некоторые преподаватели разрешают учащимся широко пользоваться на экзаменах литературой, считая даже вредным «забивать» голову всякими ненужными «подсобными» формулами и таблицами. Тренировка памяти утратила свою популярность, уступив место логическим построениям.

Однако, как показали исследования, нужна все-таки и так называемая элементарная зубрежка, заучивание наизусть больших литературных кусков и блоков, чтобы человек в любую минуту умел воспроизводить их, если потребуется, максимально используя свой внутренний аппарат считывания. Современная наука не рекомендует противопоставлять логику мышления тренировке памяти. Известный американский ученый Г. Унгар прямо связывает с состоянием памяти человека даже его долголетие: чем лучше, тренированнее память человека, говорит он, тем он дольше проживет. И порою не имеет значения, какое у человека здоровье. Дарвин, говорят, обладал блестящей памятью и в самом деле, несмотря на незавидное здоровье, жил долго. Известна также следующая рекомендация Л. Н. Толстого: «Никогда не справляйся в книге, ежели чего-нибудь забыл, а старайся сам припомнить».

Сказанное в некоторой мере — ответ на вопрос, целе-

сообразно ли во всех случаях перекладывать работу по запоминанию на компьютер.

ЭВМ с успехом имитируют многие процессы, протекающие в головном мозге человека: запоминание, сравнение, логический анализ, поиск информации. Однако разум машины лишь отражение интеллекта программиста. Инструментальную природу ЭВМ подчеркивает профессор Нью-Йоркского университета Б. Шнейдерман: «Я утверждаю, что с течением времени и даже по мере построения нами все более сложных вычислительных систем различие между творческими навыками человека и инструментальной природой ЭВМ становится все более очевидным. Мы увидим, какие задачи нужно переложить на машины. Мы еще яснее увидим, что ЭВМ — всего лишь инструменты, работающие под управлением человека, и что в них не больше интеллекта, чем в деревянном карандаше» [1].

Что нового несет внедрение ЭВМ в учебный процесс? И хотя опыт широкого использования их пока относительно небольшой, некоторые выводы можно сделать:

ЭВМ повышает активность работы учащегося, из пассивного усвоителя информации он превращается в ее добытчика;

индивидуальная работа с компьютером способствует развитию самостоятельности;

общение с ЭВМ приучает к точности, аккуратности, последовательности действий;

работа с ЭВМ способствует развитию чувства стратегии, каким путем лучше всего добиться результата, улучшает связность мышления, развивает способности к анализу и обобщению;

компьютер облегчает усвоение абстракций, позволяя многие из них представить конкретными.

Было бы несправедливым осветить только положительные стороны использования ЭВМ. Вызывает опасения бесконтрольное увлечение электронными играми, часть из которых несет в себе весьма сомнительную пользу. Неясно, с какого возраста стоит привлекать школьника к активной работе с компьютером. Педагоги справедливо отмечают, что слишком «сухой» и жесткий язык компьютера может отрицательно повлиять на эмоциональную и художественную сторону развития ребенка. Слепое, одностороннее увлечение компьютером в ущерб другим сторонам развития может нанести вред и в более зрелом возрасте. Все эти вопросы еще ждут

своего правильного разрешения. Не стоит забывать, что компьютер всего лишь инструмент, хотя и необычный. Превратить его в помощника, а не диктатора — достойная цель в любых его приложениях.

1.2. Общие требования к учебным ЭВМ

До появления микроЭВМ практическое применение вычислительных машин в сфере образования было ограничено и продвигалось медленно. Применялись в основном многотерминальные системы с одной центральной большой или миниЭВМ. Ресурсы процессора и внешних устройств использовались многими пользователями в режиме разделения времени. Эксплуатация подобных систем дорога́ и требует наличия высококвалифицированных кадров. Многотерминальным системам с одной центральной ЭВМ присущ также органический недостаток: при возникновении неисправности в ЭВМ полностью нарушается работа всего класса. По этим двум основным причинам системы разделения времени в школах практически не применяются.

Появление микроЭВМ позволило обеспечить монопольную работу пользователя: он имеет легкий, быстрый, непосредственный доступ к машине тогда, когда ему нужно. Это крайне необходимо для эффективного овладения навыками работы с машиной и создает более благоприятные условия для менее способных пользователей. Применение микроЭВМ дает немедленную обратную связь в процессе диалога человека с машиной, придает исследовательский характер процессу обучения на основе метода проб и ошибок.

Реализация аппаратных средств учебной микроЭВМ возможна двумя путями: посредством доработки и адаптации к учебной среде серийно выпускаемых микроЭВМ либо посредством разработки и проектирования новых моделей учебных микроЭВМ. Учитывая необходимость быстрого решения поставленных задач, на первом этапе было решено использовать серийно выпускаемые образцы профессиональных микроЭВМ.

В качестве аппаратной основы учебной микроЭВМ был принят диалоговый вычислительный комплекс ДВК-1М. Подробно микроЭВМ этого типа описаны во второй книге настоящей серии [3]. Основой организации ДВК является стандартная шина типа Q-bus. Центральная часть комплекса — одноплатная микроЭВМ «Элек-

троника МС 1201.01» — выполнена на базе однокристального 16-разрядного микропроцессора К1801ВМ1.

МикроЭВМ полностью отвечает техническим требованиям, предъявляемым к учебным микроЭВМ, и включает в себя: ОЗУ 56 К байт (ОЗУ пользователя); контактное устройство для БИС ПЗУ 8 К байт; параллельный байтовый интерфейс; последовательный интерфейс; интерфейс НГМД.

МикроЭВМ программно совместима с рядом отечественных и зарубежных мини- и микроЭВМ, имеющих систему команд СМ ЭВМ и «Электроника-60», что дает возможность использования их богатого программного обеспечения. Программная совместимость обеспечивает также преемственность в обучении основам вычислительной техники и программированию: придя из учебных заведений (школ, ПТУ, техникумов, вузов) в учреждения и на предприятия, молодые люди будут использовать уже знакомую технику. Все это делает перспективным применение ДВК в сфере образования.

В состав ДВК входят также одноцветный символьный дисплейный монитор и стандартная клавиатура типа клавиатуры пишущей машинки. Дисплей обеспечивает ввод с клавиатуры и отображение на экране 24×80 символов — прописных и строчных букв русского и латинского алфавитов, арабских цифр, специальных и служебных символов. Связь дисплея и микроЭВМ в ДВК осуществляется по последовательным каналам по двум линиям типа «20 мА токовая петля» со скоростью 9600 бод. Применение в микроЭВМ клавиатуры пишущей машинки со стандартной схемой расположения и набором символьных клавиш оправдано. Не сосредоточивая внимания на эргономических вопросах конструирования клавиатуры, в подтверждение правильности сказанного отметим, что многие зарубежные фирмы включают в состав вновь разработанных учебных микроЭВМ стандартную клавиатуру, близкую к профессиональной.

Использование в учебной микроЭВМ специального дисплейного монитора, а не бытового телевизионного приемника, несмотря на более высокую стоимость, обеспечивает гораздо лучшее и приемлемое для обучения качество изображения. Этот фактор имеет определяющее значение. Применение бытовых телевизоров, особенно цветных, ограничивается тем, что они не отвечают санитарно-медицинским требованиям и не обеспечивают качества изображения, удобного для зрительного вос-

приятия. Нельзя не принять во внимание тот факт, что учащийся располагается очень близко от экрана, как правило, расстояние не превышает 30—40 см. Некоторые медики-исследователи утверждают, что низкочастотное излучение, испускаемое электронно-лучевой трубкой дисплея, может вызвать стрессовое состояние, катаракты и другие болезни. Имеется некоторый положительный опыт в решении данного вопроса [4]. Фирма «Sentinal» выпускает защитный экран, пропускающий более 50% видимого света, но почти полностью задерживающий ультрафиолетовое излучение и значительно ослабляющий электромагнитное. Кроме того, сообщается, что защитный экран уменьшает блики, поглощая 46% падающего света. Сетка экрана изготовлена из тонкой проволоки для нитей накала, покрытой проводящим слоем и заключенной между двумя слоями акрила. Сетка закреплена в рамке из алюминиевого сплава, прикрепляется к экрану дисплея и задерживает более 98% электромагнитного излучения. Цена защитной сетки от 200 до 280 долл. Вопрос о влиянии излучения на человека, работающего за экраном дисплея, пока носит полемический характер.

Существенное значение для адаптации ДВК к учебной среде имеет наличие на плате микроЭВМ контактного устройства для БИС ПЗУ емкостью 8 К байт. В условиях, когда применение магнитных дисковых запоминающих устройств ограничено из-за их высокой стоимости и недостаточной надежности, этот факт имеет принципиальное значение, так как дает возможность разместить в ПЗУ интерпретатор диалогового языка программирования с некоторыми функциями операционной системы. Размещение в ПЗУ имеет решающее значение, так как после включения питания ЭВМ сразу готова к работе, без необходимости загрузки системы в оперативную память с внешних носителей. При этом также существенно возрастает надежность функционирования системы.

Наличие байтового и последовательного интерфейсов позволяет при необходимости подключать стандартные внешние устройства или включать ДВК в локальную сеть. ДВК-1М обладает умеренной стоимостью, прост в использовании, не требует специального технического обслуживания, имеет небольшие габаритные размеры и потребляемую мощность, возможность массового производства.

Таким образом, базовый комплект типа ДВК удовлетворяет требованиям, предъявляемым к учебным ЭВМ, и может служить основой для создания программных и аппаратных средств кабинетов вычислительной техники учебных заведений.

1.3. Организация простой локальной сети для учебного класса

Важное значение имеют организация работ в кабинете вычислительной техники, взаимодействие преподавателя с учащимися. Эти вопросы тесно связаны с выбором внешних запоминающих устройств и организацией локальной сети учебных микроЭВМ. Использование недорогих накопителей на кассетной магнитной ленте ограничено их малой скоростью работы, последовательным режимом доступа и невысокой надежностью. В качестве устройства внешней памяти предпочтительнее использовать НГМД. Но их высокая стоимость, потребность в «деликатном» обращении, недостаточная надежность делают невозможным комплектацию этим устройством каждого рабочего места. Аудиторная организация обучения позволяет создать простую локальную сеть учебных микроЭВМ, связанных с одной главной машиной — машиной преподавателя, единственной, имеющей полный комплект периферийного оборудования (НГМД, печатающее устройство, графопостроитель). Такая учебная локальная сеть предполагает, что ресурсы внешних устройств главной ЭВМ будут делиться между периферийными микроЭВМ. Локальная сеть позволяет исключить необходимость пользования накопителями на магнитных дисках и дискетами на каждом рабочем месте, дает возможность осуществлять оперативную связь между ЭВМ, облегчает труд преподавателя, уменьшает затраты на одного учащегося до приемлемого уровня. Локальная сеть микроЭВМ с общими внешними устройствами обладает всеми преимуществами традиционной системы разделения времени при гораздо меньшей стоимости и распределенной обработке данных.

В мощных информационно-вычислительных сетях функции передачи информации чрезвычайно сложны и иерархически делятся на уровни. Эталонная модель архитектуры взаимодействия открытых систем МОС (Международной организации по стандартизации) определяет семиуровневую иерархию связи процессов, находящихся

в различных системах [5]. Подключаемая к такой сети ЭВМ должна поддерживать все семь уровней связных протоколов, что требует в ней наличия собственных связных аппаратных и программных средств — различных адаптеров и интерфейсных схем, сетевых устройств преобразования информации, ПЗУ для хранения программ передачи данных и т. п.

Насколько не проста эта задача, можно судить по реализации кольцевой сети с маркерным доступом TRN [6]. Это результат напряженной трехлетней работы двух известных фирм: «TI» и «IBM». Связь между периферийными ЭВМ осуществляется с помощью витой пары проводов со скоростью 4 Мбит/с. Адаптер кольцевой сети персонального компьютера представляет собой печатную плату, на которой установлены три сверхбольшие интегральные микросхемы (СБИС), выполненные по n-канальной технологии, и две микросхемы средней степени интеграции с биполярными диодами Шотки: 100-контактный системный интерфейс TMS 38030; 48-контактный связной процессор TMS 38010; 48-контактный протокольный процессор TMS 39020; 22-контактный приемопередатчик кольцевой сети TMS 38051; 20-контактный контроллер TMS 38052. Кроме того, поставляется пассивный концентратор на восемь мест. Общая стоимость адаптера и концентратора составляет 1355 долл.

Для учебных приложений подобные затраты пока неприемлемы и могут быть сокращены за счет ограничения набора предоставляемых сетью возможностей, соответствующего упрощения аппаратуры и программного обеспечения сетевых протоколов.

В простой учебной сети, расположенной в пределах одной классной аудитории, необходимо прежде всего обеспечить взаимодействие главной ЭВМ (ЭВМ преподавателя) с периферийными ЭВМ (ЭВМ студента, учащегося) с целью надежной передачи обучающих программ, использования внешних устройств главной машины для хранения и ввода—вывода информации, предоставления оперативной связи с обучаемыми. Существуют различные схемы соединения микроЭВМ в локальные сети: звездообразное соединение в одной точке, кольцевое соединение в виде замкнутой окружности, подключение микроЭВМ к одному физическому каналу (шина, моноканал) и др. [5].

Наиболее простой для реализации как аппаратных, так и программных средств локальной сети является

звездообразная схема соединения с одним центральным узлом, в которой каждая периферийная ЭВМ соединяется с главной с помощью собственного канала передачи информации. Включение в сеть либо отключение периферийной ЭВМ от сети не приводит к изменению характера работы остальной части сети. В отличие от децентрализованного управления, оно не требует в каждой ЭВМ иметь полный набор программных средств для координации выполнения сетевых операций. Главная ЭВМ сети выполняет функции управления аппаратными ресурсами, файлами данных, обеспечивает защиту данных. Для учебных приложений такая структура наиболее приемлема как с точки зрения затрат на ее разработку и внедрение, так и необходимости централизации управления при аудиторной организации обученя.

К аппаратуре передачи данных и физической среде учебной сети не предъявляется жестких требований. Главные из них следующие:

надежность и достаточная скорость передачи данных, обеспечивающие работу процессов программного уровня;

низкая стоимость, простота эксплуатации и монтирования.

С учетом изложенных соображений был разработан вариант учебной локальной сети на базе микроЭВМ типа ДВК, приведенный на рис. 1.1 [7]. В качестве главной ЭВМ использован диалоговый вычислительный комплекс ДВК-2М, в состав которого входят: НГМД «Электроника 6022» с дискетами диаметром 133 мм; печатающее устройство типа УВВПЧ 30.004. Двенадцать периферийных ЭВМ ДВК-1М связаны линиями последова-

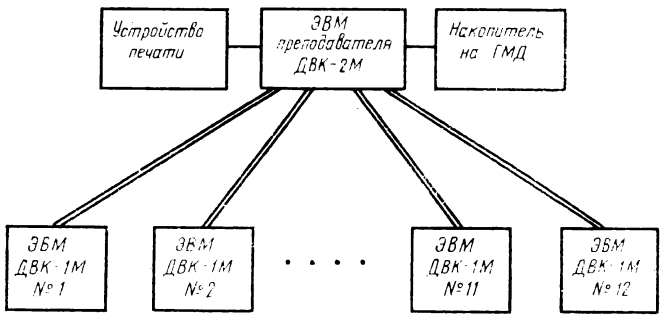


Рис. 1.1. Схема учебного класса на основе ДВК

тельной передачи данных с главной ЭВМ. Такая форма организации микроЭВМ позволяет сосредоточить управление обменом в сети в главной ЭВМ, которая содержит аппаратуру приема — передачи информации и программные средства управления файлами, работающие в среде стандартной однопользовательской операционной системы ОС ДВК (RT11).

Согласно эталонной модели семиуровневой организации вычислительных процессов, происходящих в различных ЭВМ, связываемых сетью, в рассматриваемой учебной локальной сети можно определить следующие уровни;

физический, поддерживаемый аппаратурой приема— передачи данных и физическими каналами связи;

канальный, выполняющий функцию установления, поддержания и разъединения физических соединений, реализуемый электронным коммутатором;

сетевой, транспортный, а также сеансовый и представительный уровни отсутствуют, так как нет необходимости в маршрутизации и селекции информации (в связи с тем что используется микроЭВМ одного типа, не требуется процедура представления и преобразования информации);

прикладной, реализуемый программно и осуществляющий управление обменом информацией, контроль ошибок передачи, введение общей файловой системы, распределение ресурсов главной ЭВМ, управление внешними устройствами, взаимодействие главной ЭВМ с сетевыми.

Рассмотрим вопросы, связанные с передачей данных по последовательным каналам. Последовательная передача данных означает, что информация от передатчика к приемнику передается по одной линии. В линиях последовательной передачи данных для кодирования двоичных битов 1 и 0 служат соответственно сигналы маркера и пробела. В учебной локальной сети физическая связь автономных ЭВМ с главной машиной осуществляется непосредственно (без использования модемов для преобразования уровней напряжения в колебания соответствующих частот и обратно) по паре проводов типа «токовая петля». Значение тока 20 мА представляет сигнал маркера, а 0 — сигнал пробела. Чаще используется асинхронная последовательная передача данных. Это означает, что у передатчика и приемника нет общего генератора синхроимпульсов для посылки вместе с данными. Для

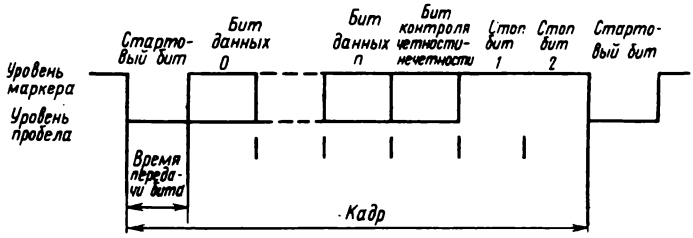


Рис. 1.2. Стандартный формат асинхронной последовательности данных

распознавания момента начала и конца передачи используется стандартный формат асинхронной последовательной передачи данных [8], приведенный на рис. 1.2. Передатчик начинает передачу посредством генерирования стартового бита (уровень сигнала пробела), за которым следуют биты символа, начиная с младшего бита и дополнительный бит контроля по четности или нечетности. Конец передачи символа кодируется с помощью двух стоп-битов, поддерживающих уровень сигнала маркера обычно в течение двойного времени передачи бита. Приемник в такой же последовательности принимает эту цепочку битов, распознавая начало, выделяя сам символ и определяя конец передачи символа.

В настоящее время интерфейсные схемы для последовательной передачи данных изготовляют в виде БИС. В рассматриваемой локальной сети используется серийно выпускаемый промышленностью асинхронный приемопередатчик К1801ВП1-065. Микросхема в общем случае является однокристальным контроллером внешних устройств, работающих на линии связи с последовательной передачей информации в дуплексном режиме. Микросхема предназначена для преобразования параллельной информации в последовательную и наоборот. На ее основе промышленностью выпускаются платы контроллеров телеграфного канала КТлК-6 и КТлК-4 (см. вторую книгу «Персональные ЭВМ» настоящей серии).

Для понимания принципа организации локальной сети рассмотрим фрагмент принципиальной электрической схемы, изображенной на рис. 1.3. Физическая связь микросхемы К1801ВП1-065 с внешними устройствами осуществляется с помощью узла оптоэлектронной развязки: оптронных приемников D1 и D2 на основе микросхемы

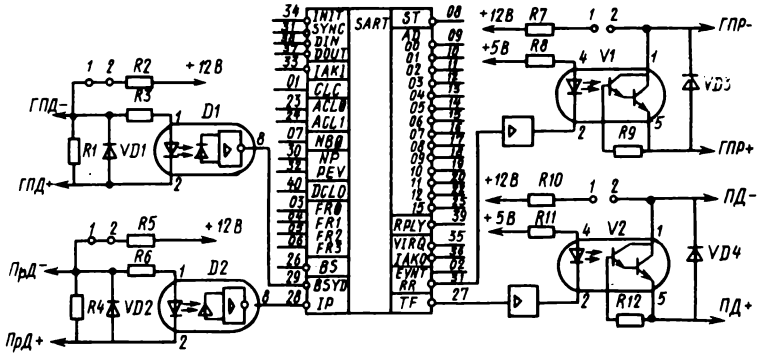


Рис. 1.3. Фрагмент принципиальной электрической схемы асинхронного последовательного канала

К293ЛП1 и оптронных передатчиков V1 и V2 на основе микросхемы АОТ110А. ГПД и ГПР — соответственно сигналы готовности передатчика и приемника, ПрД и ПД — соответственно принимаемые и передаваемые данные. Связь осуществляется по линиям канала «20 мА токовая петля», выполненным в виде свитых пар проводов. Маркеру соответствует ток 0—3 мА, пробелу — ток 15—25 мА.

Входные данные, сформированные в виде последовательности битов, называемых кадром, поступают на вход микросхемы IP. Последовательный код внутри микросхемы преобразуется в параллельный и записывается в виде байта в буферный регистр приемника. Далее этот байт может быть считан в ЭВМ, при этом буферный регистр приемника освобождается и на выходе микросхемы RR появляется сигнал, свидетельствующий о том, что можно принимать следующий кадр. Так осуществляется прием информации из линии.

Для передачи информации по линии ЭВМ записывает байт в буферный регистр передатчика микросхемы. Внутри микросхемы байт преобразуется в последовательность битов (кадр), которая подается на выход TF микросхемы. Передача данных в линию возможна только в том случае, если на входе микросхемы BSID присутствует разрешающий сигнал.

Если на другом конце линии разместить аналогичный приемопередатчик, то связь между ними организуется простым соединением сигналов: выводы RR и TF передатчиков через опторазвязки соединяются соответственно

с выводами BSID и IP приемников. На этом принципе и организована взаимозамкнутая асинхронная связь между главной и периферийными ЭВМ. Данные циркулируют между выводами TF и IP микросхем, а квитирующие (разрешающие) сигналы — между выводами RR и BSID этих же микросхем.

Скорость обмена данными может выбираться путем подачи определенных логических уровней напряжений на входы микросхемы FR0—FR3 с помощью движковых переключателей. При тактовой частоте 4608 кГц микросхема асинхронного приемопередатчика обеспечивает скорости обмена по последовательному каналу от 50 до 57 600 бод. На практике максимальная скорость передачи не превышает 19200 бод из-за недостаточного быстродействия оптопередатчика А0Т110. При использовании более скоростной микросхемы, например А0Т128, надежный обмен информацией осуществляется во всем указанном диапазоне.

Формат посылки 7 или 8 информационных битов задается сигналом NB0. Формирование и контроль бита четности или нечетности (бит паритета) или работу без него определяют сигналы NP, PEV. Выбор адресов внутренних регистров и векторов прерываний осуществляется сигналом ACL0 и ACL1.

Для организации учебной локальной сети в ЭВМ преподавателя ДВК-2М вставляются две платы контрол-

Таблица 1.1

Номер канала	Адрес вектора прерывания канала ввода	Адрес регистра состояний ввода	Адрес регистра данных ввода	Адрес регистра состояний вывода	Адрес регистра данных вывода
1	140	176500	176502	176504	176506
2	150	176510	176512	176514	176516
3	160	176520	176522	176524	176526
4	170	176530	176532	176534	176536
5	300	176540	176542	176544	176546
6	310	176550	176552	176554	176556
7	320	176560	176562	176564	176566
8	330	176570	176572	176574	176576
9	340	176600	176602	177604	176606
10	350	176610	176612	176614	176616
11	360	176620	176622	176624	176626
12	370	176630	176632	176634	176636

лера телеграфных каналов КТлК-6. Каждая плата содержит шесть идентичных каналов, реализованных на микросхемах К1801ВП1-065 с соответствующими оптопарами и движковыми переключателями для ручной установки скорости обмена, режима работы с битом паритета, адресов внутренних регистров и векторов прерываний для каждого канала.

В табл. 1.1 приведены сведения о соответствии номеров последовательных каналов сети с адресами регистров и векторов для двух плат КТлК-6.

В принятом варианте организации локальной сети аппарат прерываний для плат КТлК-6 не используется, поэтому установка векторов прерываний с помощью движковых переключателей не обязательна.

Каждая периферийная ЭВМ ДВК-1М оснащается последовательным каналом передачи информации, выполненным в виде стандартной полуплаты СА (сетевой адаптер). Сетевой адаптер разработан в Московском институте электронной техники с расчетом его использования для построения двух вариантов локальных сетей: кольцевой и звездообразной [2]. Он содержит два независимых канала, один из которых является стандартным устройством последовательного обмена, работающим на линию «20 мА токовая петля» (канал ИРПС). Именно этот канал используется в описываемой учебной локальной сети.

Схема соединения одного канала платы КТлК-6 и канала сетевого адаптера ИРПС показана на рис. 1.4.

Как видно из рисунка, связь с каждой периферийной ЭВМ осуществляется с помощью четырех пар витых проводов. Верхняя и нижняя половины схемы полностью идентичны по аппаратуре, различие заключается лишь в направлении передачи. Токовые петли запитываются со стороны приемников.

В канале ИРПС сетевого адаптера также используется асинхронный приемопередатчик К1801ВП1-065. С помощью переключателей, имеющихся на плате сетевого адаптера, устанавливаются адрес канала ИРПС на общей шине 176560_8 и вектор прерывания 360_8.

При установке платы сетевого адаптера в гнездо общей шины ДВК-1М необходимо проследить за тем, чтобы сигнал предоставления прерывания К ППР1 Н поступал на соответствующий вход сетевого адаптера. Напомним, что сигнал предоставления прерывания вырабатывается процессором и последовательно проходит через контрол-

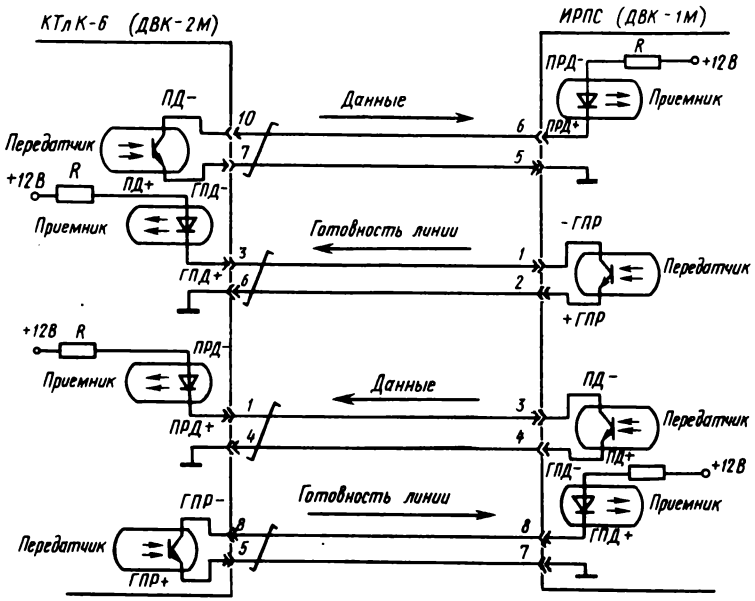


Рис. 1.4. Схема соединения последовательных каналов микроЭВМ ДВК-2М и ДВК-1М

леры внешних устройств, подключенных к общей шине. Если конкретное устройство не работает по прерыванию, то этот сигнал им просто транслируется дальше на следующее устройство. Физическое отсутствие платы контроллера некоторого внешнего устройства разрывает цепь прохождения сигнала предоставления прерывания. Такая ситуация может оказаться одной из причин неработоспособности канала передачи информации.

Для обеспечения функционирования локальной сети вместе с аппаратными средствами (платы КТлК-6 в главной ЭВМ, связующие кабели и сетевые адаптеры в периферийных ЭВМ) используются и соответствующие программные средства. В главной ЭВМ сетевые программы записаны в виде файлов на магнитной дискетте, а в периферийных машинах они являются частью интерпретирующей системы, «зашитой» в БИС ПЗУ К1801РЕ2-093. Указанная микросхема вставляется в специальную колодку (ПЗУ пользователя), расположенную на плате микроЭВМ периферийной машины.

Все функции управления обменом информации со-

средоточены в главной ЭВМ. Преподаватель, на рабочем месте которого установлена ЭВМ ДВК-2М, может выполнить следующие операции:

параллельную загрузку выбранной обучающей программы во все периферийные ЭВМ или в любые по выбору (операция загрузки наибольшей по объему обучающей программы занимает около 2 мин);

последовательную загрузку разных обучающих программ в разные периферийные машины;

подключение к любой периферийной ЭВМ и наблюдение за действиями учащегося (вся информация, которую обучаемый вводит с клавиатуры либо выводит на свой экран, отображается на экране преподавателя);

просмотр в выбранной периферийной ЭВМ программы, находящейся в ее оперативной памяти, внесение необходимых исправлений, запуск программы на счет и проверка результата ее работы.

Применение описанного принципа организации учебной локальной сети обеспечивает простую физическую среду для надежного функционирования процессов программного уровня, удовлетворяет основным требованиям организации аудиторной работы, вытекает из технических ограничений и ориентировано на использование имеющегося серийного оборудования с минимальной доработкой.

1.4. Графические средства вывода информации в ДВК

МикроЭВМ ДВК-1 и ДВК-2 способны выводить на экран дисплея только символьную информацию (буквы, цифры, специальные знаки). Вряд ли могут возникнуть сомнения по поводу необходимости графики в учебном компьютере. Построение и исследование математических функций, геометрических фигур, возможность воспроизведения рисунков и чертежей, показ многих процессов и явлений в их «живом» виде — вот далеко не полный перечень того, что может делать ЭВМ, оснащенная графикой. Возможность вывода графической информации в ДВК появилась с разработкой платы контроллера графического вывода группой выпускников Московского института электронной техники [9].

Контроллер графического дисплея (КГД) выполнен в виде полуплаты конструктива микроЭВМ «Электроника-60» и вставляется в свободное гнездо магистрали Q-bus. Кроме сигналов общей магистрали на плату

КГД подаются сигналы кадровой и строчной развертки, а также видеосигнал с дисплея 15ИЭ-00-013. КГД имеет собственную оперативную память объемом 16 К байт. Логической единице в памяти контроллера соответствует светящаяся точка на экране монитора. Размер графического поля — 286×400 точек (соответственно по вертикали и горизонтали).

Управляется КГД с помощью трех регистров, доступных по чтению и записи с общей магистрали: регистра управления (176640); регистра данных (176642); регистра адреса (176644).

Регистр управления служит для задания режимов работы (рис. 1.5). Пятнадцатый бит разрешает выдачу графической информации, когда находится в состоянии «1»; четырнадцатый бит в состоянии «1» запрещает выдачу символьной информации.

При начальном включении КГД все биты регистра управления сбрасываются в «0». Таким образом, исходным режимом работы монитора является символьный режим.

В регистре адреса (рис. 1.6) задействованы биты с нулевого по тринадцатый включительно, два старших бита не используются.

В регистре данных (рис. 1.7) используется только младший байт. Обмен данными с микроЭВМ производится побайтно. За начало отсчета принимается верх-

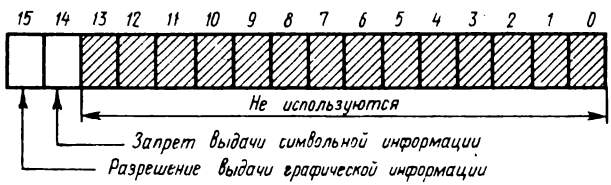


Рис. 1.5. Формат регистра управления контроллера графического дисплея

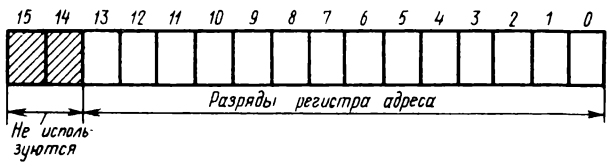


Рис. 1.6. Формат регистра адреса контроллера графического дисплея

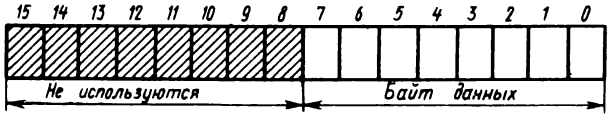


Рис. 1.7. Формат регистра данных контроллера графического дисплея

няя левая точка экрана. Крайней левой точке элемента изображения, состоящего из восьми горизонтальных точек, соответствует нулевой разряд байта графической информации регистра данных. Более подробная техническая информация содержится во второй книге настоящей серии [3].

Рассмотрим функциональные характеристики графического вывода информации, необходимые для учебного компьютера.

Предоставляемые непосредственно аппаратурой средства графического вывода очень ограничены. Они позволяют только высветить или погасить точку в определенном месте экрана, задавая адрес байта и место бита в байте данных. Для удобства пользования необходимо пакет графических функций, входящий в системное обеспечение учебного компьютера и служащий посредником между человеком и аппаратурой. Эффективность работы с графикой во многом зависит от качества написанных системных программ.

Немаловажным является вопрос выбора на экране начальной точки отсчета. Наиболее естественное ее положение — нижний левый угол экрана (рис. 1.8). При этом поле 286×400 точек легко ассоциируется с привычной декартовой системой координат. К сожалению, некоторые персональные компьютеры в качестве начала отсчета имеют точку в верхнем левом углу экрана.

Первой необходимой операцией является начальная установка экрана. Она может быть реализована функцией одного аргумента, принимающего три возможных

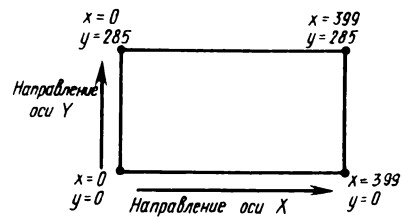


Рис. 1.8. Система координат графического поля дисплея

значения. Например, при нулевом значении аргумента память экрана полностью обнуляется, т. е. поле заполняется темными точками (предыдущая картинка стирается); при единичном значении все поле заполняется светлыми точками (светлый фон); при значении аргумента «—1» картинка инвертируется: светлые точки становятся темными, и наоборот.

Полезной является операция установки режима вывода точек. Это функция с одним аргументом, похожая на предыдущую. При значении аргумента «0» выводятся темные точки (здесь желателен светлый фон), при значении аргумента «1» — светлые точки (здесь желателен темный фон), при значении «—1» устанавливается инверсный режим выводимых точек по отношению к фону.

