МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА И СИСТЕМЫ

4 / 1989

ISSN 0233-4844

Процессор цифровой обработки сигналов с системой команд векторного типа: основные элементы — базовый матричный кристалл К1806ВП1 и микросхемы памяти К573РУ8

АПЛ-система — это программные и аппаратные средства, поддерживающие весь цикл разработки и эксплуатации АПЛ-программ и обеспечивающие доступ ко всем ресурсам вычислительного комплекса

Микропрограммный ассемблер МАСС — основное средство автоматизации проектирования вычислительных и управляющих устройств на базе секционированных микропроцессоров

Микросхемы памяти с УФ-стиранием информационной емкостью 16, 64, 256 Кбит серии К573

Операционные системы общего назначения и реального времени микроЭВМ СМ1810 — МДОС1810, МИКРОС-86, БОС1810 и ОС поддержки тестового программного обеспечения ТОС-86

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ИНФОРМАТИКА—89

20-летию сотрудничества социалистических стран в области разработки, производства и применения вычислительной техники была посвящена международная выставка, проводившаяся в начале лета этого года в Москве, на ВДНХ СССР.

На выставке были представлены: совместные разработки средств вычислительной техники и автоматизированных информационных систем различного назначения, создаваемые на их основе разнообразные вычислительные системы и комплексы; результаты коллективных усилий по выпуску совместимых друг с другом универсальных машин ЕС ЭВМ, мини- и персональных ЭВМ; изделия микроэлектроники большой степени интеграции, устройства микропроцессорной техники, мини- и микроЭВМ; локальные вычислительные сети, программные средства и т. д.

Рассмотрим вкратце наиболее интересные экспонаты каждой из стран — участниц выставки.

Советский раздел — самый представительный. На него приходится почти треть всех экспонатов.

Персональная электронная вычислительная машина «Истра-4816» — базовая модель ряда разрабатываемых и планируемых к разработке высокопроизводительных ПЭВМ, реализованных на отечественной элементной базе. По принятой в СССР классификации «Истра-4816» относится к классу ПЭВМ для различных профессиональных применений.

«Истра-4816» имеет оригинальную трехпроцессорную архитектуру. Сдин из процессоров КР580ВМ80 используется в качестве периферийного и совместно с набором периферийных модулей серии К580 образует подсистему ввода-вывода, которая аппаратно-программными средствами эмулирует аппаратную среду выбранного прототипа (в данном случае IBM PC). Два процессора КР1810ВМ86 и КР580ВМ80, именуемые системными, выполняют функции центральных процессоров.

Четырехпортовое ОЗУ объемом от 1 до 4 Мбайт доступно четырем пользователям. Доступ к ОЗУ регламентируется арбитром поля памяти. Пользователями выступают три процессора и контроллер видеотерминала, причем каждый из них имеет доступ ко всему объему ОЗУ.

Системные процессоры не имеют периферийного обрамления. В качестве средств ввода-вывода и прерывания для них используется аппаратно-программная среда периферийного процессора. Системные процессоры сигналами ввода-вывода воздействуют на систему прерываний периферийного процессора.

Операционная система ПЭВМ «Истра-4816» заключает несколько одновременно работающих ОС: MS DOC (версии 3.20—3.30), CP/M80 и др.

Персональный компьютер ЕС1842. В отличие от предыдущей модели имеет более высокое быстродействие (2 млн. операций/с), новые функциональные возможности (многозадачный режим работы на основе эмуляции микропроцессора I80286, виртуальную организацию оперативной памяти, повышенную разрешающую способность средств отображения графической информации—640×350 точек, аппаратную реализацию ряда функций построения графических изображений и т. д.). Увеличена емкость накопителей на гибких магнитных дисках и оперативной памяти, существенно улучшены характеристики графической подсистемы, расширены функциональные возможности клавиатуры.

Модули профессиональной ориентации значительно повышают эффективность работы ПЭВМ в составе комплексов, рассчитанных на различные конкретные приложения.

ПЭВМ ЕС1842 может быть подключена к локальной сети, построенной на основе кольцевой топологии с возможностью включения в сеть до 125 абонентов. Максимальное расстояние между двумя соседн-

(Продолжение см. на 1...4 с. вкл. и 3 с. обл.)

Рабочее место дежурной сестры, которая в любой момент может получить консультацию программы «Айболит»

«Медиана» обеспечивает решение в автоматическом режиме ряда медицинских задач

Пролетарии всех стран, соединяйтесь!

ОРГАН ГОСУДАРСТВЕННОГО КОМИТЕТА СССР ПО ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКЕ И ИНФОРМАТИКЕ

Издается с 1984 года

МП МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА И СИСТЕМЫ

ВЫХОДИТ ШЕСТЬ РАЗ В ГОД НАУЧНО-ТЕХНИЧЕСКИЙ И ПРОИЗВОДСТВЕННЫЙ ЖУРНАЛ 4/1989 МОСКВА

СОДЕРЖАНИЕ

УДК 681.323

В. И. Мутанов, Ю. П. Гришина, Ю. Г. Чернявский, В. И. Чернышев

ПРОЦЕССОР ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ С ВЕКТОРНОЙ СИСТЕМОЙ КОМАНД

В настоящее время развиваются два основных направления в выборе архитектуры процессора цифровой обработки сигналов (ПЦОС) [1, 2]. В первом случае это ПЦОС со встроенной системой высокоуровневых команд и алгоритмов типа быстрого преобразования Фурье (БПФ), во втором — ПЦОС с достаточно широким набором универсальных команд. При использовании последних пользователь имеет возможность выбрать оптимальный для него алгоритм программирования и модифицировать программное обеспечение по мере совершенствования алгоритмической базы. Например, переход от алгоритма БПФ Кули-Тьюки к алгоритму простых множителей Гуда и вычислению малоточечных дискретных преобразований Фурье (ДПФ) по методу Рейдера приводит к уменьшению времени обработки сигнала [3].

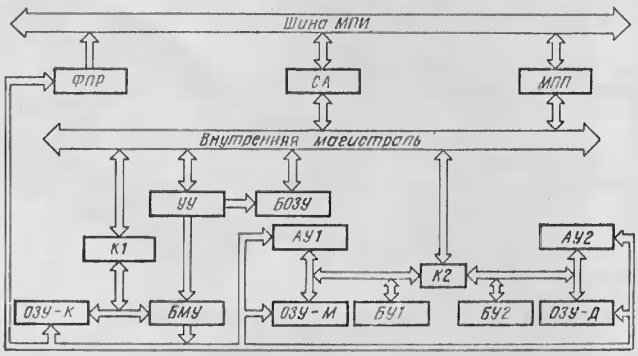
В ПЦОС со встроенной системой высокоуровневых команд большая производительность обеспечивается внутренним оптимальным конвейером для исполняемой команды. При проектировании программируемых ПЦОС также необходимо изыскивать средства ускорения вычислительного процесса. Одним из них является векторная система команд, позволяющая аппаратно реализовать метод конвейерной обработки [4] при выполнении одной и той же операции над элементами массива (вектора). Соответствующая ей архитектура процессора объединяет гибкость универсальной системы команд с высокой скоростью обработки информации.

К особенностям разработанного процессора помимо векторной системы команд относятся:

использование магистрали с интерфейсом МПИ (ГОСТ 26765.51-86) для обмена с вычислительной системой;

совмещение процессов обмена по шине МПИ и обработки информации процессором;

применение элементной базы среднего быстродействия, разработанной по КМОП-технологии;



Структурная схема процессора цифровой обработки сигналов с векторной системой команд

вычисление ДПФ с изменяющимся в широких пределах числом элементов вектора (за счет применения алгоритма простых множителей).

Основные характеристики процессора

Число разрядов	16
элементов вектора команды	32
Наибольшая длина вектора, элементов	512
Емкость ОЗУ, слов	
команд	2048
данных	4096
Цикл обращения к памяти данных, мкс	0,67
Частота внешнего тактового генератора, МГц	6
Потребляемая мощность, Вт	
статическая	0,1
динамическая	1,0
Напряжение питания, В	5±10 %
Масса, г	300
Габаритные размеры, мм	220×12×19

Производительность ПЦОС определяется временем (в миллисекундах) выполнения операций над 256-точечными комплексными векторами: сложение — 0,5, умножение — 1,7, умножение вектора на действительную или мнимую константу — 0,85, умножение вектора на комплексную константу — 1,2, вычисление квадратов модулей элементов вектора — 1,2, быстрое преобразование Фурье — 8,0. Способ представления данных — с фиксированной запятой.

Структурная схема процессора представлена на рисунке. Связь между шиной МПИ и внутренней магистралью обеспечивается магистральным приемопередатчиком (МПП) и селектором адреса (СА). Операционная часть ПЦОС обрабатывает одновременно действительную и мнимую составляющие комплексного операнда с помощью двух арифметических устройств АУ1 и АУ2, двух блоков умножения БУ1 и БУ2, ОЗУ данных для действительных (ОЗУ-Д) и мнимых (ОЗУ-М) компонентов операндов и констант. АУ1 и АУ2 можно объединить в 32-разрядный блок. Для управления операционной частью используется блок микропрограммного управления БМУ. Микрокоманды и команды хранятся в ОЗУ команд (ОЗУ-К). Буферное ОЗУ (БОЗУ) вместе с коммутаторами каналов К1 и К2 обеспечивают параллельность работы операционной части и обмен информацией по шине МПИ. Для управления режимом работы ПЦОС служит устройство управления (УУ). Формирователь ФПР вырабатывает запрос прерывания с соответствующим вектором прерывания по окончании выполнения программы. Запуск программы ПЦОС осуществляется засылкой в регистры УУ начального адреса программы и кода пуска.

Основой элементной базы ПЦОС является базовый матричный кристалл (БМК) К1806ВП1, на котором выполнены схемы УУ, БМУ, АУ1, АУ2, К1, К2, ФПР. Число типов БМК — 5. Общее число БМК в ПЦОС — 27. Все ОЗУ выполнены на десяти микросхемах К537РУ8; БУ1 и БУ2 — на четырех микросхемах К1824ВР21; МПП — на микросхеме К588ВА1; СА — К588ВТ1. Конструктивно ПЦОС реализован на одной плате с двусторонним размещением элементов.

Система команд ПЦОС состоит из векторных операционных команд и команд управления. 32-разрядные команды включают в себя четыре поля: код формата, код операции, адреса первого и второго операндов. Команды выполняются по схеме

$$V(A_2+IH)=V(A_1+IH)\oplus V(A_2+IH),$$

$$I=0, 1, 2,..., L-1,$$

где V (A) — элемент вектора с адресом A в ОЗУ данных; ⊕ — символ операции; A_1 и A_2 — адреса первых элементов векторов, заданные в полях команды. Шаг H, с которым из ОЗУ данных извлекаются элементы векторов, и число элементов L задаются командой «параметры вектора», предшествующей группе векторных команд. В состав операционных команд входят команды сложения, вычитания, сдвига элементов вектора на один разряд вправо, умножения вектора на действительную и комплексную константы, умножения векторов, вычисления квадратов модулей элементов вектора, команды обмена векторами между БОЗУ и ОЗУ данных, пересылки векторов в ОЗУ данных. ПЦОС имеет также команду вычисления двухточечного ДПФ. В формате управления реализованы команды останова, установки разрядности АУ (16 и 32) и задания параметров вектора.

Благодаря применению микропрограммного управления система команд ПЦОС может быть дополнена, что позволяет использовать процессор для решения широкого круга задач.

Телефон 402-96-81, Москва

ЛИТЕРАТУРА

1. Маклауд Д. Цифровая обработка сигналов: микросхемы готовы, а программных средств нет // Электроника.— 1988.— № 7.— С. 18—22.
2. Новый класс цифровых процессоров сигналов — векторные процессоры // Электроника.— 1986.— № 15.— С. 18—27.
3. Нуссбаумер Г. Быстрое преобразование Фурье и алгоритмы вычисления сверток / Пер. с англ.— М.: Радио и связь, 1985.
4. Мануэль Т. Начало массового выпуска суперЭВМ // Электроника.— 1988.— № 5.— С. 18—28.

Статья поступила 27.12.88

УДК 681.32

Г. А. Галуев

СПЕЦИАЛИЗИРОВАННЫЙ МАТРИЧНЫЙ ПРОЦЕССОР ДЛЯ ОБРАБОТКИ БИНАРНЫХ ИЗОБРАЖЕНИЙ

Основная трудность, возникающая в процессе создания эффективных систем обработки зрительной информации, состоит в преодолении противоречия между двумерностью фотометрической картины внешнего мира и одномерностью процесса ее последовательной обработки традиционными средствами вычислительной техники [1—4].

Предлагается достаточно простой и эффективный оператор выделения контура и анализа бинарных изображений, который осуществляет обработку каждой x_0 точки изображения и восьми соседних с ней точек x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 в окрестности размерами 3×3 в соответствии с выражением следующего вида:

$$y_0 = x_0(\overline{x_1x_3x_5x_7})(x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6 \vee x_7 \vee x_8),$$
$$z_1 = y_1y_5y_0,\ z_2 = y_2y_6y_0,\ z_3 = y_3y_7y_0,\ z_4 = y_4y_8y_0,$$
$$z_5 = y_0(z_1 \vee z_2 \vee z_3 \vee z_4),$$

где $x_0 \ldots x_8$ — точки изображения; $y_0 \ldots y_8$ — контурные точки изображения.

Оператор выделяет контур изображения, устраняет помехи в виде единичных изолированных точек, позволяет проанализировать контурное изображение и определить его основные локальные информационные признаки: прямолинейные участки, ориентацию этих участков и угловые точки контура. Применяя оператор одновременно ко всем точкам изображения, можно параллельно осуществить операцию выделения контура изображения и основных локальных информационных признаков. Практически указанная операция реализуется с помощью специализированного матричного процессора, архитектура которого адекватно отражает структуру обрабатываемых данных (матрицу точек изображения).

Процессор представляет собой матрицу ячеек (рис. 1), каждая из которых соответствует отдельной точке изображения x_6 и обрабатывает информацию в окрестности этой точки (рис. 2). Перед началом работы ячейка сбрасывается в исходное состояние установкой элемента памяти 4 (см. рис. 1) в нулевое состояние. Изображение объекта проецируется на фотоприемные элементы (ФЭ) матричного процессора. На выходах ФЭ появляются единичные сигналы, соответствующие анализируемой точке x_0 и восьми соседним с ней точкам изображения. Сигналы поступают на входы данной ячейки, элемент памяти 4 которой устанавливается в состояние, определяемое логической функцией y_0 (см. выражение 1).

В соответствии с этой функцией анализируются условия принадлежности точки x_0 контуру и наличия единичной изолированной точки. Если анализируемая единичная точка x_0 является изолированной (т. е. все соседние с ней точки равны нулю), то функция y_0 принимает нулевое значение и, следовательно, данная точка x_0 не включается в контур изображения, т. е. устраняется. Сигналы $y_1 \ldots y_8$ с выходов элементов памяти соседних ячеек, соответствующих точкам $x_1 \ldots x_8$ (см. рис. 2), поступают на входы элементов И 5, 6, 7, 8 (рис. 1) данной ячейки, на выходах которых реализуются функции $z_1 \ldots z_4$. В соответствии с этими функциями проверяется условие принадлежности анализируемой точки x_0 изображения контуру и одновременно наличие на данном фрагменте контура двух точек, расположенных по разные

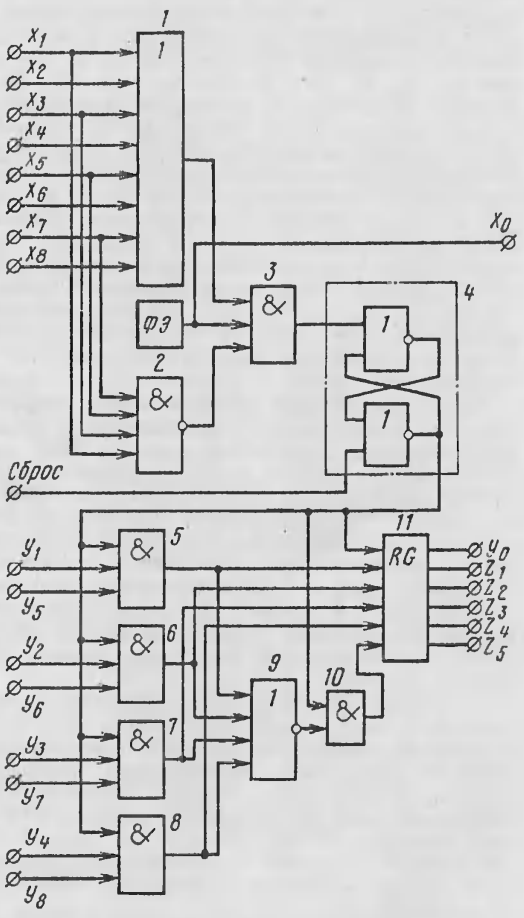


Рис. 1. Структура ячейки обработки информации

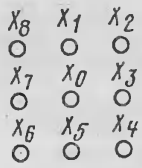


Рис. 2. Точка x_0 изображения и ее окрестность размерами 3×3

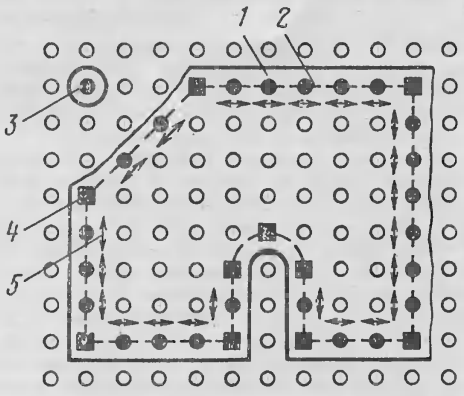


Рис. 3. Результаты моделирования работы матричного процессора

стороны от x_0, т. е. выделяются прямолинейные участки контура определенной ориентации, которая соответствует прямой, проведенной через точки x_1, x_0, x_5, или x_2, x_0, x_6, или x_3, x_0, x_7, или x_4, x_0, x_8 (рис. 2). После выполнения функций z_1...z_4 реализуется логическая функция z_5 из выражения 1, в соответствии с которой проверяется условие принадлежности точки x_0 изображения контуру при одновременном отсутствии на данном фрагменте контура прямолинейного участка, т. е. выделяются угловые точки контура. Таким образом, каждая ячейка обработки информации реализует выражение 1 для соответствующего участка изображения размерами 3×3. На выходе каждой такой ячейки появляется двоичный код $F=\langle y_0, z_1, z_2, z_3, z_4, z_5\rangle$, который записывается в регистр 11 (см. рис. 1) и однозначно определяет результаты обработки и анализа соответствующего участка изображения.

Массив 6-разрядных двоичных кодов, отображающих результат обработки всего изображения в целом, может последовательно считываться в память ЭВМ или параллельно передаваться на другой специализированный матричный процессор, где осуществляется дальнейший анализ и распознавание изображения. При этом из всего массива данных, появляющегося на выходе рассматриваемого процессора, для дальнейшей обработки отбираются только те данные, которые соответствуют выделенным контурным точкам, т. е. только отличные от нуля коды F. Последнее обстоятельство существенно снижает объем информации об изображении и упрощает процесс его дальнейшей обработки.

Матричный процессор можно выполнить на основе современной интегральной технологии в виде системы на пластине. Например, для матричного процессора размерами 100×100 ячеек (точек изображения) необходима пластина площадью 10^4 мм^2. При использовании КМОП-технологии на такой пластине можно разместить 10^4 кристаллов площадью 1 мм^2, каждый из которых содержит 10^3 вентилей совместно с фотоприемным элементом (на одну ячейку обработки информации требуется около 70 вентилей).

Существенные сложности возникают при считывании информации из матричного процессора. Чтобы организовать параллельный процесс считывания информации из всех 10^4 ячеек в последовательном коде, необходимы 10^4 выводов на пластине, а в параллельном коде — $6\cdot10^4$ выводов, что явно нереализуемо и нецелесообразно с точки зрения надежности. Наиболее приемлемый способ считывания информации — последовательное считывание в параллельном коде с предварительным отбором только тех данных, которые соответствуют выделенным контурным точкам изображения. Для организации такого способа считывания информации требуется шесть выводов для выдачи кодов F_i (i=1, 2,...,10^4) и соответствующая схема, обеспечивающая выдачу только отличных от нуля кодов F_i. Такой способ считывания данных наиболее целесообразно использовать при передаче из матричного процессора в ЭВМ, где ввод информации осуществляется последовательно.

Если данные передаются в другой специализированный процессор с матричной архитектурой, то можно использовать параллельно-последовательный способ считывания и передачи информации. Для этого одна из линеек регистров хранения кодов F_i выбирается в качестве выходной, а между остальными регистрами организуется такая система связей, при которой информация сдвигается от одной линейки регистров к другой в направлении той, что выбрана в качестве выходной. В результате этого происходит считывание одновременно 100 значений кодов F_i, находящихся в выходной линейке регистров. При этом сдвиг информации от одной линейки регистров к другой и считывание из выходной линейки осуществляются последовательным кодом, что требует 100 выводов на пластине и не представляет существенных затруднений.

Результаты моделирования работы процессора с изображением произвольного объекта 1 представлены на рис. 3. Процессор выделяет контур изображения объекта 2, удаляет помехи в виде единичных изолированных точек 3, определяет угловые точки 4 и прямолинейные участки контура и их ориентацию 5. Результаты моделирования полностью подтвердили справедливость изложенных результатов. Матричный процессор выполнен на стандартных микросхемах серии К1533 и использован в качестве аппаратной поддержки для обработки изображений при вводе в моделирующий комплекс для автоматизации нейробионических исследований [5].

347928, Таганрог, ГСП-284, Ростовская об., ул. Чехова, 2, НИИ МВС; тел. 6-25-75

ЛИТЕРАТУРА

1. Александров Е., Платонов А. Алгоритмы и системы восприятия роботов // Представление знаний в человекомашинных и робототехнических системах.— М.: ВЦ АН СССР.— 1984.— Т. 2.— С. 139—194.
2. Куафе Ф. Взаимодействие робота с внешней средой.— М.: Мир, 1985.— 285 с.
3. Павлидис Т. Алгоритмы машинной графики и обработки изображений.— М.: Радио и связь, 1986.— 400 с.
4. Галуев Г. А., Мильков Н. А. Обработка изображений на основе цифровых нейроноподобных сетей // Депонированные научные работы.— М.— 1987.— № 2.— С. 18. Деп. 22.10.86, № 7352—В86.
5. Каляев А. В., Чернухин Ю. В., Брюхомицкий Ю. А., Галуев Г. А., Ветер В. В. Моделирующий комплекс для автоматизации нейробионических исследований // Многопроцессорные вычислительные структуры.— Таганрог: ТРТИ.— 1987.— Вып. 9 (XVIII).— С. 68—71.

Статья поступила 8.12.87

УДК 681.327.28—529

В. П. Сидоренко, С. И. Яровой, А. А. Хоружий

БИС РПЗУ С УФ-СТИРАНИЕМ ИНФОРМАЦИИ СЕРИИ К573

Современный уровень развития РПЗУ с УФ-стиранием характеризуется серийным производством БИС информационной емкостью 16, 64 и 256 Кбит (ближайшая перспектива — 1 Мбит). Одновременно с увеличением информационной емкости благодаря внедрению КМОП-технологии улучшаются и другие технические характеристики РПЗУ: повышаются быстродействие и надежность, снижается потребляемая мощность, увеличивается число допустимых циклов перезаписи информации [1—3].

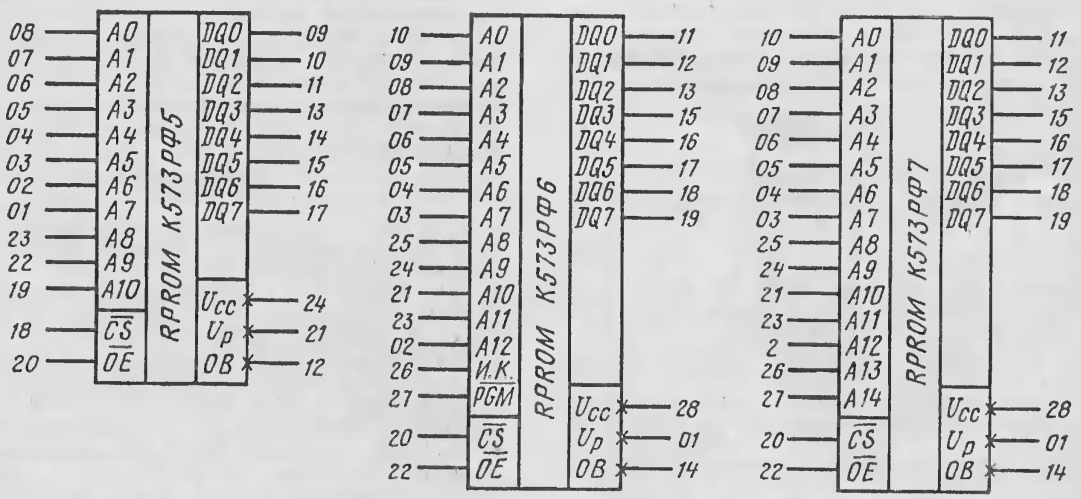
Серийно выпускаемые БИС РПЗУ с УФ-стиранием К573 — РФ5, РФ6, РФ7 (см. таблицу) — это приборы статического типа с двухвходовой архитектурой управления (рис. 1), обеспечивающей максимальную гибкость и оптимальную совместимость модулей памяти на их основе с современными высококачественными микропроцессорными системами [4]. Предусмотренный в РПЗУ режим пассивного хранения, характеризующийся пониженным уровнем потребляемой мощности (по сравнению с активным режимом потребляемый ток уменьшается примерно в 3...4 раза для n-канальных и в 30...50 раз — для КМОП РПЗУ), позволяет увеличивать информационную емкость блока памяти без существенного увеличения потребляемой мощности. В режиме хранения на управляющий вход $\overline{CS}$ (вход включения кристалла) подается напряжение высокого уровня U_{IH}; в режиме считывания на управляющие входы $\overline{CS}$ и $\overline{OE}$ (вход включения выходов) — напряжение низкого уровня U_{IL}. Сигнал $\overline{CS}$ переводит БИС из состояния пассивного хранения в состояние готовности к выполнению адресации и считывания информации. Однако до тех пор, пока на вход $\overline{OE}$ не поступит напряжение низкого уровня, выходные каскады прибора удерживаются в состоянии высокого выходного импеданса.

Данная архитектура управления модулями памяти позволяет освободить системную шину для работы с другими приборами памяти или периферийными устройствами (рис. 2) на время отключения РПЗУ и присутствия информации на шине данных. Входы $\overline{OE}$ всех приборов памяти подключены к общей шине управления, которая обеспечивает раздельное и независимое управление шиной данных; таким образом осуществляется синхронизация модуля памяти непосредственно процессором (рис. 3).

В режимах программирования и считывания БИС РПЗУ серии К573 совместимы по адресным и управляющим входам с ТТЛ-микросхемами. Выходы прибора при этом используются в качестве информационных входов.

Технические характеристики БИС РПЗУ с УФ-стиранием

Характеристика	Тип прибора		
	К573РФ5	К573РФ6	К573РФ7
Информационная емкость, Кбит	16	64	256
Организация, слов×разряд	2048×8	8192×8	32768×8
Потребляемый ток в режиме, мА			
обращения	100	150	100
хранения	25	50	40
Время выборки адреса, нс, не более	450	300; 450	250; 300; 450
Напряжение питания, U_{CC}, В	5±5 %	5±5 %	5±5 %
Напряжение программирования, U_P, В	(25±0,5)	(21±0,5)	(12,5±0,5)
Время хранения информации в режиме:			
считывания, тыс. ч.	50	20	15
хранения, лет	10	5	5
Число циклов перезаписи информации	100	25	25
Тип корпуса	2121.24-6	2121.28-6	2121.28-6



Примечание И.К. – вывод микросхемы не используется

Рис. 1. Условное графическое обозначение БИС РПЗУ с УФ-стиранием информации по функциональному назначению выводов

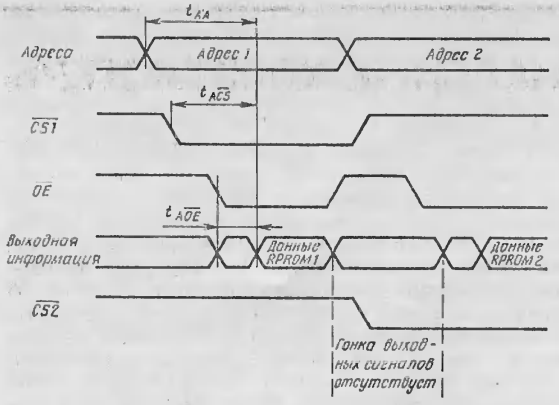


Рис. 2. Временна́я диаграмма работы БИС РПЗУ с двухвходовой архитектурой управления в составе модуля памяти

Программирование выполняется с применением последовательного или произвольного перебора адресов. Подача на управляющий вход $\overline{CS}$ напряжения высокого уровня обеспечивает режим запрета программирования и позволяет записывать информацию в РПЗУ непосредственно в блоке памяти.

В алгоритме программирования, разработанном для БИС К573РФ5, используется импульс фиксированной длительности — 50 мс, определяемой временем программирования для наихудшего случая, т. е. максимальным временем, за которое можно записать информацию в любой элемент матрицы. Такой алгоритм программирования допускает два варианта записи информации: по всем адресам с последующим контролем правильности программирования и по каждому текущему адресу.

Для БИС РПЗУ с бо́льшей информационной емкостью (К573РФ6, К573РФ7) все более очевидными становятся достоинства адаптивного алгоритма программирования (рис. 4) с подачей коротких программирующих импульсов длительностью 1 мс [5]. После подачи первого такого импульса записи выполняется считывание информации из ячеек памяти по данному адресу. Если хотя бы один запоминающий элемент окажется незапрограммированным, то подача программирующих импульсов продолжается до тех пор, пока все элементы не перейдут в требуемое состояние. Однако число импульсов программирования не должно превышать определенной величины: для БИС К573РФ6 — не более 25, для К573РФ7 — не более 45. Если после подачи очередного импульса по программируемому адресу считывается истинная информация, то на РПЗУ подается дополнительный импульс трехкратной длительности, обеспечивающий надежность хранения записанной информации. Использование адаптивного метода позволяет в несколько раз сократить время программирования РПЗУ с УФ-стиранием.

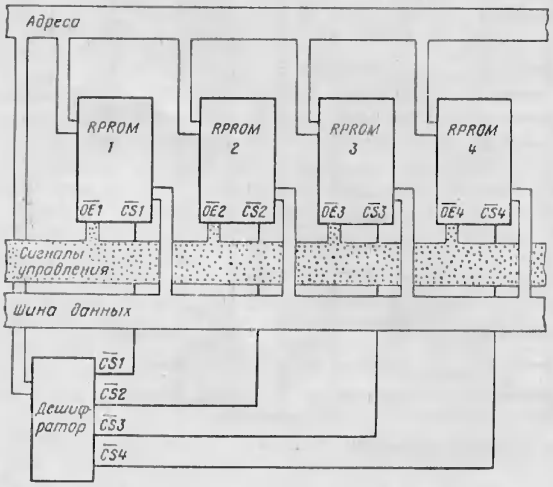


Рис. 3. Функциональная схема модуля памяти на базе БИС РПЗУ

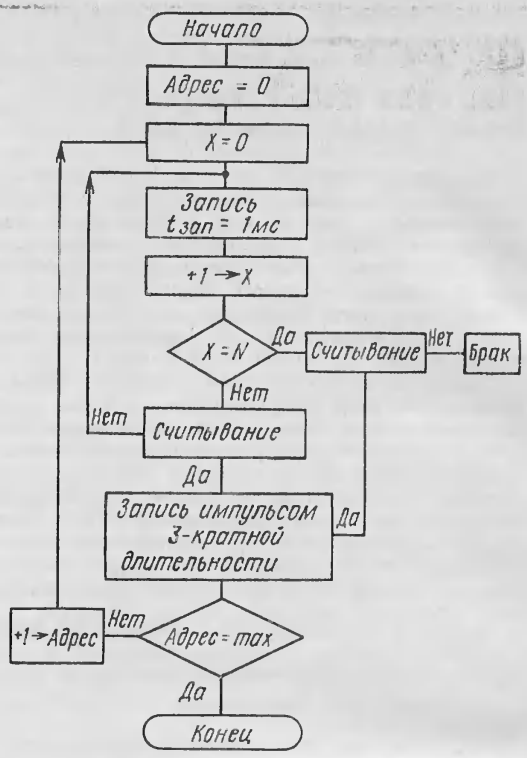


Рис. 4. Алгоритм адаптивного программирования БИС РПЗУ с УФ-стиранием информации

Стирается информация путем воздействия ультрафиолетового излучения с длиной волны λ=253,7 нм, соответствующей наиболее интенсивной линии спектра ртутных ламп, направленного на кристалл перпендикулярно его плоскости через прозрачную кварцевую крышку.

Время стирания $t_c = H/P$, с, где P — энергетическая освещенность УФ-излучения в плоскости кварцевой крышки корпуса, Вт/м^2; H — требуемое значение энергетической экспозиции, равное $1{,}5 \cdot 10^5$ Дж/м^2. После стирания все ячейки памяти находятся в состоянии высокого уровня напряжения (U_{OH}=2.4 В).

Телефон 442-95-04, Киев

ЛИТЕРАТУРА

1. А. с. 1297114 СССР. Формирователь импульсов записи/ В. П. Сидоренко, Н. Б. Груданов, А. А. Хоружий.— Опубл. 1987. Бюл. № 10.
2. А. с. 1339651 СССР. Формирователь импульсов записи/ В. П. Сидоренко, Н. Б. Груданов, А. А. Хоружий.— Опубл. 1987. Бюл. № 35.
3. А. с. 1233125 СССР. Стабилизатор постоянного напряжения/А. Я. Сирота, Ю. В. Прокофьев, П. Н. Зуб.— Опубл. 1986. Бюл. № 19.
4. John F. Rizzo. Disign with EPROMS For Future Flexibility.— Intel EPROM Applications Manual, 1980.
5. M. Van Biskirk, M. Holler, G. Korsh, B. Lee et all. E-PROMS graduate to 256K density with scaled n-channel process // Electronics.— 1983.— Vol. 56.— N 4.— P. 89—93.

Статья поступила 4.04.88

УДК 681.322.1

В. Г. Кравчук, А. А. Некрасов, В. В. Флорентьев

ОПЫТ РАБОТЫ С ПРОФЕССИОНАЛЬНЫМИ ПЕРСОНАЛЬНЫМИ КОМПЬЮТЕРАМИ ЕС1840

1. Устройство ЕС1840 (ЕС1840.05).

Профессиональная персональная ЭВМ ЕС1840 (рис. 1) состоит из пяти функциональных модулей: системного, накопителей на гибких магнитных дисках (НГМД), дисплея-монитора, клавиатуры, принтера.

1.1. Системный модуль включает в себя конструктив Е13.083.077 типа «корзина» с системной шиной, расположенной вертикально с задней стороны корзины, и стабилизированный источник питания. Конструктив рассчитан на подключение к системной шине семи горизонтальных плат, расположенных сверху вниз: ОЗУ, процессора, адаптера дисплея, адаптера НГМД, адаптеров интерфейсов. Два резервных места служат для подключения, например, адаптеров винчестерского диска и локальной сети. Все платы выполнены по восьмислойной технологии.

Системная шина состоит из восьмислойной печатной платы и установленных на ней семи разъемов СМП34С-135 на 135 контактов каждый. Платы в конструктиве устанавливаются произвольно.

Источник питания системного блока выполнен на двух печатных платах, объединенных стальным экранирующим кожухом. С торца кожуха расположен вентилятор. Источник питания включает в себя: входной фильтр, выпрямитель сетевого напряжения, блокинг-генератор, пусковой источник, параметрические стабилизаторы на +5, +12 и —12 В и микросборку управления. Микросборка отслеживает изменение напряжения +5 В и вырабатывает управляющее воздействие на блокинг-генератор. Кроме этого, микросборка управления при включении питания выдает сигнал на сброс системы. **Кнопка системного сброса не предусмотрена.** Плавкие предохранители расположены так, что **доступ к ним без снятия пломбы невозможен.**

Оперативное запоминающее устройство ЕС1840 в стандартном исполнении имеет объем 512 Кбайт. Для обеспечения нужного объема ОЗУ и контроля четности используются 72 микросхемы К565РУ5 по 64 Кбит каждая. Вместе с обрамлением плата ОЗУ содержит 110 микросхем.

Процессор. На плате процессора расположены: центральный процессор К1810ВМ86 (аналог i8086), ПЗУ ROMBIOS, системный контроллер К1810ВГ88 (аналог i8288), контроллер прерываний К1810ВМ59А (аналог i8259А), контроллер прямого доступа к памяти К1810ВТ37А (аналог i82374А), программируемый таймер К580ВИ53А (аналог i8253А), генератор тактовых последовательностей К1810ГФ84 (аналог i8284), шинные формирователи. **Общее число микросхем — 54. Подключение арифметического сопроцессора не предусмотрено.**

Адаптер дисплея. Основа адаптеров алфавитно-цифрового и графического дисплеев — контроллер СМ607 (аналог Motorola 6845). Кроме этого, на плате адаптера алфавитно-цифрового дисплея, входящего в комплектацию ППЭВМ ЕС1840, расположены восемь микросхем К541РУ2 буферной памяти экрана и девять микросхем К537РУ2 перезагружаемого знакогенератора. Адаптер обеспечивает вывод 25 строк по 90 знаков при матрице знакогенератора 9×14. **Общее число микросхем — 91.** В ЕС1840.05 работает графический дисплей МС6105 [4]. На плате адаптера графического дисплея буферная память экрана выполнена на восьми микросхемах динамического ОЗУ К565РУ5 (из 64 Кбайт можно использовать одно из основных окон по 16 Кбайт). Память перезагружаемого знакогенератора реализована на микросхеме К537РУ10. Адаптер вырабатывает аналоговые сигналы RGB, сигналы Intensity и Video, сигналы строчной и кадровой разверток HD и VD. Адаптер графического дисплея имеет следующие режимы работы: два текстовых — 25 строк по 40 знаков (знакоместо 8×8) и 25 строк по 80 знаков (знакоместо 8×8) и графический — 320×200 пиксел — 4 цвета из 16, 640×200 пиксел — монохромный.

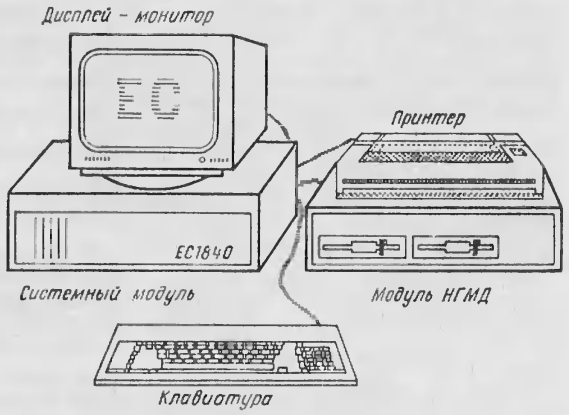


Рис. 1. Персональный компьютер ЕС1840

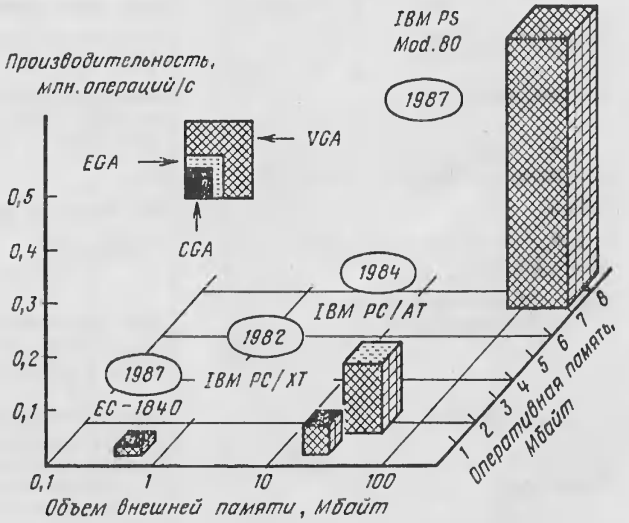


Рис. 2. Условное положение ЕС1840 в ряду персональных ЭВМ фирмы IBM

Общее число микросхем — 94. Таким образом, адаптер — близкий функциональный аналог CGA.

Адаптер НГМД выполнен на основе микросхем СМ609 (аналог i8272). Расположенные, кроме него, на плате микросхемы обрамления формируют управляющие сигналы. Адаптер позволяет программировать длину записи данных (128, 256, 512, 1024 байт на сектор) и поддерживает работу четырех дисководов. **Число микросхем — 32.**

Адаптер интерфейсов имеет два последовательных и один параллельный порты. Параллельный порт реализован на микросхемах малой интеграции, работает только в режиме вывода информации и предназначен для обслуживания принтера. **Последовательные интерфейсы созданы не в стандарте IBM PC, не поддерживаются BIOS и утилитами MC ДОС. Число микросхем — 56.**

1.2. НГМД. В ЕС1840 используются два НГМД FD-55FV производства фирмы TEAC (Япония). В ППЭВМ ЕС1840.05 в состав модуля НГМД был неожиданно включен **блок питания графического дисплея,** причем расположен он над левым дисководом. НГМД предназначен для работы с дискетами 5.25″ и обеспечивает объем записи 360 и 720 Кбайт на дискету DSDD (double side double density 96 tpi).

1.3. Дисплей-монитор ЕС1840 — алфавитно-цифровой, одноцветный, цвет свечения экрана зеленый. Блок питания конструктивно входит в монитор. Кожух устроен так, что при снятии задней крышки срабатывает блокировка, отключающая его от сети.

Монитор ЕС1840.05 — графический, черно-белый, МС6105 [3]. Блок питания расположен в модуле НГМД.

1.4. Клавиатура имеет русский и латинский регистры, среди 93 клавиш есть такие уникальные, как «рус», «инф» и «ё». В неподвижной части каждой клавиши находится геркон, контакты которого срабатывают при опускании магнита, закрепленного в подвижной части клавиши. Опрос

Таблица 1

Сопоставление ЕС1840 и IBM PC/XT

Функциональная подсистема	ЕС1840 (ЕС1840.05)	IBM PC/XT
Конструктив	Типа «корзина» с вертикально расположенной системной шиной, рассчитанной на подключение семи плат	Motherboard (одна горизонтально расположенная плата с разъемами расширений)
Процессор	К1810ВМ86	i8088
Арифметический сопроцессор	Подключение не предусмотрено	Есть
Кнопка системного сброса	Отсутствует	Есть
Адресная шина	20 разрядов	20 разрядов
Шина данных	16 разрядов	8 разрядов
Таймер	При отключении сети не предусмотрен	Есть
ROM BIOS	Не обеспечивает поддержку последовательного адаптера. Неудачное расположение кодов кириллицы. Отсутствует функция загрузки перезагружаемого знакогенератора	
ОЗУ	512 Кбайт. Отсутствует ключ, позволяющий исключить контроль четности	256 Кбайт с возможностью расширения до 640 Кбайт
Адаптер графического дисплея	Графика есть только в ЕС1840.05, в модели ЕС1840 установлены алфавитно-цифровой адаптер и монитор	
Выходные сигналы	RGB аналоговый, цифровой Intensity, HS, VS, Video	RGB аналоговый, цифровой, Intensity, VS, Video
Режимы работы	Текстовый (25×40, 25×80), графический цветной (320×200) и монохромный (640×200)	Те же
Перезагружаемый знакогенератор	Есть	Нет
НГМД	360 Кбайт на дискете 5″ 1/4	360 Кбайт на дискете 5″ 1/4
Накопитель на жестком диске типа винчестер	Отсутствует	10...20 Мбайт
Адаптеры интерфейсов: последовательный	Два нестандартных последовательных интерфейса, не поддерживаются ROM BIOS и утилитами MS DOS	Имеется возможность расширения числа и номенклатуры интерфейсов за счет подключения дополнительных плат
параллельный	Один параллельный интерфейс работает только на вывод, предназначен для обеспечения принтера	Обычно интерфейс принтера расположен на плате графического монитора
Клавиатура	93 клавиши. Скан-коды некоторых из них не соответствуют IBM PC/XT. Целесообразно скан-коды дополнительно введенных клавиш располагать после стандартных скан-кодов	83 клавиши

клавиатуры и управление ею осуществляются однокристальной микроЭВМ КМ1816ВЕ48 (аналог i8048). Программа ее работы записана в РПЗУ с ультрафиолетовым стиранием.

1.5. Принтер. В состав ЕС1840 (ЕС1840.05) включен матричный принтер RAVI 8010M (Индия), функционально эквивалентный EPSON FX-85 (Япония). Отличия: кириллица, зашитая в ПЗУ принтера, и бирка RAVI вместо бирки EPSON. Принтер позволяет работать в режимах NLQ, DRAFT и графическом, причем все режимы можно или запрограммировать или ввести с клавиатуры принтера. Принтер работает только по параллельному интерфейсу. Каретка рассчитана на стандартный формат пишущей машинки.

1.6. Документация. В комплект поставки ЕС1840 входят:

инструкция по эксплуатации;
формуляр;
ведомости эксплуатационных документов (три книги);
система программирования «Микроприз» (формуляр — две книги и руководство пользователя);
тестовые программные средства (руководство пользователя);
операционная система М86 (телетекс);
пакеты прикладных программ М86, «Абак», «Слог» (руководства пользователя);
«БЕЙСИК М86»
система АСМ86 (ассемблер, сборщик, управление вводом-выводом);
печатающее устройство FX-85 (руководство пользователя).

Альбом «Эксплуатационная документация на персональную профессиональную электронную вычислительную машину ЕС1840» (в трех частях) пользователю **не поставляется.**

2. Сопоставление ЕС1840 и IBM PC/XT.

Персональный компьютер PC/XT фирмы IBM (США) — наиболее массовая модель 1982—1985 гг. Всего за этот период выпущено около 5 млн. компьютеров этого класса, к 1987 г.— около 10,5 млн., включая следующую модель — IBM PC/AT (с 1984 г.).

Положение ЕС1840 среди персональных компьютеров ряда IBM ориентировочно представлено на рис. 2. Производительность оценивалась по тесту Гибсон-3, основание пирамиды соответствует разрешению графического адаптера (GGA — 640×200, EGA — 640×350 и VGA — 1024××768).

В табл. 1 сопоставляются ЕС1840 (ЕС1840.05) и персональный компьютер IBM PC/XT [4], наиболее близкий по техническим характеристикам к ЕС1840. Отметим еще и нестандартное расположение кириллицы в ПЗУ принтера.

Вполне очевидно, что **надежность ПК обратно пропорциональна числу микросхем, энергопотреблению и числу межсоединений.** Кроме того, **на надежность всего изделия (ПК) принципиально влияет надежность его элементной базы.**

Также отметим, что **IBM PC/XT построен по принципу открытой архитектуры, т. е. базовую конфигурацию можно расширить за счет подключения плат расширения** дополнительной памяти, контроллеров устройств и интерфейсов и т. д. Этим обстоятельством руководствуются за рубежом сотни независимых фирм-изготовителей аппаратуры. **Но поскольку конструктив ЕС1840 существенно отличается от IBM PC/XT, такими платами воспользоваться нельзя.**

Все внешние разъемы (например, последовательный интерфейс, параллельный интерфейс, выход на монитор) **принципиально отличаются от принятых в IBM PC и в мировой практике. В стандартной поставке ЕС1840 ответные части разъемов не прилагаются. Это делает невозможным, например, подсоединение ЕС1840 к линии связи с большой ЭВМ (типа ЕС1045) или соединение двух ЕС1840 между собой по последовательному интерфейсу.**

3. Конструктивные недоработки ЕС1840.

К сети ЕС1840 подключается четырьмя (ЕС1840.05 — тремя) сетевыми вилками. Это создает определенные не-

Таблица 2

Технические неисправности

ППЭВМ ЕС1840, ЕС1840.05

	ППЭВМ ЕС1840					ППЭВМ ЕС1840.05		
Заводской номер	0036	0128	0366	0367	0395	2070	2083	2149
Дата ввода в эксплуатацию	12.86	12.86	01.87	01.87	01.87	10.87	10.87	10.87
Процессор	*				*		*	
ОЗУ	*	*	*		*			
Адаптер дисплея		*	*			*	*	**
Адаптер НГМД								
Адаптер интерфейсов								
Блок питания								
НГМД		*					*	
Блок питания НГМД				*				
Дисплей			*					
Блок питания дисплея				**				
Клавиатура	*	*	*		*	*		
Принтер				*				
Вентилятор		*	*					

Примечание: * — неисправность

удобства в эксплуатации. Расположение выключателей сети с задней стороны блоков крайне неудобно, так как каждое включение и выключение машины сопровождается мучительными поисками выключателей. Сброс машины (при зависании процессора) с помощью сетевого выключателя не целесообразен, а **кнопка «Сброс» отсутствует**. Хомут крепления провода дисплея задевает за заднюю крышку системного блока и ломает ее.

Конструкция ключа на разъемах соединительных кабелей **допускает неправильное (инверсное) включение этих разъемов.**

Конструктивный недостаток — расположение блока питания дисплея над дисководом. Функционально эти узлы никак не связаны, а при работе с объемом записи на дискете 720 Кбайт **сбои из-за наводок от блока питания регулярны**. Наиболее подходящим представляется **разместить блок питания дисплея в подставке под ним** (как это сделано в ППЭВМ Нейрон И9.66).

В некоторых случаях **контакт в панельке плохой** из-за разного шага отверстий панельки (метрический) и ножек микросхем (дюймовый — в тех случаях, когда используются зарубежные микросхемы).

В адаптере графического дисплея ЕС1840.05 во всех полученных машинах присутствовала **монтажная ошибка** (поменяли местами R13 и R14), в результате чего при подключении цветного дисплея интенсивность синего цвета была сильно занижена. При приемке машины с монохромным дисплеем этот дефект обнаружить практически невозможно.

Как уже отмечалось, предохранители блоков питания находятся под крышками, опломбированными заводскими штампами (т. е. **заменить предохранитель официально может только представитель завода!**). В конструкции машины **не предусмотрена установка арифметического процессора**, в принципиальных схемах сознательно опущены необходимые для этого цепи.

4. Технические неисправности, выявленные в процессе эксплуатации ППЭВМ ЕС1840 и ЕС1840.05, представлены в табл. 2.

Более подробно неисправности проанализированы ниже (табл. 3, 4).

Зависание процессора машины 02083 вызвано некачественным контактом, что проявлялось при нагреве печатной платы. Причина некачественного контакта — разный шаг панелек и микросхем.

Замыкание конденсатора развязки на массу произошло по причине попадания его в направляющую конструктива при заводской сборке. В результате краска протерлась и обкладки конденсатора замкнулись на массу. Потребляемый ток возрос, но не настолько, чтобы сработала защита. В итоге — выход из строя блока питания (см. табл. 6).

Неисправности в машинах 02070, 02083 и 02149 были определены после подключения цветных графических дисплеев. При использовании черно-белого дисплея неисправности практически не проявляются.

Неисправность блока питания НГМД у одной из ПЭВМ проявилась один раз. Характер неисправности: не проходило обращение к НГМД, головка не опускалась. Причина неисправности — выход из строя микросхемы параметрического стабилизатора.

Характер единичной неисправности дисплея-монитора: изображение в верхней части экрана сильно растянуто по вертикали. Причина неисправности — выход из строя микросхемы, задающей импульсы кадровой развертки.

Неисправностей блока питания дисплея было две: отсутствие выходного напряжения 27 В (из-за выхода из строя транзисторов и стабилитрона) и сетевого напряжения при изломе пружины блокировки (из-за нарушения режима термообработки).

Характер неисправности клавиатуры (в течение полугода эксплуатации у пяти ПЭВМ) — систематический отказ отдельных клавиш и (у одной ПЭВМ) зависание клавиатуры. Причина первой неисправности — обрыв электрической цепи в диоде, включенном последовательно с герконом, а второй — плохой контакт ИС процессора с панелькой.

В каждой машине вышли из строя от одной до пяти клавиш вследствие обрыва диода. **После замены его на диод с более высокой вибростойкостью отказы по этой причине прекратились.**

Отказ принтера произошел из-за неподнятия каретки. Причина — изгиб рычага поднятия каретки.

Отказы вентиляторов системного блока питания произошли из-за обрыва обмотки (один раз) и заклинивания (при исправных обмотках). Причина заклинивания — выход из строя подшипников (пыль проникла через сальник).

Как видно, **основные причины неисправностей ППЭВМ ЕС1840 (ЕС1840.05): отказы элементной базы — микросхем низкой и средней интеграции и дискретных элементов; контактные ошибки; ошибки, допущенные при заводской сборке.**

Таблица 3

Неисправности модуля процессора

Параметр	Заводской номер		
	00036	00395	02083
Дата выхода из строя	06.87	09.87	10.87
Характер неисправности	Зависание процессора	Зависание процессора	Зависание процессора
Причины неисправности	Выход из строя микросхемы шинного формирователя	Выход из строя микросхемы системного контроллера	Контактная ошибка

Таблица 4

Неисправности модуля ОЗУ

Параметр	Заводской номер			
	00036	000128	00366	00395
Дата выхода из строя	09.87	01.87	12.87	08.87
Характер неисправности	Не проходит тест ОЗУ	Отказ по цепи питания	Не проходит тест ОЗУ	Не проходит тест ОЗУ
Причины неисправности	Выход из строя микросхемы динамической памяти	Замыкание конденсатора развязки на массу	Выход из строя четырех микросхем динамической памяти	Выход из строя микросхемы динамической памяти

Таблица 5

Неисправности адаптера дисплея

Параметр	Заводской номер				
	00128	00366	02070	02083	02149
Дата выхода из строя	06.87	12.87	10.87	10.87	10.87
Характер неисправности	Не проходит инициализация адаптера	Отказ буферной памяти экрана	Низкий уровень сигнала синего цвета	Низкий уровень сигнала синего цвета	1. Низкий уровень сигнала синего цвета. 2. Смешение сигналов B и G
Причины неисправности	Выход из строя микросхем средней степени интеграции	Контактная ошибка	Монтажная ошибка, резисторы R13 и R14 впаяны ошибочно	То же, что у ПЭВМ 02070	1. То же, что у ПЭВМ 02070. 2. Замыкание соседних контактных площадок из-за некачественной пайки

Таблица 6

Неисправности блока питания системного модуля

Параметр	Заводской номер	
	00128	02083
Дата выхода из строя	02.87	11.87
Характер неисправности	Отсутствуют выходные напряжения +5, +12, —12 В, вентилятор не вращается	То же
Причины неисправности	Выход из строя транзисторов пускового блока питания, транзистора блокинг-генератора, выпрямительного моста, предохранителей	Выход из строя транзисторов пускового блока питания, предохранителей

5. Аппаратные доработки ЕС1840 и ЕС1840.05.

В процессе эксплуатации в ППЭВМ ЕС1840.05 были проведены следующие аппаратные доработки:

подключены цветные графические дисплеи МС6106;

введена возможность отмены режима работы с контролем четности (то же и для ЕС1840);

процессор дополнен арифметическим сопроцессором (разработана дополнительная плата, устанавливаемая на плате процессора);

доработан последовательный интерфейс для подключения графопостроителя.

В настоящее время проводятся: установка кнопки «Сброс» с соответствующими электрическими цепями; оснащение машин схемами таймера с сохранением их работоспособности на время выключения общего питания (васкир); расширение ОЗУ до 640 Кбайт; подключение графических дисплеев к ЕС1840; оснащение машин интерфейсами, полностью совместимыми с IBM PC; включение в состав каждой машины накопителя на винчестерском диске. **Целью этих доработок является доведение машин до функциональной совместимости с IBM PC/XT.**

6. Программное обеспечение ЕС1840.

В комплект поставки ЕС1840 входит ОС М86, функционально тождественная ОС CP/M-86 фирмы Digital Research (версии, приблизительно, 1984 г.). Принципиальное отличие М86 от CP/M-86 заключается в том, что все ее директивы сконструированы на основе слов русского языка (например, СОСТ вместо DIR, КОП вместо COPY). В комплект поставки также входят прикладные программы, работающие под управлением М86, АБАК (работа с электронными таблицами, подобными SuperCalc) и ТЕЛЕТЕКС (для обслуживания последовательного интерфейса при подсоединении к большой ЕС ЭВМ).

Мы отказались от ОС М86 в пользу ОС класса MS DOS, адаптированной для обеспечения работы с русскими буквами. **Адаптация** состояла из:

переделки драйвера работы с клавиатурой (используются все отличия клавиатуры ЕС1840 от клавиатуры IBM PC/XT, в том числе и наличие клавиш работы с русским регистром); драйвер вырабатывает внутреннее ASCII-представление символов, соответствующее альтернативной кодировке [5];

создания набора программ для формирования и загрузки знакогенератора дисплея-монитора; новые образы символов загружаются в контроллер монитора, для образов (FONT'ов), используются матрицы 9×14 (стандарт MDA, ЕС1840) и 8×8 (стандарт CGA, ЕС1840.05); улучшена их читаемость на экране; для русских букв используется альтернативная кодировка;

переделки драйвера принтера; эта адаптация потребовалась для того, чтобы использовать заготовленные в RAVI-8010M образы шрифтов (в том числе и для режима NLQ), которые упорядочены по уникальным кодовым таблицам, принятым в RAVI-8010M и отличающимся и от прямой, и от альтернативной кодировок.

Вся настройка ОС производится при работе файла AUTOEXEC.BAT во время начальной загрузки системы. Кроме этого, в файле AUTOEXEC.BAT производится установка драйвера EDISK, что позволяет работать с четырьмя логическими гибкими дисками (от A до D) на двух физических. Объем каждого логического диска — 360 Кбайт.

Технологически вся работа на ЕС1840 имеющейся конфигурации строится по следующему принципу:

на логических дисках A и C (левый дисковод) располагается ОС класса MS DOS и какая-либо прикладная система (например, редактор текстов, либо система работы с электронными таблицами, либо система управления базами данных); дискета закрыта на запись;

на логических дисках B и D (правый дисковод) размещаются рабочие файлы пользователя, дискета открыта на запись.

Установка ОС класса MS DOS на ЕС1840 (ЕС1840.05) позволяет использовать единый фонд ПО для персональных компьютеров класса IBM PC.

Модернизация ЕС1840 (ЕС1840.05), связанная с установкой арифметического сопроцессора (см. раздел 6), дала возможность использовать эти компьютеры и для вычислительных задач, в том числе для задач модели-

рования, которые требуют наличия сопроцессора.

7. Заключение.

В приведенной ниже табл. 7 отказы ППЭВМ разбиты на три группы: выход из строя элементной базы; плохой контакт между микросхемой и панелькой (контактная ошибка); заводской дефект.

Выход из строя диодов клавиатуры обусловлен неправильным выбором элементной базы (заводской дефект). Время наработки на отказ оценивалось из расчета пятидневной рабочей недели, восьмичасового рабочего дня. Пять машин эксплуатировались 12 мес., три машины — 3 мес. Коэффициент использования К принят равным 0,8. Общая наработка — 12 080 ч, **наработка на отказ (с учетом К) — 350 ч.**

Таким образом, можно сделать вывод, что **ППЭВМ ЕС1840 (ЕС1840.05) соответствует мировому уровню начала 80-х годов.** У завода существуют большие возможности для увеличения времени наработки на отказ за счет улучшения культуры производства и перехода на элементную базу большей степени интеграции и надежности. **В целом, ЕС1840 (ЕС1840.05) пригодна для профессионального использования как персональный компьютер и представляет собой качественно новый шаг по сравнению с ранее выпускавшимися микроЭВМ типа ДВК2, «Искра 226».** Вместе с тем, уже сейчас совершенно очевидно, что должен быть преодолен следующий барьер на пути к современному уровню персональных компьютеров — к машине с твердым диском типа винчестер (20 Мбайт).

Функционально соответствовать IBM PC/XT, бывшему стандартом «де факто» в середине 80-х годов, судя по проспектам, будет ЕС1841 с точностью до массогабаритных характеристик и наличия закупленных за рубежом деталей (микросхем, устройства НГМД, твердого диска типа винчестер, принтера). Качественное улучшение надежностных характеристик можно ожидать при переходе на элементную базу нового поколения. Предпосылки, что к этому периоду производства отечественных персональных компьютеров произойдет и полный переход на отечественные комплектующие (все микросхемы и периферийные устройства), существуют.

Таблица 7

Основные причины отказов ППЭВМ типа ЕС1840

Функциональная подсистема	Выход из строя элементной базы	Контактная ошибка	Заводской дефект
Процессор	2	1	—
ОЗУ	3	—	1
Адаптер монитора	1	1	4
Блок питания системного модуля	2	—	—
Блок питания НГМД	1	—	—
Дисплей-монитор	1	—	—
Блок питания дисплея	1	—	1
Клавиатура	—	1	5
Принтер	—	—	1
Вентилятор	—	—	2
Сумма	11	3	14
Сумма, %	39	11	50

Телефон 196-76-44, Москва

ЛИТЕРАТУРА

1. Пыхтин В. Я. ЕС1840 — базовая персональная ЭВМ единой серии // Микропроцессорные средства и системы.— 1986.— № 4.— С. 15—16.
2. Материалы межведомственного семинара «Персональные компьютеры».— Москва: ИАЭ им. И. В. Курчатова, 1987.
3. Видеомониторы для персональных ЭВМ / С. И. Сорока и др. // Микропроцессорные средства и системы.— 1986.— № 4.— С. 34—36.
4. IBM Personal Computers at a Glance // Byte.— Vol. 9.— N 9.— Special IBM Issue.— P. 10.
5. Брябрин В. М., Ландау И. Я. Неменман М. Е. О системе кодирования для персональных ЭВМ // Микропроцессорные средства и системы.—1986. — № 4. — С. 62—63.