Операция вывода точки в определенное место экрана реализуется с помощью функции двух аргументов (координаты по горизонтальной и вертикальной осям). Пожалуй, это самая универсальная функция. Используя вычислительные возможности компьютера для расчета координат очередной выводимой точки, можно нарисовать практически любую графическую картинку. В некоторых учебных микроЭВМ графические возможности реализуются с помощью только этой одной функции.

Однако, учитывая тот факт, что интерпретирующие системы работают довольно медленно, в пакет графических функций обычно включают набор программ, называемых примитивами, которые позволяют рисовать некоторые простые фигуры путем разового обращения к ним. Например, для вывода отрезка задаются координаты начальной и конечной точек; окружности — радиус и координаты центра; прямоугольника — координаты одной из вершин, ширина и длина. Набор хорошо продуманных примитивов существенно облегчает работу с графикой.

В некоторых случаях удобно задавать координаты в виде приращений (относительные координаты). Например, при рисовании отрезка его начало задается в абсолютных координатах, а конечная точка — в относительных.

Важным является понятие графического курсора. Фактически это координата текущей точки, значение которой определяется содержимым регистра адреса КГД. В момент включения ЭВМ регистр адреса КГД обнуляется и поэтому графический курсор устанавливается в точку начала отсчета (координата ее — 0,0). Если на эк-

ран дисплея выводится отрезок, то графический курсор остается в его конечной точке. Это обстоятельство позволяет при выводе следующего отрезка задавать абсолютные координаты либо приращения только конечной точке, т. е. координаты начальной точки в этом случае заданы по умолчанию. Выигрыш здесь заключается в более короткой записи функции вывода отрезка и в некотором увеличении скорости выполнения функции.

С этими вопросами прямо связаны еще две необходимые функции: функция установки графического курсора в абсолютных координатах и аналогичная функция установки курсора в относительных координатах. Нужно заметить, что в реализации для ДВК графический курсор в отличие от символьного не подсвечивается (невидим). Это создает определенные неудобства при построении (программировании) графических изображений.

О конкретной реализации графического пакета программ в интерпретаторе языка ФОКАЛ см. гл. 2.

1.5. Обоснование выбора языка программирования для начального обучения

Распространение вычислительных машин, появление персональных ЭВМ, создание систем массового информационного обслуживания, приближение вычислительной техники к миллионам пользователей поставили перед специалистами и разработчиками программного обеспечения задачу создания эффективных программных интерфейсов «человек — ЭВМ». Важными становятся обеспечение удобства пользования и ориентация на естественные требования пользователя. При создании таких программных компонентов, как языки программирования, языки управления операционными системами, средства обращения к базам данных, средства автоматизированного обучения, различные программы редактирования текстов и обработки слов, особое внимание уделяется использованию ЭВМ неопытными пользователями. Еще в большей мере следует учитывать человеческий фактор при разработке средств, ориентированных на учебный процесс и обучение программированию.

Следует различать начальное обучение программированию и изучение программирования как некоторой систематической дисциплины, основанной на нескольких фундаментальных понятиях, призванных обеспечить язы-

ковую поддержку современной технологии программирования. Второй подход к программированию часто называют структурным программированием или программированием методом последовательных уточнений (stepwise-refinement). К языкам структурного программирования относятся АЛГОЛ, ПАСКАЛЬ, АДА, СИ. Многие концепции этих языков отражают последние достижения в методологии программирования. Но начинать с них изучение программирования крайне затруднительно по следующим двум основным причинам:

1. Программирование некоторой задачи приходится начинать с описания типов всех переменных, которые будут использоваться в ней, а также всех процедур. Начинающие испытывают затруднения в понимании того факта, что описания процедур сами не вызывают действий, а лишь описывают, **как** они должны выполняться. Процедуры начинают работать лишь при обращении к ним путем их вызова. Сложными являются также понятия замещения формальных параметров фактическими, способы передачи параметров (вызовы по значению, имени, ссылке), области действия глобальных и локальных меток и переменных.

2. Для понимания такого языка, как, например, ПАСКАЛЬ, требуется определенный и достаточно высокий уровень образования, чтобы успешно усвоить основные семантические понятия.

Выбор языка для начального обучения целесообразно производить как из соображений естественности и простоты его освоения, так и с учетом возможности в дальнейшем перехода к изучению более мощных языков программирования. Поэтому он должен быть стандартным и представлять достаточно полно основные конструкции языков программирования. С точки зрения вычислительной машины язык должен быть легко реализуем, чтобы избежать чрезмерных аппаратных и временных затрат.

Не вызывает сомнений, что язык для начинающих должен быть интерпретирующего, а не транслирующего типа. При тех скоростях трансляции, которые доступны для учебных микроЭВМ, отладка программы с большим числом ошибок становится утомительной процедурой, сводящейся к долгому ожиданию результатов трансляций, и способна отбить охоту заниматься программированием.

Полезность интерпретатора заключается в том, что он фактически совмещает в себе одновременно функции

редактора текста, транслятора и загрузчика. Программы, написанные на языке интерпретирующего типа, выполняются непосредственно строка за строкой, поэтому их легко отлаживать и вносить в них изменения.

Возможность исполнения операторов языка непосредственно с пульта ЭВМ является очень удобным средством для понимания смысла операторов.

К языкам, удовлетворяющим перечисленным требованиям, в первую очередь относятся диалоговые языки программирования БЕЙСИК и ФОКАЛ [13]. Несмотря на большую распространенность языка БЕЙСИК, для реализации в ДВК был выбран язык ФОКАЛ, являющийся стандартным языком программирования для ряда мини- и микроЭВМ. Обоснуем причины такого выбора.

Языки БЕЙСИК и ФОКАЛ обладают примерно равноценными средствами управления программой. Различия касаются условного оператора IF: в языке БЕЙСИК используется логический, а в языке ФОКАЛ — арифметический оператор IF. Оператор FOR языка БЕЙСИК требует наличия оператора конца тела цикла NEXT. В языке БЕЙСИК допускается вложенное обращение подпрограмм с помощью оператора GOSUB. Оператор DO языка ФОКАЛ также допускает вложенное обращение, но, кроме этого, он позволяет использовать и рекурсивное обращение, т. е. обращение подпрограммы самой к себе. Язык БЕЙСИК обладает некоторой избыточностью операторов управления программой, в частности функции операторов RUN и GOTO, STOP и END в языке ФОКАЛ реализуется соответственно операторами GOTO и QUIT. В отличие от БЕЙСИКА в ФОКАЛЕ на номер строки можно ссылаться с помощью переменной (GOTO M).

Вычислительные средства в языке БЕЙСИК представлены оператором присваивания LET, которому в языке ФОКАЛ соответствует оператор SET. Кроме того, ФОКАЛ содержит оператор XECUTE, выполняющий требуемые вычислительные действия без сохранения результата. Это необходимо для выполнения функций взаимодействия с внешними устройствами, ввода — вывода символьной информации, генерации случайных чисел, воздействия на некоторые данные и т. п.

Сравнение средств ввода — вывода показывает, что язык ФОКАЛ обладает более гибкими и развитыми средствами работы с внешними устройствами. Это достигается использованием оператора OPERATE, который

дает возможность пользователю назначать в каждый момент времени одно устройство ввода и одно устройство вывода, после чего все операторы ввода — вывода данных и программ языка ФОКАЛ будут работать с назначенными устройствами до следующего переназначения. В языке БЕЙСИК для ввода либо вывода программ на различные внешние устройства используются различные операторы.

В языках БЕЙСИК и ФОКАЛ имеются средства организации одномерных и двумерных массивов. В ФОКАЛЕ они образуются динамически и не требуют описания, на количество элементов массива не накладывается ограничений. Язык БЕЙСИК позволяет работать с неописанными явно одномерными массивами, содержащими не более 10 элементов, а с двумерными — не более 100 (10×10). Для организации массивов с большим количеством элементов их необходимо описать с помощью оператора DIM. При этом максимально допустимое количество элементов описанного явно массива также ограничено: для одномерного составляет 255, а для двумерного — 255×255 элементов.

Одним из преимуществ ФОКАЛа перед БЕЙСИКом является возможность использования сокращенной мнемоники операторов (до одной первой буквы). Благодаря этому программы становятся более компактными, увеличивается скорость их ввода с клавиатуры, до некоторой степени увеличивается и скорость выполнения (интерпретация программ).

Отличительной особенностью библиотеки стандартных функций языка ФОКАЛ является наличие функций управления общей магистралью, ввода — вывода символьной информации, считывания показаний системного таймера. Однако следует заметить, что более развитые версии БЕЙСИКа обеспечивают обработку строковых данных, работу с магнитными дисковыми и кассетными накопителями, позволяют строить графические изображения, управлять цветом и звуком. Все эти возможности достигаются за счет значительного увеличения объема интерпретатора. Развитые версии БЕЙСИКа требуют объемов памяти 20 К байт и более [10].

Таким образом, языки БЕЙСИК и ФОКАЛ обладают примерно одинаковыми вычислительными возможностями, хотя ФОКАЛ предоставляет больше средств для отладки и написания компактных программ и работы с периферийным оборудованием [11]. Но, говоря, о со-

поставляемых версиях языков, стоит подчеркнуть их принципиальное различие: интерпретатор ФОКАЛа в реализации для микроЭВМ типа ДВК примерно в два раза компактнее и в 2,3 раза быстрее аналогичного интерпретатора БЕЙСИКА для той же микроЭВМ.

По-видимому, этот факт можно объяснить тем, что язык ФОКАЛ разрабатывался специально для использования на ЭВМ серии PDP и в большой степени учитывает особенности архитектуры этих популярных машин. Стремление максимально удовлетворить требованиям учебной среды при ограниченном объеме ПЗУ микроЭВМ ДВК-1М (8 К байт) и послужило решающим доводом в пользу выбора более компактного языка.

Естественно, ориентация на менее распространенный язык ФОКАЛ, а не БЕЙСИК может вызвать возражения. Однако в оправдание такого выбора можно привести следующие доводы: язык ФОКАЛ близок к языку БЕЙСИК, легок для самообучения, позволяет писать компактные программы, обеспечивает простой доступ к необходимым вычислительным ресурсам и в составе минимальных технических средств решает проблему обучения начальным основам программирования.

В настоящее время все большее число специалистов склоняются к мнению, что не существует универсального «хорошего» языка. Этим объясняется частичный неуспех языка ПЛ/1: он оказался громоздким и трудным для изучения. Каждый язык наиболее удобен в своей области приложений: БЕЙСИК, ФОКАЛ, ЛОГО — для начинающих, ФОРТРАН — для проведения научно-технических расчетов, КОБОЛ — в области экономики, ПАСКАЛЬ — для обучения принципам структурного программирования, АДА — для проектирования больших и сверхбольших надежных программ, ЛИСП — для работ в области искусственного интеллекта и т. п.

Почему полезно изучать несколько языков программирования? В работе [12] приводятся следующие причины: улучшается понимание конкретного языка; расширяется активный запас полезных конструкций; появляется возможность более обоснованно выбрать язык для решения конкретной задачи; облегчается освоение нового языка.

В течение 1980—1987 гг. в Московском институте электронной техники велись работы по расширению базовой версии языка ФОКАЛ с целью его максимальной адаптации к нуждам учебного процесса. В начале работ

имелись формальное описание ФОКАЛа с помощью формул Бэкуса — Наура, работающий интерпретатор базовой версии под названием ДС СМ для мини-ЭВМ СМ-4 и словесное описание этой версии языка. Основные требования, предъявляемые к интерпретатору, были следующими: объем не более 8 К байт, резидентное функционирование в ПЗУ, максимальная надежность, достаточно развитые средства работы пользователя с микроЭВМ, эффективное использование в учебной среде.

Рассмотрим новые средства интерпретатора, добавленные к базовой версии, а также наиболее существенные изменения в его основной части. Для повышения надежности интерпретатор был переработан таким образом, чтобы обеспечить его работу непосредственно в ПЗУ без необходимости его перезаписи в оперативную память (резидентное расположение и функционирование в ПЗУ). С этой целью были исключены все обращения по записи в адреса, занимаемые интерпретатором в ПЗУ, а для системных целей в оперативной памяти выделена небольшая область (72 ячейки) для указателей, флагов и переменных, связанных с конфигурацией системы (емкость памяти, типы устройств ввода—вывода и пр.). Составлена также программа «раскрутки» интерпретатора, которой автоматически передается управление сразу после включения микроЭВМ. Эта программа распределяет адресное пространство, устанавливает векторы прерываний и начальные значения переменных в системной области. Пользователь не имеет возможности изменять содержимое этих ячеек памяти, так как введена специальная защита.

Кроме того, подсчитывается и проверяется контрольная сумма содержимого ячеек ПЗУ, а также автоматически определяется непрерывная область исправных ячеек оперативной памяти. В случае несовпадения контрольной суммы происходит выход микроЭВМ в пультовый режим с индикацией адреса 140026_8. Если же встречается неисправная ячейка памяти, то интерпретатор устанавливает новую границу и правильно функционирует, но при меньшем объеме оперативной памяти.

Введено экранное редактирование программных строк, существенно упрощающее действия пользователя при вводе и корректировке программ.

С целью облегчения отладки программ диагностические сообщения выводятся в развернутом виде, т. е. наряду с указанием номера строки, где произошла ошибка,

отображается в текстовом виде и причина ошибки. Полезной для отладки программ оказалось введение возможности приостанова выполнения программы в любой момент времени, а затем ее продолжения.

Для оперативного контроля правильности функционирования интерпретатора введен оператор запуска встроенной тестовой задачи. По ее окончании выдается одно из сообщений: «ЭВМ исправна» либо «ЭВМ неисправна». Этот оператор оказался полезным особенно для новичков, так как позволяет в сомнительных случаях лишний раз убедиться в правильности работы ЭВМ. Кроме того, он используется для проведения отбраковки изготовленных микросхем ПЗУ на этапе технического контроля.

Существенно расширена библиотека встроенных функций: добавлены арксинус, арккосинус, тангенс, арктангенс, логарифмы натуральный и десятичный, показательная функция, функция обнуления таймера.

Достаточно широкие возможности предоставляет пакет графических функций*. Для возможности работы с пакетом в состав ДВК должен входить контроллер КГД. Для работы в составе учебной локальной сети в интерпретатор включены программы, поддерживающие обмен информацией с главной ЭВМ по последовательному каналу ИРПС. При передаче файлов подсчитывается и проверяется контрольная сумма пересылаемых байтов.

Кроме того, при разработке новой версии интерпретатор ФОКАЛа были устранены некоторые ошибки, имевшиеся в базовой версии. В частности, изменения коснулись программ, реализующих исключение и сжатие программных строк, вывода символьной информации на экран, резервирования места для переменных в оперативной памяти и др.

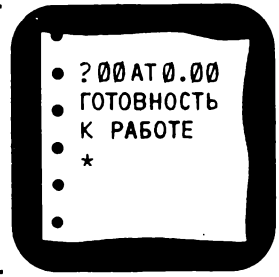
Интерпретатор ФОКАЛа изготовлен на одном из предприятий электронной промышленности в виде масочного ПЗУ. Конструктивно микросхема выполнена в виде стандартного пластмассового корпуса типа DIP и имеет 42 вывода. Последние цифры в ее обозначении К1801РЕ2-093 представляют номер зашивки.

---

* Основную работу по составлению и отладке пакета выполнил инженер А. Ф. Самойлов.

глава 2

Версия языка Фокал для начального обучения программированию



Для успешного овладения программированием должны быть соблюдены два условия. Первое — язык должен быть простым и легким для изучения, и второе — наличие микроЭВМ, за которой можно непосредственно проверить изучаемые конструкции языка. Без машины изучить язык программирования и научиться писать программы почти так же трудно, как научиться, например, ездить на велосипеде по инструкции. Все примеры в этой главе выполнялись на ЭВМ и ею же выводились на печать. Такой прием позволил свести к минимуму появление возможных ошибок. Полное совпадение начертаний букв и знаков, выводимых ЭВМ на экран дисплея, с аналогичными знаками и буквами в приводимых примерах облегчит проверку корректности последних.

2.1. Структура языка

Язык ФОКАЛ позволяет работать в двух режимах: диалоговом и программном.

В диалоговом режиме операторы вводятся непосредственно после * и выполняются немедленно после нажатия клавиши <ВК>. Символ * является знаком приглашения к выполнению любых действий. Строка без номера может содержать несколько операторов, разделенных ограничителем (;), и называется прямой командой. Прямая команда выполняется один раз и не может выполниться вторично, пока не будет набрана снова.

Если данная строка операторов должна быть сохранена в оперативной памяти для дальнейшего использования, то перед ней необходимо поместить номер строки. Строки операторов, помеченные номерами, называются косвенными строками операторов или косвенными командами.

Формат косвенной команды:

mm. nn <операторы>

где mm.nn — номер строки; mm — номер группы строк; nn — номер шага в группе.

Косвенная команда запоминается в памяти после нажатия клавиши <ВК> и не выполняется до тех пор, пока ей не будет передано управление.

Последовательность косвенных команд образует программу на языке ФОКАЛ, которая выполняется в порядке увеличения номеров строк, если нет дополнительных указаний, организуемых операторами переходов. Операторы, составляющие строку, реализуются в порядке их следования, начиная с первого; часть операторов строки может остаться нереализованной вследствие передачи управления другой строке.

Запуск программы на ФОКАЛе выполняется после введения прямой команды GO. Подобный режим работы называется программным. После завершения выполнения программы или прямой команды выдается символ *. Это означает, что система вышла в диалоговый режим и готова выполнять любые команды пользователя.

Для номеров строк в программах можно использовать любые номера с 1.01 до 99.99 с шагом 0.01, за исключением тех, которые оканчиваются на 00. После точки не обязательно указывать два знака, т. е. 2.1 эквивалентно 2.10.

Разрешается также использовать номера от 100.1 до 127.9 с шагом 0.1, за исключением тех, которые оканчиваются на 0.

В качестве алфавита языка ФОКАЛ используются буквы, цифры, ограничители, управляющие и редактирующие символы.

Буквы — латинский алфавит:

прописные — A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z;

строчные — a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z.

ФОКАЛ позволяет комментарии и текст примечаний записывать русскими прописными и строчными буквами.

Цифры — только арабские:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Цифры служат для образования чисел и номеров строк.

Ограничители. В качестве ограничителей используются:

а) знаки арифметических операций:

¬ — возведение в целую положительную степень,
* — умножение,
/ — деление,
+ — сложение,
— — вычитание;

б) знаки, используемые как разделители:
= — знак присваивания,
␣ — пробел,
<ВК> — возврат каретки,
. — десятичная точка для разделения целой и дробной части числа,
, — запятая,
; — точка с запятой,
< >( )[ ] — скобки для записи выражений,
<РУС> — русский регистр,
<ЛАТ> — латинский регистр.

Управляющие символы:

" — кавычки для печати текстовой информации,
% — задание формата вывода чисел,
? — знак трассировки,
! — возврат каретки и перевод строки,
— возврат каретки,
¤ — вывод на печать таблицы значений переменных,
@ — символ сохранения значения переменной в операторе ASK,
: — символ ожидания ввода значения переменной в операторе ASK.

Редактирующие символы:

<ГТ> — горизонтальная табуляция,
→ — сдвиг курсора вправо,
← — сдвиг курсора влево,
↦ — размыкание строки,
↤ — смыкание строки,
↱ — возврат курсора в начало строки,
<СТС> — стирание строки с позиции курсора,
<СБР> — сброс экрана,
↳ — аналогичен ВК,
_ — сброс выполнения строки.

Кроме того, используются знаки, управляющие ходом выполнения программы:

СУ/С — возвращение в диалоговый режим;

С1 — приостанов выполнения программы.

Продолжение выполнения программы осуществляется нажатием любой клавиши.

В ФОКАЛе могут быть использованы любые десятичные числа в диапазоне от 10^{-38} до 10^{+38}. Числа могут быть со знаком «+» или «—», дробные числа — с десятичной точкой или в экспоненциальном формате.

Точность представления чисел в ФОКАЛе ограничена шестью значащими цифрами. Если указано более шести цифр, число будет округлено до шести знаков.

Константа в экспоненциальном формате состоит из двух частей: собственно константы и порядка (целой степени десяти). Порядок записывается целым числом, перед которым стоит буква Е. Он может иметь знак «+» или «—».

Следующие представления чисел идентичны:

```
70
70.00
7E + 01
700E — 01
70.00003
```

Переменные в ФОКАЛе описываются их именами, а если переменная имеет индексы, то и значениями индексов.

Пример:

```
X(1), TOP(1), KE(5, 6), Y, Z, YX
```

Имена переменных могут состоять из одного или нескольких символов. Первый символ должен быть буквой (прописной или строчной буквой латинского алфавита), но не прописной буквой F, которая используется для имен функций. Остальные символы могут быть буквами или цифрами. Пользователь может написать имя переменной, состоящее более чем из двух символов, но ФОКАЛ использует только первые два символа для распознавания переменной.

Переменные с индексами могут иметь не более двух индексов. Индексы заключаются в круглые, квадратные или угловые скобки и разделяются запятой в двухиндексных переменных.

Наряду с константами в качестве индексов могут использоваться арифметические выражения, но в качестве значения индекса будет принята целая часть значения выражения. Индексы могут принимать значения от —128 до +127 для двухиндексной переменной и от —32768 до +32 767 для одноиндексной переменной. При превышении этих диапазонов значения берутся по модулю соответствующей границы.

Переменные в ФОКАЛе не требуют объявления. Пользователь должен знать, что переменные, появляющиеся в тексте программ или в прямых командах, интерпретируются ФОКАЛом следующим образом. Если переменная уже была введена ранее, то интерпретатор обращается к таблице переменных и отыскивает ее там, устанавливая указатель на ячейку, содержащую значение переменной. Если же переменная в таблице не обнаружена, то она заносится в память в том смысле, что отводится место для размещения ее имени и очищаются ячейки для размещения ее значения. Таким образом, если переменные ранее не получили значения с помощью операторов присваивания или ввода, то по умолчанию они имеют нулевое значение. Это одна из особенностей языка ФОКАЛ.

Любая переменная занимает в памяти четыре слова: одно слово для имени переменной, одно слово для индексов, два слова для значения. Простые переменные отыскиваются интерпретатором быстрее, если их имена начинаются с разных символов.

ФОКАЛ позволяет использовать одно- и двумерные массивы (или иначе, одно- и двухиндексные переменные). Индекс может быть числом, именем переменной или арифметическим выражением.

Пределы изменения индекса:

от —128 до +127 для двумерного массива;

от —32 768 до +32 767 для одномерного массива.

Индекс заключается в скобки, а в случае двумерного массива индексы разделяются запятой.

Например: X (137), PO [I, J], K<15, 20>.

Массивы в ФОКАЛЕ не требуют объявления.

Арифметические выражения в ФОКАЛе аналогичны математическим формулам. Арифметическое выражение может состоять из следующих элементов: чисел, переменных, функций, знаков арифметических операций и скобок. Следует иметь в виду, что отдельное число, отдельная переменная, отдельная функция также считается

арифметическим выражением и является его частным случаем.

Арифметические выражения вычисляются в соответствии с приоритетом операций и наличием скобок, т. е. с учетом следующих обстоятельств:

1) выражения, заключенные в скобки, имеют наивысший приоритет; это означает, что они вычисляются прежде выражений, стоящих вне скобок;

2) при отсутствии скобок порядок выполнения операций следующий: вычисление значения функции; возведение в целую, положительную степень (↑); умножение *; деление (/); сложение (+); вычитание (—);

3) в конфликтных ситуациях, когда п. 1 и 2 недостаточны для определения приоритета операций, следует помнить, что интерпретатор вычисляет выражение слева направо.

Если выражение содержит скобки внутри скобок, то сначала вычисляются внутренние скобки, а затем внешние. В ФОКАЛе можно использовать три вида скобок; круглые ( ), квадратные [ ] и угловые < >. Интерпретатором они воспринимаются одинаково, но следует помнить, что каждой открывающейся скобке должна соответствовать закрывающая скобка того же типа.

Для возведения числа y в отрицательную или дробную степень x следует пользоваться формулой $z = e^{x \ln y}$.

2.2. Оператор вывода данных

Практическое освоение языка ФОКАЛ целесообразно начинать с оператора TYPE (печатать). Его формат:

TYPE <выражение>

где выражение может быть любым арифметическим выражением, записанным по правилам языка ФОКАЛ

Примеры.

```
*TYPE 18/3*2
     3.0000*

*TYPE 18/(3*2)
     3.0000*

*TYPE (18/3)*2
    12.0000*
```

```
*TYPE 2^2^3
     64.0000*

*TYPE 2^(2^3)
    256.0000*
```

Оператор TYPE может объединять вычисление нескольких формул, которые разделяются запятыми.

Пример.

```
*TYPE 8/2-3,45-18
      1.0000     27.0000*
```

При начальном включении ЭВМ по умолчанию устанавливается формат вывода чисел с фиксированной точкой, т. е. оператор TYPE выводит 11 знаков: ведущий пробел, знак числа, четыре позиции для целой части числа, десятичную точку, четыре позиции для дробной части числа, вместо ведущих нулей и знака «+» выводятся пробелы.

Пример.

```
*TYPE -003.45
 -    3.4500*

*TYPE 38.456
     38.4560*
```

Установка другого формата чисел с фиксированной точкой производится следующим образом:

*TYPE % $w.0d$

где $0<w\leqslant 8$, $0\leqslant d\leqslant 8$.

Например, указание формата вывода целых чисел можно задать следующим образом:

```
*TYPE %4.00
*TYPE -1234
 -1234*
```

Здесь для вывода числа отводится шесть позиций. Ведущий пробел необходимо для разделения чисел, следующих друг за другом.

Пример.

```
*TYPE -8542,-3246,9281,+4186
 -8542 -3246  9281  4186*
```

Если число превышает заданный формат, то оно автоматически преобразуется в экспоненциальный формат.

Пример.

```
*TYPE 12345
  0.123450E+05*
```

что в математической записи эквивалентно

$0{,}123450 \cdot 10^5$.

Установка экспоненциального формата вывода требует использования указателя формата %. В этом случае для числа отводится 14 позиций.

Пример.

```
*TYPE %,3.951
  0.395100E+01*
```

Заданный в операторе TYPE формат вывода чисел действует до тех пор, пока не будет изменен другим указателем формата.

Используя оператор TYPE, можно выводить любой текст на экран дисплея. Для этого выводимый текст заключается в кавычки. Перед печатью русского текста следует нажать клавишу <РУС> (русский регистр), после окончания — <ЛАТ> (латинский регистр).

Пример.

```
*TYPE " РЕШЕНИЕ СИСТЕМЫ УРАВНЕНИЙ "
РЕШЕНИЕ СИСТЕМЫ УРАВНЕНИЙ
```

Нажатие клавиш <РУС> и <ЛАТ> вызывает появление на экране символа ▌. Этот значок удобен при отладке программ. При исполнении оператора TYPE значок ▌ на экран не выводится.

Символ ! вызывает перевод курсора в начало следующей строки.

Пример.

```
*TYPE  " ПРИМЕР " ,!," 2+3=" ,2+3
ПРИМЕР
2+3=       5.0000*
```

Символ # используется для возврата курсора в начало текущей строки. Проследим на экране выполнение следующего примера:

```
*TYPE " ПРИМЕР ",#," 2+3=",2+3
```

При выполнении этого оператора ЭВМ печатает слово ПРИМЕР, затем возвращает курсор в начало этой строки и на том же месте напечатает 2 + 3 = 5.ØØØØ, уничтожив слово ПРИМЕР.

2.3. Оператор присваивания SET

Оператор SET служит для присваивания переменной или индексной переменной, стоящей слева от знака =, вычисленного значения арифметического выражения, стоящего справа от знака = .

Формат оператора:

SET <имя переменной> = <арифметическое выражение>

Пример.

```
*SET X=3.14159/180
```

После выполнения оператора присваивания в памяти ЭВМ появляется новая переменная с именем X, имеющая значение вычисленного выражения. Это легко проверить с помощью оператора TYPE:

```
*TYPE X
     0.0175*
```

Следует помнить, что слева от знака присваивания = может быть только имя переменной, но не выражение, поэтому конструкции следующего вида запрещены:

```
*SET X+Y=5
*SET A=B=10
*SET Z=7,C=4
*
```

Если при выполнении оператора

```
*SET A=C
```

где C — переменная ранее не использованная в программе, то ее значение считается равным Ø. Поэтому переменная A получает значение, также равное Ø.

Существует возможность вывести на экран дисплея значения всех переменных, находящихся в оперативной памяти в данный момент. Для этой цели используется оператор TYPE с символом Ω .

Пример.

```
*SET A=3;SET X(5)=10;SET Y(1,4)=8
*TYPE ¤
S A()=     3.0000
S Y(+01,+04)=     8.0000
S X(+05,+00)=    10.0000
*
```

Элементы вывода, стоящие в операторе TYPE после символа ¤, а также операторы в строке после TYPE ¤ игнорируются.

Следует заметить, что информация по оператору TYPE ¤ выдается как запись оператора SET в режиме прямых команд.

В последней версии интерпретатора языка ФОКАЛ оператор SET сделан факультативным, как это имеет место в языке ФОРТРАН и в современных версиях языка БЕЙСИК.

В этом случае формат оператора присваивания имеет вид

<имя переменной> = <арифметическое выражение>

Пример.

```
*X=2;Y=3
*Z=X^2+Y^2
*TYPE Z
     13.0000*
```

Использование такой формы оператора присваивания более удобно и приводит к экономии ячеек памяти.

2.4. Оператор ввода

Оператор ASK служит для ввода значений переменных, переменных с индексами и элементов массивов по запросу программы. Для удобства пользователя ввод может сопровождаться печатью на устройстве вывода текстов примечаний, заключенных в кавычки с использованием символов ! и #.

Оператор ASK может использоваться как в прямых, так и в косвенных командах совместно с другими операторами языка ФОКАЛ.

Формат оператора:

ASK <список элементов ввода — вывода>

Элементы ввода — вывода отделяются запятыми. Элементами в операторе ASK могут быть:

простые переменные, переменные с индексами;

тексты примечаний — набор допустимых в языке ФОКАЛ символов, заключенных в кавычки;
символы, управляющие выводом: !, #;
указатели форматов;
символ вывода значений всех переменных
Пример.

```
*ASK X;TYPE !,'X=',X
:10
X=     10.0000*
```

Ввод данных осуществляется после печати оператором ASK символа :. Число вводится следующим образом: устройством ввода последовательно вводятся символы, изображающие число. Ввод числа заканчивается одним из ограничителей: запятой, точкой с запятой, пробелом, возвратом каретки.

Одним оператором ASK можно ввести значения нескольких переменных.

Пример.

```
*ASK X,Y(1),Z(3,2),!,"КОНЕЦ"
:1
:2
:3
КОНЕЦ*
```

Переменной X присваивается значение 1, переменной Y (1) — значение 2, переменной Z (3, 2) — 3.

Длина вводимой в ответ на символ : последовательности символов не должна превышать 71. Ввод 72-го символа вызывает вывод сообщения об ошибке 16 — переполнение строки в операторе ASK — и останов программы. В ответ на запрос оператора ASK может вводиться последовательность букв латинского алфавита от A до Z, исключая E. Это приводит к присвоению переменной значения, вычисляемого следующим образом:

```
*ASK X;TYPE !,X
:NO
   155.0000*
```

В этом случае переменная X получит значение

$$X = 14 \cdot 10^1 + 15 \cdot 10^0 = 155,$$

так как это число будет представлено в интерпретаторе как двузначное с весами для единиц и десятков, равными порядковому номеру буквы в алфавите (A = 1, B = 2, C = 3, ..., Y = 25, Z = 26) (см. Приложение 2).

Оператором ASK в качестве значения переменной может быть присвоено вычисленное значение арифметического выражения, вводимого с устройства ввода. Если в качестве присваиваемого значения переменной вводится арифметическое выражение, то ему должен предшествовать знак «+» или «—».

Пример.

```
*C:  Ф О К А Л
  1.10 SET A=7;SET B=4
  1.20 ASK "X=",X;TYPE !,X
*GO
X=:+A-B

     3.0000*
```

При повторном счете по программе и, следовательно, при повторном запросе на ввод оператором ASK текущие значения переменных можно сохранить. Для этого достаточно нажать клавишу @. На устройстве вывода будет напечатан символ @.

Пример.

```
            C:  Ф О К А Л
  1.10 SET Y=2;TYPE Y,!
  1.20 ASK X,Y,Z
*GO
     2.0000
:1
:@:3
*
```

Переменная Y сохранит ранее присвоенное ей значение 2.

Символы ! и #, управляющие выводом, влияют только на расположение текстов примечаний.

Изменение формата вывода чисел осуществляется в операторе ASK указателем формата аналогично оператору TYPE. Числовые значения переменных, вводимых оператором ASK, не зависят от примененных указателей формата и, кроме того, могут вводиться в произольной форме.

В операторе ASK может быть использован символ ¤ для вывода значений всех переменных аналогично оператору TYPE.

Использование в операторе ASK текстов примечаний позволяет правильно ориентироваться при вводе чисел, особенно в тех случаях, когда данных много.

Пример.

```
C:  Ф О К А Л
  1.10 ASK "X=",X,"Y=",Y,"Z=",Z
  1.20 TYPE X,Y,Z
*GO
X=:5
Y=:6
Z=:12
     5.0000      6.0000     12.0000*
```

В операторе ASK имеется возможность редактирования вводимых значений переменных. Для удаления ошибочно введенных символов в ответ на запрос оператора ASK используется клавиша <ЗБ>. При этом последний введенный символ уничтожается.

Для сброса всей введенной строки необходимо нажать клавишу (подчеркивание). После этого можно набирать значение переменной заново.

2.5. Операторы управления программой

В языке ФОКАЛ для управления программой служат следующие операторы: оператор безусловного перехода GO, оператор условного перехода IF, оператор цикла FOR, оператор передачи управления с возвратом DO, оператор возвращения в режим диалога QUIT, оператор выхода из подпрограммы RETURN. Рассмотрим их в перечисленном порядке следования.

В ФОКАЛе в качестве оператора безусловного перехода используется оператор GO. Он может использоваться как в прямых, так и в косвенных командах в любом месте командной строки.

Формат оператора.

GO n

где n — номер строки, которой требуется передать управление, может отсутствовать, быть задан дробным числом или именем переменной.

Если n в операторе GO отсутствует, то управление передается строке ФОКАЛа с наименьшим номером. Это свойство оператора GO можно использовать для запуска программы; если начало программы совпадает со строкой с наименьшим номером, то ее значение будет интерпретироваться как номер строки, которой требуется передать управление.

Ссылка оператором GO на несуществующую строку вызывает сообщение об ошибке 5 — НЕСУЩЕСТВУЮЩИЙ НОМЕР СТРОКИ.

Операторы ФОКАЛа, находящиеся после оператора GO в командной строке, не исполняются и рассматриваются как комментарии.

Пример.

```
*GO
```

Управление передается строке с наименьшим номером.

```
*GO 3.15;TYPE "ПЕРЕХОД"
```

Управление передается строке 3.15. Оператор TYPE не выполняется.

```
*7.1 SET M=3*2.1;GO M
```

При выполнении строки 7.1 будет вычислено значение переменной M. Оператор GO передаст управление строке с номером 6.30.

Оператор IF позволяет передавать управление по результатам сравнения с нулем значения некоторого арифметического выражения.

Оператор IF аналогичен арифметическому оператору IF языка ФОРТРАН. Оператор IF может использоваться как в прямых, так и в косвенных командах в любом месте командной строки.

Формат оператора:

IF (A) n1, n2, n3

где n1, n2, n3 — номера командных строк; A - арифметическое выражение.

В табл. 2.1 приведены возможные виды записи условного оператора IF и порядок его выполнения.

Таблица 2.1

Вид оператора	Значение A	Куда передается управление
IF (A) n1, n2, n3	$A < 0$	Командной строке n1
	$A = 0$	Командной строке n2
	$A > 0$	Командной строке n3
IF (A) n1, n2	$A < 0$	Командной строке n1
	$A = 0$	Командной строке n2
	$A > 0$	Следующему оператору
IF (A) n1	$A < 0$	Командной строке n1
	$A = 0$, $A > 0$	Следующему оператору

Пример.

```
*C:  Ф О К А Л
  1.10 ASK Y;IF (Y-10)1.2,1.3,1.4
  1.20 TYPE "Y<10";GO 1.1
  1.30 TYPE "Y=10";GO 1.1
  1.40 TYPE "Y>10";GO 1.1
*GO
:17
Y>10:4
Y<10:10
Y=10:
```

Для выхода из этой программы в диалоговый режим необходимо ввести управляющий символ СУ/С. Для этого нужно нажать клавишу СУ и, не отпуская ее, клавишу С. При этом машина напечатает:

```
? 00 AT 1.10
```

ГОТОВНОСТЬ К РАБОТЕ

В качестве номеров строк n1, n2, n3 могут использоваться имена переменных. В том случае, когда указанная в операторе IF командная строка отсутствует, работа программы прекращается и выдается сообщение об ошибке 5 — НЕСУЩЕСТВУЮЩИЙ НОМЕР СТРОКИ.

Если в операторе IF в качестве арифметического выражения используются нецелые числа, то проверка на 0 не всегда осуществима из-за плавающей арифметики.

Пример.

```
    C:  Ф О К А Л
  1.10 SET X=0.2+9.8
  1.20 IF (X-10)1.30,1.40,1.30
  1.30 TYPE " ЧИСЛО X ВЫЧИСЛЕНО С ПОГРЕШНОСТЬЮ ";QUIT
  1.40 TYPE " ЧИСЛО X ВЫЧИСЛЕНО ТОЧНО ;QUIT
  1.50 TYPE X
* GO
 ЧИСЛО X ВЫЧИСЛЕНО С ПОГРЕШНОСТЬЮ *
```

Чтобы ликвидировать эту проблему, пользователь должен либо избежать применения нецелой арифметики в операторе IF, либо контролировать программным способом погрешность представления чисел с плавающей точкой.

Пример.

```
C:  Ф О К А Л
 1.10 SET X=2+8
 1.20 IF (X-10)1.30,1.40,1.30
 1.30 TYPE " ЧИСЛО X ВЫЧИСЛЕНО С ПОГРЕШНОСТЬЮ ";QUIT
 1.40 TYPE " ЧИСЛО X ВЫЧИСЛЕНО ТОЧНО ";QUIT
 1.50 TYPE X
*GO
 ЧИСЛО X ВЫЧИСЛЕНО ТОЧНО *
```

С помощью оператора IF можно реализовать любую из операций отношений:

$=$ — равно,
$\neq$ — не равно,
$>$ — больше,
$\geqslant$ — больше или равно,
$<$ — меньше,
$\leqslant$ — меньше или равно.

Условимся, что если отношение выполняется, то управление передается оператору с меткой n1, а если не выполняется, то с меткой n2:

IF (X) n2, n1, n2 — реализация отношения равно,
IF (X) n1, n2, n1 — реализация отношения не равно,
IF (X) n2, n2, n1 — реализация отношения больше,
IF (X) n2, n1, n1 — реализация отношения больше или равно,
IF (X) n1, n2, n2 — реализация отношения меньше,
IF (X) n1, n1, n2 — реализация отношения меньше или равно.

Оператор IF позволяет программировать циклические вычисления. Приведем пример программы нахождения суммы натурального ряда чисел от 1 до 100:

```
C:  Ф О К А Л
  1.10 SET I=1;SET S=0
  1.20 SET S=S+I;SET I=I+1
  1.30 IF (I-100)1.2,1.2,1.4
  1.40 TYPE "СУММА 100 ЧИСЕЛ РАВНА",S
*GO
СУММА 100 ЧИСЕЛ РАВНА  5050.0000*
```

Следует обратить внимание на операторы присваивания в строке с номером 1.20. Так, оператор SET S = S + I имеет следующий смысл: взять текущее значение переменной S, прибавить к нему значение переменной I и полученное значение вновь присвоить переменной S.

Оператор FOR используется для записи алгоритмов, в которых параметр меняется с некоторым шагом, т. е. для организации циклов.

Формат оператора:

FOR X = A, B, C; <опер.> ; <опер.> ; ... <опер.> <ВК>

где X — параметр цикла; A — начальное значение пара-

метра; B — шаг изменения параметра; C — конечное значение параметра.

Операторы, расположенные в той же командной строке до символа <ВК>, образуют область действия оператора цикла FOR или тело оператора цикла. Параметром цикла может быть любая переменная. В качестве A, B, C могут использоваться арифметические выражения. Если величина шага B в записи оператора отсутствует, то по умолчанию цикл выполняется с шагом «1».

Оператор цикла работает следующим образом:

1) параметру цикла X присваивается значение A (SET X = A);

2) выполняются операторы из области действия цикла;

3) к параметру цикла прибавляется величина шага цикла (SET X = X + B);

4) если $X \leqslant C$, то выполняется переход на операторы из области действия цикла, если $X > C$, управление передается командной строке, следующей за строкой, содержащей оператор цикла.

Из п. 1—4 следует, что для нормальной работы циклов необходимо соблюдение условий $C > A$ и $B > 0$. В любом случае операторы из области действия цикла выполняются хотя бы один раз при X = A. Нулевое или отрицательное значение шага цикла B может привести к бесконечному повторению операторов из области действия цикла.

Если в области действия цикла есть оператор, передающий управление строке с некоторым номером, то эта строка (обращаем внимание — именно одна строка) включается в область действия цикла оператора FOR. Это одна из принципиальных особенностей организации циклов в ФОКАЛе и о ней следует помнить при составлении программ.

Другие особенности оператора цикла FOR:

при выходе из цикла по его окончании ($X > C$) параметр цикла сохраняет свое значение на момент выхода из цикла;

изменение начального значения, шага и конечного значения параметра цикла операторами из области действия цикла не изменяет числа его повторений (при первом чтении оператора цикла интерпретатор запоминает в своей внутренней области значения A, B и C и в дальнейшем использует только эти значения);

параметр цикла X может изменяться операторами из

области действия цикла. После этого цикл выполняется при этом измененном значении, увеличенном на шаг цикла, если $X \leqslant C$. Таким образом, операторами из области действия цикла можно управлять числом повторений цикла.

Проиллюстрируем сказанное примерами:

1.

```
C:  Ф О К А Л
 1.10 A=1;B=1;C=10
 1.20 FOR X=A,B,C;TYPE X;A=10;B=20;C=30;T A,B,C,!
 1.30 TYPE !!"    ПОСЛЕ ВЫХОДА ИЗ ЦИКЛА X=",X,!
*GO
     1.0000     10.0000     20.0000     30.0000
     2.0000     10.0000     20.0000     30.0000
     3.0000     10.0000     20.0000     30.0000
     4.0000     10.0000     20.0000     30.0000
     5.0000     10.0000     20.0000     30.0000
     6.0000     10.0000     20.0000     30.0000
     7.0000     10.0000     20.0000     30.0000
     8.0000     10.0000     20.0000     30.0000
     9.0000     10.0000     20.0000     30.0000
    10.0000     10.0000     20.0000     30.0000

   ПОСЛЕ ВЫХОДА ИЗ ЦИКЛА X=     11.0000
```

Изменение операторами, находящимися в области действия цикла, значений переменных A, B, C никак не сказалось на числе повторений цикла. После завершения цикла параметр X принял значение 11, т. е. C + B, заданные в строке с номером 1.10.

2.

```
C:  Ф О К А Л
 1.10 A=1;B=1;C=10
 1.20 FOR X=A,B,C;TYPE X;X=10
 1.30 TYPE !!"    ПОСЛЕ ВЫХОДА ИЗ ЦИКЛА X=",X,!
*GO
     1.0000

   ПОСЛЕ ВЫХОДА ИЗ ЦИКЛА X=     11.0000
*
```

Цикл был выполнен всего один раз, так как в области действия оператора цикла параметру цикла X было присвоено конечное значение, равное 10.

3.

```
C:  Ф О К А Л
 1.10 A=1;B=1;C=0
 1.20 FOR X=A,B,C;TYPE X
 1.30 TYPE !!"    ПОСЛЕ ВЫХОДА ИЗ ЦИКЛА X=",X,!
*GO
     1.0000

   ПОСЛЕ ВЫХОДА ИЗ ЦИКЛА X=      2.0000
*
```

Цикл выполнился всего один раз, так как конечное значение параметра меньше начального.

4.

```
C:  Ф О К А Л
 1.05 COMMENT ПРОГРАММА ПЕЧАТИ НУЛЕВЫХ ЭЛЕМЕНТОВ МАССИВА
 1.10 TYPE %2.00
 1.20 A(1)=0;A(2)=2;A(3)=3;A(4)=4;A(5)=0;A(6)=6
 1.30 FOR I=1,6;IF (A(I))1.50,1.50,1.60
 1.40 TYPE "  КОНЕЦ 1  ",!;QUIT
 1.50 TYPE "A(",I,")=",A(I),!
 1.60 COMMENT ЧИСЛО БОЛЬШЕ НУЛЯ
 1.70 TYPE "  КОНЕЦ 2  ",!
*GO
A(   1)=   0
A(   5)=   0
  КОНЕЦ 1
*
```

В этом примере в зависимости от условия, выполняемого в операторе IF (строка 1.30), в область действия цикла включается либо строка 1.50, либо 1.60. Обратите внимание, что строка с номером 1.70 ни разу не выполняется.

Чтобы включить в область действия цикла не одну строку, а несколько, следует воспользоваться оператором безусловного перехода GO.

Например, в строке с номером 1.50 добавим оператор перехода

```
 C:  Ф О К А Л
  1.50 TYPE "A(",I,")=",A(I),!;G 1.70
*
```

Если снова запустить программу на счет, результаты будут следующими:

```
 *GO
A(   1)=   0
  КОНЕЦ 2
A(   5)=   0
  КОНЕЦ 2
  КОНЕЦ 1
*
```

Здесь видно, что в область действия цикла включена строка с номером 1.70:

5.

```
C:  Ф О К А Л
  1.10 FOR I=1,2;FOR J=3,4;TYPE I,!,J,!
*GO
   1
   3
   1
   4
   2
   3
   2
   4
*
```

В приведенном примере в область действия цикла входит другой оператор цикла. Цикл по J является внутренним для цикла I. При каждом фиксированном значении I параметр J «пробегает» все свои значения, с которыми выполняются операторы из области действия цикла. Такие циклы называются вложенными.

6.

```
C:  Ф О К А Л
  1.10 SUM=0
  1.20 FOR I=1,100;SUM=SUM+I
  1.30 TYPE "СУММА 100 ЧИСЕЛ РАВНА ",SUM,!
*GO
СУММА 100 ЧИСЕЛ РАВНА   5050.0000
*
```

Аналогичная программа была приведена при рассмотрении оператора условного перехода IF. Как видно, оператор цикла позволяет записать тот же алгоритм в более простой и ясной форме.

Циклические вычисления являются важной частью программирования и фактически составляют его ядро.

Оператор DO предназначен для передачи управления на заданную командную строку или группу строк с последующим возвратом на оператор, следующий за оператором DO. Оператор DO может быть использован в режиме прямых и косвенных команд в любом месте командной строки.

Формат оператора:

```
DO n
```

где n — номер командной строки или группы строк, на которые передается управление.

Значение n может быть задано: целым числом (номер группы), дробным числом (номер строки), именем переменной (значением переменной должен быть либо номер группы, либо номер строки).

В том случае, когда n ссылается на отсутствующий в программе номер строки или группы строк, работа программы прерывается и выдается сообщение об ошибке 6 — НЕСУЩЕСТВУЮЩИЙ НОМЕР ГРУППЫ ИЛИ НОМЕР СТРОКИ В ОПЕРАТОРЕ DO.

В том случае, когда n — дробное число, т. е. указывает на номер строки, по оператру DO управление передается этой строке, а после ее исполнения — оператору, следующему за оператором DO.

Пример.

```
 C:  Ф О К А Л
  1.10 V=3.4;DO V;X=V+1.6
  1.20 TYPE X,!;QUIT
  3.40 TYPE V,!
*GO
      3.4000
      5.0000
*
```

В примере после выполнения DO V выполняется строка 3.40. После этого управление передается оператору $X = V + 1.6$.

В том случае, когда n — целое число, т. е. указывает на группу строк, управление передается на строку с минимальным в данной группе номером. Строки группы выполняются в порядке возрастания номеров строк, если в них нет операторов перехода либо до исчерпания номеров строк данной группы в программе, либо до встречи с оператором RETURN (см. далее). После этого управление передается оператору, следующему непосредственно за DO.

Пример.

1.

```
 C:  Ф О К А Л
  1.10 DO 5
  1.20 GO 6.2
  5.10 TYPE 1,!
  5.20 TYPE 2,!
  6.10 TYPE 6,!
  6.20 TYPE 8,!
*GO
      1.0000
      2.0000
      8.0000
*
```

2.

```
C:  Ф О К А Л
  1.10 DO 5
  1.20 TYPE 3,!;QUIT
  5.10 TYPE 1,!;GO 6.1
  5.20 TYPE 2,!
  6.10 TYPE 4,!
  6.20 TYPE 5,!
*GO
     1.0000
     4.0000
     3.0000
*
```

Если в строке или группе строк, на которые передается управление оператором DO, находятся операторы перехода GO или IF, то они вызывают передачу управления только на одну строку, после выполнения которой (если в этой строке, в свою очередь, не содержатся операторы перехода) управление перейдет к оператору, следующему за оператором DO.

Примеры.

1.

```
C:  Ф О К А Л
  1.10 DO 5.1
  1.20 TYPE 3,!;QUIT
  5.10 TYPE 1,!;GO 6.1
  6.10 TYPE 2,!
*GO
     1.0000
     2.0000
     3.0000
*
```

2.

```
C:  Ф О К А Л
  1.10 DO 7
  1.20 GO 7.4
  7.10 TYPE 1,!
  7.20 TYPE 2,!;RETURN
  7.30 TYPE 3,!
  7.40 TYPE 4,!
*GO
     1.0000
     2.0000
     4.0000
*
```

Операторы DO могут быть вложенными, т. е. во время выполнения оператора DO может встретиться другой такой же оператор. Глубина вложенности операторов DO определяется размером стека интерпретатора. При большой глубине вложенности операторов DO возможно переполнение стековой памяти интерпретатора. Эта ситуация вызовет сообщение об ошибке 9 — ПЕРЕПОЛНЕН СТЕК.

Максимальное число вложений — 35.

Пример.

```
C:  Ф О К А Л
  1.10 DO 2
  2.10 DO 1
*
*GO
?09 AT 2.10
ПЕРЕПОЛНЕН СТЕК
*
```

Оператор QUIT предназначен для останова выполнения программы и перевода интерпретатора в режим прямых команд. При встрече с ним работа программы приостанавливается с сохранением всех значений переменных и на устройстве вывода печатается символ *.

Оператор QUIT может использоваться как в прямых, так и в косвенных командах в любом месте командной строки.

Оператор QUIT не имеет операндов. Работа программы может быть продолжена с необходимой строки с помощью оператора GO.

Оператор RETURN используется для возврата к оператору, следующему за оператором DO.

Оператор RETURN не требует операндов. Операторы, следующие за оператором RETURN в этой строке, никогда не выполняются, поэтому после него есть смысл размещать только комментарии.

Пример.

```
C:  Ф О К А Л
  1.10 ASK Y;DO 3
  1.20 TYPE X,!;QUIT
  3.10 IF (Y)3.20,3.30,3.40
  3.20 SET X=-1;RETURN
  3.30 SET X= 0;RETURN
  3.40 SET X=+1;RETURN
*GO
:-5
 -    1.0000
*
```

Здесь в группе 3 размещена подпрограмма, обращение к которой производится оператором DO 3. Завершение работы подпрограммы, а следовательно, и выхода из нее возможны на строках 3.2, 3.3 или 3.4. Этим и объясняется наличие в каждой из них оператора RETURN.

Заканчивая рассмотрение основных операторов ФОКАЛа, приведем пример программы поиска в массиве минимального числа и его номера:

```
C:  Ф О К А Л
 1.10 TYPE %4.00,"ВВЕДИТЕ 10 ЧИСЕЛ",!
 1.20 FOR I=1,10;ASK X(I)
 1.25 XMIN=X(1);N=1
 1.30 FOR I=2,10;DO 2
 1.40 TYPE !"ПЕРВОЕ МИНИМАЛЬНОЕ ЧИСЛО",XMIN
 1.45 TYPE !"ЕГО ПОРЯДКОВЫЙ НОМЕР",N;QUIT
 2.10 IF (X(I)-XMIN)2.2,2.15,2.15
 2.15 RETURN
 2.20 XMIN=X(I);N=I
*GO
ВВЕДИТЕ 10 ЧИСЕЛ
:15 :67 :32 :9 :47 :25 :18 :87 :44 :11

ПЕРВОЕ МИНИМАЛЬНОЕ ЧИСЛО      9
ЕГО ПОРЯДКОВЫЙ НОМЕР      4*
```

В этой программе для оператора присваивания (строки 1.25 и 2.20) выбрана факультативная форма.

2.6. Отладка программ

Отладка программ включает в себя следующие моменты:

редактирование исходного текста, внесение в него изменений и дополнений;

трассировку — управление распечаткой текста программы во время ее выполнения;

автоматическую выдачу кода и текстового сообщения об ошибке.

В ФОКАЛе имеется возможность управления программой с помощью управляющих клавиш. Кроме того, в ФОКАЛе существует оператор для запуска контрольной задачи.

Оператор WRITE предназначен для вывода строки текста, группы строк либо текста всей программы. Оператор WRITE может использоваться как в прямых, так и в косвенных командах в любом месте командной строки.

Формат оператора:

```
WRITE n
```

где n может быть номером группы строк, номером строки, буквой A (от ALL — все) или отсутствовать.

Примеры.

*WRITE 3.1 — печатает строку текста 3.1;

*WRITE 3 — печатает группу строк с номером 3.

Выполнение оператора WRITE приводит к печати всего текста программы, находящейся в памяти.

В том случае, если n в операторе WRITE указывает на отсутствующую группу или строку, ошибка не фикси-

руется, при этом на устройство печати выводится символ *.

Оператор ERASE предназначен для уничтожения в оперативной памяти либо только всех переменных, либо только текста программы (всего или заданных строк), либо всего текста и всех переменных.

Формат оператора:

ERASE n

где n может быть номером строки, номером группы строк, символом T или TEXT, символом A или ALL или отсутствовать.

Ссылка оператором ERASE на строки с номерами 1—127, не вызывает ошибку. Ссылка на строки, не входящие в этот интервал, вызывает ошибку с кодами 1, 4 или 5 (см. Приложение 1).

*ERASE 7.1 — уничтожает строку 7.1 и все переменные,

*ERASE 7 — уничтожает группу строк 7 и все переменные,

*ERASE TEXT — уничтожает только весь текст программы (оставляет переменные в памяти),

*ERASE ALL — уничтожает весь текст программы и все переменные,

*ERASE — уничтожает все переменные (оставляет текст программы в памяти).

Оператор ERASE для уничтожения всех переменных может использоваться как в прямых, так и в косвенных командах в любом месте строки. С помощью такого оператора ERASE можно динамически управлять памятью, отводимой под переменные в процессе выполнения программы.

Оператор ERASE n также может использоваться как в прямых, так и в косвенных командах, но следует иметь в виду, что после его выполнения происходит прекращение выполнения программы и переход в диалоговый режим (выводится символ *).

Оператор MODIFY используется для редактирования программной строки, находящейся в оперативной памяти.

Формат оператора:

MODIFY n

где n — номер строки.

После того как пользователь набрал MODIFY n и нажал клавишу <ВК>, интерпретатор ФОКАЛа выводит на экран дисплея содержимое нужной строки. Далее можно редактировать строку, используя редактирующие клавиши клавиатуры дисплея:

<ГТ> — горизонтальная табуляция (сдвиг курсора на 8 позиций вправо);

→ — сдвиг курсора вправо на одну позицию;

← — сдвиг курсора влево на одну позицию;

⊢► — размыкание строки вправо с позиции, где расположен курсор;

◄⊣ — смыкание строки справа от позиции, где расположен курсор;

↵ — возврат курсора в начало строки;

〈СТС〉 — стирание строки с позиции курсора;

〈СБР〉 — сброс экрана (выполняется очистка экрана и сбрасывается выполнение набранной строки);

↵ — аналогичен 〈ВК〉;

_ — символ ПОДЧЕРКИВАНИЕ, выполняет сброс набранной строки.

Процесс редактирования строки с использованием оператора MODIFY заканчивается нажатием клавиши <ВК>. После появления на экране дисплея символа * система готова для выполнения любых действий пользователя. Если в процессе редактирования произошло переполнение строки, то интерпретатор автоматически заканчивает выполнение оператора MODIFY.

Оператор MODIFY не может использоваться для изменения номера строки. Чтобы заменить строку, надо набрать номер строки и новые операторы. Прежняя строка автоматически уничтожается.

Редактирование при вводе новой строки или прямой команды также осуществляется редактирующими клавишами. Если в процессе ввода и редактирования новой строки или прямой команды произошло переполнение строки, то интерпретатор игнорирует введенную строку и выдает сообщение об ошибке 18 — ПЕРЕПОЛНЕНА СТРОКА.

Ссылка оператором MODIFY на несуществующую строку вызывает сообщение об ошибке 5 — НЕСУЩЕСТВУЮЩИЙ НОМЕР СТРОКИ.

Оператор COMMENT служит для введения в программу комментариев в целях удобства чтения программы. Все, что следует после этого оператора до конца командной строки, не выполняется. При вводе программы

в оперативную память комментарии размещаются в ней и, следовательно, сокращают объем памяти, предназначенной для размещения переменных.

Формат оператора:

COMMENT <текст комментария>

Оператор VERIFY служит для оперативного контроля исправности микроЭВМ. Он не требует никаких операндов.

При нормальном прохождении контрольной задачи на устройство вывода поступает сообщение:

ЭВМ ИСПРАВНА

В случае ошибок выдается сообщение:

ЭВМ НЕИСПРАВНА

Причинами могут быть ошибки в ПЗУ интерпретатора, неисправности в аппаратной части микроЭВМ либо сбои.

При отладке программы, используя *знак трассировки ?*, можно следить за ходом выполнения программы и выводить на печать отдельные ее участки. Те участки программы, которые программист заключает в знаки трассировки ?...?, будут печататься при их выполнении. ФОКАЛ печатает каждый следующий за знаком трассировки символ до тех пор, пока не встретится следующий знак ? или пока управление не передастся пользователю.

Пример.

```
*C:  Ф О К А Л
  1.10 SET A=15;SET B=2;SET C=3
  1.20 TYPE ?A+B+C?;QUIT
*GO
A+B+C     20.0000*
```

Символ ? в текстах примечаний в операторах TYPE и ASK не воспринимается ФОКАЛом как знак трассировки.

В ФОКАЛе *управляющими клавишами* являются комбинации клавиш СУ и С (СУ/С) и С1. Нажатие комбинации клавиш СУ/С один раз в диалоговом режиме приводит к сбросу выполнения набранной строки, удалению всех переменных в памяти и печати сообщения об ошибке 0 — ГОТОВНОСТЬ К РАБОТЕ.

Двукратное нажатие комбинации клавиш СУ/С в программном режиме приводит к останову программы и выдаче сообщения об ошибке 0. Переменные и текст программы сохраняются.

Нажатие клавиши C1 приводит к приостанову выполнения программы пользователя. Продолжить выполнение программы можно нажатием любой клавиши.

ФОКАЛ может обнаруживать многие ошибки пользователя. В случае обнаружения ошибки при выполнении программы или прямой команды пользователя на экране дисплея появляется сообщение об ошибке.

Сообщение об ошибке включает код ошибки, номер строки, в которой обнаружена ошибка, и текстовое сообщение о причине ошибки. Например,

```
?03 AT  3.46
НЕПАРНЫЕ СКОБКИ
*
```

Здесь знаки ? и AT — это признак ошибки (ATTENTION — внимание), 03 — код ошибки, 3.46 — номер строки.

После вывода любого сообщения об ошибке устройствами ввода — вывода всегда назначаются клавиатура и экран дисплея. Полный список сообщений об ошибках приведен в Приложении 1.

В целях экономии ячеек оперативной памяти, а также для ускорения ввода программ с клавиатуры можно воспользоваться следующими приемами:

а) В названиях операторов следует использовать только первую букву, после которой должен идти хотя бы один пробел.

б) В списке ввода — вывода допустимо опускать запятую после закрывающей (правой) кавычки, а также после символа !.

Например,
общепринятая запись:

```
C:  Ф О К А Л
 1.10 S A=1;S B=2;S C=3
 1.20 T !,!,"A=",A,!,"B=",B,!,"C=",C,!
*
```

возможная запись:

```
C:  Ф О К А Л
 1.10 S A=1;S B=2;S C=3
 1.20 T !!"A="A,!"B="B,!"C="C,!
*
```

Таким образом экономится семь символов.

в) Пользоваться короткой формой записи оператора присваивания, если это допускает конкретная версия интерпретатора:

```
C:  Ф О К А Л
 1.10 A=1;B=2;C=3;D=-278
*
```

Заметим, что в ФОКАЛе допускается исключать пробел между номером строки и именем оператора:

```
*1.1S X=5;S Y=7
*1.2T X+Y
```

Однако этот прием не приводит к экономии ячеек, так как интерпретатор игнорирует этот пробел при записи строки в память. Наглядность же программы ухудшается.

2.7. Встроенные функции

В настоящей версии языка ФОКАЛ по сравнению с базовой существенно расширен состав встроенных функций (см. Приложение 2). Обращение к функциям осуществляется по имени, первым символом которого обязательно является буква F. За именем функции следует список аргументов, заключенный в скобки. У некоторых функций аргументы отсутствуют. В этих случаях за открывающей скобкой непосредственно следует закрывающая.

Сначала рассмотрим математические функции.

Для вычисления синуса угла предназначена функция FSIN.

Формат:

FSIN (X)

где X — значение аргумента в радианах, который может быть арифметическим выражением.

Пример.

```
*TYPE FSIN(3.14159/4)
     0.7071*
```

Если угол задан в градусах, то для преобразования аргумента используется множитель $\pi/180$.

Пример.

```
*TYPE FSIN(45*3.14159/180)
     0.7071*
```

Для вычисления *косинуса угла* используется функция FCOS.

Формат:

FCOS (X)

где X — значение аргумента в радианах, которое может быть арифметическим выражением.

Примеры:

```
1. *TYPE FCOS(0.5000)
        0.8776*


2. *TYPE FCOS(2*3.14159)
        1.0000*
```

Для вычисления *тангенса угла* предназначена функция FTAN.

Формат:

FTAN (X)

где X — значение аргумента в радианах, который может быть арифметическим выражением.

Примеры:

```
1.  *TYPE FTAN(0.87)
          1.1853*
```

2. Тангенс угла в градусах

```
*TYPE FTAN(45*3.14159/180)
     1.0000*
```

Для вычисления арксинуса используется функция FASIN. Если аргумент по модулю больше 1, выдается сообщение по ошибке 20 — В ФУНКЦИЯХ FASIN и FACOS АРГУМЕНТ ПО МОДУЛЮ >1. Функция FASIN получает вычисленное значение угла в радианах.

Формат:

FASIN (X)

где X — арифметическое выражение.

Пример.

```
*TYPE FASIN(0.5)
     0.5236*
```

Для вычисления *арккосинуса* предназначена функция FACOS. Если аргумент по модулю больше 1, выдается сообщение об ошибке 20 — В ФУНКЦИЯХ FASIN И FACOS АРГУМЕНТ ПО МОДУЛЮ > 1. Функция FACOS получает вычисленное значение угла в радианах.

Формат:

FACOS (X)

где X — арифметическое выражение.

Пример.

```
*TYPE FACOS(0.5)
     1.0472*
```

Для вычисления *арктангенса* используется функция FATAN. Угол, полученный при вычислении, выражен в радианах.

Формат:

FATAN (X)

где X — арифметическое выражение.

Пример.

```
*TYPE FATAN(3.91)
     1.3204*
```

Для вычисления *натурального логарифма* предназначена функция FLOG. Если аргумент меньше или равен нулю, выдается сообщение об ошибке 19 — ЛОГАРИФМ НУЛЯ ИЛИ ОТРИЦАТЕЛЬНОГО ЧИСЛА.

Формат:

FLOG (X)

где X — арифметическое выражение.

Пример.

```
*TYPE FLOG(5.17)
     1.6429*
```

Для вычисления *десятичного логарифма* используется функция FLOG 10. Если аргумент меньше или равен нулю, выдается сообщение об ошибке 19 — ЛОГАРИФМ НУЛЯ ИЛИ ОТРИЦАТЕЛЬНОГО ЧИСЛА.

Формат:

FLOG 10 (X)

где X — арифметическое выражение.

Пример.

```
*TYPE FLOG10(10)
     1.0000*
```

Для вычисления *экспоненты* предназначена показательная функция FEXP.

Формат:

FEXP (X)

где X — арифметическое выражение.

Пример.

```
*TYPE FEXP(5)
   148.4130*
```

Для получения *знаковой части числа* используется функция FSGN. Если аргумент меньше нуля, то функция FSGN получает значение —1; если равен нулю, FSGN получает значение 0; если больше нуля, то функция получает значение +1.

Формат:

FSGN (X)

где X — арифметическое выражение.

Примеры.

```
1.  *TYPE FSGN(6-4)
          1.0000*

2.  *TYPE FSGN(0)
          0.0000*
```

Для получения *целой части выражения,* являющегося аргументом функции, предназначена функция FITR.

Формат:

FITR (X)

где X — арифметическое выражение.

Функция принимает значение целой части аргумента со знаком аргумента.

Примеры.

```
1.  *TYPE FITR(55.27)
         55.0000*

2.  *TYPE FITR(7.86)
          7.0000*

3.  *TYPE FITR(-4.1)
     -    4.0000*
```

Для получения *абсолютной величины* выражения используется функция FABS.

Формат:

FABS (X)

где X — арифметическое выражение.

Пример.

```
1.  *TYPE FABS(-17)
         17.0000*

2.  *TYPE FABS(30.05)
         30.0500*
```

Для вычисления *квадратного корня* из арифметического выражения применяется функция FSQT. Если выражение имеет отрицательное значение, то выдается сообщение об ошибке 17 — КОРЕНЬ КВАДРАТНЫЙ ИЗ ОТРИЦАТЕЛЬНОГО ЧИСЛА.

Формат:

FSQT (X)

где X — арифметическое выражение.

Пример.

```
*TYPE FSQT(625)
     25.0000*
```

Функция FRAN предназначена для *генерации псевдослучайных чисел,* распределенных равномерно в интервале (—1, 1).

Формат:

FRAN ()

Пример.

```
 C:  Ф О К А Л
  1.10 FOR I=1,7;SET X(I)=FRAN();TYPE X(I)
  1.20 TYPE !
*GO
  -0.9214   0.0956 - 0.2698   0.8988 - 0.3956 - 0.3358   0.3759
*GO
 - 0.0783 - 0.5294 - 0.1917   0.9594 - 0.3900   0.6697 - 0.9988
*GO
   0.8746   0.3280 - 0.3692 - 0.1111   0.9821   0.8683   0.5767
*GO
   0.2867   0.1427   0.6965   0.7701   0.3362 - 0.8722 - 0.4228
*
```

При повторном применении FRAN () «перетасовывает» числа и выдает непредсказуемую последовательность случайных чисел.

Функция FCHR предназначена для *приема и (или) печати символов,* соответствующих кодам КОИ-7 (см. Приложение 2).

Основное назначение этой функции состоит в преобразовании символов из кода КОИ-7 в десятичную форму или, наоборот, из десятичной формы в коды КОИ-7.

Формат:

FCHR (список аргументов)

В том случае, когда аргумент функции отрицателен, будет считываться следующий символ (следующие 8 бит) с устройства ввода. Сама функция принимает при этом десятичное значение кода КОИ-7 считываемого символа.

Пример.

```
* TYPE FCHR (—1)
```

После нажатия клавиши <ВК> машина ожидает ввода символа.

Вводим букву А (на экране не отображается), после чего на экран выдается ее десятичное значение

65.0000

Таким образом происходит перевод из КОИ-7 в десятичную форму.

Если аргумент равен нулю или положителен, то целая часть его величины (трактуемой как десятичное число) будет преобразована в соответствующий символ КОИ-7 и передана на устройство вывода. Следом будет выведено значение функции, представляющее собой целую часть значения аргумента.

Пример.

```
*TYPE FCHR(-1)
  65.0000
```

В том случае, когда используется несколько аргументов, функция примет десятичное значение последнего введенного или выведенного символа.