Статья поступила 28.04.88

УДК 681.326

Д. В. Горшков, Г. В. Зеленко, А. В. Шишкин

МИКРО 16—ОДНОПЛАТНАЯ ПЭВМ НА ОСНОВЕ МИКРОПРОЦЕССОРА КР1810ВМ86

На одной плате ПЭВМ МИКРО 16 размещены МП КР1810ВМ86, ОЗУ объемом 128 Кбайт, ПЗУ 64 Кбайт, экранное ОЗУ 16 Кбайт, схемы управления вводом-выводом, таймеры разверток и синхронизации, логика выбора режимов и т. д. (рис. 1). Программное обеспечение МИКРО 16 включает базовую систему ввода-вывода (БСВВ), БЕЙСИК, тестовые и игровые программы. Отдельные подпрограммы БСВВ, обслуживая «свои» устройства ввода-вывода, имитируют работу ВУ IBM PC.

Логическая организация схемы отображения МИКРО 16 позволяет поддерживать следующие графические режимы IBM PC: среднего разрешения 320×200 точек и высокого разрешения 640×200 точек. Адреса портов устройств ввода-вывода соответствуют адресам аналогичных портов IBM PC, что позволяет использовать даже те программы, которые для повышения эффективности работы непосредственно обращаются к аппаратным ресурсам ПЭВМ, минуя подпрограммы БСВВ.

Устройством отображения ПЭВМ может служить телевизионный приемник цветного или черно-белого изображения, а внешним устройством памяти — обычный магнитофон. Для использования ПЭВМ в профессиональной деятельности предусматриваются подключение НГМД и работа с ОС MS DOS или CP/M-86.

Запланирован вариант ПЭВМ для учебного процесса, в котором ЭВМ учащихся не имеют индивидуальных устройств внешней памяти и коллективно используют ВУ и файловую систему центральной ПЭВМ.

На плате ПЭВМ МИКРО 16 размерами 320×200 мм размещено 59 микросхем, в том числе блок памяти объемом 128 Кбайт, состоящий из 16 микросхем К565РУ5Д. Для тактирования МП служит тактовый генератор КР580ГФ84, на выходе CLK которого формируется частота синхронизации, равная 4 МГц и в четыре раза меньшая опорной частоты кварцевого резонатора. Это достигается принудительной остановкой внутреннего счетчика-делителя микросхемы КР580ГФ84 в каждом такте синхронизации МП на время, равное одному периоду опорной частоты, с помощью сигнала, подаваемого на вход CSYNC от схемы управления. Выбранное соотношение между опорной частотой и частотой синхронизации обеспечивает синхронную работу МП и схемы отображения при их обращении к ОЗУ ПЭВМ.

Схема управления формирует взаимодействия элементов ПЭВМ с ОЗУ (рис. 2). Время доступа к ОЗУ делится на фазы процессора и отображения

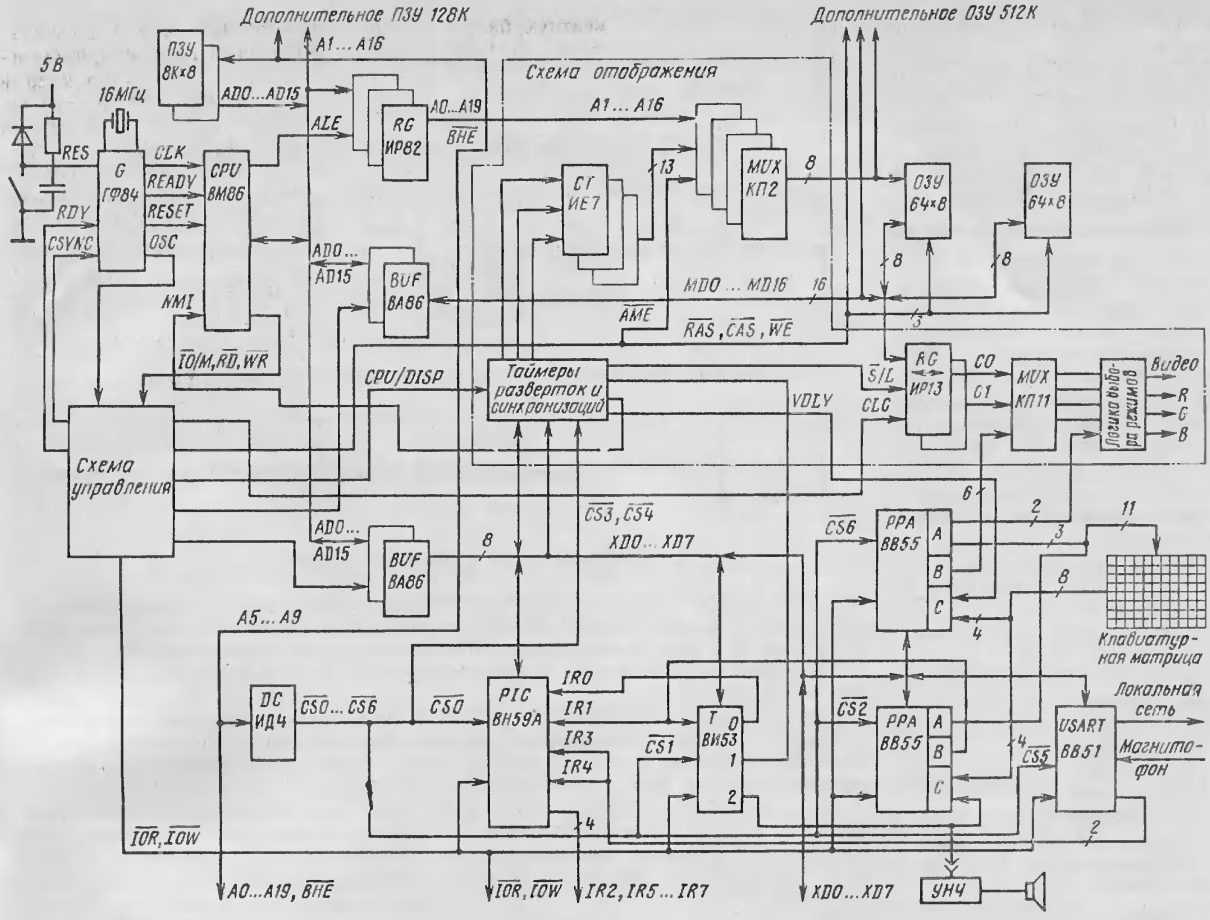


Рис. 1. Функциональная схема ПЭВМ МИКРО 16

длительностью по 500 нс в соответствии с сигналом CPU/DISP. В фазе отображения слово считывается из экранной области ОЗУ и записывается в сдвиговые регистры схемы отображения. МП при обращении к ОЗУ переводится в состояние ожидания до прихода положительного фронта сигнала CPU/DISP. В фазе процессора завершается цикл шины МП (до наступления очередной фазы отображения). В структуру ПЭВМ введены два шинных формирователя КР580ВА86, отделяющие шину данных ОЗУ от мультиплексированной шины и исключающие конфликты на шине МП в фазе отображения.

ОЗУ ПЭВМ можно расширить до 640 Кбайт путем подключения внешних микросхем памяти, смонтированных на дополнительных печатных платах, размещаемых над основной. Дополнительным блокам памяти присваиваются младшие адреса, а блоку памяти на плате ПЭВМ, всегда содержащему экранный буфер, отводятся адреса, следующие непосредственно за адресами внешнего блока.

В отличие от ПЭВМ IBM PC, где экранное ОЗУ реализовано в виде дополнительной памяти, размещенной по адресам, начиная с 0B8000H, в МИКРО 16 под буфер экрана отводятся старшие 16 Кбайт ОЗУ ПЭВМ. При всех обращениях к экранному ОЗУ происходит аппаратная модификация адресов, необходимая для программной совместимости.

В режиме среднего разрешения кодировка каждой точки осуществляется двумя битами, т. е. один байт содержит информацию о четырех точках (рис. 3). Слово считывается из ОЗУ за один цикл обращения и записывается в два сдвиговых регистра схемы отображения, причем в один — все четные, а в другой — нечетные разряды слова. Восемь 2-разрядных кодов (C1, C0), последовательно появляющихся при сдвиге на выходах регистров, соответствуют восьми цветным точкам строки растра изображения.

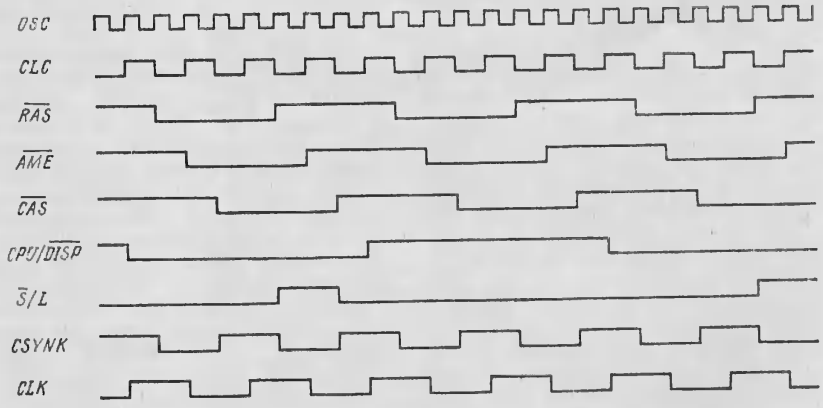


Рис. 2. Временная диаграмма работы ОЗУ

Таблица 1

C1	C0	Бит 5=0	Бит5=1
0	0	Цвет фона	Цвет фона
0	1	Зеленый	Голубой
1	0	Красный	Лиловый
1	1	Желтый	Белый

Изображение формируется в пределах рабочего окна, занимающего центральную часть экрана. Боковые, не задействованные для отображения, участки экрана образуют рамку. При использовании цветного телевизионного приемника изображение представлено точками трех цветов, выбираемых из набора (палитры): зеленого, красного, желтого, голубого, лилового, белого (табл. 1). Фон, на котором строится изображение, окрашивается в один из 16 цветов (табл. 2.).

Таблица 2

R	G	B	I	Цвет фона
0	0	0	0	Черный
0	0	1	0	Синий
0	1	0	0	Зеленый
0	1	1	0	Голубой
1	0	0	0	Красный
1	0	1	0	Лиловый
1	1	0	0	Коричневый
1	1	1	0	Светло-серый
0	0	0	1	Серый
0	0	1	1	Светло-синий
0	1	0	1	Светло-зеленый
0	1	1	1	Светло-голубой
1	0	0	1	Светло-красный
1	0	1	1	Светло-лиловый
1	1	0	1	Желтый
1	1	1	1	Белый (яркий)

Таким образом, в пределах рабочего окна одновременно отображаются четыре различных цвета. Рамка окрашивается в один из этих цветов. Текущая комбинация цветов на экране выбирается в соответствии с кодом в регистре цвета (рис. 4).

При использовании телевизионного приемника черно-белого изображения в режиме среднего разрешения цветам соответствуют восемь градаций яркости. В режиме высокого разрешения каждая точка кодируется одним битом (рис. 5); бит, установленный в единицу, отображается в виде светлой точки. Фон в этом режиме всегда черный, а рамка — черная, белая или полосатая.

В ПЭВМ предусмотрены два установочных гнезда для микросхем ПЗУ серии К573 информационной емкостью до 64 Кбайт. В ПЗУ хранится системное программное обеспечение. Можно также подключить внешнее ПЗУ объемом до 128 Кбайт. Для системного и дополнительного ПЗУ отведены

7 C1	6 C0	5 C1	4 C0	3 C1	2 C0	1 C1	0 C0
Точка 1		Точка 2		Точка 3		Точка 4	

Рис. 3. Кодировка точек в режиме среднего разрешения

7	6	5	4	3	2	1	0
Цвет рамки		Выбор палитры	Интенсивность основного цвета	Цвет фона			

Рис. 4. Вариант комбинации цветов в зависимости от кода в регистре цвета

7	6	5	4	3	2	1	0
Точка 1							Точка 8

Рис. 5. Кодировка точек в режиме высокого разрешения

старшие адреса адресного пространства, причем в дополнительном ПЗУ, выполненном в виде сменной кассеты, могут храниться трансляторы, прикладные и игровые программы.

Две микросхемы КР580ВА86 предназначены для организации 8-разрядной шины ввода-вывода. К ней подключены программируемые БИС, реализующие функции контроллера прерываний, системного таймера, параллельного интерфейса ввода-вывода, последовательного интерфейса, программируемых таймеров.

Контроллер прерываний КР1810ВН59А предназначен для обслуживания запросов прерываний от таймера и последовательного интерфейса. Канал 0 БИС системного таймера КР580ВИ53 программируется в режиме делителя частоты и служит для отсчета реального времени в системных и прикладных программах. Выходной сигнал канала используется в качестве запроса прерывания, подаваемого на вход IR0 контроллера прерываний. Канал 1 участвует в формировании кадровых гасящих импульсов видеосигнала. Канал 2 таймера предназначен для формирования звуковых эффектов.

Порты А и В БИС параллельного интерфейса КР580ВВ55 запрограммированы на вывод, порт С — на ввод в режиме 0. Порт А используется для сканирования клавиатуры. Линии порта В служат для поддержки работы клавиатуры и формирования звуковых эффектов. Две линии порта С также задействованы в этой схеме. Еще четыре линии используются для хранения младших разрядов кода опроса клавиатуры. Для подключения магнитофона служат свободные линии портов В и С. Вторая БИС КР580ВВ55 программируется аналогично первой. Младшие пять линий порта А используются для задания режимов отображения, три оставшиеся линии совместно с линиями порта А первого интерфейса — для сканирования матрицы клавиатуры. Порт В выполняет функции регистра цвета для схемы отображения. Младшие четыре разряда порта С составляют регистр состояния схемы отображения, остальные линии предназначены для опроса клавиатуры.

БИС последовательного приемопередатчика КР580ВВ51 необходимы для организации локальной сети с использованием МИКРО 16 в качестве интеллектуального терминала.

На двух БИС КР580ВИ53 реализована схема синхронизации, предназначенная для управления счетчиками схемы отображения и формирования телевизионного видеосигнала. Схема обеспечивает структуру экранного буфера, аналогичную IBM PC. Организация экранного ОЗУ МИКРО 16 показана на рис. 6. Формат изображения можно изменять программно с помощью таймеров.

Адреса портов устройств ввода-вывода ПЭВМ МИКРО 16 соответствуют адресам аналогичных портов IBM PC. На плате размещаются

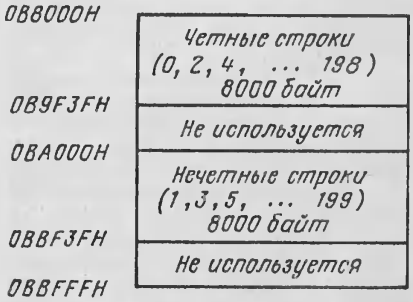


Рис. 6. Структура экранного ОЗУ МИКРО 16

разъемы, на которые выведены мультиплексированная локальная шина адреса-данных для подключения внешнего ПЗУ, системная шина адреса и буферизованная 8-разрядная шина данных для подключения устройств ввода-вывода, в том числе контроллера НГМД, а также сигналы управления и синхронизации для различных схем расширения, четыре линии для приема запросов прерываний от ВУ, свободные линии портов, которые можно использовать для связи, например, с печатающим устройством.

Предусмотрен специальный разъем для подключения дополнительных блоков ОЗУ с минимальными аппаратурными затратами, разъемы для телевизионного приемника или монитора (монохромного или цветного) и усилителя звуковой частоты. Через разъем магнитофона ПЭВМ можно связать с локальной сетью. В центре печатной платы расположены разъемы для клавиатуры типа МС7007, выполненной в виде матрицы нормально разомкнутых контактов.

Опытная эксплуатация показала, что ПЭВМ МИКРО 16 обладает хорошей программной совместимостью с IBM PC. Большинство пакетов прикладных программ для данного типа ПЭВМ работают на МИКРО 16 без каких-либо изменений. Наибольшая совместимость достигается при использовании профессионального» варианта МИКРО 16 в среде ОС МС ДОС или СР/М-86. МИКРО 16 наиболее целесообразно использовать в качестве интеллектуального терминала в локальной сети.

Телефон 268-39-21, Москва

Статья поступила 28.06.88

УДК 519.685.4

А. В. Бармин, М. И. Каганов, А. В. Кондрашев

ДВК КАК ПЕРСОНАЛЬНАЯ АПЛ-машина

АПЛ-машина предназначена для решения научных, инженерных и коммерческих задач в режиме диалога человек — ЭВМ. Ее также можно использовать в качестве интеллектуальной терминальной станции в сетях сбора информации; особенно перспективно применение в обучении (на любом уровне). Все модули, составляющие АПЛ-машину, серийно выпускаются отечественной промышленностью, что делает ее доступной широкому кругу пользователей.

Язык программирования АПЛ (A Programming Language) за свою 25-летнюю историю пережил и периоды подъема, когда некоторые считали, что он заменит многие традиционные языки программирования, и периоды относительной потери популярности, вызванные дороговизной аппаратных средств и рыночной политикой фирм-разработчиков программного обеспечения, предпочитавших продавать законченные программные продукты, сделанные с помощью АПЛ, а не сами АПЛ-системы. Однако даже в неблагополучные для АПЛ годы число пользователей АПЛ-систем неуклонно возрастало. Так, например, к концу 70-х годов в США АПЛ широко использовался в системах с разделением времени (по некоторым данным с помощью АПЛ решалось до 75 % задач). Сегодня, когда главный акцент в мировой индустрии ЭВМ сделан на персональные компьютеры, АПЛ занял свое полноправное место в качестве универсального диалогового языка программирования для ПЭВМ. АПЛ-системы имеются для всех марок зарубежных профессиональных ПЭВМ вне зависимости от типа центрального процессора и версии операционной системы [1], а для одной из самых популярных в мире ПЭВМ типа IBM PC — даже несколько реализаций АПЛ-систем [2].

О популярности АПЛ за рубежом могут свидетельствовать такие факты: ежегодно собирается международный конгресс, посвященный проблемам АПЛ; АПЛ входит в число языков, выделенных ACM (Association for Computing Machinery) как наиболее важных, и в рамках этой международной организации работает специальная рабочая группа; концерном Springer-Verlag издается коммерческий журнал «APL News», а в США выходит научный журнал «APL Quote Quad»; издательство APL Press публикует литературу по теории и практике АПЛ-систем; канадская фирма I. P. Sharp Associates развернула и поддерживает мировую АПЛ-сеть; во многих школах и университетах АПЛ — базовое средство обучения математике, методам алгоритмизации, программированию и с помощью пакетов АПЛ-программ другим дисциплинам.

Несмотря на признание АПЛ во всем мире, отечественные пользователи практически не знакомы с ним. Это, в первую очередь, объясняется тем, что ни одна отечественная ЭВМ не имела в составе своего стандартного программного обеспечения АПЛ-системы. Долгое время в нашей стране АПЛ был доступен лишь небольшой группе пользователей, владеющих импортным оборудованием. Сегодня практически для всех ЭВМ, серийно выпускаемых в СССР, имеются отечественные реализации АПЛ-систем, что существенно расширило круг программистов, работающих на АПЛ. Наиболее используемые отечественные системы — АПЛВИДЕО (для ЕС ЭВМ) и АПЛ/ВЦАН (для СМ ЭВМ) [3, 4]. Уже стала традиционной межотраслевая школа-семинар по АПЛ, которая ежегодно проводится в начале весны в Обнинске.

Тем не менее, огромному большинству пользователей, которым АПЛ мог бы помочь в решении их задач, он не известен или представляется (часто понаслышке) чем-то весьма экзотическим и малопригодным для практического применения. Поэтому, прежде чем описывать конкретную реализацию АПЛ-системы, авторы хотели бы остановиться на следующем вопросе.

Кому и для чего нужен АПЛ?

АПЛ пригодится ученым-прикладникам, инженерам, экономистам, системным аналитикам, руководителям — всем, для кого программирование не является основным видом профессиональной деятельности (этих людей называют конечными пользователями). АПЛ — это средство, с помощью которого конечные пользователи могут самостоятельно без участия профессиональных программистов создавать свои приложения. Насущная потребность в использовании вычислительной техники сейчас ощущается во всех сферах жизни общества, но сама по себе ЭВМ не способна решить ни одной задачи. Практическое использование вычислительной установки всегда осуществляется через программирование для нее. Однако традиционные языки программирования, требующие от программиста достаточно большого объема специальных знаний, обладают слишком высоким «порогом освоения», который, как представляется, не способно преодолеть большинство потенциальных конечных пользователей. Чтобы стать хорошим программистом, скажем на ПЛ/1, необходим талант, которым обладает относительно небольшое число людей. Недаром «программистская библия» Кнута называется «Искусство программирования для ЭВМ». К тому же конечный пользователь должен оставаться специалистом в своей области, а не переквалифицироваться в профессионального программиста.

В нашей стране ситуация усугубляется еще тем, что отечественные ЭВМ отлично «помнят» свое происхождение и упорно (не без участия профессионалов) продолжают «говорить» по-английски. Некоторые программисты всерьез считают, что работа на англо-говорящих ЭВМ приносит несомненную пользу, заставляя людей изучать иностранный язык, что повышает общий культурный уровень в стране?! Уже практически повсеместно ЭВМ сданы в утиль, а на их месте установлены «компьютеры»; загадочные (но очень благозвучные) «директории» размещаются там, где совсем недавно находились каталоги и оглавления; «флопы» успешно «бутируются», а «программы линкуются». Специальная литература и описания так называемых адаптированных (?!) программных средств служат неиссякаемым источником «чипов», «опций», «гексадесятичных чи-

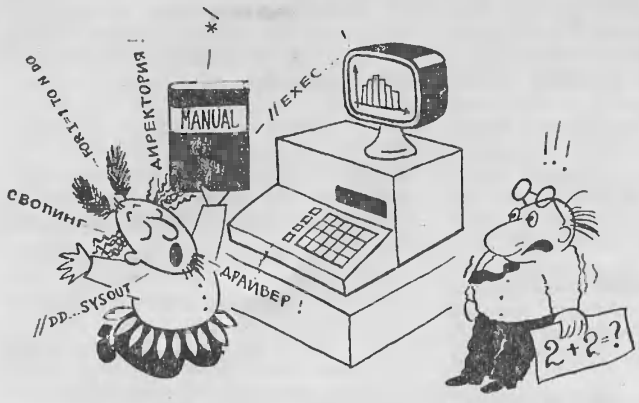


Рис. 1

сел» и им подобных «шедевров». Процессу засорения русского языка способствует и многочисленная журналистская армия, освещая в периодической печати проблемы «компьютеризации».

Некоторые программисты-профессионалы и ученые от программистской науки должны, наконец, понять, что «изобретение» (часто не оправданное) очередного специального термина укладывает еще один блок в стену, разделяющую программистов и конечных пользователей (рис. 1, иллюстрации к статье выполнены художником У. Колесовой).

АПЛ обладает чрезвычайно низким порогом освоения и не требует от программиста никаких знаний об аппаратуре вычислительного комплекса или о правилах взаимодействия с операционной системой. Даже ребенок за 5 мин осваивает элементарные возможности АПЛ-системы (рис. 2). Общение с АПЛ-системой может осуществляться на национальном языке.

Вторая категория пользователей, которым АПЛ может быть очень полезен,— это программисты, создающие приложения по заказам конечных пользователей. От результативности их деятельности во многом зависит эффективность использования парка ЭВМ. В фундаментальном исследовании [5] убедительно доказывается, что в ближайшие годы необходимо поднять производительность труда программистов, занимающихся реализацией приложений, не менее чем в 50 раз. Производительность труда напрямую связана с затратами на создание программных продуктов. В США затраты в гражданских областях уже составляют более 10, а в военных — более 75 долл. в расчете на одну машинную команду полностью завершенной разработки [6].

Сторонники традиционных языков программирования связывают свои надежды в отношении повышения производительности труда прикладных программистов с применением структурных методов. Но трезвый анализ показывает, что никакая технология программирования не способна сама по себе повысить производительность труда более чем на 50 %, если не перейти к использованию языка более высокого уровня [7, 8].

АПЛ обеспечивает десятикратный рост производительности труда программистов при реализации приложений по сравнению с такими языками, как КОБОЛ или ПЛ/1 [5]. Главная причина этого роста: АПЛ позволяет сконцентрировать все силы программиста на том, что он хочет сделать на ЭВМ, в отличие от традиционных языков, которые постоянно ставят его перед проблемой, как это запрограммировать (рис. 3).

АПЛ может с успехом использоваться там, где традиционные компилируемые языки просто не годятся: во-первых, при создании пользовательских систем, плохо поддающихся формализации (АСУ — широко и печально известный представитель систем этого класса), и, во-вторых, для быстрого поиска ответов на вопросы типа «а что, если...».

Неудачи внедрения АСУ стали уже притчей во языцех. Большинство АСУ «умирает» в момент сдачи их в эксплуатацию. Одна из главных причин постоянных неудач — использование традиционной схемы реализации таких систем, основанной на априорном предположении, что формальные требования к любой АСУ могут быть сформулированы заранее. Действительно, требования к системе управления, например движением ракеты, можно и должно предъявлять до начала программирования. Иное дело системы, предназначенные для автоматизации неформальной деятельности людей. Большинство организационно-экономических АСУ, ориентированных на человеческие нужды, гибнет в формальных категориях.

Наличие четких и исчерпывающих требований к будущей системе, оформленных в виде Технического задания,— необходимое условие для начала программирования на традиционных языках типа КОБОЛа или ПЛ/1. Однако, как показывает практика, полные и конструктивные требования не могут быть сформулированы заранее: программисты плохо представляют себе нужды конечных пользователей, а последние, в свою очередь, не знакомы с возможностями ЭВМ и склонны преувеличивать или преуменьшать их. Более 80 % ошибок в системе связано с неправильно сформулированными требованиями и, как следствие, с ошибками в проекте [5].

Никакое Техническое задание не способно предусмотреть все возможные социальные и организационные последствия перехода к новым методам управления. Внедрение АСУ изменяет технологию работы людей, делает доступной информацию, сбор которой ранее был затруднен, рвет старые и устанавливает новые деловые связи между работниками, изменяя их статус. Все это ведет к тому, что требования к системе не являются стабильными. Решение задачи изменяет саму задачу! Конечные пользователи, накапливая опыт работы с системой, начинают лучше понимать, что́ именно они хотели бы иметь, и требуют от разработчиков внесения соответствующих модификаций в программы, настаивая на том, чтобы это происходило мгновенно.

Единственное разумное решение этой проблемы — отказ от традиционных языков и методов программирования в пользу автоматизированных программных средств, которые способны привлечь к разработке автоматизированных систем электронной обработки данных самих конечных пользователей.

Практически во всех областях техники при реализации сложных проектов создаются макеты — относительно дешевые прототипы будущих конструкций. В программировании они почти не используются, так как применение традиционных языков программирования делает макет очень дорогим, а на его реализацию уходит почти столько же времени, что и на создание самой системы. АПЛ неоднократно и с успехом использовался для построения прототипов больших и сложноформализуемых систем. Он позволяет запрограммировать макет за несколько дней, в крайнем случае недель. Совместная работа программиста и конечного пользователя за терминалом ЭВМ при соз-

Рис. 2

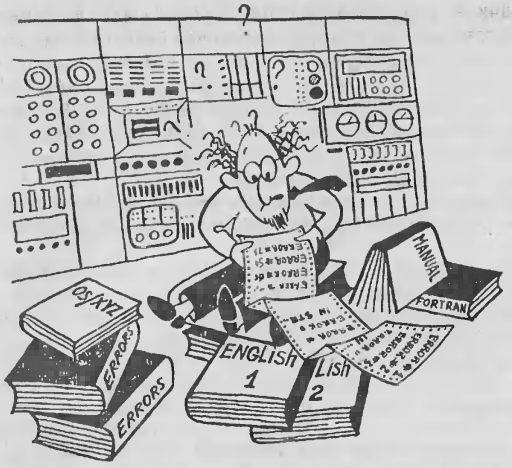


Рис. 3

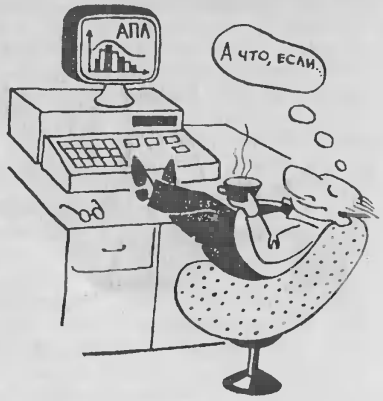


Рис. 4

дании прототипа (вещь, совершенно невозможная при программировании на ПЛ/1) дает им возможность лучше понять конечную цель. Параллельно решается задача обучения конечных пользователей. Построенный прототип становится спецификацией будущей АСУ. Известны случаи, когда такого рода системы полностью создавались на АПЛ [9—11]. Использование АПЛ дает возможность конечным пользователям самостоятельно вносить необходимые изменения в систему, если в этом есть потребность. Профессиональные программисты привлекаются только в особо сложных случаях.

Что касается решения задач, отвечающих на вопросы типа «а что, если...», то здесь у АПЛ просто нет серьезных конкурентов. Компилируемые языки непригодны из-за непомерно большого времени разработки программ, а другие интерпретируемые языки не обладают необходимой мощностью и гибкостью. В планировании своей финансовой деятельности на АПЛ сделали ставку такие крупные западные фирмы, как ICI, Shell, Xerox, STSC [11] и др. Практически вся внутренняя информация фирмы IBM обрабатывается на АПЛ [9].

Многие профессионалы скептически относятся к АПЛ, считая его непригодным для решения задач системного программирования. Конечно, системная программа должна быть написана на языке, более эффективном, чем АПЛ. Однако моделирование работы создаваемого системного модуля можно провести на АПЛ, что уменьшит время разработки и повысит ее надежность. Некоторые фирмы-разработчики ПО успешно применяют АПЛ для этих целей.

Несколько слов об использовании АПЛ в обучении. Авторы придерживаются точки зрения тех, кто считает, что необходимо учить потенциальных пользователей тому, как применять ЭВМ в их практической деятельности, и лишь в самых крайних случаях «опускаться» до изучения конкретного языка программирования. В конце концов, намного важнее научить грамотно ставить задачу ЭВМ, чем заставить вызубрить десяток-другой операторов языка БЕЙСИК, практическое использование которых для многих останется тайной. Осознание того, что машина может в принципе, а чего не сможет никогда — главная цель «компьютерного ликбеза». Для решения этой задачи наилучшим образом, как считают некоторые, подходят обучающие среды, которые общаются с человеком на языках существенно более высокого уровня, чем БЕЙСИК или АПЛ, не позволяя деревьям синтаксических конструкций заслонить леса практических целей. АПЛ — почти идеальное средство для создания таких систем силами самих педагогов, а соответствующий опыт накоплен не только за рубежом, но и у нас в стране [12, 13].

Авторы, безусловно, не предлагают заменить языком АПЛ все другие языки программирования. Все языки имеют право на существование. Однако каждому машинному языку предназначена своя «экологическая ниша» — свой круг задач, при решении которых он наиболее эффективен. АПЛ, не лишенный недостатков, является, по мнению многих, универсальным средством персональных вычислений. Он лучший инструмент для тех, кто использует ЭВМ не столько в качестве «вычислителя», сколько в качестве «думателя» (рис. 4).

Что такое АПЛ?

Язык АПЛ, предложенный в 1962 г. доктором математики К. Э. Айверсоном [14], обладает такими качествами, что его сравнение практически с любым другим языком программирования просто некорректно: АПЛ не принадлежит ни к одному поколению из существующих машинных языков [15]. По утверждению Дж. Мартина, АПЛ — язык сверхвысокого уровня, единственный пока кандидат в языки пятого поколения.

Прежде всего, необычна сама форма языка: АПЛ-программы не содержат никаких «ритуальных» слов типа BEGIN, REAL, DO и им подобных. Зато в АПЛ используются специальные значки для обозначения встроенных функций языка. (Многие из этих значков отсутствуют на стандартной клавиатуре, что затрудняет распространение АПЛ в СССР.) Лексика АПЛ интернациональна, так как не привязана ни к одному из естественных языков; корни АПЛ — в обычной математической записи.

В АПЛ не существует описаний типов данных. АПЛ-система сама следит за корректностью использования информации, осуществляя необходимые преобразования, если это требуется. Это несколько снижает эффективность АПЛ-программ по сравнению с компилируемыми языками, но устраняет источник возникновения очень неприятных и трудно «вылавливаемых» ошибок. Любому, кто хоть раз пытался объяснить человеку, далекому от программирования, различие между записями 2 и 2.0 в языке Фортран, понятно, насколько это тяжело сделать. Ведь с точки зрения многих людей, осваивающих программирование, «роза пахнет розой, хоть розой назови ее, хоть нет». АПЛ не требует понимания такого рода различий, возлагая эту задачу на ЭВМ. Программист, работающий с АПЛ, имеет дело только с числами или символами.

Мощность АПЛ заключается не в обилии различных синтаксических конструкций, а в огромном числе синтаксически эквивалентных встроенных функций. Понятие функции — единственное базовое понятие языка АПЛ. АПЛ обладает рядом непроцедурных свойств. В отличие от процедурных языков, которые полностью детализируют процесс обработки информации, непроцедурные языки лишь указывают, что именно необходимо сделать, оставляя самой машине выбор способа достижения конечной цели. Эти свойства языка освобождают программиста от рутинной работы, позволяя ему мыслить крупными блоками, и резко сокращают длину программ. АПЛ — программы в десятки и сотни раз короче, чем аналогичные, написанные на тради-

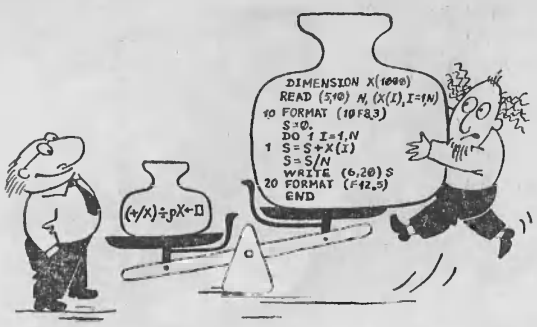


Рис. 5

ционных языках. Средняя длина АПЛ-программ не превышает 10 строк. На рис. 5 представлены две эквивалентные программы на АПЛ и Фортране, которые вычисляют среднее значение вектора числа по формуле: $S=1/N \sum_{i=1}^{N} X_i$. Заметим, что в отличие от АПЛ-программы, которая полностью готова к исполнению АПЛ-системой, программа на Фортране еще нуждается в компиляции и компоновке, что требует от конечного пользователя соответствующих знаний.

АПЛ лучше других языков приспособлен для программирования процессов обработки массивов данных. В нем разрешено делать все то (и даже сверх того), что допустимо в линейной алгебре, например хорошо известное и часто встречающееся на практике матричное умножение — частный случай «внутреннего произведения» АПЛ, которое позволяет строить около 400 различных скалярных произведений. Такие средства АПЛ, способные порождать новые функции, называются, как и в математике, операторами (не путать с операторами языков программирования). Всего с учетом возможностей, предоставляемых операторами, в АПЛ имеется до 1000 встроенных функций, которые покрывают бо́льшую долю потребностей программиста. Большинство АПЛ-программ удается писать без циклов. Эти свойства языка всегда привлекали разработчиков программного обеспечения для ЭВМ параллельных архитектур. Многие языки для таких машин напоминают АПЛ [16, 17].

АПЛ-система — это комплекс программных и аппаратных средств, поддерживающий весь цикл разработки и эксплуатации АПЛ-программ и обеспечивающий доступ пользователя ко всем программным и аппаратным ресурсам вычислительного комплекса. АПЛ-система — это интегрированная вычислительная система, объединяющая в рамках единой идеологии все необходимые пользователю средства.

Взаимодействие программиста с АПЛ-системой очень напоминает работу человека с листом бумаги, карандашом и калькулятором. Различие только в том, что емкость листа, называемого рабочей областью, достигает иногда сотен килобайт, а возможности «калькулятора» совпадают с возможностями ЭВМ. Роль карандаша выполняет клавиатура дисплея. Программист может работать сразу со многими программами и массивами данных в отличие от БЕЙСИК-системы, которая имеет дело только с одной программой. Массивы данных — самостоятельные объекты рабочей области и не зависят от АПЛ-программ. Рабочие области можно хранить в АПЛ-библиотеке. Точно так же, как человек заглядывает в справочник, когда ему необходимо вспомнить нужное соотношение, пользователь АПЛ-системы переписывает в свою рабочую область нужную программу или массив данных из АПЛ-библиотеки. АПЛ-система может как запоминать результаты вычислений, так и работать в режиме счета на «промокашке», когда результаты не сохраняются, а лишь выводятся на экран.

Программист общается с АПЛ-системой с помощью так называемых системных команд, которые не являются частью языка АПЛ, поэтому диалог может осуществляться на любом национальном языке, что не влияет на переносимость самих АПЛ-программ. При некорректных действиях пользователя или при обнаружении ошибки во время исполнения программы система выдает вразумительное сообщение и точно указывает место ошибки.

Специальные отладочные режимы работы АПЛ-системы значительно упрощают процесс тестирования создаваемых программ.

Ограниченный объем журнальной статьи не позволяет подробно остановиться на свойствах АПЛ. Всем, кому он показался достойным для дальнейшего ознакомления, авторы рекомендуют книгу [18], которая заслуженно считается лучшим учебником по АПЛ.

АПЛ-машина

Сама идея микроЭВМ со встроенной АПЛ-системой не нова. Например, непосредственной предшественницей ПЭВМ типа IBM 5150, больше известной среди программистов как IBM PC, была настольная ЭВМ модели IBM 5100, являвшаяся АПЛ-машиной.

АПЛ-машина — не принципиально новая ЭВМ, а набор серийных и доработанных устройств. Ее основа — однoплатная микроЭВМ типа МС1201 или аналогичная.

Репрограммируемое постоянное запоминающее устройство типа МС3405.01 (полуплата конструктива «Электроника 60») емкостью 100 Кбайт используется для хранения программы системы АПЛ/ВЦАН МИКРО. Именно это устройство превращает ДВК в АПЛ-машину. Могут быть использованы также два РПЗУ типа МС3404.04. АПЛ-система — сложная программа, емкость которой обычно превосходит 100 Кбайт. МикроЭВМ типа ДВК имеет всего 56 Кбайт оперативной памяти, что не позволяет реализовать полностью ОЗУ-резидентную систему. Поэтому в составе вычислительного комплекса обязательно должно быть внешнее запоминающее устройство с прямым доступом. Накопители на гибких магнитных дисках не могут обеспечить ни необходимого быстродействия, ни достаточной надежности при считывании в ОЗУ нужного в данный момент фрагмента АПЛ-системы. К тому же в нашей стране имеется немало ЭВМ, вообще лишенных периферийных устройств. Поэтому цель авторов — создание мощной универсальной диалоговой вычислительной системы, функционирование которой не опиралось бы на использование НГМД.

В адресном пространстве процессора в диапазоне 165000.. 165777 располагается служебная программа АПЛ-системы объемом 4 Кбайт со страничной организацией. Физически ПЗУ со служебной программой находятся на плате МС3405.01. В состав служебной программы входят: начальный загрузчик системы, драйверы некоторых устройств, программы эмулятора арифметики с числами в формате с плавающей точкой (для микроЭВМ МС1201.01) и другие программы. Для запуска АПЛ-системы достаточно передать управление по адресу 165000 (никакие системные устройства не требуются): АПЛ-система готова к использованию сразу же после подачи питания на ЭВМ. Перед выгрузкой в ОЗУ резидента АПЛ-системы осуществляется тестирование системного ПЗУ и проверка корректности хранящейся в нем информации. В процессе работы АПЛ-системы правильность считываемой информации не контролируется.

Главная проблема всех разработчиков отечественных АПЛ-систем — отображение на стандартном дисплее специальных символов языка АПЛ. Обычно они заменяются на комбинации стандартных. На такие неудобства охотно идут люди, познавшие все преимущества программирования на АПЛ, но для начинающих пользователей это может служить серьезным отталкивающим фактором. Авторы АПЛ/ВЦАН решили эту задачу путем модификации дисплея «Электроника 15ИЭ-00-013», которым комплектуются комплексы ДВК1 и ДВК2, контроллера символьного дис-

плея МС2711 (или КСМ), входящего в состав ДВК3, или дисплея СМ7209, используемого в ПЭВМ МЕРА 60.

В дисплее «Электроника 15ИЭ-00-013» устанавливается дополнительная микросхема типа К556РТ5 на плату генератора сигналов. Это однократно прожигаемое ПЗУ хранит образы больших латинских букв и АПЛ-символов. В обычном режиме работы дисплея сохраняется стандартный набор символов. В режиме АПЛ-терминала, включаемого из командной строки устройства, на месте малых латинских букв генерируются АПЛ-символы. Вся процедура доработки занимает около 20 мин.

В контроллере МС2711 перепрограммируется ПЗУ знакогенератора, выполненное на основе микросхемы К573РФ2. В результате этого дисплей ДВК3 теряет способность воспроизводить 32 псевдографических символа, на месте которых генерируются символы АПЛ. Таким же способом модифицируются контроллер МС2719 и дисплей СМ7209.

Немаловажное значение для полноценной работы пользователя АПЛ-системы имеет АЦПУ, способное отображать спецсимволы АПЛ. В качестве таких печатающих устройств могут быть применены: УВВПЧ-30-004, Robotron-6312, EPSON FX 800/1000 или D-100, на которых нестандартные символы отображаются программным путем. Авторами разработан также способ модификации достаточно распространенного устройства DZM-180. С помощью несложной переделки блока знакогенератора можно заставить это АЦПУ работать как в коде КОИ-7, так и в коде КОИ-8 (кодировка переключается). АПЛ-символы имеют восьмеричные коды в диапазоне 241...277.

В состав АПЛ-машины могут входить также необязательные, но поддерживаемые АПЛ/ВЦАН МИКРО модули: контроллер графического дисплея и мультиплексор последовательного канала для построения локальной сети ЭВМ (могут быть использованы адаптеры типа МС4608.02, МС4102 или им аналогичные).

АПЛ-система

АПЛ/ВЦАН МИКРО — это представитель семейства АПЛ-систем [19] для отечественных мини- и микроЭВМ с системой команд типа СМ4.

Язык АПЛ/ВЦАН точно соответствует проекту на стандарт языка АПЛ [20], что обеспечивает полную переносимость АПЛ-программ (с точностью до системных функций) между большинством зарубежных АПЛ-систем и АПЛ/ВЦАН. Набор системных функций АПЛ/ВЦАН состоит из стандартных функций, обязательных для реализации во всех АПЛ-системах и отражающих специфику комплексов ДВК и операционной системы RT-11.

Несмотря на то, что АПЛ/ВЦАН МИКРО — автономная система, она выдерживает внешние соглашения ОС RT-11. Объем рабочей области АПЛ-системы колеблется в зависимости от версии в пределах 37...38 Кбайт.

Пользователю предоставляется достаточно высокий уровень сервиса при работе с клавиатурой АПЛ-машины: поддерживается экранное редактирование вводимых строк, используется функциональная клавиатура для ввода часто встречающихся системных команд, имеется возможность вернуть на экран последнюю введенную строку для ее последующего редактирования. АПЛ-программы создаются и редактируются с помощью экранного редактора АПЛ-системы. Кроме того, программист может взять на себя управление экраном и клавиатурой системного дисплея. Это позволяет достаточно просто программировать различные «меню», экранные редакторы и т. п.

При реализации внешней формы диалога с АПЛ-системой авторы стремились точно следовать книге [18], чтобы она могла стать, наряду с системной документацией, пособием по программированию на АПЛ-машине. Весь диалог пользователя с АПЛ-системой осуществляется на русском языке.

АПЛ/ВЦАН МИКРО поддерживает работу с файловой системой RT-11 на устройствах внешней памяти, работающих с гибкими магнитными дисками диаметрами 203 мм (одинарная плотность) и 133 мм (одинарная и двойная плотности). Дисковые драйверы написаны М. И. Потемкиным. Использование стандартной файловой системы дает возможность обрабатывать в АПЛ-системе файлы, созданные другими программными средствами, и, наоборот, файлы, организованные в АПЛ-системе, можно использовать в других прикладных программах. Многие пользователи АПЛ/ВЦАН с успехом применяют это на практике. АПЛ-библиотеки также реализуются в рамках стандартной файловой системы. Компактность АПЛ-программ позволяет хранить на одном гибком диске с одинарной плотностью записи до 400 программ среднего размера.

В состав АПЛ-машины входит сменное РПЗУ пользователя емкостью 32 Кбайт. Оно предназначено для «зашивки» прикладных программ, написанных на АПЛ. Хранение «внутри» ЭВМ не только системного, но и прикладного программного обеспечения позволяет в ряде случаев вообще отказаться от НГМД. Эти возможности могут быть использованы при построении учебных классов на базе сети АПЛ-машин.

Графические средства, поддерживаемые АПЛ-машиной, включают примитивы построения точек, отрезков прямых и дуг эллипсов, а также примитивы закраски замкнутых контуров, операций с «окнами», вывода символьной информации и прямого доступа к графическому экрану. Если в состав вычислительного комплекса входит АЦПУ, способное выводить графическую информацию, то возможно получение твердой копии экрана графического дисплея. Графика поддерживается на черно-белом контроллере ДВК3, имеющем растр 286×400, или на 8-цветном контроллере МС4702 с растром 256×256.

Для работы с периферийными устройствами, которые не поддерживаются системно, или для реализации непредусмотренных разработчиками средств возможно исполнение АПЛ-системой машинных программ, написанных по определенным соглашениям. Такая программа — обычный числовой массив, находящийся в рабочей области АПЛ-системы, который может быть сформирован средствами АПЛ или прочитан с диска. Практически эта возможность позволяет программисту неограниченно расширять набор системных функций, а также использовать АПЛ-машину для управления процессами в реальном масштабе времени.

АПЛ-машина может быть применена в качестве интеллектуальной терминальной станции в сети ЭВМ. Для этой цели предназначены системные средства, обеспечивающие работу в режиме эмуляции терминала, реализующие байтовый обмен и поддерживающие прием-передачу массивов данных по линии связи. Обмен информацией осуществляется по последовательному каналу, который со стороны ДВК может быть организован, например с помощью адаптера МС4608.02. В режиме эмуляции терминала пользователь АПЛ-машины превращается в пользователя центральной ЭВМ, ресурсы АПЛ-машины при этом ему временно становятся недоступными. Байтовый обмен дает возможность организовать сетевой протокол и служит для целей синхронизации обмена между абонентами. Системные средства, поддерживающие прием-передачу массивов, позволяют не только принимать или передавать информацию по линии связи, но и осуществлять необходимые преобразования структуры и типа данных со стороны АПЛ-машины.

Если АПЛ-машина подключена по терминальной линии к ЭВМ типа СМ4/СМ1420, функционирующей под управлением ОС TSX, то возможно использование распределенной файловой системы. Распределенная файловая система — это пакет программ, поддерживающий работу пользователя АПЛ-системы с внешними устройствами центральной ЭВМ. С точки зрения программиста, обмен с внешними устройствами центральной ЭВМ ведется так, как будто они принадлежат самой АПЛ-машине. Со стороны СМ ЭВМ обмен поддерживает специальная программа связи объемом 4 Кбайт (всего для ее работы требуется

10 Кбайт), которая в нужный момент автоматически стартуется АПЛ-машиной по линии связи. Распределенная файловая система дает возможность владельцам микроЭВМ ДВК1 использовать ее не только в качестве дисплея, как это вынуждено делать большинство, но и в качестве полноценной ЭВМ. По своим функциональным возможностям эти средства похожи на те, что предоставляются программисту DD-драйвером RT-11. Отметим, однако, что распределенная файловая система работает более эффективно, так как по линии передается только полезная информация, а сама работа с внешними устройствами осуществляется средствами СМ ЭВМ.

В работах по реализации АПЛ-машины прямо или косвенно принимало участие большое число людей. А. Ф. Ковалев потратил массу личного времени на всестороннее тестирование АПЛ/ВЦАН. Благодаря его рекомендациям авторы существенно повысили быстродействие системы. Большая группа пользователей во главе с В. М. Куприяновым и А. О. Скомороховым оказала огромную помощь по выявлению и устранению ошибок в системе. Ряд ценных идей, реализованных в системе, был предложен Н. И. Пунтиковым и А. И. Мирошниковым.

Трехлетний опыт работы показал все преимущества превращения ДВК в АПЛ-машину. Во-первых, многократно возрастает реальная отдача от использования ЭВМ. По уровню сервиса, потенциальным возможностям, легкости освоения и быстродействию АПЛ-машина не имеет аналогов среди других универсальных диалоговых систем, способных функционировать на ДВК. Во-вторых, АПЛ-машина работает существенно более надежно, чем ДВК, так как из постоянной работы исключается самый ненадежный элемент вычислительного комплекса — НГМД. Обращения к дискам крайне редки, а ошибки не имеют катастрофических последствий в отличие от сбоев, возникающих при использовании дисковых программных средств.

В настоящее время несколько государственных и кооперативных предприятий совместно ведут разработку нового системного устройства АПЛ-машины, превосходящего по функциональным возможностям конструкцию, описанную выше.

Для АПЛ-машины уже разработано немало прикладных программ: пакет статистической обработки и планирования эксперимента; различные системы обработки данных, ориентированные на использование непрофессионалами [21]; генератор отчетов и экранный редактор форм и многое другое. В нескольких высших и средних специальных учебных заведениях работают компьютерные классы, преподавание в которых ведется на основе использования АПЛ. Группой ленинградских программистов реализован программный интерфейс АПЛ-системы с системой РБД-МИКРО.

Распространение системы осуществляет научно-технический кооператив «Диалог» (249020, Обнинск Калужской обл., ул. Курчатова, 21).

Телефон 135-25-98, Москва, Кондрашев Андрей Владимирович

ЛИТЕРАТУРА

1. Lange C. A New Dawn for APL // Datamation, September 1983.—P. 129–133.
2. Bensimon J. STSC APL*PLUS and IBM PC APL Two APLs for the IBM PC//Byte, March 1984.— p. 246—264.
3. Малашинин И. И., Коннонов А. И. Реализация АПЛ на ЕС ЭВМ — В кн.: Прикладная информатика — М.: Финансы и статистика.— 1981.— Вып. 1 — С. 155—170.
4. Кондрашев А. В. Реализация АПЛ на СМ ЭВМ// Программирование.— 1986.— № 1.— С. 55—62.
5. Martin J. Application Development Without Programmers — Savant Institute, January 1981.
6. Orwick G. T. The Plight of Programming // Computerworld, September 1980.
7. Jones T. C. The Limits of Programming Productivity, Guide and Share Application Development Symposium.— New York, 1979.
8. Jones T. C. Optimizing Program Quality and Programmer Productivity, Proceedings of GUIDE 45-Atlanta, November 1977.
9. Bradshaw R. Application Development Productivity Within IBM Information System, Guide and Share Application Development Symposium.— New York, 1979.
10. Robinson R. S. Financial Planning Application of APL in J. RayMcDermott, in APL in Practice-Wiley, 1980.
11. Fisc R. S. What if: The Making of a Vice President of Finance, in APL in Practice-Wiley, 1980.
12. Iverson K. E. Introducing APL to Teachers.— APL Press.— 1976.
13. Лукша О. П., Реймаров Г. А. Обучение исследованию с использованием вычислительной системы АПЛ — В кн.: Улучшение подготовки, переподготовки и усиление роли инженерных кадров в условиях НТП. — М.: ВИМИ, 1986.— С. 21—23.
14. Iverson K. E. A Programming Language, Wiley, 1962.
15. Cherlin E. M. APL Is Not a Programming Language // APL Market News.— 1985.— Vol. 17.— N 3.— P. 1—2.
16. Бульонков М. А., Быстров А. В., Дудоров Н. Н., Котов В. Е. Предварительное описание языка БАРС.— Препринт.— Новосибирск, 1985.— 44 с.— ВЦ СО АН СССР, № 556.
17. Аверин С. В., Игуменова И. А., Ильинская З. С., Шевелев С. Л. Интерактивная система обработки изображений «Образ».— Препринт.— Новосибирск, 1983.— 31 с.— ВЦ СО АН СССР, № 429.
18. Гилман Л., Роуз А. Курс АПЛ. Диалоговый подход — М.: Мир, 1979.
19. Кондрашев А. В. Диалоговая вычислительная система АПЛ/ВЦАН — М.: ВЦ АН СССР, 1985.
20. Falkoff A. D., Orth D. L. Development of an APL Standard // APL Quote Quad.— 1979.— Vol. 2.— P. 415—453.
21. Кондрашев А. В., Соколов В. Н. АПЛ-технология: пользовательские языки и синтез программ // Управляющие системы и машины.— 1988.— № 4.— С. 82—86.

Статья поступила 16.12.87.

Гонорар за статью перечислен авторами во Всесоюзный детский фонд им. В. И. Ленина.

УДК 681.3

В. В. Медведев

АНАЛОГОВЫЙ ИНТЕРФЕЙС ПЭВМ АГАТ

Применение ПЭВМ для хранения и обработки непрерывных сигналов предполагает использование аналого-цифровых преобразователей. Для медленно меняющихся сигналов задачи дискретизации и квантования непрерывного сигнала применительно к ПЭВМ АГАТ, в комплекте которой отсутствует аналого-цифровой преобразователь, могут быть решены с помощью таймера. В составе ПЭВМ имеются два однотактных таймера КР1006ВИ1, включенных по схеме одновибраторов. В штатном исполнении они предназначены для преобразования в цифровой код сигналов от двух игровых аналого-цифровых пультов. Ниже описаны аппаратные и программные средства для преобразования и записи непрерывного по времени и уровню сигнала в ОЗУ ПЭВМ АГАТ для хранения и дальнейшей обработки. Реализация такого интерфейса не требует изменения схемы включения таймера в

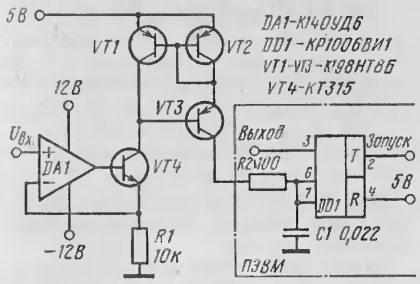


Рис. 1. Принципиальная схема устройства ввода аналогового сигнала

ПЭВМ. Аппаратная часть интерфейса соединяется с ПЭВМ через разъем «Пульт» на задней стенке системного блока.

Аналого-цифровые пульты идентичны и каждый включает в себя переменный резистор и кнопку. Резистор является внешним элементом времязадающей RC-цепи и обеспечивает переменную длительность генерируемого одновибратором импульса $T_и = 1{,}1 \times R(\alpha) C_1$, где $T_и$ — длительность импульса одновибратора; $R(\alpha)$ — сопротивление переменного резистора как функция угла α поворота подвижного контакта; C_1 — емкость RC-цепи.

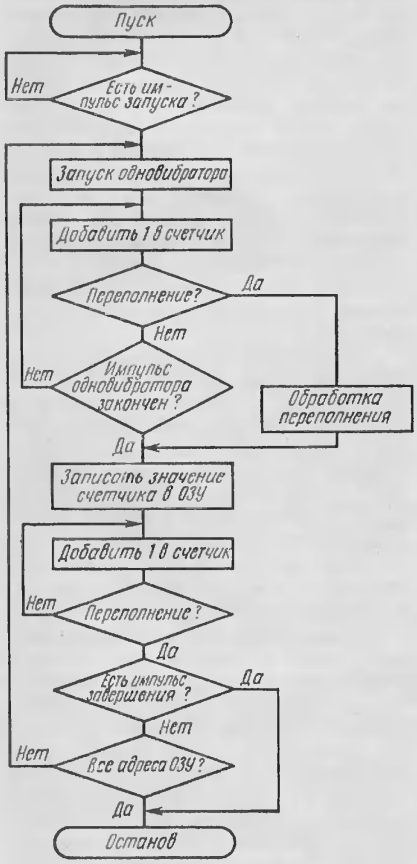


Рис. 2. Алгоритм записи сигнала в ОЗУ по входу одного из пультов

```
;
7ϕϕϕ   A9 ϕϕ      LDA #¤ϕϕ     ;ЗАГРУЗИТЬ
7ϕϕ2   85 ϕϕ      STA ¤ϕϕ      ;  НАЧАЛЬНЫЙ
7ϕϕ4   A9 1ϕ      LDA #¤1ϕ     ;  АДРЕС
7ϕϕ6   85 ϕ1      STA ¤ϕ1      ;
7ϕϕ8   Aϕ ϕϕ      LDY #¤ϕϕ     ;НАЧАЛЬНОЕ ЗНАЧЕНИЕ ИНДЕКСА
7ϕϕA   A2 ϕϕ      LDX #¤ϕϕ     ;ОБНУЛИТЬ  СЧЕТЧИК
7ϕϕC   2C 61 Cϕ   BIT ¤Cϕ61    ;ЦИКЛ-ОЖИДАНИЕ СТАРТОВОГО
7ϕϕF   1ϕ FB      BPL ¤7ϕϕC    ;  ИМПУЛЬСА СО ВХОДА КНОПКИ
7ϕ1ϕ   8D 7ϕ Cϕ   STA ¤Cϕ7ϕ    ;ЗАПУСТИТЬ  ОДНОВИБРАТОРЫ
7ϕ13   E8         INX          ;ДОБАВИТЬ ЕДИНИЦУ В СЧЕТЧИК
7ϕ14   Fϕ 31      BEQ ¤7ϕ37    ;ПРИ ПЕРЕПОЛНЕНИИ ПЕРЕЙТИ НА П/П
7ϕ16   2C 64 Cϕ   BIT ¤Cϕ64    ;УСТАНОВИТЬ СТ. БИТ В ПРИЗНАК
7ϕ19   3ϕ F8      BMI ¤7ϕ13    ;НЕ  НУЛЬ - ПРОДОЛЖИТЬ  ЦИКЛ
7ϕ1B   8A         TXA          ;ЗАПИСАТЬ СОДЕРЖИМОЕ
7ϕ1C   91 ϕϕ      STA (¤ϕϕ),Y  ;  СЧЕТЧИКА В ПАМЯТЬ
7ϕ1E   2C 64 Cϕ   BIT ¤Cϕ64    ;ЦИКЛ„ДО ПЕРЕПОЛНЕНИЯ
7ϕ21   Fϕ ϕϕ      BEQ ¤7ϕ23    ;  В СЧЕТЧИКЕ"
7ϕ23   E8         INX          ;
7ϕ24   Dϕ F8      BNE ¤7ϕ1E    ;
7ϕ26   C8         INY          ;УВЕЛИЧИТЬ ИНДЕКС
7ϕ27   Dϕ E7      BNE ¤7ϕ1ϕ    ;СТ. БАЙТ АДРЕСА  НЕ УВЕЛИЧИВАТЬ
7ϕ29   2C 61 Cϕ   BIT ¤Cϕ61    ;ЕСТЬ ИМПУЛЬС ОКОНЧАНИЯ ?
7ϕ2C   1ϕ ϕ8      BPL ¤7ϕ36    ;ДА, ЗАКОНЧИТЬ
7ϕ2E   E6 ϕ1      INC ¤ϕ1      ;НЕТ, СТ. БАЙТ АДРЕСА УВЕЛИЧИТЬ
7ϕ3ϕ   A5 ϕ1      LDA ¤ϕ1      ;  И СРАВНИТЬ  ЕГО  С  КОНЕЧНЫМ
7ϕ32   C9 7ϕ      CMP #¤7ϕ     ;КОНЕЧНЫЙ  АДРЕС  ДОСТИГНУТ ?
7ϕ34   Dϕ DA      BNE ¤7ϕ1ϕ    ;НЕТ, ПРОДОЛЖИТЬ
7ϕ36   6ϕ         RTS          ;ЗАКОНЧИТЬ ПРОГРАММУ
```

Рис. 3. Программа поддержки аналогового интерфейса ПЭВМ АГАТ

Заменяя пульты устройством для ввода аналогового сигнала, необходимо обеспечить зависимость зарядового тока времязадающего конденсатора от напряжения на входе (рис. 1).

Преобразование напряжения в длительность импульса начинается после подачи на вход «Запуск» таймера низкого логического уровня. На выходе таймера формируется фронт выходного импульса и начинается заряд конденсатора C_1, который продолжается до тех пор, пока напряжение на конденсаторе не достигнет величины $2U_п/3$, где $U_п$ — напряжение питания таймера. На выходе формируется срез импульса, а конденсатор разряжается по цепи разряда таймера (вывод 7). Длительность генерируемого импульса $T_и = 2C_2R_1U_п/3U_{вх}$.

Программная поддержка аналогового интерфейса обеспечивает запуск одновибратора, вычисление длительности импульса, генерируемого одновибратором, запись значения в ОЗУ (рис. 2) и постоянство периода дискретизации. Преобразование сигнала начинается при появлении фронта импульса запуска программы, о чем можно судить, анализируя старший бит по адресу ¤C061 (здесь и далее знак ¤ предшествует числу, записанному в шестнадцатеричной форме). После запуска одновибратора, при выполнении любой операции с адресом ¤C070, измеряется длительность генерируемого импульса и записывается значение по текущему адресу в ОЗУ (рис. 3). О существовании высокого уровня на выходе одновибратора можно судить по состоянию старшего бита по адресу ¤ C064.