Примеры.

```
1.   *TYPE FCHR(65,66)
     AB   66.0000*

2.   *TYPE FCHR(65,66,-1)
     AB   67.0000*
```

Во втором примере печатаются символы АВ, соответствующие кодам КОИ-7 (десятичные числа 65 и 66). После этого с устройства ввода считывается следующий символ (пусть С), десятичное значение которого (67.0000) и становится значением самой функции.

Аргументом функции FCHR может быть сама функция FCHR, например

```
*SET Z=FCHR(FCHR(-1))
```

В процессе выполнения этой командой строки будет принят один символ с устройства ввода. FCHR (—1) принимает соответствующее десятичное значение (например, если вводимый символ А, то FCHR (—1) принимает значение 65 в десятичном счислении).

Далее, так как происходит выполнение FCHR (65), то будет напечатан символ A (для данного примера).

Переменной *Z* оператором SET будет присвоено значение 65.

Следует помнить, что при положительных аргументах код символа, выводимого на устройство вывода, вычисляется по модулю 128.

Например,

```
*SET X=FCHR(65);TYPE !,X,!
A
  65.0000
*
```

в то время как

```
*SET X=FCHR(193);TYPE !,X,!
A
  193.000
*
```

Функция FCHR может быть использована в обучающей программе для анализа ответов обучаемого.

Например, ответ (A, B или C) на поставленный вопрос может анализироваться так:

```
C:  Ф О К А Л
 1.01 SET REQ=FCHR(-1);X FCHR(REQ);T !
 1.03 IF (REQ-65)3.10,4.10,3.10
 3.10 TYPE "НЕВЕРНО, ПОПРОБУЙТЕ ЕЩЕ РАЗ",!;GO 1.01
 4.10 TYPE "ВЕРНО",!
*GO
B
НЕВЕРНО, ПОПРОБУЙТЕ ЕЩЕ РАЗ
C
НЕВЕРНО, ПОПРОБУЙТЕ ЕЩЕ РАЗ
A
ВЕРНО
*
```

Функция FCLK позволяет при обращении к ней считывать показания часов (таймера), т. е. определять временной интервал в «тиках». Задающим генератором для таймера является 50-герцевая сеть, поэтому один «тик» равен 0.02 с. Таким образом, значение функции FCLK увеличивается на один «тик» (на единицу) через каждую 50-ю долю секунды после того, как был запущен таймер. Часы запускаются после вызова интерпретатора ФОКАЛ поднятием вверх клавиши «Таймер».

Функция FCLK не использует никакого аргумента. Максимальная емкость часов около 46,5 ч.

Формат:

FCLK ()

Примеры.

```
1.   *TYPE FCLK()
      2746.0000*

2.  *T FCLK()/50
       81.0800*
```

Во втором примере время из «тиков» переводят в секунды. Функция FZCLK применяется для обнуления таймера. Функция FZCLK не использует никакого аргумента.

Формат:

FZCLK ()

Пример.

```
*XECUTE FZCLK()
```

Функция FSBR используется для выполнения нумерованной группы строк (или одной строки) как подпрограммы. Подпрограмма представляет собой функцию одной специальной переменной, которая выполняет роль формального параметра.

Формат обращения к функции FSBR:

SET V = FSBR (n, arg)

где V — название переменной, которой присваивается вычисленное значение функции; n — номер группы, реализующей алгоритм программируемой функции (число или переменная); arg — аргумент функции FSBR (число, переменная или арифметическое выражение).

Интерпретатор ФОКАЛа, встретив в тексте программы обращение к FSBR, вычисляет значение аргумента функции и присваивает полученное значение специальной переменной &. Затем осуществляется передача управления на первую строку программы с номером n. Возврат из подпрограммы, реализующей алгоритм программируемой функции, осуществляется по команде RETURN либо по исчерпании строк в группе. После завершения выполнения подпрограммы последнее вычисленное значение становится значением функции.

В список встроенных функций ФОКАЛа не входит функция вычисления котангенса. Ниже приведен пример программы его вычисления с помощью функции FSBR, где аргумент задается в градусах:

```
*C:  Ф О К А Л
  1.10 ASK X;P=X*3.14159/180
  1.20 TYPE !,FSBR(5,R),!
  1.30 QUIT
  5.10 &=1/FTAN(&)
*GO
:45

     1.0000
*
```

Функция FSBR позволяет обращаться сама к себе (рекурсивное обращение). Следует помнить, что в этом случае интенсивно используется стековая область памяти и поэтому возможно ее переполнение при слишком большом числе вложений.

Пример вычисления факториала:

```
 C:  Ф О К А Л
  1.10 ASK N
  1.20 TYPE %9.00,!,FSBR(5,N),!
  1.30 GO 1.10
  5.10 IF (1-&)5.20;RETURN
  5.20 SET &=&*FSBR(5,&-1)
*GO
:10

    3628800
:13


 ?09 AT 5.20
ПЕРЕПОЛНЕН СТЕК
*
```

Функция FCS используется для установки символьного курсора в определенное место экрана.

Формат:

FCS (номер позиции в строке, номер строки)

Первый аргумент «номер позиции в строке» может принимать значения от 0 до 79, при значениях аргумента больше 79 курсор устанавливается в 79-ю позицию.

Второй аргумент «номер строки» может принимать значения от 0 до 23, при значениях аргумента больше 23 курсор устанавливается на 23-й строке.

В качестве параметров аргументов могут использоваться только целые положительные значения переменных или арифметических выражений.

Пример.

```
*XECUTE FCS(50,15);TYPE "A"
```

На 15-й строке экрана в 50-й позиции выведется символ А.

Функция FX используется для выполнения дополнительных действий с периферийными устройствами и для обращения к ячейкам памяти.

Формат функции:

FX (код операции, адрес общей шины, данные)

Первый аргумент «код операции» может принимать значения —1, 0, +1 и определяет тип выполняемой операции:

+1 — чтение слова,

0 — чтение слова логическое с конъюнкцией,

—1 — запись слова по адресу общей шины.

Второй аргумент «адрес общей шины» должен быть или восьмеричным числом, или названием переменной, и должен быть четным.

Третий аргумент «данные» должен быть десятичным числом в интервале от —32768 до +32767 или арифметическим выражением.

При операции «чтение (+1)» выполняется чтение слова по «адресу общей шины». Третий аргумент не нужен. Функция принимает десятичное значение считанного слова.

При операции «запись (—1)» выполняется запись слова («данных») по адресу общей шины. ФОКАЛ обладает способностью самозащиты, т. е. запрещает запись в ячейке оперативной памяти, используемые интерпретатором ФОКАЛа для своей работы (0—1320). При попытке записи в запрещенную область оперативной памяти выдается сообщение об ошибке 13 — ЗАПРЕЩЕННЫЙ АДРЕС ШИНЫ В ФУНКЦИИ FX. При операции запись функция FX принимает десятичное значение записываемого слова.

При операции «чтение слова с конъюнкцией (0)» выполняется конъюнкция слова памяти и данных. Функция получает десятичное значение результата операции конъюнкции.

Если в функции FX указан несуществующий адрес на общей шине, то выдается сообщение об ошибке 21 — НЕСУЩЕСТВУЮЩЕЕ УСТРОЙСТВО.

Примеры.

```
1. *TYPE FX(1,6)
     224.0000*
```

Данные считываются из ячейки с адресом 6.

2.
```
*TYPE FX(-1,20000,10)
     10.0000*
```

В ячейку оперативной памяти с восьмеричным адресом 20 000 запишется число 10 и выведется на печать.

```
*TYPE FX(0,20000,7)
     2.0000*
```

Выполняется операция конъюнкции над содержимым ячейки с адресом 20 000 и числом 7 и результат выводится на печать.

В двоичной системе счисления операция конъюнкции будет выглядеть так:

логическое умножение & 1010 — число 10
0111 — число 7
0010 — число 2

В табл. 2.2 содержится справочная информация для работы с функцией FX. В левой графе таблица записаны десятичные значения чисел для обращения к битам сло-

Таблица 2.2

Десятичное число	Восьмеричное число	Номера битов в слове
—32768	100000	15
16384	40000	14
8192	20000	13
4096	10000	12
2048	4000	11
1024	2000	10
512	1000	9
256	400	8
128	200	7
64	100	6
32	40	5
16	20	4
8	10	3
4	4	2
2	2	1
1	1	0
—32640	100200	15 и 7
192	300	7 и 6
255	377	7, 6, 5, 4, 3, 2, 1, 0
—256	177400	15, 14, 13, 12, 11, 10, 9, 8

ва. В средней графе — восьмеричные значения чисел. В правой графе записаны номера битов слова, соответствующие числам слева.

2.8. Дополнительные операторы

Дополнительными называют операторы, позволяющие расширить возможности работы как со стандартными, так и нестандартными устройствами ввода — вывода.

В первую очередь рассмотрим оператор XECUTE. Он выполняет вычисление функции или арифметического выражения без вывода на печать их результатов. Оператор XECUTE может использоваться как в прямых, так и в косвенных командах в любом месте строки.

Формат оператора:

∗ XECUTE <арифметическое выражение>

Обычно оператор XECUTE используется для выполнения функций.

Примеры.

1.
```
*XECUTE FCHR(65,66)
AB*
```

2.
```
*XECUTE FCHR(FCHR(-1))
```

3. Чтобы датчик случайных чисел заставить вызывать одну и ту же последовательность, что иногда полезно при отладке программ, следует воспользоваться следующим приемом:

```
C:  Ф О К А Л
 1.10 X FRAN(1)
 1.20 FOR I=1,6;TYPE FRAN()
 1.30 TYPE !
*GO
    0.1659 -  0.5618    0.8044 -  0.9171 -  0.1559 -  0.5509
*GO
    0.1659 -  0.5618    0.8044 -  0.9171 -  0.1559 -  0.5509
*
```

4. Оператор XECUTE вместе с функцией FCHR позволяет выполнять команды управления дисплеем (система команд № 1 алфавитно-цифрового дисплея 15ИЭ-00-013):

а) *XECUTE FCHR(7)

б) *XECUTE FCHR(12)

в) *XECUTE FCHR(14)

г) *XECUTE FCHR(15)

Здесь: *а* — звуковой сигнал; *б* — очистка экрана; *в* — переход на русский регистр; *г* — переход на латинский регистр.

Значения других управляющих кодов даны в Приложении 4.

5. Кодирование русских слов осуществляется следующим образом:

```
*XECUTE FCHR(14);ASK Z;TYPE
:ТОЧКА
  218111.00
```

Переменной Z присваивается закодированное значение слова ТОЧКА:

$$\text{ТОЧКА} = \text{Т}\cdot 10^4 + \text{О}\cdot 10^3 + \text{Ч}\cdot 10^2 + \text{К}\cdot 10^1 + \text{А}\cdot 10^0 =$$
$$= 20\cdot 10^4 + 15\cdot 10^3 + 30\cdot 10^2 + 11\cdot 10^1 + 1\cdot 10^0 = 218111$$

где русские буквы рассматриваются как цифры, равные порядковому номеру буквы в списке Приложения 3, а не как имя переменной. Такое свойство ФОКАЛа можно использовать в обучающих программах. Вводимое новое слово кодируется и сравнивается со значением переменной Z. В приведенной ниже программе анализируется правильность ввода слова ТОЧКА:

```
C:  Ф О К А Л
 1.10 XECUTE FCHR(14);ASK "КОНТРОЛЬНОЕ СЛОВО",Z
 1.20 XECUTE FCHR(14);ASK X
 1.30 IF (X-Z)1.40,1.50,1.40
 1.40 TYPE "НЕПРАВИЛЬНО",!;GO 1.20
 1.50 TYPE "ПРАВИЛЬНО",!;QUIT
* GO
КОНТРОЛЬНОЕ СЛОВО:ТОЧКА
ТОЧЬКА
НЕПРАВИЛЬНО
ТОЧКА
ПРАВИЛЬНО
*
```

Так как ФОКАЛ различает только шесть значащих цифр, то при кодировании слова длиной более шести букв может возникнуть неоднозначность в распознавании, например слова «электрификация» и «электрофикация» не различаются.

Существует еще один способ кодирования русских слов. Его удобно использовать в обучающих программах. Контрольное слово вводится непосредственно в оператор IF следующим образом: в латинском регистре сначала вводится цифра «0», затем нужно перейти в нижний регистр и, смотря только на клавиатуру, ввести

соответствующие русские буквы. При этом на экране дисплея будут появляться строчные маленькие буквы латинского алфавита. Хотя это не очень удобно, но с этим приходится мириться. Дело в том, что коды русского слова, введенного с помощью оператора ASK, и слова, написанного строчными латинскими буквами, совпадают. Сказанное поясняется примером:

```
C:  Ф О К А Л
 1.10 TYPE !"НОВЫЙ ПРЕДМЕТ В 9,10 КЛАССАХ?";XECUTE FCHR(14);ASK Z
 1.20 IF (Z-0 informatica)1.30,1.40,1.30
 1.30 TYPE "НЕВЕРНО"!;GO 1.10
 1.40 TYPE "ПРАВИЛЬНО"!;QUIT
*
НОВЫЙ ПРЕДМЕТ В 9,10 КЛАССАХ?: ИНФОРМАТИКА
ПРАВИЛЬНО
*
```

В строке с номером 1.10 формируется вопрос, затем ЭВМ переходит в состояние ожидания. Введенное слово сравнивается с контрольным в следующей строке с помощью условного оператора перехода. В случае неверного ответа вопрос снова повторяется.

6. Кодирование английских слов затруднено из-за запрета использования буквы Е (см. оператор ASK). Если в английском слове нет букв Е, то кодирование можно осуществить с помощью оператора IF. Цифра 0 перед набором латинских букв информирует ФОКАЛ о том, что следующие после нуля буквы будут рассматриваться как цифры, равные порядковому номеру в алфавите (см. Приложение 3).

Пример.

```
*SET Z=0STOP

*C:  Ф О К А Л
  1.10 ASK "НАПИШИТЕ ПО АНГЛИЙСКИ СЛОВО СТОП",X
  1.20 IF (X-0STOP)1.30,1.40,1.30
  1.30 TYPE "НЕПРАВИЛЬНО",!;GO 1.10
  1.40 TYPE "ПРАВИЛЬНО",!;QUIT
*GO
НАПИШИТЕ ПО АНГЛИЙСКИ СЛОВО СТОП:СТОР
НЕПРАВИЛЬНО
НАПИШИТЕ ПО АНГЛИЙСКИ СЛОВО СТОП:STOP
ПРАВИЛЬНО
*
```

7. Используя функцию FCHR и переменные с индексами, можно снять ограничения, имеющие место в предыдущих примерах.

Пример.

```
C:  ● О К А Л
 1.10 I=1;TYPE "НАПЕЧАТАЙТЕ СВОЕ ИМЯ",!;XECUTE FCHR(14)
 1.20 X(I)=FCHR(FCHR(-1))
 1.30 IF (X(I)-13)1.40,1.50,1.40
 1.40 I=I+1;GO 1.20
 1.50 TYPE !,"ЗДРАВСТВУЙ ";XECUTE FCHR(14)
 1.60 FOR K=1,I;XECUTE FCHR(X(K))
 1.70 TYPE !
*GO
НАПЕЧАТАЙТЕ СВОЕ ИМЯ
РОМАН
ЗДРАВСТВУЙ РОМАН
*
```

Строки 1.10—1.40 формируют одномерный массив, где каждой переменной соответствует код определенной буквы. Признаком окончания ввода имени является символ <ВК> (код 13). В строке 1.50 с помощью оператора цикла происходит обратное преобразование кодов в символы.

Оператор OPERATE служит для назначения устройств ввода — вывода символьной информации. Оператор OPERATE может использоваться как в прямых, так и в косвенных командах в любом месте командной строки. Устройство, назначенное пользователем, сохраняется до тех пор, пока не произойдет одно из следующих событий:

назначение другого устройства ввода—вывода оператором OPERATE;

аварийный останов программы по сообщению об ошибке;

прерывание работы программы с клавиатуры дисплея двукратным нажатием комбинации клавиш СУ/С.

В последних двух случаях, а также при запуске интерпретатора устройствами ввода—вывода по умолчанию назначаются соответственно клавиатура и экран дисплея. Никакие другие операторы, а также завершение программы не меняют распределения устройств ввода—вывода.

Формат оператора:

OPERATE <список имен устройств>

Имена устройств ввода—вывода следующие:

K — клавиатура дисплея (устройство ввода);
T — экран дисплея (устройство вывода);
R — считыватель с перфоленты;
P — перфоратор ленточный;
L — построчно-печатающее устройство (устройство вывода).

В каждый момент времени может быть назначено не более одного устройства ввода и одного устройства вывода в произвольной комбинации.

```
OPERATE KL
```

Здесь устройством ввода назначается клавиатура дисплея K, устройством вывода — устройство печати на бумагу L.

Оператор OPERATE используется в паре с операторами ввода—вывода TYPE, ASK, WRITE.

Примеры.

1. Вывод на перфоленту массива данных:

```
*C:  Ф О К А Л
  1.10 OPERATE PUNCH
  1.20 FOR I=1,100;TYPE X(I)
  1.30 TYPE !;COMMENT <ВК>  ДЛЯ ОПЕРАТОРА ASK
  1.40 OPERATE T;QUIT
*
```

Массив данных на перфоленте должен заканчиваться символом <ВК>, по этой причине в строку 1.30 включен оператор TYPE !

2. Вывод массива данных с перфоленты:

```
 C:  Ф О К А Л
  1.10 OPERATE R;FOR I=1,100;ASK X(I)
  1.20 OPERATE K;QUIT
*
```

3. Вывод программы на перфоленту:

```
*OPERATE P;WRITE ALL;OPERATE T
```

4. Ввод программы с перфоленты:

```
*OPERATE R
```

После ввода с перфоленты необходимо двукратное нажатие клавиш СУ/С, в результате на экран дисплея выводится сообщение

```
?00 AT  0.00
ГОТОВНОСТЬ К РАБОТЕ
*
```

Перед вводом новой программы оператором OPERATE READ рекомендуется выполнять оператор ERASE ALL, а перед запуском программы — ERASE.

При вводе текста новой программы оператором OPERATE READ происходит наложение вводимой программы на старую программу в памяти. Таким образом, те строки

старой программы, номера которых не совпадают с номерами строк вводимой программы, в оперативной памяти сохраняются.

5. Вывод значений всех переменных на перфоленту:

```
*OPERATE P;TYPE ¤
```

Ввод в память значений переменных с перфоленты осуществляется следующим образом:

```
*OPERATE R
```

После ввода перфоленты необходимо двукратное нажатие клавиш СУ/С для возврата в диалоговый режим.

Оператор KILL предназначен для установки внешних устройств в исходное состояние: обнуляются буферные регистры и сбрасываются биты разрешения прерывания. При работе оператора KILL выполняется команда ассемблера RESET.

Оператор KILL не требует операндов. При его выполнении прекращается работа всех внешних устройств, работающих в режиме прерывания. Например, после выполнения операнда KILL в программном режиме сбрасывается бит разрешения прерывания от клавиатуры в регистре команд и состояний клавиатуры, поэтому прервать выполнение программы становится невозможным до ее окончания.

Оператор KILL может использоваться как в прямых, так и в косвенных командах в любом месте командной строки.

Ниже приведен пример, показывающий, как с помощью операторов KILL и XECUTE можно защитить фрагмент работающей программы от прерывания до момента окончания его работы. Этот прием можно использовать в обучающих программах, когда нежелательно вмешательство пользователя во время работы определенных частей программы.

```
C:  Ф О К А Л
 1.10 TYPE %4.00;C    УСТАНОВИТЬ ФОРМАТ ВЫВОДА
 1.20 KILL;C ЗАПРЕТИТЬ ПРЕРЫВАНИЕ ОТ КЛАВИАТУРЫ
 1.25 C РАСПЕЧАТАТЬ ЧИСЛА ОТ 1 ДО 200
 1.30 FOR I=1,200;T I
 1.40 XECUTE FX(1,177562);C  ХОЛОСТОЕ СЧИТЫВАНИЕ БУФЕРА КЛАВИАТУРЫ
 1.50 XECUTE FX(-1,177560,64);C  РАЗРЕШИТЬ ПРЕРЫВАНИЕ ОТ КЛАВИАТУРЫ
 1.55 C РАСПЕЧАТАТЬ ЧИСЛА ОТ 201 ДО 400
 1.60 T !!!;FOR I=201,400;T I
*
```

В процессе работы программы на печать выводятся целые числа от 1 до 200. Прервать вывод чисел невоз-

можно до тех пор, пока не выполнится оператор в строке 1.50, разрешающий прерывание. Оператор в строке 1.40 предотвращает вывод на печать символа, введенного с клавиатуры во время попытки останова работающей программы.

2.9. Графические средства

При наличии в составе ДВК контроллера графических изображений (КГД) появляется возможность построения на экране дисплея различных изображений. Рабочее поле имеет форму прямоугольника и состоит из 286×400 точек. Каждой из 114 400 точек соответствует определенный адрес бита экранной памяти КГД. Светлая точка представляется значением бита «1», а темная точка — значением бита «0». Для управления построением изображений в ФОКАЛе используются графические функции. Обращение к ним осуществляется по имени, первым символом которого является буква F, так же как и к остальным встроенным функциям языка ФОКАЛ. За именем функции следует список аргументов, заключенных в скобки; необязательные параметры могут опускаться.

Язык ФОКАЛ включается в себя 11 встроенных графических функций, которые имеют следующие имена и назначение:

FI — начальная установка экрана;
FM— установка режима вывода;
FP — вывод точки в абсолютных координатах;
FA — установка графического курсора в абсолютных координатах;
FD — вывод точки в относительных координатах;
FC — установка графического курсора в относительных координатах;
FV — вывод отрезка (вектора) в абсолютных координатах;
FL — вывод отрезка (вектора) в относительных координатах;
FR — вывод окружности;
FS — вывод прямоугольника;
FF — закрашивание замкнутого контура (в поздних версиях ФОКАЛа).

Рассмотрим действие каждой функции в перечисленном порядке.

При начальном включении ДВК он автоматически

входит в символьный режим, при котором никакие построения изображений невозможны.

Функция начальной установки экрана FI служит для включения графического режима.

Формат функции:

FI (код установки)

Код установки может иметь значения 0, 1, —1. Все графические функции выполняются с помощью оператора XECUTE.

При выполнении функции

```
*XECUTE FI(0)
```

устанавливается темный фон. Все прежние графические изображения стираются. Символьная информация остается без изменений, и если она не нужна, то ее можно удалить с помощью клавиши СБР (сброс). Заметим, что только программа может быть в любой момент восстановлена с помощью оператора WRITE (имеются в виду пронумерованные программные строки). Фактически функция начальной установки FI(0) очищает все ячейки экранной памяти КГД (записывает нули).

При выполнении функции

```
*XECUTE FI(1)
```

устанавливается светлый фон, т. е. все ячейки экранной памяти КГД заполняются единицами. Прежние графические изображения стираются.

Выполнение функции

```
*XECUTE FI(-1)
```

приводят к тому, что картинка на экране «инвертируется», т. е. все светлые места становятся темными и наоборот (негативное изображение). Эта операция заключается в замене в экранной памяти всех нулей на единицы и наоборот.

Функция FM используется для установки режима вывода изображений. Существует возможность установки трех режимов вывода: вывод светлых точек, вывод темных точек и инверсия выводимых точек по отношению к фону.

Формат функции:

FM (код режима)

Код режима может принять одно из значений 0, 1, —1.

При начальном включении ДВК код режима автоматически устанавливается в единицу.

После выполнения функции

```
*XECUTE FM(0)
```

изображения будут выводиться темными точками. Чтобы они были видны на экране, фон должен быть светлым.

Выполнение функции

```
*XECUTE FM(1)
```

восстанавливает стандартный режим вывода изображений светлыми точками (фон должен быть темным, чтобы выводимые точки были видны).

После выполнения функции

```
*XECUTE FM(-1)
```

устанавливается инверсный режим вывода изображений по отношению к фону.

Функция FP выводит в любое место экрана светлую или темную точку в зависимости от установленного режима. Аргументами функции являются координаты точки по осям X и Y соответственно.

Формат функции:

FP (X, Y),

где X и Y — координаты выводимой точки.

Пример 1. Вывести светлую точку в центр экрана:

```
*C:  Ф О К А Л
  1.10 X FI(0);X FM(1);X FP(200,142)
*
```

Пример 2. Построить горизонтальную линию, проходящую через центр экрана:

```
 C:  Ф О К А Л
  1.10 X FI(0);X FM(1);C НАЧАЛЬНАЯ УСТАНОВКА ЭКРАНА И РЕЖИМА
  1.20 F I =0,399;X FP(I,142);C ПОСТРОЕНИЕ ЛИНИИ
*
```

Пример 3. Вывести на экран дисплея синусоиду с параметрами: амплитуда — 100 (единиц); период — 200; постоянная составляющая — 140.

```
C:  Ф О К А Л
 1.10 X FI(0);X FM(1)
 1.20 A=100;T=200;A0=140;PI=3.14159
 1.30 F X=0,399;X FP(X,A*FSIN(2*PI*X/T)+A0)
*
```

В результате выполнения этой программы на экран дисплея будут выведены два периода синусоиды.

Для графического режима существует понятие *графического курсора.* В отличие от символьного курсора (мигающая марка) графический курсор невидим. Текущему положению графического курсора соответствует конкретное содержимое регистра адреса КГД. После включения ДВК этот регистр автоматически обнуляется, при этом графический курсор устанавливается в левый нижний угол экрана (точка с координатами X = 0, Y = = 0). Рассмотрим две следующие функции, которые помогут наглядно уяснить суть понятия графического курсора.

Функция FA используется для установки графического курсора в нужное место экрана. Выполнение этой функции не приводит ни к каким видимым действиям на экране.

Формат функции:

FA (X, Y)

где X, Y — координаты графического курсора.

Функция FD служит для вывода точки с относительными координатами.

Формат функции:

FD (Δx, Δy)

где Δx, Δy — смещение координат выводимой точки относительно текущих координат графического курсора. Смещения могут быть как положительными, так и отрицательными. С помощью этой функции легко определить на экране место, где находится графический курсор:

```
C:  Ф О К А Л
 1.10 X FI(0);C НАЧАЛЬНАЯ УСТАНОВКА ЭКРАНА
 1.20 X FM(1);C УСТАНОВКА РЕЖИМА ВЫВОДА
 1.30 X FP(200,142);C ВЫВОД ТОЧКИ
*
```

Пример 1. Построить горизонтальную линию, проходящую через центр экрана:

```
C:  Ф О К А Л
 1.10 X FI(0);X FM(1)
 1.20 X FA(0,142);C  УСТАНОВКА КУРСОРА В ИСХОДНОЕ СОСТОЯНИЕ
 1.30 F I=0,399;X FD(1,0);C  ПОСТРОЕНИЕ ЛИНИИ
*
```

Пример 2. Построить биссектрису левого верхнего угла экрана:

```
C:  ● О К А Л
 1.10 X FI(0);X FM(1)
 1.20 X FA(0,285);C  УСТАНОВКА КУРСОРА
 1.30 F I=0,286;X FD(1,-1);C  ПОСТРОЕНИЕ БИССЕКТРИСЫ
*
```

Функция FC служит для смещения графического курсора относительно его текущего положения.

Формат функции:

FC (Δx, Δy)

где Δx, Δy — смещение относительно текущих координат курсора.

Примеры использования этой функции будут приведены позднее.

Для построения широкого круга графических изображений можно обойтись перечисленными функциями. Однако следующие далее функции расширяют возможности пользователя как с точки зрения упрощения построения графических изображений, так и с точки зрения увеличения скорости их вывода на экран дисплея.

Функция FV используется для вывода отрезка (вектора) в абсолютных координатах.

Формат функции:

FV ($X_{кон}$, $Y_{кон}$ [, $X_{нач}$, $Y_{нач}$])

где $X_{кон}$, $Y_{кон}$ — конечные координаты вектора; $X_{нач}$, $Y_{нач}$ — начальные координаты вектора.

Начальные координаты вектора можно опустить, если вектор начинается с того места, где находится графический курсор.

Пример 1. Провести диагонали экрана:

```
C:  ● О К А Л
 1.10 X FI(0);X FM(1)
 1.20 X FV(399,0,0,285);C  ПОСТРОЕНИЕ ПЕРВОЙ ДИАГОНАЛИ
 1.30 X FV(399,285,0,0);C  ПОСТРОЕНИЕ ВТОРОЙ ДИАГОНАЛИ
*
```

Пример 2. Построить график функции $\cos x$ с помощью отрезков:

```
C:  ● О К А Л
 1.10 X FI(0);X FM(1)
 1.20 X FA(0,240);C  УСТАНОВКА КУРСОРА
 1.30 A=100;T=200;A0=140;PI=3.14159
 1.40 F X=0,3,399;X FV(X,A*FCOS(2*PI*X/T)+A0)
*
```

Длина отрезков, с помощью которых аппроксимируется функция, зависит от величины шага в операторе цикла. Попробуйте его увеличить, уменьшить; сравните изображения на экране.

Функция FL также служит для вывода отрезка (вектора), но его конечная точка задается в относительных координатах.

Формат функции:

FL (Δx, Δy [, $X_{нач}$, $Y_{нач}$])

где Δx, Δy — смещение координат конца вектора соответственно по осям *x* и *y* относительно начальной точки; $X_{нач}$ и $Y_{нач}$ — координаты начала вектора.

Параметры, стоящие в квадратных скобках, могут быть опущены, тогда за начало вектора принимаются координаты графического курсора.

Заметим, что приращения Δx и Δy могут быть как положительными, так и отрицательными. Графический курсор при значениях координаты X больше 399 или меньше нуля «уходит» за пределы экрана соответственно вправо или влево. При значениях координаты Y больше 285 графический курсор «уходит» вверх за экран, а меньше нуля — вниз за экран.

Пример. Вывести на экран дисплея периодическую последовательность прямоугольных импульсов с параметрами: период — 30 (единиц); длительность — 5; амплитуда — 30; начало координат — X = 20, Y = 142.

```
C:  ● O K A Л
 1.10 X FI(0);X FM(1)
 1.20 X FA(20,142);C  НАЧАЛЬНАЯ УСТАНОВКА КУРСОРА
 1.30 F I=1,12;X FL(25,0);X FL(0,30);X FL(5,0);X FL(0,-30)
*
```

В строке 1.30 организован цикл, который выводит на экран 12 прямоугольных импульсов с заданными параметрами.

Функция FR предназначена для вывода на экран окружности.

Формат функции:

FR (R[, $X_ц$, $Y_ц$])

где R — радиус выводимой окружности; $X_ц$, $Y_ц$ — координаты центра выводимой окружности.

Параметры $X_ц$, $Y_ц$ могут быть опущены, тогда центр выводимой окружности разместится в точке нахождения

графического курсора. После окончания вывода окружности графический курсор остается в ее центре.

Пример 1. Построить в центре экрана окружность радиусом 50 единиц:

```
C:  Ф О К А Л
 1.10 X FI(0);X FM(1)
 1.20 X FR(50,200,142);C  ВЫВОД ОКРУЖНОСТИ
*
```

Пример 2. Построить в центре экрана круг радиусом 50 единиц:

```
C:  Ф О К А Л
 1.10 X FI(0);X FM(1)
 1.20 X FA(200,142);C  УСТАНОВИТЬ КУРСОР В ЦЕНТР ЭКРАНА
 1.30 F I=1,50;X FR(I);C  ВЫВОД ОКРУЖНОСТЕЙ
*
```

В строке 1.30 выводятся окружности с постепенно увеличивающимся радиусом от 1 до 50.

Можно выполнить инверсию изображения:

X FI (—1).

В результате выполнения этой функции фон станет светлым, а круг — темным. Узор из ряда светлых точек, оставшихся в круге, — результат погрешности аппроксимации при построении окружностей.

Функция FS служит для построения прямоугольника.

Формат функции:

FS (L, H [X, Y]),

где L — длина прямоугольника; H — высота прямоугольника; X, Y — координаты начала вывода прямоугольника (левый нижний угол).

Параметры, указанные в квадратных скобках, могут быть опущены. Тогда прямоугольник будет строиться из точки нахождения графического курсора.

После вывода прямоугольника графический курсор остается в противоположном углу от начала точки построения прямоугольника. Аргументы L и H могут иметь отрицательные значения, при этом прямоугольник строится влево и вниз от начальной точки.

Пример 1. Построить прямоугольник 100×50 из точки X = 150, Y = 120:

```
C:  Ф О К А Л
 1.10 X FI(0);X FM(1)
 1.20 X FS(100,50,150,120);C  ПОСТРОИТЬ ПРЯМОУГОЛЬНИК
*
```

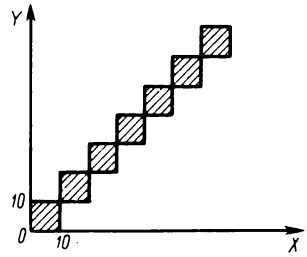


Рис. 2.1. Цепочка квадратов

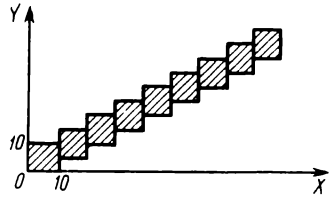


Рис. 2.2. Цепочка смещенных квадратов

Пример 2. Построить цепочку квадратов из точки X = 0, Y = 0 так, как показано на рис. 2.1:

```
C:  Ф О К А Л
 1.10 X FI(0);X FM(1)
 1.20 X FA(0,0);C  ПОСТАВИТЬ КУРСОР В ИСХОДНУЮ ТОЧКУ
 1.30 F I=1,25;X FS(10,10);C  РАЗМЕР КВАДРАТА 10 X 10
*
```

Пример 3. Изменить предыдущий пример так, чтобы каждый последующий квадрат начинался с середины боковой стороны предыдущего (рис. 2.2):

```
C:  Ф О К А Л
 1.10 X FI(0);X FM(1)
 1.20 X FA(0,0);C ПОСТАВИТЬ КУРСОР В ИСХОДНУЮ ТОЧКУ
 1.30 F I=1,35;X FS(10,10);X FC(0,-5)
*
```

По окончании построения каждого квадрата функция FC сдвигает графический курсор на пять единиц вниз от текущего положения.

Функция FF служит для «закрашивания» замкнутых контуров.

Формат функции:

FF (X, Y)

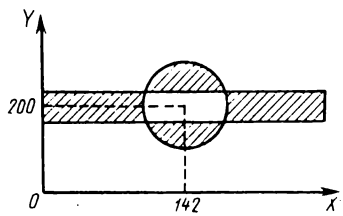


Рис. 2.3. Геометрическая фигура

где X, Y — координаты любой точки внутри контура. Эта функция имеется лишь в поздней версии ФОКАЛа.

Пример 1. Построить круг радиусом 40 единиц в центре экрана:

```
C:  Ф О К А Л
 1.10 X FI(0);X FM(1)
 1.20 X FR(50,200,142);C ПОСТРОИТЬ ОКРУЖНОСТЬ
 1.30 X FF(180,120);C ЗАКРАСИТЬ КРУГ
*
```

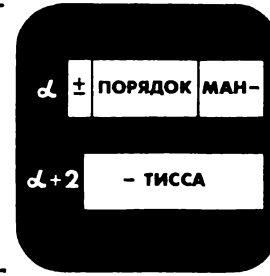
Если точку в функции FF поставить вне окружности, то произойдет закрашивание фона, внутри которого останется темный круг.

Пример 2. Построить фигуру, показанную на рис. 2.3:

```
C:  Ф О К А Л
 1.10 X FI(0);X FM(1)
 1.20 X FR(50,200,142);X FF(200,142);C  ПОСТРОИТЬ КРУГ
 1.30 X FM(-1);C  УСТАНОВИТЬ ИНВЕРСНЫЙ РЕЖИМ ВЫВОДА
 1.40 X FS(399,40,0,122);C  ПОСТРОИТЬ ПРЯМОУГОЛЬНИК
*
```

глава 3

Основы внутренней организации интерпретатора Фокала



Интерпретатор языка ФОКАЛ представляет собой набор подпрограмм, таблиц, флагов, указателей, переменных и текстов сообщений, обеспечивающий в целом простой диалог и все необходимые средства для загрузки, отладки и выполнения программ пользователя. На рис. 3.1 показано доступное для программиста адресное пространство памяти микроЭВМ «Электроника МС 1200.01». Оно состоит из семи банков (с нулевого по шестой), каждый из которых имеет объем 4096 16-разрядных слов. Интерпретатор ФОКАЛа изготовлен в виде микросхемы БИС ПЗУ и занимает в адресном пространстве место шестого банка (соответствующая этому банку область оперативной памяти отключается с помощью специального движкового переключателя, расположенного на плате микроЭВМ). Таким образом, для интерпретатора ФОКАЛа остаются доступными шесть банков оперативной памяти общим объемом 48 К байт, начиная с байта с адресом 0 по байт с адресом 137777 включительно (адреса даны в восьмеричной системе счисления).

Часть ячеек оперативной памяти с нулевого адреса до 1346-го используется для внутренних нужд интерпретатора (верхняя заштрихованная область на рис. 3.1). В них хранятся векторы прерываний, буфер для временного размещения вводимой строки, рабочие ячейки, флаги, адреса системных внешних устройств. Здесь же расположен стек интерпретатора, «дно» которого находится по адресу 1100. Это область защищена программными средствами от вмешательства пользователя. При попытке записи в любую ячейку указанной области выдается предупреждающее сообщение: «Запрещенный адрес шины в функции FX». Сам интерпретатор не нуждается в защите благодаря его расположению в ПЗУ.

3.1. Размещение программных строк в оперативной памяти

Программные строки на ФОКАЛе хранятся в оперативной памяти в виде одностороннего связанного списка [12]. Буфер текста располагается с адреса 1346, и по мере ввода программных строк граница его смещается в направлении, указанном на рис. 3.1. Оставшаяся свободной область ячеек оперативной памяти выделяется

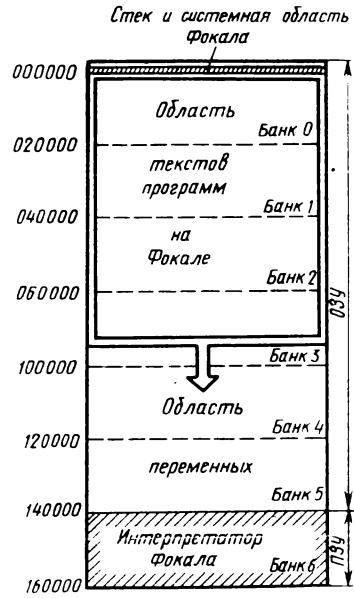


Рис. 3.1. Распределение адресного пространства памяти микроЭВМ «Электроника МС 1200.01»

для размещения переменных. При переполнении буфера текста выдается соответствующее диагностическое сообщение. Впрочем, такая ситуация наступает очень редко, так как буфер текста способен вместить более 48 тыс. символов. Когда буфер текста заполняется полностью, область для переменных сокращается до четырех ячеек памяти (подробнее о размещении переменных см. в § 3.2).

Формат программной строки языка ФОКАЛ реализован следующим образом. В первых двух байтах располагается смещение, которое показывает относительное положение начала следующей строки со старшим номером. В следующих двух байтах находится закодированный номер строки. Далее следуют байты, в которых находятся соответствующие символы строки. Заканчивается любая строка символом ⟨ВК⟩ (возврат каретки, внутренний код 216). Для удобства распознавания интерпретатором ФОКАЛа определенной группы символов (см. Приложение 5) они из кода ASCII переводятся во внутренний код. При выводе этих символов на печать выполняется обратное преобразование.

Принятый способ размещения программных строк ФОКАЛа в памяти предполагает начало любой строки с границы слова, т. е. с четного адреса байта. Поэтому если общее число символов в строке оказывается нечетным, то интерпретатор автоматически к концу строки добавит один нулевой байт (точнее, сдвинет текущий указатель на четный адрес).

Пусть, например, после начального запуска интерпретатора вводится программная строка:

```
1.01 S A=1 <ВК>
```

В буфере текста она займет пять слов. Переводя микроЭВМ в пультовый режим (для этого соответствующая клавиша на передней панели переключается из положения «Прогр.» в положение «Пульт»), можно «просмотреть» на экране дисплея восьмеричное содержимое ячеек памяти начиная с адреса 001346 (начало буфера текста):

Адрес	Содержимое	Пояснение
001346	176430	Смещение
001350	000402	Номер строки
001352	100123	␣ S
001354	107501	= A
001356	107061	<ВК> 1

В ячейках 001352—001356 в старших и младших байтах упакованы соответствующие символы. Для наглядного представления можно «вручную» расписать восмеричный адрес и содержимое каждого байта:

Адрес	Содержимое	Пояснение
001352	123	S
001353	200	␣
001354	101	A
001355	217	=
001356	061	1
001357	216	ВК

Рассмотрим, каким образом интерпретатор использует смещение. Когда ЭВМ считывает содержимое ячейки памяти по адресу 001346, счетчик команд приобретает значение 001350 (инкремент 2). Для вычисления адреса начала следующей программной строки интерпретатор складывает содержимое счетчика команд и смещение в текущей строке:

```
  001350
+ 176430
------
1←000000
```

Нулевое значение суммы (единица переноса, вышедшая за пределы разрядной сетки, в расчет не принимается) является признаком отсутствия других программных строк в буфере текста.

Теперь введем следующую строку:

```
1.03 S B=2 <ВК>
```

Содержимое буфера текста увеличится еще на пять слов. Вновь просмотрим содержимое ячеек памяти:

Адрес	Содержимое	Пояснение
001346	000010	Смещение
001350	000402	Номер строки 1.01
001352	100123	␣ S
001354	107501	= A
001356	107061	<ВК> 1
001360	176416	Смещение
001362	000407	Номер строки 1.03
001364	100123	␣ S
001366	107502	= B
001370	107062	<ВК> 2

Обратите внимание на то, что смещение в первой строке стало иным. Вычислим адрес начала следующей строки:

```
  001350
+ 000010
  001360
```

Это начало строки с номером 1.03. Сложив коды:

```
  001362
+ 176416
← 000000
```

убеждаемся, что дальше других строк в буфере текста нет.

Итак, если программные строки вводятся с возрастающими номерами (шаг безразличен), то смещения в каждой строке указывают адрес следующей строки в одном направлении (рис. 3.2). Ясно, что поиск строки с определенным номером должен всегда начинаться с начала списка. Далее, используя смещения переходят от одной строки к другой, каждый раз сравнивая заданный номер строки с текущим. Положение (адрес) номера текущей строки всегда известно — он располагается вслед за смещением.

В качестве отправной точки в ФОКАЛе используется

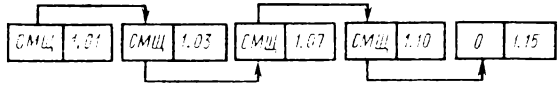


Рис. 3.2. Схема ссылок в строках, введенных в порядке возрастания их номеров

так называемая нулевая строка. Нулевая строка генерируется при начальном запуске интерпретатора, содержимое ее и место в оперативной памяти строго фиксированы. Это единственная строка, имеющая нулевой номер. Если буфер программных строк пуст, то смещение в нулевой строке таково, что сумма его значения и адреса смещения, увеличенного на два, равна нулю. В общем случае смещение в нулевой строке указывает на начало строки в буфере текста с наименьшим номером. Содержимым нулевой строки является единственный оператор — оператор комментария:

```
C:  ФОКАЛ <ВК>
```

Теперь рассмотрим случай, когда строки вводятся не в порядке возрастания их номеров:

```
1.01 S A=1 <ВК>

1.03 S C=3 <ВК>

1.02 S B=2 <ВК>

1.04 S D=4 <ВК>

*
```

После ввода указанных программных строк проанализируем содержимое ячеек буфера текста:

Адрес	Содержимое	Адрес	Содержимое
001346	000022	001372	177764
001350	000402	001374	000405
001352	100123	001376	100123
001354	107501	001400	107502
001356	107061	001402	107062
001360	000022	001404	176372
001362	000407	001406	000412
001364	100123	001410	100123
001366	107503	001412	107504
001370	107063	001414	107064

Смещение в нулевой строке находится по адресу 001320. Содержимое ячейки памяти по этому адресу: 001320/000024.

Адрес следующей строки:

$$001320 + 2 + 24 = 001346,$$

т. е. это строка с номером 1.01 и по адресу 001346 находится ее собственное смещение. Выполним переход на следующую строку:

$$001346 + 2 + 22 = 001372.$$

По указанному адресу находится смещение строки с номером 1.02.

Вычислим очередной переход:

```
  001372
+      2
  177764
--------
  001360
```

Получен адрес смещения, принадлежащий строке 1.03. И наконец, переход на последнюю, четвертую, строку:

$$001360 + 2 + 22 = 001404.$$

Убедимся, что дальше строк нет:

$$001404 + 2 + 176\,372 = 000000.$$

Итак, приведенный пример наглядно иллюстрирует реализованный в ФОКАЛе механизм размещения программных строк в буфере текста и принцип их связи между собой.

Отметим, что программные строки независимо от их номера располагаются в буфере текста в том же порядке, в котором они вводятся в ЭВМ (рис. 3.3). Однако смещения (они вычисляются интерпретатором при вводе каждой новой строки) обеспечивают последовательный

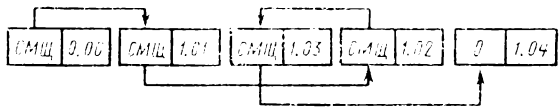


Рис. 3.3. Схема ссылок в строках, введенных не в порядке возрастания их номеров

переход от одной строки к другой в порядке возрастания их номеров.

В этом можно убедиться, распечатав на экране дисплея содержимое буфера текста с помощью прямой команды:

```
* W <ВК>
C:   Ф О К А Л
  1.01  S  A=1
  1.02  S  B=2
  1.03  S  C=3
  1.04  S  D=4
*
```

Остается рассмотреть случай, когда номер вновь вводимой строки совпадает с одним из номеров ранее введенной строки. Пусть последовательно в память ЭВМ вводятся следующие программные строки:

```
1.01 S A=1 <ВК>
1.02 S D=4 <ВК>
1.03 S C=3 <ВК>
*
```

Содержимое ячеек памяти в буфере текста, соответствующее введенным программным строкам, приведено ниже:

1346/ 000010	1360/ 000010	1372/ 176404
1350/ 000402	1362/ 000405	1374/ 000407
1352/ 100123	1364/ 100123	1376/ 100123
1354/ 107501	1366/ 107504	1400/ 107503
1356/ 107061	1370/ 107064	1402/ 107063

Теперь введем новую строку, номер которой совпадает с номером второй строки:

```
1.02 S B=2 <ВК>
*
```

и проанализируем изменения, происшедшие в буфере текста:

1346/ 000022	1360/ 176416	1372/ 177764
1350/ 000402	1362/ 000407	1374/ 000405
1352/ 100123	1364/ 100123	1376/ 100123
1354/ 107501	1366/ 107503	1400/ 107502
1356/ 107061	1370/ 107063	1402/ 107062

Во-первых, видим, что из памяти удалена старая строка с номером 1.02, а новая строка с тем же номером добавлена в конце. Во-вторых, произошла коррекция смещений в соответствующих строках (адреса 1346, 1360, 1372).

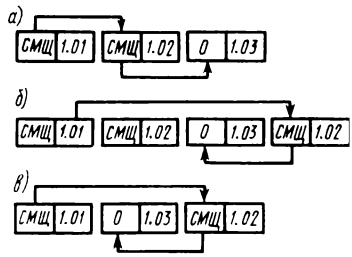


Рис. 3.4. Схема ссылок в строках при введении новой строки с совпадающим номером:
а — первая фаза; *б* — вторая фаза; *в* — третья фаза

Работу, выполняемую интерпретатором, можно наглядно представить следующим образом (рис. 3.4). В первой фазе (рис. 3.4, *а*) вводятся в буфер текста строки с номерами 1.01, 1.02, 1.03. Стрелками условно показаны ссылки, определяемые соответствующими смещениями в каждой строке. Во второй фазе (рис. 3.4, *б*) добавлена новая строка с номером 1.02. Старая строка с тем же номером осталась без ссылок. В третьей фазе (рис. 3.4, *в*) интерпретатор удаляет старую строку и осуществляет подвижку влево строк 1.03 и вновь введенной строки 1.02 так, чтобы исчезло пустое место.

Таким образом, интерпретатор «берет на себя заботу» по упорядочению программных строк в порядке возрастания их номеров и удалению старых строк с повторяющимися номерами.

Программные строки в ФОКАЛе разбиты на группы. Всего может быть 99 групп (от 00 до 99), и каждая группа может содержать от одной до 99 строк. Следовательно, общее возможное количество строк в ФОКАЛе $99 * 99 = 9801$ мало ограничивает свободу действий опытного программиста, не говоря уже о начальном этапе обучения.

Группы в языке ФОКАЛ играют большую роль. Фактически они представляют возможность разбить программу на модули (подпрограммы). Каждая группа в любой момент может быть выведена отдельно на печать (экран) с помощью прямой команды

```
* WRITE n
```

где n — номер соответствующей группы, или может быть полностью удалена с помощью оператора

```
* ERASE n
```

Обращение к группе осуществляется с помощью оператора DO n, причем управление передается строке с наименьшим номером в данной группе. Выход из группы, если в ней нет других операторов перехода, осуществляется либо по исчерпании строк в группе (выход по умолчанию), либо по оператору RETURN. При выходе из подпрограммы управление возвращается оператору, следующему после оператора обращения DO n.

Принятый в языке ФОКАЛ способ обращения к подпрограммам отличается простотой и лаконичной формой записи, хотя следует признать его ограниченные возможности по сравнению со средствами, предоставляемыми более мощными языками, программирования, такими, как ФОРТРАН, ПАСКАЛЬ, ПЛ/1 и др.

На начальном этапе изучения основ программирования пользователю приходится довольно часто вносить исправления в программу и многократно запускать ее на счет до тех пор, пока не будут полностью устранены все ошибки. Поэтому удобные средства отладки программ играют существенную роль.

Включенный в состав интерпретатора ФОКАЛа экранный редактор строк при небольшом объеме с достаточной эффективностью решает эту проблему. Функции его просты и легко прослеживаются на экране дисплея: сдвиг символьного курсора вдоль строки влево и вправо: размыкание или смыкание строки от места положения курсора; вставка и удаление символов; возврат к началу строки, табуляция; удаление остатка строки.

Наряду с перечисленными средствами отладки в ФОКАЛе имеется возможность распечатки фрагментов программы во время ее исполнения, т. е. трассировка. Для ее выполнения необходимый участок программы ограничивается слева и справа знаком трассировки «?».

Практика показала, что простые и удобные средства отладки программ явились одной из причин, побудивших отдать предпочтение ФОКАЛу как языку начального обучения.

3.2. Размещение переменных в оперативной памяти

Как видно из рис. 3.1, для размещения переменных выделяется область памяти от конца буфера текста программ до начала размещения интерпретатора. Таким образом, размер области переменных динамически изменя-

ется: чем больше места занимает текст программы, тем меньше места остается для переменных.

Каждая переменная занимает в памяти четыре последовательно расположенных ячейки. В первой ячейке находится имя переменной — одна или две буквы; во второй — индекс (либо отсутствует, либо один или два в зависимости от размерности переменной); в третьей и четвертой ячейках — значение переменной в формате с плавающей точкой.

Переменные распределяются в отведенной области по методу рассеянной памяти (метод хеширования) [12]. Суть его состоит в том, что переменные хранятся в памяти не в последовательных ячейках, а рассеиваются случайным образом по всей выделенной для них области. Каждая новая переменная помещается таким образом, что ее присутствие или отсутствие может быть позднее установлено без необходимости поиска по всей области. С этой целью для размещения переменной вычисляется ее смешанный код (хеш-адрес). К функции рассеивания предъявляются противоречивые требования: с одной стороны, она должна по случайному закону равномерно рассеивать переменные по выделенной области памяти, а с другой стороны, быть достаточно простой, чтобы избежать излишних временных затрат на ее вычисление.

В интерпретаторе языка ФОКАЛ вычисление смешанного кода реализовано по следующему алгоритму:

если x1 — первая буква имени переменной; x2 — вторая буква имени переменной; y1 — первый индекс; y2 — второй индекс, то вычисляется выражение

$$x2 \cdot 2^8 + x1 + y2 \cdot 2^8 + y1;$$

в полученном 16-разрядном двоичном слове меняются местами младший и старший байты (используется команда ассемблера SWAB);

окончательно полученное выражение берется по модулю размера области памяти, отведенного для переменных, и по модулю 10_8, так как каждая переменная занимает четыре слова.

При начальном запуске интерпретатора происходит автоматическое обнуление ячеек оперативной памяти от начала буфера текста (адрес 001346) до начала интерпретатора ФОКАЛа (адрес 140000). Затем, по мере ввода программных строк, область переменных также постоянно очищается. Переменные появляются в отведенной области только после запуска программы на счет.

Если после запуска программы и выхода в диалог снова вводятся новые программные строки, то они «заходят» в область переменных. Именно поэтому интерпретатор устанавливает новую область для переменных и очищает ее (записывает нули).

Во время работы программы встретившаяся переменная сначала записывается в стек и вычисляется ее смешанный код (хеш-адрес). Если по полученному адресу ячейка свободна (содержит нуль), то происходит запись переменной (имени, индексов, если они есть, и значения); если ячейка занята, то происходит сдвиг указания на четыре слова и снова проверяется, свободна ли ячейка. Таким образом, коллизии разрешаются методом линейного поиска свободного места от вычисленного хеш-адреса. Если свободного места нет, то жертвуют именем переменной с нулевой мантиссой, при этом если в программе снова встретится переменная с тем же именем, то интерпретатору придется подыскивать для нее новое место. При исчерпании свободных мест в буфере переменных выдается диагностическое сообщение: «Нет места для переменных в памяти».

В случае большого количества переменных в программе или слишком малого места для них из-за большой длины текста программы вследствие частого возникновения коллизий (поиска свободных мест для размещения переменных) может произойти значительное снижение скорости выполнения программы. Это обстоятельство нужно учитывать на практике. В качестве одной из мер может быть рекомендовано использование в программе оператора вычеркивания переменных ERASE, если это допустимо по смыслу самой программы.

Принятый в ФОКАЛе динамический способ распределения переменных в памяти облегчает программирование, что является важным фактором для языка, предназначенного для начального обучения. Действительно, по мере появления в тексте программы каждой новой переменной, интерпретатор подыскивает для нее место в оперативной памяти. Это относится как к простым переменным, так и к переменным с индексами. Поэтому в ФОКАЛе нет необходимости описания массивов в начале программы (задания размерности и максимальных значений индексов). Например, для переменных X(1,1) и X(10, 15) будут выделены два места в памяти, тогда как в БЕЙСИКе или ФОРТРАНе понадобилось бы описание массива DIMENSION (10, 15) и при этом

в памяти было бы зарезервировано 150 мест. Другими словами, в ФОКАЛе место в памяти выделяется только для конкретных элементов массива. Выгода здесь двоякая: и экономия ячеек памяти, и упрощение программирования.

Рассмотрим способ представления чисел в ФОКАЛе. В состав интерпретатора входит набор программ, обеспечивающих обработку чисел в формате с плавающей точкой, т. е. в ФОКАЛе определен только вещественный тип чисел. Один из операндов и результат всех операций помещаются в накопитель (аккумулятор) с плавающей точкой. Накопитель, носящий имя FLAC, состоит из трех последовательных слов оперативной памяти, обозначенных BE, HORD, LORD и содержащих соответственно порядок, старшие разряды мантиссы со знаком и младшие разряды мантиссы (рис. 3.5).

Порядок представляется в дополнительном коде, величина которого ограничена и находится в диапазоне от -200_8 до $+177_8$ включительно. Это объясняется тем, что числа с плавающей точкой хранятся в памяти в двухсловном формате и для представления порядка отводится один байт. Таким образом, наибольший положительный порядок в слове BE равен 000177_8, а наибольший по модулю отрицательный порядок равен 177600_8 (дополнительный код числа 200_8).

Мантисса числа с плавающей точкой представляется в дополнительном коде в нормализованном виде и занимает три байта: старший и младший байты слова HORD и старший байт слова LORD. Младший байт слова LORD не используется (в нем записывается внутренний код пробела 200_8).

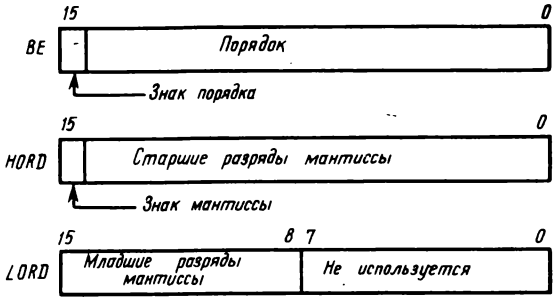


Рис. 3.5. Представление числа с плавающей точкой в аккумуляторе

Имеется возможность «просмотра» представления различных чисел с плавающей точкой в ячейках накопителя. Для этого вначале нужно исполнить прямую команду

* XECUTE <число> <ВК>

перевести микроЭВМ в пультовый режим и «открыть» ячейки с адресами, соответствующие словам BE, HORD и LORD (1254, 1256, 1260).

Ниже приведены примеры представления чисел +1 и —1 в накопителе с плавающей точкой:

Адрес	Содержимое	Адрес	Содержимое
001254	000001	001254	000001
001256	040000	001256	140000
001260	000200	001260	000200

Выполнив прямую команду

* XECUTE 1/7 <ВК>

получим в накопителе представление дробного числа:

Адрес	Содержимое
001254	177776
001256	044444
001260	111200

В ячейке по адресу 1254 записан порядок (число —2 в дополнительном коде), положительная мантисса в нормализованном виде хранится в слове по адресу 1256 и в старшем байте слова по адресу 1260.

Числа с плавающей точкой в памяти ЭВМ хранятся в несколько ином формате, занимая два последовательных слова (рис. 3.6).

При выполнении операций и функций над числами с плавающей точкой двухсловный формат (внешний) преобразуется в трехсловный формат (внутренний). Например, дробь $+^1/_7$ во внешнем формате выглядит так:

Адрес	Содержимое
α	111376
$\alpha+2$	044444

По адресу α в младшем байте слова записан порядок числа (дополнительный код числа —2), а в старшем байте — младшие разряды мантиссы. В следующем слове расположены знак мантиссы и ее старшие разряды.

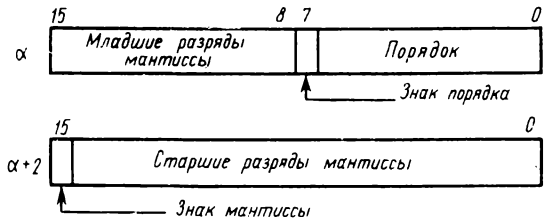


Рис. 3.6. Представление числа с плавающей точкой в памяти ЭВМ

Диапазон представимых чисел зависит практически только от числа разрядов, отведенных под порядок в рассматриваемом формате

$$2^{-200} \leqslant |X| \leqslant 2^{+177},$$

или в десятичной системе счисления

$$10^{-38} \leqslant |X| \leqslant 10^{+38}.$$

Общая длина мантиссы составляет 23 бит, не считая знакового разряда. Учитывая известное соотношение между десятичной и двоичной системами счисления

$$n = \lg 2\, m = 0{,}3010 \cdot 23 = 6{,}923,$$

где n — число десятичных разрядов; m — число двоичных разрядов, заключаем, что точность представления чисел в ФОКАЛе составляет 6—7 значащих десятичных цифр.

Реально при выполнении операций над числами с плавающей точкой возникают дополнительные погрешности, вызванные потерями значащих разрядов при выравнивании порядков, округлениями, преобразованиями из десятичной системы счисления в двоичную и наоборот. Учитывая это, интерпретатор ФОКАЛа реализован таким образом, что на печать выводится не более шести значащих десятичных цифр, причем шестая цифра может быть округлена. Несколько пониженная точность является своеобразной платой за компактность, простоту реализации и достаточно высокую скорость интерпретации основных конструкций ФОКАЛа.

Отсутствие в языке представлений и операций над числами целого типа (тип INTEGER) не означает, что в ФОКАЛе невозможны действия с целыми числами. Известно, что в ограниченном диапазоне, когда количество значащих цифр целого числа в двоичном представлении

умещается в разрядах, отведенных под мантиссу, целые числа представляются точно.

Например, максимальное целое положительное число в формате с плавающей точкой будет выглядеть следующим образом:

Адрес	Содержимое
α	177427
$\alpha+2$	077777

Знаковый разряд мантиссы содержит цифру 0, а во всех остальных 23 разрядах — цифру 1. Порядок представляет младший байт слова по адресу α, и в приведенном примере он равен 27_8 или 23_{10}, что соответствует количеству единиц в мантиссе. Таким образом, в двух соседних ячейках памяти в формате с плавающей точкой записано целое положительное число:

$$N_{max} = 2^{23} - 1_{,} = 8388607.$$

Так как мантисса отрицательных чисел представляется в дополнительном коде, наибольшее по абсолютной величине целое отрицательное число во внешнем двухсловном формате будет выглядеть так:

Адрес	Содержимое
α	000027
$\alpha+2$	100000

и ему будет соответствовать число

$$-N_{max} = -2^{24} = -8388608.$$

Казалось бы, что в пределах указанного диапазона арифметические операции над целыми числами должны выполняться точно. На самом деле это не так, и в этом можно убедиться на примере програмы суммирования натурального ряда чисел от 1 до N и от 2 до N.

По результатам вычислений на ЭВМ можно убедиться, что начиная со значения $N = 1190$ результаты суммирований перестают отличаться из-за пропадания единицы младшего разряда. Это объясняется недостаточной точностью работы пакета программ плавающей арифметики интерпретатора ФОКАЛа.

Рассмотрим еще один пример. Исполним в диалоговом режиме оператор

```
* XEXUTE 8388607
```

и просмотрим в пультовом режиме содержимое ячеек аккумулятора:

```
BE    000027
HORD 077777
LORD 177600
```

Убеждаемся, что указанное число из символьного формата переведено во внутренний точно (напомним, что младший байт ячейки не используется).

Однако если выполнить оператор

```
* TYPE 8388607
8388620 *
```

то видно, что результат отличается от ожидаемого, т. е. перевод в десятичную систему счисления выполнен приближенно (напомним, что на печать выводится только шесть значащих цифр).

С появлением отечественного микропроцессора К1801ВМ2 и разработанной на его основе одноплатной микроЭВМ «Электроника МС 1201.02» представилась возможность создать версию интерпретатора ФОКАЛа с более высокой точностью выполнения операций с плавающей точкой, сохранив его основное достоинство — компактность.

В состав команд этой ЭВМ входят четыре команды плавающей арифметики: сложение, вычитание, умножение и деление. Набор этот имеет специальное название FIS — Floating Instruction Set.

Формат представления чисел с плавающей точкой отличается от ранее рассмотренного и показан на рис. 3.7.

Число также размещается в двух последовательных словах, однако двоичный порядок располагается в первом слове с 14-го по 7-й бит, старшие ряды мантиссы занимают с 6-го по 0-й бит этого же слова. Младшие разряды мантиссы занимают следующее слово. Знак мантиссы кодирует 15-й бит первого слова.

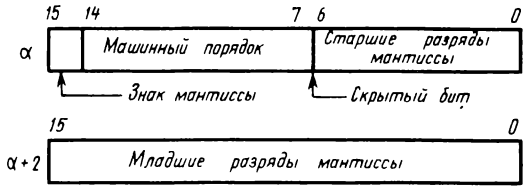


Рис. 3.7. Стандартный формат представления числа с плавающей точкой

Порядок числа представлен с избытком, равным 128 (200_8), в виде так называемого машинного порядка

$$p_м = p + 128,$$

где $p_м$ — машинный порядок; p — истинный порядок.

Истинный порядок может принимать значения от -128 (200_8) до $+127$ (177_8), а машинный порядок — соответственно от 0 до 255 ($0_8 - 377_8$). Введение машинного порядка позволяет использовать только целые положительные числа для его записи, что упрощает реализацию в процессоре действий над порядками. Например, сравнение машинных порядков можно выполнять без предварительного анализа их знаков.

Мантисса числа записывается в прямом коде. Она всегда считается нормализованной, т. е. ее абсолютное значение лежит в пределах

$$^1/_2 \leqslant m \leqslant 1 - 2^{-n},$$

где 2^{-n} — единица младшего разряда мантиссы.

В связи с тем что старший разряд нормализованной мантиссы всегда равен единице, имеется принципиальная возможность при хранении числа в памяти этот бит опускать. При обработке чисел центральный процессор этот бит восстанавливает. Таким образом, хотя в записи числа мантисса содержит 23 бит (7 бит в первом слове и 16 во втором), в действительности она имеет 24 бит, учитывая так называемый «скрытый» бит. Заметим, что скрытый бит всегда имеет значение 1.

В виде исключения скрытый бит считается равным нулю, если все число равно нулю. Чтобы можно было изобразить число нуль, «жертвуют» наименьшим отрицательным порядком. Аппаратура процессора сделана таким образом, что если истинный порядок имеет значение -128 (-200_8), т. е. машинный порядок равен 0, то она автоматически во всех остальных разрядах генерирует нули. Следовательно, нуль с плавающей точкой и нуль с фиксированной точкой имеют одинаковое представление.

В рассматриваемом формате наименьшее по абсолютному значению действительное число

$$X_{min} = 2^{-p_{max}} m_{min} = 2^{-127} \cdot {}^1/_2 = 0{,}2938739 \cdot 10^{-38},$$

а наибольшее

$$X_{max} = 2^{+p_{max}} m_{max} = 2^{+127}(1 - 2^{-24}) = 1{,}701410 \cdot 10^{+38}.$$

При выходе чисел за указанные пределы ЭВМ фиксирует переполнение. Здесь уместно остановиться на терми-

нологии. Если результат некоторой операции превышает максимально допустимое число, то обычно выдается диагностическое сообщение «переполнение порядка». Для противоположного случая, когда результат меньше минимального, могут быть такие варианты сообщений: «исчезновение порядка» либо «антипереполнение порядка». Первый термин скорее всего связан с понятием машинного порядка. Действительно, так как для представления машинного порядка используются только целые положительные числа, то при появлении отрицательных значений можно считать, что порядок «исчез». По-видимому, неискушенному в вычислительной технике пользователю все же удобнее оперировать с истинными порядками. Поэтому в новой версии интерпретатора ФОКАЛа для случаев переполнения были выбраны следующие диагностические сообщения: «порядок больше E + 38» либо «порядок меньше E — 38».

В микроЭВМ «Электроника МС 1201.02», как и в мини-ЭВМ типа СМ-4, переполнение в операциях над числами с плавающей точкой фиксируется аппаратным образом с помощью вектора внутреннего прерывания, находящегося по адресу 244_8 — 246_8.

Количество значащих цифр числа с плавающей точкой определяется числом двоичных разрядов, отводимых под мантиссу (знаковый бит не учитывается). Вместе со скрытым битом количество двоичных разрядов в мантиссе равно 24, поэтому

$$n_{\text{дес}} = 0{,}3010 \cdot 24 = 7{,}224.$$

Таким образом, действительные числа в рассмотренной форме записи имеют не менее семи значащих десятичных цифр.

Нетрудно видеть, что максимальное целое положительное число, последняя цифра которого в двоичной записи отлична от нуля, записанное в форме с плавающей точкой, будет иметь значение

$$N_{\text{max}} = 2^{24} - 1 = 16777215.$$

На рис. 3.8 приведена запись этого числа в двух соседних ячейках памяти ЭВМ. Знаковый 15-й бит равен нулю, так как число положительное; машинный порядок равен 230_8; все биты мантиссы содержат единицы, в том числе и скрытый бит.

В отношении максимального целого числа необходимо сделать уточнение. Оно понимается в том смысле, что все целые числа от 1 до 16777215 в разрядной сетке

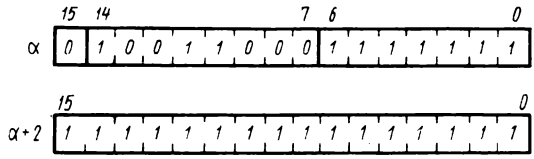


Рис. 3.8. Представление максимального числа

будут записаны точно. Будет записано точно и число 16777216, а также и бóльшие числа, кратные двойке. Однако для удобства проведения операций над целыми числами целесообразно придерживаться указанной границы.

Следует также принять во внимание, что при операциях сложения, вычитания и умножения целые числа остаются целыми (случаи превышения диапазона не рассматриваются), но в случае деления, когда частное не является целым, появляются действительные числа.