Переполнение счетчика и переход на подпрограмму обработки переполнения проводятся в цикле. Такая ситуация может возникнуть, если длительность импульса не согласована со скоростью опроса. Дополнительный цикл «до переполнения в счетчике» доводит содержимое счетчика до максимального значения независимо от того, какое число было в нем (независимо от ширины имульса). Это необходимо для достижения равномерности дискретизации при условии, если длительности одного шага циклов равны. Период дискретизации для программы на ассемблере ПЭВМ АГАТ составляет 2,8 мс. Максимальная частота в спектре аналогового сигнала не превышает 175 Гц.

В программе поддержки интерфейса для записи сигнала отводится адресное пространство ¤ 1000...¤6FFF. Сама программа располагается с адреса ¤7000 и занимает в ОЗУ 55 байт. Подпрограмма обработки переполнения начинается с адреса ¤7037 и здесь не приводится. Отведенное адресное пространство заполняется сигналом за 96 с. Время записи можно увеличить, увеличивая период дискретизации введением одного или нескольких пустых операторов NOP.

При незначительных изменениях в программе можно вводить одновременно два аналоговых сигнала, используя адреса ячеек второго аналого-цифрового пульта: ¤ C062 — состояние кнопки 2, C070 — запуск одновибраторов (один адрес для обоих одновибраторов), ¤C065 — выход одновибратора 2. Быстродействие при этом упадет вдвое. Оператор перехода на программу отработки переполнения можно исключить, если максимальная

* Коломбет Е. А. Таймеры.— М.: Радио и связь, 1983.

```
ϕϕϕ   A9 81      LDA #¤81      ;ЗАГРУЗИТЬ НАЧ. ЗНАЧЕНИЕ
ϕϕ2   18         CLC           ;ОЧИСТИТЬ ПРИЗНАК ПЕРЕНОСА
ϕϕ3   8D 7ϕ CϕSTA ¤Cϕ7ϕ      ;ЗАПУСТИТЬ ОДНОВИБРАТОРЫ
ϕϕ6   ED 64 CϕSBC ¤Cϕ64      ;--
ϕϕ9   ED 64 CϕSBC ¤Cϕ64      ; !
ϕϕC   ED 64 CϕSBC ¤Cϕ64      ; !
  .     .  .  .   .   .       !255 ОПЕРАТОРОВ
  .     .  .  .   .   .       !
  .     .  .  .   .   .       !ПРЕОБРАЗОВАТЬ ШИРИНУ
  .     .  .  .   .   .       ! ИМПУЛЬСА В КОД
3ϕ7   ED 64 CϕSBC ¤Cϕ64      ;--
3ϕA   9ϕ ϕ7     BCC ¤312      ;--
3ϕC   ϕA         ASL           ; !
3ϕD   E9 ϕ1      SBC #¤ϕ1      ; !
3ϕF   4C ϕA ϕ3   JMP ¤316      ; !ДЕКОДИРОВАТЬ
312   38         SEC           ; !
313   E9 81      SBC #¤81      ; !
315   ϕA         ASL           ;--
316   91 ϕϕ      STA (¤ϕϕ),Y   ;СОХРАНИТЬ ЗНАЧЕНИЕ В ПАМЯТИ
```

Рис. 4. Фрагмент программы ускоренного преобразования

длительность генерируемого импульса не вызывает переполнения счетчика. Тогда максимальная частота в спектре входного сигнала составит примерно 225 Гц. Однако ее можно увеличить до 500 Гц, если заменить цикл опроса ячейки Д С064 последовательностью из 255 одинаковых операторов. В этом случае отпадает необходимость в дополнительном цикле и проверке на переполнение (рис. 4).

Недостаток описанного интерфейса — нелинейность характеристики преобразования. Преодолеть его без изменения схемы включения таймера в ПЭВМ весьма затруднительно.

Телефон 134-16-21, Ленинград

Статья поступила 23.12.88

УДК 681.327.5.072.2

Н. И. Пинчук, В. Д. Тищенко, С. С. Шалугин, А. К. Школяренко

ПРОГРАММНО-АППАРАТНЫЙ КОМПЛЕКС ДЛЯ ПОДКЛЮЧЕНИЯ ПЕЧАТАЮЩИХ УСТРОЙСТВ ТИПА ЕС7040 К ПЭВМ ЕС1840, ЕС1841

В состав комплекса входит модуль интерфейса ввода-вывода (МИВВ), предназначенный для согласования системных шин ПЭВМ ЕС1840, ЕС1841 с интерфейсом ввода-вывода ЕС ЭВМ, и соответствующая программная поддержка.

Технические характеристики МИВВ

Максимальная скорость передачи данных, Кбайт/с	250
Максимальное число подключаемых ПУ	4
Способ передачи данных	Программное управление или прямой доступ в память
Число портов	
ввода	2
вывода	2
Интерфейс для подключения МИВВ	
к ПЭВМ	Системная шина ПЭВМ ЕС1840 или ЕС1841
к ПУ	По ОСТ 4ГО.304.000-84
Напряжение питания, В	5
Потребляемый ток, А, не более	2,5

Наименование, формат портов ввода и вывода и их адреса приведены в табл. 1—4. Информационные и управляющие сигналы интерфейса ввода-вывода ЕС ЭВМ устанавливаются (сбрасываются) путем записи единиц (нулей) в соответствующие разряды портов вывода ШИН-К и управляющих сигналов канала. Состояние информационных и управляющих интерфейсных сигналов абонента проверяется чтением портов ввода ШИН-А и управляющих сигналов абонента. Сигналы интерфейса ввода-вывода ЕС ЭВМ (начальная выборка канала; выборка, вводимая УВУ; прием байта состояния) выполняются под управлением программы в соответствии с алгоритмом функционирования интерфейса. Данные по командам ЗАПИСЬ, ЧТЕНИЕ или УПРАВЛЕНИЕ передаются под управлением программы или по каналу прямого доступа в память, если в разряд РПДП порта вывода управляющих сигналов канала записана 1.

При переходе одного из интерфейсных сигналов ТРБ-А, УПР-А, ИНФ-А в состояние Лог. 1 запрос прерывания по линии IRQ3 формируется при условии, что в разряд РПР порта вывода управляющих сигналов канала записана Лог. 1, а в разряд РПДП — Лог. 0. Если в разряды РПР и РПДП вывода управляющих сигналов канала записана Лог. 1, то запрос прерывания формируется в результате перехода из 0 в 1 любого из сигналов УПР-А или Т/С.

Приемники и передатчики данных, приемники и передатчики сигналов управления обеспечивают согласование по уровням сигналов с интерфейсом ввода-вывода (см. рисунок). Приемники выполнены на микросхемах КР559ИП7, передатчики — на КР559ИП4. Приемопередатчик данных (D7...D0), обеспечивающий согласование сигналов МИВВ с системной шиной ПЭВМ, реализован на ИМС КР580ВА86. Дешифратор команд (ИМС К555КП2) формирует сигналы выбора блока памяти ввода или вывода, ре-

Таблица 1

Разряды порта вывода ШИН-К, адрес 38ЕН

7	6	5	4	3	2	1	0
ШИН-К0	ШИН-К1	ШИН-К2	ШИН-К3	ШИН-К4	ШИН-К5	ШИН-К6	ШИН-К7

Примечание. ШИН-К0...ШИН-К7 — информационные шины канала по ОСТ 4ГО.304.000-84.

Таблица 2

Разряды порта вывода управляющих сигналов канала, адрес 38FH

7	6	5	4	3	2	1	0
РАБ-К	ВБР-К	Сигналы идентификации канала		СМС-К	БЛК-К	РПР	РПДП
		0	0	—	—		
		0	1	—	АДР-К		
		1	0	—	УПР-К		
		1	1	—	ИНФ-К		

Примечание. РАБ-К, ВБР-К, АДР-К, УПР-К, ИНФ-К, СМС-К, БЛК-К — управляющие сигналы канала по ОСТ 4ГО.304.000-84; РПР — разрешение прерывания; РПДП — разрешение прямого доступа в память.

Таблица 3

Разряды порта ввода ШИН-К, адрес 38ЕН							
7	6	5	4	3	2	1	0
ШИН-А0	ШИН-А1	ШИН-А2	ШИН-А3	ШИН-А4	ШИН-А5	ШИН-А6	ШИН-А7

Примечание. ШИН-А0...ШИН-А7 — информационные шины абонента по ОСТ 4ГО.304.000-84.

Таблица 4

Разряды порта ввода управляющих сигналов абонента, адрес 38FH							
7	6	5	4	3	2	1	0
РАБ-А	ТРБ-А	Сигналы идентификации абонента		ВБР-А	0	ОШ.ЧЕТ	Т/С
		0	0	—	—		
		0	1	—	АДР-А		
		1	0	—	УПР-А		
		1	1	—	ИНФ-А		

Примечание. РАБ-А, ТРБ-А, АДР-А, УПР-А, ИНФ-А, ВБР-А — управляющие сигналы канала по ОСТ 4ГО.304.000-84. ОШ.ЧЕТ — ошибка четности; устанавливается аппаратно при обнаружении ошибки по паритету в полученной от абонента информации; сбрасывается после начального сброса и считывания. Т/С — конец передачи ПДП; устанавливается аппаратно после окончания передачи данных по КПДП по сигналу Т/С системной шины.

гистра управления, буферной памяти при обращении к портам МИВВ. Узел прерывания устанавливает запрос на прерывание по системной шине. Блоки памяти ввода и вывода предназначены для временного хранения информации, принимаемой с ШИН-А0...ШИН-А7 (порт ввода 38FH) и передаваемой на ШИН-К0...ШИН-К7 (порт вывода 38FH). Блоки введены с целью увеличения скорости передачи данных по принципу FIFO. Емкость каждого блока — 2 байт, состав — две ИМС К589ИР12 и одна К555АГ3. Регистр управления (ИМС К555ТМ8) предназначен для хранения кода, записанного в порт 38FH. Буферная память управления (ИМС КР580ВА86) обеспечивает считывание состояния сигналов управления интерфейса ввода-вывода.

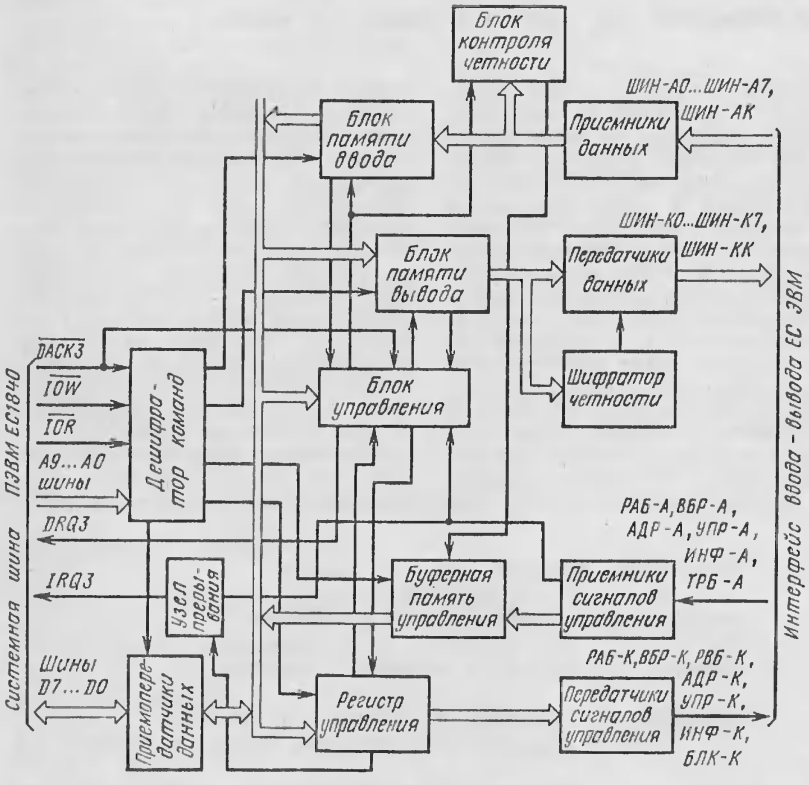
Блок контроля четности (ИМС К155ИП5) необходим для обнаружения ошибки по четности в информации, принимаемой с ШИН-А0...ШИН-А7, и установки в 1 указателя ОШ.ЧЕТ порта ввода 38FH; шифратор четности — для формирования сигнала контроля четности информационного байта, выдаваемого по ШИН-КК.

Модуль МИВВ выполнен на двусторонней печатной плате размерами 200×240 мм в стандарте ПЭВМ ЕС1840 и установлен в системный блок. На плате со стороны рычагов предусмотрен разъем, выходящий на интерфейс ввода-вывода ЕС ЭВМ. Модуль можно использовать для подключения других типов ВУ, выходящих на интерфейс ввода-вывода согласно ОСТ 4ГО.304.000-84 и обеспечивающих передачу данных со скоростью до 250 Кбайт/с.

Программное обеспечение состоит из программы вывода на печать, тестов МИВВ и ПУ. Программа вывода написана на ассемблере ПЭВМ ЕС1840, ЕС1841 и предназначена для программной поддержки вывода информации на ПУ типа ЕС7040. Объем программы 4 Кбайт. Перед запуском программы вывода оператор задает список вводимых файлов, число копий и тип кодировки. После запуска комплекс автоматически выводит файлы на печать. Программу можно использовать как команду PRINT АЛЬФА-ДОС. Отличие состоит лишь в отсутствии возможности вывода на печать символов псевдографики.

Сервисный тест МИВВ предназначен для проверки работоспособности МИВВ в шлейфовом режиме с использованием заглушек, а тест ПУ — для полной проверки ПУ и МИВВ. Оба теста написаны на ассемблере ПЭВМ ЕС1840, ЕС1841 и хранятся на дискете. Объем тестов составляет 4 и 6 Кбайт соответственно. Запускаются тесты с клавиатуры ПЭВМ, диагностическая информация выводится на экран.

Комплекс можно использовать при создании автоматизированных станций кодировки и вывода больших объемов информации, участков тиражирования и выпуска конструкторской документации, локальных сетей ПЭВМ, реализующих совместный доступ к системному ресурсу (высокоскоростному ПУ).



Структурная схема модуля интерфейса ввода-вывода

252207, Киев, пр. Академика В. М. Глушкова, 20, СКТБ ПО ИК АН УССР; тел. 266-32-25

Статья поступила 30.01.89

УДК 681.3.06

Б. Ю. Алифанов, В. Ю. Дубравин

ПРОГРАММНАЯ СИСТЕМА ГЕНЕРАЦИИ ТЕКСТА ЗАЯВКИ НА ИЗОБРЕТЕНИЕ

Важная задача ускорения НТП — автоматизация процесса оформления заявки на изобретение [1]. Предложенная в работах [2...4] система позволяет, в частности, автоматизировать формирование текста заявки и первичных документов по ней. К множеству входных данных для предложенной системы относятся признаки заявляемого технического решения (ЗТР), разделенные автором заявки на ограничительную и отличительную части [2]. Однако для некоторых классов ЗТР, к которым относятся многие устройства вычислительной техники и радиоэлектроники, количество признаков изобретения (элементов и связей между ними) весьма велико. Описать их словами сложно и утомительно, поэтому количество ошибок в описании растет и, следовательно, экспертиза замедляется из-за дополнительной переписки с ВНИИГПЭ. С другой стороны, ЗТР данного типа можно описать с помощью специального формально определенного языка, дающего возможность автоматически проверять полноту и непротиворечивость описания устройства. Предлагаемая программная система генерации текста заявки на изобретение (СГТИ) расширяет некоторые возможности системы [2...4] для определенного класса ЗТР.

Устройства, описываемые структурно-функциональными схемами (СФС), позволяют автоматизировать процесс синтеза их словесного описания на русском языке. СФС содержит структурные элементы, характеризуемые множеством «равноправных» выводов (например, резисторы, транзисторы и т. п.), и функциональные элементы, характеризуемые принципиально отличающимися друг от друга множествами входов и выходов (например, счетчики, триггеры и т. п.). СФС, с одной стороны,— это удобный для человека метод описания устройств, с другой стороны, она эквивалентна краткому формальному описанию на специальном языке — входном языке СГТИ. К устройствам, описываемым СФС, относится и широкий класс устройств вычислительной техники и радиоэлектроники.

Для представления синтаксиса **входного языка рассматриваемой СГТИ (языка описания структурно-функциональных схем)** используем нотацию Бэкуса—Наура. Устройство — это множество элементов, имеющее название:

```
<устройство> ::= <название> <элементы>

<название> ::= ( "<строка символов>" )

<элементы> ::= <элемент> | <элементы> <элемент>
```

Элемент характеризуется номером, именем и множеством выводов, которое синтаксически всегда подразделяется на множества входов и выходов, в том числе и для структурных элементов:

```
<элемент> ::= <номер> <имя элемента> : <множество входов>
                  <множество выходов> $

<имя элемента> ::= <имя> | & <имя> | + <имя>
```

Признак структурного элемента — использование знака «+» в имени элемента. С точки зрения синтеза текстового описания безразлично, относится ли вывод структурного элемента к множествам входов или выходов (следует лишь соблюдать правило, согласно которому выход любого элемента может быть соединен только со входом некоторого элемента и наоборот). Если в имени элемента используется знак «&», то данный элемент описывает выход устройства (как элемент, имеющий только один вход и не имеющий выходов) или его вход (как элемент, имеющий только один выход и не имеющий входов). При этом для описания пустого множества входов или выходов используется знак «—»:

```
<множество входов> ::= - | <входы>

<множество выходов> ::= - | <выходы>

<входы> ::= <вход> | <входы> , <вход>

<выходы> ::= <выход> | <выходы> , <выход>
```

Каждый вывод описывается заданием имени и связи (ссылки на некоторый другой вывод см. на с. 25):

Если имя вывода не задано, то при генерации текста данному выводу присваивается имя вида «первый вход» или «третий выход» и т. п. в зависимости от порядкового номера данного вывода при перечислении. Если некоторые два вывода связаны, то связь можно описать при задании первого или второго из этих выводов, или обоих одновременно (в последнем случае описания должны быть непротиворечивыми).

Таким образом, язык описания СФС позволяет в сжатой и удобной для человека форме задать все существенные признаки устройства и их названия на русском языке, достаточные для синтеза описания устройства. Дополнительные преимущества языка СФС: можно автоматически выявлять противоречия и неопределенности в описании устройства; можно получать расширенное описание устройства (листинга), более удобное для просмотра и анализа, чем описание на языке СФС, легче вносить изменения в описание на языке СФС, чем в словесное описание; значительно уменьшается объем хранимой информации и упрощается ее поиск при использовании данных [2].

Описание устройств (аналогов, про-

```
<вход> ::=  | <имя> <связь> | <связь> | <имя>
<выход> ::=  | <имя> <связь> | <связь> | <имя>
<связь> ::= <номер элемента> / <номер вывода>
<имя> ::= "<строка символов>"
```

тотипа и ЗТР) на языке СФС — это входная информация рассматриваемой СГТИ (рис. 1).

Компилятор выявляет синтаксические ошибки, неопределенности и противоречия в описаниях на языке СФС, выдает сообщения о них и транслирует правильные описания во внутреннее представление. По выходной информации компилятора формирователь листингов выдает листинги, удобные для проверки правильности исходных описаний на языке СФС. Компаратор сравнивает внутренние представления ЗТР и прототипа, разделяя описание ЗТР на ограничительную и отличительную части. Соответствие между элементами устанавливается по их именам, а между связями — по именам участвующих элементов и выводов (без учета порядковых номеров выводов и элементов). Генератор текста по внутренним представлениям синтезирует текстовые фрагменты заявки на изобретение.

СГТИ выдает следующие **фрагменты заявки на изобретение:**

по описанию аналога на языке СФС — раздел «характеристика аналога» описания изобретения;

по описанию прототипа на языке СФС — раздел «характеристика прототипа» описания изобретения;

по описанию ЗТР на языке СФС — раздел «пример конкретного выполнения» описания изобретения и основу реферата изобретения;

по описаниям прототипа и ЗТР на языке СФС — раздел «сущность изобретения» описания изобретения и формулу изобретения;

Для иллюстрации возможностей СГТИ рассмотрим два простых устройства, имитирующих прототип (рис. 2) и ЗТР (рис. 3). Описание прототипа на языке СФС будет выглядеть следующим образом:

```
("асинхронный счетчик")
1 "первый триггер":"информационный"1/2,"тактовый"3/1;
  "прямой","инверсный"$
2 "второй триггер":"информационный"2/2,"тактовый"1/2;
  "прямой","инверсный"2/1$
3 & "тактовый вход устройства":-;$
4 & "выход младшего разряда устройства":1/1;-$
5 & "выход старшего разряда устройства":2/1;-$
```

Описание ЗТР на языке СФС:

```
("синхронный счетчик")
1 "первый триггер":"информационный"1/2,"тактовый"4/1;
  "прямой","инверсный"1/1$
2 "второй триггер":"информационный","тактовый"4/1;
  "прямой"$
3 "элемент исключающее ИЛИ":1/1,2/1;2/1$
4 & "тактовый вход устройства":-;$
5 & "выход младшего разряда устройства":1/1;-$
6 & "выход старшего разряда устройства":2/1;-$
```

По описаниям ЗТР и прототипа СГТИ сгенерировала следующие фрагменты. Раздел описания изобретения — «характеристика прототипа»:

«Наиболее близким к заявляемому техническому решению является асинхронный счетчик, содержащий первый триггер, информационный вход которого соединен с инверсным выходом первого триггера, тактовый вход которого соединен с тактовым входом устройства, второй триггер, информационный вход которого соединен с инверсным выходом второго триггера, тактовый вход которого соединен с инверсным выходом пер-

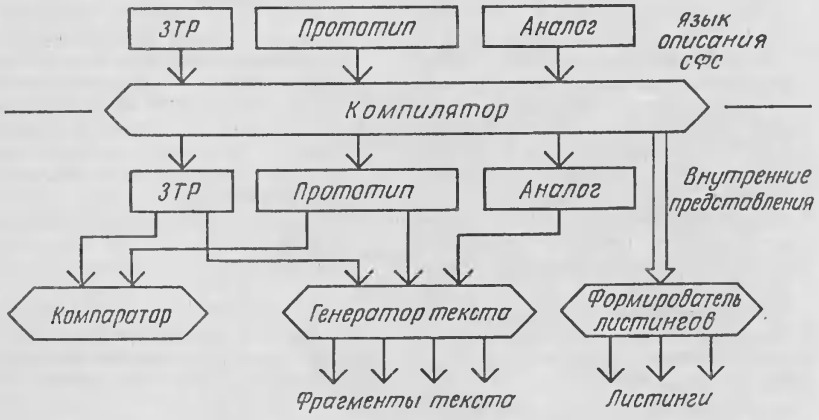


Рис. 1. Структурная схема программной системы для генерации текста заявки на изобретение (СГТИ)

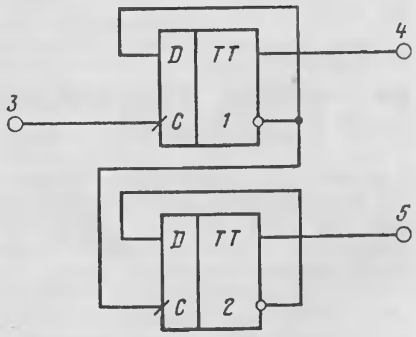


Рис. 2. Устройство, имитирующее прототип

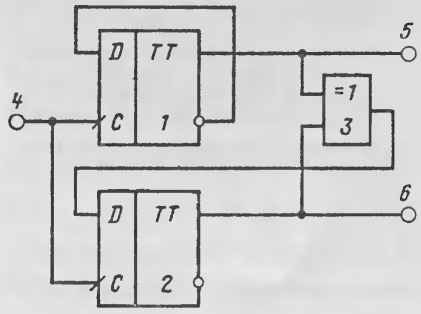


Рис. 3. Устройство, имитирующее заявляемое техническое решение

вого триггера, прямой выход которого соединен с выходом младшего разряда устройства, прямой выход второго триггера соединен с выходом старшего разряда устройства».

Раздел описания изобретения — «пример конкретного выполнения»:

«Заявляемый синхронный счетчик содержит первый триггер 1, второй триггер 2, элемент 3 исключающее ИЛИ, причем информационный вход первого триггера 1 соединен с инверсным выходом первого триггера 1, тактовый вход которого соединен с тактовым входом 4 устройства, информационный вход второго триггера 2 соединен с выходом элемента 3 исключающее ИЛИ, первый вход которого соединен с прямым выходом первого триггера 1, тактовый вход второго триггера 2 соединен с тактовым входом 4 устройства, второй вход элемента 3 исключающее ИЛИ соединен с прямым выходом второго триггера 2, выход 5 младшего разряда устройства соединен с прямым выходом первого триггера 1, выход 6 старшего разряда устройства соединен с прямым выходом второго триггера 2».

Раздел описания изобретения — «сущность изобретения»:

«Поставленная цель достигается тем, что в устройство, содержащее первый триггер, информационный вход которого соединен с инверсным выходом первого триггера, тактовый вход которого соединен с тактовым входом устройства, второй триггер, выход младшего разряда устройства соединен с прямым выходом первого триггера, выход старшего разряда устройства соединен с прямым выходом второго триггера, дополнительно введен элемент исключающее ИЛИ, первый вход которого соединен с прямым выходом первого триггера, информационный вход второго триггера соединен с выходом элемента исключающее ИЛИ, второй вход которого соединен с прямым выходом второго триггера, тактовый вход которого соединен с тактовым входом устройства».

Формула изобретения:

МКИ _____

Формула изобретения

Синхронный счетчик, содержащий первый триггер, информационный вход которого соединен с инверсным выходом первого триггера, тактовый вход которого соединен с тактовым входом устройства, второй триггер, выход младшего разряда устройства соединен с прямым выходом первого триггера, выход старшего разряда устройства соединен с прямым выходом второго триггера, отличающийся тем, что с целью ⟨цель изобретения⟩ в него дополнительно введен элемент исключающее ИЛИ, первый вход которого соединен с прямым выходом первого триггера, информационный вход второго триггера соединен с выходом элемента исключающее ИЛИ, второй вход которого соединен с прямым выходом второго триггера, тактовый вход которого соединен с тактовым входом устройства.

Начальник патентно-лицензионного отдела

Авторы

Опыт эксплуатации СГТИ показывает, что в подавляющем большинстве случаев система правильно формирует падежные окончания слов, однако ввиду отсутствия формальных правил формирования падежных окончаний естественного русского языка возможны отдельные ошибки, преимущественно при употреблении слов женского и среднего рода. После завершения работы СГТИ необходимо с помощью экранного редактора вставить в текст описания изобретения раздел «описание работы устройства», цель изобретения, индекс МКИ и сопроводительные документы по заявке.

Синтезированные СГТИ фрагменты составляют до 50 % текста описания заявки на изобретение, причем автоматизируется написание наиболее рутинных и трудоемких для человека частей.

При использовании СГТИ в составе системы [2] можно достичь еще более высокой степени автоматизации процесса оформления заявок на устройства ВТ и РЭА, чем при автономном применении данных систем.

Программное обеспечение СГТИ, написанное на языке высокого уровня Си, повышает ее мобильность. Одна из версий СГТИ функционирует в ОС Unix. Предполагается перенести СГТИ и в другие ОС: RT-11, RSX-11, VMS и др.

Телефон 534-54-71, Москва

ЛИТЕРАТУРА

1. Микропроцессорные средства и системы.— 1987.— № 4.— С. 37.
2. Сарьян В. К., Смолич Г. Г. Взаимодействие больших и малых баз данных в автоматизированной системе оформления заявок, ориентированной на индивидуальных пользователей // Микропроцессорные средства и системы.— 1987.— № 4.— С. 37—39.
3. Сарьян В. К., Корытов В. В. Увеличение времени жизни диалоговой системы по оформлению первичных документов по заявкам // Там же.— С. 40—41.
4. Блинников В. И., Минашин В. П., Бедин Б. Д., Сарьян В. К. Система безбумажной технологии делопроизводства по заявкам на изобретения // Там же.— С. 42—43.

Статья поступила 25.05.88

УДК 681. 3

А. Н. Лупенко

ПРОГРАММА ПРЕОБРАЗОВАНИЯ КОДА ГРЕЯ В ДВОИЧНЫЙ

В системах автоматизации технологических процессов для кодирования разнообразных параметров часто применяется код Грея, исключающий неоднозначность отсчета измеряемых величин в моменты смены кодов и обеспечивающий более надежную работу систем по сравнению с двоичным кодом.

Для программного преобразования кода Грея в двоичный целесообразно использовать алгоритм, описывающий работу комбинационных преобразователей отраженного двоичного кода в позиционный код, основанный на рекуррентном соотношении *, в соответствии с которым значение a_i i-го разряда двоичного кода числа определяется через значения b_n, b_{n-1}, ..., b_1 кода Грея:

$$a_n = b_n,$$
$$a_{n-1} = b_n \oplus b_{n-1},$$
$$a_1 = b_n \oplus b_{n-1} \oplus \ldots \oplus b_1, \qquad (1)$$

* Гитис Э. И. Преобразователи информации для электронных цифровых вычислительных систем.— М.: Энергия, 1975.— С. 157—160.

т. е. значение a_i, за исключением старшего разряда, зависит от значения a_{i+1} ранее определенного разряда:

$$a_i = a_{i+1} \oplus b_i. \qquad (2)$$

На основании (1) и (2) разработана программа преобразования 8-разрядного кода Грея в 8-разрядный двоичный код с помощью микроЭВМ на основе микропроцессора КР580ИК80А. Длина программы составляет 22 шага. Время выполнения подпрограммы — не более 80 мкс при тактовой частоте микропроцессора 2,5 МГц. Текст программы на ассемблере приведен ниже:

```
;*****************************************************
;*   ПРОГРАММА ПРЕОБРАЗОВАНИЯ КОДА ГРЕЯ В ДВОИЧНЫЙ КОД   *
;*****************************************************

GRAY: MOV  B,A  ; ПЕРЕСЫЛКА ИСХОДНЫХ ДАННЫХ
      RAL       ;
      MOV  C,A  ;
      MVI  A,0  ;
      RAL       ; ВЫДЕЛЕНИЕ ЦИФРЫ СТАРШЕГО РАЗРЯДА
      MOV  D,A  ;
      MVI  E,7  ; ЗАГРУЗКА СЧЕТЧИКА ЦИКЛОВ
NEXT: MOV  A,C  ; НАЧАЛО ОПРЕДЕЛЕНИЯ СЛЕДУЮЩЕЙ ЦИФРЫ
      XRA  B    ;
      RAL       ; ВЫДЕЛЕНИЕ СЛЕДУЮЩЕЙ ЦИФРЫ
      MOV  C,A  ; ПЕРЕСЫЛКА ПРОМЕЖУТОЧНОГО РЕЗУЛЬТАТА
      MOV  A,D  ;
      RAL       ;
      MOV  D,A  ;
      INR  E    ; ИНКРЕМЕНТ СЧЕТЧИКА ЦИКЛОВ
      MOV  A,E  ;
      JNZ  NEXT ;
      END       ;
```

Исходное число в коде Грея хранится в регистре B. В регистре C размещается промежуточный результат суммирования по модулю 2 и сдвига влево. Регистр D используется для формирования двоичного кода в процессе выполнения программы и хранения результата преобразования. Счетчик циклов организован в регистре E.

Программа работает следующим образом. Восьмиразрядный код Грея $b_8b_7b_6b_5b_4b_3b_2b_1$ заносится из аккумулятора в регистр B. Затем осуществляется сдвиг влево и старшая цифра b_8 переходит в триггер переноса, после чего содержимое аккумулятора пересылается в регистр C. Аккумулятор обнуляется, и производится еще один сдвиг влево. При этом цифра b_8, равная a_8 в соответствии с (1), переходит в младший разряд байта и содержимое аккумулятора пересылается в регистр D. Его значение равно $0000\ 000a_8$. Счетчику циклов присваивается значение 7, поскольку еще предстоит определить значение семи кодирующих цифр.

Промежуточный результат из регистра C пересылается в аккумулятор. Затем выполняется операция сложения по модулю 2 содержимого регистра B и аккумулятора и сдвиг результата влево на один разряд; в триггере переноса образуется цифра двоичного кода $a_7 = b_8 \oplus b_7$. Содержимое аккумулятора отсылается в регистр C, а содержимое регистра D сдвигается в аккумуляторе влево, при этом цифра a_7 из триггера переноса «подсоединяется» к предыдущей цифре a_8.

После инкрементирования счетчика циклов проверяется на нуль его содержимое. Если результат не равен нулю, происходит переход на метку NEXT и определяется следующая цифра; в противном случае программа заканчивается. Двоичный код, полученный в результате преобразования, будет размещен в регистре D и аккумуляторе. Используя алгоритм (1), можно разработать программу преобразования для других значений n.

Настоящая программа была применена в программируемом устройстве управления краном-штабелером автоматизированного склада, выполненном на базе микроконтроллера «Электроника К1-20».

Телефон 5-29-38, Львов

Статья поступила 27.11.87

УДК 681.3.06

И. П. Саркисов

СИСТЕМА АВТОМАТИЗИРОВАННОЙ РАЗРАБОТКИ АЛГОРИТМОВ И ПРОГРАММ ДЛЯ МИКРОЭВМ «ИСКРА 226» (СИСТЕМА АРАП)

Система АРАП (рис. 1) — инструментальное средство разработки программ на языке БЕЙСИК, поддерживающее разработку алгоритма, его программную реализацию и оптимизацию программы. Такое разделение на этапы обусловлено стремлением учесть специфику каждого из них. Для более полной и эффективной инструментальной поддержки выделенных этапов для каждого из них разработаны специально на него ориентированные средства.

Разработка алгоритма заключается в построении (редактировании) описания алгоритма и автоматическом (без участия разработчика) и автоматизированном (при участии разработчика) поисках ошибок в нем. Основу описания алгоритма составляет описание логики алгоритма, которое строится по принципу пошаговой детализации [1]. В связи с этим язык описания логики алгоритмов содержит элементы трех типов: обобщенного действия, подлежащего дальнейшей детализации; конкретного действия, не подлежащего дальнейшей детализации; условия. Элементы объединяются в две базовые конструкции. Каждая конструкция — это описание некоторого обобщенного действия.

Первая базовая конструкция используется, если на некотором шаге детализации обобщенное действие A разбивается на два последовательно выполняемых действия U_1 и U_2, каждое из которых может быть обобщенным или конкретным:

$$A \rightarrow U_1U_2.$$

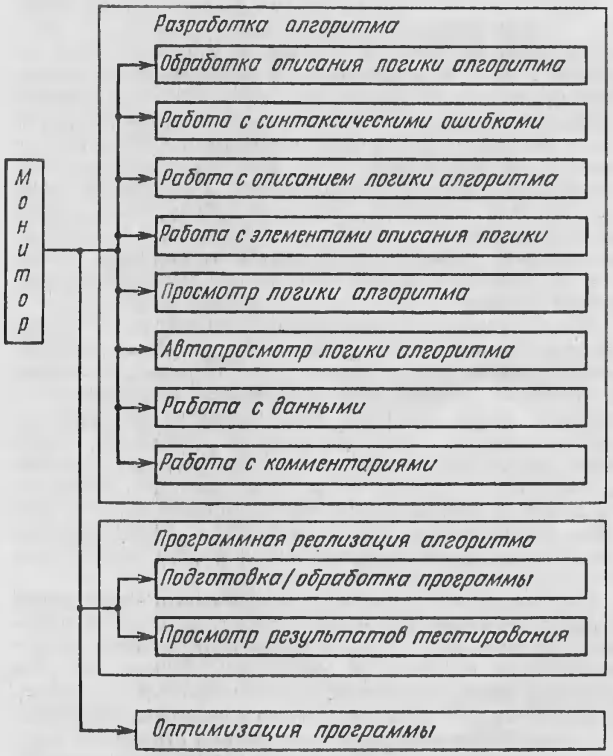


Рис. 1. Структурная схема системы АРАП.
Внутренние блоки — режимы работы, внешние — этапы

Вторая базовая конструкция используется, если обобщенное действие A при детализации предполагает разветвление по альтернативным условиям C_1, ..., C_n, в зависимости от которых должны быть выполнены действия V_1,...,V_n соответственно:

A → C_1V_1
! C_2V_2
. . . . .
! C_nV_n

Элементы определяются следующим образом:

обобщенное действие — произвольная последовательность символов, заключенная в угловые скобки, например ⟨СОРТИРОВКА⟩;

конкретное действие — произвольная последовательность символов без пробелов, например ВЫВОД*РЕЗУЛЬТАТА;

условие — произвольная последовательность символов без пробелов с префиксом IF, например IF*A=B.

В качестве примера представлено описание логики алгоритма вычисления суммы S положительных элементов массива A(1),...,A(N):

⟨СУММИРОВАНИЕ⟩ → ⟨ИНИЦИАЛИЗАЦИЯ⟩
⟨ВЫЧИСЛЕНИЯ⟩
⟨ИНИЦИАЛИЗАЦИЯ⟩ → ВВОД (N, A) S=0; I=1
⟨ВЫЧИСЛЕНИЯ⟩ → IF*I<=N ⟨ОБРАБОТКА A(I)⟩
I=I+1 ⟨ВЫЧИСЛЕНИЯ⟩
! IF*I>N ВЫВОД*S
⟨ОБРАБОТКА A(I)⟩ → IF*A(I)>0 S=S+A(I)
! IF*A(I)<=0

Для конкретных действий необязательно использовать действия, которые просто «дальше некуда» детализировать. Простой пример и определяет простоту конкретных действий. Право выбора нижнего уровня детализации остается за разработчиком алгоритма.

Система обеспечивает как ввод и редактирование описания логики алгоритма, так и его обработку. В частности, одна из функций системы — это поиск ряда логических ошибок в описании. К автоматически обнаруживаемым ошибкам относятся: незавершенность и избыточность описания, зацикливание, различные формы нарушений в описании разветвлений, например отсутствие альтернативы, неоднозначность и др. Конечно, отсутствие этих ошибок не гарантирует правильности алгоритма, но важно, что вся рутинная работа по обнаружению таких ошибок снимается с разработчика, и он может сконцентрировать свое внимание на вещах более сложных, например на участии в автоматизированном поиске некоторых других ошибок.

Общий принцип автоматизированного поиска ошибок состоит в том, что система извлекает из описания информацию, которая может помочь при проверке, и предоставляет ее разработчику, а *основная возможность* — просмотр логики алгоритма. Разработчик может в диалоговом режиме по шагам просмотреть любую на выбор ветвь алгоритма, причем каждый ее фрагмент просматривается на произвольном уровне детализации. Кроме того, можно автоматически формировать ветвь системой (режим автопросмотра логики алгоритма). Таким образом, разработчик имеет возможность проверять соответствие описания алгоритма самому алгоритму.

Система также позволяет подготавливать комментарии и описание данных. *Комментарий* представляет собой экранную страницу (или меньше) текста произвольного вида и может быть поставлен в соответствие любому элементу описания логики алгоритма, а *список данных* — любому конкретному действию или условию алгоритма, называемому также терминалом (или терминалами действия и условия соответственно). Список может содержать данные любых уровней абстракции и не обязательно все для соответствующего терминала.

Данные по отношению к терминалу могут быть входными, выходными и входными-выходными. *Входным* считается данное, значение которого используется в соответствующем терминале, но не изменяется, например входные данные I и N для терминала IF*I>N; *выходным* — данное, значение которого не используется в терминале, но изменяется, например S и I для терминала S=0; I=1; *входным-выходным* — данное, являющееся одновременно и входным и выходным. Например, I для терминала I=I+1.

Списки данных обрабатываются системой для поиска ошибок двух типов: неопределенности данных и зацикливания. *Неопределенность* данных означает, что найдена ветвь алгоритма, в которой для некоторого терминала данное является входным, но ни для одного из предыдущих терминалов ветви оно не является выходным; *зацикливание* — что данные, по которым проверяется условие выхода из цикла, не изменяются. В отличие от неопределенности данных зацикливание может быть обнаружено при участии разработчика, так как система проверяет, изменяется ли в анализируемом цикле каждое из данных, являющихся входными для каждого терминала условия. Таким образом, система предъявляет разработчику все «подозрительные» места и тот сам решает, действительно ли там содержится ошибка или нет.

Важной функцией системы является предоставление разнообразных возможностей редактирования описания логики алгоритма, комментариев, описания данных. Вся эта информация может использоваться в качестве основы рабочей документации разработки.

Программная реализация алгоритма. На этом этапе происходят кодирование, тестирование и отладка первого варианта программы, называемого отладочным.

Кодирование. Каждому терминалу описания логики алгоритма (т. е. конкретному действию и условию) ставится в соответствие закодированный обычным способом программный фрагмент, называемый телом терминального модуля. Эти фрагменты включаются в предварительно создаваемую системой заготовку программы, состоящую из универсального блока управления и заготовок терминальных модулей — строки заголовка и оператора возврата.

Блок управления осуществляет обращения к терминальным модулям в соответствии с описанием логики алгоритма, которое выступает здесь в качестве управляющих данных для программы, состоящей из блока управления и набора терминальных модулей (рис. 2). Таким образом, программист полностью освобождается от обязанностей по организации передачи управления между терминальными модулями, что приводит к упрощению процесса кодирования, сокращению его объема и, соответственно, снижению числа возможных ошибок.

Тестирование выполняется вне системы АРАП обычным способом, но с использованием средств тестирования и отладки, встроенных в блок управления. Средства тестирования и отладки действуют на уровне терминальных модулей и выполняют следующие функции: проверку альтернативных условий на взаимоисключаемость и взаимодополняемость, учет покрытия логики программы [2], трассировку.

Проверка альтернативных условий обусловлена требованием, чтобы при разветвлении выполнялось одно и только одно из них. Поэтому при проверке вызываются поочередно все терминальные модули данной группы условий. Если выполнилось (т. е. модуль установил специальный

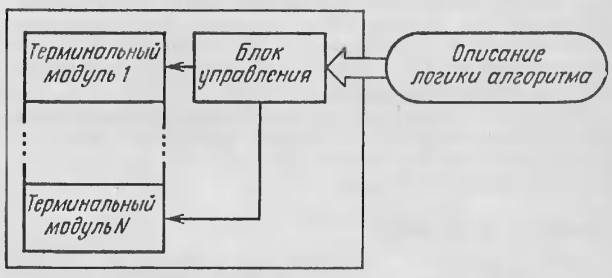


Рис. 2. Структура отладочного варианта программы

признак) более одного условия, значит нарушилась взаимоисключаемость, если не выполнилось ни одно — нарушилась взаимодополняемость.

В описании логики алгоритма можно использовать специальный терминал условия ELSE, имеющий тот же смысл, что и в языках программирования. Терминалу ELSE не нужно ставить в соответствие терминальный модуль. Однако наличие в группе альтернативных условий терминала ELSE лишает возможности проверки на взаимодополняемость, так как, если не выполнено ни одно из предыдущих условий группы, считается выполненным условие ELSE. Поэтому рекомендуется применять ELSE при описании наиболее простых разветвлений, а в других случаях каждое условие описывать явно и кодировать. Это, во-первых, позволит провести более полное тестирование и, во-вторых, на этапе оптимизации терминал ELSE все равно будет автоматически введен в каждую группу альтернативных условий.

Учет покрытия логики программы — подсчет числа выполнений терминальных модулей с фиксацией времени выполнения каждого из них. Информация накапливается от теста к тесту. В конце работы программы могут быть выданы число (общее по всем зафиксированным тестовым прогонам и среднее) и время (общее и среднее) выполнения каждого модуля. Те же результаты тестирования могут быть просмотрены и при работе в системе АРАП. Получаемая информация помогает следить за состоянием (степенью полноты) тестирования, а также выявлять критические области программы [3], т. е. наиболее долго выполняющиеся терминальные модули. Такие модули рекомендуется оптимизировать в первую очередь.

Трассировка — вывод на экран сообщения о выполняемом в данный момент терминальном модуле.

Изменения на уровне алгоритма вносятся в описание его логики. При этом, если в измененное описание не включены новые терминалы, текст программы не изменяется. Как расширение этой ситуации предусмотрена возможность включения в программу большего числа терминальных модулей, чем требуется для описания логики отдельного алгоритма. Таким образом, в случае, когда для конкретного класса задач уже подготовлен достаточный набор терминальных модулей, система может функционировать в режиме генерации программ.

Оптимизация программы выполняется автоматически. В результате формируется текст второго (оптимизированного) варианта программы, который существенно превосходит отладочный по требуемому объему памяти и быстродействию. Такой вариант уже не содержит ни блока управления со всеми встроенными в него средствами тестирования и отладки, ни лишних терминальных модулей (если они были в отладочном варианте), а также не использует описания логики алгоритма в качестве управляющих данных, так как все логические связи устанавливаются непосредственно в программе.

Оптимизированный вариант не предполагает внесения в него изменений. Если, например, обнаружено, что на некоторых входных данных оптимизированный вариант работает неверно, то на этих же данных выполняется отладочный вариант (он хранится в течение всего жизненного цикла программы). С помощью средств тестирования и отладки ошибка обнаруживается, исправляется и оптимизация выполняется вновь.

На рис. 3 двойной рамкой отмечены процессы, выполняемые непосредственно при работе в системе АРАП. Стрелка от информации к процессу означает, что информация является входной, т. е. используется процессом; стрелка от процесса к информации — что информация является выходной, т. е. изменяется в данном процессе.

При разработке алгоритма создается (или редактируется при повторении этого процесса) описание алгоритма (описание логики алгоритма и данных, а также комментарии). Подготовка отладочного варианта программы заключается в формировании системой его заготовки. Обработка выполняется, когда программа уже существует и в описание логики алгоритма внесены изменения. В этом случае проверяется соответствие между терминалами описания ло-

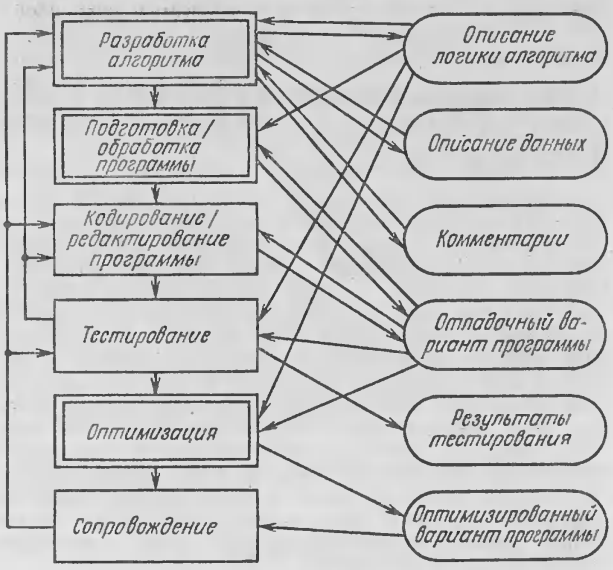


Рис. 3. Схема связей процессов и информации при использовании системы АРАП

гики алгоритма и терминальными модулями программы. При необходимости система добавляет в текст программы заготовки отсутствующих терминальных модулей, а лишние модули не удаляет.

На схеме отмечены только те связи, которые осуществляются физически, т. е. когда информация в данном процессе либо загружается в оперативную память, либо вводится или выводится программными средствами. Связи же логического характера, такие как использование любых составляющих описания алгоритма при кодировании/редактировании программы, на схеме отсутствуют, хотя в действительности, конечно, существуют.

Если при тестировании обнаруживаются ошибки, то в зависимости от того, где нужно произвести изменения, в алгоритме или программе, осуществляется переход к соответствующему процессу. В эксплуатацию поступает оптимизированный вариант программы. Смысл процесса «сопровождение» в схеме несколько сужен в том плане, что обычно сопровождение — это исправление или модификация программы в процессе ее эксплуатации, а здесь сопровождение — это эксплуатация вплоть до принятия решения об исправлении или модификации. Сами же исправление и модификация происходят при переходе к соответствующему процессу.

Информация, необходимая для освоения системы АРАП, может быть получена пользователем непосредственно в ходе работы в системе. Общий объем справочной информации составляет около 1500 экранных строк (примерно в 5 раз больше объема настоящей статьи). Более 50 % программ, входящих в состав системы АРАП, разработаны с помощью самой системы АРАП. Объем системы составляет 250 Кбайт (одна сторона гибкого магнитного диска).

В 1987 г. система АРАП сдана в отраслевой фонд алгоритмов и программ Центросоюза (117962, Москва, ГСП-1, ул. Донская, 8, ПТО «Центросоюзсистема»).

Телефон 236-15-13, Москва

ЛИТЕРАТУРА

1. Хьюз Дж., Мичтом Дж. Структурный подход к программированию.— М.: Мир, 1980.
2. Майерс Г. Искусство тестирования программ.— М.: Финансы и статистика, 1982.
3. Ван Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ.— М.: Мир, 1981.

Статья поступила 1.12.87

УДК 681.324

Р. И. Белицкий

ЯДРО ОПЕРАЦИОННОЙ СИСТЕМЫ МУЛЬТИПРОЦЕССОРА С МАГИСТРАЛЬНОЙ СТРУКТУРОЙ

Эффективность применения многопроцессорных систем общего назначения, как правило, обусловливается наличием в них нескольких одновременно протекающих вычислительных процессов. Управление таким набором взаимодействующих процессов частично осуществляется аппаратурой вычислительной системы, а в остальном — группой программ ОС, часто называемой ядром. Экспериментальный образец мультимикропроцессорной системы с магистральной структурой [1] состоит из набора элементарных процессоров, объединенных несколькими общими шинами, ряда модулей общей памяти (по одному на каждую шину) и одного устройства памяти семафоров. Элементарный процессор содержит в себе микропроцессор К1810ВМ86, локальную память, блоки ассоциативной адресации и системной синхронизации, собственные модули общей памяти (по одному на каждую шину) и некоторые вспомогательные устройства.

Программирование в системе осуществляется на Макроассемблере микропроцессора К1810ВМ86, расширенном описателями и операторами для явного задания потенциального параллелизма: SHARED, PARBEGIN, PAREND, BEGIN, BEGEND, PARALLEL FOR, EXIT, P, V, WAIT, LET.

Основные функции рассматриваемого ядра: выполнение начальных установок, распределение работ между элементарными процессорами, программная поддержка системных операторов, обработка системных прерываний.

Начальные установки выполняются каждым элементарным процессором по одному разу (в самом начале работы) и включают в себя установку в исходное состояние блока ассоциативной адресации и указателя стека, расположенного в локальной памяти элементарного процессора, а также выбор признака исполняемой работы и адреса начала локальной памяти для прикладных программ. Блок ассоциативной адресации используется для динамической идентификации элементарного процессора, в частности для выдачи ему системных прерываний.

Распределение работ между элементарными процессорами основано на следующем. Параллельная программа, исполняемая мультипроцессором, имеет иерархическую структуру [2] с группой совместно исполняемых программ, каждая из которых ограничена операторами BEGIN, BEGEND и заключена в операторные скобки PARBEGIN, PAREND. (В соответствии с их графическим представлением участки программ между парами BEGIN, BEGEND названы ветвями, а операторы PARBEGIN — вершинами.) Работа заключается в выполнении программы ветви и управлении непосредственно подчиненными ей, т. е. текстуально вложенными, ветвями. Вся информация о имеющихся работах и о подчинении одних ветвей другим отражается в управляющих таблицах, размещенных в общей памяти, поэтому распределение работ сводится к поочередному выбору свободными элементарными процессорами из управляющих таблиц информации, относящейся к ветвям, активизированным к текущему моменту времени другими элементарными процессорами при выполнении ими операторов PARBEGIN.

Из принятых в языке рассматриваемого мультипроцессора системных операторов ОС поддерживаются PARBEGIN, PARALLEL FOR, BEGEND, EXIT, остальные реализованы аппаратно. Эти операторы выполняются следующим образом:

PARBEGIN (создание новой вершины) — формируется динамическая часть таблицы новой вершины, имя вершины заносится в список активных вершин и выдается системное прерывание, информирующее все свободные процессоры о появлении новых работ, представляющих собой подчиненные ветви активизированной вершины. После этого процессор переходит к ожиданию завершения выполнения всех подчиненных вершине ветвей, затем продолжает выполнять свою ветвь. Одновременно с ожиданием процессор пытается выполнить некоторые из созданных им работ, чтобы уменьшить время своего простоя.

PARALLEL FOR — аналогичен предыдущему оператору и служит лишь указанием транслятору, что создается однопараметрическое семейство ветвей, а не набор разнородных ветвей, как в операторе PARBEGIN.

BEGEND (конец ветви) — возможны два случая: процессор выполняет только эту ветвь или также и другие ветви. В первом случае процессор лишь модифицирует управляющую таблицу вершины, которой подчинена выполненная ветвь, и пытается выбрать для исполнения ветвь той же вершины или, при отсутствии таковой, не выполнявшуюся ветвь другой вершины. Во втором случае процессор после модификации управляющей таблицы вершины возвращается к выполнению оператора PARBEGIN.

EXIT (досрочный выход всех процессоров из ветвей, подчиненных заданной вершине) — указанная в операторе вершина удаляется из списка активных вершин, а затем выдается системное прерывание, уведомляющее все процессоры, выполняющие подчиненные вершине ветви, о необходимости закончить их и перейти либо к поиску работы, либо к продолжению соответствующего оператора PARBEGIN.

Единственными источниками системных прерываний в текущей версии мультипроцессора с магистральной структурой являются операторы EXIT. Возможность восприятия таких прерываний, передаваемых по системной магистрали, обусловлена применением динамической адресации элементарных процессоров по обрабатываемой ими информации, что, как отмечалось в [1], наряду с применением ассоциативной адресации данных — это основа архитектуры потоко-асинхронной мультимикропроцессорной системы. В случае системного прерывания основой идентификации элементарного процессора является имя вершины, которой подчинена исполняемая им ветвь. Это имя заносится в блок ассоциативной адресации, как только процессор загружает очередную ветвь, и удаляется оттуда при выполнении оператора BEGEND. Обработка системного прерывания заключается в отыскании по внутренним управляющим таблицам процессора имени вершины (если она имеется), которую он должен удалить из списка активных элементов с выдачей соответствующего системного прерывания из-за ее подчиненности исходной удаляемой вершине, а затем в переходе на поиск работы или на продолжение выполнения оператора PARBEGIN.

Программа ядра состоит из подпрограмм: NACHUST — начальные установки, POISKRAB — поиск работы, PARNACH — выполнение операторов PARBEGIN и PARALLEL FOR, KONEC — выполнение оператора BEGEND, VYHOD — выполнение оператора EXIT, SYSOBR — обработка системного прерывания. Кроме того, поскольку для большей гибкости в использовании ОС и возможности ее простой модификации перечисленные подпрограммы вызываются посредством программных прерываний, в состав ядра включен также массив адресов точек входов в соответствующие подпрограммы.

Программа ядра ОС мультипроцессора с магистральной структурой написана на Макроассемблере микропроцессора К1810ВМ86. Выбор для программирования языка низкого уровня обусловлен необходимостью введения ПЗУ с ядром ОС в состав каждого элементарного процессора, что накладывает жесткие ограничения на объем ядра. Объем программы, реализующей описанные выше функции, составил 3 Кбайт, в связи с чем ее удалось разместить в двух микросхемах ПЗУ объемом 2 Кбайт каждая.

В настоящее время программа полностью отлажена и работает на экспериментальном образце мультимикропроцессорной системы. Одновременно с помощью моделирующей системы исследуется влияние разработанной ОС на

производительность мультипроцессора для получения рекомендаций при последующих модификациях программы.

Телефон 265-55-30, Киев

ЛИТЕРАТУРА

1. Белицкий Р. И. Адресация данных в потоковой мультимикропроцессорной системе с магистральной структурой // Микропроцессорные средства и системы.— 1985.— № 1.— С. 17—20.
2. Дейкстра Э. Взаимодействие последовательных процессов.— В кн.: Языки программирования / Под ред. Ф. Женюи.— М.: Мир, 1972.— С. 9—86.

Статья поступила 12.02.88

КРАТКИЕ СООБЩЕНИЯ

УДК 681.3.06

А. Р. Деменюк

ПРОГРАММА ПРЕОБРАЗОВАНИЯ ЗНАЧЕНИЯ УРОВНЯ НАПРЯЖЕНИЯ ИЗ МИЛЛИВОЛЬТ В ДЕЦИБЕЛЛЫ ДЛЯ МИКРОПРОЦЕССОРА КР580ИК80

При сопряжении АЦП с микроЭВМ или контроллером нередко возникает необходимость программного преобразования значения уровня сигнала из милливольт в децибеллы.

Особенности программ, реализующих это преобразование, определяются:

видом и разрядностью АЦП в зависимости от диапазона, точности и скорости измерения;

быстродействием и объемом памяти исходя из диапазона и точности преобразования.

Предлагаемая программа на ассемблере для микропроцессора КР580ИК80 — вариант измерения и преобразования в децибеллы уровня сигналов в диапазоне —60,0... +59,9 дБ с точностью 0,1 дБ.

Использование быстродействующих АЦП с организацией поддиапазонов позволяет упростить аппаратную часть, сократить время выполнения программы преобразования и существенно уменьшить объем памяти, занимаемый массивом эталонных значений напряжений.

При организации поддиапазонов коэффициент передачи сигнала во время измерения его уровня аппаратно или программно-аппаратно устанавливается таким, чтобы уровень сигнала на входе 10-разрядного АЦП поразрядного кодирования с источником опорного напряжения 10,24 В находился в интервале 775...7761 мВ (0,0... 19,9 дБ).

Поддиапазон, дБ	$K_{пер}$	Код поддиапазона	Конечный результат преобразования
—60,0..—40,1	1000	6	(HL) — 0258H
—40,0..—20,1	100	5	(HL) — 0190H
—20,0..—0,1	10	4	(HL) — 00C8H
0,0...+19,9	1	3	(HL)
20,0...+39,9	0,1	2	(HL)+00C8H
+40,0...+59,9	0,01	1	(HL)+0190H

Для измерения уровня входного сигнала в диапазоне —60,0...+59,9 дБ организуется шесть поддиапазонов с соответствующими коэффициентами передачи ($K_{пер}$) сигнала (см. таблицу).

Для расчета конечного результата программа преобразования уровня сигнала в децибеллы оперирует и кодом поддиапазона. В ней применен табличный метод, который упрощает программу и существенно сокращает время ее работы.

Для преобразования измеренного значения необходимо составить соответствующий массив эталонных значений мощности или написать программу пересчета значения уровня по мощности из значения уровня по напряжению: $P_м = P_н - 10\lg(|Z|/600)$, где Z — модуль сопротивления, на котором определяются уровни; $P_м$, $P_н$ — значения уровня сигнала по мощности и напряжению. В этом случае необходимо ввести код модуля сопротивления и представить вычитаемое формулы в виде константы в соответствии с этим кодом.

690000, Владивосток, ул. Октябрьская, д. 8, Дальтехэнерго, электроцех

ЛИТЕРАТУРА

1. Абдуллаев Н. Т., Измайлова Л. З., Тургиев Э. А. Организация работы АЦП в микропроцессорной системе // Приборы и системы управления — 1984.— № 10.
2. Бахтиаров Г. Д., Малинин В. В., Школин В. П. Аналого-цифровые преобразователи.— М.: Сов. радио, 1980.
3. Гиатек Ю. Р. Справочник по цифро-аналоговым и аналого-цифровым преобразователям.— М.: Радио и связь, 1982.

Статья поступила 21.12.87

УДК 681.3.06

Д. М. Блинов

РЕДАКТОР ЭКРАННЫХ ОКОН

Редактор экранных окон — это составная часть экспертной системы и может использоваться другими программными средствами для создания и редактирования окон на экране монитора ПЭВМ.

Окно — это прямоугольная площадь на экране монитора (поле окна), обрамленная рамкой. Информация размещается в нем сверху-вниз и слева-направо. Информацию в окне можно «листать» и перемещать влево и вправо за его границы.

Для оперирования с окнами нами принята единая форма представления их характеристик. Эти характеристики хранятся в виде записи данных в файле на внешнем носителе, т. е. каждому окну, объявленному в программе, ставится в соответствие запись из характеристик этого окна (рис. 1).

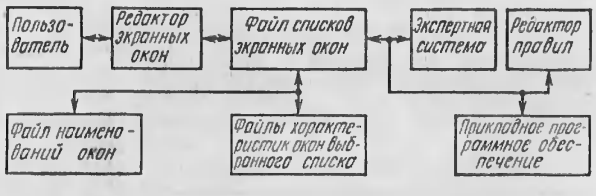


Рис. 1. Взаимосвязь редактора окон с программным обеспечением и данными

Пользователь	Редактор
SED(SED есть идентификатор редактора)	Показывается главное меню редактора
Выбирает пункт „Редактировать“	Показывается меню видов редактирования
Выбирает пункт „Цвета“	Показывается меню фрагментов окна для раскраски
Выбирает пункт „Символы окна“	Показывается меню из восьми цветов: черный, голубой, зелено-голубой, красный, малиновый, коричневый, белый
Выбирает пункт „Голубой“	Укажите имя файла
WIND.DAT	(Если имя файла не будет указано, то информация будет сохранена в файле с прежним именем)

Рис. 2. Пример диалога пользователя с редактором окон

Здесь в таблице (см. рис. 2) заглавными буквами показаны сообщения редактора или пользователя и строчными — объяснение их действий.

Редактор экранных окон передается другим организациям в виде загрузочного (около 72 К), текстового файла, описывающего порядок использования этой программы, и контрольного примера в виде базы данных, включающей в себя файл списков экранных окон, файл наименований окон и файл характеристик окон для тренировки пользователя по их редактированию, созданию или дополнению.