В табл. 3.1 приведены представления некоторых чисел в памяти ЭВМ в форме с плавающей точкой.

В качестве примера рассмотрим выполнение команды сложения чисел с плавающей точкой. Формат команды FADD в восьмеричном представлении выглядит следующим образом:

0 7 5 0 0 R,

где R — один из регистров общего назначения, используемый как указатель стека. При выполнении команды операнды выбираются из соответствующего стека, вычисляется результат, который снова заносится в стек.

В качестве указателя стека выберем регистр R1 и занесем в него в пультовом режиме адрес 2000:

R1/002000.

По адресу 2000 занесем в память ЭВМ последовательно два операнда:

R1 → 2000/ 040200 } «1» с плавающей точкой
2002/ 000000 }

2004/ 040400 } «2» с плавающей точкой
2006/ 000000 }

В памяти теперь организован стек, в котором по адресам 2000 и 2002 записан первый операнд («источник»), а по адресам 2004 и 2006 — второй операнд

Таблица 3.1

Десятичное число	Двоичное с плавающей точкой	Представление в памяти	
1	$0{,}1 \times 2^1$	040200	000000
2	$0{,}1 \times 2^2$	040400	000000
5	$0{,}101 \times 2^3$	040640	000000
10	$0{,}101 \times 2^4$	041040	000000
—1	$-0{,}1 \times 2^1$	140200	000000
0,5	$0{,}1 \times 2^0$	040000	000000
0,25	$0{,}1 \times 2^{-1}$	037600	000000
—0,25	$-0{,}1 \times 2^{-1}$	137600	000000
0,1	$0{,}110011001... \times 2^{-3}$	037314	146315
$\pi/2$	$0{,}110010010... \times 2^1$	040311	007732

(«приемник»). Указатель стека (регистр R1) стоит по адресу 2000.

Далее занесем по адресу 1000 команду сложения и в следующую ячейку команду останова:

```
1000/075001     FADD R1
1002/000000     HALT
```

Дадим команду выполнить

```
1000G
```

и вновь просмотрим содержимое стека:

```
        2000/ 040200 }
        2002/ 000000 }  «1» с плавающей точкой
R1 →    2004/ 040500 }
        2006/ 000000 }  «3» с плавающей точкой
```

Обратите внимание, что результат находится по месту «приемника» (прежний операнд затерт), «источник» остался без изменения, а указатель стека переместился на адрес «приемника». 2004.

Аналогично выполняются и остальные команды плавающей арифметики. Заметим, что в команде вычитания уменьшаемое располагается по адресу «приемника», а в команде деления по адресу «приемника» находится делимое. Во всех случаях результат записывается по адресу «приемника».

Напомним также, что в отличие от арифметических команд с фиксированной точкой ADD, SUB, MUL, DIV команды с плавающей точкой «реагируют» на переполнение разрядной сетки возникновением внутреннего прерывания по адресу 244_8.

Рассмотренный формат представления чисел с плавающей точкой является стандартным для ЭВМ с системой команд машин типа PDP-11. К ним относится и семейство отечественных машин СМ ЭВМ (СМ3, СМ4, СМ 1401, СМ 1403); микроЭВМ «Электроника-60», ДВК-3, ДВК-4; мини-ЭВМ «Электроника-100/25», «Электроника-79» и др.

Полезно сравнить способы представления чисел с плавающей точкой в машинах ЕС ЭВМ и СМ ЭВМ.

Число с плавающей точкой с одинарной точностью в ЕС ЭВМ также занимает 32 двоичных разряда (рис. 3.9). Биты нумеруются слева направо. Нулевой бит кодирует знак числа, биты с первого по седьмой представляют машинный порядок, оставшиеся 24 бит — мантиссу. Истинный порядок изменяется в диапазоне

$$-64 \leqslant p \leqslant +63,$$

а машинный — в диапазоне

$$0 \leqslant p_м \leqslant +127.$$

Характерной особенностью машин ЕС ЭВМ является то обстоятельство, что для представления чисел с плавающей точкой в них используется шестнадцатеричная система счисления. Нормализованная мантисса (без учета знака) лежит в пределах

$$1/16 \leqslant m \leqslant 1 - 16^{-n},$$

или в шестнадцатеричной системе счисления

$$0{,}100000 \leqslant m \leqslant 0{,}FFFFFF.$$

Таким образом, мантисса содержит шесть шестнадцатеричных цифр (каждая шестнадцатеричная цифра

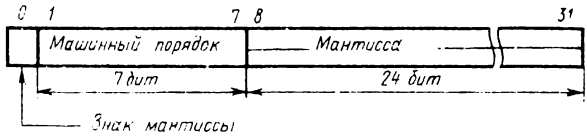


Рис. 3.9. Представление числа с плавающей точкой в ЕС ЭВМ

представляется двоичной тетрадой). Отметим, что минимальная нормализованная мантисса в двоичном представлении имеет после запятой подряд идущие три нуля:

$$0{,}0001000...0 \leqslant m \leqslant 0{,}1111111...1.$$

С учетом шестнадцатеричного основания максимальное по абсолютному значению число

$$X_{\max} = 16^{+p_{\max}} m_{\max} = 16^{+63}(1 - 16^{-6}) = 0{,}723705 \cdot 10^{+76}.$$

Соответственно минимальное число

$$X_{\min} = 16^{-p_{\max}} m_{\min} = 16^{-64} \cdot 1/16 = 16^{-65} = \\ = 0{,}539761 \cdot 10^{-78}.$$

Итак, диапазон представимых чисел с плавающей точкой в ЕС ЭВМ значительно превышает соответствующий диапазон чисел в СМ ЭВМ при одинаковом числе двоичных разрядов, выделяемых для записи числа. Однако увеличение диапазона достигается за счет некоторого снижения точности представления чисел. Действительно, максимальная относительная погрешность представления чисел с плавающей точкой определяется соотношением

$$\delta_{\max} = \frac{S^{-n}S^{p}}{m_{\min}S^{p}} = \frac{S^{-n}}{1/S} = S^{-n+1},$$

где S — основание системы счисления; $m_{\min}$ — нормализованная мантисса (минимальное значение); n — число разрядов в мантиссе.

Для СМ ЭВМ $S = 2$, $n = 24$, $m_{\min} = 1/2$ и поэтому $\delta_{\max} = 2^{-23} = 1{,}19 \cdot 10^{-7}$.

Для ЕС ЭВМ $S = 16$, $n = 6$, $m_{\min} = 1/16$ и $\delta_{\max} = 16^{-5} = 9{,}54 \cdot 10^{-7}$.

В качестве положительного момента для представления чисел с шестнадцатеричным основанием можно отметить ускорение выполнения некоторых операций, в частности нормализации и денормализации, за счет того, что сдвиг производится сразу на четыре двоичных разряда. Кроме того, уменьшается вероятность появления ненормализованных чисел в процессе вычислений, так как в нормализованной мантиссе допускается наличие трех нулей после запятой.

С целью повышения точности выполнения арифметических операций и уменьшения погрешности вычислений стандартных функций интерпретатор ФОКАЛ был переработан под стандартное представление чисел с плавающей точкой. Это дало возможность использовать команды плавающей арифметики микроЭВМ «Электроника МС 1201.02», а также значительную часть пакета стандартных программ микроЭВМ «Электроника-60». Объем интерпретатора при этом вопрос незначительно. Напомним, что чем больше места в адресном пространстве ЭВМ занимает интерпретирующая система, тем меньше места остается для программ пользователя.

Приведем несколько примеров, полученных с использованием новой версии интерпретатора ФОКАЛа:

```
* TYPE FSQT(16777216)
      4096
* TYPE FATAN(1)*4
      3.14159274
* TYPE 2↑24-1
      16777215
* TYPE 2↑24
      0.1677722E08
```

При испытании программы суммирования целых чисел правильный результат получается вплоть до числа $N = 5792$:

$$\sum_{1}^{N} = 16776528; \quad \sum_{2}^{N} = 16776527.$$

Суммирование чисел до $N = 5793$ приводит к результату, который не умещается как целое число в принятом формате. Интерпретатор выводит на печать значение суммы в экспоненциальном формате с семью значащими цифрами:

$$\sum_{1}^{N} = 0.1678232\text{E}08 \quad \sum_{2}^{N} = 0.1678232\text{E}08.$$

Как видно, результат вычислен приближенно.

Таким образом, использование стандартного формата представления чисел с плавающей точкой позволило повысить точность выполнения арифметических операций (практически получить дополнительную значащую цифру). Можно с достаточной уверенностью утверждать, что

точность вычисления на языке ФОКАЛ с использованием микроЭВМ «Электроника МС 1201.02» не хуже, чем на машинах ЕС ЭВМ с использованием ФОРТРАНа с обычной точностью.

Динамический способ распределения переменных в памяти является одной из примечательных особенностей ФОКАЛа. Для устранения трудностей, которые могут возникнуть при уяснении принципов реализации принятого метода, рассмотрим конкретный пример.

Пусть выполнена прямая команда

```
* ERASE ALL <BK>
```

выполняющая очистку буферов текста и переменных, и вводится следующая строка:

```
1.01 SET A = 1 <BK>
```

После выполнения этой строки в память машины запишутся переменная А и ее значение. Найдем ее адрес в соответствии с алгоритмом, приведенным ранее. Так как это однобуквенная переменная, то интерпретатор автоматически впереди и этой буквы добавит символ «пробел». Букве А соответствует восьмеричный код 101, а символу «пробел» — 200 (во внутреннем представлении). В некоторой ячейке памяти, принадлежащей области стека, имя переменной будет записано в старшем (код пробела) и младшем (код буквы А) байтах:

10 000 000 01 000 001,

или в восьмеричном представлении

1 0 0 1 0 1.

Именно в таком виде впоследствии имя переменной будет записано в первое слово четырехсловного представления переменной. Так как индексы отсутствуют, то во втором слове будут записаны нули.

Займемся вычислением смешанного кода переменной, т. е. адреса первого слова в оперативной памяти.

В соответствии с обозначениями, принятыми ранее, имеем в нашем случае:

x2 = 200; x1 = 101;

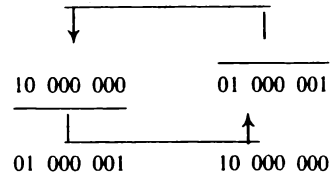
y2 = 000; y1 = 000.

Смысл определяемых выражением действий достаточно прост и заключается в помещении значения x2 в старший байт слова, а x1 — в младший. Соответственно значение индекса y2 помещается в старший байт следую-

щего слова, а значение индекса y1 — в младший байт того же слова. Затем выполняется операция сложения слов. Как и следовало ожидать, результат вычислений

1 0 0 1 0 0

в этом конкретном случае совпал с именем переменной вследствие нулевых значений индексов. Далее выполняем операцию обмена местами старшего и младшего байта:



или в восьмеричном представлении

0 4 0 6 0 0

Переходим к определению размера области памяти, отведенной для размещения переменной. Информация о начале области содержится в системной рабочей ячейке ФОКАЛа с именем START (адрес 1304). Содержимое этой ячейки памяти динамически изменяется по мере увеличения размера буфера текста. Информация о конце области записана в ячейке с именем BOTTOM (адрес 1310). В случае полного объема оперативной памяти и ее исправности (для ФОКАЛа 48 К байт) это адрес самой последней ячейки 137776. Далее идет ПЗУ с интерпретатором ФОКАЛа. Заметим, что сама ячейка BOTTOM в область переменных не входит.

Содержимое ячейки START для рассматриваемого примера равно 1466, отсюда размер области переменных, вычисленный как дополнение,

001466 — (137776 — 4) = 041474.

Здесь конечный адрес уменьшен на 4 для того, чтобы оставить место для одной переменной.

Операция определения смешанного кода по модулю размера области памяти включает применение команды BIC — очистка битов по маске. Дадим ее в двоичных кодах:

0 100 001 100 111 100 (маска),
0 100 000 110 000 000

0 000 000 010 000 000

т. е. во втором операнде обнуляются те биты, напротив которых стоят единицы маски. Затем полученный результат берем по модулю 10_8. Для этого в последнем выражении обнуляются три младших бита (они здесь нулевые).

И наконец, заключительная операция — складываем адрес начала буфера переменных (он находится в ячейке START) с полученным выражением:

```
  001466
+ 000200
--------
  001666
```

Это и есть смешанный код, или хеш-адрес переменной A. «Открыв» по указанному адресу четыре последовательных ячейки, убеждаемся, что в них записано имя переменной и ее значение:

Адрес	Содержимое	Пояснение
001666	100101	␣ A
001670	000000	индексы
001672	000001	число с плавающей
001674	040000	точкой

На микроЭВМ можно поставить небольшой эксперимент, чтобы оценить максимальную емкость буфера переменных и время, необходимое для его заполнения. Используем с этой целью следующие прямые команды:

```
* ERASE ALL <ВК>
* FOR I = 1, 10000; SET A (I) = 1 <ВК>
```

По истечении 6—7 мин на экране дисплея появится сообщение

```
? 11 AT 0.00
НЕТ МЕСТА ДЛЯ ПЕРЕМЕННЫХ В ПАМЯТИ
```

Переведя микроЭВМ в пультовый режим, можно убедиться, что, начиная с адреса 1466 (адрес начала буфера переменных) и по адрес 137774 включительно, будут записаны индексированные переменные с одинаковым значением (единица в форме с плавающей точкой).

Чтобы вернуть микроЭВМ в режим интерпретации, нужно перевести клавишу в положение «Прогр.» и нажать латинскую букву P на клавиатуре. Появление символа * сигнализирует о возвращении в режим интерпретации. Запросив значение переменной I

```
TYPE I <ВК>
6041.0000
```

узнаем, что максимальное значение индекса до момента переполнения было равно 6040. Относительное большие

временные затраты объясняются тем, что по мере увеличения количества записываемых переменных интерпретатору все «труднее» становится подыскивать место для новых переменных.

Из сказанного становится также ясной причина выдачи всех переменных из памяти по команде

```
* TYPE ¤<ВК>
```

в формате оператора SET. Такой способ дает возможность проведения и обратной операции. Например, если в качестве устройства вывода назначить перфоратор, то выведенные переменные могут быть снова введены в память через ленточный считыватель (см. оператор OPERATE).

3.3. Усовершенствование датчика случайных чисел

Случайные числа на ФОКАЛе вырабатываются при обращении к встроенной функции FRAN. Закон распределения чисел близок к равномерному, интервал распределения заключен между —1 и +1. Простая проверка позволяет убедиться в невысоком качестве выдаваемых последовательностей случайных чисел. Для этого заставим ЭВМ в течение некоторого времени непрерывно выполнять функцию вывода на экран дисплея точки, координаты которой генерирует датчик случайных чисел:

```
1.10 X FP (100+50*FRAN(),100+50*FRAN());G 1.10
```

При идеальном датчике случайных чисел должен получиться квадрат размером 100×100, равномерно заполненный светлыми точками. На самом деле из-за крайне высокой корреляции между подряд идущими случайными числами точки группируются в семейство волнообразных линий с пустыми промежутками между ними.

В [21] описан датчик случайных чисел, основанный на линейном конгруэнтном методе. Генерируемая последовательность псевдослучайных целых чисел определяется рекуррентным соотношением

$$Y_{n+1} = aY_n + c(\mathrm{mod}m),\ n \geqslant 1.$$

Значение m зависит от длины разрядной сетки используемой ЭВМ. Так как в нашем случае машины 16-разрядные, а целые числа должны быть положительными, то

$$m = 2^{15} = 32768.$$

Значения коэффициентов c и a в соответствии с рекомендациями из книги Д. Кнута [22]

$$c \approx m(1/2 - \sqrt{3}/6) \approx 6925,$$
$$a \approx (1/8)m\pi \approx 12869.$$

Коэффициент c должен быть нечетным, а коэффициент a удовлетворять следующим условиям:

$$a(\mathrm{mod}8) = 5,$$
$$m/100 < a < m - \sqrt{m}.$$

Подстановкой числовых значений m и a легко убедиться, что эти условия выполняются.

Ниже приведена программа встроенной функции генератора случайных чисел на языке ассемблера. Необходимые пояснения даются в комментариях, методика написания встроенных функций освещена в [23]:

```
XRAN:  MOV # IY, RO     ; адрес хранения IY в RO
       TST FLAC         ; опросить содержимое аккумулятора
       BEQ 1¤           ; перейти, если 0
       MOV # 1, (RO)    ; запись 1 в IY
1¤:    MOV (RO), R1     ; переслать IY в R1
       MUL # 12869., R1 ; умножить на коэффициент a
       ADD # 6925., R1  ; прибавить коэффициент c
       BPL 2¤           ; перейти, если результат ⩾ 0
       ADD # 32768., R1 ; иначе сложить с модулем m
2¤:    MOV R1, (RO)     ; новое целое записать в IY
       FITOR            ; преобразовать в вещественное
       FMULL+IMMED      ; умножить на константу
       34400            ; константа в форме с плавающей точкой
       O                ; 1/32768
       RTS PC           ; выход
```

При каждом обращении к функции FRAN (с нулевым аргументом) выдается вещественное число из интервала [0,1]. Иногда бывает необходимо генерировать каждый раз одну и ту же последовательность случайных чисел. Для этого следует исполнить вначале один раз функцию FRAN с ненулевым значением аргумента. Как видно из приведенной программы, в качестве начального значения целой константы будет использована 1.

Для оценки качества нового датчика случайных чисел рекомендуем провести наблюдение на экране дисплея за заполнением светлыми точками квадрата размером 100×100:

```
1.10 X FP(200 + 100 * FRAN(),50 + 100 * FRAN()); G 1.10
```

Сравнение окажется явно в пользу последнего варианта. Другие важные характеристики датчика случайных чисел — математическое ожидание и среднеквадратическое отклонение — достаточно близки к теоретическим. Например, для выборки из 2000 случайных чисел были получены следующие значения:

$МО = 0{,}4980;\ \sigma = 0{,}2839.$

Соответствующие теоретические значения для равномерного закона распределения случайных чисел на интервале [0,1] равны:

$МО = 0{,}5000;\quad \sigma = 0{,}2870.$

3.4. Системное программное обеспечение учебной локальной сети

Для функционирования учебной локальной сети на базе ДВК необходимы следующие программные компоненты:

программные средства в ПЗУ интерпретатора ФОКАЛа для приема и передачи файлов в главную микроЭВМ;

программные средства управления файлами в главной микроЭВМ;

программные средства для приема и передачи файлов из главной микроЭВМ в периферийные.

В связи с тем что все функции управления обменом в сети возложены на главную микроЭВМ, основная нагрузка ложится на ее программные средства. Принятие такого решения продиктовано ограниченным объемом ПЗУ интерпретатора в периферийных машинах (4 К слов). Для главной машины ДВК-2М таких ограничений практически нет, так как она имеет операционную систему на гибких магнитных дисках. Поддержка программных средств управления файлами (ведение каталога, запись, считывание, удаление, переименование, сжатие и т. п.) осуществляется стандартной операционной системой RT11. Системные программы, поддерживающие протокол связи с периферийными микроЭВМ, написаны на языке ассемблера с использованием системных макрокоманд ее библиотеки.

К программным средствам интерпретатора ФОКАЛа, обеспечивающим протокол связи периферийной ЭВМ с главной, предъявляются следующие требования:

приемлемая скорость передачи файлов по каналу связи;

надежность (безошибочность) передачи файлов;

компактность (в виду ограниченного объема ПЗУ).

Реализованный в ФОКАЛе механизм ввода программных строк (см. § 3.1), хотя и обладает удобством использования и гибкостью, в то же время имеет один существенный недостаток — ограниченную скорость ввода. Объясняется этот факт тем, что процессору приходится выполнять достаточно большой объем работы, связанный с:

записью вводимой программной строки сначала во внутренний буфер строки;

переписью строки из внутреннего буфера в буфер текста;

вычислением смещений для обеспечения перехода от одной строки к другой в порядке возрастания их номеров;

удалением старой строки с совпадающим номером;

подвижкой программных строк с целью заполнения свободного места, оставшегося от вычеркнутой строки;

установкой текущих значений указателей конца буфера текста и начала области переменных (ячейки BUFR = 1302 и STARTV = 1304);

очисткой ячеек памяти буфера переменных, если указатель конца буфера текста «надвигается» на указатель начала буфера переменных.

При вводе программных строк с клавиатуры этот недостаток практически незаметен. Несколько сложнее обстоит дело при вводе программы с перфоленты. Через перфоленточный считыватель она движется рывками (каждый раз считывается строка, а затем следует пауза, связанная с упаковкой строки в буфер текста). Даже при считывании символьной информации, записанной в накопителях на жестких магнитных дисках мини-ЭВМ СМ4, время чтения интерпретатором ФОКАЛ программных строк составляет 5—10 мин для обучающих программ среднего объема. Очевидно, такая скорость передачи программ практически неприемлема для класса обучения программированию, содержащего 12 микроЭВМ (следует признать, что и это количество машин для класса недостаточно).

Для увеличения скорости ввода и вывода программ было решено передавать их единым блоком в формате абсолютной загрузки [16]. С этой целью в интерпретатор ФОКАЛа были включены программы, обеспечивающие ввод — вывод буфера текста единым блоком через последовательный канал. При этом попутно решается еще од-

на важная проблема: осуществляется контроль передачи путем суммирования передаваемых байтов и сравнения полученной суммы с контрольной суммой, записанной в конце файла.

При таком способе скорость передачи ограничивается последовательным каналом связи ИРПС. Практика показала, что при записи и считывании обучающих программ максимальной длины (порядка 40 К байт) время передачи не превышает 1,5—2 мин.

Для передачи программ, записанных на гибких магнитных дисках главной машины ДВК-2М, в периферийные машины ДВК-1М используется служебная программа PUT. SAV*. Эта программа обеспечивает параллельную побайтную передачу выбранного файла во все двенадцать периферийных микроЭВМ одновременно. Возможность быстрой загрузки обучающей программы по выбранной теме во все машины за короткий промежуток времени (порядка 2 мин) является важной и полезной функцией для организации эффективной работы в классе. Именно это обстоятельство продиктовало необходимость наличия двенадцати каналов последовательной связи в составе главной ЭВМ. Конструктивно каналы ИРПС расположены на двух платах КТЛК-6.

В качестве примера приведен протокол загрузки программы с именем TASK в периферийные машины с номерами 1, 2, 3, 4, 5, 7, 9, 12:

```
  R PUT
ПАРАЛЛЕЛЬНЫЙ ЗАГРУЗЧИК  V02.00
ВВЕДИТЕ ЧЕРЕЗ  ',' НОМЕРА ДВК-1  (1...12)
ИСПРАВЛЕНИЯ ВНОСИТЕ <ЗБ>, КОНЕЦ ВВОДА - <ВК>
 1,2,3,4,5,7,9,12

УСТАНОВЛЕНА СВЯЗЬ С ДВК-1   #1
УСТАНОВЛЕНА СВЯЗЬ С ДВК-1   #2
УСТАНОВЛЕНА СВЯЗЬ С ДВК-1   #3
УСТАНОВЛЕНА СВЯЗЬ С ДВК-1   #4
УСТАНОВЛЕНА СВЯЗЬ С ДВК-1   #5
УСТАНОВЛЕНА СВЯЗЬ С ДВК-1   #7
        НЕТ СВЯЗИ С ДВК-1   #9
УСТАНОВЛЕНА СВЯЗЬ С ДВК-1   #12

ЗАГРУЖАТЬ ? <'Д' ИЛИ 'Н' > <ВК>
Д
ВВЕДИТЕ ИМЯ ФАЙЛА И <ВК>
 TASK
XX  ВЫДАН ФАЙЛ  TASK
 *
```

* Служебные программы приема и передачи файлов в составе локальной сети написаны инженером А. Л. Вильсоном.

По директиве .R PUT происходит загрузка служебной программы в оперативную память ДВК-2М и передача ей управления. Далее преподаватель с помощью клавиатуры вводит те номера периферийных машин, в которые нужно загрузить обучающие или иные программы. Обратите внимание, что в протоколе отмечается и факт отсутствия связи (канал 9). Наиболее вероятно возникновение такой ситуации в случае, когда периферийный ДВК не находится в режиме интерпретации или не включен. В нижней строке протокола на месте знаков XX отображается текущий номер передаваемого блока, а по окончании передачи — общее количество блоков.

Отлаженная программа, находящаяся в памяти периферийной ЭВМ, может быть записана на дискетту главной машины. Все операции выполняются за пультом центральной машины. Работу по приему программы, составленной на ФОКАЛе, в оперативную память ДВК-2М и запись ее на дискетту выполняет специальная служебная программа с именем GET. SAV. Ниже приведен протокол записи файла с именем TASK:

```
  R GET
КОПИРОВЩИК ПРОГРАММ ИЗ ДВК-1 НА ДИСК    V02.01
ВВЕДИТЕ  НОМЕР ДВК-1  (1...12)
ИСПРАВЛЕНИЯ ВНОСИТЕ <ЗБ>, КОНЕЦ ВВОДА - <ВК>
4
ВВЕДИТЕ ИМЯ ФАЙЛА И <ВК>
*TASK=

ПОД ФАЙЛ TASK=
ВЫДЕЛЕНО 240 БЛОКОВ
ВАС УСТРАИВАЕТ ?  ('Д' ИЛИ 'Н' И <ВК>)
Д

XX ПРИНЯТ ФАЙЛ  TASK=
*
```

После появления сообщения об окончании записи файла можно перейти к записи программ из других периферийных ЭВМ. Для выхода из режима приема прогрмм следует нажать клавишу <ПС> (перевод строки).

Преподаватель имеет возможность подключить любую из периферийных ЭВМ к машине преподавателя с целью просмотра, корректировки, проверки выполнения и распечатки программы, находящейся в памяти ДВК-1М. Взаимодействие периферийной и главной машины обеспечивает служебная программа с именем NETP. SAV. Ниже приведен протокол:

```
.R NETP
ПРОГРАММА ВЗАИМОДЕЙСТВИЯ ДВК-2М С ДВК-1М
ВОЗВРАТ В СИСТЕМУ ПО КЛАВИШЕ  <ПС>

ВВЕДИТЕ НОМЕР ДВК-1
5

СВЯЗЬ УСТАНОВЛЕНА

НУЖЕН ЛИ ПАРАЛЛЕЛЬНЫЙ ВЫВОД НА УСТРОЙСТВОПЕЧАТИ (ДА,НЕТ)
НЕТ
*
```

Если вводится ответ НЕТ, то информация выводится только на экран дисплея главной ЭВМ, если вводится ДА, то одновременно происходит вывод и на построчно-печатающее устройство (получение «твердой» копии).

Во время этой операции соответствующий ДВК-1М заблокирован.

Следует отметить, что общая нагрузка на главную ЭВМ при работе в учебном классе относительно невелика. Дисковая система работает только во время передачи или приема программ. Такой щадящий режим работы обеспечивает достаточную долговечность класса в целом.

Аппаратно-программные средства учебной локальной сети хотя и ориентированы на использование языка ФОКАЛ, в то же время не накладывают жестких ограничений на применение других языков программирования. Так, программы, написанные на ФОРТРАНе, ПАСКАЛе, ассемблере и преобразованные в формат абсолютной загрузки, могут быть переданы для исполнения в периферийные ЭВМ. При этом не требуется никакой коррекции программ локальной сети. Передаваемые программы должны быть самостартируемыми, т. е. автоматически запускаться после загрузки. При необходимости загрузки новой программы следует перезапустить интерпретатор ФОКАЛа (выключить и снова включить питание периферийной ЭВМ).

Имеется также возможность загрузить монитор операционной системы в любую выбранную периферийную машину, группу или во все без исключения*. На машине преподавателя запускается специальная программа, обслуживающая в режиме кольцевого опроса запросы с периферийных машин. Так как работа происходит в стандартной операционной системе, возможности учебного класса значительно возрастают. Перечислим некоторые

* Программы разработаны студентом В. М. Полетаевым и адаптированы к учебной локальной сети инж. Г. Ж. Джаошвили.

из них. Для обучаемого, находящегося за пультом периферийной ЭВМ, открыт доступ к файловой системе машины преподавателя. Он может выбрать любую программу и загрузить в свою ЭВМ. В частном случае это может быть интерпретатор БЕЙСИКа или любого другого языка программирования. Появляется возможность работы с текстовыми редакторами и другими компонентами операционной системы.

Основным сдерживающим фактором использования указанного режима работы является низкая скорость передачи по последовательным каналам, что сильно затрудняет работу с компиляторами, требующими интенсивного обмена с накопителем на гибких магнитных дисках.

Описанный вариант использования учебной локальной сети не требует обязательного наличия микросхем ПЗУ с интерпретатором ФОКАЛа в периферийных машинах. В этом случае загрузка необходимых компонентов операционной системы начинается с запуска программы начальной загрузки, находящейся в системном ПЗУ периферийной ЭВМ. Для этого в пультовом режиме набирается директива 177550L.

Более подробно особенности работы локальной сети и интерпретирующей системы описаны в Прил. 6.

глава 4
Комплект учебной вычислительной техники на базе бытовой микроЭВМ «Электроника БК-0010»

Рассмотренный в предыдущих главах комплект учебной вычислительной техники на базе диалоговых вычислительных комплексов (ДВК) типа «Электроника НЦ 80-20/1,2М» при всех своих несомненных достоинствах обладает одним существенным недостатком сравнительно высокой стоимостью. Это обусловлено тем, что разработка ДВК проводилась прежде всего с учетом использования их для решения инженерных, экономических, статистических и других задач, требовавших достаточно высокой вычислительной мощности и развитого периферийного оборудования. Использование ДВК для учебных целей, особенно в средних школах и профессионально-технических училищах, несколько избыточно, но оправдано в условиях острой нехватки средств вычислительной техники для организации кабинетов изучения основ информатики и вычислительной техники.

Существенно меньшей стоимостью при достаточно широких возможностях обладает комплект учебной вычислительной техники, в котором в качестве рабочего места преподавателя используется ДВК типа «Электроника НЦ 80-20/2М» в полной комплектации, а на рабочих местах учащихся применяются серийно выпускаемые бытовые микроЭВМ типа «Электроника БК-0010» [19]. Такой комплект стоит в три раза дешевле рассмотренного в предыдущих главах [20].

4.1. Бытовая микроЭВМ «Электроника БК-0010»

Бытовая микроЭВМ «Электроника БК-0010», имея сравнительно низкую стоимость, предназначена для проведения расчетов и вычислений, не требующих больших объемов оперативной памяти, обучения неподготовленных, с различным уровнем образования людей навыкам программирования и основам работы на ЭВМ, управления работой различными видами бытовой техники и т. д. Несомненно достоинством микроЭВМ «Электроника БК-0010» является возможность подключения к ней в качестве дисплея бытового телевизора, а в качестве внешней памяти — бытового магнитофона.

Конструктивно микроЭВМ «Электроника БК-0010»

состоит из двух функционально и конструкивно завершенных узлов информационно-вычислительного устройства и блока сетевого питания.

Информационно-вычислительное устройство смонтировано на двух печатных платах — вычислителя и клавиатуры. Все активные элементы информационно-вычислительного устройства расположены на плате вычислителя. Обе платы установлены в пластмассовый корпус с габаритными размерами 370 × 180 × 70 мм. На верхней лицевой панели корпуса устанавливается цветной шильд клавиатуры, на котором нанесено обозначение всех клавиш и выделены отдельные функциональные зоны. Помимо того, шильд защищает внутренний объем информационно-вычислительного устройства от попадания посторонних частиц.

Информационно-вычислительное устройство имеет отсек пользователя, открыв который можно заменить шильд клавиатуры и ПЗУ (РПЗУ), установленные в двух розетках типа РС-24, для ориентации микроЭВМ на решение каких-либо частных (специальных) задач и придания новых функциональных назначений ряду клавиш.

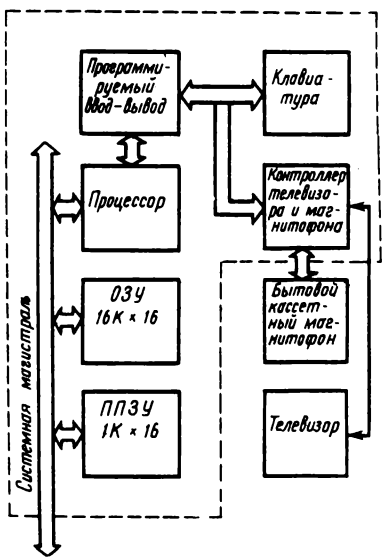


Рис. 4.1. Структурная схема микроЭВМ «Электроника БК-0010»

Разъемы для подключения микроЭВМ к внешним устройствам расположены на задней панели корпуса информационно-вычислительного устройства.

В состав микроЭВМ «Электроника БК-0010» входят (рис. 4.1): процессор, оперативное запоминающее устройство емкостью 32 К байт; постоянное запоминающее устройство емкостью 32 К байт; контроллер клавиатуры и собственно клавиатура; параллельный программируемый интерфейс ввода—вывода; контроллеры кассетного магнитофона, последовательного программируемого

интерфейса ввода—вывода, ОЗУ и бытового телевизионного приемника; внутренняя системная магистраль микроЭВМ; блок питания.

Все основные узлы микроЭВМ связаны с процессором через единую системную магистраль. Благодаря простоте конструкции, использованию элементной базы повышенной степени интеграции, встроенным средствам контроля и диагностики достигаются компактность, высокая надежность и ремонтопригодность.

Возможность наращивания аппаратных средств обеспечивается магистральной структурой микроЭВМ, наличием параллельного и последовательного программируемых интерфейсов и выводом внутренней системной магистрали на отдельный разъем.

Процессор микроЭВМ «Электроника БК-0010» выполнен на базе однокристального 16-разрядного микропроцессора К1801ВМ1 и предназначен для выполнения следующих функций:

вычисления адресов операндов и выполнения команд;

обмена информацией с другими устройствами, подключенными через каналы связи;

содержательной обработки операндов;

реакции на воздействие клавиатуры и устройств пользователя, подсоединенных к параллельному программируемому интерфейсу.

Память состоит из ОЗУ емкостью 32 К байт и ПЗУ емкостью 32 К байт. Блок ОЗУ включает 16 микросхем ОЗУ динамического типа КР565РУ6Г. ПЗУ представляет собой БИС К1801РЕ1 емкостью 4 К 16-разрядных слов и схемой связи с системной магистралью. Предусмотрена возможность установки БИС ПЗУ с программой пользователя в свободную розетку, расположенную в отсеке пользователя.