Телефон 254-32-95 (с 16 до 18 ч.), Москва

Сообщение поступило 27.07.88

КРАТКОЕ СООБЩЕНИЕ

УДК 681.3.03

В. В. Рак, А. М. Родымюк

УНИВЕРСАЛЬНАЯ ПОДПРОГРАММА БАЙТОВОГО ДЕЛЕНИЯ

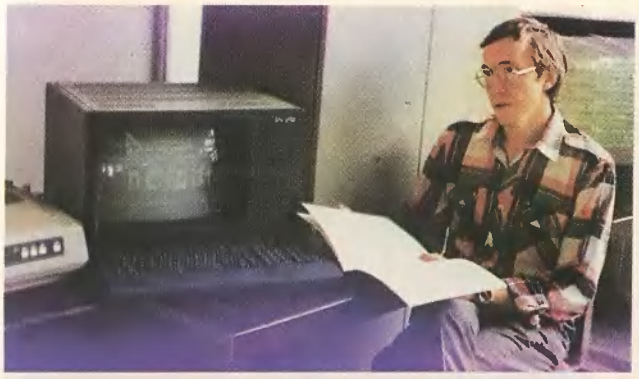
Несколько вариантов подпрограмм многобайтового деления, предложенных в работах [1, 2], предполагают ограничение частного: оно может быть лишь 16-разрядным. Нужно следить за значением бита переноса, который указывает на переполнение, и анализировать остаток.

Предлагаемая подпрограмма DIV4B (рис. 1) деления целого положительного 4-байтового числа на целое положительное 2-байтовое число с получением 4-байтового частного и округлением остатка не требует таких ограничений. За основу взят алгоритм «длинного» деления 4-значного числа на 2-значное [1]. Алгоритм подпрограммы предусматривает возможность произвольной комбинации значений делимого и делителя в пределах разрядности. Деление на нуль равносильно делению на 10000H (65536 десятичных). Если при округлении увеличенный в два раза остаток больше делителя, то частное увеличивается на единицу. Это легко проверить, так как делитель после выхода из подпрограммы DIV2B находится в дополнительном коде, а остаток — в регистрах HL.

Подпрограмму DIV2B (рис. 2) можно использовать для деления с получением 16-разрядных частного DE и остатка HL.

```
;.....................................................;
;                                                     ;
;     DIV4B - ПОДПРОГРАММА ЦЕЛОЧИСЛЕННОГО ДЕЛЕНИЯ     ;
;             С ОКРУГЛЕНИЕМ ЧАСТНОГО                  ;
;                                                     ;
;                  СОДЕРЖИМОЕ РЕГИСТРОВ               ;
;     ПЕРЕД ОБРАЩЕНИЕМ : !     ПОСЛЕ ВОЗВРАТА :       ;
;    (HL-DE) - ДЕЛИМОЕ   ! (HL-DE) - ЧАСТНОЕ          ;
;    (BC)    - ДЕЛИТЕЛЬ  ! (BC)    - ДЕЛИТЕЛЬ В        ;
;                        !         ДОПОЛНИТЕЛЬНОМ КОДЕ ;
;                                                     ;
;      НАЧАЛЬНЫЙ АДРЕС ПОДПРОГРАММЫ - 0000H           ;
;      ИСПОЛЬЗУЕМАЯ ПОДПРОГРАММА    - DIV2B          ;
;.....................................................;
;
0000              ORG   0000H  ; РАСПОЛОЖИТЬ ПОДПРОГРАММЫ С  0000H
;
;    ПРОВЕРКА  КОЛИЧЕСТВА  ЗНАЧАЩИХ  БАЙТОВ  ЧАСТНОГО
;
0000 7D     DIV45: MOV   A,L    ; СРАВНЕНИЕ ДЕЛИТЕЛЯ И ДВУХ  СТАРШИХ
0001 91            SUB   C      ;    БАЙТОВ ДЕЛИМОГО ПУТЕМ ВЫЧИТАНИЯ
0002 7C            MOV   A,H    ;    И  АНАЛИЗА  БИТА  ПЕРЕНОСА
0003 98            SUB   B      ;
0004 DA1D00        JC    DIV40  ; ЗНАЧАЩИМИ БУДУТ НЕ БОЛЕЕ ДВУХ БАЙТОВ
;
;  ЗНАЧАЩИМИ  БУДУТ  БОЛЕЕ  ДВУХ  БАЙТОВ  ЧАСТНОГО
;  ДЕЛЕНИЕ  СТАРШИХ  ДВУХ  БАЙТОВ  ДЕЛИМОГО
;
0007 D5            PUSH  D      ; ЗАПОМИНАНИЕ МЛАДШИХ БАЙТОВ ДЕЛИМОГО
0008 110000        LXI   D,0H   ; ОЧИСТКА  РЕГИСТРОВ   D , E
000B EB            XCHG         ; ОБМЕН  ( HL )  И  ( DE )
000C CD2C00        CALL  DIV2B  ; (DE) := (HL-DE):(BC),  (HL)-ОСТАТОК
;
;  ДЕЛЕНИЕ ПОЛУЧЕННОГО ОСТАТКА И МЛАДШЕЙ ЧАСТИ ДЕЛИМОГО
;
000F EB            XCHG         ; ПЕРЕСЫЛКА ДВУХ БАЙТОВ ЧАСТНОГО В HL
0010 F3            XTHL         ; ЗАПИСЬ ЧАСТНОГО И ЧТЕНИЕ  ДЕЛИМОГО
0011 EB            XCHG         ; ПЕРЕСЫЛКА МЛ. ЧАСТИ ДЕЛИМОГО В  DE
0012 CD3300        CALL  DIV27  ; (DE) := (HL-DE):(BC), (HL)-ОСТАТОК
;
;  НАЧАЛО ОКРУГЛЕНИЯ В СЛУЧАЕ БОЛЬШОЙ ЗНАЧИМОСТИ ЧАСТНОГО
;
0015 29            DAD   H      ; УВЕЛИЧЕНИЕ  ОСТАТКА  В  ДВА  РАЗА
0016 09            DAD   B      ; СРАВНЕНИЕ  ДЕЛИТЕЛЯ  И  ОСТАТКА
0017 E1            POP   H      ; ВЫБОРКА  СТАРШИХ  БАЙТОВ ЧАСТНОГО
0018 33            INX   SP     ; ВОССТАНОВЛЕНИЕ УКАЗАТЕЛЯ СТЕКА ДЛЯ
0019 33            INX   SP     ;       ВЫХОДА  ИЗ  ПОДПРОГРАММЫ
001A C32500        JMP   DIV48  ; ПЕРЕХОД НА ЗАВЕРШЕНИЕ  ОКРУГЛЕНИЯ
;
;  ДЕЛЕНИЕ И ОКРУГЛЕНИЕ В СЛУЧАЕ МАЛОЙ ЗНАЧИМОСТИ ЧАСТНОГО
;
001D CD2C00 DIV40: CALL  DIV2B  ; (DE) := (HL-DE):(BC), (HL)-ОСТАТОК
0020 29            DAD   H      ; УВЕЛИЧЕНИЕ  ОСТАТКА  В  ДВА  РАЗА
0021 09            DAD   B      ; СРАВНЕНИЕ  ДЕЛИТЕЛЯ  И  ОСТАТКА
0022 210000        LXI   H,0H   ; ОЧИСТКА 2 СТАРШИХ БАЙТОВ ЧАСТНОГО
;
;  ЗАВЕРШЕНИЕ  ОКРУГЛЕНИЯ  И  ВОЗВРАТ  ИЗ  ПОДПРОГРАММЫ
;
0025 D0     DIV48: RNC           ; ВОЗВРАТ, ЕСЛИ  НЕТ  ОКРУГЛЕНИЯ
0026 1C            INR   E      ; УВЕЛ. МЛАДШЕГО БАЙТА ЧАСТНОГО НА 1
0027 C0            RNZ          ; ВОЗВРАТ, ЕСЛИ НЕТ ПЕРЕНОСА
0028 14            INR   D      ; ПЕРЕНОС  ЕДИНИЦЫ В СЛЕДУЮЩИЙ БАЙТ
0029 C0            RNZ          ; ВОЗВРАТ, ЕСЛИ НЕТ ПЕРЕНОСА
002A 23            INX   H      ; ПЕРЕНОС  ЕДИНИЦЫ В СТАРШИЕ БАЙТЫ
0028 C9            RET          ; ВОЗВРАТ
```

Рис. 1. Подпрограмма целочисленного деления с определением частного DIV4B



Управляющие вычислительные комплексы ПС 1001 представляют набор технических и программных средств

Абонентский пункт ЕС8536.М представляет собой комплекс аппаратно-программных средств, спроектированный на базе получающего массовое распространение персонального компьютера ПК8020 (комплекса учебной вычислительной техники «Корвет») и функционально самостоятельного сетевого контроллера СК8020. Пункт предназначен для работы в системах и сетях телеобработки данных ЕС ЭВМ в качестве интеллектуальной терминальной станции, обеспечивающей по телефонным каналам связи дистанционный доступ к ресурсам универсальных ЭВМ и может использоваться в территориальных распределенных системах автоматизированного управления и обработки данных.

ними ПЭВМ — 1500 м; скорость передачи информации в сети 250... 500 Кбит/с.

Реализация принципа программной смены кодовых таблиц и знакогенераторов в дисплее и печатающем устройстве служит основой применения различных алфавитов и создания национальных версий системного и программного обеспечения.

«Айболит»—пакет прикладных программ—незаменимый помощник при индивидуальной диагностике, классификации и терапии острых расстройств сердечной деятельности и системного кровообращения.

В основе пакета лежат разработанные в Институте сердечно-сосудистой хирургии им. А. Н. Бакулева математические модели сердечно-сосудистой системы, отражающие общие физиологические закономерности, позволяющие включать клинический опыт и индивидуальную специфику больных.

Пакет дает возможность врачу за несколько минут выявить причину расстройства кровообращения и определить индивидуальное наилучшее для каждого больного лечение.

Пакет включает программы расчета параметров модели по результатам расшифровки ЭКГ, данным артериального, венозного, легочного артериального и легочного венозного давлений. Программы написаны на языках БЕЙСИК и ассемблер.

«Айболит» функционирует на базе персональных ЭВМ ЕС1841, Агат. Пакет программ легко адаптируется на другие широко распространенные в мире ПЭВМ; для работы достаточно 64 Кбайт оперативной памяти.

Использование пакета прикладных программ «Айболит» при лечении больных после операции на сердце позволило в 1,5 раза уменьшить количество осложнений и почти в два раза снизить летальность.

Заслуживает внимания и разработка Киевской межотраслевой экспериментальной лаборатории — **«Медиана».** Это — многофункциональный прибор, эффективный диагностический инструмент, сочетающий возможности электрокардиографа, реографа, векторкардиографа, спирографа, мониторной системы; позволяет автоматизировать диагностические и психофизиологические исследования.

Система ввода информации в прибор от пациента — «on — line» и по телефонному каналу связи. Результаты исследования выводятся на мозаичную печать.

Прибор работает автономно. Сам расшифровывает полученную информацию, устанавливает диагноз. Его отличает высокая надежность: наработка на отказ — 10 000 ч. Незаменим при испытаниях на профессиональную пригодность.

Кодировщик изображений — периферийное устройство, расширяющее область применения ЭВМ и повышающее эффективность ее использования. Основная функция кодировщика — работа в системах автоматизированного проектирования в качестве производительного устройства ввода в ЭВМ изображений с документов, выполненных на бумаге. При этом производится: кодирование в растровой форме и ввод в ЭВМ бинарных и полутоновых изображений.

Устройство обеспечивает регулировку освещенности зоны считывания, порога бинарного кодирования и перемещения датчика в любую точку рабочего поля.

Для кодирования изображений с документов использована телекамера типа КТ-2-2 на базе ПЗС — матрицы со стандартным телевизионным сигналом.

Управляющий вычислительный комплекс ПС 1001 используется в АСУТП, в одноуровневом многомашинном УВК, на нижнем уровне иерархических АСУТП, в качестве автономной ЭВМ. Его отличает оригинальный принцип организации функционирования троированного комплекса. Комплекс может включать в себя от одного до трех процессоров. Если в троированном комплексе отказывает один из процессоров, он автоматически переходит в режим дублированного комплекса. Если в дублированном комплексе выходит из строя один из процессоров, устанавливается режим работы нерезервированного комплекса. После восстановления отказавшего процессора его состояние приводится в точное соответствие с действующими процессорами и он включается в синхронную работу.

Комплекс отладки микропроцессорных систем «Электроника НЦ-803» позволит осуществить отладку, контроль и диагностирование микропроцессорных средств и систем на этапах проектирования, производства и эксплуатации. Его можно встраивать в гибкие производственные системы.

В состав комплекса входит: любая ЭВМ, совместимая по архитектуре с ЭВМ семейства «Электроника», СМ4, СМ3; блок логических устройств «Электроника» НМС 59401.1»; устройство электрофизического сопряжения.

Программное обеспечение комплекса работает под управлением операционной системы РАФОС или ФОДОС.

Среди экспонатов Польской Народной республики надо отметить персональный микрокомпьютер MAZOVIA 2032 и микрокомпьютерную абонентскую телексную станцию МТХ-50.

MAZOVIA 2032 разработан на мик-

Микрокомпьютерная абонентская телексная станция МТХ-50

Персональный микрокомпьютер MAZOVIA 2032

ропроцессоре I80386 и сопроцессоре I80287 или I80387, имеет память 2...16 Мбайт, системный тактовый генератор на 16/20 МГц с возможностью работы с частотой 8 или 12 МГц, единый графический адаптер EGA, гибкие магнитные диски емкостью 1,2 Мбайт и 360 Кбайт, накопитель на жестком магнитном диске емкостью 40 Мбайт. Системный блок выпускается в настольном исполнении и в форме стойки.

Встроенная базовая система ввода-вывода обеспечивает полную совместимость с IBM PC AT и возможность использования операционных систем: DOS 3.30, Microsoft, OS/2, SCO XENIX 386. MAZOVIA 2032 применяется в системах автоматизированного проектирования, для редактирования, управления большими базами данных, интерактивной обработки изображений, работы с многими пользователями.

МТХ-50 незаменима при организации связи по коммутационным каналам связи, причем без ограничений относительно дальности передачи, длины и количества набираемых номеров абонентов. С использованием микрокомпьютерной программы-редактора текстов станция позволяет подготавливать телеграммы, автоматически отправлять их в несколько адресов, многократно набирать номер абонента в случае его занятости и т. д. При этом резко сокращается время продолжения связи, на 40 % снижается стоимость использования сети.

В основе станции — 8-разрядный микрокомпьютер МК-50, видеотерминал, два накопителя на гибких магнитных дисках, преобразователь ТХ-50 и печатающее устройство Д-100.

Народная Республика Болгария наряду с известными и хорошо зарекомендовавшими себя персональными компьютерами «Правец» представила и ряд других интересных разработок:

32-разрядную микроЭВМ ИЗОТ 1057С — для автоматизации административно-управленческой деятельности, в качестве CAD/CAM системы для машинной графики, лабораторных исследований и т. д. Центральный процессор — на интегральных микросхемах со сверхбольшой степенью интеграции.

Графическую систему, созданную на базе персонального компьютера «Правец 16А». Благодаря модульному принципу построения и возможности подсоединения различных периферийных устройств эта система может быть использована как для автоматизации конторской деятельности, так и в мощных инженерных и научных станциях, работающих в гомогенных и гетерогенных сетях. Система предусматривает возможность ввода, обработки, хранения и вывода графической информации.

Систему обработки изображений ИЗОТ 1073Е, в которой процесс обработки данных, представленных в виде изображений, выполнен на базе специального процессора ввода, обработки, накопления, вывода и отображения полутоновой и графической информации. Модульный принцип построения системы позволяет расширять ее за счет подключения дополнительных технических средств и периферийных устройств.

Функциональные возможности системы: ввод изображений для цифровой обработки с телевизионной камеры или с магнитных носителей; перенос изображения на магнитные носители для долговременного хра-

Система обработки изображений ИЗОТ 1073Е

нения; отображение на экране цветного монитора вводимых изображений и результатов его обработки и т. д. Области применения: дистанционное зондирование, астрономия, ядерная физика, биология и медицина, геология, контроль окружающей среды и т. д.

Институт технической кибернетики Словацкой академии наук (ЧССР) представил **ускоритель персональных компьютеров на базе архитектуры RISC процессора.** Этот ускоритель значительно повышает мощность персональных компьютеров, совместимых с IBM PC. Архитектура RISC (Reduced instruction set computer) процессора позволяет создать 32-разрядный микропроцессор, сравнимый по производительности с процессорами архитектуры CISC (complex instruction set computer), но с уменьшенной на порядок плотностью компановки СБИС. Сокращенное число инструкций архитектуры RISC процессора, их простая структура дают возможность осуществить конвейерную декодировку и отработать инструкции в одном машинном цикле. Отличительная особенность процессора с архитектурой RISC в том, что большинство инструкций он выполняет с помощью внутренних рабочих регистров, обеспечивая тем самым увеличение скорости вычислений процессора до 4 млн. операций/с.

Процессор с архитектурой RISC вместе с ОЗУ и интегральной схемой управления памятью расположен на одной печатной плате стандартных размеров. Данный процессор взаимодействует с 16-разрядным процессором, используя канал прямого доступа к ОЗУ персонального компьютера. Достаточно вставить плату с процессором архитектуры RISC в 16-разрядную персональную вычислительную машину IBM PC AT, и пользователь получает новое качество вычислений — 32-разрядный процессор и 4 Мбайт оперативной памяти.

Ускоритель совместим со всеми периферийными устройствами персонального компьютера, работающего в качестве сервера ввода-вывода.

Скорость выполнения задач по сравнению с 8 МГц IBM PC AT увеличивается в 2—8 раз, в зависимости от числа вызовов на систему ввода-вывода персонального компьютера.

Для разработки прикладных программ созданы инструментальные средства языков: ассемблера, C, F77, LISP, Пролог.

Среди множества экспонатов Венгерской Республики — **персональный компьютер ПРОПЕР-132.** В его основе — процессор I80386 с сопроцессором I80387; шина памяти — 32-разрядная 8 Мбайт, адресная — 24 бит, шина ввода-вывода — 16-разрядная, накопитель на жестком магнитном диске емкостью 40... 80 Мбайт, дисплей EGA, производительность 1,5...4 млн. операций/с. Для удобства пользования системный узел разработан в форме стойки (напольный вариант).

На базе четырех микроЭВМ ПРОПЕР-132 создана **локальная вычислительная сеть ПРОНЕТ,** успешно выдержавшая международные совместные испытания в конце прошлого года. В конфигурацию входит локальная сетевая среда на базе аппаратного сбеспечения ARCNET с агрегатной пропускной способностью 2,5 Мбит/с. Связь между сетевой средой и ядром архитектуры осуществляется с помощью NETBIOS.

СупермикроЭВМ типа ТРА-11/530— самая мощная из своего семейства. По производительности — 2,7 млн. операций/с — она сопоставима с мегамини-ЭВМ. Наряду с этим данная ЭВМ характеризуется компактностью за счет высокой степени интеграции составных элементов и большой на-

До 60 % видеотерминалов поставляет Венгерская Республика в социалистические страны. В настоящее время фирма VIDEOTON разрабатывает широкий спектр видеотерминалов. Наиболее последние разработки — семейство монохромных алфавитно-цифровых видеотерминалов VDX 52600 и семейство цветного графического видеотерминала VDC 52700.

Персональный компьютер с платой ускорителя PC на базе архитектуры RISC процессора

Графический таблет Роботрон СМ6422

Роботрон А6470 — система обработки изображений в интерактивном режиме

дежностью. Применяется для разработки и эксплуатации систем CAD/CAM, требующих быстродействия обработки, большой центральной памяти и емкости ввода-вывода.

Ядром этой микроЭВМ является 32-разрядный процессор на СБИС. Емкость ОЗУ — 8...64 Мбайт; 32-разрядная шина памяти (LBUS3); операционная система MOC-VP, обслуживающая виртуальную память, представляет широкий выбор методов программирования.

Большое место в экспозиции Германской Демократической Республики занимал комбинат Роботрон. Его персональные ЭВМ ЕС 1834, СМ 1910, К 1840 и созданные на их основе автоматизированные рабочие места известны далеко за пределами республики. Применение таких рабочих станций для САПР в конструкторских и проектных организациях, например, в машиностроении, электротехнике и электронике повышает производительность до 500 % и сокращает затраты на НИР и ОКР до 20 %.

Роботрон А 6470 — расширяемая система для обработки изображений в интерактивном режиме. Устройство построено по модульному принципу, имеет доступ к отдельным точкам изображения. Эта система может работать в диалоговом и пакетном режимах в целях автоматического и периодического анализа серий изображений. Система без труда приспосабливается к различным режимам мультиспектральной, мультитемпоральной и локальной обработки изображений, начиная с экспериментальной обработки небольшого числа изображений (Роботрон А 6471) и кончая периодичной обработкой на наивысших скоростях обработки (Роботрон А 6472, А 6473).

Роботроновские системы обработки изображений в сочетании с пакетами прикладных программ можно встретить во многих странах мира. Более 60 таких систем и в нашей стране. Широкому применению системы способствуют разработанные в сотрудничестве с советскими специалистами прикладных решений.

С помощью этой системы была произведена дистанционная разведка Земли из космоса, проведен автоматизированный анализ микроскопических снимков в медицине, биологии, материаловедении и т. д.

Максимальный комфорт обслуживания, высокая производительность при выполнении функций и большое рабочее поле полихроматического дисплея с разрешающей способностью 640×480 точек — это преимущества **интерактивного графического терминала Роботрон СМ1647.** Он, как правило, применяется в рамках САПР/ГАП в качестве АРМ для обеспечения работы в диалоговом режиме. Дополнительные преимущества: локальный интеллект, обеспечивающий интерактивную работу и способность к выполнению функций GKS. Основой для этого является внутренняя структура терминала, реализованная на комплексах параллельного управления с несколькими процессорами. Она соответствует развитию в международном масштабе графических терминалов на базе 16-разрядных микропроцессоров, обеспечивающих высокую разрешающую способность и локальную обработку.

Разные режимы отображения выбираются нажатием клавиш. Кроме того, с управляемой микропроцессором компактной клавиатуры ввод информации может производиться и с графического таблета Роботрон СМ6422.

Как утверждают румынские специалисты, разработанная ими **компьютерная система биоэнергетической регулировки «Пульсар С-2000»** на десять лет опережает существующие способы выявления заболеваний.

Известно, что на поверхности кожи имеются зоны (определяемые экспериментально или физико-химическими методами), соответствующие тому или иному внутреннему органу. Данная система позволяет путем электрического и лазерного воздействий выявить зону больного органа, вывести на экран его топографическое изображение, воспроизвести обнаруженные в нем отклонения и определить уровень интенсивности биоэнергодинамического воздействия. Затем полученные данные сопоставить с клиническими и разработать оптимальное лечение.

«Пульсар С-2000» включает базо-

Компьютерная система биоэнергетической регулировки «Пульсар С-2000»

```
;.....................................................;
;                                                     ;
;    DIV2B - ПОДПРОГРАММА ЦЕЛОЧИСЛЕННОГО ДЕЛЕНИЯ      ;
;                                                     ;
;                  СОДЕРЖИМОЕ РЕГИСТРОВ               ;
;    ПЕРЕД ОБРАЩЕНИЕМ : !     ПОСЛЕ ВОЗВРАТА :       ;
;    (HL-DE) - ДЕТИМОЕ   ! (DE)- ЧАСТНОЕ , (HL)- ОСТАТОК ;
;    (BC)    - ДЕЛИТЕЛЬ  ! (BC) - ДЕЛИТЕЛЬ В          ;
;                        !        ДОПОЛНИТЕЛЬНОМ КОДЕ  ;
;                                                     ;
;    НАЧАЛЬНЫЙ АДРЕС ПОДПРОГРАММЫ - 002CH             ;
;.....................................................;
;
;    НАХОЖДЕНИЕ  ДОПОЛНИТЕЛЬНОГО  КОДА  ДЕЛИТЕЛЯ
;
002C 78      DIV2B: MOV   A,B    ; ПОЛУЧЕНИЕ  ОБРАТНОГО  КОДА  (B)
002D 2F             CMA          ;
002E 47             MOV   B,A    ;
002F 79             MOV   A,C    ; ПОЛУЧЕНИЕ  ОБРАТНОГО  КОДА  (C)
0030 2F             CMA          ;
0031 4F             MOV   C,A    ;
0032 03             INX   B      ; УВЕЛИЧЕНИЕ НА  1  ОБРАТНОГО КОДА
;
0033 CD3600  DIV27: CALL  DIV21  ; ДЕЛЕНИЕ 3 СТАРШИХ БАЙТОВ ДЕЛИМОГО
;
;  ПОДПРОГРАММА  ДЕЛЕНИЯ  (HL-D) НА (BC) В ДОПОЛН. КОДЕ
;
0036 7A      DIV21: MOV   A,D    ; ПЕРЕСЫЛКА  (D)  В  A  ДЛЯ ДЕЛЕНИЯ
0037 53             MOV   D,E    ; ЗАПОМИНАНИЕ  (E)
0038 1E08           MVI   E,8    ; ЗАГРУЗКА  СЧЕТЧИКА ЦИКЛОВ
;
;  НАЧАЛО ЦИКЛА , СДВИГ ДЕЛИМОГО ВЛЕВО НА ОДИН РАЗРЯД
;
003A 29      DIV22: DAD   H      ; СДВИГ  СТАРШИХ  БАЙТОВ  ДЕЛИМОГО
003B DA5800         JC    DIV23  ; ПЕРЕХОД  ПРИ  ПЕРЕПОЛНЕНИИ
003E 87             ADD   A      ; СДВИГ (A), ОЧИСТКА БИТА ЧАСТНОГО
003F D24300         JNC   DIV24  ; ПЕРЕХОД  ПРИ ОТСУТСТВИИ ПЕРЕНОСА
0042 23             INX   H      ; ДОБАВЛЕНИЕ  ЕДИНИЦЫ  ПЕРЕНОСА
;
;  ВЫЧИТАНИЕ  ДЕЛИТЕЛЯ   ИЗ ДЕЛИМОГО
0043 E5      DIV24: PUSH  H      ; ЗАПОМИНАНИЕ СТАРШИХ БАЙТОВ ДЕЛИМОГО
0044 09             DAD   B      ; ВЫЧИТАНИЕ ДЕЛИТЕЛЯ ИЗ ДЕЛИМОГО
0045 D25100         JNC   DIV25  ; ПЕРЕХОД  ПРИ  НАЛИЧИИ  ЗАЕМА
;
;  УСТАНОВКА БИТА ЧАСТНОГО В ЕДИНИЦУ ПРИ ОТСУТСТВИИ ЗАЕМА
0048 33             INX   SP     ; ВОССТАНОВЛЕНИЕ УКАЗАТЕЛЯ СТЕКА
0049 33             INX   SP     ;
004A 3C             INR   A      ; ЗАСЫЛКА ЕДИНИЦЫ В БИТ ЧАСТНОГО
004B 1D             DCR   E      ; УМЕНЬШЕНИЕ СЧЕТЧИКА  ЦИКЛА
004C C23A00         JNZ   DIV22  ; ЕСЛИ НЕ НУЛЬ, ТО ПЕРЕХОД НА НАЧАЛО
004F 5F             MOV   E,A    ; ПЕРЕСЫЛКА БАЙТА ЧАСТНОГО В  E
0050 C9             RET          ; ВЫХОД  ИЗ  ПОДПРОГРАММЫ
;
;  УСТАНОВКА  БИТА ЧАСТНОГО В НУЛЬ  ПРИ  НАЛИЧИИ  ЗАЕМА
0051 E1      DIV25: POP   H      ; ВОССТАНОВЛЕНИЕ ДЕЛИМОГО ПРИ ЗАЕМЕ
0052 1D             DCR   E      ; УМЕНЬШЕНИЕ  СЧЕТЧИКА  ЦИКЛА
0053 C23A00         JNZ   DIV22  ; ЕСЛИ НЕ НУЛЬ, ТО ПЕРЕХОД НА НАЧАЛО
0056 5F             MOV   E,A    ; ПЕРЕСЫЛКА БАЙТА ЧАСТНОГО В  E
0057 C9             RET          ; ВЫХОД  ИЗ  ПОДПРОГРАММЫ
;
; ЗАВЕРШЕНИЕ СДВИГА И ФОРМИРОВАНИЯ ЧАСТНОГО ПРИ
                                   ПЕРЕПОЛНЕНИИ
0058 8F      DIV23: ADC   A      ; СДВИГ И ЗАСЫЛКА  1  В БИТ ЧАСТНОГО
0059 D25D00         JNC   DIV26  ; ПЕРЕХОД ПРИ ОТСУТСТВИИ ПЕРЕНОСА
005C 23             INX   H      ; ДОБАВЛЕНИЕ  ЕДИНИЦЫ  ПЕРЕНОСА
005D 09      DIV26: DAD   B      ; ВЫЧИТАНИЕ ДЕЛИТЕЛЯ ИЗ ДЕЛИМОГО
005E 1D             DCR   E      ; УМЕНЬШЕНИЕ  СЧЕТЧИКА  ЦИКЛА
005F C23A00         JNZ   DIV22  ; ЕСЛИ НЕ НУЛЬ, ТО ПЕРЕХОД НА НАЧАЛО
0062 5F             MOV   E,A    ; ПЕРЕСЫЛКА БАЙТА ЧАСТНОГО В  E
0063 C9             RET          ; ВЫХОД  ИЗ  ПОДПРОГРАММЫ
```

Рис 2. Подпрограмма целочисленного деления DIV2B

Для удобства пользователей подпрограммы DIV4B и DIV2B написаны без макросредств. Кроме того, подпрограмма DIV4B размещается с нулевого адреса, что позволяет легко перенести ее с любого другого адреса, увеличив адрес в командах передачи управления.

Телефон 9-84-48, Черновцы, Родымюк А. М.

ЛИТЕРАТУРА

1. Джерри Л. Гудрич. Очень эффективная программа умножения и деления для микропроцессора 18080 // Электроника.— 1982.— Т. 55.— № 4. С. 74—75.
2. Глазов А. Б., Костарев С. А., Суханова Е. В. Эффективные программы умножения для микропроцессора КР580ИК80А // Микропроцессорные средства и системы.— 1986.— № 5.— С. 43—44.

Сообщение поступило 18.01.88

УДК 681.3.06

В. А. Гришин

РАБОТА С ДАТАМИ В БАЗЕ ДАННЫХ КОМПЛЕКСА РТК МИКРО

Для микроЭВМ «Электроника 60» и ДВК наибольшее распространение получила СУБД реляционного типа, имеющая экранный и программный интерфейсы и окруженная набором сервисных и инструментальных средств технологического комплекса РТК МИКРО. Однако в редакциях данной СУБД отсутствует возможность работы с данными типа ДАТА, что сужает область ее применения.

Решать эту задачу можно по-разному, например ввести новый тип данных в базу данных, хотя это и трудоемко, к тому же связано с наличием у пользователя исходных модулей СУБД и требует доработки всех программ СУБД. Другой способ — добавить к программному интерфейсу СУБД две процедуры, не нарушая программ комплекса РТК МИКРО.

Одна из разработанных с этой целью процедур (рис. 1) принимает на вход дату как строку символов в формате ГГММДД и преобразует ее в число дней, прошедших с 31 декабря 1940 г. до введенной даты. Другая процедура выполняет обратное преобразование (рис. 2).

При использовании этих процедур даты в базе данных могут храниться и вызываться в экранном интерфейсе, например как символьные строки фиксированной длины. Если принять для дат форматы ГГММДД, ГГ-ММ-ДД или ГГ.ММ.ДД, то к ним можно применить операции сравнения, что позволяет реализовать многие функции поиска и выборки данных по хранимым в базе данных датам. Для выполнения арифметических операций даты средствами программного интерфейса считываются из базы данных, преобразуются с помощью процедуры DACHI в число дней. После необходимых вычислений результаты процедурой CHIDA могут быть преобразованы в даты как строки символов, помещены в базу данных, выведены на печать или дисплей в удобочитаемом виде.

Описанный способ работы с датами реализован в прикладных программах, которые на этапе редактирования свя-

```
LINE  STMT LEVEL  NEST   SOURCE STATEMENT

   1                     PROCEDURE DACHI(DT1:ALFA;VAR KLD1:INTEGER);
   2                     /* ПРЕОБРАЗОВАНИЕ ГГММДД В ЧИСЛО ДНЕЙ */
   3                     CONST RUS=16B;LAT=17B;
   4                      VAR
   5                       I,G1,M1,D1:INTEGER;
   6                       DM1:ARRAY[1..12] OF  INTEGER;
   7                     BEGIN
   8     1     2     1     DM1[1]:=31;DM1[2]:=28;DM1[3]:=31;
   9     4     2     1     DM1[4]:=30;DM1[5]:=31;DM1[6]:=30;
  10     7     2     1     DM1[7]:=31;DM1[8]:=31;DM1[9]:=30;
  11    10     2     1     DM1[10]:=31;DM1[11]:=30;DM1[12]:=31;
  12    13     2     1      G1:=(ORD(DT1[1])-48)*10+ORD(DT1[2])-48;
  13    14     2     1      M1:=(ORD(DT1[3])-48)*10+ORD(DT1[4])-48;
  14    15     2     1      D1:=(ORD(DT1[5])-48)*10+ORD(DT1[6])-48;
  15    16     2     1      IF (G1>127)OR(G1<40)OR(M1>12)OR
  16                           (M1<1)OR(D1>31)OR(D1<1) THEN
  17    17     2     2       WRITELN(CHR(RUS),'ДАТА НЕВЕРНА',CHR(LAT))

  18    18     2     2      ELSE BEGIN
  19    19     2     3      G1:=G1-40;
  20    20     2     3      KLD1:=(G1-1)*365+(G1-1) DIV 4+D1;
  21    21     2     3      FOR I:=1 TO M1-1 DO KLD1:=KLD1+DM1[I];
  22    23     2     3      IF (M1>2)AND(G1 MOD 4=0) THEN KLD1:=KLD1+1;
  23    25     2     3     END;END;
```

Рис. 1

```
LINE  STMT LEVEL  NEST   SOURCE STATEMENT

   1                     PROCEDURE CHIDA(KLD2:INTEGER;VAR DT2:ALFA);
   2                     /* ПРЕОБРАЗОВАНИЕ ЧИСЛА В ДАТУ:ГГММДД */
   3                      VAR
   4                       I,J,G2,M2,D2:INTEGER;
   5                       DG2:ARRAY[1..4] OF INTEGER;
   6                       DM2:ARRAY[1..12] OF INTEGER;
   7                     BEGIN
   8     1     2     1     DG2[1]:=365;DG2[2]:=365;
   9     3     2     1     DG2[3]:=365;DG2[4]:=366;
  10     5     2     1     DM2[1]:=31;DM2[2]:=28;DM2[3]:=31;
  11     8     2     1     DM2[4]:=30;DM2[5]:=31;DM2[6]:=30;
  12    11     2     1     DM2[7]:=31;DM2[8]:=31;DM2[9]:=30;
  13    14     2     1     DM2[10]:=31;DM2[11]:=30;DM2[12]:=31;
  14    17     2     1      G2:=0;WHILE KLD2>0 DO
  15    19     2     2       BEGIN
  16    20     2     3        FOR I:=1 TO 4 DO
  17    21     2     4         BEGIN IF KLD2>0 THEN BEGIN J:=I;
  18    25     2     7          G2:=G2+1;KLD2:=KLD2-DG2[J];
  19    27     2     7         END;END;
  20    29     2     3       END;KLD2:=KLD2+DG2[J];
  21    31     2     1      M2:=0;IF (G2 MOD 4)=0 THEN DM2[2]:=29;
  22    34     2     1      WHILE KLD2>0 DO
  23    35     2     2       BEGIN
  24    36     2     3        M2:=M2+1;KLD2:=KLD2-DM2[M2];
  25    38     2     3       END;D2:=KLD2+DM2[M2];
  26    40     2     1        G2:=G2+40;
  27    41     2     1       DT2[1]:=CHR(G2 DIV 10 + 48);
  28    42     2     1       DT2[2]:=CHR(G2 MOD 10 + 48);
  29    43     2     1       DT2[3]:=CHR(M2 DIV 10 + 48);
  30    44     2     1       DT2[4]:=CHR(M2 MOD 10 + 48);
  31    45     2     1       DT2[5]:=CHR(D2 DIV 10 + 48);
  32    46     2     1       DT2[6]:=CHR(D2 MOD 10 + 48);
  33    47     2     1     END;
```

Рис. 2

зей объединялись с процедурами DACHI и CHIDA. Влияние этих процедур на быстродействие программ в процессе работы с базой данных определялось посредством измерения времени их работы и работы некоторых процедур и функций программного интерфейса. Результаты измерений приведены ниже.

Процедура	Время работы, мс
DACHI	2,9
CHIDA	12,3
RGOL	71
RREC	83
BI	0,43
BP^N	0,78

Измерения проводились на вычислительном комплексе 15ВУМС28-025 с базой данных, содержащей 200 записей, 35 столбцов.

Телефон 171-06-59, Москва

Сообщение поступило 30.01.89

УДК 681.3.06

Д. В. Барковский, А. П. Жарков, И. Г. Креславский

ПРОГРАММНЫЙ ПАКЕТ МЕТАКС — ИНСТРУМЕНТ ДЛЯ ГЕНЕРАЦИИ АССЕМБЛЕРОВ

Пакет программ МЕТАКС представляет собой метаассемблер, способный определять множество языков ассемблера и вести трансляцию на любом из них.

Разработанный пакет программ отличается от существующих метаассемблеров тем, что обеспечивает поддержку процессов использования и эффективной генерации трансляторов с ассемблера для различных архитектур микропроцессорных систем. Пакет МЕТАКС позволяет: выбирать и определять различные системы команд и языки ассемблера в рамках доступного МЕТАКС их множества; программировать алгоритмы работы микропроцессоров в терминах выбранного языка; транслировать программы в машинный код целевого устройства.

Пакет программ МЕТАКС как набор кроссассемблеров. Это основной режим взаимодействия МЕТАКС с пользователем. МЕТАКС обрабатывает текстовые файлы с записанными в них программами на языке ассемблера. Пользователь указывает, каким из ассемблеров, на которые МЕТАКС настроен, он пользуется. При работе с выбранным языком МЕТАКС реализует основные, стандартные для языков ассемблера, возможности:

переводит записанные, в соответствии с известными МЕТАКС правилами ассемблера, мнемонические обозначения команд языка в коды машинных команд;

фиксирует синтаксические и логические ошибки в исходном тексте программы;

осуществляет сбор программных меток и реализует их абсолютную и косвенную адресацию;

позволяет вводить символические обозначения констант;

вычисляет выражения (с учетом приоритета операций) при определении операндов команд и символьных констант;

формирует (в различных форматах) файл машинного кода, используемый для дальнейшей отладки программ на аппаратных средствах;

формирует листинг программы на ассемблере, который включает исходный текст, адреса команд и генерируемый машинный код (в необходимых системах счисления), пояснительные тексты ошибок, таблицы имен, используемых в программе;

выполняет макроподстановки;

производит (при необходимости) условную трансляцию.

Ход процесса трансляции определяется директивами, общими для всего множества возможных ассемблеров. Это директивы управления программным счетчиком, условной трансляцией, выдачей листинга, начала и конца программы и др. Примеры использования МЕТАКС приведены на рис. 1, *а, б*.

Настройка пакета программ МЕТАКС на новый язык ассемблера. Для включения в МЕТАКС нового языка ассемблера необходима каталогизация сведений: возможные команды ассемблера и допустимые, различающиеся структуры записи операндов каждой команды языка; символьные имена и константы, которые могут употребляться в качестве операндов; соответствия между командами ассемблера, записью операндов и кодами команд, выполняемых микропроцессором. Данная информация излагается пользователем на специальном (определенном в МЕТАКС) языке описания систем команд и переводится им во внутреннее представление.

Определение ассемблера заключается в описании: мнемоник команд ассемблера (имен); синтаксиса записи операндов команды; количества, расположения и принимаемых значений полей бит в машинных инструкциях, их соответствия числовым значениям операндов; символов и констант, возможно используемых в качестве операндов, и соответствующих числовых значений.

МЕТАКС документирует процесс формирования ассемблера, выдавая листинг исходного текста, описания и таблицы для проверки их корректности. На рис. 2 демонстрируется пример работы МЕТАКС в режиме компиляции описаний ассемблеров. Определение команды следует за директивой PATTERNS. Объявлено 18 типов адресации. Первой описывается команда непосредственной пересылки содержимого аккумулятора в регистр. Данный вид адресации задают следующие элементы определения: длина соответствующей кодовой инструкции 8 бит (может быть задана по умолчанию); синтаксис записи поля операндов — «A,%_», где «_» в общем случае задает переменное числовое значение или символьное имя из числа введенных директивой ASSEMBLER CONSTANT (конкретно здесь должен устанавливаться номер используемого регистра); шаблон генерируемой инструкции — [3V, '11101'B], устанавливающий порядок следования битовых полей кодируемой инструкции и включающий в себя конструкции языка описания шаблонов, с помощью которых полям битов присваиваются переменные или постоянные значения. Так первые три бита инструкции зависят от переменного значения операнда команды (номера регистра): 3V (три переменных бита). '11101'B — двоичная константа, устанавливающая остальные 5 бит команды. Аналогично определены все последующие разновидности команд пересылки.

Механизм функционирования пакета программ МЕТАКС. Программный пакет МЕТАКС можно отнести к классу адаптивных метаассемблеров, реализующих свободную нотацию полей [1]. Если рассматривать структуру команды ассемблера в общем виде, то в ней можно выделить следующие поля: ⟨метка⟩:⟨имя команды⟩ ⟨операнды⟩; ⟨комментарии⟩, например LAB:MOV A,R1#CON; пересылка

Метаассемблер отличается от ассемблера тем, что поля ⟨имя команды⟩ и ⟨операнды⟩ не имеют предопределенного множества значений и допускают их определение пользователем, причем фиксирование таких множеств ведет к превращению метаассемблера в обычный ассемблер. Работа метаассемблера (рис. 3) разделена соответственно на две фазы: настройки, в которой определяется множество символов, допустимых к использованию в качестве значений полей ⟨имя команды⟩ и ⟨операнды⟩, и транс-

```
 МЕТААССЕМБЛЕР МЕТАКС В02.04 : ФАЗА ТРАНСЛЯЦИИ  26-ОКТ-87
МОНИТОР  КЛАВИАТУРЫ

 1 # TITLE "МОНИТОР  КЛАВИАТУРЫ "
 2 # MICRO
 3
 4 0000 26 80     START:  MVI     H,'80'H ;
 5 0002 DB FD     NO1:    IN      'FD'H   ;
 6 0004 E6 OF             ANI     'F'H    ;
 7 0006 C2 00 00          JNZ     START   ;
 8 0009 25                DCR     H       ;
 9 000A C2 00 02          JNZ     NO1     ;
10                \ ЦИКЛ ОПРОСА  КЛАВИАТУРЫ
11 000D 06 EF     BEGIN:  MVI     B,'EF'H ;
12 000F OE 04             MVI     C, 4    ;
13                \ B - СКАНИРУЮЩИЙ  НАБОР
14                \ C - ВЕС ТЕКУЩЕЙ  ГОРИЗОНТАЛИ
15 0011 78                MOV     A,B     ;
16 0012 D3 FO     LOOP1:  OUT     'FO'H   ;
17 0014 DB FO             IN      'FO'H   ;
18 0016 6F                MOV     L,A     ; СОХРАНЕНИЕ  ОТКЛИКА В L
19 0017 E6 OF             ANI     'F'H    ; ВЫДЕЛЕНИЕ  МЛАДШЕЙ  ТЕТРАДЫ
20 0019 C2 00 26          JNZ     YES1    ;
21                \ ИНАЧЕ  ПОЛУЧЕН  НУЛЕВОЙ  ОТКЛИК
22 001C OD                DCR     C       ;
23 001D CA 00 OD          JZ      BEGIN   ;
24                \ ИНАЧЕ  НЕ  ВСЕ  ГОРИЗОНТАЛИ  ОПРОШЕНЫ
25 0020 78                MOV     A,B     ;
26 0021 07                RLC             ;
27 0022 47                MOV     B,A     ;
28 0023 C3 00 12          JMP     LOOP1   ;
29 0026 26 80     YES1:   MVI     H,'80'H ;
30                \ УСТРАНЕНИЕ  ДРЕБЕЗГА
31 0028 DB FD     LOOP2:  IN      'FD'H   ;
32 002A BD                CMP     L       ; СРАВНЕНИЕ  С  ОБРАЗЦОМ
33 002B C2 00 OD          JNZ     BEGIN   ;
34                \ ПОЛУЧЕННЫЙ  ОТКЛИК  ЯВЛЯЕТСЯ  ВЕРНЫМ
35 002E 7D                MOV     A,L     ;
36 002F E6 08             ANI     8       ;
37 0031 C2 00 40          JNZ     CNTR    ;
38                \ ИНАЧЕ  КНОПКА  НЕ  УПРАВЛЯЮЩАЯ
39 0034 79                MOV     A,C     ;
40 0035 3D                DCR     A       ;
41 0036 07                RLC             ;
42 0037 07                RLC             ;
43 0038 4F                MOV     C,A     ;
44 0039 7D                MOV     A,L     ;
45 003A 3D                DCR     A       ;
46 003B E6 OF             ANI     'F'H    ;
47 003D 81                ADD     C       ;
48                \ ВЕС  КЛАВИШИ  ВЫЧИСЛЕН
49 003E 79                MOV     A,C     ;
50 003F C9                RET             ;
51                \ ОБРАБОТКА  УПРАВЛЯЮЩЕЙ  КНОПКИ
52 0040 7D        CNTR:   MOV     A,L     ;
53 0041 3D                DCR     A       ;
54 0042 E6 OF             ANI     'F'H    ;
55 0044 47                MOV     B,A     ;
56                \ ИЗМЕНЕНИЕ АДРЕСА  ВОЗВРАТА
57 0045 D1                POP     DE      ;
58 0046 C9                RET             ;
59 # END
```

а

Рис. 1. Листинги, формируемые МЕТАКС при трансляции программы, составленной на языке ассемблера микропроцессора КР580ИК80 (*а*) и ОЭВМ КI816ВЕ51 (*б*) → см. на с. 36

```
МЕТААССЕМБЛЕР МЕТАКС В02.04 : ФАЗА ТРАНСЛЯЦИИ  20-АПР-87
УМНОЖЕНИЕ 8 НА 8 РАЗРЯДОВ

  1 # TITLE "УМНОЖЕНИЕ 8 НА 8 РАЗРЯДОВ"
  2 # CONST
  3         XA = R2 ,
  4         COUNT = R3
  5 # MICRO
  6 # ORIGIN ´35´H
  7 0035 BA 00 MPY8X8: MOV     XA,#00        ; MULTIPLICAND[0]:=0
  8 0037 BB 08         MOV     COUNT,#8      ; COUNT:=8
  9 0039 12 43 MPY8LP: JBO     MPY8A         ; REPEAT
 10 003B 2A           XCH     A,XA          ;  IF MULTIPLICAND[0] = 0
 11 003C 97           CLR     C             ;  THEN
 12 003D 67           RRC     A             ;   BEGIN
 13 003E 2A           XCH     A,XA          ;    MULTIPLICAND :=
 14 003F 67           RRC     A             ;    MULTIPLICAND / 2
 15 0040 EB 39        DJNZ    COUNT,MPY8LP  ;   END
 16 0042 83           RET                   ;  ELSE BEGIN
 17 0043 2A    MPY8A: XCH     A,XA          ;    MULTIPLICAND[15-8] :=
 18 0044 61           ADD     A,@R1         ;    MULTIPLICAND[15-8]+MULTIPLIER
 19 0045 67           RRC     A             ;    MULTIPLICAND :=
 20 0046 2A           XCH     A,XA          ;    MULTIPLICAND / 2
 21 0047 67           RRC     A             ;   END
 22 0048 EB 39        DJNZ    COUNT,MPY8LP  ;  COUNT := COUNT - 1
 23 004A 83           RET                   ; UNTIL COUNT=0
 24
 25 # END

******* ОБНАРУЖЕНО ОШИБОК :  0
```

Рис. 1б

б

```
МЕТААССЕМБЛЕР МЕТАКС В02.04 : ФАЗА ОПРЕДЕЛЕНИЙ  23-ОКТ-87
MCS-51 INSTRUSTION SET

  1  # TITLE "MCS-51 INSTRUSTION SET"
  2  # ASSEMBLER CONSTANT
  3     P0   = ´10000000´B , P1   = ´10010000´B ,
  4     P2   = ´10100000´B , P3   = ´10110000´B ,
  5     TCON = ´10001000´B , SCON = ´10011000´B ,
  6     IE   = ´10101000´B , IP   = ´10111000´B ,
  7     PSW  = ´11010000´B , ACC  = ´11100000´B ,
  8     B    = ´11110000´B
  9
 10  # PATTERNS
 11  MOV  >
 12           8:      "A,#_"          [ 3V , ´11101´B    ] ,
 13           16:     "A,_"           [ 8V , ´11100101´B ] ,
 14           8:      "A,@#_"         [ 1V , ´1110011´B  ] ,
 15           16:     "A,#_"          [ 8V-, ´01110100´B ] ,
 16           8:      "#_,A"          [ 3V , ´11111´B    ] ,
 17           16:     "#_,_"          [ 8:3V, 0:8V,  3P, ´10101´B ] ,
 18                   "#_,#_"         [ 8:3V, 0:8V-, 3P, ´01111´B ] ,
 19                   "_,A"           [ 8V , ´11110101´B ] ,
 20                   "_,#_"          [ 8V , 3V , ´10001´B    ] ,
 21           24:     "_,_"           [ 8V , 8V , ´10000101´B ] ,
 22           16:     "_,@#_"         [ 8V , 1V , ´1000011´B  ] ,
 23           24:     "_,#_"          [ 8:8V , 0:8V- , 8P , ´01110101´B ] ,
 24           8:      "@#_,A"         [ 1V , ´1111011´B ] ,
 25           16:     "@#_,_"         [ 8:1V , 0:8V ,  1P , ´1010011´B ] ,
 26                   "@#_,#_"        [ 8:1V , 0:8V- , 1P , ´0111011´B ] ,
 27           16:     "C,_"           [ 8V , 0:3V, 5P, ´10100010´B ] ,
 28           16:     "_,C"           [ 8V , 0:3V, 5P, ´10010010´B ] ,
 29           24:     "DPTR,#_"       [ 16V, ´10010000´B ] ;
 30  # END

******* ОБНАРУЖЕНО ОШИБОК : 0
```

Рис. 2. Пример описания команды MOV языка ассемблера ОЭВМ К1816ВЕ51

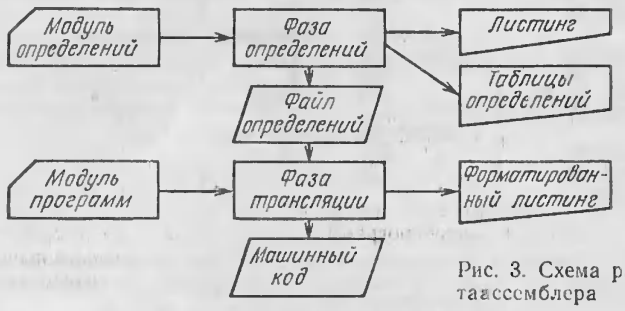


Рис. 3. Схема работы метаассемблера МЕТАКС

ляции, в которой значения всех полей ассемблерной команды конкретизируются.

Согласно фазам работы метаассемблера в МЕТАКС выполняются разные программные компоненты. Программа фазы настройки обрабатывает текстовый файл, в котором описано множество команд ассемблера, и генерирует файл определений, служащий «словарем» языка для следующей фазы. Программа фазы трансляции использует этот файл для определения правил трансляции текстовых файлов с программами на соответствующем языке ассемблера и, обрабатывая их, генерирует файл машинного кода. Таким образом, кроссассемблер состоит из программы фазы трансляции и набора файлов определений, а пополняется созданием еще одного файла определений нового языка.

Если поле имени команды состоит, как правило, лишь из символьного имени, то поле операндов, как известно из практики, зачастую может иметь достаточно сложную синтаксическую структуру. Принцип свободной нотации допускает относительную свободу в применении различных способов записи операндов, при этом определяется множество форм, которые может принимать это поле. В МЕТАКС для распознавания значения, принимаемого полем операндов, применен метод выделения идентификатора записи. В его основу положен алгоритм, имеющий на входе строку записи операндов, а на выходе строку-идентификатор. В идентификаторе фиксируются (в порядке поступления): все свободные символы, не являющиеся знаками операций; символьные константы, не имеющие числовых значений; операнды, имеющие числовые значения (численные, символьные константы, имена меток и выражения с ними).

Например, в команде MOV A,#32 идентификатор записи поля операндов может быть представлен, как «A,#__», т. е. эквивалентно определению синтаксиса в фазе настройки. Распознавая команду по ее имени и идентификатору поля записи операндов в фазе трансляции, МЕТАКС генерирует соответствующий ей машинный код, который даже для одного имени команды может быть разной длины.

Такой способ функционирования МЕТАКС накладывает ограничения на возможное множество языков ассемблера. Согласно [2] оно включает языки с нерегулярными системами команд. Необходимо подчеркнуть, что свободная нотация обеспечивает максимально полное приближение создаваемого языка к существующему стандарту, что видно из приведенных листингов. Тем не менее, издержки универсализации могут возникать, однако специфичные детали языков можно учесть макроподстановками. В большинстве случаев достигается полная переносимость программ, отсутствие которой является су-

ственным недостатком некоторых известных метаассемблеров [3, 4].

Реализация пакета программ МЕТАКС. Использование программ пакета МЕТАКС возможно на любой ЭВМ, обладающей дисковой операционной системой, включающей компилятор с языка Паскаль. Однако программные средства пакета ориентированы на персональные и микроЭВМ, что позволяет не только обеспечить его доступность широкому кругу разработчиков микропроцессорной техники, но и повысить оперативность проектирования за счет проведения этапов разработки программного обеспечения и отладки аппаратных средств в рамках одной инструментальной микроЭВМ. Дефицит оперативной памяти некоторых микроЭВМ преодолевается оверлейным построением программ пакета, механизмом динамического выделения памяти. Разработаны и используются версии пакета МЕТАКС на микроЭВМ ДВК2М, ЕС1840, генерирующие абсолютный машинный код.

Возможна быстрая установка МЕТАКС на другие ЭВМ, а также разработка версии, генерирующей перемещаемый объектный код. В этом случае пакет будет включать в себя также программы компоновщика и библиотекаря. Тестирование пакета МЕТАКС проводилось на примерах разработки программного обеспечения КР580ИК80 и ОЭВМ К1816ВЕ51. Пакет МЕТАКС поставляется по договору о приобретении программного средства. По просьбе заказчика МЕТАКС настраивается на требуемый язык ассемблера, сопровождаются и дорабатываются программы пакета.

Телефон 324-99-81, Москва

ЛИТЕРАТУРА

1. Skordalakis E. Meta-assemblers //IEEE Micro.— 1983.— Vol. 3, N 2.— P. 6—16.
2. Уокерли Дж. Архитектура и программирование микроЭВМ: в 2-х книгах / Пер. с англ.— М.: Мир, 1984.— Кн. 2.— С. 154.
3. Heath R., Patel S. How to write a Universal Cross-Assembler // IEEE Micro.— 1981.— Vol. 1, N 4.
4. Лукьянов Д. А. Как написать кросс-транслятор с языка ассемблер //Микропроцессорные средства и системы.— 1985.— № 4.— С. 35—41.

Статья поступила 15.12.87

УДК 681.3.06

И. Г. Креславский, Д. В. Барковский, А. П. Жарков, Н. В. Синицын

МИКРОПРОГРАММНЫЙ АССЕМБЛЕР МАСС

В настоящее время при создании специализированных вычислительных и управляющих устройств широко используются секционированные микропроцессоры (СМП) [1—5]. Основное средство автоматизации проектирования ПО для СМП — микропрограммный ассемблер или микроассемблер (МА).

Первый обзор МА для СМП был опубликован в 1978 г. [6]. В работе рассмотрена трехфазная структура построения, приведены примеры работы и особенности таких МА, как AMDASM, CROMIS, RAPID, DAPL, дан сравнительный анализ и сформулированы качественные критерии оценки МА.

В обзоре [7] подведены

Таблица 1

Функциональные и технические характеристики МА МАСС

Наименование	Состояние	Примечание
Общие характеристики		
Тип МА	—	Адаптивный
Язык реализации	—	Паскаль
Машинная независимость	Да	—
Возможность настройки на различные микропроцессорные комплекты	Да	К1804, К1802, К1800, К589, К583, К1838, К1832, К1822 и др.
Модульное микропрограммирование	Нет	—
Переменная длина микрокоманды, бит	Да	До 512
Число переменных полей, определяемых на шаблоне микрокоманды	—	До 256
Максимальная длина константы, поля шаблона, бит	—	16
Возможность установки полей по умолчанию	Да	—
Контроль неиспользуемых разрядов	Да	В обеих фазах
Свободный синтаксис входных языков	Да	—
Возможность ведения комментариев	Да	В обеих фазах
Контроль свободной оперативной памяти	Да	»
Измерение времени трансляции	Да	»
Фаза определений		
Объем свободной оперативной памяти ДВК, Кбайт	Да	Свыше 24
Специальный язык описания шаблонов	Да	3 типа команд, 7 команд языка
Использование макроопределений	Да	—
Выбор направления передвижения маркера при описании шаблона	Да	—
Условная трансляция	Да	До 8 уровней вложения
Использование выражений	Да	3 унарных и 11 бинарных операций, 6 уровней приоритета, скобки
Управление выдачей листинга	Да	6 директив, 3 ключа
Документирование символьных таблиц	Да	3 вида таблиц
Просмотр созданных таблиц определений	Да	2 вида таблиц
Форматирование таблицы шаблонов	Да	По ширине вывода
Синтаксический контроль, обнаружение и идентификация ошибок	Да	Свыше 50 регистрируемых ошибок, указание места ошибки
Атрибутирование переменных полей	Да	5 атрибутов
Балансировка таблиц шаблонов, констант	Да	—
Фаза трансляции		
Объем свободной оперативной памяти ДВК, Кбайт		Свыше 20
Скорость трансляции ДВК, байт/мин		Не менее 1000
Форматы хранения машинного кода		2 формата
Использование макроопределений	Да	До 4 уровней вложения, до 10 параметров
Создание и подключение макробиблиотек	Да	—
Условная трансляция микропрограмм	Да	До 10 уровней вложения, использование в макроопределениях

Продолжение табл. 1.

Наименование	Состояние	Примечание
Использование выражений	Да	3 унарных, 13 бинарных операций, 7 уровней приоритета, скобки
Управление микропрограммным счетчиком	Да	3 директивы
Различные форматы вывода листинга	Да	Параллельно-построчный, расслоенный, блочный
Форматирование листинга	Да	7 директив, 5 ключей
Объединение микрокоманд в микрослове	Да	—
Использование локальных блоков	Да	До 20 уровней вложения
Синтаксический контроль, обнаружение и идентификация ошибок	Да	Свыше 60 регистрируемых ошибок, указание места ошибки
Документирование символьных таблиц	Да	3 вида таблиц
Построение таблиц перекрестных ссылок	Да	4 вида таблиц

итоги восьмилетнего развития МА как одного из классов метаассемблеров, классифицированы основные методы построения и сформулированы свойства «идеального» метаассемблера, уточнено понятие мощности ассемблера, подробно обсужден вопрос влияния синописа языка ассемблера на схемы генерации машинного кода. В итоговой таблице собраны сведения о 30 метаассемблерах, оценка которых проведена более чем по 25 параметрам.

Дальнейшее развитие и совершенствование МА привело к созданию систем микропрограммирования. Самые мощные на сегодня — M29 фирмы Advanced Micro Devices [8] и METASTEP фирмы Step Engineering [9].

В нашей стране известны такие средства автоматизации проектирования ПО МПУ на СМП, как системы АСПРОМ, МЕТАМИКРО [10], микроассемблеры ГНОМ [11], МЕАСС [12] и др.

Требования к современным микроассемблерам: возможность настройки на архитектуру микропрограммируемых устройств, независимость от микропроцессорного комплекта, высокая мобильность, гибкий язык описания форматов микрокоманд, установка полей микрокоманд по умолчанию, контроль используемых в микрокомандах разрядов, надежный механизм кодирования микрокоманд, простой и выразительный синтаксис записи входных модулей, возможность ведения комментариев, документирование работы отдельных фаз МА; обнаружение и идентификация ошибок, вывод символьных таблиц для использования на этапе отладки микропрограмм.

Мощность МА определяется возможностью поддержки модульного микропрограммирования, реализации макросредств и управления процессом трансляции, обработки сложных выражений. Технические характеристики МА включают в себя: вид инструментальной ЭВМ, язык программирования, тип МА в соответствии с классификацией [7], максимальную длину микрокоманды, скорость трансляции микропрограмм, предельные объемы символьных таблиц.

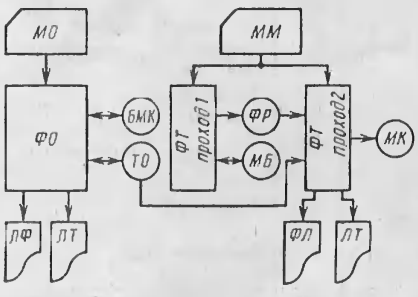


Рис. 1. Общая схема работы МА МАСС:
БМК — библиотека метаконстант, ТО — таблица определений, ЛФ — листинг фазы, ЛТ — листинг таблиц, МБ — макробиблиотека, ФР — файл расширения, ФЛ — форматированный листинг, МК — машинный код

Цели создания микропрограммного ассемблера

Перечисленные требования были положены в основу разработки МА МАСС, так как ни один из известных отечественных МА не удовлетворяет им в полной мере (табл. 1). При создании МАСС также решались задачи:

реализации МА для эксплуатации на микроЭВМ в условиях ограниченных объемов оперативной памяти;

достижения максимальной машинной независимости МА;

увеличения производительности труда микропрограммиста, снижения вероятности случайных ошибок кодирования программ за счет включения дополнительных сервисных возможностей и использования надежных механизмов генерации микрокоманд;

значительного увеличения мощности МА средствами параметрических макрорасширений, макробиблиотек и условной трансляции микропрограмм;

обеспечения возможности использования алгоритмических конструкций структурного программирования при кодировании микропрограмм;

организации вывода листинга в формате /Адрес /Машинный код/ Текст микропрограммы/ с широкими возможностями по форматированию каждой зоны листинга;

использования простого и легкочитаемого синтаксиса с минимумом ограничений на форму представления текстовой информации.

Работа МА разделена на две фазы: фаза определений (ФО) обрабатывает описание системы команд проектируемого МПУ, фаза трансляции (ФТ) ассемблирует микропрограммы и генерирует абсолютный машинный код (рис. 1). МАСС относится к классу адаптивных МА с нотацией ассемблерной строки функционального типа [7]. Работа таких МА основана на использовании шаблонов микрокоманд (МК), которые задают дискретную функцию в пространстве команд, выполняемых МП устройством. Вызов шаблона из микропрограммы определяет единственную команду из множества данного шаблона.

В модуле определений (МО), который является входным для ФО, программист указывает длину МК и вводит ее шаблоны и константы, необходимые на этапе кодирования микропрограмм. Для символического обозначения функций, выполняемых отдельными ИМС микропроцессорного комплекта, служат метаконстанты, которые определяют фиксированные поля шаблонов МК. Для настройки МА на рабочий микропроцессорный комплект можно организовать библиотеку метаконстант.

Шаблоны МК описываются на специальном языке. Команды языка подразделяются на три группы: перемещения по шаблону, определения фиксированных и переменных полей. Программист свободен в выборе имен шаблонов и констант.

Фаза определений обрабатывает линейные макроподстановки в режиме ввода шаблонов, что делает их запись выразительной и легкой для прочтения другими программистами. Возможность условной трансляции в модуле позво-

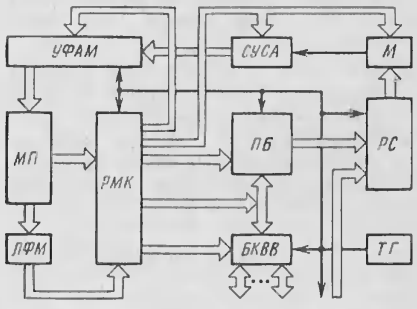


Рис. 2. Структурная схема СПВВ:
ПБ — процессорный блок, МП — микропрограммная память, РМК — регистр микрокоманды, ЛФМ — логика формата микрокоманд, УФАМ — устройство формирования адреса микропрограммы, СУСА — схема управления следующим адресом, М — мультиплексор, ТГ — тактовый генератор, РС — регистр состояния, БКВВ — блок каналов ввода-вывода

ляет оперативно изменять систему команд. В процессе его обработки фаза выводит листинг трансляции, а по окончании — таблицы метаконстант, констант, шаблонов МК. В случае безошибочной обработки модуля шаблоны и константы упаковываются в таблицы определений, которые сохраняются на ВЗУ. Удобный режим работы фазы — просмотр уже созданных таблиц и их документирование.

На вход ФТ поступают модуль микропрограмм (ММ) и таблицы определений. Фаза компилирует входной модуль за два прохода. На проходе 1 проводится обработка макровызовов, директив условной трансляции, создаются символьные таблицы меток и шаблонов, собирается информация для таблиц перекрестных ссылок и др. Этот проход выполняет функции по созданию и присоединению макробиблиотек. В результате его работы создается временный файл расширения.

В начале прохода 2 в оперативную память загружаются таблицы определений, содержащие в упакованном виде шаблоны МК и константы. Программист определяет таблицы, с которыми будет работать фаза, и настраивает МА на любую систему команд, определенную в первой фазе. На проходе 2 генерируется код и выводится листинг работы ФТ. Форматированный листинг содержит зоны адреса, кода и текста исходной микропрограммы. После окончания трансляции в листинг дополнительно могут быть выведены символьные таблицы, таблицы перекрестных ссылок, карта загрузки микропрограммной памяти. Машинный код генерируется в формате отображения памяти или в формате МА AMDASM. Код выводится на ВЗУ и может быть подготовлен для прошивки в РПЗУ или использован для отладки на макете МПУ с помощью специальных технических средств.

Условия применения МАСС

Микроассемблер поставлен на микроЭВМ «Электроника 60», ДВК, ЕС1840, IBM PC в среде дисковых операционных систем и состоит из двух программ: ФО системы команд МПУ и ФТ микропрограмм. Общий объем загрузочного кода — 100 Кбайт.

При разработке микропрограмм специализированного процессора ввода-вывода (СПВВ) использовался МА МАСС. Процессор построен на базе комплекта БИС серии К1804; разрядность двунаправленной шины данных — 16, длина МК — 24, адресуемая микропрограммная память — 4096, число каналов ввода-вывода — 8 (рис. 2). Формат МК СПВВ вертикального типа объединяет три операционные формы управления: арифметико-логическим блоком и операциями ввода-вывода, непосредственной загрузкой констант и ветвлением (табл. 2).

Для кодирования микропрограммы поиска максимального и минимального элементов массива в модуле определений были описаны необходимые шаблоны МК и константы. Листинг содержит стилизованное изображение шаблонов МК сравнения, ввода-вывода, управления микропрограммой, инкрементирования регистра, возврата из подпрограммы (рис. 3).

Микропрограмма поиска содержит 19 МК (рис. 4). В листинге адрес представлен в 16-ричном основании, машинный код — двоичном, неиспользуемые биты обозначены знаком «—».

Листинг той же микропрограммы с применением алгоритмических конструкций структурного программирования демонстрирует возможности совместного использования параметрических макрорасширений и локальных блоков (рис. 5). Код в листинге представлен в 16-ричной форме, значения неиспользуемых битов равны нулю.

```
МИКРОАССЕМБЛЕР MASS B02.20 : ТАБЛИЦА ШАБЛОНОВ
СИСТЕМА КОМАНД СПВВ

1. CLEAR
0---0000----11-011100011
....VVVV.................  16: 4 [RAM_REG]

2. CMP
0---00000000111001010001
........VVVV.............  12: 4 [RAM_REG]
....VVVV.................  16: 4 [RAM_REG]

3. GOTO
11------0011000000000000
..........AAAAAAAAAAAA   0:12

4. IF
---00000----------------
...D.....................  20: 1 [INV]
....VVVV.................  16: 4 [MS_COND]

5. IN
00000000----01-011011111
.VVV.....................  20: 3 [CHAN]
....VVVV.................  16: 4 [RAM_REG]

6. INCR
0---0000----111011000011
....VVVV.................  16: 4 [RAM_REG]

7. JUMP
11------1111000000000000
............AAAAAAAAAAAA   0:12

8. LOAD
10--00000000000000000000
....VVVV.................  16: 4 [RAM_REG]
........VVVVVVVVVVVVVVVV   0:16

9. MOV
0---0000000011-011011100
........VVVV.............  12: 4 [RAM_REG]
....VVVV.................  16: 4 [RAM_REG]

10. OUT
0000----000010-001011100
........VVVV.............  12: 4 [RAM_REG]
.VVV.....................  20: 3 [CHAN]

11. RETS
11-10000101O-----------

*******  ОБРАБОТАНО КОНСТАНТ : 14
*******  ОБРАБОТАНО ШАБЛОНОВ : 11
*******  ОБНАРУЖЕНО  ОШИБОК : 0
```

Рис. 3. Листинг таблицы шаблонов микрокоманд СПВВ

Таблица 2

Формат микрокоманд СПВВ

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	К2	К1	К0	Б3	Б2	Б1	Б0	А3	А2	А1	А0	Чт	Зп	Пр	П8	П7	П6	П5	П4	П3	П2	П1	П0
	Адрес канала			Адрес Б-регистра				Адрес А-регистра				Чтение	Запись	Перенос	Выбор приемника			Функция АЛУ			Выбор источников		
1	0	—	—	Б3	Б2	Б1	Б0	Д15	Д14	Д13	Д12	Д11	Д10	Д9	Д8	Д7	Д6	Д5	Д4	Д3	Д2	Д1	Д0
		Резерв		Адрес Б-регистра				Непосредственные данные															
1	1	ЗА	Ин	У3	У2	У1	У0	В3	В2	В1	В0	А11	А10	А9	А8	А7	А6	А5	А4	А3	А2	А1	А0
		Загрузка адреса	Инверсия	Выбор условий ветвления				Управление ветвлением				Адрес ветвления											

```
МИКРОАССЕМБЛЕР MASS B02.20 : ФАЗА ТРАНСЛЯЦИИ
 ПОИСК МАКСИМУМА И МИНИМУМА МАССИВА

    # TITLE " ПОИСК МАКСИМУМА И МИНИМУМА МАССИВА"
    # DEFINITION FILE "MM3"

    # CONSTANTS
       AS = 1024 , WL = 12

    # MICROPROGRAM

    0000 0---0001---- MAXMIN: CLEAR   INDEX           ; НА   УС
         11-011100011
    0001 10--00000000         LOAD    MAX_IND,AS     ;  ЧА   ТА
         010000000000
    0002 10--00111111         LOAD    MAX, -(1<<WL) ;   ЛЬ   НОВ
         000000000000
    0003 10--01010001         LOAD    MIN, 1<<WL     ;    НАЯ   КА
         000000000000

    0004 0---00010000 MO:     CMP     MAX_IND,INDEX ; КОНТРОЛЬ
         111001010001
    0005 11-000010011         IF      ,ZERO          и\   ГРАНИЦЫ
         000000010010         GOTO    END            ;      МАССИВА

    0006 0000----0001         OUT     INDEX,ARR_ADR ;
         10-001011100
    0007 00010010----         IN      ARR_DATA,ELEM ; ВВОД ЭЛЕМЕНТА
         01-011011111
                                                     \ МАССИВА
    0008 0---00110010         CMP     ELEM,MAX       ;
         111001010001
    0009 11-100100011         IF      NOT,NEG        и
         000000001100         GOTO    M1             ;
    000A 0---00110010         MOV     ELEM,MAX       ; НОВЫЙ МАКСИМУМ
         11-011011100
    000B 0---01000001         MOV     INDEX,MAX_ADDR;
         11-011011100

    000C 0---00100101 M1:     CMP     MIN,ELEM       ;
         111001010001
    000D 11-100100011         IF      NOT,NEG        и
         000000010000         GOTO    M2             ;
    000E 0---01010010         MOV     ELEM,MIN       ; НОВЫЙ МИНИМУМ
         11-011011100
    000F 0---01100001         MOV     INDEX,MIN_ADDR;
         11-011011100

    0010 0---0001---- M2:     INCR    INDEX    ; УВЕЛИЧИТЬ УКАЗАТЕЛЬ
         111011000011
    0011 11------1111         JUMP    MO       ; ПОВТОРИТЬ ЦИКЛ
         000000000100
    0012 11-100001010 END:    RETS             ; ВЫХОД ИЗ ПОДПРОГРАММЫ
         ------------

    # END

******* ОБНАРУЖЕНО ОШИБОК : 0
```

Рис. 4. Листинг микропрограммного модуля

Заключение

Функциональные возможности, технические и эксплуатационные характеристики МА МАСС проверены на тестах и в ходе разработки высокопроизводительных МПУ. В составе комплекса средств проектирования МА МАСС демонстрировался на ВДНХ, внедрен двумя НИИ и используется в учебном процессе ряда вузов страны.

Комплект документации включает описание МА и справочник микропрограммиста.

115409, Москва, Каширское шоссе, 31, МИФИ, Кафедра 26; тел. 324-99-81, Креславский Игорь Геннадьевич

ЛИТЕРАТУРА

1. Мик Дж., Брик Дж. Проектирование микропроцессорных устройств с разрядно-модульной организацией / Под ред. А. Г. Филиппова. Пер. с англ.— М.: Мир, 1984.
2. Проектирование цифровых систем на комплектах микропрограммируемых БИС / С. С. Булгаков, В. М. Мещеряков, В. В. Новоселов, Л. А. Шумилов; Под ред. В. Г. Колесникова.— М.: Радио и связь, 1984.
3. Злотник Е. М. Секционированные микропроцессоры.— Минск: Наука и техника, 1984.
4. Микропроцессоры: системы проектирования и отладки / В. А. Мясников, М. Б. Игнатьев, А. А. Кочкин, Ю. В. Шейнин. Под ред. В. А. Мясникова, М. Б. Игнатьева.— М.: Энергоатомиздат, 1985.
5. Клингман Э. Проектирование специализированных микропроцессорных систем / Под ред. В. И. Гуревича, Н. П. Фурсикова, М. В. Шабанова. Пер. с англ.— М.: Мир, 1985.
6. Powers, V. Michael and Jose H. Hernandez. Microprogram assemblers for bit-slice microprocessors // IEEE Computer.— 1978.— V. 11.— N 7.— P. 108—120.
7. Skordalakis E. Meta-assemblers // IEEE Micro.— 1983.— V. 3.— N 2.— P. 6—16.
8. Eager, J. Michael. M29 — an advanced retargetable microcode assembler // SIGMICRO Newsletter.— 1983.— N 4.— P. 92—100.
9. Wilburn, L. Darell, Schleimer Stephen. STEP development tools: METASTEP language system // SIGMICRO Newsletter.— 1985.— N 4.— P. 157—164.
10. Семенков О. И. и др. Настраиваемый инструментальный комплекс для разработки систем на секционированных микропроцессорах // УСиМ.— 1984.— № 2.— С. 36—39.
11. Алексенко А. Г., Гапоненко А. В., Иванников А. Д., Курилов И. Д. Разработка и отладка микропрограммного обеспечения цифровых систем на основе секционированных микропроцессоров // Микропроцессорные средства и системы.— 1986.— № 1.— С. 37—43.
12. Мозговой Г. П., Семенова С. С., Семин Е. И., Трещалин О. В. Мета-ассемблер МЕАСС для микропроцессорных систем с наращиваемой разрядностью // Микропроцессорные средства и системы.— 1986.— № 3.— С. 20—22.


```
МИКРОАССЕМБЛЕР MASS B02.20 : ФАЗА ТРАНСЛЯЦИИ
ПОИСК МАКСИМУМА И МИНИМУМА МАССИВА

    # TITLE "ПОИСК МАКСИМУМА И МИНИМУМА МАССИВА"
    # DEF FILE "MM3"
    # MACRO CALL "MAXMIN"
    # VERTIC 3

    # CONSTANTS  AS=1024,WL=12

    # MICROPROGRAM

    0000 01 0C E3 MAXMIN:
    0001 80 04 00  INIT_MM          AS,WL;
    0002 83 F0 00
    0003 85 10 00
    0004 00 1E 51  WHL_NE           INDEX,MAX_IND;
    0005 C1 30 12
    0006 00 18 5C    GET_ARR        INDEX,ELEM;
    0007 12 04 DF
    0008 02 3E 51    IF_GT          ELEM,MAX;
    0009 D2 30 0C
    000A 03 2C DC      STORE_MAX    ELEM,INDEX;
    000B 04 1C DC
                     ENDIF;
    000C 05 2E 51    IF_GT          MIN,ELEM;
    000D D2 30 10
    000E 05 2C DC      STORE_MIN    ELEM,INDEX;
    000F 06 1C DC
                     ENDIF;
    0010 01 0E C3    INCR           INDEX;
    0011 C0 F0 04  ENDW;
    0012 D0 A0 00  RETS;

    # END

******* ОБНАРУЖЕНО ОШИБОК : 0
```

Рис. 5. Листинг микропрограммного модуля

Статья поступила 15.12.87

УДК 681.326

И. Г. Креславский, А. П. Жарков

ЭМУЛЯТОР МИКРОПРОГРАММНОЙ ПАМЯТИ

Эмуляция микропрограммной памяти (МП) — один из основных методов отладки микропроцессорных устройств (МПУ), выполненных на базе секционированных МПК. Эмуляторы МП широко используются в системах проектирования МПУ [1—3]. Разработанный эмулятор ориентирован на эксплуатацию в составе персональной системы проектирования МПУ, которая базируется на микроЭВМ «Электроника 60» или ДВК2М (см. рисунок). Он работает в режимах загрузки, просмотра и прогона микропрограмм и имеет следующие особенности:

конструктивную совместимость с информационным каналом МПИ и небольшие размеры (реализован на полуплате);

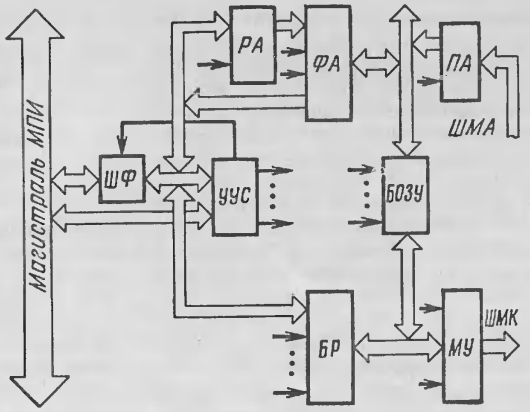
возможность ввода микропрограммного адреса в инструментальную ЭВМ, а также формирования и выдачи одиночных микрокоманд без изменения содержимого памяти микропрограмм эмулятора;

возможность отключения выходной шины микрокоманд переводом ее в высокоимпедансное состояние.

Основные технические характеристики

Число разрядов микрокоманды . . .	64
Емкость микропрограммной памяти, байт .	8192
Время выборки микрокоманды, нс, не более	50

Основной блок эмулятора — быстродействующее оперативное запоминающее устройство (БОЗУ), в котором хранится рабочая микропрограмма. В режиме прогона в эмулятор через приемники адреса (ПА) поступает адрес микропрограммы, который передается в БОЗУ, откуда выбирается соответствующая микрокоманда. Затем микрокоманда через магистральные усилители (МУ) передается в МПУ. Запись и считывание информации из БОЗУ осуществляется с помощью регистра адреса (РА) и буферного регистра (БР) микрокоманды. Формирователь адреса (ФА) позволяет в режиме останова считать микропрограммный адрес в ЭВМ. Устройство управления и связи (УУС) координирует работу всех блоков



Структурная схема эмулятора:

ШФ — шинные формирователи, ПА — приемники адреса, МУ — магистральные усилители, ШМА — шина микропрограммного адреса, ШМК — шина микрокоманды

эмулятора в различных режимах и выполняет интерфейсные функции при обмене информацией с инструментальной ЭВМ.

Эмулятор МП обслуживается с помощью системы подготовки микропрограмм ТУРБО-БИТ.

Телефон 324-99-81, Москва

ЛИТЕРАТУРА

1. Система с эмуляцией 35 нс памяти // Электроника.— 1980.— № 24.
2. Wilburn D., Mick J. STEP-27 development station // SIGMICRO Newsletter.— 1984.— N 2.— P. 22—36.
3. Микропроцессоры: системы проектирования и отладки / В. А. Мясников, М. Б. Игнатьев, А. А. Кочкин, Ю. Е. Шейнин.— М.: Энергоатомиздат, 1985.

Статья поступила 11.02.88

УДК 681.3.06

В. Н. Пинаев

ДИНАМИЧЕСКАЯ МОДИФИКАЦИЯ ПРОГРАММЫ НА БЕЙСИКЕ

При программировании на БЕЙСИКе системы РАФОС программу пользователя можно дополнять программой, считанной с внешнего устройства, заданного оператором OVERLAY. Этот оператор предлагаю использовать для настройки и модификации программы по динамически определяемым параметрам. Существует пример программы*, где динамически определяется оператор присваивания с правой частью-функцией, вводимой в диалоге пользователем. Но подобное применение оператора OVERLAY на ПЭВМ неудобно из-за необходимости использовать магнитный диск. Поэтому автором был разработан драйвер (48 слов) псевдоустройства PU для кратковременного хранения информации в ОЗУ (рис. 1).

* Трояновский В. М. Реализация диалогового определения функций в БЕЙСИКе // Программирование.— 1987.— № 1

```
        .TITLE  PU
        .IDENT  /V02.02/
        .MCALL  .DRDEF
        .DRDEF  PU,375,0,0,0,0 ; PU - ИМЯ  УСТРОЙСТВА
                               ; 375 - БАЙТ-ИДЕНТИФИКАТОР
        .DRBEG  PU             ; УСТАНОВКА В О-БЛОКЕ
                               ; ПЕРВЫХ  ПЯТИ  СЛОВ  ДРАЙВЕРА
        MOV     PUCQE,R5       ; R5-АДРЕС ЭЛЕМЕНТА СПИСКА ОЧЕРЕДИ
        TST     6(R5)          ; ЗНАК '-' ОЗНАЧАЕТ, ЧТО ПОСТУПИЛ
        BMI     WRITE          ; ЗАПРОС "ВЫВОД НА УСТРОЙСТВО"
        TST     WCN            ; WCN - СЧЕТЧИК СЛОВ. ЕСЛИ ОН <0, ТО
        BPL     PREOF          ; НЕОБХОДИМО ВЫДАТЬ "КОНЕЦ ФАЙЛА"
        NEG     WCN            ; ИТАК, "ЧТЕНИЕ С УСТРОЙСТВА"
                               ; WCN СДЕЛАЕМ ПОЛОЖИТЕЛЬНЫМ
        MOV     BUF,R2         ; КОПИРУЕМ WCN СЛОВ В БУФЕР,
        MOV     4(R5),R3       ; АДРЕС КОТОРОГО ЗАПИСАН В  R2,
        MOV     WCN,R4         ; ИЗ БУФЕРА, АДРЕС КОТОРОГО
AGAIN:  MOV     (R2)+,(R3)+    ; ХРАНИТСЯ  В  BUF
        SOB     R4,AGAIN
        BR      EXIT
PREOF:  BIS     #EOF%,@-(R5) ; "ВЫДАЧА КОНЦА ФАЙЛА"
        BR      EXIT
WRITE:  MOV     6(R5),WCN    ; "ВЫВОД НА  УСТРОЙСТВО". ФИКСИРУЕМ
        MOV     4(R5),BUF    ; РАЗМЕР  БУФЕРА  И  ЕГО  АДРЕС
        BR      EXIT
WCN:    .WORD   10
BUF:    .WORD   10
EXIT:   BR      PUEXIT
        RTS     PC
PUEXIT::
PUINT:: .DRFIN  PU
        .DREND  PU
        .END
```

Рис. 1. Текст драйвера псевдоустройства

Особенности драйвера. При выводе на псевдоустройство драйвер запоминает адрес передаваемого ему буфера. Сам буфер при этом не копируется. При чтении с псевдоустройства драйвер «перекачивает» в выходной буфер информацию из входного буфера. Подобная схема корректна, если при работе с драйвером PU не используются другие внешние устройства. Драйвер рассчитан на передачу одного блока информации. Возможна модификация драйвера с включением в него собственного буфера. Перед использованием псевдоустройства драйвер необходимо загрузить командой LOAD PU.

Пример (рис. 2) демонстрирует применение псевдоустройства для диалоговой настройки программы. После первого запуска программа модифицирует сама себя, добавляя оператор описания массива и удаляя строки с двадцатой по шестидесятую. Затем программа запускается повторно, чтобы выделить память под массив. Аналогично можно динамически формировать оператор определения функции. Подобная настройка удобна тем, что не надо «вручную» редактировать текст программы. Отметим, что если динамически формируются другие операторы (ввода-вывода, присваивания, перехода), то после модификации не нужно останавливать программу для ее повторного запуска.

```
10 PRINT "СКОЛЬКО ЭЛЕМЕНТОВ В МАССИВЕ"; \ INPUT N
20 OPEN "PU:" FOR OUTPUT AS FILE #1
30 PRINT #1,"10 DIM X(";N;")"
40 FOR I=20 TO 60 STEP 10 \ PRINT #1,I \ NEXT I
50 PRINT "А СЕЙЧАС НАБЕРИТЕ КОМАНДУ RUN"
60 CLOSE #1 \ OVERLAY "PU:" LINE 1000
70 REM ДАЛЕЕ СЛЕДУЮТ ОПЕРАТОРЫ СОБСТВЕННО ПРОГРАММЫ
1000 END
```

Рис. 2. Пример диалоговой настройки

Написанная автором на БЕЙСИКе программа проверки корректности арифметического выражения позволяет избежать синтаксических ошибок при модификации программы. Упомянутые программа и драйвер псевдоустройства входят в состав автоматизированной обучающей системы для ДВК. Эта система, в частности, может сравнивать ответ обучаемого, задаваемый им в виде некоторого арифметического выражения, с эталоном, вычисляя при проверке оба выражения в контрольных точках с использованием драйвера псевдоустройства.

Таким образом, применять псевдоустройство для динамической настройки и модификации удобно и выгодно на персональных компьютерах, так как исключаются обращения к магнитному диску.

Телефон 2-19-17, Рыбинск

Сообщение поступило 6.01.88

ФОРУМ МП

ОТ РАЗРАБОТКИ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ К КОНСТРУИРОВАНИЮ КОМПЬЮТЕРОВ*

Никлаус Вирт

Выступление при получении премии Тьюринга

Вирт ищет подходящий формализм для системного программирования. Начав с языка НЕЛИАК (через АЛГОЛ 60), он пришел к языкам Эйлер и АЛГОЛ W, к языкам Паскаль и Модула-2 и, наконец, к Лилит. При этом он добивается удивительных результатов.

Очень приятно получить премию Тьюринга; высокая оценка работы, проделанной за столько лет, одновременно радует и ободряет. Мне хотелось бы поблагодарить ACM за присуждение этой престижной премии. Особенно приятно, что я получаю ее в Сан-Франциско, где начиналась моя профессиональная деятельность.

Мое удовлетворение при известии об этой премии омрачилось, как только я осознал, что должен прочитать Тьюринговскую лекцию. Для того, кто скорее инженер, чем оратор или проповедник, эта обязанность вызывает серьезное беспокойство. На первый план среди возникающих вопросов выходит следующий: что ждут от такой лекции? Одни хотят добиться понимания методов моей работы либо ждут оценки того, каково ее место или оказываемое ею влияние. Другие хотят услышать, как возникли стоящие за этим идеи. Кто-то хочет знать мнение специалиста о тенденциях, результатах и изделиях в будущем. А некоторые надеются на доверительную оценку текущего положения вещей, которая будет либо славить потрясающие успехи нашей технологии, либо сетовать на ее ужасающие побочные эффекты и перегибы.

Прежде чем принять решение, я посоветовался с несколькими людьми, читавшими ранее Тьюринговские лекции, и обнаружил, что простое краткое сообщение об истории своей работы вполне приемлемо. Для того чтобы лекция была не просто занимательной, я попытался подвести итог тому, что, как полагаю, я узнал к настоящему времени. Честно говоря, этот выбор мне вполне подходит, так как я не претендую на бóльшее знание о будущем, чем у большинства других, а также не люблю, чтобы мне задним числом доказывали, что я был не прав. К тому же моя сильная сторона заключается вовсе не в искусстве читать проповеди о текущих достижениях и ошибках. Это не означает, что я бесстрастно наблюдаю сегодняшнее состояние компьютерного дела, в особенности беспорядочную борьбу с коммерцией.

Конечно, когда я вступил в компьютерную область в 1960 г., она не была ни в самом центре коммерческого внимания, ни в университетских программах. Во время моего обучения в Швейцарском Федеральном Технологическом институте (ETH) единственное, что я слышал о компьютерах,— факультативный курс профессора Амброса Р. Спейсера, который позднее стал президентом ИФИП. Разра-

* Перевод с англ. Б. Я. Левина.

ботанный им компьютер ERMETH был мало доступен обычным студентам, и по этой причине мое приобщение к компьютерной сфере отложилось до тех времен, когда я стал слушать курс численного анализа в университете Лаваля в Канаде. Но, увы, машина Alvac 3E бóльшую часть времени не работала, и упражнения в программировании оставались на бумаге в виде непроверяемых последовательностей шестнадцатеричных кодов.

Моя следующая попытка была несколько более успешной: в Беркли я был поставлен перед любимой машиной Гари Хаски, компьютером Bendix G-15. Хотя Bendix G-15 обеспечивал определенное чувство успеха, выдавая результаты, казалось, что суть искусства программирования состоит в разумном размещении инструкций на барабане. Если это искусство игнорировалось, то программы могли выполняться в сто раз медленнее. Но польза от обучения была очевидной: нельзя было позволить себе игнорировать мельчайшие детали. Не было возможности прикрыть недостатки вашего проекта простой покупкой большей памяти. Оглядываясь назад, можно сказать, что самой привлекательной чертой являлось то, что все детали машины были видны и могли быть поняты. Ничего еще не было спрятано внутри электронных схем, СБИС или магической операционной системы.

С другой стороны, было очевидно, что компьютеры будущего должны программироваться более эффективно. Поэтому я отказался от мысли изучить разработку аппаратуры ради изучения того, как использовать ее более элегантно. Мне повезло, что я вступил в исследовательскую группу, которая была занята разработкой — или скорее улучшением — компилятора и его использованием на IBM 704. Язык был назван НИЛИАКом, это был диалект АЛГОЛа 58. Выгоды, которые давал такой «язык», быстро стали очевидными, но задача автоматической трансляции программ в машинный код поставила бросившие нам вызов проблемы. Это как раз то, что ищешь, добиваясь степени доктора. Компилятор, написанный на языке НЕЛИАК, был сильно запутан. Казалось, что эта дисциплина состоит на один процент из науки и на 99 процентов из колдовства, и этот перекос нужно было менять. Становилось ясно, что программы должны разрабатываться в соответствии с теми же принципами, что и электронные схемы, т. е. должно быть четкое подразделение на части, через границы которых проходят только несколько проводников. Только понимание в каждый момент времени одной части дает надежду в конце понять все в целом.

Эта попытка получила сильный первоначальный толчок от появившегося сообщения об АЛГОЛе 60. АЛГОЛ 60 стал первым языком, который был ясно определен: его синтаксис задан с помощью строгого формализма. В результате мы осознали, что ясное определение является необходимым, но недостаточным условием для надежной и эффективной реализации. Контакт с Аадрианом ван Вейнгарденом, одним из соразработчиков АЛГОЛа, высветил главный предмет более отчетливо: могут ли принципы АЛГОЛа быть еще более сжаты и выкристаллизованы?

Так начались мои приключения в языках программирования. Первый эксперимент привел к диссертации и языку Эйлер — походу с садовыми ножницами через джунгли свойств и возможностей языка. В результате была достигнута академическая элегантность, от которой было мало толку на практике — почти полная противоположность более поздним языкам программирования, структурированным и имеющим типы данных. Но он создал основу для систематической разработки компиляторов, и была надежда, что ее можно будет расширить, не теряя ясности, для приспособления к будущим возможностям.

Эйлер привлек внимание Рабочей Группы ИФИП, которая занималась составлением планов на будущее АЛГОЛа. Язык АЛГОЛ 60, разработанный специалистами по вычислительной математике и для специалистов в этой области, имел систематическую структуру и четкое определение, что, хотя ему недоставало компиляторов и промышленной поддержки, было высоко оценено теми, кто имел математическую подготовку. Чтобы получить признание, он должен был расширить свою область применения. Рабочая Группа взяла на себя задачу подобрать наследника, но скоро разделилась на два лагеря. С одной стороны были те, кто намеревался установить еще одну веху на пути разработки языков, а с другой стороны — те, кто чувствовал, что время поджимает и подходящее расширение АЛГОЛа 60 было бы плодотворной попыткой. Я принадлежал к этой второй группе и внес предложение, которое при голосовании провалилось. Начиная с этого времени, мое предложение несколько раз улучшалось за счет вклада Тони Хоара (члена той же группы) и было реализовано на первой IBM 360 Стенфордского университета. Язык стал позднее известен как АЛГОЛ W и был использован в нескольких университетах для целей обучения.

Стоит упомянуть небольшой промежуточный этап в этих значительных по объему усилиях по реализации. Новая IBM 360 предоставляла только ассемблер и, конечно, Фортран. Ни тот, ни другой, ни я, ни мои аспиранты не очень хотели иметь в качестве инструмента для разработки компилятора. Поэтому я

набрался храбрости определить еще один язык, на котором планировалось написать компилятор АЛГОЛа: был достигнут компромисс между АЛГОЛом и возможностями, предоставляемыми ассемблером, это был машинный язык с алголоподобными структурой операторов и объявлениями. Стоит отметить, что язык был определен за пару недель; я написал кросскомпилятор на компьютере Burroughs B-5000 за четыре месяца, а усердный студент за такой же срок перенес его на IBM 360. Этот предварительный этап помог значительно ускорить работы по АЛГОЛу. Хотя он считался полезным лишь для наших собственных текущих потребностей, после чего его следовало отбросить, он быстро получил собственную жизнь. ПЛ360 для многих стал эффективным средством и вдохновил сходные разработки для других машин.

По иронии судьбы успех ПЛ360 был одновременно признаком неудачи АЛГОЛа W. Область применения АЛГОЛа была расширена, но как инструмент для системного программирования он все еще имел очевидные недостатки. Возникли проблемы: как удовлетворить одним языком многочисленным требованиям? Сама цель стала сомнительной. ПЛ/1, выпущенный примерно в то же время, обеспечил еще одно доказательство этого утверждения. Идея «единого ножа для швейцарской армии» имеет свои достоинства, но когда ее доводят до абсурда, этот нож становится камнем на шее. Кроме того, размер компилятора АЛГОЛа W вырос за пределы, в которых можно чувствовать себя мысленно охватившим и понимающим всю программу. Желание добиться более строгого и в то же время более подходящего для системного программирования формализма не было осуществлено. Системное программирование требует эффективного компилятора, генерирующего эффективный код, который работает без зафиксированного, постоянного, скрытого и объемного так называемого пакета периода исполнения. Этой цели не достиг ни АЛГОЛ W, ни ПЛ/1 — оба по причине сложности языка и неадекватности целевого компьютера.

Осенью 1967 г. я вернулся в Швейцарию и через год смог собрать группу из трех помощников для реализации языка, который позднее стал известен как Паскаль. Освобожденный от давления, оказываемого необходимостью все согласовывать с комитетом, я смог сконцентрироваться на том, чтобы включать те свойства, которые я сам считал существенными, и исключать те, усилия по реализации которых, по моим понятиям, были несоизмеримы с той пользой, которую они, в конце концов, приносили. Иногда преимуществом оказываются и узкие рамки, в которые ставят суровые ограничения людских ресурсов.

Порою декларируется, что Паскаль был разработан как язык для обучения. Хотя это и правильно, его использование в обучении не было единственной целью. В действительности я не верю в использование в обучении таких средств и формализмов, которые не годятся для какой-нибудь практической задачи. По сегодняшним стандартам Паскаль имеет очевидные недостатки для программирования больших систем, но 15 лет назад он представлял собой разумный компромисс между тем, что было желанно, и тем, что было эффективно. В ETH мы ввели Паскаль в классы для занятия программированием в 1972 г., несмотря на существование значительной оппозиции. Это имело успех, так как дало возможность преподавателю концентрировать внимание скорее на структурах и общих представлениях, чем на особенностях и странностях, т. е. скорее на принципах, чем на технике.

Наш первый компилятор Паскаля был реализован для семейства компьютеров CDC6000. Он был написан на самом Паскале. Никакого ПЛ6000 не потребовалось, и я рассматривал это как существенный шаг вперед. Тем не менее, генерируемый код определенно уступал коду, генерируемому компиляторами Фортрана для соответствующих программ. Скорость является существенным и легко измеряемым критерием, и мы полагали, что обоснованность концепции языка высокого уровня будет принята промышленностью только в том случае, если потери производительности исчезнут или, по крайней мере, уменьшатся. Имея это в виду, была предпринята вторая попытка, по существу одним человеком, чтобы получить высококачественный компилятор. Цель была достигнута Урсом Амманом в 1974 г., и этот компилятор был затем широко распространен во многих университетах и отраслях промышленности. Однако цена была высока: усилия по генерации хорошего (т. е. даже не оптимального) кода пропорциональны несоответствию между языком и машиной, а CDC6000 разрабатывалась, конечно, не с прицелом на языки высокого уровня.

Снова, по иронии судьбы, основная польза была получена там, где ее меньше всего ожидали. После того, как о существовании Паскаля стало известно, несколько человек обратились к нам за помощью в реализации Паскаля на различных других машинах, делая упор на то, что они предполагают использовать его в целях обучения и скорость для них не самое главное. По этой причине мы решили создать версию компилятора, которая бы генерировала код для машины, разработанной нами самими. Этот код позднее стал известен как

П-код. Версию для П-кода было легко сконструировать, так как новый компилятор был разработан по существу как упражнение в структурном программировании путем пошагового уточнения, и по этой причине первые несколько шагов уточнения можно было принять без изменений. Оказалось, что Паскаль-П с исключительным успехом способствовал распространению языка среди множества пользователей. Если бы нам хватило ума предвидеть широту этого движения, мы бы приложили больше усилий и внимания для разработки и документирования П-кода. А так наша попытка удовлетворить все просьбы одним махом осталась побочным делом. Это показывает, что, даже имея благие намерения, можно выбрать неверные цели.

Однако Паскаль добился по настоящему широкого признания только после того, как Кен Боулз из Сан-Диего осознал, что П-система может быть хорошо реализована на новых микрокомпьютерах. Его усилия по разработке подходящего окружения, включающего в себя интегрированные компилятор, файловую систему, редактор текста и отладчик, совершили прорыв: Паскаль стал доступен тысячам новых пользователей компьютеров, которые не были обременены устоявшимися привычками или задушены стремлением сохранить совместимость с прошлыми программами.

Тем временем я завершил работу над Паскалем и решил исследовать новую привлекательную область мультипрограммирования, в которой Хоар заложил существенные основы, а Бринч Хансен проложил путь своим Параллельным Паскалем. Попытка выделить конкретные правила для дисциплины мультипрограммирования быстро привела меня к формулированию их в терминах небольшого набора программистских возможностей. Для того чтобы подвергнуть правила настоящему испытанию, я встроил их в фрагментарный язык, название которого было создано согласно моей главной цели — модульности в программных системах. Модуль оказался позднее главным достоинством этого языка: он придал абстрактному понятию «упрятывания информации» конкретную форму и ввел существенный порядок как в мультипрограммировании, так и вне его. Модула также содержала средства для выражения параллельных процессов и их синхронизации.

К 1976 г. мне несколько надоели языки программирования и тщетные попытки создать хорошие компиляторы для существующих компьютеров, разработанных для старомодного «ручного» кодирования. К счастью, я получил возможность провести свой свободный от лекций год (предоставляется раз в семь лет — *прим. переводчика*) в исследовательской лаборатории корпорации Ксерокс в Пало Альто, где идея мощной персональной рабочей станции не только впервые появилась, но и была практически реализована. Вместо того, чтобы делить со многими другими единый большой компьютер и бороться за свою долю через провод с полосой частот в 3 кГц, я теперь пользовался собственным компьютером, расположенным под моим столом, с каналом более чем 15 МГц. Влияние увеличения в 5000 раз невозможно себе представить ни в одной области: оно поразительно. Более всего меня воодушевлял тот факт, что после 16 лет моей работы на компьютеры теперь, казалось, что компьютер работает на меня. Впервые я вел ежедневную корреспонденцию и писал отчеты с помощью компьютера вместо того, чтобы проектировать новые языки, компиляторы и программы для использования их другими. Еще одним откровением было то, что компилятор для языка Меса, сложность которого намного превосходила сложность Паскаля, мог быть реализован на такой рабочей станции. Эти новые условия работы на столько порядков превосходили то, с чем я был знаком дома, что я решил попытаться там установить такое же окружение.

Наконец-то я решил окунуться в разработку аппаратуры. Это решение было подкреплено моим старым отвращением к существующим архитектурам компьютеров, которые делали жалкой жизнь разработчика компиляторов, если у него была склонность к систематичности и простоте. Мысль разработать и построить целиком компьютерную систему, состоящую из аппаратуры, микрокода, компилятора, операционной системы и программных утилит, быстро оформилась в моем воображении — разработка, которая будет свободна от любых ограничений, требующих совместимости с PDP-11 или IBM 360, Фортраном, Паскалем, Юниксом или каким-нибудь еще другим объявившимся сегодня увлечением или стандартом, исходящим от какого-нибудь комитета.

Но чувства свободы недостаточно для того, чтобы добиться успеха в техническом проекте. Нельзя обойтись без тяжелого труда, решимости, тонкого чувства того, что является существенным, а что эфемерным, и еще надо немного удачи. Первой удачей стал телефонный звонок разработчика аппаратуры, который интересовался возможностью попасть в наш университет, чтобы изучить методы программного обеспечения и получить степень доктора философии. Почему бы нам не поучить его программному обеспечению, а ему не поучить нас аппаратному обеспечению?

Прошло немного времени и мы вдвоем стали действующей бригадой, а Ричард Оран вскоре был столь захвачен новой разработкой, что

почти полностью забыл как о программном обеспечении, так и о своей докторской степени. Это меня не слишком беспокоило, так как я был полностью занят разработкой частей аппаратуры; спецификацией микро- и макрокодов и программированием интерпретатора макрокодов; планированием программного обеспечения в целом и особенно программированием текстового редактора и редактора диаграмм, использующих новый растровый дисплей с высоким разрешением и маленькое чудо по имени Мышь в качестве устройства управления курсором на экране. Это упражнение в программировании существенно диалоговых прикладных программ потребовало изучения и применения методов, совершенно чуждых обычным разработкам компиляторов и операционных систем.

Проект в целом был настолько разнообразным и сложным, что казалось безответственным начинать его, особенно имея в виду малое число имеющихся у нас ассистентов, занятых неполный рабочий день (в среднем их было примерно семь человек). Главной угрозой было то, что могло потребоваться слишком много времени, в течение которого энтузиазм нас двоих будет поддерживаться исключительно нашим упорством, чтобы дать возможность другим, еще не осознавшим всю прелесть идеи рабочей станции, стать такими же энтузиастами. Чтобы удержать проект в разумных рамках, я придерживался трех догм: целью был компьютер **с одним процессором**, на котором работает **один пользователь** и который запрограммирован **на одном языке**. Заслуживает внимания тот факт, что эти краеугольные камни были диаметрально противоположны духу времени, который благоволил к исследованиям многопроцессорных конфигураций, многопользовательских операционных систем с разделением времени и к стольким языкам, сколько можно собрать.

Ограничив себя одним языком, я очутился перед трудным выбором, последствия которого будут сказываться в широкой области — речь идет о выборе языка. Из существующих языков ни один меня не привлекал. Они не могли ни удовлетворить всем требованиям, ни понравиться разработчику компилятора, знающему, что его задача должна быть выполнена в разумный промежуток времени. В частности, язык должен был быть приспособлен ко всем нашим пожеланиям относительно возможностей структурирования, основанным на десятилетнем опыте работы с Паскалем, и он должен был обслуживать ситуации, которые до сих пор обрабатывались только с помощью кодирования на ассемблере. Короче говоря, выбор состоял в том, чтобы разработать потомка обоих языков: как хорошо зарекомендовавшего себя Паскаля, так и экспериментальной Модулы. Он был назван Модула-2. Модуль — это ключевое средство для подведения под одну крышу противоречивых требований: высокого уровня абстракции для обеспечения безопасности, которая достигается избыточными проверками, и средств низкого уровня, которые позволяют обеспечить доступ к индивидуальным свойствам конкретного компьютера. Он позволяет программисту скрывать в нескольких небольших частях системы использование средств низкого уровня, тем самым защищая его от попадания в их ловушки в самых неожиданных местах.

Проект Лилит доказал, что разработка одноязыковой системы не только возможна, но и выгодна. Все, от драйверов до текстовых и графических редакторов, написано на одном и том же языке. Фактически нет различия между модулями, принадлежащими операционной системе, и модулями, принадлежащими пользовательским программам.

Это различие почти исчезает, а вместе с ним исчезает и тяжесть монолитного, громоздкого резидентного блока кода, с которым все вынуждены мириться, хотя он всем мешает. Более того, проект Лилит доказал выгодность хорошо соответствующих друг другу разработок аппаратуры и программного обеспечения. Эта выгода может быть измерена в терминах скорости. Сравнения времени выполнения программ на Модуле показали, что Лилит часто превосходит VAX 750, чья сложность и стоимость в несколько раз превосходят сложность и стоимость Лилита. Их также можно измерять в терминах пространства: код программ на Модуле для Лилита короче кода для PDP-11, VAX или M68000 в 2—3 раза и короче кода для NS 32000 в 1,5—2 раза. Кроме того, кодогенерирующие части компиляторов для этих микропроцессоров значительно сложнее, чем для Лилита, из-за малоподходящего набора их команд. Это увеличение длины должно быть умножено на низкую плотность кода, что бросает мрачную тень на сильно рекламируемое соответствие современных микропроцессоров языкам высокого уровня и показывает, что их претензии сильно преувеличены.

Перспектива будущего повторения этих разработок в миллионах экземпляров приводит в довольно большое уныние, так как просто благодаря своему количеству они становятся нашими стандартными строительными блоками. К сожалению, прогресс в технологии полупроводников столь стремителен, что он затмил прогресс в архитектуре, и они, по-видимому, стали меньше соответствовать друг другу. Конкуренция заставляет изготовителей переносить на кремний новые разработки задолго до того,

как они доказали свою эффективность. И в то время, как объемистое программное обеспечение можно по крайней мере изменить, а в лучшем случае и заменить, сложность уже дошла теперь до самих микросхем. И мало надежды, что мы лучше справимся со сложностью, когда она касается скорее аппаратного, чем программного обеспечения.

Многих по обе стороны этого забора сложность очень прельщает и будет прельщать. Мы на самом деле живем в сложном мире и стараемся решать присущие ему сложные задачи, которые действительно требуют сложных механизмов. Однако это не должно ослаблять нашего желания получить элегантные решения, которые убеждают своей ясностью и эффективностью. Простые элегантные решения более эффективны, но их труднее найти, чем сложные, и они требуют больше времени, что мы слишком часто считаем непозволительным.

Раньше чем закончить, разрешите мне попытаться выделить общие характеристики упомянутых мною проектов. Очень важным техническим приемом, который редко используется где-нибудь столь эффективно, как в информатике, является раскрутка. Мы использовали ее фактически во всех проектах. При разработке инструментальных средств, будь то язык программирования, компилятор или компьютер, я разрабатывал его таким образом, что он оказывался полезным уже на следующем шаге: ПЛ360 был разработан для реализации АЛГОЛа W; Паскаль — для реализации Паскаля П; Модула-2 — для реализации всего программного обеспечения рабочей станции, Лилит — для обеспечения подходящего окружения всей нашей будущей работы — от программирования до разработки и документирования электронных схем, от подготовки докладов до разработки шрифтов. Раскрутка — это самый эффективный способ извлечь выгоду благодаря собственным усилиям, так же как пострадать от собственных ошибок.

Это обязательно заставляет рано проводить разделительную черту между тем, что существенно, и тем, что преходяще. Я всегда пытался выявить то, что является существенным, дает неоспоримые преимущества, и сосредоточить на этом свое внимание. Например, я считаю существенным включение в язык программирования последовательной и согласованной системы объявлений типов данных, в то время как особенности разных видов оператора FOR либо способность компилятора различать большие и малые буквы — это вопросы малосущественные. В разработке компьютеров я считаю решающим выбор видов адресации и обеспечение полноты и согласованности наборов (со знаком и без знака) арифметических команд, включающих надлежащие прерывания при переполнении; в противоположность этому, особенности механизма многоканальных прерываний, их приоритеты гораздо менее существенны. Еще важнее обеспечить, чтобы несущественное никогда не сталкивалось с систематической, структурированной разработкой основных свойств. Несущественное лучше добавлять, приспосабливая к уже существующей, хорошо структурированной основе.

Иногда трудно противостоять давлению, оказываемому для включения различных возможностей, которые «тоже было бы неплохо иметь в наличии». Вполне реальна опасность, что желание угодить всем будет мешать задаче получить целостный проект. Я всегда старался выяснить, какую цену придется заплатить за получаемый выигрыш. Например, когда рассматривается включение некоторого свойства языка или добавление особого режима компиляции для достаточно часто встречающейся конструкции, нужно сопоставить получаемую выгоду и дополнительную стоимость его реализации и просто его присутствия, которое ведет к увеличению системы. Разработчики языка при этом часто терпят неудачу. Я охотно допускаю, что некоторые свойства Ады, для которых нет соответствия в Модуле-2, иногда хорошо иметь, но в то же время я сомневаюсь, что они оправдывают свою цену. А цена значительная: во-первых, хотя разработка обоих языков началась в 1977 г., компиляторы Ады только теперь начали появляться, в то время как мы используем Модулу с 1979 г. Во-вторых, говорят, что компиляторы Ады — это гигантские программы, состоящие из сотен тысяч строк кода, в то время как наш последний компилятор Модулы имеет размер всего порядка пяти тысяч строк. Я по секрету сознаюсь, что этот компилятор с Модулы уже на пределе допустимой сложности, и я бы чувствовал себя совершенно неспособным создать хороший компилятор для Ады. Даже если можно было бы игнорировать усилия, направленные на построение ненужно больших систем, и стоимость памяти, содержащей их код, все равно реальная стоимость прячется в невидимых усилиях бесчисленных программистов, отчаянно пытающихся понять и использовать эти системы эффективно.

Другой общей характеристикой обрисованных проектов явился выбор инструментов. Я убежден, что инструмент должен соответствовать результатам; он должен быть таким простым, как только возможно, но не проще. Когда значительную часть всего проекта берутся одолеть этим инструментом, то он в действительности становится непродуктивным. В про-

ектах Эйлера, АЛГОЛа W, ПЛ360 уделялось много внимания развитию методов таблично управляемого восходящего анализа. Позднее я переключился опять на простой нисходящий метод рекурсивного спуска, легко понятный и, без сомнения, достаточно мощный, если синтаксис языка выбран разумно. При разработке аппаратуры Лилит мы ограничили себя хорошим осциллографом, только изредка требовался логический анализатор. Это было возможно благодаря относительно систематическому, без трюков, общему представлению о процессоре.

Каждый отдельный проект был прежде всего познавательным экспериментом. Лучше всего познаешь, когда изобретаешь. Только когда я фактически участвую в разработке проекта, я могу добиться достаточного знания о существенных трудностях и приобрести достаточную уверенность в том, что удастся одолеть все присущие ему детали. Я никогда не мог отделить разработку языка от его реализации, потому что жесткое определение языка без обратной связи от конструкции его компилятора казалось мне самонадеянным и непрофессиональным. Таким образом, я участвовал в конструировании компиляторов, электронной схемы и редакторов текста и графики, а это повлекло за собой микропрограммирование, много программирования высокого уровня, разводку схемы, компоновку плат и даже прокладку электрических соединений. Это может показаться странным, но я гораздо больше люблю своими руками проводить опыты, чем руководить группой. Я также узнал, что исследователи гораздо охотнее признают над собой руководство фактического члена их группы, с которым близки, чем эксперта-специалиста по организации работ, будь то управляющий в промышленности или университетский профессор. Я стараюсь помнить, что обучение путем подачи хорошего примера часто — самый эффективный, а иногда — и единственно возможный метод.

И последнее. Каждый из этих проектов был доведен до конца благодаря энтузиазму и желанию преуспеть, основанных на знании, что эти усилия имеют смысл. Это, возможно, самая существенная, но в то же время самая неуловимая и трудно проверяемая предпосылка успеха. Мне повезло иметь членов бригады, которые заражались энтузиазмом, и я пользуюсь этим удобным случаем, чтобы поблагодарить их всех за ценный вклад в общее дело. Моя искренняя благодарность относится ко всем, кто принимал участие, будь то непосредственная работа в группе или косвенное участие: испытание наших результатов и обеспечение обратной связи, содействие появлению идей через критику или одобрение или создание общества пользователей. Без них ни АЛГОЛ W, ни Паскаль, ни Модула-2, ни Лилит не стали бы тем, чем они стали. Этой премии Тьюринга удостоен и вклад моих соратников.

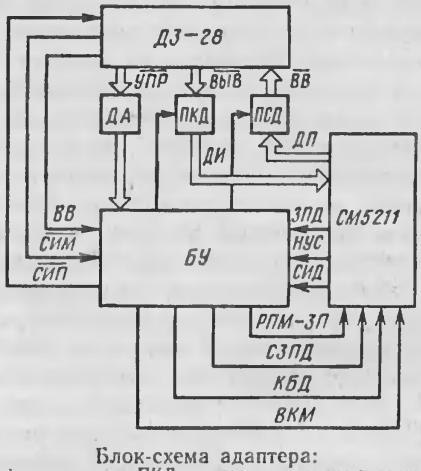
КРАТКОЕ СООБЩЕНИЕ

УДК 681.327.8+621.317

Ю. Д. Афонин, А. Р. Бекетов, М. В. Мещерягин

АДАПТЕР ДЛЯ ПОДКЛЮЧЕНИЯ КАССЕТНОГО ВЗУ СМ5211 К МИКРОЭВМ «ЭЛЕКТРОНИКА Д3-28»

Разработанный адаптер позволяет подключить внешнее запоминающее устройство (ВЗУ) СМ5211 к микроЭВМ Д3-28. При этом реализуются все возможности ВЗУ. Адаптер выполнен на одной плате, которая устанавливается на место штатного блока сопряжения СМ5211/721010 (см. рисунок).



Блок-схема адаптера:
ДА — дешифратор адреса, ПКД — приемник команд и данных, ПСД — передатчик байта состояния и данных, БУ — блок управления

Адрес устройства по шине УПР через дешифратор поступает в блок управления и инициализирует его. В зависимости от кода на шине УПР могут выполняться следующие функции: передача команд в накопитель, запись данных, передача байта состояния накопителя в Д3-28 и чтение данных. Команды или данные из Д3-28 по шине ВЫВ через приемник поступают в накопитель и по сигналу РПМ-3П запоминаются. Передачу команд сопровождает низкий уровень сигнала ВКМ. Байт состояния или данные при чтении из накопителя через передатчик поступают на шину ВВ Д3-28. Передача байта состояния осуществляется при высоком уровне сигнала СИД, взаимодействие ЭВМ и накопителя — по сигналам ВВ, СИП, ЗПД. Сигнал СИП вырабатывается только после прихода сигнала ЗПД от накопителя при низком уровне сигнала ВВ. Таким образом, скорость чтения или записи устанавливается самим накопителем и составляет примерно 1,2 Кбайт/с. Сигнал КБД (конец блока данных) устанавливается после окончания записи блока данных.

Использование накопителя с адаптером в составе комплекса ВУМС-001-10 в операционной среде ОС ВТ-11С-МХТИ позволило ввести стандартизованную запись, уменьшить число остающихся битовых ошибок до 10^{-9}, ускорить поиск нужного файла, примерно в три раза увеличить скорость записи и чтения.

620002, Свердловск, УПИ им. С. М. Кирова, кафедра редких металлов; тел. 44-87-08

Сообщение поступило 15.03.88

УДК 621.373—529

М. О. Погосян, Г. С. Агинян, В. Г. Арутюнян

ВАРИАНТЫ ПОСТРОЕНИЯ ГЕНЕРАТОРОВ ПРЯМОУГОЛЬНЫХ ИМПУЛЬСОВ С ПРОГРАММИРУЕМОЙ ДЛИТЕЛЬНОСТЬЮ

Генератор с программируемым периодом $T_{прог}$ в диапазоне 2,0...65536 мкс и программируемой длительностью $\tau_{прог}$ в диапазоне 1,0 мкс...65535 мкс.

Дискретность установки $T_{прог}$ и $\tau_{прог}$ равна 100 нс. Максимальная скважность $Q=T_{прог}/\tau_{прог}$ равна 65536. Точность установки $T_{прог}$ и $\tau_{прог}$ в основном определяется точностью и стабильностью кварцевого генератора.

В схеме использованы два программируемых таймера (ПТ) КР580ВИ53. Поскольку максимальная тактовая частота БИС КР580ВИ53—2 МГц (минимальная дискретность установки периода равна 0,5 мкс), использование таймера без внешнего обрамления не позволяло установить $T_{прог}$ и $\tau_{прог}$ дискретностью 100 нс. Для решения этой задачи избран следующий принцип построения генератора [1].

С помощью двух программируемых двоичных 4-разрядных счетчиков эталонная частота 10 МГц делится соответственно на 10 или 11 (в зависимости от того, содержит ли $T_{прог}$ и $\tau_{прог}$ дробную часть). При этом на выходе переноса счетчиков формируются прямоугольные импульсы с периодом 1,0 мкс или 1,1 мкс.

Пусть $\tau_{цел}$ — целая часть $\tau_{прог}$ а $\tau_{дроб}$ — дробная часть $\tau_{прог}$ (соответственно $T_{цел}$ — это целая часть $T_{прог}$, а $T_{дроб}$ — дробная часть $T_{прог}$). Требуемые $\tau_{прог}$ и $T_{прог}$ получаются соответствующим подсчетом импульсов $\tau_{дроб}$ и $T_{дроб}$, кратных 1,1 мкс, и $\tau_{цел}$ и $T_{цел}$, кратных 1,0 мкс, соответственно на входе CLK0 (τ) и CLK1 (T) таймера 1 (D1).

В общем виде запишем $\tau_{прог}=1{,}1\times\tau'_{дроб}+1{,}0\times(\tau_{цел}-\tau'_{дроб})$, $T_{прог}=1{,}1\times T'_{дроб}+1{,}0\times(T_{цел}-T'_{дроб})$, а после упрощения получим $\tau_{прог}=0{,}1\times\tau'_{дроб}+1{,}0+\tau_{цел}$, $T_{прог}=0{,}1\times T'_{дроб}+1{,}0\times T_{цел}$.

Очевидно, что $\tau'_{дроб}$ и $T'_{дроб}$ изменяются в пределах 1...9.

Счетчик СЧ0 ПТ (таймер 1 загружается числом $\tau_{цел}$, а счетчик СЧ1 того же ПТ — числом $T_{цел}$. Счетчик СЧ2 не используется. Аналогично счетчик СЧ0 ПТ и таймер 2 загружаются числом $\tau'_{дроб}$, а счетчик СЧ1 — числом $T'_{дроб}$ (рис. 1).

Для управления работой генератора использованы режимы 1, 2 и 5 работы таймера КР580ВИ53 [2, 4].

Режим 1 — режим программируемого одновибратора. Таким образом, использован тот факт, что уровень выхода OUT ПТ становится низким (рис. 2) одновременно с перепадом на входе GATE таймера КР580ВИ53.

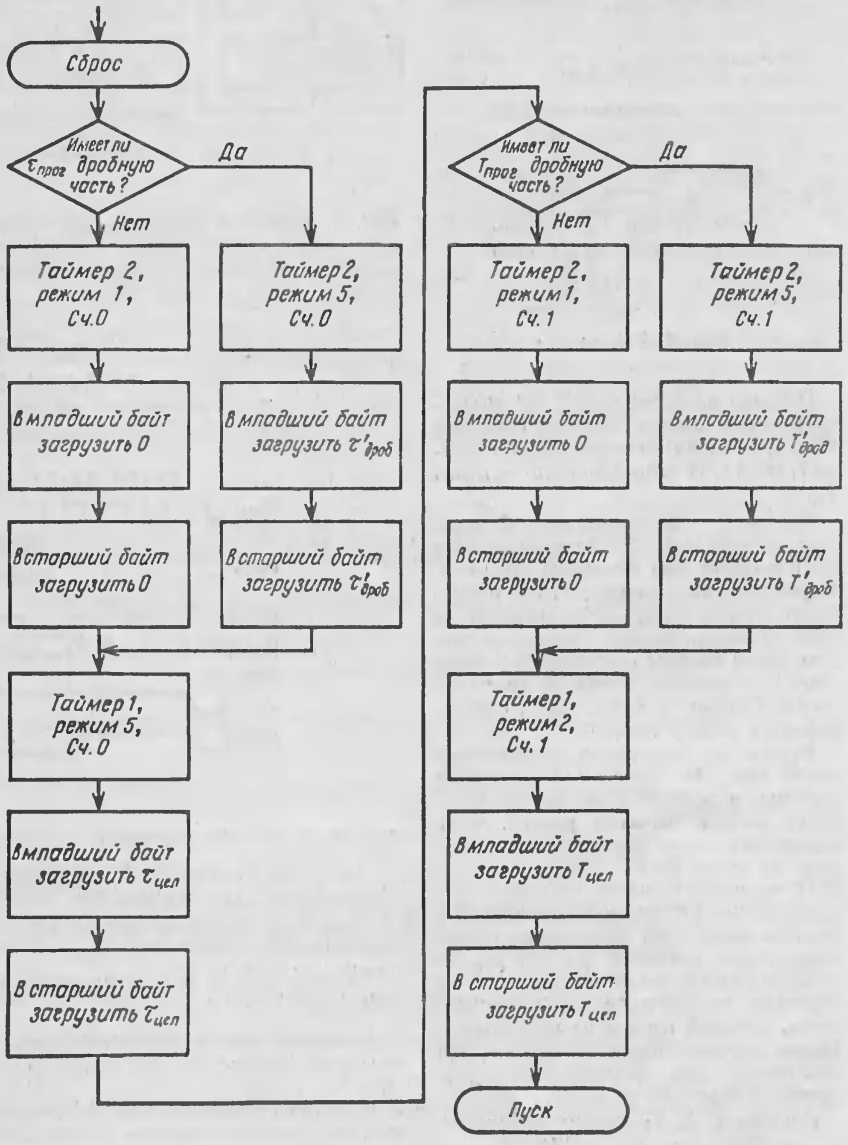


Рис. 1. Алгоритм загрузки управляющих слов и информации в таймер 1 и таймер 2 для получения необходимых $\tau_{прог}$ и $T_{прог}$

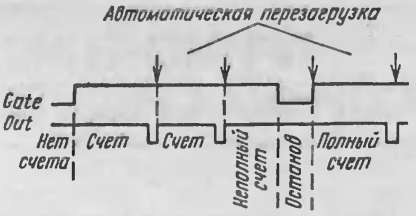


Рис. 2. Режим «1»

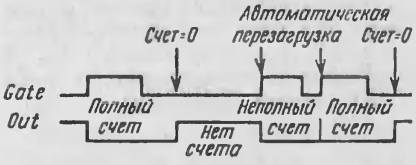


Рис. 3. Режим «2»

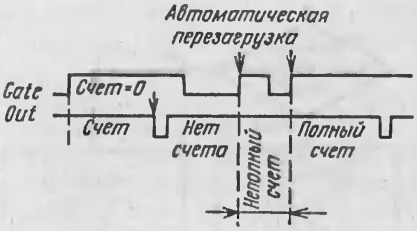


Рис. 4. Режим «5»

Перепад на входе GATE ПТ обеспечивается в момент запуска сигналом ПУСК, а также выходом OUT1 таймера 1, где в СЧ1 записано число — целое $T_{прог}$.