Контроллер ОЗУ наряду с управлением и регенерацией ОЗУ динамического типа выполняет функции контроллера бытового телевизора.

Независимая схема регенерации обеспечивает чтение слов из ОЗУ и выдачу их на экран. Работу схемы регенерации и цикл обращения процессора к ОЗУ синхронизирует арбитр. Контроллер обеспечивает формирование и отображение алфавитно-цифровой и графической информации на экране в формате 512 $\times$ 256 точек при двух градациях яркости и 256 $\times$ 256 точек при четырех градациях яркости. Такой формат позволяет формировать 24 информационных и одну служебную строку. В каждой

строке в зависимости от режима работы может размещаться 32 символа (матрица 16 × 18 точек) или 64 символа (матрица 8 × 8 точек). Контроллер выполнен на БИС К1801ВП1-37.

Контроллер кассетного накопителя на магнитной ленте построен по принципу программно-аппаратного управления. Все основные функции контроллера реализованы на программном уровне. Аппаратная часть состоит из регистра, предназначенного для временного хранения информации, и схемы преобразователей уровня сигналов. Контроллер обеспечивает запись до 256 К байт на стандартную магнитофонную кассету типа МК60 со скоростью 1200 бод. Метод записи — разновидность широтно-импульсной модуляции. Контроллер клавиатуры предназначен для формирования и параллельного ввода кодов символов (КОИ-7) в системную магистраль микроЭВМ. Контроллер выполнен на БИС К1801ВП1-014.

Клавиатура микроЭВМ представляет собой печатную плату с установленными на ней 92 переключателями ПКН-150. Маркировка клавиш нанесена на цветном шильде клавиатуры. Разными цветами выделены отдельные функциональные группы клавиш: алфавитно-цифровые (зеленые), регистровые (синие), клавиши редактирования (желтые), элементы графики, функциональные (программируемые), управляющие (красные). Клавиатура обеспечивает следующие режимы работы: ввод букв русского и латинского алфавита строчных и прописных; ввод элементов графики; ввод специальных знаков; прерывание программы; шаговый режим; индикацию управляющих символов; редактирование набранного текста; графический режим; запись точки в графическом режиме; стирание точки в графическом режиме; установку табулируемой позиции; переход курсора к следующей табулируемой позиции; ввод в микроЭВМ строки текста.

Последовательный программируемый интерфейс выполнен программно. МикроЭВМ обеспечивает обмен данными с внешними устройствами по протоколу обмена R-232 в диапазоне скоростей от 50 до 9600 бод.

Параллельный программируемый интерфейс ввода — вывода предназначен для подключения устройств пользователя к микроЭВМ через 16 информационных линий ввода и 16 линий вывода. Согласование по уровням ТТЛ.

Кроме информационных линий интерфейс содержит две дополнительные линии: сброс — для начальной уста-

Таблица 4.1

Обозначение линий (сигналов)	Наименование линий	Число линий в группе
АД 15—00	Адрес — данные	16
ОБН	Синхронизация обмена	1
ОТВ	Ответ устройства	1
ДЧТ	Чтение данных	1
ДЗП	Запись данных	1
ЗПР	Запрос на прерывание	1
РЗР	Разрешение на прерывание	1
ОСТ	Останов	1
ПВС	Прерывание по внешнему событию	1
ПЗП	Признак записи байта	1
УСТ	Установка	1

новки устройств пользователя и ПТР — запрос на прерывание от устройств пользователя по фиксированному вектору 100. Питание устройств пользователя должно осуществляться от отдельного источника питания.

Системная магистраль микроЭВМ содержит 26 линий, наименование и назначение которых приведены в табл. 4.1. Системная магистраль соответствует магистрали МПИ (ОСТ 11.305.903—80).

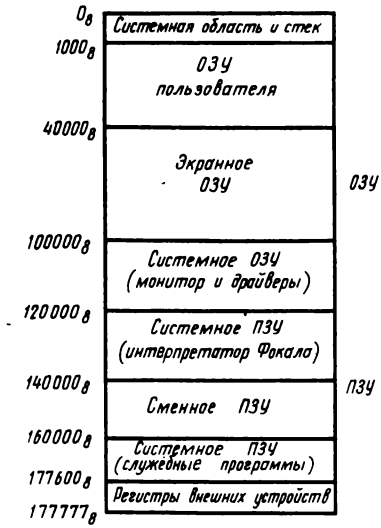


Рис. 4.2. Распределение адресного пространства микроЭВМ «Электроника БК-0010»

По системе команд и архитектуре микроЭВМ «Электроника БК-0010» совместима с микроЭВМ ряда «Электроника 60». Некоторые отличия в распределении адресного пространства объясняются требованиями минимальной стоимости и особенностями применяемых периферийных устройств.

Постоянное и оперативное запоминающие устройства и область экранной памяти микроЭВМ «Электроника БК-0010» расположены в едином адресном прост-

ранстве величиной 64 К байт, распределение которого приведено на рис. 4.2.

МикроЭВМ может работать на двух конфигурациях адресного пространства, устанавливаемых программно. Различие этих конфигураций состоит в распределении адресного пространства между ОЗУ пользователя и ОЗУ экранном памяти.

Область ОЗУ пользователя:	
конфигурация 1	0...37777
конфигурация 2	0...67777
Область ОЗУ экранной памяти:	
конфигурация 1	40000...77777
конфигурация 2	70000...77777
Монитор и драйверы внешних устройств	100000...117777
Интерпретатор языка высокого уровня ФОКАЛ БК-0010	120000...137777
Свободное место для ПЗУ с программой пользователя	140000...157777
Мониторная система диагностики и контроля	160000...177577
Системные регистры	177600...177777

Быстродействие микроЭВМ «Электроника БК-0010» составляет около 300 тыс. операций типа «регистр — регистр» в секунду, потребляемая мощность не более 20 В·А, масса не более 4 кг.

4.2. Комплект учебной вычислительной техники на базе ДВК и микроЭВМ «Электроника БК-0010»

В состав комплекта учебной вычислительной техники на базе ДВК и микроЭВМ «Электроника БК-0010» (КУВТ ДВК-БК) входят:

рабочее место преподавателя (РМП) на базе ДВК «Электроника НЦ 80-20/2М» в полной комплектации с вмонтированными двумя платами шестиканальных контроллеров телеграфных каналов (ДВК-2МШ);

до 12 рабочих мест учащихся (РМУ) на базе микроЭВМ «Электроника БК-0010» с подключенным блоком интерфейса радиального последовательного ИРПС (БК-0010Ш);

соединительный кабель;

системные программные средства;

пакет прикладных программ.

В качестве видеомониторов на рабочих местах учащихся

могут использоваться любые бытовые телевизоры, имеющие выход «видео» для формирования черно-белого изображения или вход КЗС (красный — зеленый — синий) для формирования цветного изображения. На экране видеомонитора может отображаться алфавитно-цифровая и графическая информация в формате 512 × 256 точек при двух градациях яркости и 256 × 256 точек при четырех градациях яркости.

Интерфейс радиальный последовательный (ИРПС) выполнен в виде отдельного блока и подключается к разъему внутренней магистрали микроЭВМ. Блок ИРПС предназначен для связи микроЭВМ «Электроника БК-0010Ш» с рабочим местом преподавателя по асинхронному последовательному каналу. Основным элементом ИРПС является БИС К1801ВП1-035 (К1801ВП1-065), которая осуществляет прием последовательной посылки, преобразование ее в параллельный код и запись в буферный регистр, а также обратное преобразование. ИРПС осуществляет обмен информацией с каналом микроЭВМ с помощью четырех регистров: регистра состояния приемника; буферного регистра приемника; регистра состояния передатчика; буферного регистра передатчика.

Прерывание программы с выдачей адреса вектора прерывания может производиться как от приемника, так и от передатчика. Блок ИРПС может производить прием и передачу посылок формата 7 и 8 бит. Управление форматом посылки осуществляется имеющимися в блоке переключателями. Предусмотрена работа в режиме с паритетом по четности и нечетности. Скорость обмена лежит в диапазоне 50—9600 бод. Связь с каналом ИРПС организована по четырем двухпроводным линиям типа «токовая петля» через оптронную развязку. Две линии используются для приема и передачи информации и две — для приема и передачи готовности.

Структура сети КУВТ ДВК-БК соответствует структуре КУВТ на базе ДВК (локальная сеть радиального типа), в которой все функции управления и обмена программами сосредоточены на рабочем месте преподавателя. Периферийные устройства (НГМД, АЦПУ) являются общими для всех и находятся на РМП. Аппаратную поддержку локальной сети со стороны РМП обеспечивают две платы КТЛК, а со стороны РМУ — блок ИРПС. Связь между РМП и каждым РМУ обеспечивается по трем последовательным линиям. В КУВТ возможен режим параллельной работы преподавателя и уча-

щегося на микроЭВМ учащегося с одновременным отображением информации на мониторах рабочих мест преподавателя и учащегося.

4 3. Системное программное обеспечение КУВТ ДВК-БК

Системное программное обеспечение КУВТ ДВК-БК включает в себя соответствующие программы РМП и РМУ.

Системное программное обеспечение РМП состоит из средств, входящих в стандартный набор операционной системы ОС ДВК и разработанных специально для обслуживания сети КУВТ программ, функционирующих под управлением ОС ДВК. Как и в КУВТ на базе ДВК, в КУВТ ДВК-БК возможны три режима работы:

GET — для копирования программ в абсолютном формате из микроЭВМ «Электроника БК-0010Ш» в файл на ДВК-2МШ»;

PUT — для параллельной загрузки программ в абсолютном формате из файла на ДВК-2МШ одновременно с нескольких микроЭВМ «Электроника БК-0010Ш» по выбору преподавателя;

NETP — для контроля с РМП за работой учащегося на РМУ и распечатки программ из РМУ на устройстве печати ДВК-2МШ.

Программа NETP позволяет преподавателю, находящемуся на РМП, вмешиваться в работу учащегося за любым РМУ, проверять его действия, вносить необходимые коррективы, распечатывать программу из РМУ на устройстве печати, которое имеется только на РМП, и возвращать управление на терминал РМУ. Причем все действия преподавателя по корректировке учащийся непосредственно наблюдает на самом мониторе.

Системное программное обеспечение РМУ реализовано в БИС ПЗУ. Оно состоит из ПЗУ монитора и драйверов; тестового ПЗУ; ПЗУ интерпретатора языка ФОКАЛ.

ПЗУ монитора и драйверов и тестовое ПЗУ жестко закреплены на плате и закрыты от свободного доступа пользователя. ПЗУ языка ФОКАЛ находится в отсеке пользователя и является сменным. Так же находится разъем под второе сменное ПЗУ. Предусмотрена возможность замены ПЗУ с языком ФОКАЛ на ПЗУ с любым другим языком программирования, например БЕЙСИК. Монитор позволяет: вызвать один из трех режимов тести-

рования; вызвать интерпретатор языка ФОКАЛ; осуществить загрузку с магнитной ленты программ, написанных в машинных кодах, и их запуск.

МикроЭВМ «Электроника БК-0010Ш» имеет: драйвер клавиатуры; драйвер бытового телевизора; драйвер бытового кассетного магнитофона.

Программы-драйверы являются связующим звеном между программой языка ФОКАЛ и аппаратными средствами микроЭВМ. Обращение к драйверам осуществляется с помощью командных прерываний ЕМТ с заданным аргументом, определяющим заданную функцию.

Тестовое ПЗУ содержит в себе мониторную систему диагностики, в состав которой входят три программных модуля: контроля (ТК); самодиагностики (ТС); внешней диагностики (ТД).

Модуль контроля предназначен для контроля функционирования микроЭВМ, остальные модули используются при ее наладке. Модуль ТС позволяет вводить и запускать на выполнение программы в машинных кодах.

4.4. Подготовка к работе и обеспечение функционирования КУВТ ДВК-БК

Управление всеми функциями обмена программами в классе возложены на преподавателя, находящегося за главной ЭВМ (ДВК-2МШ). Для этого в его распоряжении имеются специальные системные программы, хранящиеся на гибком магнитном диске. Имеется возможность работы в трех режимах:

режим передачи программы из ДВК-2МШ в несколько РМУ («Электроника БК-0010Ш»);

режим приема программы из РМУ в ДВК-2МШ;

режим взаимодействия с выбранным РМУ для просмотра, корректировки и запуска на выполнение программы обучаемого.

Для включения РМУ необходимо на каждом рабочем месте произвести следующие действия:

1) включить тумблер питания «Сеть» видеомонитора (телевизора);

2) включить тумблер питания «Сеть» блока питания;

3) на экране видеомонитора должно появиться сообщение:

```
?00 AT 00.0
Готовность к работе *
```

Наличие звездочки указывает на готовность системы к выполнению любых действий пользователя.

Для включения РМП необходимо произвести следующие действия:

1) включить тумблер питания «Сеть» на задней панели блока сопряжения;

2) включить кнопку питания «Сеть» на передней панели дисплея;

3) нажать кнопки на передней панели блока сопряжения: ПИТ, ПУСК, ПР. На экране появится сообщение:

```
160000
@
```

4) на задней панели дисковода переключатель «Сеть» поставить в положение ВКЛ (на передней панели дисковода загорится лампочка индикации). На дисковод «0» поставить дискетту с операционной системой ОС ДВК;

5) с помощью клавиатуры задать адрес дисковода с системной дискеттой (в нашем случае «0»):

```
@ X0
```

Произойдет загрузка операционной системы с дискетты и на экране появится сообщение:

```
ОС ДВК V04.01
. ASS MX1:ДК:
.
```

Рассмотрим действия операторов (учащихся и преподавателя) для осуществления различных режимов работы КУВТ ДВК-БК*.

Режим передачи программ в РМУ из РМП. Рассмотрим пример одновременной передачи программы, написанной на языке ФОКАЛ с именем TASK. FOC, в несколько РМУ под номерами: 1, 5, 8, 10, 12. Ниже представлен протокол обмена (подчеркнутые символы и слова выводит машина):

```
. R PUT <ВК>
ПАРАЛЛЕЛЬНЫЙ ЗАГРУЗЧИК V01.00
ВВЕДИТЕ ЧЕРЕЗ «,» НОМЕРА БК-0010Ш
ИСПРАВЛЕНИЯ ВНОСИТЕ <  >, КОНЕЦ  ВВОДА — <ВК>
1, 5, 8, 10, 12 <-К>
УСТАНОВЛЕНА СВЯЗЬ С БК-0010Ш #1
УСТАНОВЛЕНА СВЯЗЬ С БК-0010Ш # 5
```

---

* Программы обмена разработал канд. техн. наук Т. А. Куправа.

```
УСТАНОВЛЕНА СВЯЗЬ С БК-0010Ш # 8
УСТАНОВЛЕНА СВЯЗЬ С БК-0010Ш # 10
УСТАНОВЛЕНА СВЯЗЬ С БК-0010Ш # 12
ЗАГРУЖАТЬ (<Д> ИЛИ <Н>) <ВК>
Д <ВК>
ВВЕДИТЕ ИМЯ ФАЙЛА И <ВК>
* TASK ВК
XX ВЫДАН ФАЙЛ
ВВЕДИТЕ ЧЕРЕЗ «,» НОМЕРА БК-0010Ш
ИСПРАВЛЕНИЯ ВНОСИТЕ <ЗБ>, КОНЕЦ ВВОДА — <ВК>
```

где XX — номер блока, передаваемого в данный момент по каналу связи.

После завершения передачи файла можно загрузить любую другую программу в эти же или другие БК-0010Ш. Для выхода из режима передачи файла нужно нажать клавишу <ПС>, произойдет возврат в систему.

Примечание 1. Если на экране преподавателя выдается сообщение

```
НЕТ СВЯЗИ С БК-0010Ш # X
```

необходимо проверить, находится ли соответствующий БК-0010Ш в режиме готовности.

Примечание 2. Если на экране появилось сообщение

```
ОШИБКА КОНТРОЛЬНОЙ СУММЫ
```

необходимо повторить процедуру передачи файла.

Режим приема программ из РМУ в РМП. Программа пользователя, составленная на языке ФОКАЛ и находящаяся в памяти «Электроника БК-0010Ш», может быть записана на дискетту главной ЭВМ. Для этих целей служит специальная программа GET.SAV. Ниже представлен протокол записи файла с именем TASK.FOC из РМУ под номером 4 в главную ЭВМ:

```
R GET <ВК>
ВВЕДИТЕ НОМЕР БК-0010Ш
4 <ВК> ВВЕДИТЕ ИМЯ ФАЙЛА И ЗНАК =
* TASK = <ВК>
БК-0010Ш В РЕЖИМЕ ПЕРЕДАЧИ
XX ПРИНЯТ ФАЙЛ TASK =
ВВЕДИТЕ НОМЕР БК-0010Ш
```

где XX — номер блока, передаваемый в данный момент по каналу связи.

Файл с именем TASK.FOC перепишется из ДВК-1МШ

на рабочую дискетту ДВК-2МШ. Для выхода из режима приема программ необходимо нажать клавишу <ПС>, произойдет возврат в систему.

Примечание 1. Если не выдалось сообщение о записи файла, посмотрите, не пропустили ли вы знак = после имени файла.

Примечание 2. Если во время записи программы пользователя на дискетту появилось сообщение

СИСТЕМНАЯ ОШИБКА

следует проверить наличие свободных блоков на рабочей дискетте и в случае необходимости произвести сжатие библиотеки или удалить ненужные файлы.

Примечание 3. При появлении сообщения

ОШИБКА КОНТРОЛЬНОЙ СУММЫ

процедуру записи повторить.

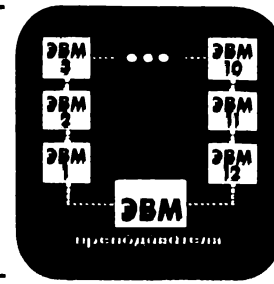
Программа взаимодействия РМП с РМУ. Находясь за главной ЭВМ, преподаватель имеет возможность подключить ЭВМ учащегося к ДВК-2МШ для просмотра, корректировки, выполнения и вывода на печать программы, находящейся в памяти микроЭВМ «Электроника БК-0010Ш». При этом вывод информации на экран монитора РМУ не блокируется на период связи. Ниже представлен протокол программы взаимодействия:

```
RNETP <ВК>
ПРОГРАММА ВЗАИМОДЕЙСТВИЯ ДВК-2МШ С БК-0010Ш
ВОЗВРАТ В СИСТЕМУ ПО КЛАВИШЕ <ПС>
ВВЕДИТЕ НОМЕР БК-0010Ш
4 <ВК> СВЯЗЬ С БК-0010Ш УСТАНОВЛЕНА
НУЖЕН ЛИ ПАРАЛЛЕЛЬНЫЙ ВЫВОД НА УСТРОЙСТВО
ПЕЧАТИ (ДА, НЕТ)
```

Если вводится ответ ДА <ВК>, то происходит вывод информации на экраны мониторов РМП и РМУ, а также на устройство печати.

Язык ФОКАЛ для микроЭВМ «Электроника БК-0010» имеет некоторые отличия от языка ФОКАЛ для ДВК. Так, в состав операторов языка для бытового компьютера введен новый оператор LIBRARY, позволяющий записывать, хранить и считывать файлы, используя бытовой магнитофон «Электроника». Для работы с портом ввода—вывода применяется оператор FP. Графических функций всего три: установка курсора, формирование точки и формирование вектора.

глава 5
Учебный вычислительный комплекс на базе персональной микроЭВМ «Электроника УК НЦ»



Уже на ранней стадии разработка и проектирование учебного вычислительного комплекса велись с учетом требований учебного процесса. Комплекс предназначен для использования в общеобразовательных школах, средних профессионально-технических училищах, межшкольных учебно-производственных комбинатах, педагогических учебных заведениях, институтах усовершенствования учителей, высших и средних специальных учебных заведениях и на факультетах повышения квалификации преподавателей. Учебный вычислительный комплекс представляет собой совокупность технических и программных средств, ориентированных на обучение и практическую работу по основам информатики и вычислительной техники, разработку обучающих программ по другим предметам, выработку навыков применения вычислительной техники в различных областях учебной, производственной, научной и экономической деятельности.

5.1. Персональная микроЭВМ «Электроника УК НЦ»

МикроЭВМ «Электроника УК НЦ» (учебный компьютер Научного центра [18]) представляет собой конструкцию настольного исполнения, состоящую из одноплатной микроЭВМ, клавиатуры и встроенного источника питания. На рабочих местах преподавателя и учащегося используются разные модификации микроЭВМ, соответственно «Электроника МС 0511.02» и «Электроника МС 0511.01». Они отличаются исполнением одноплатной микроЭВМ — на рабочем месте учащегося находится более простой вариант, в котором отсутствуют устройства управления (контроллеры) накопителем на гибких магнитных дисках, выводом на печать и графопостроителем.

Состав технических средств рабочего места преподавателя показан на рис. 5.1. В него входят собственно микроЭВМ «Электроника МС 0511.02», видеомонитор, построенный на базе стандартного телевизионного приемника «Юность 406», сдвоенный накопитель на гибких

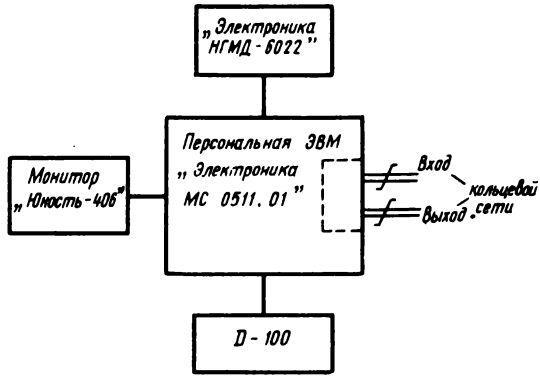


Рис. 5.1. Состав технических средств рабочего места преподавателя

магнитных дисках «Электроника НГМД-6022» и устройство печати D-100.

Видеомонитор «Юность 406» относится к устройствам отображения информации со средней разрешающей способностью. Для графических изображений он представляет поле размером 256×512 точек. В целях обеспечения безопасности видеомонитор и микроЭВМ (устройства, находящиеся на рабочих местах учащихся) питаются напряжением переменного тока 42 В, частотой 50 Гц.

Накопитель микроЭВМ «Электроника НГМД 6022» используется для хранения системных, прикладных и обучающих программ. Объем доступной информации, находящейся на двух дискетах, составляет приблизительно 0,5 Мбайт.

Вывод алфавитно-цифровой и графической информации осуществляется с помощью бесшумного печатающего устройства мозаичного типа.

Основные технические характеристики устройства печати D-100:

Скорость печати	— 100 зн/с
Количество знаков в строке	— 80/132
Матрица знака	— 9 × 7
Код	— КОИ-8
Размер буфера	— 2 К байт
Набор знаков	— До 256
Интерфейс	— ИРПР
Логические сигналы	— «ТТЛ-логика»
Питание	— 220 В, 50 Гц
Потребляемая мощность	— 120 В · А
Размеры	— 410×328× ×130 мм
Масса	— 12 кг

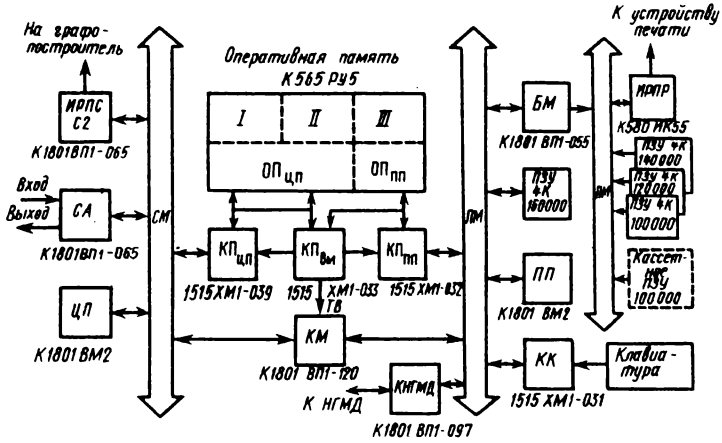


Рис. 5.2. Структурная схема микроЭВМ «Электроника УК НЦ»

Для обеспечения работы микроЭВМ в составе локальной сети она имеет сетевой адаптер СА, конструктивно выполненный в виде отдельной печатной платы. В случае автономного использования микроЭВМ (как персональной машины) сетевой адаптер может не поставляться.

Рассмотрим в общих чертах структурную схему микро-ЭВМ «Электроника УК НЦ», представленную на рис. 5.2*. Характерной особенностью приведенной структуры является наличие двух процессоров — центрального (ЦП) и периферийного (ПП), а также соответствующих магистралей — системной (СМ) и периферийной (ПМ). Оба процессора идентичны и выполнены в виде БИС К1801ВМ2.

Общий объем оперативной памяти микроЭВМ составляет 192 К байт. Она реализована на 24 микросхемах типа К565РУ5. Каждая микросхема представляет собой ОЗУ динамического типа с организацией 64 К$\times$1 бит. Память включает в себя три области: оперативную память центрального процессора объемом 64 К байт ($ОП_{цп}$), оперативную память периферийного процессора объемом 32 К байт ($ОП_{пп}$) и видеопамять монитора объемом 96 К байт, которая в свою очередь содержит

* Материалы представили разработчики микроЭВМ инженеры А. Н. Полосин, Н. Г. Карпинский, А. И. Половянюк, М. И. Дябин, И. О. Лозовой.

разделы I, II и III, каждый объемом по 32 К байт.

В зависимости от режима работы разделы памяти I, II и III могут использоваться либо для управления соответственно красным, зеленым и синим прожекторами цветной электронно-лучевой трубки монитора, либо в случае монохромного монитора раздел I используется в качестве видеопамяти, а другие два — в качестве «электронного диска».

Для управления памятью используются контроллеры, реализованные на основе БИС универсальных вентильных матриц 1515ХМ1-039, 033 и 032. На структурной схеме им соответствуют обозначения: $КП_{цп}$ — контроллер памяти центрального процессора, $КП_{вм}$ — контроллер памяти видеомонитора; $КП_{пп}$ — контроллер памяти периферийного процессора.

Связь между СМ и ПМ обеспечивается посредством контроллера межмагистральной связи (КМ), реализованного на универсальной вентильной матрице К1801ВП1-120. Со стороны СМ контроллер КМ представляет три независимых канала: стандартный канал терминала с адресами регистров на системной магистрали

177560,
177562,
177564,
177566,

канал управления контроллером накопителя на гибких магнитных дисках с адресами

176660
176662,
176664,
176666,

канал вывода информации с адресами

176674,
176676.

К внутренней магистрали микроЭВМ подсоединены контроллер управления накопителем на гибких магнитных дисках КНГМД (микросхема К1801ВП1-097), контроллер клавиатуры КК (микросхема 1515ХМ1-031) и системное ПЗУ (микросхема К1801РЕ2) с адресным пространством на внутренней магистрали 160000—177776. К этой же магистрали через буферную микросхему К1801ВП1-055 (БМ) подсоединены контроллер

байтового параллельного интерфейса для радиального подключения ИРПР, реализованный в виде микросхемы К580ИК55, и три БИС ПЗУ типа К1801РЕ2, каждое емкостью 4 К слов и соответственно с начальными адресами на внутренней магистрали 100000, 120000, 140000. Указанные БИС ПЗУ используются для системных целей — в них хранятся начальные загрузчики, различные таблицы, образцы алфавитно-цифровых символов, графические примитивы и т. п. Кроме того, имеется возможность подключения внешнего кассетного ПЗУ объемом 2×24 К байт, доступ к которому реализуется через «окно» на внутренней магистрали с адресами 100000—117776. Отметим важность этого момента: из кассетного ПЗУ можно загружать интерпретаторы диалоговых языков программирования БЕЙСИК, ФОКАЛ, РАПИРА и др., пакеты обучающих и игровых программ, текстовые редакторы, мониторы операционных систем и т. п.

К системной магистрали микроЭВМ могут дополнительно подключаться еще два устройства: сетевой адаптер (СА) для включения микроЭВМ в локальную кольцевую сеть и байтовый последовательный интерфейс (ИРПС) с выходом на стык С2 для подключения графопостроителя.

Следует отметить еще одну важную особенность рассматриваемой структуры. В отличие от ДВК-1М и микроЭВМ «Электроника БК-0010», в которых часть их адресного пространства занята интерпретатором языка программирования, памятью экрана, монитором, драйверами внешних устройств, адресное пространство центрального процессора микроЭВМ «Электроника УК НЦ» с адресами 000000—157776 соответствует ячейкам оперативной памяти, а адреса 160000—177776 — страница ввода—вывода, т. е. учебный компьютер использует стандартные соглашения, принятые для семейства популярных машин типа СМ4. Это позволяет без всяких изменений пользоваться широко распространенными операционными системами (RT-11, РАФОС, ОС ДВК и др.), а также богатым набором готовых прикладных программ.

Периферийный процессор вместе с расположенными в его адресном пространстве оперативной памятью 32 К байт и ПЗУ 32 К байт можно рассматривать в качестве интеллектуального контроллера, управляющего видеопамятью монитора, клавиатурой, НГМД, устройством печати и кассетным ПЗУ.

По замыслу разработчиков микроЭВМ, принятые решения обеспечат гибкость и широту использования ЭВМ, а также достаточно долгий срок ее жизни до морального устарения.

Основные технические характеристики микроЭВМ «Электроника МС 0511.01/02»:

Тип микропроцессора	— КП1801ВМ2А 2 шт.
Быстродействие	— 600 тыс.опер /с
Общий объем памяти	— 192 К байт
Объем памяти центрального процессора	— 64 К байт
Объем памяти периферийного процессора	— 32 К байт
Объем видеопамяти	— 96 К байт
Объем ПЗУ	— 32 К байт
Внешнее ПЗУ кассета	— 2 × 24 К байт
Возможность отображения алфавитно-цифровой информации	— 24 строки по 80 символов
Возможность отображения графической информации	— 640 × 288 точек
Клавиатура	— стандарт MSX
Система команд	— «Электроника-60»
Операционные системы	— RT-11, ОС ДВК, РАФОС
Языки программирования	— БЕЙСИК, ФОКАЛ, ПАСКАЛЬ, ФОРТРАН
Элементная база	— серии КП1801, КР1515, КР565, КР580, К555, К155, К531, транзисторы, резисторы и т. п.
Питание	— однофазная сеть 42 В, 50 Гц
Потребляемая мощность, не	— 15 В· А
Размеры	— 448×290×80 мм
Масса, не более	— 7 кг

5.2. Организация учебного класса

В состав учебного вычислительного комплекса входит одно рабочее место преподавателя и двенадцать рабочих мест учащихся. Рабочее место учащегося построено на основе персональной микроЭВМ «Электроника МС 0511.02». В качестве периферийного устройства используется видеомонитор «Юность 406». Предусматривается также возможность подключения кассетного магнитофона.

Все двенадцать микроЭВМ, расположенные на рабочих местах учащихся, а также микроЭВМ преподавателя объединены в локальную кольцевую сеть [2, 18]. На рис. 5.3 приведена схема соединения. В каждой ЭВМ

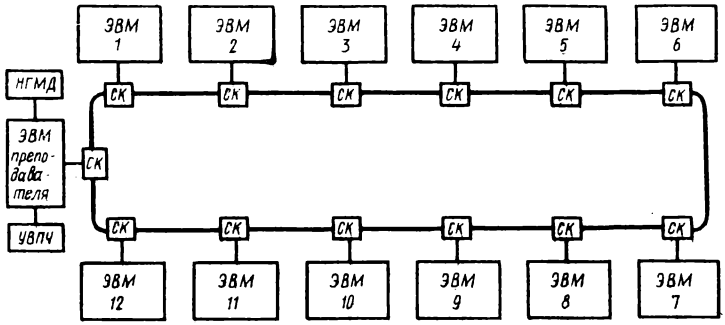


Рис. 5.3. Схема учебного класса на основе микроЭВМ «Электроника УК НЦ»

имеется сетевой адаптер СА*, имеющий один вход и один выход. Адаптер подключается к кольцу с помощью соединительной колодки (СК). Физически кольцо представляет собой витую пару проводов, проложенную между всеми соединительными колодками. Способ передачи информации в сети последовательный, асинхронный с контролем байтов на четность, скорость передачи 57 000 бод.

Использование кольцевой локальной сети для объединения ЭВМ позволяет уменьшить зависимость сети от центрального узла. В рассмотренных ранее учебных классах на основе микроЭВМ ДВК-1 и БК-0010 применялась звездообразная локальная сеть, в качестве центрального узла которых служила микроЭВМ ДВК-2. Количество подключаемых периферийных ЭВМ к центральному узлу ограничивалось числом последовательных каналов в контроллере главной ЭВМ. Поскольку для организации связи с двенадцатью периферийными ЭВМ нужны две платы КТлК-6 (каждая на шесть последовательных каналов), все розетки на общей магистрали блока сопряжения оказались занятыми: плата микроЭВМ, плюс плата контроллера НГМД, плюс две платы КТлК-6.

Другими словами, в звездообразной локальной сети подключение дополнительной периферийной микроЭВМ требует включения в состав центрального узла нового канала и прокладки кабеля связи (четыре витых пары

* Сетевой адаптер разработали канд. техн. наук В. И. Белильников и Ю. И. Волков, инженер В. Г. Шапкин.

для классов, описанных в предыдущих главах).

В этом отношении кольцевая локальная сеть имеет неоспоримые преимущества — во всех микроЭВМ, включая и ЭВМ преподавателя, сетевые адаптеры имеют одинаковую конструкцию, связь между ними осуществляется по единственной витой паре проводов и добавление новых машин не представляет затруднений. Фактически для подключения новой микроЭВМ нужно разорвать в удобном месте кольцо и впаять дополнительную соединительную колодку с проводами соответствующей длины.