Режим 2 — программируемый делитель частоты (рис. 3). Уровень выхода OUT высокий при установке режима 2. Перепад сигнала на входе GATE инициирует счет с начального значения до нуля. Уровень выхода становится низким, когда счетчик достигнет состояния Лог. 1, и высоким, когда он достигнет Лог 0. Сигнал на входе GATE синхронизирует работу таймера.

Режим 5 — аппаратно запускаемый строб (рис. 4). После инициализации счетчика в режиме 5 на выходе OUT будет сигнал высокого уровня. Счет начинается после формирования перепада на входе GATE. Уровень выхода OUT становится низким на время синхросигнала CLK при достижении счета, равного нулю. Счет запускается с инициирующего значения каждый раз по перепаду на входе GATE. Если регистр счетчика перезагружается в течение счета, текущий период не нарушается. Новое значение счета становится эффективным при формировании следующего перепада на входе.

Пример. 1. Требуется установить $T_{прог}=32{,}6$ мкс; $\tau_{прог}=0{,}1\times 3\,\tau'_{дроб}+1{,}0\times 16\,\tau_{цел}$.

Загрузка управляющих слов и информации в соответствии с диаграммой рис. 1 показана на рис. 5.

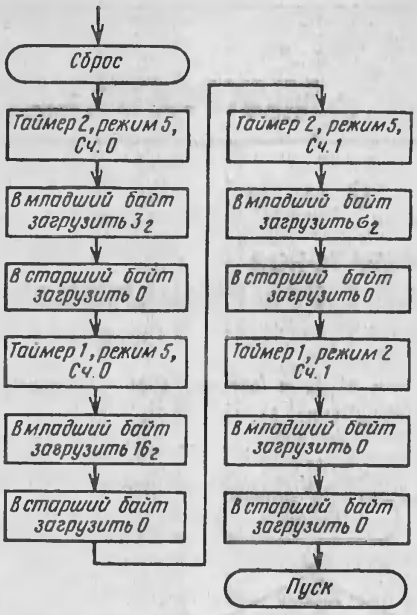


Рис. 5. Алгоритм загрузки управляющих слов и информации в таймер 1 и таймер 2 для получения $T_{прог}=32{,}6$ мкс и $\tau_{прог}=16{,}3$ мкс

Рис. 6. Алгоритм загрузки управляющих слов и информации в таймер 1 и таймер 2 для получения $T_{прог}=10$ мкс и $\tau_{прог}=4$ мкс

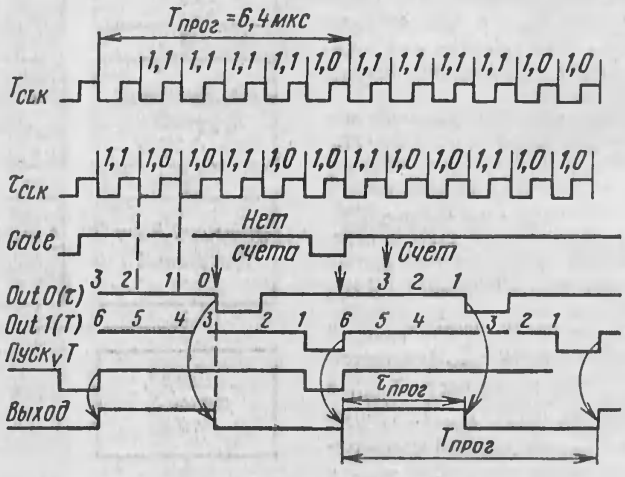


Рис. 7. Временная диаграмма получения $T_{прог}=6{,}4$ мкс и $\tau_{прог}=3{,}1$ мкс

Пример 2. Требуется установить $T_{прог}=10$ мкс; $\tau_{прог}=4$ мкс, т. е. целые. Исходя из приведенных в начале статьи соотношений запишем $T_{прог}=0{,}1\times 0\,T'_{дроб}+1{,}0\times 10\,T_{цел}$, $\tau_{прог}=0{,}1\times 0\,\tau'_{дроб}+1{,}0\times 4\,\tau_{цел}$.

В соответствии с диаграммой рис. 1 загрузка пройдет так, как показано на рис. 6.

Исходя из вышеизложенного принципа невозможно получить следующий спектр частот для $\tau_{прог}$ и $T_{прог}$: 1,2... 1,9 мкс; 2,3...2,9 мкс; 3,4...3,9 мкс; 4,5... 4,9 мкс; 5,6...5,9 мкс; 6,7...6,9 мкс; 7,8... 7,9 мкс; 8,9 мкс, поскольку здесь целые числа $\tau'_{дроб}$ и $T'_{дроб}$ везде больше целых чисел, соответственно, $\tau_{цел}$ и $T_{цел}$.

На временной диаграмме (рис. 7) в качестве примера показано получение $\tau_{прог}=3{,}1$ мкс; $T_{прог}=6{,}4$ мкс. Необходимые импульсы получаются на инверсном выходе D-триггера.

Данный генератор программировался с помощью микроЭВМ «Электроника 60». Они обменивались информацией через разработанный модуль интерфейса КСМ (канал связи модульный).

Когда используется только модуль генератора, можно применять стандартное устройство параллельного обмена 15КС-180—032.

Генератор прямоугольных импульсов с программируемым периодом $T_{прог}$ в

диапазоне 2 мкс...131 мс и программируемой длительностью $\tau_{прог}$ в диапазоне 100 нс...1,5 мкс.

Дискретность установки $T_{прог}$ и $\tau_{прог}$ равна 100 нс.

Для построения генератора с программируемыми $T_{прог}$ и $\tau_{прог}$ с учетом включения периода $T'_{дроб} > T_{цел}$ и $\tau'_{дроб} > \tau_{цел}$ использован отличный от предыдущего принцип [3].

Пусть $T_{цел}$ — целая часть $T_{прог}$, а $T'_{дроб}$ — дробная часть $T_{прог}$. Значение $T_{прог}$ устанавливается по закону $T_{прог} = 100(T'_{дроб} + 10\,T_{цел})$ нс, где $T'_{дроб}$ меняется в пределах 1...9, а $T_{цел}$ в диапазоне 2...131072 мкс (с учетом поставленной задачи).

Пример — установка произвольного $T_{прог}$ (допустим 2,6 мкс): $T_{прог} = 100(6 + 10 \cdot 2) = 2600$ нс $= 2{,}6$ мкс. Таким образом, необходимое $T_{прог}$ получается «сшиванием» соответствующей дробной части $T_{дроб}$ и целой части $T_{цел}$.

Генератор состоит из трех функциональных узлов. В первый входят: элементы, обеспечивающие получение времен для $T_{дроб}$ в диапазоне 100...900 нс, во второй — элементы, обеспечивающие получение времен для $T_{цел}$ в диапазоне 2...131072 мкс, третий узел включает в себя элементы, позволяющие получать времена для $\tau_{прог}$ в диапазоне 100 нс... 1,5 мкс.

Требуемое $T_{прог}$ получается суммированием времени $T_{дроб}$, получаемого на выходе счетчика D4, с временем $T_{цел}$, получаемым на выходе OUT0 ПТ КР580ВИ53 и счетчика D6; $\tau_{прог}$ получается на выходе счетчика D5.

На рис. 8 показана временна́я диаграмма получения $T_{прог} = 2{,}6$ мкс и $\tau_{прог} = 500$ нс, а на рис. 9 — принцип получения необходимых импульсов с заданным $T_{прог}$ и $\tau_{прог}$ на инверсном выходе триггера D9.2.

В момент поступления сигнала СБРОС элементы схемы устанавливаются в первоначальные состояния (рис. 10). В результате этого вентиль, на вход которого подаются эталонные импульсы с частотой 10 МГц, запирается.

После предварительной записи информации в триггер D1, D2, D3.1 и таймер D15 с момента поступления сигнала ПУСК одновременно (рис. 10) запускаются счетчики D4 и D5. Триггер D2 служит для предварительной записи $\tau_{прог}$. Счетчик D5 работает в режиме вычитания. После достижения счетчиком нуля инверсный выход триггера D8.2 устанавливается в Лог. 0. Это запрещает подачу счетных импульсов на вход «—1» элемента D5, т. е. отсчет $\tau_{прог}$ закончен. Этим же самым импульсом с D5 соответствующий код $\tau_{прог}$ с выходов триггера D2 перезаписывается на вход счетчика D5. В триггер D1 предварительно записывается $T_{дроб}$. Одновременно с началом отсчета $\tau_{прог}$ за-

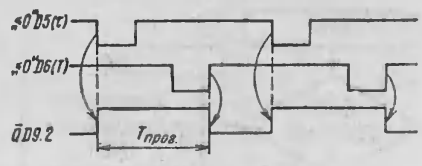


Рис. 9. Временные диаграммы получения импульсов на инверсном выходе триггера D9.2

пускается счетчик D4. Поскольку счетчик D4 работает в режиме вычитания, в момент достижения нуля на выходе «≤0» генерируется импульс, который устанавливает прямой выход триггера D8.1 в состояние Лог 1. Этот уровень Лог. 1 разрешает прохождение эталонных импульсов с частотой 10 МГц через вентиль на вход счетчика D11. Одновременно этим же самым импульсом с D4 $T'_{дроб}$ перезаписывается в счетчик D4. Другой инверсный выход триггера D8.1, находящийся в состоянии Лог. 0, запрещает прохождение счетных импульсов через вентиль на вход «—1» счетчика D4. Таким образом, получение дробной части $T'_{дроб}$ закончено.

С момента запуска генератора прямой выход триггера D9.1 находится в состоянии Лог. 1, т. е. на входе GATE0 таймера D15 Лог. 1.

В соответствии с режимом 2 таймера с момента формирования среза импульса с D13 начинается счет.

При достижении счетчиком счета, равного Лог. 1, генерируются прямо-

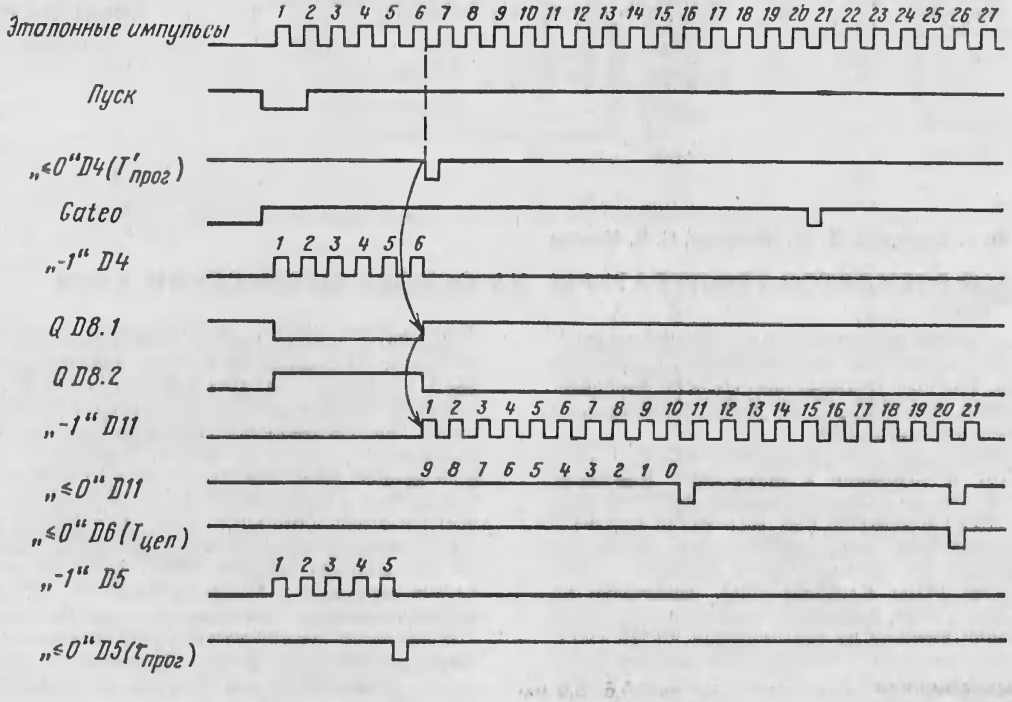


Рис. 8. Временна́я диаграмма: $T_{прог} = 2{,}6$ мкс, $\tau_{прог} = 500$ нс

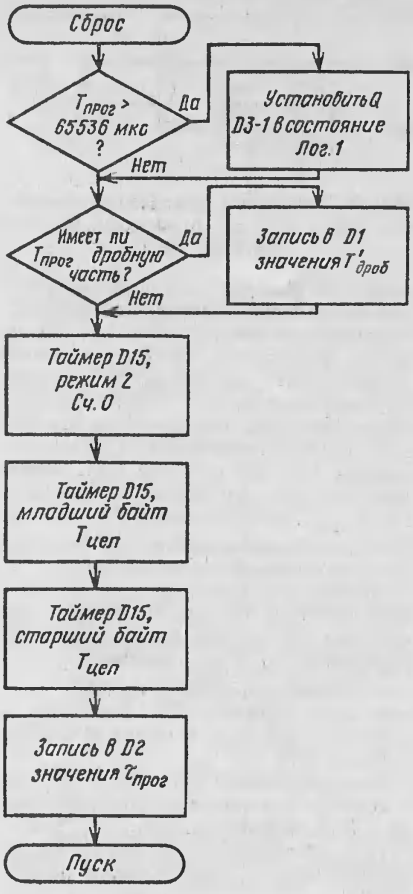


Рис. 10. Блок-схема программирования генератора

угольные импульсы, длительность которых равна периоду входных импульсов — 2,0 мкс. Эти импульсы подаются на вход «—1» счетчика D6, на информационные входы D1, D3, D4 которого поданы Лог. 0, а вход D2 может иметь состояние как Лог. 1, так и Лог. 0.

Если требуемый $T_{прог}$ лежит в диапазоне 1...65536 мкс, вход D2 элемента D6 находится в состоянии Лог 0. При $T_{прог}$ 65536 мкс D2 устанавливается в состояние Лог. 1. Состояние D2 предварительно записывается программно. Элемент памяти — триггер D3.1, прямой выход которого соединен со вторым разрядом входа элемента D6.

Счетчик D11 используется в качестве делителя частоты на 10. Учитывая ограничения на счетные импульсы таймера D15, выход «$\leqslant$0» счетчика D11 подан на одновибратор D13.

Каждый раз при окончании формирования программируемого периода $T_{прог}$ новый цикл генератора начинается перепадом импульса со счетчика D6.

В качестве эталонного генератора используется кварцевый генератор без LC-контура [5], выход которого подан на триггер Шмитта, формирующий прямоугольный импульс.

Отметим, что при необходимости можно увеличить $\tau_{прог}$, соответственно наращивая разрядность на D2 и D5. Можно увеличить нижний и верхний пределы $T_{прог}$.

В первом случае необходимо использовать счетчик D11 в качестве делителя на пять (изменяя соединения на его входах), т. е. на вход CLK0 D15 будут поступать импульсы с частотой 2 МГц.

В этом случае нижний предел $T_{прог}$ будет равен 1,0 мкс. Верхний предел $T_{прог}$ увеличивается соответствующим наращиванием разрядности на D3.1. Точность установки $T_{прог}$ и $\tau_{прог}$ в основном определяется стабильностью кварцевого генератора.

Приведенный выше генератор программировался с помощью ДВК «Электроника НМС 01 90.1».

Адрес для справок: 375078, Ереван, НПО АНИ. Телефон: 35-48-13

ЛИТЕРАТУРА

1. Манасевич В. Синтезаторы частот. Теория и проектирование.— М.: Связь, 1979, с. 262—265.
2. ОСТ 11 348.917—82. Микросхемы интегральные полупроводниковые. Серия КР580. Руководство по применению.— С. 65—83.
3. Маслий В. Н., Кулик А. А. Делитель частоты с переменным коэффициентом деления // ПТЭ.— 1985.— № 1.— С. 125—126.
4. Торгов Ю. И. Программируемый таймер КР580ВИ53 и его применение // Микропроцессорные средства и системы.— 1984.— № 4.— С. 77—84.
5. Титце У., Шенк К. Полупроводниковая схемотехника / Пер. с нем.— М.: Мир, 1982, с. 300—301.

Статья поступила 06.05.87

УДК 681.326—181.4

С. Д. Бушуев, М. Г. Диктерук, Д. Ж. Жунусов, С. В. Иносов

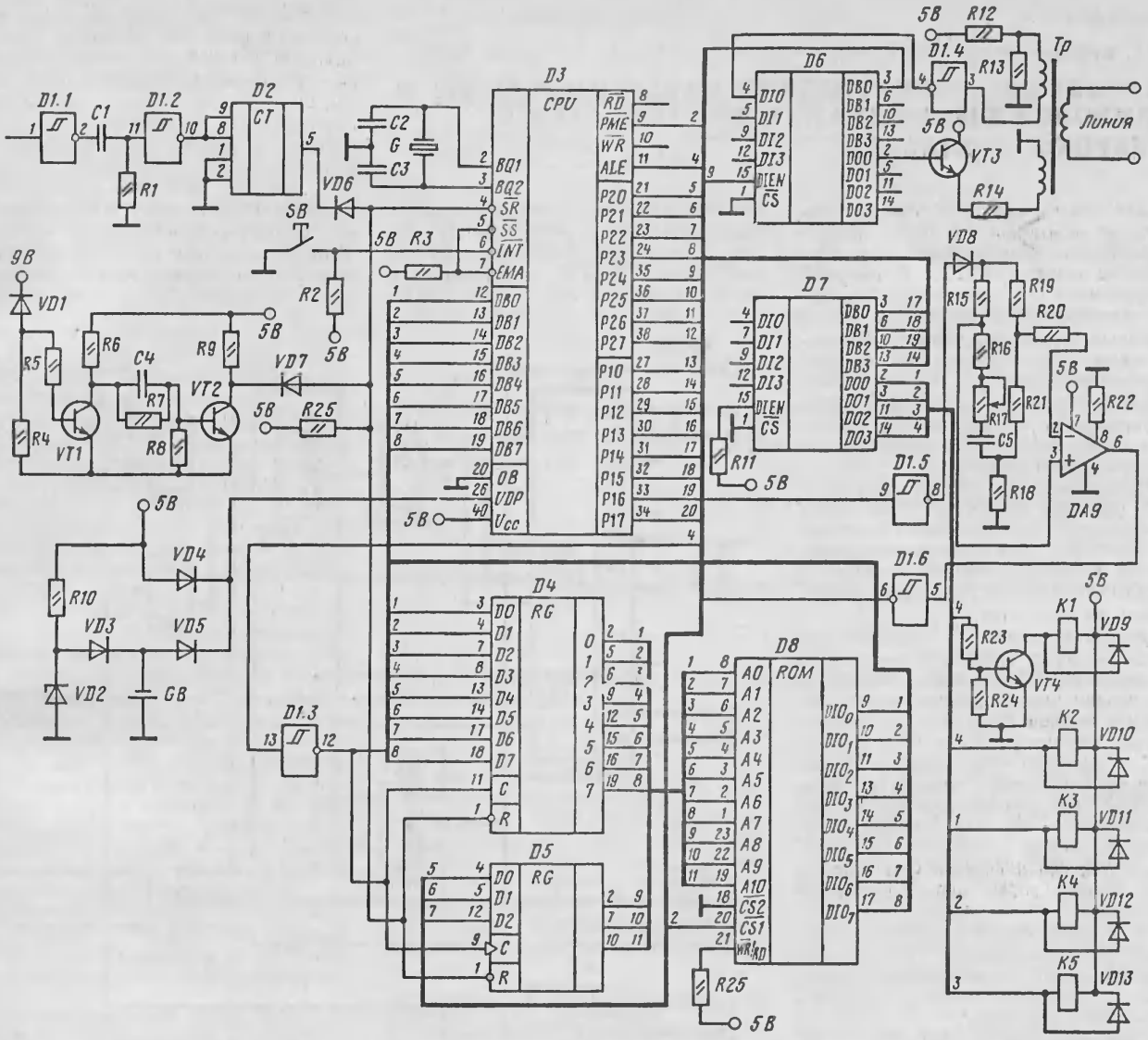
ЛОКАЛЬНЫЙ РЕГУЛЯТОР ТЕМПЕРАТУРЫ НА ОСНОВЕ ОЭВМ СЕРИИ К1816

Локальный регулятор предназначен для контроля и изменения основных параметров технологического процесса термообработки: текущей температуры, времени пропарки, графика изменения температуры. Регулятор работает совместно с центральной микроЭВМ.

Основу прибора (см. рисунок) составляет ОЭВМ К1816ВЕ39 (D3) в сочетании с регистрами К555ИР27 (D4) и К155ТМ8 (D5) для хранения адреса, шинными формирователями (ШФ) К589АП26 (D6, D7), РПЗУ К573РФ5 (D8), преобразователем сопротивление-код К140УД20 (DA9), десятичным делителем частоты К155ИЕ1 (D2), триггером Шмитта (ТШ) К155ТЛ2 (D1), усилителем на транзисторе КТ315 (VT3), формирователем импульса сброса при отключении питания на транзисторах КТ315 (VT1, VT2), схемой дополнительного питания энергонезависимого ОЗУ ОЭВМ на стабилитроне КС137 (D2) и диодах КД511 (VD3... VD5), гальваническим элементом питания GB, коммутаторами для управления исполнительными механизмами.

Тактовую частоту работы ОЭВМ определяет кварцевый резонатор G с частотой 8 МГц, подключенный к входам BQ 1 (2) и BQ 2 (3). Для инициализации функционирования К1816ВЕ39 вырабатывается сигнал системного сброса. После прихода импульса сброса выборка команд из РПЗУ начинается с нулевого адреса. Адреса выставляются на 8-разрядной шине DB, трех младших разрядах порта P2 и заносятся в регистры D4, D5 по сигналу ALE. С выходов этих регистров адреса поступают на адресные шины РПЗУ, где по данному адресу выбирается соответствующая команда при наличии сигнала PME. Порт DB после выдачи адреса переходит в высокоимпедансное состояние и готов принять команду (инструкцию), выбираемую из РПЗУ.

В процессе выполнения команд программы на соответствующие шины портов P1, P2 выдаются сигналы управления, необходимые для запуска преобразователя сопротивление-код и считывания кода температуры, подачи сигналов управления на коммутаторы исполнительных механизмов.



Принципиальная схема локального регулятора температуры

Для организации связи ОЭВМ с центральной микроЭВМ разработана схема, состоящая из ШФ D6, импульсного трансформатора ТР, ТШ, D1.4, усилителя на транзисторе VT3 и двухпроводной линии связи (ЛС). Управление режимом ввода-вывода информации через ШФ осуществляется сигналом DIEN. Передаваемый в ЛС сигнал усиливается по току транзистором VT3, затем через импульсный трансформатор ТР, ТШ, D1.4 и ШФ D6 поступает для обработки в ОЭВМ. Резисторы R12 и R13 предназначены для согласования волнового сопротивления линии связи. Шифратор D8 выводит сигналы управления на коммутаторы.

Преобразователь сопротивление-код, выполненный на операционном усилителе К140УД20 (DA 10), позволяет считывать аналоговую информацию с любых резистивных датчиков, например с термометров сопротивления. Для считывания используется всего один логический (одноразрядный) вход и один выход порта К1816ВЕ39. Преобразователь не требует управляющих сигналов и отдельного блока питания (используется источник 5 В ОЭВМ), основной элемент преобразователя — электрический мост, образованный резисторами R15...R17, R20, R21. На случай кратковременного отключения питания предусмотрен гальванический элемент дополнительного питания GB.

Локальный регулятор внедрен на Черкасском ДСК № 1 на кассетных установках. Локальная сеть включает центральную микроЭВМ «Роботрон 1715» и ряд ОЭВМ (до 32), подключенных к двухпроводной линии связи. Удаление центральной ЭВМ от локальных регуляторов достигает 400 м при скорости передачи данных 1200 Бод.

Применение локальных регуляторов позволяет при более низких аппаратурных затратах оперативно изменять режим пропарки, сокращать количество теплоносителя за счет точной отработки графика пропарки.

252037, Киев, Воздухофлотский пр., 31, Киевский инженерно-строительный институт; тел. 272-94-00

Статья поступила 24.03.88

УДК 681.325.5-181.4

С. С. Бубович

УСТРОЙСТВО ФОРМИРОВАНИЯ СИНХРОИМПУЛЬСОВ И СИМВОЛОВ В ИЗОБРАЖЕНИИ ДЛЯ ТВ-СИСТЕМ С МИКРОКОНТРОЛЛЕРОМ

Для наблюдения за состоянием объектов и контроля все более широко применяются замкнутые оптико-телевизионные системы (ОТС). В наиболее современных ОТС используются микроконтроллеры (МК) для автоматизации контроля, подстройки, управления механизмами светорегулирования, подфокусировки объектива [1]. При этом избыток вычислительной мощности МК можно применить для расширения сервисных возможностей ОТС, сокращения ее аппаратурного состава.

Предлагаемое устройство совместно с МК (на базе микросхем серии К580) обеспечивает формирование ТВ-синхросигналов (гасящих и синхроимпульсов), вывод псевдографических символов и шкалы отображения аналоговой информации на индикатор.

Устройство (рис. 1) содержит два таймера серии К580, три микросхемы серии К155. МК управляет таймерами по входам адреса, данных, чтения, записи и выбора кристалла. На счетные входы счетчиков СЧ1 и СЧ2 таймера М1 и счетчика СЧ3 таймера М2 подаются тактовые импульсы фазы $\overline{Ф2ТТЛ}$ синхрогенератора МК. По сигналу «Сброс» МК предварительно устанавливаются триггеры М3, М5. Триггеры М3, М5 и счетчик СЧ1 таймера М2 связаны с МК через порт ввода-вывода.

Импульсы общего тактового генератора при формировании ТВ-синхросигналов и символов отображаемой информации позволяют жестко привязать символы к ТВ-растру.

Синхросигналы формируются так. После включения МК по программе инициализации МП последовательно настраивает для работы в режим 1 (ждущий мультивибратор) счетчики СЧ1 и СЧ2 таймера М1. Во время последующей работы счетчик СЧ1 запрещает работу счетчику СЧ2 подачей низкого логического уровня со своего выхода окончания счета (КН1). После запрограммированного временного интервала счетчик СЧ1 высоким логическим уровнем с выхода КН1 активизирует работу счетчика СЧ2. Счетчик СЧ2 пересчитывает импульсы фазы Ф2, начиная с импульса, следующего за фронтом сигнала запуска, поступающего с выхода КН1. Отработав свой запрограммированный временной интервал, счетчик СЧ2 вновь разрешает работу счетчику СЧ1 (счетчики работают поочередно) При этом на выходе счетчика СЧ1 формируется последовательность гасящих импульсов двойной строчной частоты — 2 f СГИ (рис. 2)

Аналогично программируются и работают счетчики СЧ1 и СЧ2 таймера М2, вырабатывающие последовательность кадровых гасящих импульсов (КГИ). Эти счетчики переключаются с частотой импульсов 2 f СГИ, поступающих с выхода инвертора М4. С выхода КН3 счетчика СЧ3 таймера М1 снимаются строчные синхроимпульсы (ССИ) Счетчик также настраивается в режим ждущего мультивибратора — запускается импульсом (СГИ). Длительность

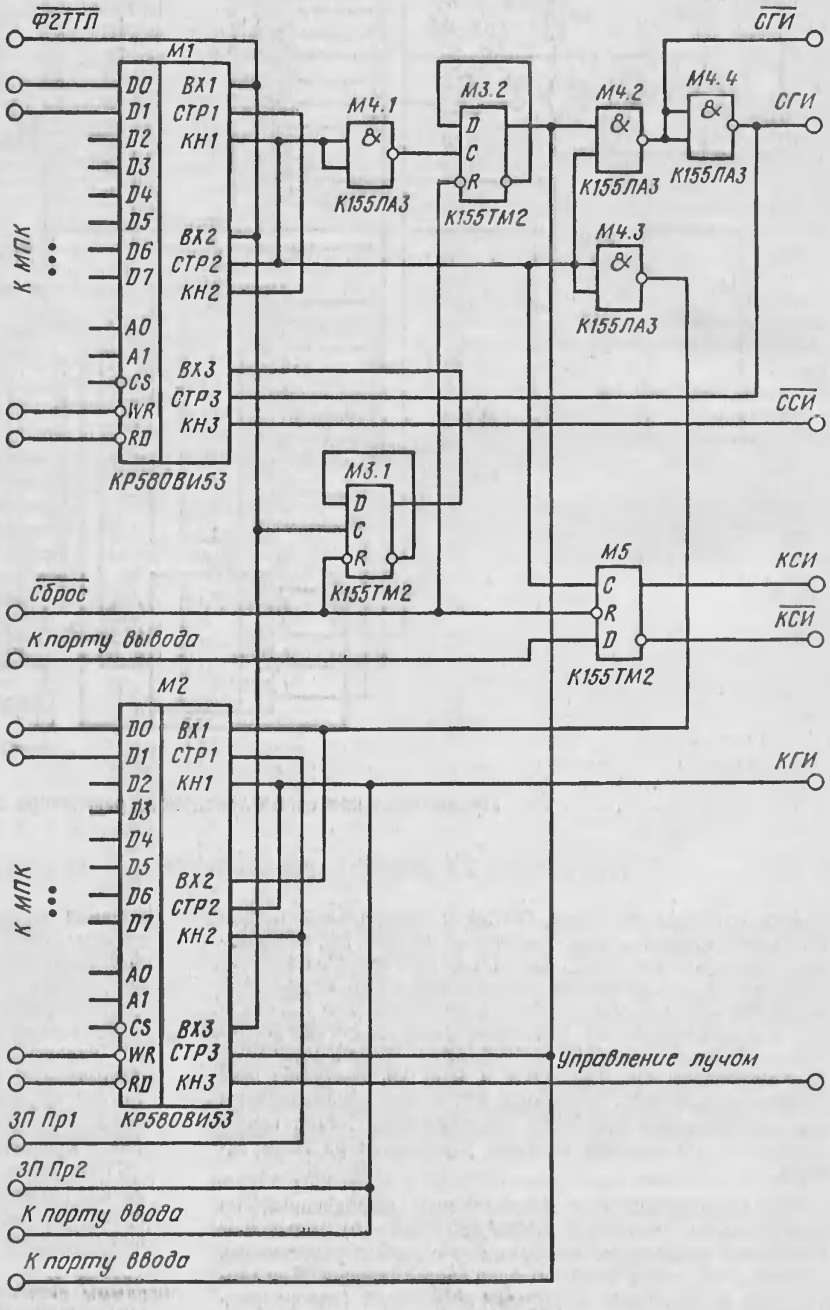


Рис. 1. Принципиальная схема устройства формирования синхроимпульсов и символов в изображении

импульса, формируемого счетчиком,— 5 мкс, задержка его фронта относительно фронта импульса СГИ — 1,25 мкс (см. временные диаграммы на рис. 2).

Для формирования кадрового синхроимпульса (КСИ) МП, обслуживая по запросу ЗП ПР2, циклически анализирует состояние счетчика СЧ2 таймера М2 в режиме считывания «на лету» и уменьшает его содержимое на величину n (число периодов импульсов двойной строчной частоты, на которое должен быть задержан фронт КСИ относительно фронта КГИ). Затем МП через свободный разряд порта ввода-вывода устанавливает высокий логический уровень на D-входе триггера М5. Следующим импульсом частоты 2 f СГИ, поступающим на вход синхронизации этого триггера, информация переписывается в триггер, на выходе которого формируется фронт импульса КСИ. Аналогично по истечении еще n периодов формируется срез КСИ. При n=5 длительность импульса КСИ — 160 мкс, задержка относительно фронта импульса КГИ — 160 мкс.

В счетчики СЧ1 и СЧ2 таймера М1 загружаются (рис. 3) константы 28Н и 16Н соответственно. При этом на выходе счетчика СЧ1 формируются импульсы длительностью 12 мкс и периодом следования 32 мкс. В счетчики СЧ1 и СЧ2 таймера М2 загружаются константы 02, 3FH и 30Н. На выходе счетчика 1 формируются импульсы длительностью 1,6 мс и периодом 20 мс. Константы рассчитаны для частоты импульсов Ф2ТТЛ, равной 2 МГц. Адреса регистров таймеров указаны условно и должны быть конкретизированы при подключении устройства к контроллеру. На процедуру формирования импульса КСИ МП затрачивает около 20 % времени обратного хода кадровой развертки.

Для формирования упрощенных символов на ТВ-индикаторе используется счетчик СЧ3 таймера М2. Сигнал с его выхода через буферную схему смешивается с видеосигналом и управляет модуляцией луча индикатора. При вертикальном направлении строчной развертки символы (рис. 4) можно использовать для отображения на экране состояния различных узлов ОТС (например, достижения крайних положений механизмами перемещения диафрагмы и светофильтров или их отказов при проведении автоматизированного контроля).

Символы формируются так. Счетчик СЧ3 настраивается в режим 5 (генератор аппаратно формируемого строба) и далее в него загружаются значения констант, определяющих момент окончания счета и, следовательно, взаимное расположение элементов фигур на рис. 4. Счетчик переключается с тактовой частотой Ф2ТТЛ, счет разрешается сигналом с выхода триггера М3. Подпрограмма формирования символов обрабатывается по сигналу запроса прерывания ЗП Пр1 (рис. 5)

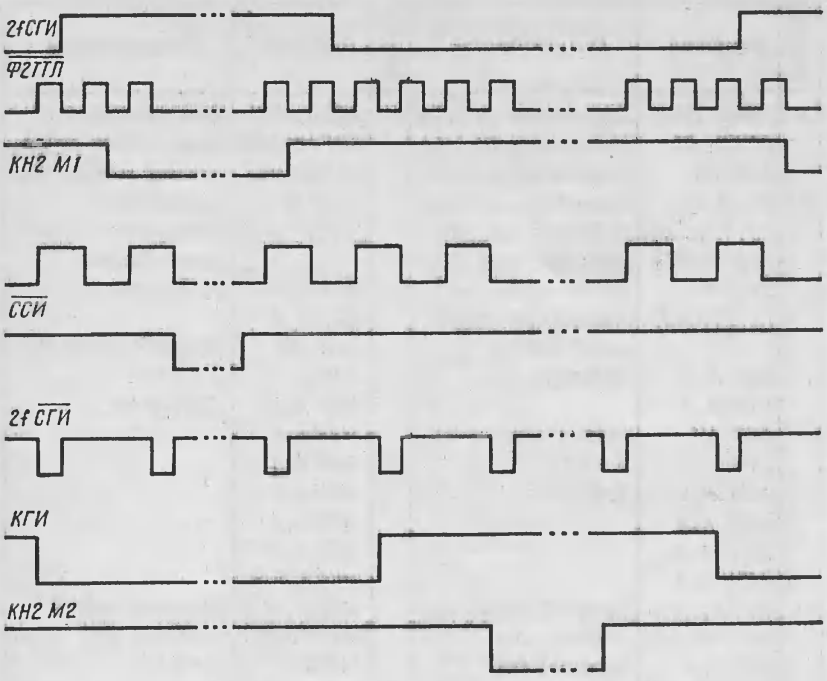


Рис. 2. Временные диаграммы работы устройства (масштаб показан условно)

В режиме считывания «на лету» МП анализирует состояние счетчика СЧ1 таймера М2 и, после совпадения значения его младшего байта с величиной сравнения, перенастраивает счетчик. Константы, с которыми сравнивается содержимое счетчика СЧ1 (определяющие номер строки смены информации), а также константы, загружаемые в счетчик СЧ3, должны быть предварительно введены в ОЗУ МК во время выполнения основной программы и расположены в ячейках попарно. Адресация к ним происходит через регистровую пару Н1, а переход от одного значения к другому — путем инкрементирования содержимого регистра 1. Использование интервальных таймеров для формирования символов требует значительно меньшего объема ОЗУ, чем в случае [2]. Отметим, что в режиме считывания «на лету» период циклического опроса счетчика выбран равным периоду изменения его состояния, т. е. 32 мкс (64 такта). Это позволяет задавать положение символов с точностью до одной ТВ-строки.

Мнемокод	Комментарий
MVI A, 12H	Программирование счетчика
OUT (адр. РУС М1)	СЧ1 таймера М1
MVI A, 28H	в режим 1
OUT (адр. СЧ1 М1)	
MVI A, 52H	Программирование счетчика
OUT (адр. РУС М1)	СЧ2 таймера М1
MVI A, 16H	в режим 1
OUT (адр. СЧ2 М1)	
MVI A, 72H	Программирование счетчика
OUT (адр. РУС М1)	СЧ3 таймера М1
MVI A, 05H	в режим 1
OUT (адр. СЧ3 М1)	
MVI A, 32H	Программирование счетчика
OUT (адр. РУС М2)	СЧ1 таймера М2
MVI A, 3FH	в режим 1
OUT (адр. СЧ1 М2)	
MVI A, 02H	
OUT (адр. СЧ1 М2)	
MVI A, 52H	Программирование счетчика
OUT (адр. РУС М2)	СЧ2 таймера М2
MVI A, 30H	в режим 1
OUT (адр. СЧ2 М2)	

Рис. 3. Фрагмент программы первоначального запуска таймеров, используемых для формирования ТВ-синхросигналов

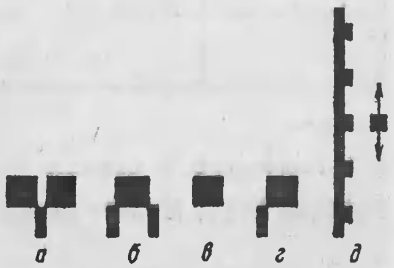


Рис. 4. Вид символов псевдографической информации

а — отказ светофильтров; *б* — отказ диафрагмы; *в* — отказ заслонки; *г* — разрешение (исправность); *д* — шкала

Метка	Мнемокод	Комментарий	Метка	Мнемокод	Комментарий
	PUSH PSW	Сохранение содержи-	M3	MVI A,02H	Считывание младшего байта содержимого счетчика СЧ1
	PUSH HL	мого регистров A,F,H,L		OUT(РУС М2)	
	LXI HL	Загрузка адреса в HL		IN(СЧ1 М2)	
M1	MVI A,02H	Считывание младше-		CMP M	Сравнение
	OUT(РУС М2)	го байта содержи-		IN(СЧ1 М2)	Считывание старшего байта
	IN(СЧ1 М2)	мого счетчика СЧ1			
	CMP M	Сравнение		MOV A,A	Задержка
	IN(СЧ1 М2)	Считывание стар-		MOV A,A	
		шего байта		JNZ M3	Переход к метке М3
	MOV A,A	Задержка		INR L	L=L+1
	MOV A,A			MOV A,A	Задержка
	JNZ M1	Переход к метке М1		MOV A,A	
	INR L	L=L+1		MOV A,A	
	MOV A,A	Задержка		MOV A,A	
	MOV A,A			MOV A,A	
	MOV A,A			MOV A,A	
	MOV A,A			MVI A,00H	
	MVI A,9A H	Загрузка инструкции		MOV A,M	Загрузка кода в счетчик СЧ3
	OUT(РУС М2)	режима 5		OUT(СЧ3 М2)	
	MOV A,M	Загрузка кода в		INR L	L=L+1
	OUT(СЧ3 М2)	счетчик СЧ3	M4	MOV A,02H	Считывание младшего байта содержимого счетчика СЧ1
	INR L	L=L+1		OUT(РУС М2)	
M2	MVI A,02H	Считывание млад-		IN(СЧ1 М2)	
	OUT(РУС М2)	шего байта содер-		CMP M	Сравнение
	IN(СЧ1 М2)	жимого счетчика СЧ1		IN(СЧ1 М2)	Считывание старшего байта
	CMP M	Сравнение			
	IN(СЧ1 М2)	Считывание стар-		MOV A,A	Задержка
		шего байта		MOV A,A	
	MOV A,A	Задержка		JNZ M4	Переход к метке М4
	MOV A,A			INR L	L=L+1
	JNZ M2	Переход к метке М2		MOV A,A	Задержка
	INR L	L=L+1		MOV A,A	
	MOV A,A	Задержка		MOV A,A	
	MOV A,A			MOV A,A	
	MOV A,A			MOV A,A	
	MOV A,A			MOV A,A	
	MOV A,A			MOV A,A	
	MOV A,A			MVI A,9A H	Загрузка инструкции режима(запрет счета)
	MVI A,00H			OUT(РУС М2)	
	MOV A M	Загрузка кода в		MVI A,NN H	Сообщение о конце обслуживания прерывания
	OUT(СЧ3 М2)	счетчик СЧ3		OUT(адр. Пр.)	
	INR L	L=L+1		POP PSW	Восстановление содержимого регистров
				POP HL	
				RET	Возврат

Рис. 5. Фрагмент подпрограммы формирования символов

При формировании шкалы отображения аналоговой информации используются два режима работы счетчика СЧ3: режим 2 (делитель частоты) — для формирования штрихов и режим 5 — для формирования подвижной метки. Перенастройка происходит во время низкого уровня напряжения на выходе КН1 счетчика СЧ1 таймера М2, формирующего импульсы КГИ, и выходе триггера М3, что обнаруживается микропроцессором при циклическом опросе соответствующих разрядов порта ввода-вывода. К этим операциям он приступает, завершив формирование импульса КСИ и запретив при этом программное обслуживание прерываний по входу ЗП Пр1. Положение метки определяется цифровым кодом, поступающим в МПК с внешнего устройства через порт ввода-вывода и размещаемым в ОЗУ. Программа формирования шкалы подобна программе формирования символов.

Предложенное устройство, включающее в себя незначительное количество элементов, существенно расширяет функции МК, выполняемые им в замкнутой оптико-телевизионной системе. В отличие от широко известного устройства сопряжения микроконтроллера с электронно-лучевой трубкой на базе микросхемы К580ВГ75 [3] предлагаемое устройство без дополнительного применения ПЗУ знакогенератора и буферного регистра формирует набор синхросигналов для функционирования замкнутых телевизионных ТВ-систем.

ЛИТЕРАТУРА

1. Бычков Б. С., Тимофеев Б. Н. // ТКТ.— 1986.— № 11.— С. 18.
2. Перетягин И. В., Цибульник А. Н. // ПТЭ.— 1985.— № 4.— С. 147.
3. Зеленко Г. В. // Микропроцессорные средства и системы.— 1985.— № 3.— С. 60.

Статья поступила 12.07.88

УДК 681.3.06

А. А. Гальченко, В. В. Самойлов

РЕЗУЛЬТАТЫ ИЗМЕРЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ ЭВМ

Для измерения производительности ЭВМ обычно используют наборы специальных эталонных программ, называемых бенчмарками. В статье рассматриваются две наиболее распространенные бенчмарки: «Решето» [1, 2] и «Вычисления», которые фактически являются стандартом для микроЭВМ. В информационных сообщениях о зарубежных персональных микрокомпьютерах, как правило, приводятся времена выполнения этих программ. Ниже подобная информация дается для некоторых зарубежных и отечественных ЭВМ. Материал статьи можно использовать для определения быстродействия вычислительных машин и для сравнения их характеристик.

Бенчмарка «Решето» находит простые числа, используя известный алгоритм Эратосфена [3]. Она применяется для определения эффективности реализации базового набора инструкций

ЭВМ: пересылок, переходов, целой арифметики и т. п. Инструкции плавающей арифметики в программе не используются. Текст «Решето» на Фортране (рис. 1) немного отличается от оригинала: во-первых, в эстетических целях, изменены номера меток и, во-вторых, исправлена одна описка, которая увеличивала время выполнения программы на ничтожно малую величину. В [2] есть тексты «Решето» на всех популярных языках программирования.

Программа «Вычисления» выполняет 10 000 умножений и 10 000 делений чисел с плавающей запятой и применяется для определения эффективности реализации этого класса инструкций. На рис. 2 приведена программа «Вычисления», переписанная с БЕЙСИКа.

Результаты выполнения бенчмарок «Решето» и «Вычислений» приведены в табл. 1 и 2. При прогонке бенчмарок создавались условия, позволявшие выявить быстродействие компьютера. Например, при работе в системах разделения времени обеспечивались, по возможности, монопольный режим выполнения бенчмарки и оптимальные по быстродействию коды программ. Для этого применялась оптимизация транслятора, отключалась трассировка ошибок исполнения, учитывались особенности аппаратуры.

Основной вывод из полученных результатов: нет четких границ между микро-, мини- и просто ЭВМ по такой важнейшей их характеристике, как быстродействие. Действительно, в верхних частях таблиц находятся такие разные компьютеры, как IBM PC AT, «Электроника 79», СМ1700 и ЕС1055. При выполнении бенчмарок на отечественных ЭВМ наблюдался большой разброс времени исполнения. Например, для машин «Искра 226» и ДВК3 он достигал 30 %. Хорошие характеристики показала ПЭВМ «Электроника 85», которая успешно конкурирует с лучшими зарубежными образцами микроЭВМ.

Таблица 1

Результаты выполнения программы «Решето»

ЭВМ	ОС	Компилятор	Время, с
VAX 11/785	YMS	Фортран 77	0,7
IBM PC AT	MS DOS	Турбо Си	1,8
Э 79	RSX-11M	C	2,2
Э-79	RSX-11M	Фортран 77	3,0
M24**	MS DOS	C (Microsoft)	3.1
СМ1700	МОС ВП	Фортран 77	3,25
VAX-11/730	VMS	Фортран 77	3,25
ЕС1055	CMS VM	Фортран	3,5
IBM PC AT	MS DOS	Турбо Паскаль	3,9
ЕС1040	ОС	Фортран	4,4
СМ1420	RSX-11M	C (Decus)	5,0
СМ4	МНОС	C (UNIX)	5,4
СМ4	RSX-11M	Паскаль (OMSI)	5,7
СМ4	RSX-11M	C (Whitesmith)	5,8
Э-85	RT-11	C (Whitesmith)	5,9
ПРАВЕЦ 16	MS DOS	Турбо Си	6,0
IBM PC XT	MS DOS	C (Microsoft)	6,0
СМ4	RSX-11M	C (Decus)	6,3
СМ1420	RSX-11M	Фортран 77	6,3
СМ4/20	RSX-11M	C (Decus)	6,8
СМ4	RSX-11M	Фортран 77	8,0
СМ4/20	PT-11	Фортран	8,0
СМ4/20	RSX-11M	Фортран 77	8,4
Э-85	RT-11	Фортран	8,5
IBM PC XT	VENIX	C	9,8
СМ4	RSX-11M	Фортран	10,0
ДВК2	XI	C (Whitesmith)	10,7
ДВК3	RT-11	Фортран	11,0
IBM PC XT	MS DOS	Турбо Паскаль	14,0
IBM PC XT	MS DOS	Фортран 77 (Microsoft)	14,2
IBM PC XT	MS DOS	C86	15,2
Э-60 (M2)	XI	C (Whitesmith)	15,3
ПРАВЕЦ+Z80	CP/M	Фортран 80	26,0
СМ1300	RT-11	Фортран	29,0
СМ4	RSX-11M	FIG-FORTH	40,0
СМ4	МНОС	Фортран	59.4
IBM PC AT	MS DOS	BASICA	60,0*
XEN-1***	MS DOS	GW-BASIC	73.0*
СМ4	RSX-11M	БЕЙСИК 11	136,0*
«Искра 226»		БЕЙСИК	147,9*
ПРАВЕЦ 16	MS DOS	BASICA	182,0*
ПРАВЕЦ 16	MS DOS	GW-BASIC	186,0*
IBM PC XT	MS DOS	BASICA	187,0*
ПРАВЕЦ 83	DOS	БЕЙСИК	245,0
YAMAHA	MSX DOS	БЕЙСИК	325,0*

* Время выполнения одной итерации.
** Персональный компьютер фирмы OLIVETTI
*** Персональный компьютер фирмы APRICOT.

Рис. 1. Листинг программы «Решето» на Фортране

```
      logical flags(8191)
      integer i,j,k,count,iter,prime
      write (1,100)
100   format (' 10 iterations')
      do 40 iter=1,10
      count = 0
      do 10 i=1,8191
10    flags(i) = .true.
      do 30 i=1,8191
      if (.not.flags(i)) goto 30
      prime = i+i+3
      count = count+1
      k = i+prime
      if (k.gt.8191) goto 30
      do 20 j=k,8191,prime
20    flags(j) = .false.
30    continue
40    continue
      write (1,200)prime,count
200   format(1x,16,  is the largest of',i6,' primes')
      end
```

```
      real a,b,c
      nr = 5000
      a = 2.71828
      b = 3.14159
      c = 1.
      do 10 i=1,nr
      c = c*a
      c = c*b
      c = c/a
      c = c/b
10    continue
      type 100,c-1
100   format (1x,'error = ',g11.4)
      end
```

Рис. 2. Листинг программы «Вычисления» на Фортране

Таблица 2

Результаты выполнения программы «Вычисления»

ЭВМ	ОС	Компилятор	Время, с
VAX-11/785	VMS	Фортран 77	0,08
ЕС1055	VM	Фортран	0,2
Э-79	RSX-11M	Фортран 77	0,25
VAX-11/730	VMS	Фортран 77	0,3
СМ4/20	RSX-11M	Фортран 77	0,5
IBM PC XT	MS DOS	Фортран 77	0,75
СМ4	RSX-11M	Фортран	2,0
Э-85	RT-11	Фортран	3,5
IBM PC AT	MS DOS	Турбо Си	10,8
СМ1300	RT-11	Фортран	12,0
IBM PC XT	MS DOS	QUICKBASIC	14,0
IBM PC AT	MS DOS	BASICA	20,0
СМ4	RSX-11M	БЕЙСИК-11	43,0
ПРАВЕЦ 16	MS DOS	Турбо Си	48,7
ПРАВЕЦ 16	MS DOS	BASICA	59,5
IBM PC XT	MS DOS	BASICA	61,0
ПРАВЕЦ 16	MS DOS	GW-BASICA	62,0

В заключение выражаем признательность И. Н. Попову, М. А. Исаеву, И. И. Горянину, И. И. Нейману, Р. Ф. Накипову, М. И. Потемкину, Ю. П. Гречкину, В. Н. Литвиновой, А. Б. Истомину, С. В. Лысакову, которые, прогоняя бенчмарки или предоставляя машинное время, способствовали скорейшему завершению этой работы. Благодарим также Л. И. Карпуп за помощь в подготовке к публикации этой статьи.

Телефон 923-96-68, д. 217, Пущино

ЛИТЕРАТУРА

1. Gilbreath J. A high-Level Language Benchmark // BYTE.— 1981.— N 8.— P. 180.
2. Gilbreath J., Gilbreath G. Eratosthenes Revisited. Once More through the Sieve // BYTE.— 1983.— N 1.— P. 283.
3. Кнут Д. Искусство программирования для ЭВМ. Т. 2.— М.: Мир, 1977.

Статья поступила 18.03.88

ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА МИКРОЭВМ

УДК 681.325

А. Ф. Кузнецов

УСТРОЙСТВО ВВОДА ГРАФИЧЕСКОЙ ИНФОРМАЦИИ В ЭВМ

Устройство предназначено для ввода в оперативную память ЭВМ графической информации с одновременным отображением на экране дисплея. От устройств подобного типа оно отличается точным соответствием положений элементов на специальном прямоугольном поле и экране дисплея. Ввод информации напоминает естественное рисование или запись карандашом. Недостаток устройства — механическая связь «пера» с датчиками координат. Рабочее поле прибора имеет размеры 150×190 мм (необходимо выдержать соотношение 15:19); разрешающая способность изображения на экране дисплея 256×256 точек; частота записи координат в выходной буферный регистр 100 Гц. Частота воспроизведения точек (векторов) на экране зависит от быстродействия ЭВМ, для БК-0010 она составляет примерно 500 точек/с. Подключить устройство можно к любой ЭВМ, имеющей не менее двух 8-разрядных или одного 16-разрядного параллельного порта ввода с входными ТТЛ-уровнями. Системное ПЗУ ЭВМ должно иметь драйверы координатной графики. Питание осуществляется стабилизированным источником напряжения ±5В.

Конструкция и детали. Устройство выполнено в пластмассовом корпусе размерами 318×222×57 мм. В крышке проделано прямоугольное окно 150×190 мм, против которого располагается экран и устройство механической связи с датчиками координат, представляющее собой две подвижные, перпендикулярно расположенные рейки из органического стекла с продольными прорезями.

Все элементы механической части и печатная плата электрической схемы укреплены на коробчатом шасси размерами 256×140×45 мм из дюралюминия и толщиной 1,5 мм. Механическая часть устройства должна обеспечивать независимое перемещение планок А (рис. 1) по всей длине и ширине поля с превращением их поступательного движения во вращательное движение резисторов R2 и R5. Каждая планка А прикреплена винтом к двум верхним частям канатиков, каждый канатик представляет собой замкнутую нить, проходящую через два шкива Б. Нижняя часть одного из канатиков каждой пары проходит еще через

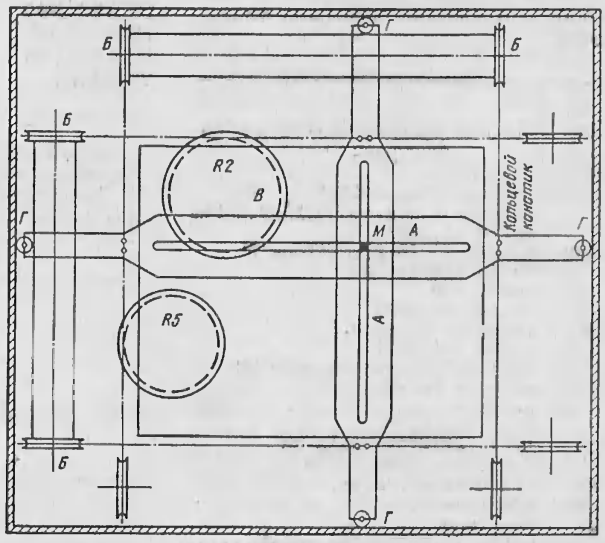


Рис. 1. Кинематическая схема устройства

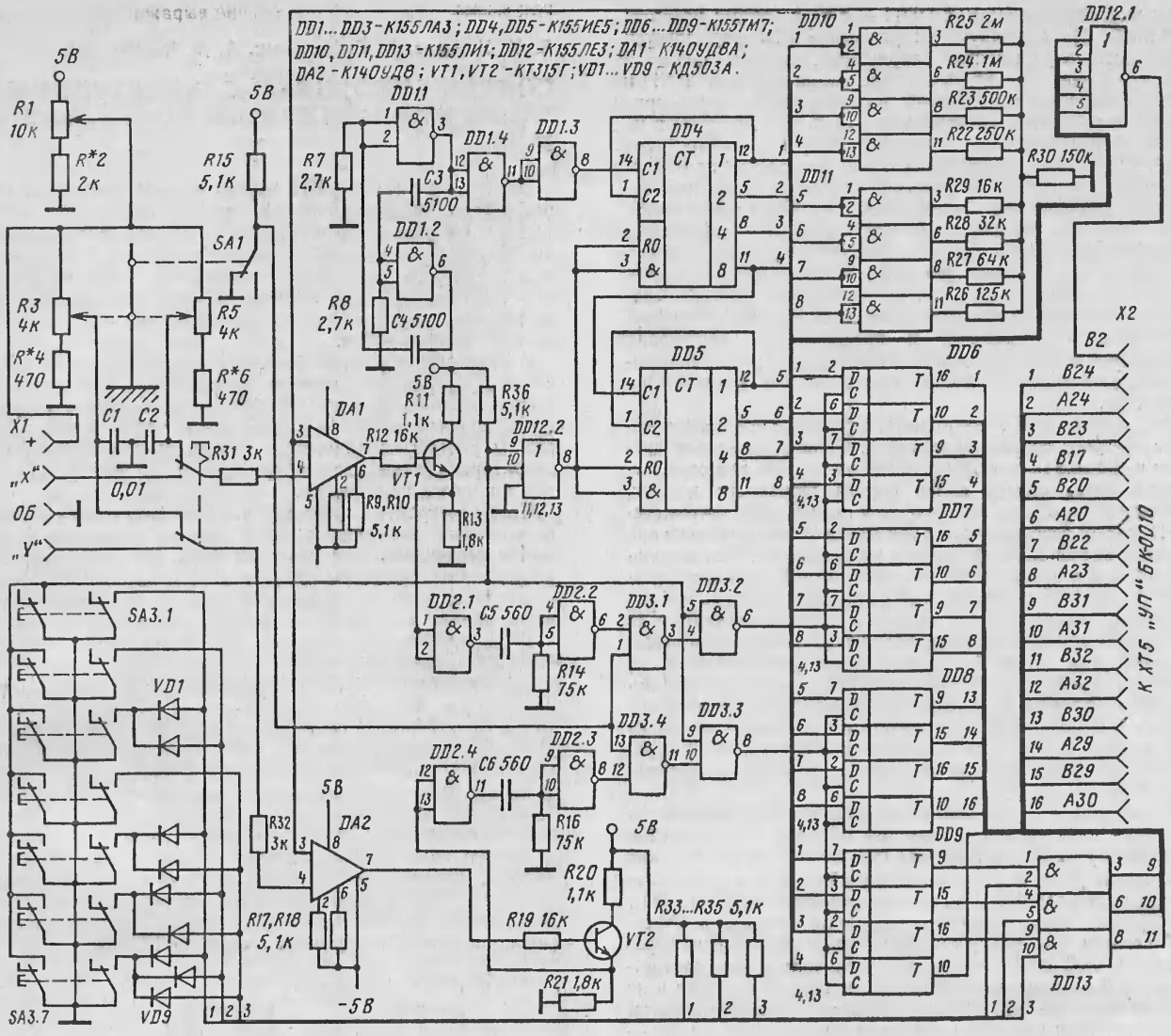


Рис. 2. Принципиальная схема устройства

один шкив В, надетый на ось переменного резистора. Одна пара шкивов планки соединена тонкой трубкой из дюралюминия для обеспечения параллельности движения. При этом углы поворота шкивов всегда одинаковы, что при равенстве их диаметров дает одинаковое перемещение канатиков и, следовательно, концов планки. Для устранения проскальзывания канатиков их нужно обернуть более одного раза вокруг шкивов. Необходимо предусмотреть также пружины для натяжения канатиков. Одна система верхних канатиков должна располагаться выше другой на расстоянии, чуть большем толщины планки, для уменьшения взаимного трения между планками и нитями. На концах планок расположены ролики Г из пластмассы, которые уменьшают трение между планками и стенками корпуса прибора. Размеры шкивов В должны быть такими, чтобы свободный ход канатика при вращении переменного резистора от одного крайнего положения до другого был равен длине соответствующей стороны поля.

Исполняющий элемент изготовлен из корпуса пишущей ручки. В отверстие нижнего конца ручки вставляется и укрепляется пишущий узел, применяемый в чертежных готовальнях. Он имеет небольшой свободный ход вдоль оси ручки с возвратной пружиной и внутренним концом включает тумблер при нажатии на ручку в момент выполнения рисунка.

Принципиальная схема устройства приведена на рис. 2. Исполняющий элемент (ИЭ) механически связан с двумя переменными резисторами R3 и R5 и тумблером SA1. Тумблер SA1 срабатывает при касании грифеля исполняющего элемента рабочего поля. Движок резистора R3 перемещается при изменении положения исполняющего элемента по горизонтали, а движок R5 — по вертикали. Это достигается работой механической части устройства. Таким образом, изменения напряжений, снимаемых с резисторов R3 и R5, пропорциональны координатам X и Y ИЭ. Эти напряжения подаются на инверсные входы микросхем DA1 и DA2, выполняющих роль компараторов. С помощью переменных резисторов R1, R4 можно изменять масштаб изображения. Генератор прямоугольных импульсов собран на микросхеме DD1 по схеме симметричного мультивибратора. Счетчик на микросхемах DD4 и DD5 работает в циклическом режиме. Выходы счетчика соединены с D-входами двух буферных регистров на микросхемах DD6...DD9 для записи двоичных кодов координат положения ИЭ а через элементы

И на микросхемах DD10 и DD11 — с резисторами матрицы R22...R29. На суммирующем резисторе R30 формируется ступенчато-возрастающее напряжение, которое подается для сравнения на вторые входы компараторов DA1 и DA2. Перепады напряжений на выходах компараторов через согласующие транзисторы VT1 и VT2 поступают на входы формирователей импульсов записи на микросхеме DD2, а импульсы записи через ключи управления на микросхеме DD3 — на входы разрешения записи буферных регистров. Первая пара ключей (DD3.1 и DD3.4) управляется тумблером SA1 ИЭ, вторая (DD3.2 и DD3.3) — клавишами выбора режима SA3. Ключи на МС DD3 прерывают запись координат при переводе ИЭ в другое место без рисования. При вводе координат в БК-0010 необходимо избежать совпадения во времени процессов записи информации в буферный регистр устройства и программного обращения к входному порту ЭВМ для чтения данных и учесть неодновременность записи значений координат X и Y при их неравенстве.

Эти условия можно выполнить, осуществляя запись в момент нулевого состояния счетчика. Однако интервала времени в 400 мкс может оказаться недостаточно для осуществления более одного цикла опроса готовности данных, поэтому в устройстве используются первые семь тактов счета. При этом освобождаются три линии шины, но только при нулевых значениях информации на остальных. Они используются для связи устройства с программой, минуя клавиатуру БК-0010. Первые семь строк от записи графической информации освобождаются подбором сопротивлений R4 и R6 так, чтобы при помещении исполняющего элемента в левый верхний угол точка на экране имела координаты X=0, Y=8 при масштабе 1:1. Элемент DD12.1 своим высоким логическим уровнем на выходе сигнализирует об отсутствии записи в буферный регистр устройства и возможности чтения данных в порт ЭВМ. Сигнал в ЭВМ вводится через системный порт (четвертый разряд регистра с адресом 177716_8).

Информация от управляющих клавиш SA3 передается во время пауз записи следующим образом. При нажатии на клавишу на выходе элемента DD12.2 появляется высокий логический уровень, устанавливающий счетчик в нулевое состояние. Сигналы Лог.1 на выходе элементов DD3.2 и DD3.3 разрешают запись нулевого состояния счетчика в буферный регистр. Поскольку три линии шины подключены через элементы МС DD13, то на них установится код, соответствующий номеру нажатой клавиши; появится сигнал и на выходе элемента DD12.1. Программа анализирует принятые значения: если результат окажется меньше семи, то это — сигнал управления программой, если нет — координаты.

В качестве резисторов R3, R5 датчиков координат желательно использовать проволочные резисторы с линейными характеристиками, возможно бо́льшим радиусом дуги активного элемента и числом витков спирали более 256. Напряжение, подаваемое на вывод резистора R1 и микросхемы DD10 и DD11, должно быть стабильным, поэтому лучше «питать» эти элементы от отдельной сухой батареи или аккумулятора.

Настройка устройства заключается в подборе резисторов матрицы для обеспечения линейности ступенчато-возрастающего напряжения на резисторе R30, а также резисторов R4 и R6. Возможно потребуется подобрать резисторы R12 и R19 или заменить транзисторы VT1 и VT2 аналогичными с бо́льшим коэффициентом передачи тока для обеспечения надежного срабатывания логических элементов DD2.1 и DD2.4 при перепадах напряжений на выходах компараторов. В компьютере БК-0010 необходимо проверить наличие перемычки S2, соединяющей ввод порта с входом регистра системного порта (вывод 4 D12).

Программа обслуживания устройства состоит из двух частей, осуществляющих запись и воспроизведение информации. Объем памяти составляет около 1 Кбайта.

613400, п. Кумены Кировской области, ул. Гагарина, д. 39а, кв. 13; тел. 1-14-93

Статья поступила 1.02.88

УДК 681.326

В. И. Карасев, В. Г. Коломыц, А. Д. Чернявский

СОПРЯЖЕНИЕ ДВК3 С ЭЛЕКТРОННЫМ ТАБЛО КОЛЛЕКТИВНОГО ПОЛЬЗОВАНИЯ 15ИТ

Электронное табло 15ИТ можно широко применять для построения информационных систем. Размеры табло (1,5×3 м) позволяют отображать на нем информацию, доступную одновременно большому числу людей, например при проведении оперативных совещаний, когда информация, характеризующая состояние объекта управления в виде таблиц, графиков, диаграмм, оперативно отображается на электронном табло.

В автономном режиме работы табло информация заносится с клавиатуры дисплея, входящего в комплект поставки, в память дисплея и отображается на экране. Информация из памяти дисплея переносится на табло. Автономный режим работы табло требует обязательного присутствия оператора для вывода информации, т. е. этот режим не для широкого применения.

Чтобы построить информационные системы коллективного пользования, электронное **табло следует применять в качестве периферийного устройства ЭВМ**, для чего авторами разработаны аппаратно-программные средства, позволяющие отображать на табло статическую и динамическую информацию в символьной форме, выводимую с ДВК3.

К микроЭВМ «Электроника МС 1201» табло подключается по интерфейсу ИРПР, который в ДВК стандартной конфигурации используется для работы с печатающим устройством. Для управления режимами записи-считывания табло коллективного пользования интерфейс ИРПР платы микроЭВМ «Электроника МС 1201» дорабатывается. Режимом работы табло управляет сигнал с контакта 13 — СИ2 (рис. 1). Для его формирования используется RS-триггер (К155ТМ2). Сброс триггера (установка в Лог.0) осуществляется сигналом «Сброс» интерфейса ИРПР, т. е. записью в двенадцатый разряд регистра состояния интерфейса с адресом 177514 (стирание информации на табло). В состояние Лог.1 триггер устанавливается неиспользуемым разрядом данных [7] или записью единицы в седьмой разряд регистра данных интерфейса с адресом 177516.

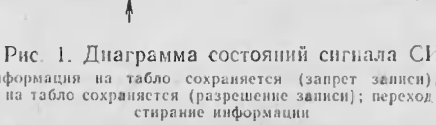


Рис. 1. Диаграмма состояний сигнала СИ2.

U_0 — информация на табло сохраняется (запрет записи), U_1 — информация на табло сохраняется (разрешение записи); переход U_0 в U_1 — стирание информации

Принципиальная схема доработанного интерфейса для управления электронным табло коллективного пользования представлена на рис. 2.

Работоспособность триггера обеспечивается перемычкой с контакта 48 разъема XT2 на +5 В в микроЭВМ «Электроника МС 1201.02».

Программное обеспечение комплекса реализовано в ОС ДВК.

Из-за различия временных характеристик электронного табло и устройства печати, стандартный драйвер устройства печати ОС ДВК, под управлением которой работает ДВК3, не обеспечивает устойчивый вывод информации на электронное табло, поэтому разработан дополнительный драйвер для работы с электронным табло. В качестве имени устройства используется «ТВ:».

Обмен информацией с табло происходит через стандартный регистр состояния и регистр данных устройства печати с адресами соответственно 177514 и 177516. Для очистки экрана электронного табло необходимо занести в регистр состояния код 40000 (восьмеричный) и в регистр данных

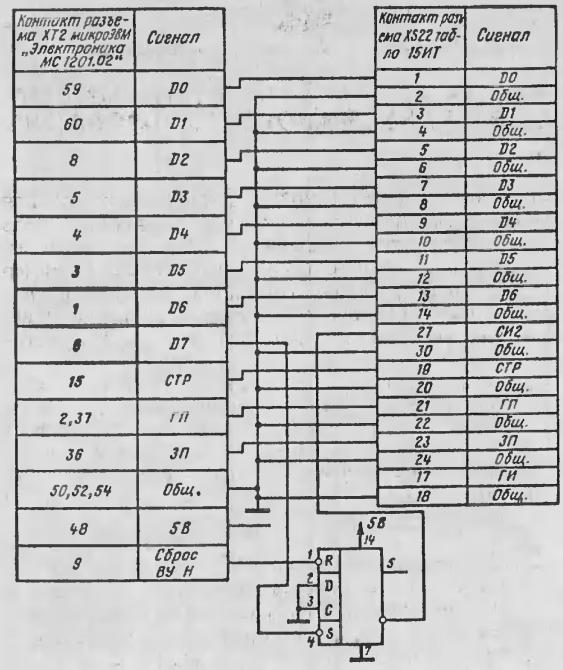


Рис. 2. Принципиальная схема интерфейса управления электронным табло коллективного пользования

код 200 (восьмеричный). Из-за особенностей временны́х параметров электронного табло код 200 в регистр данных необходимо записывать в цикле с параметром цикла, примерно равным 20.

После того, как в регистре состояния установлен бит «Готовность устройства», символы на экран выдаются занесением в регистр данных кода символа.

Драйвер электронного табло ТВ написан на макроассемблере и состоит из шести секций: определения, заголовок, запуск операции, обработка прерываний, завершение операции и заключение.

В секции определений заданы адреса регистров состояния и данных электронного табло, объем устройства в блоках (0 для электронного табло как устройства последовательного доступа), код идентификации устройства (377), тип устройства — только для вывода.

Секция заголовка формируется макрокомандой .DRBEG. Заголовок содержит информацию в фиксированных ячейках драйвера, которая используется монитором ОС в качестве интерфейса. В качестве аргументов в макрокоманде указываются: имя устройства (ТВ), адрес вектора прерывания, размер устройства в блоках (0), тип устройства (устройство последовательного доступа).

Секция запуска операции начинается непосредственно за макрокомандой .DRBEG и получает управление от монитора в начале обслуживания каждого элемента очереди. Возврат из этой секции выполняется с помощью команды «RTS PC».

Секция обработки прерываний начинается макрокомандой .DRAST, содержащей в качестве аргументов имя устройства и приоритет процессора при обработке прерываний.

Для завершения операции вывода используется макрокоманда .DRFIN, имеющая в качестве аргумента имя устройства. Макрокоманда генерирует последовательность команд, заносящую в регистр 4 адрес начала очереди к драйверу и передающую управление программе завершения операции в мониторе.

В секции заключения макрокоманда .DREND генерирует в конце драйвера таблицу адресов программ монитора для выполнения служебных функций и вычисляет размер драйвера для загрузки.

Для загрузки драйвера используется команда монитора INSTALL TB. Чтобы занести драйвер в систему, необходимо включить команду INSTALL в косвенный командный файл загрузки системы или генерировать монитор с помощью программы SYSGEN. Для этого требуется скорректировать таблицы «$PNAME» (имя драйвера) и слово состояния устройства в таблице «$STAT».

Из двух прикладных программ, использующих разработанный драйвер и демонстрирующих возможности электронного **табло для информирования посетителей выставки**, первая выводит заранее подготовленный текст на экран табло в виде «ролика» с регулируемой паузой между кадрами, а вторая — в виде «бегущей строки».

Разработанный аппаратно-программный комплекс демонстрировался в 1988 г. на выставке «Перестройка промышленности г. Москвы на интенсивный путь развития» (ВДНХ СССР) для информации посетителей о ходе перестройки на предприятиях столицы.

Телефон 531-14-03, Москва

ЛИТЕРАТУРА

1. Кокорин В. С., Кридинер Л. С., Попов А. А., Хохлов М. М. Тенденция развития диалоговых вычислительных комплексов // Микропроцессорные средства и системы.— 1986.— № 4.— С. 11—15.
2. Дшхунян В. Л., Борщенко Ю. И., Отрохов Ю. Л., Шишарин С. А. Одноплатные микроЭВМ «Электроника 1201» // Микропроцессорные средства и системы.— 1985.— № 2.— С. 8—13.
3. Мячев А. А., Иванов В. В. Интерфейсы вычислительных систем на базе мини- и микроЭВМ / Под ред. Б. Н. Наумова.— М.: Радио и связь, 1986.
4. Валикова Л. И., Вигдорчик Г. В., Воробьев А. Ю., Лукин А. А. Операционная система СМ ЭВМ РАФОС / Под ред. В. П. Семика.— М.: Финансы и статистика, 1984.

Статья поступила 23.03.88

УДК 681.326.35.77

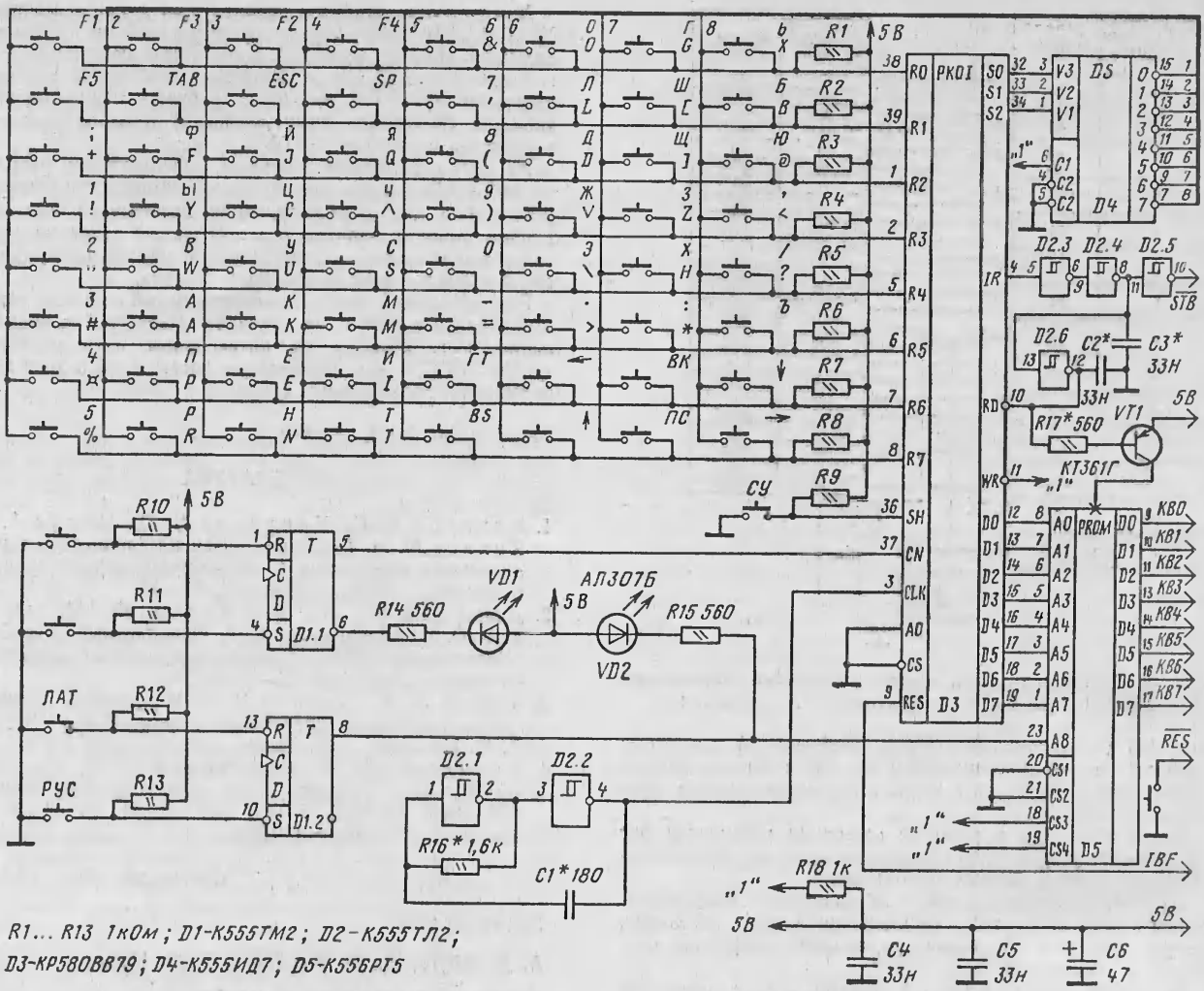
А. Д. Мруга, Д. Ю. Щербаков, В. М. Пинчук

ПРОСТОЙ КОНТРОЛЛЕР АЛФАВИТНО-ЦИФРОВОЙ КЛАВИАТУРЫ

Контроллер предназначен для ПЭВМ «Ириша» [1], но может работать с любой микроЭВМ, имеющей выносную клавиатуру. Его основу составляет БИС контроллера клавиатуры (КК) КР580ВВ79, сканирующая матрицу 8×8 клавиш.

В исходное состояние БИС КК устанавливается подачей сигнала высокого уровня на вход RES (в ПЭВМ «Ириша» после инициализации) с линии ответа IBF. БИС КР580ВВ79 обладает свойством после начальной установки сигналом RES выполнять функции в режиме с запрещением ввода кодов при нажатии двух и более клавиш, а также генерировать на выходе IRQ сигнал при правильном нажатии клавиш (одна нажата, остальные отжаты). Это свойство БИС КК позволило создать простое автономное устройство управления клавиатурой всего на пяти микросхемах (см. рисунок).

Узел формирования сигнала стробирования, выполненный на элементах D2.3...D2.6, по сигналу с выхода элемента D2.6 генерирует сигнал разрешения считывания КК, поступающий на вход RD микросхемы D3, а по сигналу с выхода элемента D2.5 — строб STB, по срезу которого код нажимаемой клавиши заносится во входной регистр. При подаче на вход RD сигнала низкого уровня выходы КК открываются и информация о нажатых клавишах поступает на адресные выходы A0...A7 ПЗУ D5 для перекодировки в соответствии с координатой нажимаемой клавиши. Перекодировка осуществляется нажатием клавиш СУ сигнала с выхода



Принципиальная схема контроллера алфавитно-цифровой клавиатуры на основе БИС КР580ВВ79

триггера ВЕРХНИЙ/НИЖНИЙ РЕГИСТР D1.1 и триггера РУС/ЛАТ D1.2.

Напряжение питания на ПЗУ D5 подается только на время считывания. Контроллер тактируется заниженной частотой 1 МГц с выхода генератора, собранного на элементах D2.1, D2.2. В остальном включение БИС КК не отличается от рекомендованного в работе [2].

252056, Киев, пр. Победы, 39, Киевский политехнический институт; тел. 441-12-58

ЛИТЕРАТУРА

1. Романов В. Ю., Барышников В. Н., Сороков М. А., Паначев Ф. И. Персональная ЭВМ «Ирша»: периферийные устройства, источник питания // Микропроцессорные средства и системы.— 1986.— № 3.— С. 53—59.
2. Каган Б. М., Сташин В. В. Основы проектирования микропроцессорных устройств автоматики.— М.: Энергоиздат.— 1987 — С. 254—256.

Сообщение поступило 15.03.88

УДК 681.325.5

Е. Н. Осипов

СОВМЕСТНАЯ РАБОТА БИС КР1506ХЛ2 И К1816ВЕ48 В СИСТЕМАХ УПРАВЛЕНИЯ БЫТОВОЙ РАДИОЭЛЕКТРОННОЙ АППАРАТУРОЙ

Современные системы управления сложной бытовой радиоэлектронной аппаратурой (БРЭА) на основе микроЭВМ позволяют значительно повысить комфортность, надежность и стабильность работы аппаратуры.

Сама микроЭВМ не может выполнять ряд специальных функций управления без интерфейсных устройств. Используя БИС КР1506ХЛ1 (передатчик) и КР1506ХЛ2 (приемник), разработанные специально для систем дистанционного и местного управления БРЭА, и однокристальные микроЭВМ [1] (ОЭВМ), можно успешно реализовать ряд функций современных функционально-насыщенных систем управления БРЭА [2]: дистанционное управление (ДУ) несколькими устройствами от одного пульта ДУ; автоматическое включение-выключение; бесшумную настройку и т. д.

Не ставя целью описание всего многообразия вариантов совместного использования БИС управления и микроЭВМ, покажем на конкретном примере

экономичную передачу команд управления между БИС КР1506ХЛ2 и популярной ОЭВМ К1816ВЕ48.

Особенность БИС КР1506ХЛ2: коды всех команд (выполняются ли они в самой БИС или нет) транслируются в последовательной форме через выход ДА и сопровождаются тактовыми сигналами на выходах SYN1 и SYN2. Таким образом можно вводить и использовать в ОЭВМ все 1008 команд.

К сожалению, вывод последовательного кода команд организован в БИС КР1506ХЛ2 не очень удобно — каждый информационный бит сопровождается не одним, а четырьмя тактовыми импульсами, следующими непрерывно (независимо от того, выводится код команды или нет). На рис. 1 показана временная диаграмма вывода кода команд с указанием временных характеристик при использовании кварцевого резонатора с частотой 4000 кГц.

Отметим, что не все доступные широкому пользователю микропроцессоры и однокристальные микроЭВМ способны принимать последовательный код в таком формате и с такой скоростью, с какой он выводится из БИС КР1506ХЛ2. Преобразование последовательного кода в параллельный с помощью обычных регистров сдвига требует дополнительных аппаратных средств для формирования тактовых импульсов и привязки их к стартовому биту информационного пакета.

При разработке систем управления для телевизора и цветного кассетного видеомагнитофона реализован программный прием кода команд с минимальными аппаратными затратами (на базе ОЭВМ серии К1816ВЕ48).

На схеме (рис. 2) рассматриваемой здесь части системы управления телевизором показаны также цепи формиро-

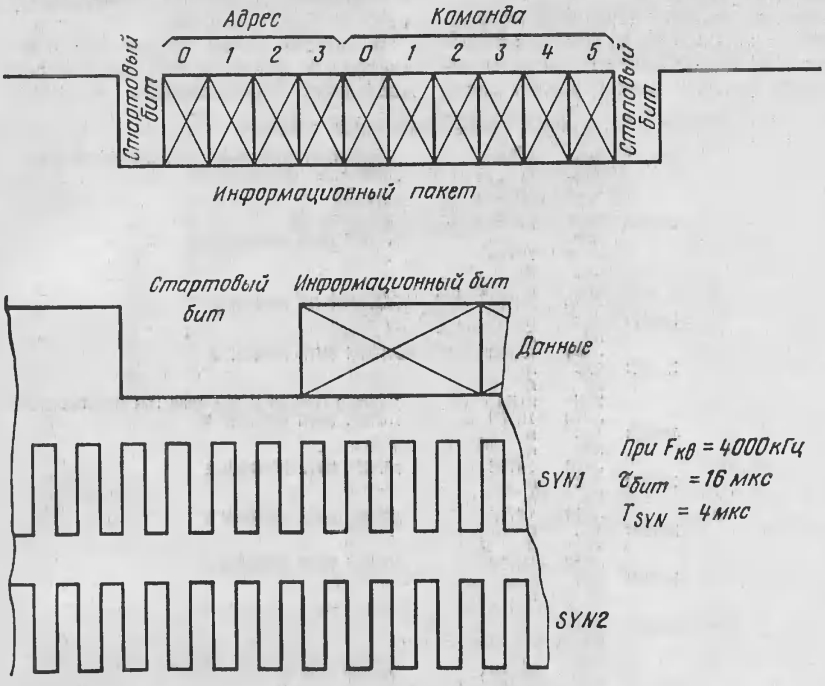


Рис. 1. Временная диаграмма процесса вывода кода

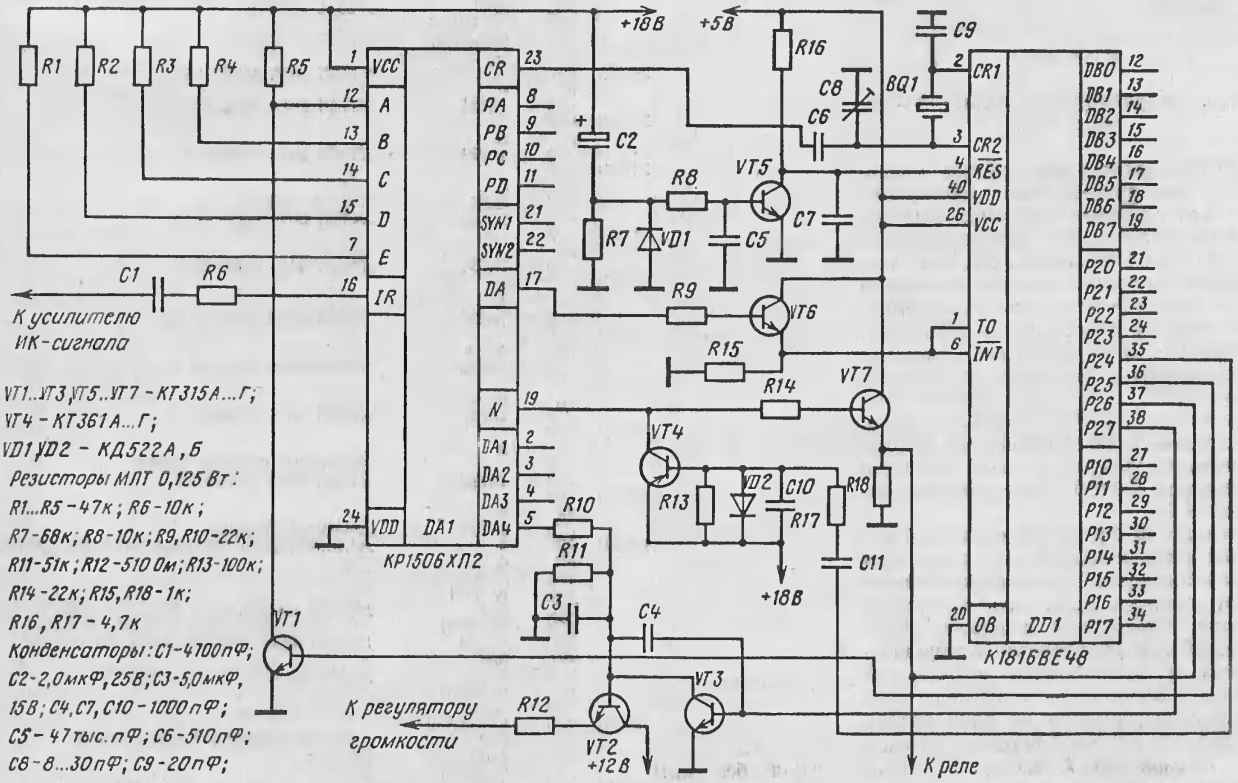


Рис. 2. Схема подключения БИС КР1506ХЛ2 к микроЭВМ К1816ВЕ48

вания сигнала RESET, программного включения-выключения питания и блокировки регулятора громкости для реализации «бесшумной настройки» в режиме автопоиска передатчика.

В этом случае тактовые сигналы вообще не используются, а коды команд через согласующий каскад на транзисторе VT6 подаются параллельно на два входа микроЭВМ — T0 и $\overline{\text{INT}}$. Все остальное реализуется программно — обработкой информации в параллельных ветвях программы, одна из которых принимает Лог. 1, а другая — Лог. 0 в соответствии с алгоритмом (рис. 3).

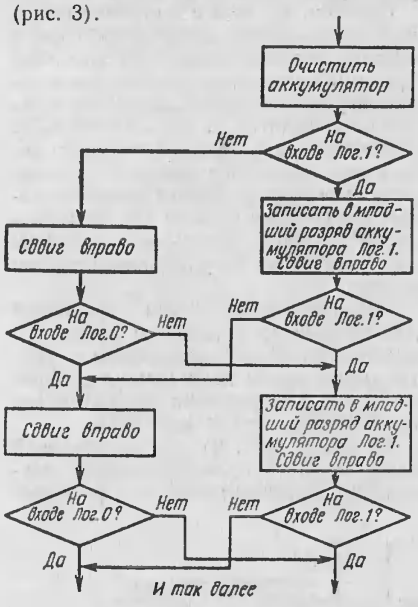


Рис. 3. Алгоритм приема последовательного кода

Использование двух входов микроЭВМ для приема кода обусловлено особенностью системы команд, так как вход $\overline{\text{INT}}$ может анализироваться только одной командой JNT. Для увеличения скорости приема необходим также анализ состояния Лог. 1 указанного порта и анализ порта $\overline{\text{INT}}$.

Обязательное условие — жесткая синхронизация тактовых генераторов обеих БИС. Благодаря высокой чувствительности БИС КР1506ХЛ2 в ведомом режиме для ее устойчивой работы достаточен уровень сигнала тактового генератора К1816ВЕ48 на выводе 3 (порядка 3 В).

Стартовый бит информационной посылки воспринимается как запрос прерывания основной программы, и микроЭВМ начинает выполнять программу приема информации с адреса 003Н. Каждый информационный бит анализируется и записывается в младший разряд аккумулятора с последующим сдвигом информации на один разряд вправо. Код адресной группы пересылается на хранение в регистр R6, а код команды — в R7.

На рис. 4 представлен листинг программы. Алгоритм программы достаточно прост и подробного описания не требует. Поясним только два момента: организацию защиты от повторного ввода команд и дальнейшую обработку команд в микроЭВМ.

При удержании в течение некоторого времени клавиши ввода команды на выходе БИС КР1506ХЛ2 код команды выводится многократно с периодом около 180 мс. Во многих случаях от многократного ввода команд необходимо защититься. В данном примере для защиты используется временная селекция с помощью встроенного таймера и одного из регистров ОЗУ.

В каждом цикле прерывания в регистр ОЗУ с адресом 21Н записывается константа, определяющая выдержку

```
                ПРИЕМ   ПОСЛЕДОВАТЕЛЬНОГО   КОДА
INT:    MOV     R7,A            ;СОХРАНИТЬ СОДЕРЖИМОЕ АККУМУЛЯТОРА В R7
        CLR     A               ;ОЧИСТИТЬ  АККУМУЛЯТОР
        JMP     INT00

INT00:  MOV     R6,A            ;ОЧИСТИТЬ R6
        NOP                     ;ПРИЕМ БИТА АДРЕСА 1
        JNT0    INT0A
        INC     A
        RR      A
        JNT0    INT0B           ;ПРИЕМ БИТА АДРЕСА 2
INT01:  INC     A
        RR      A
        JNT0    INT0C           ;ПРИЕМ БИТА АДРЕСА 3
INT02:  INC     A
        RR      A
        XCH     A,R6            ;ЗАПИСЬ АДРЕСА В R6, ОЧИСТКА АККУМУЛЯТОРА
        JNT0    INT0D           ;ПРИЕМ БИТА КОМАНДЫ 0
INT03:  INC     A
        RR      A
        JNT0    INT0E           ;ПРИЕМ БИТА КОМАНДЫ 1
INT04:  INC     A
        RR      A
        JNT0    INT0F           ;ПРИЕМ БИТА КОМАНДЫ 2
INT05:  INC     A
        RR      A
        JNT0    INT10           ;ПРИЕМ БИТА КОМАНДЫ 3
INT06:  INC     A
        RR      A
        JNT0    INT11           ;ПРИЕМ БИТА КОМАНДЫ 4
INT07:  INC     A
        RR      A
        NOP                     ;КОРРЕКЦИЯ ВРЕМЕНИ ПРИЕМА
        JNT0    INT12           ;ПРИЕМ БИТА КОМАНДЫ 5
INT08:  INC     A
        RR      A
        JNT0    INT13           ;АНАЛИЗ  СТОП-БИТА
INT09:  INC     A               ;ЗАПИСЬ КОДА В R7
        XCH     A,R7            ;ЗАПИСЬ КОНСТАНТЫ ДЛЯ ЗАЩИТЫ ПОВТОРНОГО
        SEL     RB1             ;ВВОДА КОМАНД
        MOV     R0,#21H
        MOV     @R0,#0BH
        RETR
INT0A:  NOP                     ;ПРИЕМ БИТА АДРЕСА 1
        RR      A
        JT0     INT01           ;ПРИЕМ БИТА АДРЕСА 2
INT0B:  NOP
        RR      A
        JT0     INT02           ;ПРИЕМ БИТА АДРЕСА 3
INT0C:  NOP
        RR      A
        XCH     A,R6            ;ЗАПИСЬ АДРЕСА В R6
        JT0     INT03           ;ПРИЕМ БИТА КОМАНДЫ 0
INT0D:  NOP
        RR      A
        JT0     INT04           ;ПРИЕМ БИТА КОМАНДЫ 1
INT0E:  NOP
        RR      A
        JT0     INT05           ;ПРИЕМ БИТА КОМАНДЫ 2
INT0F:  NOP
        RR      A
        JT0     INT06           ;ПРИЕМ БИТА КОМАНДЫ 3
INT10:  NOP
        RR      A
        JT0     INT07           ;ПРИЕМ БИТА КОМАНДЫ 4
INT11:  NOP
        RR      A
        NOP                     ;КОРРЕКЦИЯ ВРЕМЕНИ ПРИЕМА
        JT0     INT08           ;ПРИЕМ БИТА КОМАНДЫ 5
INT12:  NOP
        RR      A
        JT0     INT09           ;АНАЛИЗ СТОП-БИТА
INT13:  RR      A               ;ОКОНЧАТЕЛЬНОЕ ФОРМИРОВАНИЕ КОДА КОМАНДЫ
        RR      A
        CPL     A
        ANL     A,#3FH
        SEL     RB1             ;АНАЛИЗ СОДЕРЖИМОГО РЕГИСТРА ЗАЩИТЫ
        MOV     R0,#21H         ;ОТ ПОВТОРНОГО ВВОДА КОМАНД. ПЕРЕХОД В
        XCH     A,@R0           ;INT14, ЕСЛИ КОМАНДА ВВОДИТСЯ ВПЕРВЫЕ
        SEL     RB0
        XCH     A,R5            ;СБРОС ФЛАЖКА "ПЕРВЫЙ ПРИЕМ"
        ANL     A,#0FEH
        JMP     INT15
INT14:  XCH     A,@R0
        SEL     RB0             ;УСТАНОВКА ФЛАЖКА "ПЕРВЫЙ ПРИЕМ"
        XCH     A,R5
        ANL     A,#0FCH
INT15:  XCH     A,R5
        JMP     INT09
```

Рис. 4. Программа приема последовательного кода

```
                ОБРАБОТКА ПРЕРЫВАНИЯ ПО ТАЙМЕРУ

INT:    SEL     RB1             ;СОХРАНИТЬ АККУМУЛЯТОР
        MOV     R0,#20H
        MOV     @R0,A
        INC     R0              ;УСТАНОВИТЬ АДРЕС РЕГИСТРА R21
        MOV     A,@R0           ;ПРОВЕРИТЬ СОДЕРЖИМОЕ РЕГИСТРА
        JZ      INT0            ;ПЕРЕХОД, ЕСЛИ 0.
        DEC     A               ;ДЕКРЕМЕНТИРОВАТЬ И ЗАПИСАТЬ В R21
        MOV     @R0,A
INT0:   RETR
```

Рис. 5. Фрагмент ПО обработки прерываний по переполнению таймера

```
                АНАЛИЗ КОМАНДЫ

ML6:    MOV     A,R5            ;АНАЛИЗ ПРИЗНАКА КОМАНДЫ
        JB0     ML8             ;ЕСЛИ НЕТ, ПЕРЕХОД В ML8
        ORL     A,#03H          ;ОЧИСТКА ФЛАЖКОВ
        XCH     A,R5
        CPL     A
        JB1     ML7             ;АНАЛИЗ ПРИЗНАКА ПОВТОРНОГО ВВОДА КОМАНДЫ
        MOV     A,#04H          ;И КОДОВ КОМАНД, КОТОРЫЕ РАЗРЕШЕНЫ ДЛЯ
        XRL     A,R7            ;ОДНОКРАТНОГО ИЛИ ПОВТОРНОГО ВВОДА
        JZ      ML7             ;ПЕРЕХОД В ML7 ДЛЯ ВВОДА И В ML8 В СЛУЧАЕ
        DEC     A               ;ЗАПРЕЩЕНИЯ ВВОДА
        JZ      ML7
        XRL     A,#08H
        JNZ     ML8
ML7:    MOV     A,R7
        ____________
        ____________

ML8:    ADD     A,#00H
        ____________
        ____________
```

Рис. 6. Фрагмент ПО анализа кода команд

```
                ВЫКЛЮЧЕНИЕ

OFF:    ORL     P2,#20H         ;ВЫВОД  ЛОГ.1  В P25
        MOV     R0,#00H         ;ОРГАНИЗАЦИЯ ЗАДЕРЖКИ ДЛЯ ФОРМИРОВАНИЯ
        MOV     R2,#00H         ;ДЛИТЕЛЬНОСТИ ИМПУЛЬСА
OFF1:   DJNZ    R0,OFF1
        DJNZ    R2,OFF1
        ANL     P2,#0DFH        ;ВЫВОД  ЛОГ.0  В P25
        RET

                ВКЛЮЧЕНИЕ

ON:     ANL     P2,#0FFH        ;ВЫВОД  ЛОГ.0  В P24
        MOV     R0,#00H         ;ОРГАНИЗАЦИЯ ЗАДЕРЖКИ ДЛЯ ФОРМИРОВАНИЯ
ON1:    DJNZ    R0,ON1          ;ДЛИТЕЛЬНОСТИ ИМПУЛЬСА
        ORL     P2,#10H         ;ВЫВОД  ЛОГ.1  В P24
        RET
```

Рис. 7. Фрагменты ПО программного включения-выключения аппарата

времени (несколько больше, чем период повторения вывода кода команд). При выполнении подпрограммы приема кода команд содержимое указанного регистра анализируется. Если оно равно нулю, то в регистре R5 устанавливается флаг, сигнализирующий, что команда вводится первый раз. Если содержимое регистра не равно нулю, то устанавливается флаг повторного ввода команды.

В подпрограмме (рис. 5) обработки прерываний по переполнению таймера указанный регистр периодически декрементируется до его обнуления. Таким образом, если клавиша ввода команды удерживается в нажатом состоянии, то содержимое указанного регистра постоянно обновляется в подпрограмме приема команд, но, как только клавиша отпущена, регистр очищается (период времени очистки $T > > 180$ мс).

Для некоторых команд повторный ввод не только не вреден, но даже необходим (для команд ручной подстройки приемника). В основной программе анализируются коды команд, флажки в регистре R5 и (в зависимости от их содержимого) разрешается или запрещается выполнение повторных команд (рис. 6). На рис. 7 показаны фрагменты ПО, используемые для программного включения и выключения устройства.

На элементах C2, C5, R7, R8, R16, C7, VD1 и VT5 выполнена схема формирования сигнала RESET при включении питания. Элементы R10, R11, C3 образуют делитель напряжения и интегратор регулятора громкости. На транзисторе VT2 выполнен эмиттерный повторитель, а элементы VT3, C4 служат для запирания VT2 (выключения звука) во время выполнения подпрограммы автопоиска передатчиков.

Источники питания +18 В и +5 В должны обеспечивать питание схемы в «дежурном» режиме, а источник +12 В в целях экономии энергии включается в «активном» режиме.

Сигнал для управления реле включения аппарата в активный режим с вывода 19 КР1506ХЛ2 через повторитель на VT7 подается на соответствующее устройство. Этот же сигнал подается на порт P26 микроЭВМ для последующего анализа режима работы устройства. В активном режиме на выходе VT7 потенциал высокий (около 5 В).

Телефон 460-45-66, Москва

ЛИТЕРАТУРА

1. Крылов Е. И. Однокристальные микроЭВМ серий К1814, К1816, КР1820 // Микропроцессорные средства и системы.— 1985.— № 2.
2. Гутовец Н. И., Митрофанов А. В. и др.— Микропроцессоры и однокристальные микроЭВМ в системах управления сложной бытовой РЭА // Электронная промышленность.— 1982.— № 9.— С. 26—27.

Статья поступила 19.08.88.

ВСЕ О МИКРОЭВМ СМ1810, СМ1800

УДК 681.3.049

В. Д. Гуськов, А. Л. Еремеев, П. Б. Чучкалов

ЦЕНТРАЛЬНЫЙ ПРОЦЕССОР МИКРОЭВМ СМ1810

Вычислительным ядром всех комплексов СМ1810 является 16-разрядный модуль центрального процессора (МЦП), состоящий из 16-разрядного МП КМ1810ВМ86 [1] с развитой системой команд, устройств оперативной и постоянной памяти, таймера, контроллера прерываний, схем подключения внешних устройств через интерфейсы стык С2 и ИРПР-М. Конструктивно МЦП выполнен на двух платах типа Е2 размерами 220×233,4 мм.

Основные технические характеристики процессора

Тактовая частота, МГц	5
Объем памяти, Кбайт	
ОЗУ	256
РПЗУ	8...64
Максимальный объем адресуемой памяти, Мбайт	16
Число уровней прерывания	9
БИС ввода-вывода	КР580ВВ51А, КР580ВИ53, КР580ИК55, КР1810ВН59А
Системный интерфейс	И41
Внешние интерфейсы	Стык С2, ИРПР-М

Сигналы синхронизации для МП КМ1810ВМ86 и схем модуля формируются генератором тактовых импульсов КР1810ГФ84 частотой 5 МГц (см. рисунок). Процессор имеет локальную шину, обеспечивающую обращение МП к локальной памяти и внутренним портам ввода-вывода.

Управляющие сигналы локальной шины формируются контроллером шины КР1810ВГ88. Обмен информацией МЦП с другими модулями системы осуществляется в соответствии с интерфейсом И41 [2]. Двадцатичетырехразрядный адрес позволяет адресовать 16 Мбайт в системной памяти. Четыре старших разряда адреса управляются программно, поэтому доступ к системной памяти происходит страницами по 1 Мбайт. Число адресуемых портов ввода-вывода определяется архитектурой МП.

Для повышения вычислительной мощности в МЦП применяется арифметический сопроцессор КМ1810ВМ87, обеспечивающий выполнение арифметических операций с плавающей точкой, тригонометрических, логарифмических и степенных функций аппаратным способом. Это позволяет применять МЦП в системах обработки планово-экономической, текстовой информации и результатов научных экспериментов.

Процессор содержит четыре 28-контактные соединительные розетки, предназначенные для установки микросхем РПЗУ с УФ-стиранием. Диапазон адресов постоянной памяти при использовании микросхем различной информационной емкости приведен ниже.

Информационная емкость микросхем	Диапазон адресов в шестнадцатеричной системе счисления
2 К×8	FE000...FFFFF
4 К×8	FC000...FFFFF
8 К×8	F8000...FFFFF
16 К×8	F000...FFFFF

При включении питания МП выбирает первую команду из ячейки с адресом FFFF0, поэтому программа начальной инициализации и загрузки обычно размещается в РПЗУ. В МЦП предусмотрена схема формирования дополнительных тактов ожидания (от 1 до 3) при обращении МП к постоянной памяти, допускающая использование элементов памяти с различным временем выборки.

Оперативное ЗУ МЦП, построенное на микросхемах К565РУ5Г, имеет доступ со стороны локальной шины интерфейса И41 и осуществляет обмен 8- и 16-разрядной информацией. Схема регенерации памяти реализована на микросхеме КМ1810ВТ3.

При доступе со стороны локальной шины памяти присваиваются адреса 00000...3FFFF. Со стороны интерфейса И41 ОЗУ может занимать любую область 16 Мбайт адресного пространства с шагом 16 Кбайт. Предусмотрена возможность отключения доступа к памяти со стороны локальной шины по адресам 30000...3FFFF.

Оперативное ЗУ МЦП располагает схемами контроля 16-разрядных данных по коду Хэмминга и портами диагностики. Схема контроля, выполненная на микросхеме К555ВЖ1, позволяет обнаруживать одиночные и двойные ошибки, формировать соответствующие признаки ошибок и сигналы прерывания, которые могут быть переданы в МП. Если ошибка одиночная, то информация корректируется.

С помощью портов диагностики можно задать следующие режимы работы ОЗУ МЦП:

разрешение выдачи прерывания по любой или только двойной ошибке;

запрет записи информации в контрольные разряды накопителя в режиме записи;

запрет коррекции одиночной ошибки в операциях чтения;

запись в порт диагностики последней ошибки или контрольных разрядов при каждом чтении из памяти или вычисленных контрольных разрядов при каждой записи в память.

Используя режимы диагностики, можно программным путем протестировать все элементы памяти, включая контрольные разряды, и определить неисправность с точностью до элемента накопителя.

Схема прерывания МЦП позволяет обслуживать восемь маскируемых запросов прерывания и один немаскируемый. Последний имеет самый старший приоритет и используется главным образом для обработки аварийных ситуаций, например при выходе из строя источника питания. Предусмотрена возможность программной блокировки немаскируемого запроса прерывания. Схема прерывания реализована на контроллере прерываний КР1810ВН59А. Число запросов прерывания можно увеличить до 64 путем подключения к МЦП подчиненных контроллеров прерывания КР1810ВН59А, расположенных в других модулях.

Все запросы прерываний как от внутренних схем модуля, так и с интерфейса И41 поступают на наборное поле. С его помощью запросы подаются на любой уровень приоритета; несколько запросов собирается с присвоением одного уровня приоритета. Наборное поле запросов прерывания позволяет изменять схему прерывания МЦП для работы с различными ОС.

Есть в МЦП и программируемый параллельный канал ввода-вывода, выполненный на микросхеме КР580ИК55, позволяющий подключать устройства, выходящие на интерфейс ИРПР-М (центроникс). Программируемый последовательный канал ввода-вывода, реализованный на микросхеме КР580ВВ51А, обеспечивает подключение терминалов с интерфейсом типа стык С2. Этот канал разрешает обмен данными со скоростью 110...9600 Бод.

В состав МЦП входит также программируемый таймер КР580ВИ53. Счетчик 2 таймера используется для синхронизации микросхемы КР580ВВ51А; счетчики 0 и 1 — для отсчета задаваемых интервалов времени в диапазоне 1,6... 58,3 мс, а также для формирования сигналов прерывания в реальном масштабе времени.

Регистр управления МЦП программно формирует два сигнала, управляющих входами GATE счетчиков 0 и 1 программируемого таймера КР580ВИ53; сигнал блокировки немаскируемого запроса прерывания; сигнал для монопольного захвата внутренней оперативной памяти или интерфейса И41; два независимых сигнала запроса прерывания, передаваемых на линии интерфейса И41. С помощью регистра управления можно программно приостанавливать счетчики таймера, задающие интервалы времени, блокировать или разрешать обработку на МП немаскируемого запроса прерывания, задавать монопольный режим работы МЦП с интерфейсом И41 и внутренней оперативной памятью и формировать два независимых сигнала запроса межпроцессорных прерываний.

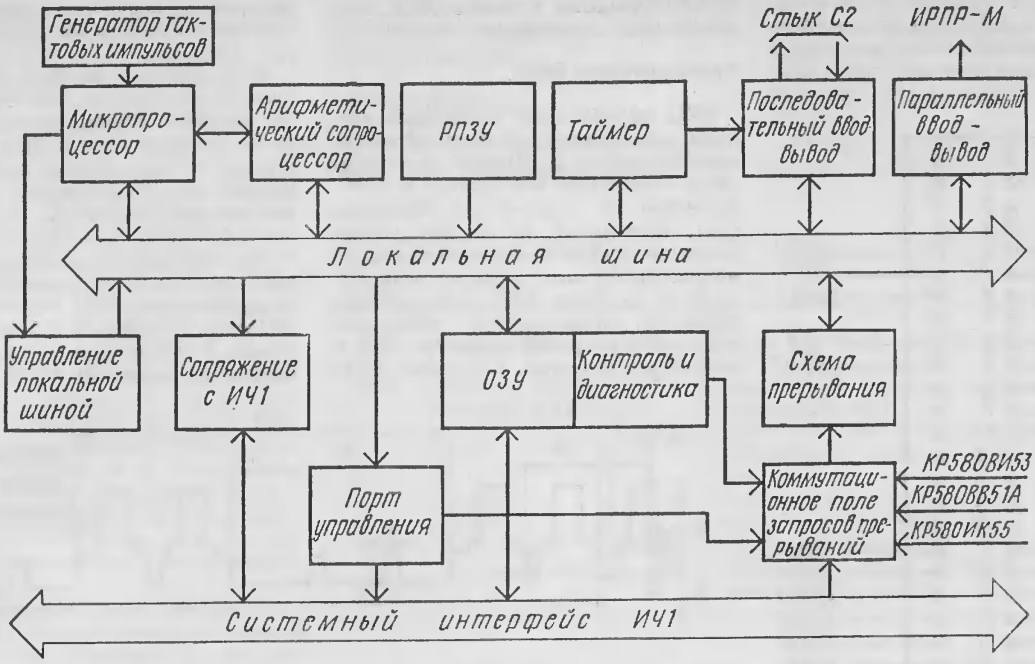
Модуль процессора выполняется в нескольких вариантах, рассчитанных на применение в комплексах СМ1810 с различными ОС. Варианты исполнения МЦП отличаются содержимым РПЗУ, распределением адресного пространства ОЗУ и типами запросов прерываний, поступающих через наборное поле к контроллеру прерываний КР1810ВН59А.

Во всех вариантах исполнения РПЗУ МЦП объемом 32 Кбайт построено на микросхемах К573РФ6 (8 К×8) и расположено по адресам F8000...FFFFF. В РПЗУ содержатся программы для работы с конкретной ОС. Например, для работы под управлением ОС ДОС1810 локальная память МЦП разделяется на две области. Одна объемом 64 Кбайт доступна для обращения только со стороны интерфейса И41 в диапазоне адресов 000000...00FFFF, другая объемом 192 Кбайт предназначена для обмена информацией только с МП МЦП по адресам 00000...2FFFF. Последняя имеет окно емкостью 2 Кбайт в диапазоне адресов 04000... 047FF для обращения к интерфейсной памяти. Окно используется для организации обмена информацией между двумя процессорами комплекса СМ1810, работающего с ДОС1810.

Вторым процессором комплекса является 8-разрядный МЦП, построенный на основе МП КР580ИК80А, имеющий в своем составе двухвходовое ОЗУ с адресами доступа со стороны интерфейса И41 04000...047FF, совпадающими с адресами окна 16-разрядного МЦП.

На входы контроллера прерываний КР1810ВИ59А для варианта исполнения МЦП, работающего под управлением ДОС1810, поступают следующие запросы прерывания:

Уровень 0 — от счетчика 1 таймера КР580ВИ53
Уровень 1 — интерфейсный сигнал INTI/, запрос прерывания от кнопки с пульта управления комплекса
Уровень 2 — от счетчика 0 таймера КР580ВИ53
Уровни 3,4 — не используются
Уровень 5 — сигнал межпроцессорных прерываний с регистра управления МЦП



Уровень 6 — не используется
Уровень 7 — интерфейсный сигнал INT7/, запрос прерывания от модуля связи через последовательный интерфейс с устройством печати

Для ОС БОС1810 локальная память МЦП полностью доступна со стороны МП и интерфейса И41 по адресам 000000...03FFFF. На входы контроллера прерываний КР1810ВН59А в варианте исполнения МЦП, работающего под управлением БОС1810, поступают следующие запросы прерываний:

Уровень 0 — от сопроцессора КМ1810ВМ87
Уровень 1 — интерфейсный сигнал INT1/, запрос прерывания от кнопки с пульта управления комплекса
Уровень 2 — от счетчика 0 таймера КР580ВИ53
Уровень 3 — интерфейсный сигнал INT3/, запрос прерывания от контроллера управления гибкими дисками
Уровень 4 — от КР580ИК55, работающего с устройством печати
Уровень 5 — интерфейсный сигнал INT5/, запрос прерывания от контроллера управления гибкими и жесткими дисками
Уровень 6 — от канала ввода КР580ВВ51А, работающего с клавиатурой дисплея
Уровень 7 — от канала вывода КР580ВВ51А, работающего с экраном дисплея

Процессор с такой архитектурой может использоваться для создания простейших систем управления оборудованием в виде локальных управляющих контроллеров, мультипроцессорных систем реального времени, сложных распределенных сетевых вычислительных систем управления крупными комплексами технологических установок и агрегатов и систем, выполняющих экономические, научные и инженерные расчеты.

117812, Москва, ул. Вавилова, 24, ИНЭУМ; тел. 455-56-41

ЛИТЕРАТУРА

1. Кобылинский А. В, Москалевский А. И., Темченко В. А. Однокристальный высокопроизводительный 16-разрядный микропроцессор КМ1810ВМ86 // Микропроцессорные средства и системы.— 1986.— № 1, 2.
2. Отраслевой стандарт. Система малых электронных вычислительных машин. Интерфейс И41. Технические требования. ОСТ 25.969-83.

Статья поступила 15.06.88

УДК 681.3

Г. М. Бобков

ВИДЕОКОНТРОЛЛЕР СМ1810.7006

Видеоконтроллер цветной (ВКЦ) используется в ЭВМ СМ1810 для отображения символьной и графической информации на экране цветного и черно-белого индикаторов. Под управлением малой дисковой операционной системы МДОС1810 (OS PC, версия 3.1) ВКЦ программно совместим с адаптером CGA (Color Graphics Adapter).

ВКЦ выполнен на одной плате Е2 размерами 248×245×16 мм и установлен в монтажном блоке СМ1810. Связь с системной шиной осуществляется через разъем СНП 59-96/94. Потребляемый ток составляет не более 3,52 А. Предусмотрены разъемы R, G, B, S для подключения посредством коаксиальных кабелей цветного индикатора с входным сопротивлением 75 Ом и разъем для светового пера и цветного индикатора, имеющего входы ТТЛ. Черно-белый индикатор подключается к разъему S. При этом на плате ВКЦ должна быть перепаяна соответствующая перемычка.

Системная шина СМ1810 заведена на два разъема монтажного блока — основной и дополнительный. ВКЦ использует цепи основного разъема, номенклатура и назначение которых при-

ведены в табл. 1. Разводка цепей шины дана на рис. 1. ВКЦ при любом обращении к внутренней памяти выставляет на внешние линии сигналы запрета ОЗУ и ПЗУ. При использовании сигнала BHEN/обращение к памяти ВКЦ осуществляется двухбайтным словом.

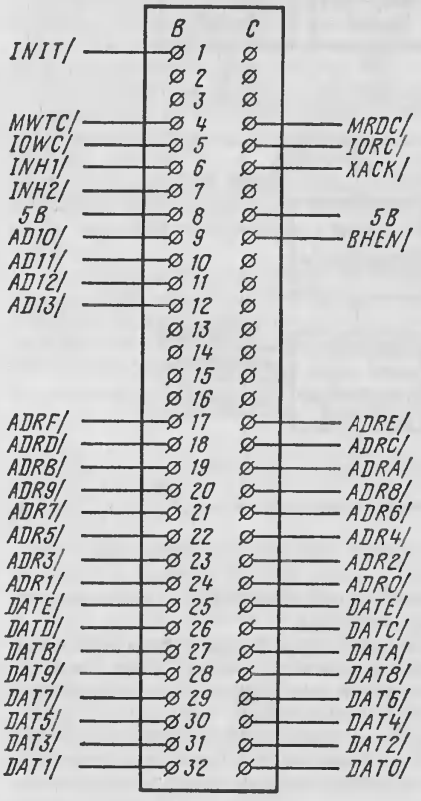


Рис. 1. Разводка цепей магистрали СМ1810 на разъем ВКЦ;
на разъеме не указаны контакты ряда А: А1...А4, А18...А27 — земля; А6...А8 — напряжение питания 5 В

Таблица 1

Линия	Назначение
DA0/...DAF/	Двунаправленные линии данных
ADR0/...ADRF/	Линии адреса
AD10/...AD13/	Старшие разряды линии адреса
INIT/	Начальная установка
MWTC/	Запись в память
MRDC/	Чтение из памяти
IOWC/	Запись в порт
IORC/	Чтение из порта
INH1/	Запрет ОЗУ
INH2/	Запрет ПЗУ
BHEN/	Разрешение приема старшего байта данных
XACK/	Подтверждение передачи (ответ)

Примечание. Черта у наименования сигнала означает, что активный уровень сигнала — низкий

Характеристики ВКЦ

ВКЦ вырабатывает следующие сигналы: синхросмесь для строчной и кадровой разверток S, «Видео» (в синхросмесь «замешана» информация для отображения на черно-белых индикаторах), сигналы R, G, B для управления цветностью отображаемой информации. Кроме того, предусмотрены выходы с уровнями ТТЛ для сигналов цветности, интенсивности, импульсов строчной и кадровой разверток. Вид и параметры выходных импульсов ВКЦ при использовании имеющегося математического обеспечения приведены на рис. 2 и в табл. 2.

ВКЦ построен на базе программируемого контроллера ЭЛТ СМ607 (аналог МС6845), поэтому временные параметры разверток могут быть изменены пользователем с учетом применяемого индикатора. Рекомендуемые индикаторы: цветные М32Ц11/2, М42Ц11/2 и черно-белый А543-13. Обычные цветные телевизоры требуют доработки до RGB-индикатора и не обеспечивают высокого разрешения. ВКЦ позволяет работать в символьном и графическом режимах. В символьном информация отображается форматом 40×25 строк на

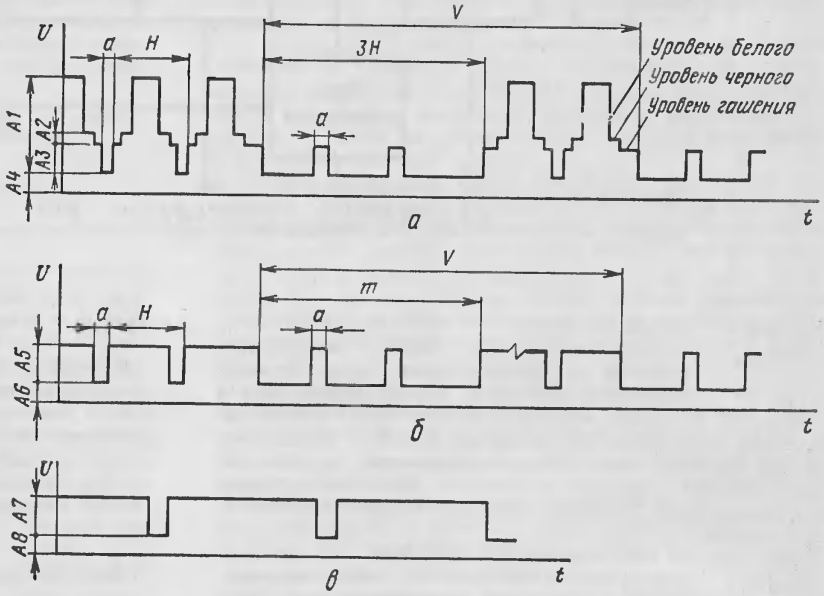


Рис. 2. Параметры выходных сигналов ВКЦ:
а — сигнал на выходе для черно-белых индикаторов; *б* — синхронизирующие импульсы для цветных индикаторов; *в* — видеоимпульсы для цветных индикаторов

Таблица 2

Параметр, обозначение, единица измерения	Норма
Частота кадровой развертки, V, Гц	60±2
Период строчной развертки, H, мкс	64±0,1
Длительность строчного синхроимпульса и импульсов врезок в кадровом синхроимпульсе, a, мкс	5,0±0,5
Длительность кадрового синхроимпульса, m	3H
Амплитуда выходного импульса для черно-белого индикатора, A1, В	1,0±0,1
Защитный интервал между уровнем черного и уровнем гашения, A2, В	0,33...0,045
Уровень синхронизирующих импульсов, A3, В	0,3...0,05
Постоянное напряжение подпора выходного импульса для черно-белого индикатора, A4, В	0...0,2
Амплитуда выходного импульса синхронизации для цветных индикаторов, A5, В	2,0...4,0
Постоянное напряжение подпора выходного импульса синхронизации для цветных индикаторов, A6, В	0...0,7
Амплитуда видеоимпульса для цветных индикаторов, A7, В	0...1
Постоянное напряжение подпора видеоимпульса для цветных индикаторов, A8, В	0...3

экране среднего и 80×25 строк на экране высокого разрешения. Матрица знакоместа составляет 8×8, матрица символа — 7×7 точек.

Атрибуты выделения в черно-белом режиме: нормальная и повышенная яркость, мерцание, инверсия. В режиме цвета — 16 цветов символов, восемь цветов фона, мерцание. Объем памяти генератора знаков позволяет отображать 256 различных символов.

Коды символов знакогенератора соответствуют так называемой «альтернативной» таблице символов. Первая часть таблицы (коды 00...7F в шестнадцатеричном представлении) содержит символы латинского алфавита, цифры и вспомогательные знаки. Во второй части расположены большие и малые русские буквы в алфавитном порядке с кодами 80H...AFH и E0H... EFH. Здесь достигается полная совместимость с ПЭВМ типа IBM PC по символам псевдографики (коды B0H... DFH).

Буферная оперативная память ВКЦ доступна со стороны системной оперативной памяти, начиная с адреса B8000H. В режиме 40×25 используется 2 Кбайт ОЗУ, что позволяет хранить в буфере восемь страниц текста; в режиме 80×25—4 Кбайт ОЗУ и четыре страницы текста. В обоих режимах может быть установлен один из 16 цветов бордюра (обрамление текста).

Графика воспроизводится в режимах среднего и высокого разрешения. В режиме среднего разрешения выводится 320×200 точек. Цвет каждой точки задается двумя битами, позволяющими видеть ее в четырех цветах при двух цветовых палитрах. В режиме высокого разрешения отображается монохромная графика 640×200 точек в одном из 16 цветов. В обоих графических режимах используется 16 Кбайт памяти.

Кроме того, возможен вывод графики в режиме низкого разрешения 160×100 точек, когда каждая программируемая точка воспроизводится на экране индикатора двойными точками по вертикали и горизонтали одним из 16 цветов. Режим занимает 8 Кбайт памяти.

Важным отличием графических режимов от символьных является отсутствие маркера при работе с текстами. Прикладные программы, использующие графический формат, формируют «собственные маркеры».

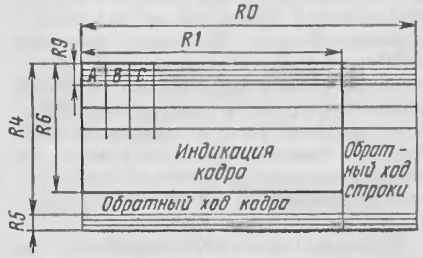


Рис. 3. Формат развертки по кадру

Работа с ВКЦ

С точки зрения системного программиста, ВКЦ представляет собой набор регистров, определяющих режим работы и буферную оперативную память емкостью 16 Кбайт. Перечень регистров и выполняемые ими функции приведены в табл. 3.