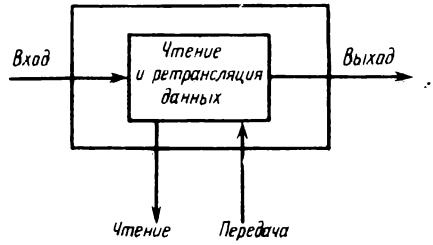


Рис. 5.4. Схема узла кольцевой локальной сети

Рассмотрим общие принципы функционирования кольцевой локальной сети. Каждый узел локальной сети можно условно представить в виде, показанном на рис. 5.4. Информация по кольцу передается последовательно в одном направлении от одного узла к другому в виде пакета данных. Логика узла обеспечивает следующие функции: ретрансляцию данных с входа на выход с одновременным чтением либо чтение поступающих на вход данных и одновременно передачу «своих» данных на выход. Пакеты данных, передаваемые по кольцу, постоянно регенерируются, проходя через каждый узел. Они циркулируют до тех пор, пока не будут удалены узлом — отправителем информации.

Существует несколько способов доступа к кольцу: метод вставки регистра или буфера, метод тактируемого доступа, метод передачи маркера [17]. Для учебного класса выбран метод передачи маркера, или, как его иногда называют, «эстафетный метод передачи». Для него характерны несложная аппаратная реализация и обеспечение равноправного доступа к кольцу. Маркер представляет собой легко распознаваемую короткую последовательность символов, которая непрерывно циркулирует по кольцу. Если устройство имеет данные, подлежащие передаче, то оно вынуждено ждать до тех пор, пока предшествующее устройство не вышлет ему маркер. Получив маркер, рассматриваемое устройство временно «поглощает» его и выдает в кольцо свои дан-

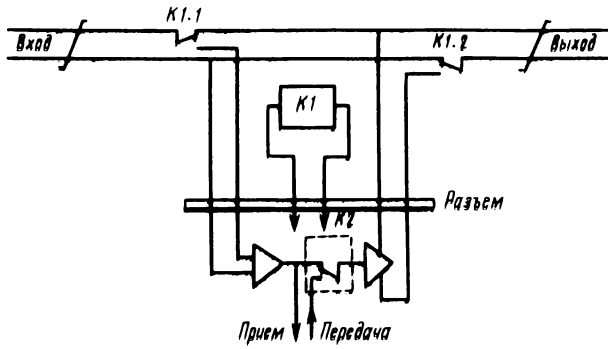


Рис. 5.5. Функциональная схема узла кольцевой локальной сети

ные. Пакет данных движется от одного узла к другому и может быть прочитан тем узлом, которому предназначалась передача. При этом узел-приемник в прошедшем через него пакете данных устанавливает признаки благополучного или неблагополучного приема пакета данных. Когда передаваемый пакет данных по кольцу снова вернется в узел-отправитель, он проверяется на признаки. Если передача была без ошибок, то пакет данных удаляется из кольца устройством-отправителем (принимается в его приемный регистр) и устройство-отправитель возвращает маркер в кольцо. Если передача сопровождалась ошибками, то устройство-отправитель может несколько раз повторить передачу.

Основные трудности в кольцевой локальной сети с передачей маркера возникают в двух случаях: при потере маркера или если устройство-отправитель не стирает свой пакет. Возможно также появление нескольких маркеров. С целью ликвидации указанных ситуаций в разработанном варианте локальной сети предусмотрены соответствующие программно-аппаратные средства.

Включение или выключение любой микроЭВМ, находящейся на рабочем месте учащегося, не должно нарушать целостности и работоспособности кольца. С этой целью в каждой соединительной колодке установлено электромагнитное реле К1, обмотка которого запитывается от источника питания микроЭВМ (рис. 5.5). При выключении конкретной микроЭВМ обмотка соответствующего реле обесточивается и группы контактов К1.1 и К1.2 переключаются таким образом, что сетевой адап-

тер этой микроЭВМ исключается из кольца. Средства обнаружения и исправления ошибок позволяют продолжить функционирование кольца, несмотря на возникающий переходный процесс при включении или выключении микроЭВМ. На этом же рисунке показан условно электронный ключ К2. Функции его следующие: когда ключ К2 находится в верхнем положении, происходит ретрансляция данных с одновременным чтением; при переключении ключа в нижнее положение данные из кольца поступают в приемный регистр (удаляются из кольца) и одновременно возможна передача новых данных в кольцо.

Заканчивая рассмотрение материалов по кольцевой сети, отметим ее сильные и слабые стороны.

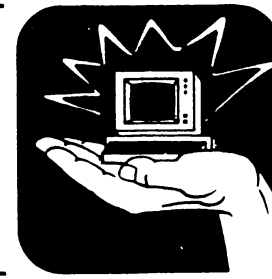
Кольцевая сеть имеет следующие преимущества: простая и недорогая связь между устройствами; легкость подключения новых устройств; возможность передачи данных одновременно всем устройствам («широковещательная передача»); просто осуществляется контроль ошибок при передаче; равные права доступа всех устройств к кольцу; гарантированное время передачи данных.

Однако у кольцевой сети есть и недостатки: надежность функционирования сети зависит от надежности ее каждого узла; слабым местом сети является множество последовательно соединенных пар электромеханических контактов реле на пути следования сигнала; большие накладные расходы по процессорному времени и памяти в каждом устройстве, связанные с необходимостью поддержки работоспособности сети; довольно сложная процедура начальной загрузки кольца (приведение в рабочее состояние при включении).

Последние два недостатка будут преодолены после окончания разработки «интеллектуального» сетевого адаптера, в состав которого входят собственный процессор, постоянная и оперативная память.

Заключение

Перспективы использования ЭВМ в учебном процессе



Какими бы путями ни шло дальнейшее развитие ЭВМ, используемых в учебном процессе, по-видимому, на долгий срок останутся действенными следующие требования, определяемые самой сущностью процесса обучения:

ясность — в одинаковых ситуациях ЭВМ проявляет одинаковое поведение, неожиданные эффекты отсутствуют;

простота обучения — пользователь может обучаться в процессе работы на ЭВМ;

легкость пользования — небольшое число активно используемых операторов языка программирования, хорошая диагностика ошибок, простота внесения исправлений и отладки программы, малое время ответа;

надежность — ЭВМ готова к работе всегда, когда требуется, сбои происходят редко.

Фактором, сдерживающим возможность более эффективного использования ЭВМ в учебном процессе, является низкая скорость обмена информацией в существующих локальных сетях. Желательно ее повысить в ближайшее время по крайней мере до 0,5—1 Мбод. Решение этой задачи должно базироваться на современных достижениях технологии БИС, для того чтобы обеспечить высокую надежность функционирования при приемлемой стоимости и габаритах.

Другим существенным ограничением является недостаточная надежность накопителей на гибких магнитных дисках, во многом определяемая высокой нагрузкой при работе в составе локальной сети. Указанная проблема может быть решена путем установки в машине преподавателя плат «электронных дисков». Основу электронного диска представляют микросхемы БИС динамической памяти. Для пользователя логическая организация работы с электронными дисками ничем не отличается от работы с накопителями на магнитных дисках. В условиях постоянного снижения стоимости БИС памяти такое решение представляется экономически оправданным. Заметим, что в одноплатных микроЭВМ нового поколения, выполненных на основе микропроцессора К1801ВМ3, эта проблема решается автоматически. Наличие диспетчера и оперативной памяти объемом 1 Мбайт на той же плате позволяет организовать электронный диск, используя стандартный драйвер операционной системы.

Приложения

Приложение 1

Диагностические сообщения ФОКАЛа

Код ошибки	Текстовые сообщения о причине ошибки
00	Готовность к работе
01	Неправильный номер строки
02	Неправильное имя функции или переменной
03	Непарные скобки
04	Неправильная команда
05	Несуществующий номер строки
06	Несуществующий номер группы или строки в операторе DO
07	Неправильный формат оператора
08	Двойной или пропущенный символ в выражении
09	Переполнен стек
10	Переполнена память текстом программ
11	Нет места для переменных в памяти
12	Порядок больше $E \pm 38$
13	Запрещенный адрес шины в функциях FX
14	Деление на $\emptyset$
15	Недопустимый показатель степени
16	Переполнена строка в операторе ASK
17	Корень квадратный из отрицательного числа
18	Переполнена строка
19	Логарифм нуля или отрицательного числа
20	В функциях FASIN или FACOS аргумент по модулю > 1
21	Несуществующее устройство
22	Ошибка ввода
23	Ввод: ?
24	Вывод: ?
25	ЭВМ исправна
26	ЭВМ неисправна

Встроенные функции ФОКАЛа

1. Математические функции

Функция	Название функции
FSIN (арг.)	Синус (аргумент в радианах)
FCOS (арг.)	Косинус (аргумент в радианах)
FTAN (арг.)	Тангенс (аргумент в радианах)
FASIN (арг.)	Арксинус (результат в радианах)
FACOS (арг.)	Арккосинус (результат в радианах)
FATAN (арг.)	Арктангенс (результат в радианах)
FLOG (арг.)	Натуральный логарифм
FLOG10 (арг.)	Десятичный логарифм
FEXP (арг.)	Показательная функция
FSGN (арг.)	Знаковая часть числа
FITR (арг.)	Целая часть числа
FABS (арг.)	Абсолютная величина числа
FSQT (арг.)	Корень квадратный
FSBR (группа, арг.)	Функция, программируемая показателем
FRAN ( )	Генератор случайных чисел (от —1 до +1)

2. Функции управления вводом—выводом

Функция	Название функции
FCHR (арг., арг., ...)	Ввод—вывод символов
FX (функция, адрес, данные)	Функция доступна к общей шине

3. Функции управления таймером

Функция	Название функции
FZCLK ( )	Обнуление таймера
FCLK ( )	Считывание таймера

4. Графические функции

Функция	Название функции
FI (код)	Начальная установка экрана
FM (код)	Установка режима вывода точек
FP (x, y)	Вывод точки в абсолютных координатах
FD (Δx, Δy)	Вывод точки в относительных координатах

Функция	Название функции
FA (x, y)	Установка графического курсора в абсолютных координатах
FC (Δx, Δy)	Установка графического курсора в относительных координатах
FV ($x_к$, $y_к$ [, $x_и$, $y_и$])	Вывод отрезка вектора в абсолютных координатах
FL (Δx, Δy [, $x_и$, $y_и$])	Вывод отрезка вектора в относительных координатах
FR (R [, $x_ц$, $y_ц$])	Вывод окружности радиусом
FS (L, H [, x, y])	Вывод прямоугольника (L — длина, H — высота)
FF (x, y)	Закрашивание замкнутого контура

Приложение 3

Таблица кодов символов дисплея 15ИЭ-00-013

Код		Символ	Код		Символ
восьмеричный	десятичный		восьмеричный	десятичный	
040	32	␣	060	48	0
041	33	!	061	49	1
042	34	"	062	50	2
043	35	#	063	51	3
044	36	¤	064	52	4
045	37	%	065	53	5
046	38	&	066	54	6
047	39	'	067	55	7
050	40	(	070	56	8
051	41	)	071	57	9
052	42	*	072	58	:
053	43	+	073	59	;
054	44	,	074	60	<
055	45	—	075	61	=
056	46	.	076	62	>
057	47	/	077	63	?

Код		Символ		Порядковый номер
восьмеричный	десятичный	регистр ЛАТ	регистр РУС	
100	64	@	ю	
101	65	A	а	1
102	66	B	б	2
103	67	C	ц	3
104	68	D	д	4
105	69	E	е	5
106	70	F	ф	6
107	71	G	г	7
110	72	H	х	8
111	73	I	и	9
112	74	J	й	10
113	75	K	к	11
114	76	L	л	12
115	77	M	м	13
116	78	N	н	14
117	79	O	о	15
112	80	P	п	16
121	81	Q	р	17
122	82	R	р	18
123	83	S	с	19
124	84	T	т	20
125	85	U	у	21
126	86	V	ж	22
127	87	W	в	23
130	88	X	ь	24
131	89	Y	ы	25
132	90	Z	з	26
133	91	[	ш	8
134	92	\	э	—
135	93	]	щ	—
136	94	¬	ч	—
137	95	\	ъ	—
140	96	—	Ю	0
141	97	a	А	1
142	98	b	Б	2
143	99	c	Ц	3

Код		Символ		Порядковый номер
восьмеричный	десятичный	регистр ЛАТ	регистр РУС	
144	110	d	Д	4
145	101	e	Е	5
146	102	f	Ф	6
147	103	g	Г	7
150	104	h	Х	8
151	105	i	И	9
152	106	j	Й	10
153	107	k	К	11
154	108	l	Л	12
155	109	m	М	13
156	110	n	Н	14
157	111	o	О	15
160	112	p	П	16
161	113	q	Я	17
162	114	r	Р	18
163	115	s	С	19
164	116	t	Т	20
165	117	u	У	21
166	118	v	Ж	22
167	119	w	В	23
170	120	x	Ь	24
171	121	y	Ы	25
172	122	z	З	26
173	123	{	Ш	27
174	124	|	Э	28
175	125	}	Щ	29
176	126	—	Ч	30
177	127	■	■	

Приложение 4

Система команд № 1 дисплея 15ИЭ-00-013

Код		Символ	Функция
восьмеричный	десятичный		
007	7	—	Звонок
010	8	■	Перемещение курсора в начало страницы

Код		Символ	Функция
восьмеричный	десятичный		
012	10	ПС	Перемещение курсора на строку вниз
013	11	СТС	Стирание текста от позиции курсора до конца строки
014	12	СБР	Сброс (очистка) экрана
015	13	ВК	Возврат курсора в начало текущей строки
016	14	РУС	Русский шрифт
017	15	ЛАТ	Латинский шрифт
022	18	≜	Сдвиг текста вверх на одну строку
023	19	↦	Сдвиг текста вправо на одну позицию от позиции курсора
024	20	↤	Сдвиг текста влево на одну позицию от позиции курсора
025	21	↵	Перемещение курсора в начало следующей строки
031	25	→	Перемещение курсора на одну позицию вправо по строке
036	26	←	Перемещение курсора на одну позицию влево по строке
034	28	↑	Перемещение курсора на одну строку вверх
035	29	↓	Перемещение курсора на одну строку вниз
037	31	СТР	Стирание текста от позиции курсора до конца страницы

Приложение 5

Внутренние коды интерпретатора ФОКАЛА

Восьмеричный код	Символ	Восьмеричный код	Символ
200	⌴	211	)
201	+	212	|
202	−	213	>
203	/	214	,
204	*	215	;
205	⌟	216	<ВК>
206	(	217	=
207	[	220	<РУС>
210	<	221	<ЛАТ>

Некоторые особенности работы учебной локальной сети и интерпретирующей системы ФОКАЛ в варианте загрузки монитора операционной системы в периферийные ЭВМ

МикроЭВМ типа ДВК прочно удерживают ведущие позиции среди отечественных профессиональных микроЭВМ. Выпуск их постоянно растет. Их популярность объясняется программной совместимостью с многочисленным семейством мини- и микроЭВМ ряда СМ ЭВМ, современной элементной базой, достаточно высокой надежностью работы. Объединение ЭВМ типа ДВК в локальные сети, особенно в условиях учебных заведений, позволяет существенно повысить эффективность их использования.

Для построения локальной сети в ЭВМ преподавателя вставляются две платы контроллера телеграфных каналов типа КТлК-6, а в каждую периферийную машину — полуплата сетевого адаптера (СА). Каждая периферийная машина связана с центральной восемью проводами (четыре токовые петли 20 мА). Удобство работы с локальной сетью во многом определяется скоростью передачи информации по каналам связи. Практика работы в учебном заведении показала, что скорость передачи 57 600 Бод (максимальная скорость, которая может быть обеспечена микросхемой асинхронного приемо-передатчика К1801ВП1-065), является приемлемой. При этом время ожидания окончания передачи даже для относительно больших файлов остается в разумных пределах — не больше нескольких десятков секунд. Для плат КТлК-6 требуется незначительная доработка, заключающаяся в замене микросхем оптопередатчиков АОТ110 на АОТ128 в каналах передачи данных. Микросхемы, вырабатывающие сигналы ГПР (готовность приемника), можно не менять.

Процедура загрузки монитора операционной системы в периферийную ЭВМ заключается в следующем. На центральной машине в оперативном разделе запускается программа поддержки сети, а на периферийной ЭВМ вводится следующая последовательность символов (подчеркнутые символы выводит ЭВМ):

@ 177556/000050 200 <ВК>

@ L

Если в качестве периферийной машины используется ДВК-1, то во второй строке перед буквой L набирается код 177550.

Спустя некоторое время на экране периферийной ЭВМ появится сообщение о загрузке монитора операционной системы RT11SJ. Далее вводится дата. Появление символа точки свидетельствует об окончании загрузки и возможности ввода с клавиатуры директив монитору. Все запросы периферийной ЭВМ обслуживаются через драйвер сети NT.SYS.

С помощью директивы DIR пользователь может распечатать на своем экране каталог системного или рабочего диска центральной ЭВМ, выбрать необходимую программу, загрузить и запустить ее на счет с помощью директивы R или RUN.

Возможность работы в среде стандартной операционной системы качественно меняет подход в использовании учебной локальной сети. Пожалуй главное — это отказ от диктата навязывания одного языка программирования. Преподаватель, учащийся, любой пользователь имеет право выбора языка в соответствии со своими наклонностями,

вкусами и привычками. Такой подход позволяет успешно продвигаться по пути совершенствования умения и навыков в работе с ЭВМ, начиная с простых языков программирования с ограниченными возможностями и переходя к более мощным и совершенным языкам.

Работа по адаптации интерпретатора ФОКАЛа к среде операционной системы RT11 была проведена автором совместно с выпускником МИЭТ инженером Вильсоном А. Л. При этом ставилась цель не только сохранить преемственность с версией интерпретатора, записанной в БИС ПЗУ К1801РЕ2-093, но и включить новые возможности, повышающие удобство работы с интерпретирующей системой.

В результате были написаны две версии. Версия ФОКАЛа VO1 (FOCAL1) может работать на любых моделях ДВК. Арифметический пакет подпрограмм, входящий в состав интерпретатора, обеспечивает шесть значащих цифр, причем последняя цифра может быть округлена. Версия ФОКАЛа VO2 (FOCAL2) использует набор команд расширенной арифметики EIS и FIS и предназначена для моделей ДВК-2 и ДВК-3 с микропроцессором К1801ВМ2. Пакет стандартной арифметики гарантирует семь значащих цифр и точную работу с целыми числами в диапазоне от —16777215 до +16777215. Версии несколько различаются по объему: FOCAL1 содержит 21 блок, а FOCAL2 — 24 блока. Кроме того, версии различаются форматом вывода чисел (целые числа во второй версии выводятся с десятичной точкой). В остальном они идентичны.

Ниже приводятся основные отличия новых версий ФОКАЛа в сравнении с описанием языка в гл. 2.

Проведена корректировка оператора MODIFY. Он теперь допускает изменение номера строки, что несомненно удобно для редактирования программы. Однако следует осторожно пользоваться этой возможностью, чтобы не переставить модифицируемую строку на место уже существующей, в результате чего последняя будет утрачена. Уменьшено также число функциональных клавиш, управляющих редактированием. Фактически можно пользоваться только тремя: ЗАБОЙ, СДВИГ КУРСОРА НА ОДНУ ПОЗИЦИЮ ВЛЕВО и СДВИГ КУРСОРА НА ОДНУ ПОЗИЦИЮ ВПРАВО. В этом случае стали ненужными две клавиши, использовавшиеся в прежней версии:

|→ — размыкание строки вправо от позиции курсора;

←| — смыкание строки влево от позиции курсора.

При вводе символьной информации часто возникает следующая ситуация. Пользователь нажал клавишу и тут же убедился, что символ неправильный. Для исправления ему достаточно нажать клавишу ЗАБОЙ, при этом только что введенный символ стирается, курсор смещается на одну позицию влево. Теперь можно ввести правильный символ. Если поставить курсор в середину некоторой строки, то последовательное нажатие клавиши ЗАБОЙ будет удалять символы, предшествующие курсору. Если вводить новые символы, то они будут вставляться слева от позиции курсора, а правая часть строки будет автоматически сдвигаться. Такое редактирование представляется более удобным и естественным, чем в прежней версии.

В новой версии вместо оператора VERIFY (запуск встроенной контрольной задачи) введен новый оператор VACANT, служащий для получения информации о количестве свободных слов, имеющихся в оперативной памяти ЭВМ. Подсчет ведется следующим образом. Сначала определяется количество слов, занятых под текст программы: вычисляется разность содержимых ячеек BUFF (конец текста) и BUFBEG (начало буфера текста). Затем подсчитывается количество переменных от адреса STAPTV (начало области переменных) до адреса

BOTTOM (конец интерпретатора). Полученное значение умножается на четыре, так как каждая переменная занимает в памяти четыре слова. Сложив число ячеек, занятых текстом программы, с числом ячеек, занятых переменными, получим общее число занятых ячеек. Зная общее количество ячеек памяти в интерпретаторе, выделяемое для текста и переменных (разность содержимых ячеек BOTTOM и BUFBEG), легко определить число свободных ячеек.

Для увеличения объема памяти для программ на ФОКАЛе можно использовать директиву монитора SET SL OFF. Если работа происходит вне локальной сети, то можно воспользоваться также и директивой SET USR SWAP. Убедиться в увеличении объема памяти можно с помощью оператора VACANT.

В разработанных версиях ФОКАЛа номера строк могут быть вплоть до номера 127.99 (в прежней версии номера строк с 100.1 до 127.9 могли идти только с шагом 0.1).

Приостанов вывода информации на экран осуществляется путем нажатия комбинаций клавиш СУ/S, а продолжение — СУ/Q (это стандартные действия в операционной системе).

Несколько изменен формат вывода сообщений ФОКАЛа, он стал более удобным для отечественного пользователя:

? СТОП XX.XX — <текст сообщения>

где XX.XX — номер строки косвенной команды, в которой произошла ошибка. В случае появления ошибки в прямой команде номер на печать не выводится.

В противоположность старой версии в новой комбинация клавиша СУ/С не приводит к удалению переменных из памяти.

В новых версиях ФОКАЛа после оператора выдачи переменных TYPE□ могут выполняться другие операторы (в прежних версиях происходил автоматический возврат в диалог).

В связи с тем, что в мониторе операционной системы существует свой таймер, функция считывания таймера FCLK ( ) переработана, а функция обнуления таймера FZCLK ( ) исключена.

Доработке подверглись графические функции вывода отрезков в абсолютных и относительных координатах FV и FL. В модернизированных функциях разрешается задавать координаты не только одной, двух, но и более точек. При построении ломаных и многоугольников запись получается более компактной. В качестве примера рассмотрим построение квадрата со сторонами 100 единиц в относительных координатах.

Для прежней версии:

```
1.1 X FI();  X FA(100,100)
1.2 X FL(0,100);X FL(100,0);X FL(0,—100);X FL(—100,0)
```

Для новых версий:

```
1.1 X FI() ; X FA(100,100)
1.2 X FL(0,100,100,0,0,—100,—100,0)
```

Во второй версии ФОКАЛа допустимо также и возведение в целую отрицательную степень. При этом показатель степени обязательно должен заключаться в скобки.

Существенной переработке подвергся датчик случайных чисел. Об этом говорилось в гл. 3 настоящей книги. Функция FRAN() вырабатывает псевдослучайную последовательность чисел в диапазоне от 0 до 1 с законом, близким к равномерному (в прежних версиях диапазон заключался в пределах от —1 до +1).

Введен новый оператор HAVE, который, как и оператор ASK, служит для ввода с клавиатуры или из файла значений переменных и элементов массивов. Но в отличие от оператора ASK он трактует вводимые буквы как имена переменных, что позволяет вводить арифметические выражения в более естественной форме, т. е. без лидирующего знака + или —. Оператор HAVE удобно использовать в обучающих программах для проверки правильности вводимых формул на запрос машины.

Особое место в новых версиях ФОКАЛа занимает группа операторов LIBRARY, предназначенная для работы с файлами.

ФОКАЛ допускает работу с двумя типами файлов — в абсолютном и текстовом форматах. Достоинством использования абсолютного формата является высокая скорость считывания и записи файлов (этот вопрос обсуждался в разделе 3.4 настоящей книги). Недостатком его является то обстоятельство, что может быть загружена только одна программа, при этом предыдущая программа стирается.

От указанного недостатка свободны файлы, представленные в текстовом формате. Однако следует помнить о том, что номера вновь вводимых строк не должны совпадать с номерами строк уже присутствующих в памяти. В противном случае они будут затерты (имеются в виду старые строки).

Большим удобством представления файлов в текстовом формате является возможность их обработки любым из стандартных редакторов, входящих в состав математического обеспечения ДВК. Становятся доступными такие средства, как поиск по образцу, запоминание одной или блоков строк с последующим их перемещением в другое место, выделение группы строк и запись их в отдельный файл и т. д.

С помощью текстового редактора можно создать файл, состоящий только из прямых команд. При считывании такого файла в память ЭВМ исполнение прямых команд происходит сразу по мере их поступления.

Таким способом на экране дисплея может быть нарисована графическая картинка, вычислены некоторые значения переменных и записаны в память, при этом текст программы, находящийся в памяти, не увеличивается.

Операторы для работы с файлами и соответствующие пояснения приведены ниже. В связи с тем, что группа операторов LIBRARY позволяет работать с различными устройствами, оператор OPERATE в новых версиях исключен.

Мнемоника	Оператор	Пример	Действие
LIBRARY GET FILE	Прочитать программу из библиотеки	L G MX:TEST	Загрузить файл MX:TEST в абсолютном формате, предыдущая программа стирается
LIBRARY XECUTE FILE	Выполнить программу из библиотеки	L X MX3:BETA	Загрузить файл MX3:BETA в абсолютном формате и запустить на счет, предыдущая программа стирается

Мнемоника	Оператор	Пример	Действие
LIBRARY SAVE FILE	Записать программу в библиотеку	L S AKTA	Записать программу в файл DK:AKTA в абсолютном формате
LIBRARY INPUT FILE	Открыть для ввода текстовый файл из библиотеки	L I SY:TEST	Оператор переключает ввод с клавиатуры в файл SY:TEST. Чтение происходит вплоть до конца файла, или до оператора LIBRARY CLOSE (см. ниже)
LIBRARY OUTPUT FILE	Открыть для вывода текстовый файл	L O LP: L O SY:RES.DAT	Открыть для вывода файл на устройство построчной печати Открыть для вывода файл SY:RES.DAT на системном устройстве
LIBRARY CLOSE	Закрыть текущий файл, открытый командой L 0 или L 1	L C LP: L O LP:;W 12;L C	Закрыть файл, выводимый на устройство печати Вывести на печать 12-ю группу строк программы

Примечания:

1. Все файловые операции требуют присутствия резидентного драйвера соответствующего устройства, поэтому перед работой с устройством, которое не является системным, необходимо выполнить загрузку драйвера командой LOAD. При отсутствии драйвера выдается сообщение «НЕСУЩЕСТВУЮЩЕЕ УСТРОЙСТВО».

2. Расширением по умолчанию для операторов L G, L X, L S является FOC, а для операторов L 1, L 0 — .FTX.

3. При использовании операторов L 1, L 0 происходит дублирование ввода или вывода на терминал. Иногда эта возможность является лишней, поэтому, указав после имени файла ключ /N, можно запретить дублирование. При этом несколько повысится скорость ввода (вывода):

```
L I DATA/N
L 0 LP:/N
```

Следует учитывать, что применение ключа /N при вводе программы в диалоговом режиме не позволяет обнаружить конец ввода, так как последний символ *, означающий приглашение к работе после ввода, не печатается. Для исключения ошибок в подобных ситуациях ввод

программы должен осуществляться без ключа /N либо текст вводимой программы должен содержать в конце прямую команду L C.

Не следует писать такую комбинацию операторов:

```
L I TEST; L C
```

так как файл в начале будет открыт, а потом сразу закрыт, в результате ввода не произойдет.

Правильно будет просто:

```
L I TEST
```

при этом произойдет ввод, что можно наблюдать на экране терминала, и по достижении конца ввода файл будет автоматически закрыт.

4. По умолчанию система выделяет под выходной файл половину наибольшей свободной области на диске. Выдача сообщения «НЕТ МЕСТА» свидетельствует о том, что файл слишком велик. Существует возможность запросить всю наибольшую область, указав после имени файла комбинацию [—1], например:

```
L 0 PROGR|—1|/N
L 0 MX:DTA.DAT|—1|; TЎ; L C; T «КОНЕЦ»!
```

5. Любая ошибка при выполнении программы вызывает закрытие выходного файла, сообщение о ней выводится на терминал и происходит возврат в диалог. Таким образом, все результаты работы сохраняются в выходном файле вплоть до ошибки.

6. В данных версиях ФОКАЛа возможно одновременное открытие только одного файла, поэтому при попытке выполнить еще одну файловую операцию возникает ошибка «канал занят», текущий файл закрывается и происходит выход в диалог.

7. Операторы LIBRARY могут использоваться в тексте программы, однако следует помнить, что после выполнения:

L G, L S — происходит выход в диалог.

L X — запускается загруженная программа,

L I, L O, L C — продолжается выполнение программы.

8. При загрузке программ в абсолютном формате (.FOC) интерпретатор ФОКАЛа обеспечивает преобразование адресов загрузки, что делает его совместимым с ранними версиями. Однако после загрузки такой программы выдается сообщение «ОШИБКА ВВОДА», что не должно вызывать тревоги у пользователя. Необходимая реакция на это сообщение — это набрать команду ERASE (стирание переменных), чтобы очистить буфер переменных, далее с помощью оператора WRITE убедиться, что программа находится в памяти. Чтобы избежать подобной ситуации в дальнейшем, следует создать новую копию программы оператором LIBRARYSAVE FILE.

Список литературы

1. *Шнейдерман Б.* Психология программирования / Пер. с англ. — М.: Радио и связь, 1984.

2. *Преснухин Л. Н.* и др. Адаптер локальной вычислительной сети на базе БИС К1801ВП1-065//Микропроцессорные средства и системы, 1986, № 6, с. 57—59.

3. *Кокорин В. С. и др.* Персональные ЭВМ. — М.: Высшая школа, 1988.

4. *Крейг Д. Роуз.* Экран, защищающий от электромагнитного и ультрафиолетового излучения//Электроника, 1985, т. 58, № 6, с. 19—20.

5. *Флинт Д.* Локальные сети ЭВМ. — М.: Финансы и статистика, 1986, с. 82—86.

6. *Розенберг Р.* Сюрпризы кольцевой сети с маркерным доступом фирмы IBM//Электроника, 1985, т. 58, № 22, с. 3—6.

7. *Преснухин Л. Н. и др.* Учебный класс на основе диалоговых вычислительных комплексов//Микропроцессорные средства и системы, 1985, № 3, с. 39—41.

8. *Уокерли Дж.* Архитектура и программирование микроЭВМ/ Пер. с англ. — М.: Мир, 1984, кн. 2, с. 330—340.

9. *Безобразов В. С. и др.* Контроллер графического дисплея// Микропроцессорные средства и системы, 1986, № 3, с. 50—53.

10. *Корчак А. Е.* Интерпретатор языка программирования БЕЙСИК для микроЭВМ//Микропроцессорные средства и системы, 1985, № 3, с. 21—24.

11. *Храпченко Т. С.* Сравнительная характеристика диалоговых языков БЕЙСИК и ФОКАЛ//Программное обеспечение малых управляющих комплексов. — М.: ИНЭУМ, 1976, вып. 55, с. 23—31.

12. *Пратт Т.* Языки программирования: разработка и реализация / Пер. с англ. — М.: Мир, 1979, с. 12—16, 96—98.

13. *Писаревский А. Н. и др.* ФОКАЛ — диалоговый язык для миниЭВМ. — Л.: Машиностроение, 1985.

14. *Стоун Г. С., Сиворек Д. П.* Введение в организацию ЭВМ и структурных данных / Пер с англ. — М.: Машиностроение, 1980, с. 231—235.

15. *Фролов Г. И. и др.* Языковые средства микроЭВМ «Электроника БК-0010»//Методы разработки и реализации программного обеспечения ЭВМ. Сб. научных трудов. — М.: МИЭТ, 1985, с. 36—41.

16. *Фролов Г. И. и др.* Программирование на языке ФОКАЛ/ Под ред. *Л. Н. Преснухина.* — М.: МИЭТ, 1984.

17. *Ги К.* Введение в локальные вычислительные сети / Пер. с англ. — М.: Радио и связь, 1986.

18. *Полосин А. Н. и др.* Учебный компьютер «Электроника УК НЦ»//Микропроцессорные средства и системы, 1986, № 6.

19. *Косенков С. М. и др.* Бытовая персональная микроЭВМ «Электроника БК-0010»//Микропроцессорные средства и системы, 1985, № 1.

20. *Фролов Г. И. и др.* Комплектный класс технических средств на базе микроЭВМ «Электроника БК-0010Ш» и «ДВК-2МШ»// Микропроцессорные средства и системы, 1986, № 4.

21. *Форсайт Дж. и др.* Машинные методы математических вычислений/Пер. с англ. — М.: Мир, 1980, с. 260—267.

22. *Кнут Д. Е.* Искусство программирования. Том 2. Получисленные алгоритмы/Пер. с англ. — Мир, 1977.

23. *Вернер В. Д. и др.* Средства сопряжения. Контролирующие и информационно-управляющие системы. Микропроцессоры. Кн. 2/Под ред. Л. Н. Преснухина. — М.: Высшая школа, 1986, с. 292—317.

Оглавление

МикроЭВМ

В ВОСЬМИ КНИГАХ

ГЕОРГИЙ ИВАНОВИЧ ФРОЛОВ, ВАДИМ АНАТОЛЬЕВИЧ ШАХНОВ, НИКОЛАЙ АЛЕКСЕЕВИЧ СМИРНОВ

КН. 8

МикроЭВМ в учебных заведениях

Заведующая редакцией Н. И. Хрусталева. Редактор И. Г. Волкова. Мл. редакторы Г. Г. Бучина, Е. В. Судьенкова. Художник В. М. Боровков. Художественный редактор В. И. Мешалкин. Технический редактор Г. А. Фетисова. Корректор Г. И. Кострикова

ИБ 7155

Изд. № СТД—588. Сдано в набор 10.03.88. Подп. в печать 20.10.88. Т—15664. Формат 84 × 108$^{1}/_{32}$. Бум. тип. № 2. Гарнитура литературная. Печать офсетная. Объем 8,40 усл. печ. л. 17,22 усл. кр.-отт. 7,86 уч.-изд. л. Тираж 150 000 экз. Зак. № 1269. Цена 50 коп.

Издательство «Высшая школа». 101430. Москва, ГСП-4, Неглинная ул., д. 29/14.

Ярославский полиграфкомбинат Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли. 150014, Ярославль, ул. Свободы, 97.