Контроллер ЭЛТ содержит 18 внутренних регистров, через которые зада-

Таблица 3

Шестнадцатеричный адрес	Регистр	Команда	Результат выполнения команды
3D4	Индексный	Запись	Запись адреса внутреннего регистра контроллера ЭЛТ
3D5	Данных контроллера ЭЛТ	»	Запись информации во внутренний регистр контроллера ЭЛТ
		Чтение	Чтение старших и младших байтов адреса отображения, адресов маркера и светового пера
3D8	Режима	Запись	Установка одного из режимов ВКЦ
3D9	Цвета	»	Задание цвета отображения
3DA	Состояния	Чтение	Чтение состояния ВКЦ
3DC	Светового пера	Запись	Разрешение работы светового пера
3DB	»	»	Маскирование светового пера

Таблица 4

Регистр	Адрес	Число	Команда	Функция регистра
R0	00	8	IOWC/	Длительность растровой строки
R1	01	8	»	Индикация по строке
R2	02	8	»	Время установки HS
R3	03	4	»	Длительность HS
R4	04	7	»	Число строк символов в кадре
R5	05	5	»	Число строк для успокоения кадра (задний уступ по кадру)
R6	06	7	»	Индикация по кадру
R7	07	7	»	Время установки VS
R8	08	2	»	Вид развертки
R9	09	5	»	Число растровых строк в символе
R10	0A	7	»	Начало маркера (положение маркера в матрице символа)
R11	0B	5	»	Конец маркера
R12	0C	6	IOWC/ IORC/	Стартовый адрес изображения, старший байт
R13	0D	8	»	Стартовый адрес изображения, младший байт
R14	0E	6	»	Адрес маркера, старший байт
R15	0F	8	»	Адрес маркера, младший байт
R16	10	6	»	Адрес светового пера, старший байт
R17	11	8	»	Адрес светового пера, младший байт

Таблица 5

Режим	Отображение	Формат	Разряды регистра					
			5	4	3	2	1	0
Символьный	Черно-белое	40×25	1	0	1	1	0	0
	Цветное	»	1	0	1	0	0	0
	Черно-белое	80×25	1	0	1	1	0	1
	Цветное	»	1	0	1	0	0	1
Графический	Черно-белое	320×200	X	0	1	1	1	0
	Цветное	»	X	0	1	0	1	0
	Черно-белое	640×200	X	1	1	1	1	0

Примечание. 5 — разрешение мерцания; 4 — графика 640×200; 3 — разрешение индикации; 2 — установка черно-белого режима; 1 — графика 320×200; 0 — символы 80×25; X — состояние любое

Рис. 4. Формат байта атрибутов

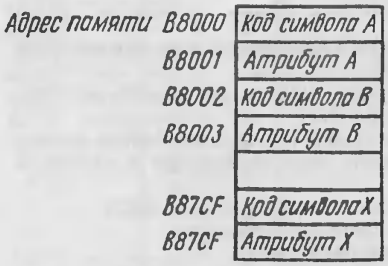


Рис. 5. Распределение памяти в символьном режиме

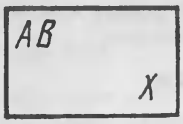


Рис. 6. Расположение символов на экране

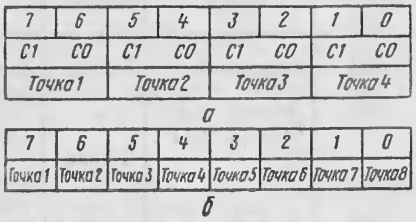


Рис. 7. Формат данных в графических режимах:
а — 320×200 точек; *б* — 640×200 точек

Таблица 6

Разряд	Атрибут	Функция
0	Синий	Цвет и яркость бордюра в символьном режиме
1	Зеленый	Цвет и яркость фона в режиме 320×200 точек
2	Красный	Монохромный цвет и яркость в режиме 640×200 точек
3	Интенсивность	Повышенная яркость
4	Палитра	Яркость графики в режиме 320×200 точек
5		Выбор палитры в режиме 320×200 точек

ются временные параметры отображения. Номер регистра записывается в байте по адресу 3D4, а информация в него — по адресу 3D5. Перечень внутренних регистров приведен в табл. 4, формат и описание разрядов регистра режима даны в табл. 5 и на рис. 3. Значения разрядов регистров цвета и формат байта атрибута представлены соответственно в табл. 6 и на рис. 4.

Цвета первой палитры: цвет заданного фона, зеленый, красный, коричневый; второй палитры: цвет заданного фона, голубой, пурпурный, белый; цвет фона — один из 16; значения разрядов регистра состояния: 0— разрешение индикации; 1 — триггер светового пера установлен; 2 — разрешение работы со световым пером; 3 — обратный ход кадра.

Каждый знак в символьном режиме записывается двумя байтами: четным — код символа, нечетным — атрибут. Палитра цветов в зависимости от атрибута показана в табл. 7, распределение памяти в символьном режиме — на рис. 5, расположение символов на экране — на рис. 6.

В графическом режиме 320×200 точек байт информации содержит данные о четырех точках графики (рис. 7, *а*), в режиме 640×200 — о восьми (рис. 7, *б*); цвет точки определяется

Таблица 7

Атрибут				Цвет
R	G	B	I	
0	0	0	0	Черный
0	0	1	0	Синий
0	1	0	0	Зеленый
0	1	1	0	Голубой
1	0	0	0	Красный
1	0	1	0	Пурпурный
1	1	0	0	Коричневый
1	1	1	0	Белый
0	0	0	1	Серый
0	0	1	1	Ярко-синий
0	1	0	1	Ярко-зеленый
0	1	1	1	Ярко-голубой
1	0	0	1	Ярко-красный
1	0	1	1	Ярко-пурпурный
1	1	0	1	Желтый
1	1	1	1	Ярко-белый

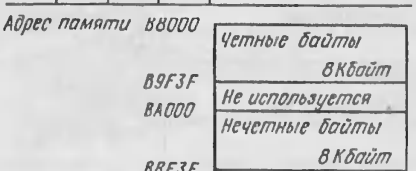


Рис. 8. Распределение памяти в графическом режиме

Таблица 8

С1	С0	Функция
0	0	Цвет точки не задан; фон определяется регистром цвета
0	1	Выбор первого цвета
1	0	Выбор второго цвета
1	1	Выбор третьего цвета

двумя разрядами (табл. 8). Распределение памяти в графическом режиме представлено на рис. 8. По адресу B8000 первые точки высвечиваются в левом верхнем углу экрана.

Принцип работы ВКЦ

Структурная схема ВКЦ представлена на рис. 9. ВКЦ работает в режимах обращения к регистрам, видеопамяти БД и в режиме регенерации. При обращении к регистрам инициализируется контроллер ЭЛТ КД, задается режим работы, цвет изображения, опрашивается регистр состояния. Обмен информацией происходит 8-битными словами. ВКЦ после исполнения команды обращения к регистрам выставляет сигнал ответа ХАСК/.

В зависимости от состояния сигнала BHEN/ обращение к видеопамяти БД со стороны интерфейса производится 8- или 16-разрядным словом: при BHEN/=0 — 8-разрядным; BHEN/==1 — 16-разрядным.

При обращении к БД 16-разрядным словом в символьном режиме младший байт содержит код символа, старший байт — код атрибута; в графическом режиме 320×200 точек младший байт — код четырех точек четного, старший байт — четырех точек нечетного адресов; в графическом режиме 640×200 точек младший байт — код восьми точек четного, старший байт — восьми точек нечетного адресов.

Байты последовательно записываются в БД или считываются из него. При этом автоматически модифицируется адрес БД для второго байта. Адресуется БД посредством регистра адреса процессора РАП или видеоконтроллера ЭЛТ РАВ. После исполнения команды обращения к видеопамяти ВКЦ выставляет сигнал ХАСК/.

В режиме регенерации считывание информации из видеопамяти и формирование сигналов для отображения на индикаторах производятся синхронно с разверткой луча. КД через регистры РАВ последовательно опрашивает ячейки видеопамяти БД. Четный байт БД заносится в регистр кода символа РКС, нечетный — в регистр атрибута РАТ. Выходы этих регистров нагружены на память генератора символов ПГС, регистры графики РГ1, РГ2 и входы мультиплексора МХ.

Информация с РКС заносится в ПГС: четные биты — в старшие разряды РГ1, нечетные — в старшие разряды РГ2. Информация с РАТ посту-

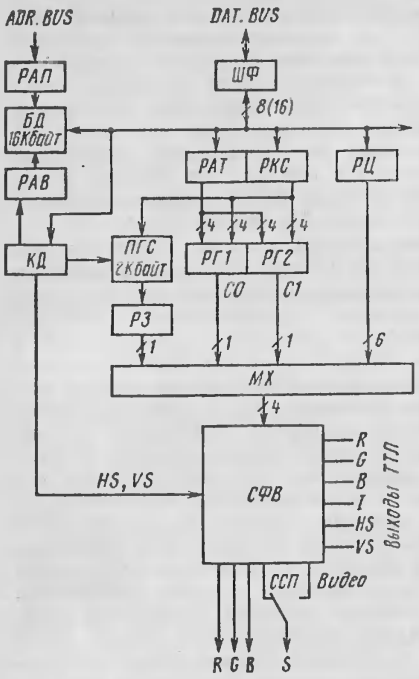


Рис. 9. Структурная схема ВКЦ: РАП — регистр адреса процессора; БД — буфер дисплея; РАВ — регистр адреса видеоконтроллера; КД — контроллер дисплея; ПГС — память генератора символов; РЗ — регистр знака; ШФ — шинный формирователь данных; РАТ — регистр атрибутов; РКС — регистр кода символа; РГ1 — регистр графики 1; РГ2 — регистр графики 2; РЦ — регистр цвета; МХ — мультиплексор; СФВ — схема формирования видеосигнала; HS, VS — сигналы строчной и кадровой синхронизации, ССП — аналоговый выход синхросмеси

пает на мультиплексор МХ: четные биты — в младшие разряды РГ1, нечетные — в младшие разряды РГ2. Одновременно информация с ПГС записывается в регистр знака РЗ. Параллельный код из сдвиговых регистров РГ1, РГ2, РГ3 выдвигается старшими разрядами вперед, управляя МХ, формирующим цвета видеокода. Информация в символьном режиме отображается последовательностью светящихся точек. Изображение делится на символы, представленные в виде прямоугольных матриц точек.

Начертание символов определяется содержанием ячеек памяти знакогенератора ПГС; каждой единице соответствует светящаяся точка на экране индикатора, нулю — заданный в РЦ цвет фона.

В режиме 320×200 точек за один такт сдвига информации из РГ1 и РГ2 на МХ анализируется состояние выходов С0 и С1 — цвет точки и заданная палитра; в режиме 640×200 точек — состояние С0 (первая половина такта) и состояние С1 (вторая половина такта). На время действия единицы на выходах С0 и С1 МХ формирует цвет точки, заданный РЦ. Таким образом за четыре такта сдвигов высвечивается восемь точек из РКС, за следующие четыре такта — восемь точек из РАТ.

Выходы с мультиплексоров МХ подключены к выходному регистру формирования цвета, расположенному в схеме формирования видеосигнала СФВ. Информация о цвете точки синхронно с разверткой поступает на выходные усилители. В СФВ формируется синхросмесь, заполняется врезками кадровый синхроимпульс, устанавливается видеосигнал для цветного и черно-белого индикаторов.

Программное обеспечение

Аппаратная совместимость ВКЦ с видеоконтроллером CGA во многом определяет совместимость графического программного обеспечения, наработанного для видеоконтроллера CGA. В рамках МДОС1810 управление видеоизображением на самом нижнем уровне осуществляется через BIOS с использованием программного прерывания INT 16 (INT 10H): устанавливаются текстовые и графические режимы, тип маркера, устанавливаются и читаются координаты маркера, организуется «прокрутка» дисплейного окна вверх или вниз, определяются оттенки цветов (цветовой палитры), формируются страницы, записывается и читается символ или точка по заданным координатам и т. д.

Другой вариант управления ВКЦ на уровне МДОС1810 — с использованием управляющих ESC-последовательностей, обрабатываемых драйвером ANSI.SYS. Драйвер позволяет устанавливать всевозможные режимы ВКЦ, выдавать на экран сообщение, изменять цвет и позицию маркера, очищать участки экрана.

Для тестирования видеоконтроллера ВКЦ в рамках ТОС1810 разработан тест проверки работоспособности ВКЦ. При прогоне теста выполняется проверка видеопамяти, командных и статусных регистров, установки режимов, размеров и скорости мерцания маркера, вывода текстовой информации в текстовом и графическом режимах, изменения цвета и палитры.

Телефон 455-57-41, Москва

Статья поступила 13.04.88

УДК 681.324

В. И. Глухов

МОДУЛЬ СИСТЕМНОГО КОНТРОЛЯ МИКРОЭВМ СМ1810

Для организации внутрисистемного обмена информацией в микроЭВМ СМ1810 использован интерфейс И41 (параллельная магистраль с асинхронным обменом данными) [1]. Он позволяет создавать мультипроцессорные вычислительные системы открытого типа. Устройства, подключенные к магистрали И41, могут быть задатчиками (процессоры, контроллеры периферийных устройств с прямым доступом в память) или исполнителями (устройства оперативной или постоянной памяти, контроллеры внешних устройств, не имеющие прямого доступа в память) в зависимости от своего функционального назначения. Задатчики управляют интерфейсной магистралью и формируют управляющие сигналы, а исполнители работают под управлением задатчиков.

Интерфейс И41 допускает существование на общей магистрали нескольких задатчиков и исполнителей и определяет порядок их взаимодействия.

Операции, выполняемые при обмене информацией между задатчиками и исполнителями, следующие: арбитраж задатчиков, захват магистрали наиболее приоритетным задатчиком, обмен данными (операции типа «чтение» или «запись»), прерывание и восстановление вычислительного процесса после аварии в сети переменного тока. В микроЭВМ СМ1810 эти операции выполняются модулем системного контроля СМ1810.2005, который содержит: арбитр задатчиков интерфейса И41; схемы рестарта; блок сопряжения с органами управления и индикации микроЭВМ СМ1810; нагрузочные сопротивления линий интерфейса И41. Арбитр позволяет организовать арбитраж задатчиков тремя способами: последовательным, параллельным и циклическим.

В общем виде процесс арбитража выглядит следующим образом [2]. При возникновении в задатчике сигнала, являющегося запросом на доступ с магистрали интерфейса И41, формируется системный запрос на доступ к магистрали. Арбитр принимает эти запросы, выделяет наиболее приоритетный и отвечает сигналом разрешения на доступ к интерфейсной магистрали. Далее задатчик «захватывает» магистраль, берет на себя управление ею и приступает к обмену информацией с устройством-исполнителем. *Выдача запросов в арбитр и прием сигналов разрешения на доступ от арбитража выполняются задатчиками синхронно по срезу синхроимпульсов BCLK/.*

Последовательный арбитраж задатчики выполняют самостоятельно, получая от арбитра только синхроимпульсы BCLK/: на вход разрешения доступа к интерфейсной магистрали (BPRN/) задатчика с наиболее высоким приоритетом постоянно подается разрешающий потенциал; на вход BPRN/задатчика с менее высоким приоритетом — сигнал с выхода разрешения BPRO/. Так связаны все

задатчики системы. Задатчик, получивший право доступа к магистрали, при захвате последней выставляет на своем выходе BPRO/ сигнал запрета на доступ к магистрали для всех задатчиков с менее высоким приоритетом. Если в задатчике, имеющем разрешение на доступ к магистрали, не сформировался внутренний запрос, то этот задатчик поддерживает на своем выходе BPRO/ сигнал разрешения, принимаемый следующим по уровню приоритета задатчиком.

Реализация последовательного арбитража в мультимикропроцессорной системе требует введения в каждый задатчик дополнительного оборудования. Приоритет задатчика зависит от того, какое физическое место он занимает в системе. Изменение приоритета достигается перестановкой задатчиков в конструктиве. Главный недостаток последовательного арбитража — невозможность включения в систему большого числа задатчиков, так как по требованиям интерфейса И41 их последовательное переключение должно происходить в течение одного такта синхронизации интерфейсной магистрали BCLK/:

$$T_{BCLK/} > \sum_{i=1}^{n} t_i,$$

где $T_{BCLK/}$ — период синхронизации; t_i — время распространения сигнала от входа BPRN/ до выхода BPRO/ в задатчике; n — число задатчиков в системе (как правило, три-четыре).

Параллельный арбитраж задатчиков выполняется специальным арбитром, на входы которого поступают сигналы запроса на доступ к магистрали. Сигналы запроса BREQ/ формируются задатчиками по внутреннему запросу на доступ синхронно со срезом импульса BCLK/. Параллельный арбитр представляет собой комбинацию из приоритетного шифратора и дешифратора, число выходов которого равно числу входов приоритетного шифратора. Приоритетный шифратор принимает сигналы запроса от задатчиков, пришедшие на его входы в данный период синхронизации, формирует на своих выходах код, соответствующий номеру входа с высшим приоритетом. Этот код превращается дешифратором в позиционный сигнал, который подается в задатчик, выставивший наиболее приоритетный запрос в качестве сигнала разрешения BPRN/. При параллельном арбитраже приоритет задатчика зависит от того, на какой вход арбитра поступает его запрос. Параллельный арбитраж — это наиболее быстрый метод арбитража. Задержка появления разрешающего сигнала относительно сигнала запроса определяется только временем срабатывания приоритетного шифратора и дешифратора.

Циклический арбитраж задатчиков аналогичен параллельному. Различие только в реализации схемы арбитра. Циклический арбитр автоматически изменяет приоритеты задатчиков после выполнения каждой операции информационного обмена. Каждый задатчик в некоторый период времени приобретает высший приоритет, который затем уступит задатчику, подключенному на соседний вход арбитра.

Циклический арбитр содержит: регистр, запоминающий запросы, пришедшие к моменту освобождения магистрали задатчиком, получившим доступ в предыдущем цикле арбитража; счетчик циклов арбитража, переключаемый по снятию сигнала BUSY/ задатчиком, освобождающим магистраль; дешифратор, формирующий сигнал разрешения BPRN/; логическую схему, управляющую фиксацией запросов в регистре. Число состояний счетчика циклов равно числу входов запроса на доступ к магистрали. Логическая схема «защелкивает» регистр по условию: «есть запросы на доступ к магистрали BREQ/ и магистраль свободна». С учетом текущего состояния счетчика циклов арбитража дешифратор выделяет наиболее приоритетный запрос и формирует сигнал разрешения BPRN/. Задатчик, имеющий в текущем цикле арбитража наиболее приоритетный запрос, захватывает магистраль и выполняет операцию обмена информацией с исполнителем. Освобождая магистраль в конце операции, задатчик снимает сигнал BUSY/, что вызывает изменение состояния счетчика циклов. Если при этом существуют запросы от других задатчиков, то описанная выше процедура повторяется с учетом изменившихся приоритетов задатчиков. Таким образом, каждый задатчик будет циклически изменять свой приоритет. Все возможные комбинации приоритетов реализованы в дешифраторе. Выдача сигналов BREQ/ и прием сигналов BPRN/ синхронизированы в задатчиках срезами импульсов BCLK/.

Арбитр микроЭВМ СМ1810 способен обеспечить арбитраж 16 задатчиков. Установкой перемычки в наборном поле арбитража выбирается циклический арбитраж, при удалении этой перемычки — параллельный арбитраж. Применение циклического арбитража целесообразно в системах с большим числом задатчиков, имеющих равные права на доступ к магистрали И41. В состав арбитра входят также генератор импульсов BCLK/, синхронизирующих операции захвата магистрали задатчиками, и генератор вспомогательной серии синхроимпульсов CCLK/.

Схема рестарта необходима для восстановления вычислительного процесса после аварийного отключения питания, вызванного падением сетевого напряжения ниже предельно допустимого уровня. Алгоритм восстановления, реализуемый схемой рестарта, соответствует требованиям интерфейса И41. При снижении сетевого напряжения до предельно допустимого значения источником питания микроЭВМ СМ1810 вырабатывается сигнал ACLO и за счет энергии, накопленной в его конденсаторах, гарантируется поддержание в течение 3 мс напряжений питания в системе. За это время должен быть прерван вычислительный процесс и обеспечено сохранение информации в памяти, питаемой от резервного источника. По ACLO прежде всего вырабатывается сигнал прерывания PFIN/, который, как правило, подается на вход немаскируемого прерывания микропроцессора. Одновременно с PFIN/ формируется сигнал PFSN/, являющийся признаком аварийной ситуации.

Через 2,5 мс вырабатывается сигнал защиты памяти MPRO/, блокирующий любые операции в памяти системы во время питания последней от резервного источника. За 2,5 мс, разделяющие моменты установки сигналов PFIN/ и MPRO/, процессор системы должен успеть сохранить в памяти, питаемой от резервного источника, информацию, необходимую для продолжения вычислительного процесса после подачи питания в систему. Признаком восстановления напряжения питающей сети является снятие сигнала ACLO. При этом снимаются сигналы PFIN/ и MPRO/ и в течение 5 мс поддерживается сигнал начальной установки INIT/. После снятия INIT/ система начинает восстанавливаться, при этом сигнал PFSN/ показывает, что возобновляется процесс, прерванный по аварийному отключению питания. PFSN/ сбрасывается программно-управляемым сигналом PFSR/, который формируется одним из задатчиков интерфейсной магистрали. Схема рестарта питается от резервного источника, что обеспечивает ее работоспособность в отсутствие сетевого напряжения.

Модуль системного контроля обеспечивает также сопряжение с органами управления и индикации, которые размещены на передней панели микроЭВМ СМ1810. С передней панели микроЭВМ могут быть инициированы сигнал начальной установки INIT/ и три сигнала запроса прерывания. Штатные значения уровней прерывания (формируемых от кнопок передней панели) — 0, 1, 7. Сигнал начальной установки INIT/ также будет автоматически сформирован при включении основного источника питания. На передней панели микроЭВМ СМ1810 индицируется основное и резервное питание 5 В, а также состояние сигналов интерфейса RUN/ (РАБОТА) и HALT/ (ОСТАНОВ). Светодиод РАБОТА поддерживается в излучающем состоянии одновибратором, на вход которого поступают от микропроцессора одного из задатчиков интерфейса импульсы ALE, формируемые в каждом цикле работы МП КР1810ВМ86. Если МП задатчика находится в состоянии

останова, то сформированный им интерфейсный сигнал HALT/ включает светодиод ОСТАНОВ. Светодиод РАБОТА при этом будет погашен.

В модуле системного контроля размещены нагрузочные сопротивления интерфейса И41, поддерживающие высокий уровень напряжения (что соответствует нулю) на линиях при отсутствии работающих задатчиков. Значения сопротивлений соответствуют требованиям интерфейса И41.

Модуль системного контроля устанавливается всегда в первое место конструктива СМ1810, предназначенного для установки задатчиков. Задатчики можно также установить в блоках расширения, где для них организован местный арбитраж. Арбитр блока расширения связывается с одним из входов арбитра модуля системного контроля. Это позволяет иметь систему с числом задатчиков, превышающим 16.

Телефон 455-56-41, Москва

ЛИТЕРАТУРА

1. ОСТ 25 969—83. Интерфейс И41. Технические требования.
2. Глухов В. И. Арбитраж в мультимикропроцессорных конфигурациях микроЭВМ СМ1810: Сб. науч. тр. ИНЭУМ «Модели микроЭВМ семейства СМ1810».— М.: Институт электронных управляющих машин, 1987.

Статья поступила 13.04.88

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МИКРОЭВМ СМ1810

В состав программного обеспечения микроЭВМ СМ1810 входят различные операционные системы (ОС) (тестовая, общего назначения, инструментальная и реального времени, а также прикладное программное обеспечение), что позволяет использовать ее во многих областях применений.*

Операционные системы общего назначения включают в себя МДОС1810 (совместима с MS DOS) и МИКРОС-86 (совместима с СР/М-86); в ОС реального времени входят ОС СФП1810 (совместима с RMX 88+MMX 800) и БОС1810 (совместима с RMX 86).

Операционная система со специализацией функций процессоров (ОС СФП1810) — чисто исполнительное средство, предназначенное для организации вычислительных процессов реального времени. В модули расширения системы входят драйвер терминала, распределитель памяти, драйверы устройств связи с объектами, система ввода-вывода и т. д. На уровне обращений из пользовательских программ ядро и модули расширения ОС СФП1810 совместимы (кроме системы ввода-вывода) с соответствующими компонентами ОС МОС РВ и ОС СФЛ для микроЭВМ СМ1800. Система ввода-вывода и формат дисковых файлов ОС СФП1810 являются совместимыми с БОС1810. В качестве инструментального средства для разработки пользовательских программ под управлением ОС СФП1810 могут применяться инструментальная ОС ДОС1810 (это ее основное назначение) и БОС1810. Трансляторы и различные утилиты обслуживания объектных модулей в этих системах одни и те же.

В рамках всех ОС для СМ1810 реализован универсальный программный интерфейс (УПИ), который является унифицированным набором системных вызовов. Если программный продукт использует только вызовы УПИ, то он без адаптации может быть перенесен из среды одной ОС в другую.

Базовое прикладное программное обеспечение состоит из следующих средств: баз данных; текстовых редакторов, процессоров и форматтеров; базовых графических средств, в том числе программного обеспечения для деловой графики; средств межмашинного обмена, эмуляции терминалов верхней ЭВМ, программного обеспечения для построения локальных сетей ЭВМ; интегрированных пакетов, объединяющих в себе базы данных, текстообработку, генерацию и обработку таблиц, машинную графику и средства межмашинного обмена информацией.

Ниже рассматриваются следующие операционные системы: ТОС-86, МДОС1810, БОС1810 и МИКРОС-86.

* Шкамарда А. Н. Шестнадцатиразрядные микроЭВМ семейства СМ1800 // Микропроцессорные средства и системы.— 1986.— № 5.— С. 6—10.

УДК 681.3.06

С. А. Вяткин

ТЕСТОВАЯ ОПЕРАЦИОННАЯ СИСТЕМА

Тестовая операционная система (ТОС-86), входящая в состав программного обеспечения для микроЭВМ СМ1810, предназначена для контроля работоспособности микроЭВМ и входящих в нее модулей и функциональных устройств, обнаружения сбоев, локализации неисправностей, а также для технического контроля вычислительных комплексов на заводе-изготовителе. ТОС-86 состоит из начальных тестов, супервизора и набора полных тестов.

Начальные тесты расположены в постоянной памяти модуля центрального процессора (МЦП-16) и предназначены для выполнения начального тестирования основных функциональных элементов микроЭВМ СМ1810. Они обеспечивают настройку на скорость системной консоли, проверку МЦП-16, постоянной и оперативной памяти, контроллеров накопителей на жестких и гибких магнитных дисках, коммуникационных модулей и печати.

Начальные тесты запускаются при включении питания или нажатии клавиши *сброс* на пульте управления микроЭВМ. После запуска производится тестирование основных функциональных микроЭВМ с выводом на системную консоль диагностических сообщений, содержащих минимальную информацию. Более детальная проверка должна выполняться с помощью полных тестов конкретных устройств, которые входят в состав ТОС-86.

Начальные тесты состоят из базового и вспомогательных тестов. Базовый тест предназначен для проверки ресурсов микроЭВМ, необходимых для работы вспомогательных тестов. При возникновении ошибки во время работы базового теста начальные тесты не имеют возможности вывода сообщения об ошибке на системную консоль, поэтому номер ошибки, соответствующий типу проверки, выдается в бесконечном цикле на пульт управления СМ1810. Базовый тест инициализирует память, проверяет микропроцессор К1810ВМ86, таймер, используемый каналом связи с системной консолью, и работоспособность канала связи.

После успешного завершения базового теста начинают исполняться вспомогательные тесты. Они выводят на системную консоль имена проверенных модулей и устройств, а также диагностические сообщения. Состав вспомогательных тестов в порядке их прохождения следующий: вспомогательных функций модуля МЦП-16; оперативной памяти; контроллеров накопителей на жестких и гибких магнитных дисках; модулей связи с последовательными интерфейсами; печатающего устройства.

Супервизор — ядро ТОС-86. Он организует исполнение отдельных тестов или их цепочки. Супервизор состоит из загрузчика тестов, команд, обработчика V-переменных, настройщика параметров и системных вызовов.

Загрузчик тестов загружает тесты в оперативную память микроЭВМ для их работы под управлением супервизора.

Команды: установка нового значения времени и даты; отображение содержимого оглавления диска; загрузка и запуск теста либо продолжение работы теста (если его работа была прервана оператором); загрузка V-переменных из файла на диске; сохранение значений V-переменных в файле на диске; назначение устройства для вывода протокола работы; организация пакетного режима.

Обработчик V-переменных. В ТОС-86 реализован универсальный механизм хранения и доступа к системной информации в V-переменных. V-переменная — структура данных, имеющая определенный тип со своим номером (0..0FFFFH). В V-переменных хранятся параметры теста, данные для супервизора и текстовые сообщения, что позволяет: определять начальное состояние теста (т. е. его переменных) путем загрузки V-переменных из файла на диске; определять язык диалога, задавая тот или иной файл с текстами, загружаемыми также в V-переменные; сохранять содержимое V-переменных с данными в файле на диске.

Все V-переменные делятся на классы: 0 — данные для супервизора, 1 — тексты для супервизора, 3 — общие тексты для всех тестов, 4 — данные для тестов, 5 — тексты для тестов.

Обработчик V-переменных содержит системные вызовы V-переменными, которые позволяют создавать, удалять и получать доступ к V-переменной, пересылать значение из одной V-переменной в другую, загружать из файла, сохранять в файле или уничтожать все V-переменные определенного класса.

Настройщик параметров работает в режиме меню, при этом экран организуется следующим образом: первая строка отображает общесистемную информацию; вторая — информацию о тесте; предпоследняя — команды оператору; последняя — диагностические сообщения или подсказки оператору; остальная область экрана — рабочая, где отображается меню для задания параметров либо информация о прохождении теста.

Имеется общая процедура настройки параметров, использующая непосредственно структуры данных, индивидуальные для каждого теста. Настройка параметров в режиме меню позволяет осуществлять два вида работ:

1. Выбор одного из многих. В рабочей области экрана появляется пронумерованный перечень альтернативных возможностей, и оператор должен указать номер интересуемой его возможности. Это используется, например, при выборе подтеста.

2. Задание значений переменных. На экране в рабочей области появляется пронумерованный перечень переменных с их текущими значениями. Оператор указывает номер той переменной, которую он хочет изменить, а затем вводит новое значение переменной.

Настройка параметров для теста — это чаще всего много последовательно выполняющихся меню. Описания меню и последовательность их вызова определяются при написании теста и представляют собой структуру данных, создаваемую с помощью специального набора макросов для Макроассемблера-86.

В процессе настройки параметров по цепочке меню можно отобразить на экране имена всех меню данной цепочки, перейти на настройку указанного меню, прекратить настройку параметров по цепочке меню в любой момент времени. Кроме того, вызвать подпрограмму, входящую в тест и выполняющую некоторые индивидуальные действия, например инициализацию устройства.

Системные вызовы позволяют осуществлять следующие действия: позиционирование курсора в рабочей области экрана; вывод текста в рабочую область, командную строку, строку подсказки, файл протокола; вывод всего содержимого экрана в файл протокола; чистку рабочей области, командной строки, строки подсказки, всего экрана; ввод текста; обновление строки информации о тесте; вывод сообщения об ошибке в рабочую область; организацию конца цикла исполнения подтеста; возврат в супервизор ТОС-86 после завершения работы теста; обработку ошибок ТОС-86 и прерываний; отображение десяти последних ошибок; управление системной консолью; временное включение-выключение протокола.

Системные вызовы в большинстве случаев достаточны для написания тестов. Иногда в процессе работы теста требуется преобразовать переменные различных типов из машинного представления в текстовые и обратно. Для этого в ТОС-86 имеются библиотеки, содержащие подпрограммы, которые обеспечивают данные преобразования.

Полные тесты организованы и работают под управлением супервизора ТОС-86 единым образом:

после загрузки и запуска теста в V-переменные загружаются текстовые сообщения и, если необходимо, данные для теста;

настраиваются параметры (если тест запущен в пакетном режиме, то настройка параметров не производится, они должны быть настроены заранее и сохранены в файле на диске);

запускается выбранный подтест. Во время работы подтеста информация с его прохождении выводится в рабочую область и, если протокол включен, в файл протокола;

в процессе исполнения подтеста оператор может прервать работу теста и выйти в супервизор.

Телефон 455-42-75, Москва

Статья поступила 13.04.88

УДК 681.3.06

И. И. Бабанов

МАЛАЯ ДИСКОВАЯ ОПЕРАЦИОННАЯ СИСТЕМА

Малая дисковая операционная система (МДОС1810) — однопользовательская, однопрограммная, инструментальная диалоговая ОС. Она позволяет строить конкретные проблемно-ориентированные системы, применяемые для обработки научно-технической и экономической информации, и выполнять значительную вычислительную обработку в диалоговом режиме (например, графические системы, лабораторные применения и т. д.). Развитое программное обеспечение МДОС1810 служит основой для разработок пакетов прикладных программ.

МДОС1810 дает возможность, с одной стороны, формировать удобную операционную обстановку для разработки программного обеспечения, с другой,— создавать автоматизированные рабочие места с простыми средствами доступа конечных пользователей к прикладным пакетам и программам. Кроме того, она обеспечивает работу с внешними устройствами СМ1810, а также с другими устройствами.

МДОС1810 совместима с ОС MS DOS и PC DOS, имеющими статус фактического стандарта для 16-разрядных персональных компьютеров. В частности, МДОС1810 полностью работоспособна на отечественных ЭВМ типа ЕС1840/41, «Искра 1130/1030» и на многих аналогичных зарубежных ПЭВМ.

В состав МДОС1810 входит мобильная система программирования, которая содержит набор инструментальных средств для создания прикладного и системного программного обеспечения для микроЭВМ, базирующихся на микропроцессоре К1810ВМ86, и для встраиваемых одно- и многопроцессорных систем. Эти средства полностью аналогичны средствам в ОС ДОС1810 и БОС1810.

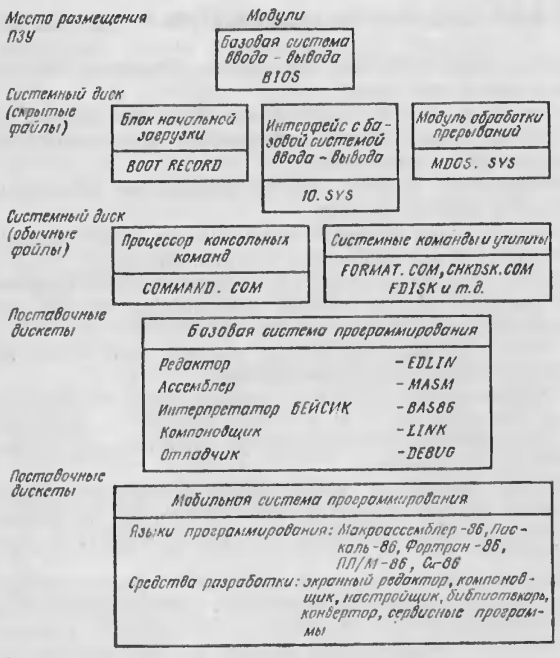
МДОС1810 обеспечивает подготовку, отладку и выполнение программ на СМ1810 со следующими функциями: инициализация системы; создание и редактирование исходных текстов программ; динамическое распределение памяти; трансляция с языков программирования Ассемблер-86, Паскаль-86, Фортран-86, ПЛ/М-86, Макроассемблер-86 и Си-86; интерпретация программ, написанных на языке программирования БЕЙСИК-86; компоновка перемещаемых модулей из отдельно оттранслированных программ и

настройка в загрузочный модуль; возможность компоновки программ, написанных на разных языках программирования; отладка программ; загрузка и выполнение программ; управление файловой системой (создание, копирование, удаление, сравнение, восстановление и переименование файлов); инициализация, форматирование, копирование, сравнение и восстановление дискет; программная поддержка жесткого диска; иерархическое распределение файлов на диске (организация многоуровневых директорий); восстановление файлов; обширная диагностика об ошибках; автозапуск программ сразу же после инициализации; пакетная обработка; включение в систему дополнительных драйверов устройств пользователя; запуск фоновой программы печати группы файлов одновременно с диалоговой работой пользователя; конфигурация МДОС1810 на этапе инициализации; поддержка виртуального (электронного) диска; переназначение стандартного ввода-вывода (например, возможность работы со всеми последовательными устройствами как файлами); построение конвейеров (программных каналов), которые позволяют выходные данные одной программы использовать в качестве входных для другой; контроль доступа к файлам и блокировка записей в файлах (контроль доступа к файлам необходим в многопользовательских или сетевых системах для исключения возможности одновременного доступа к файлам); возможность деления пространства жесткого диска на разделы, обслуживающие разные ОС (например, совместное использование жесткого диска ОС МДОС1810 и МИКРОС-86); создание и обслуживание библиотек объектных модулей;

МДОС1810 состоит из базовой системы ввода-вывода, записанной в постоянной памяти; загрузчика; процессора консольных команд (COMMAND.COM), обработки прерываний (MDOS.SYS); интерфейса с базовой системой ввода-вывода (IO.SYS); системных команд и утилит; базовой и мобильной систем программирования (см. рисунок).

IO.SYS — программный *интерфейс с базовой системой ввода-вывода* (BIOS), записанной в постоянной памяти. Модуль IO.SYS состоит из набора программ (драйверов), обслуживающих нижний уровень ввода и вывода внешних устройств.

Модуль обработки прерываний MDOS.SYS обеспечивает интерфейс верхнего уровня с программами пользователя. Он содержит подпрограммы, поддерживающие работу файловой системы, внешних устройств в ситуациях, связанных с завершением программ, а также ряд функций, доступных программам пользователя.



Состав МДОС1810

Процессор консольных команд (COMMAND.COM) — программа, которая, с одной стороны, ведет диалог с пользователем, а с другой,— поиск на диске вызванных пользователем программ, загружает их с диска в «транзитную область» (незанятая ОС часть оперативной памяти) и запускает их на выполнение. Все выполняемые программы («транзитные команды») в качестве файлов типа .COM или .EXE (файлы, имеющие расширение имени файла .EXE и .COM) расположены на диске. В процессор консольных команд резидентно встроены более 20 системных команд, постоянно находящиеся в оперативной памяти. Эти системные команды называются резидентными, так как часто используются, что повышает эффективность работы. Кроме того, процессор консольных команд обрабатывает командные файлы (файлы типа. BAT).

Резидентные системные команды находятся в процессоре COMMAND.COM. В их состав входят следующие команды:

DIR — просмотр оглавления диска;
ERASE (DEL) — удаление файлов;
REN — переименование файлов;
TYPE — выдача содержимого файла на консоль;
COPY — копирование файлов;
ASSIGN — переназначение дисковых устройств;
ECHO, FOR, GOTO, IF, PAUSE, REM, SHIFT — подкоманды пакетной обработки;
BREAK — проверка на прерывание от оператора;
CTTY — переназначение консоли (стандартного устройства ввода-вывода);
CLS — очистка экрана;
CHDIR, MKDIR, RMDIR — изменение, создание и удаление поддиректорий;
DATE, TIME — изменение даты и времени;
PATH — задание области для поиска;
PROMPT — установка новой системной подсказки;
VER — выдача номера версии;
VOL — выдача и установка имени диска;
VERIFY — установка режима проверки записи;
SET — установка общедоступных параметров так называемой системной среды.

Системные транзитные команды общего назначения содержатся на системной дискете и являются автономными сервисными программами (файлами типа .COM или .EXF). В их состав входят следующие команды:

ATTRIB — установка атрибутов файла;
BACKUP — архивное копирование (создание резервных копий на другие носители);
CHKDSK — информация о текущем состоянии на диске;
COMMAND — второй командный процессор;
COMP — сравнение файлов;
DYSKCOMP — сравнение дискет;
DYSKCOPY — копирование дискет;
FDISK — начальная подготовка жесткого диска;
EXE2BIN — преобразование загрузочных файлов типа .EXE;
FIND — поиск текстовых вхождений в файле;
FORMAT — форматирование дисков;
GRAFTABL — загрузка дополнительных символов для графического режима;
GRAPHICS — распечатка графических изображений;
JOIN — логическое объединение директорий на двух дисках в одну;
LABEL — установка имени дисков;
MODE — установка режима внешних устройств;
MORE — вывод текста поэкранно;
PRINT — фоновая печать;
RECOVER — восстановление файлов с дисков;
REPLACE — системная замена и копирование файлов;

RESTORE — восстановление ранее резервируемых файлов с помощью команды BACKUR;
SHARE — подключение файлов коллективного пользования и защита файлов;
SORT — сортировка текстовых файлов;
SUBST — подстановка виртуальных дисков;
SYS — копирование МДОС;
TREE — изображение дерева (иерархии) директорий;
XCOPY — выборочное копирование групп файлов и директорий.

В состав **базовой системы программирования** входят строковый редактор EDLIN, отладчик DEBUG, компоновщик LINK, ассемблер MASM и интерпретатор языка БЕЙСИК-86.

Ассемблер MASM — это транслятор с символического машинно-ориентированного языка программирования ассемблер-86 для 16-разрядного микропроцессора К1810ВМ86. Он за два прохода обрабатывает входной исходный файл. На выходе генерируются файлы: объектный, листинга и символических имен.

В процессе ассемблирования с помощью директив можно выполнить назначение логических сегментов участкам программы, условное ассемблирование, определение элементов данных, резервирование и инициализацию областей памяти, включение одного файла в другой, управление форматом выдачи распечатки. Кроме того, благодаря использованию макрокоманд можно определять «собственные» машинные команды.

Строковый редактор EDLIN — программа для создания, изменения и вывода на экран текстовых файлов. С его помощью можно создавать текстовые файлы и записывать их на диск; корректировать существующие текстовые файлы; удалять, редактировать, вставлять и выводить на экран строки; искать, удалять и заменять текст в пределах одной или нескольких строк.

Отладчик DEBUG служит для интерактивной отладки пользовательских программ, загружаемых с помощью специальной команды DEBUG. Отладчик предоставляет оператору возможность выполнения следующих функций: индикации и изменения содержимого регистров и ячеек памяти; выдачи на экран содержимого ячеек памяти в мнемоническом коде; ввода команд ассемблера в мнемоническом коде; сравнения блоков памяти; ввода и вывода в физические порты; чтения файлов или абсолютных секторов диска в память и записи файлов или секторов из памяти; заполнения памяти константой; дизассемблирования машинных кодов; исполнения программ — пошаговое и с точками останова; работы с дисками на физическом уровне.

Пошаговая обработка команд осуществляется путем использования режима останова после каждого шага. Программные вызовы могут рассматриваться в режиме пошаговой (покомандной) обработки как одиночные команды. Точки останова реализуются через программное прерывание (INT3). DEBUG сохраняет первоначальное содержимое по адресу останова и после прерывания вновь восстанавливает содержимое этой ячейки памяти.

Компоновщик LINK служит для редактирования связей между объектными модулями (полученными ассемблером MASM) перед их выполнением (компоновка объектных файлов). Редактор связей (LINK) объединяет отдельные объектные модули, осуществляет поиск файлов в библиотеке для определения неразрешенных внешних ссылок, разрешает внешние перекрестные ссылки, создает листинг, в котором указаны разрешенные внешние ссылки и сообщения об ошибках, и загрузочные модули.

Интерпретатор языка БЕЙСИК-86, работающий под управлением МДОС1810, позволяет пользователю вводить, редактировать, отлаживать и выполнять программы, написанные на расширенном диалекте языка БЕЙСИК. Язык БЕЙСИК-86 имеет широкий набор команд, операторов и стандартных функций. Могут быть определены написанные пользователем дополнительные функции. Интерпретатор предоставляет возможность выполнять ввод-вывод файлов, записанных на дисках, а также открывать, переименовывать, удалять файлы, изменять атрибуты файлов и выводить оглавление диска без выхода в МДОС1810.

Мобильная система программирования (МСП) состоит из широкого набора средств подготовки программ, в который входят трансляторы с языков программирования: Макроассемблер-86, Фортран-86, Паскаль-86, ПЛ/М-86, Си-86. Кроме того, в МСП входят средства модульной разработки программ: экранный редактор текстов, компоновщик, настройщик, библиотекарь и сервисные программы.

Мобильная система программирования в первую очередь обеспечивает набор инструментальных средств для создания прикладного и системного программного обеспечения для микроЭВМ, созданных на основе микропроцессора К1810ВМ86, и для встраиваемых одно- и многопроцессорных систем. МСП базируется на универсальном программном интерфейсе (УПИ), представляющем собой набор системных подпрограмм, которые реализуют различные способы взаимодействия программ с файлами, стандартным вводом-выводом и системными ресурсами. УПИ реализуется в ОС БОС1810 и ДОС1810 и позволяет переносить программы, разработанные на его основе, с одной системы на другую без адаптации.

Программной поддержкой УПИ является пакет RUN, работающий с программными продуктами, базирующимися на УПИ в программной среде МДОС1810. К таким программным продуктам относятся языки программирования — ПЛ/М-86, Макроассемблер-86, Фортран-86, Паскаль-86, Си-86, компоновщик LINK86, настройщик LOC86, библиотекарь LIB86, преобразователь в 16-ричный формат файла OH86, редактор текстов, пакет программ подготовки документации. Пакет RUN состоит из четырех программ:

RUN, запускающей наработанное под УПИ программное обеспечение;

FIX, корректирующей нестандартные исходные тексты программ, генерируемых определенными текстовыми редакторами, такими, например, как WORDSTAR. Эти редакторы допускают форматируемые и специфичные данные, не распознающиеся в среде программных продуктов, написанных под УПИ;

GENCOM, преобразующей загрузочный файл формата типа .86 в файл формата типа .COM;

программы на основе УПИ (трансляторы, компоновщик и т. д.).

В **пакет программ обслуживания объектных модулей** для 16-разрядного микропроцессора входят программы:

компоновщик LINK86; объединяет объектные модули программы в одно целое;

настройщик LOC86; настраивает объектный модуль на конкретный адрес памяти и формирует загрузочный модуль, готовый к выполнению;

библиотекарь LIB86; создает библиотеки объектных модулей и работает с ними;

преобразования формата объектных модулей (OH86); преобразует объектный модуль в 16-ричный формат;

создания таблицы перекрестных ссылок (CREF86), позволяет получить общую таблицу перекрестных ссылок для разрабатываемой пользователем прикладной системы, в том числе состоящей из многих модулей, написанных на различных языках программирования.

LINK86 позволяет создавать файлы, настраиваемые на адрес при загрузке, т. е. в случае разработки программ для 16-разрядного микропроцессора применение программ LOC86 не обязательно.

Данные программы обрабатывают объектные модули, имеющие единую структуру, вне зависимости от языка, на котором были написаны соответствующие исходные модули.

Автоматическое выполнение программ. На дискете МДОС может находиться особый командный файл, имеющий имя AUTOEXEC.BAT. Он содержит пакет заданий, которые выполняются автоматически каждый раз при запуске системы. С его помощью пользователь может выполнять программы и команды оперативно, каждый раз при загрузке МДОС.

Конфигурация системы. Одна из особенностей МДОС1810 — возможность изменения конфигурации системы. При загрузке МДОС1810 по умолчанию устанавливается множество параметров, которые управляют системой. Они указывают на то, какой объем памяти отводится на использование дисковыми файлами, сколько дисковых файлов могут быть одновременно доступны, когда можно прервать программу, какие задействованы устройства и т. д. Эти параметры могут быть изменены специальными командами, записанными в файл CONFIG.SYS.

Для изменения конфигурации системы используются следующие команды: установка расширенной проверки по нажатию клавиш CTRL и C (BREAK); определение числа дисковых буферов (BUFFERS); инсталирование драйверов устройств (DEVICE); определение числа файлов, которые могут быть открыты с использованием управляющих блоков файлов (FCB); определение максимального числа файлов, которые могут быть одновременно открыты в системе (FILES); установка максимального числа доступных дисководов (LASTDRIVE); определение имени нового процессора команд (SHELL).

Используя команды, заданные в файле CONFIG.SYS, можно создать собственные, нестандартные возможности, которые конфигурируют МДОС соответствующим образом на время текущего сеанса (до очередной перезагрузки МДОС). Одна из основных конфигурационных команд — DEVICE, позволяющая подключить дополнительные драйверы к системе.

В составе МДОС поставляются драйверы, обеспечивающие: ANSI.SYS —дополнительные средства управления экраном и клавиатурой, DRIVER.SYS — работу блочно-ориентированных устройств, а также драйвер VDISK.SYS, позволяющий создать в оперативной памяти так называемый электронный (виртуальный) диск.

Телефон 455-42-75, Москва

Статья поступила 13.04.88

УДК 681.3.06

А. Д. Азаров

БОЛЬШАЯ ОПЕРАЦИОННАЯ СИСТЕМА

Большая операционная система (БОС1810), совместимая с ОС RMX86,— основа для специализированного программного обеспечения. БОС1810 обеспечивает сервисные средства, необходимые в управлении многозадачностью, динамическим распределении памяти, поддержке файловой системы, обработке прерываний и т. д. Имея эти средства, разработчики концентрируют свое внимание на программных средствах, непосредственно относящихся к конкретному применению, что сильно сокращает время разработки прикладной системы.

БОС1810, с одной стороны,— многоцелевая система, которая может исполняться на развитых универсальных микроЭВМ и служить в качестве инструментальной. В ее состав входят языки программирования Макроассемблер-86, ПЛ/М-86, Паскаль-86, Фортран-86, Си-86, компоновщик LINK86, настройщик LOC86, библиотекарь LIB86, символьный отладчик, экранно-ориентированный и меню-управляемый редактор текстов и другие инструментальные средства. С другой — встраиваемая система: на универсальной микроЭВМ можно создать версию, которая будет исполняться минимальным набором аппаратных средств. Сгенерированные пользователями версии БОС1810 отличаются в десятки раз по своим функциональным возможностям и объему занимаемой ими памяти.

Процесс создания прикладной системы (конфигурация) осуществляется одним из компонентов БОС1810 — интерактивным конфигуратором. При конфигурировании на экран терминала выводятся меню с различными параметрами аппаратных и программных средств. Нужная конфигурация задается в ходе диалога, состоящего из вопросов интерактивного конфигуратора и ответов пользователей. Интерактивный конфигуратор позволяет сохранить результаты предыдущего конфигурирования, поэтому небольшое изменение можно сделать быстро, не отвечая на все вопросы.

Одна часть меню связана с настройкой системы на конкретные аппаратные средства: базовые порты и расстояния между портами отдельных микросхем на плате центрального процессора, номера портов и уровни прерываний различных контроллеров ввода-вывода, диапазоны адресов имеющейся оперативной и постоянной памяти и т. д.; другая — с выбором нужных компонентов БОС1810 и отбрасыванием остальных. БОС1810 состоит из набора подсистем, называемых слоями, каждая из которых предоставляет пользователю ряд функций. Можно выбрать только необходимые для будущей системы слои (отбросив все остальные), а внутри выбранного слоя — часть его функций; допустить еще бо́льшую детализацию и выбрать только конкретные системные вызовы, предоставляемые данным слоем.

И наконец, третья часть меню позволяет указать программное обеспечение, написанное самим пользователем. Оно добавляется к выбранным компонентам БОС1810 и в результате получается законченная система.

В БОС1810 входят следующие слои: ядро, системы ввода-вывода (базовая и расширенная), драйверы устройств и терминала, интерфейс с оператором, загрузчики прикладных программ, динамический и системный отладчик, универсальный программный интерфейс, а также начальный загрузчик, использование и конфигурирование которого осуществляются автономно от самой БОС1810.

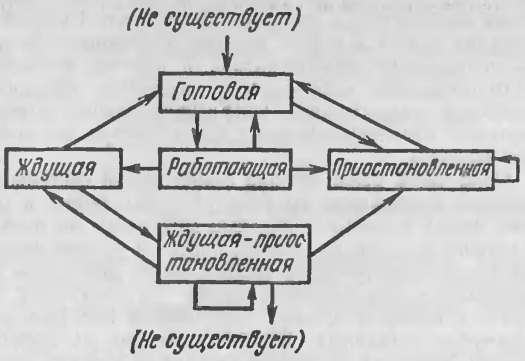
Ядро — сердцевина БОС1810, организующая вычислительные процессы в мультипрограммном режиме. Функции ядра: диспетчеризация, управление доступом к системным ресурсам, обеспечение связей между отдельными процессами и реакций системы на внешние события.

Основная концепция, используемая в ядре БОС1810,— *объектно-ориентированная архитектура.* Сущностью ее является предоставление ядром набора «строительных» блоков, из которых конструируются другие слои БОС1810 и прикладное программное обеспечение пользователей, а также системных вызовов. Эти блоки называются *объектами* и разделяются на следующие классы: задачи, задания, сегменты, почтовые ящики, семафоры, регионы и т. п. Ядро следит за каждым из объектов с помощью 16-ричного числа, однозначно идентифицирующего объект среди всех объектов системы. Это число называется *обозначением* объекта и устанавливается ядром при создании последнего.

Системный вызов для управления объектами, с точки зрения программиста, выглядит, как обычная внешняя процедура. Одним из параметров данной процедуры является *код завершения,* указывающий на удачное или неудачное завершение системного вызова. Если системный вызов нормально выполнил все необходимые действия, то код завершения равен нулю; если в процессе своей работы системный вызов столкнулся с ситуацией, которую не может преодолеть, он возвращает этому параметру ненулевое численное значение, идентифицирующее особую ситуацию.

БОС1810, используя существующие объекты, может создавать свои собственные типы объектов, системные вызовы для них и включать их в состав системы.

Задачи — единственные активные объекты в системе. Они имеют две цели: сделать определенную часть работы и взять на себя управление процессором, чтобы продвинуться в достижении своей первичной цели. С точки зрения программиста, задача оформляется, как обычная процедура. Когда несколько задач одновременно хотят «занять» процессор, ядро осуществляет их диспетчеризацию, базируясь на состоянии задачи и ее приоритете. *Приоритет задачи* — это целое число между 0 и 255 включительно.



Возможные переходы между состояниями задачи

Чем меньше число, указывающее приоритет, тем выше приоритет задачи.

Задача всегда находится в одном из пяти *состояний исполнения* (ждущая, приостановленная, ждущая-приостановленная, готовая и работающая):

ждущая — задача ждет удовлетворения некоторого запроса. Кроме того, она сама может поместить себя в это состояние;

приостановленная — задачу помещает в это состояние другая задача, и она ожидает прерывания или приостанавливает себя. С каждой задачей связана глубина приостанова, которая показывает число приостановов. Прежде чем задача сможет покинуть это состояние, каждой операции приостанова должна быть поставлена в соответствие операция продолжения;

ждущая-приостановленная — при приостанове ждущей задачи. В данном состоянии время ожидания задачи может истечь и она перейдет в состояние *приостановленная*. Кроме того, если другая задача возобновляет задачу, находящуюся в состоянии *ждущая-приостановленная*, последняя перейдет в состояние *ждущая*;

готовая — задача не является *ждущей, приостановленной* или *ждущей-приостановленной*.

Алгоритм арбитража, используемый ядром, состоит в том, что *работающей* является готовая задача с максимальным (численно-минимальным) приоритетом.

Ядро не выделяет задачам процессор методом разделения времени (с равными интервалами). Напротив, во время работы прикладной системы на базе БОС1810 происходят внешние и внутренние по отношению к ней события, которые заставляют задачи переходить из одного состояния в другое (см. рисунок). Поэтому ОС БОС1810 — это управляемая событиями система.

В многозадачной системе должны быть обеспечены методы для координации задач друг с другом. Эти методы реализованы с помощью механизма обменников (регионов, семафоров и почтовых ящиков) и позволяют задачам выполнять обмен информацией, взаимоисключение и синхронизацию.

Регионы ограничивают доступ к критическим секциям кодов или данных. После того, как задаче предоставлен доступ к ресурсам, защищенным регионом, никакие другие задачи не могут использовать эти ресурсы, а сама задача не может быть удалена или приостановлена.

Семафоры используются для взаимного исключения задач.

Почтовые ящики обеспечивают связь между задачами путем посылки в них объектов. Каждый почтовый ящик имеет две очереди: для задач (очередь задач), ждущих получение объектов, и для объектов (очередь объектов), посланных задачами, но пока еще не принятых. Ядро следит за тем, чтобы ждущие задачи получали объекты, как только те появляются. Поэтому в каждый момент времени как минимум одна из очередей почтового ящика пуста.

Для достижения полного преимущества мультипрограммирования необходимо обеспечить собственную среду для каждой программы, что подразумевает раздельное использование памяти, файлов и объектов. Именно эту функцию и выполняют в системе *задания* (рабочей среде), в которой находятся объекты, такие как задачи, почтовые ящики и т. п. Любой объект в системе может использоваться задачами из разных заданий, но принадлежать он может только одному заданию.

Задания в системе организованы в виде дерева. *Корневое* — это задание, поддерживаемое ядром. Задачи внутри некоторого задания могут создавать новые задания и т. д. Вновь созданное задание называется потомком, задание, в рамках которого происходит создание,— родителем. Каждое задание в дереве получает свои ресурсы (например, необходимую ему память) от своего родителя.

Один из основных вопросов систем реального времени — обработка прерываний. Внешние события проявляются асинхронно по отношению к внутренним работам прикладной системы. Прерывание, сигнализирующее о появлении внешнего события, запускает аппаратный «вызов» обращения к ячейке памяти, находящейся в области, именуемой *таблицей векторов прерываний*. Далее управление передается к определяемой пользователем процедуре обработки прерываний, называемой *драйвером прерываний*. Если обработка прерывания требует мало времени и не требует системных вызовов, кроме тех, которые относятся к обработке прерываний, то драйвер прерываний в действительности обрабатывает прерывание. В противном случае он включает *задачу обработки прерываний*, которая и занимается прерыванием.

Системы ввода-вывода. БОС1810 поддерживает две системы ввода-вывода, чтобы позволить выбрать ту из них, которая лучше удовлетворяет требованиям конкретной прикладной системы. Возможно их совместное использование.

Базовая система ввода-вывода — более гибкая из двух систем ввода-вывода. Она обеспечивает большие возможности и предъявляет минимальные требования к прикладной системе. Гибкость базовой системы ввода-вывода позволяет пользователю строить свой собственный алгоритм буферизации, предоставляет асинхронные системные вызовы, дает возможность задаче пользователя управлять операциями ввода-вывода в деталях. Системные вызовы ее включают большое число параметров. Используя эти параметры, задачи могут тщательно приспосабливать выполнение каждого системного вызова к требованиям конкретной прикладной системы.

Расширенная система ввода-вывода является «надстройкой» над базовой системой ввода-вывода и пользуется ее системными вызовами. Она более проста вследствие следующих свойств:

автоматической буферизации операций ввода-вывода. За исключением того, что требуется указать, сколько буферов следует использовать расширенной системе ввода-вывода, задачи пользователя не должны вовлекаться в проблемы буферизации;

синхронности системных вызовов. Освобождая прикладную программу от проблемы явной синхронизации системных вызовов, расширенная система ввода-вывода снижает сложность прикладной системы;

освобождения пользователей от утомительных деталей. Системные вызовы расширенной системы ввода-вывода требуют меньше параметров, чем системные вызовы базовой системы ввода-вывода.

Драйверы устройств. В составе БОС1810 поставляется набор драйверов устройств для поддержки модулей микроЭВМ СМ1810, значительно упрощен процесс написания пользователем собственных драйверов устройств.

Интерфейс с Оператором ответствен за загрузку и выполнение программных файлов. Он состоит из набора готовых команд, поставляемых вместе с ОС; набора системных вызовов, помогающих программистам писать свои собственные команды; стандартного интерпретатора командных строк (CLI); поддержки многопользовательского режима; поддержки специальных знаков в именах файлов.

Загрузчик прикладных программ загружает программы под управлением задач БОС1810. Он обеспечивает системные вызовы, которые загружают программы из внешней памяти в оперативную. Системные вызовы загрузчика позволяют программам исполняться в системах, не обладающих достаточным объемом памяти, чтобы хранить все эти программы в памяти одновременно. Основные возможности загрузчика: настройка на адрес во время загрузки и оверлейная загрузка.

Универсальный программный интерфейс (УПИ) — набор системных вызовов, совместимых с каждой ОС СМ1810. Если некоторая прикладная программа пользуется только системными вызовами УПИ, то она может быть перенесена с одной ОС на другую. УПИ входит в состав ДОС1810, БОС1810, МДОС1810 и ОС СФП1810.

Отладочные инструменты. *Отладчик БОС1810* — инструмент, «настроенный» на структуры данных, поддерживаемые ядром. Использование отладчика позволяет:

вмешиваться в работу или проверять любую задачу, в то время как остальные задачи в системе продолжают выполняться. В этом состоит отличие отладчика БОС1810 от системного отладчика БОС1810, требующего, чтобы прикладная система была «заморожена»;

контролировать поведение системы без вмешательства в ее выполнение;

проверять и интерпретировать структуры данных, связанных с ядром и его объектами.

Вторым отладочным инструментом является *МОНИТОР-86*, обеспечивающий пошаговое исполнение программы, устанавливающий исполнительные точки останова и точки останова при обращении к памяти, вывод содержимого памяти в различных форматах, выполнение операции ввода-вывода через порты, перемещение, нахождение и сравнение блоков памяти, дизассемблирование.

Системный отладчик расширяет возможности МОНИТОРА-86. В добавление к функциям МОНИТОРА-86 системный отладчик позволяет: идентифицировать и интерпретировать системные вызовы БОС1810; отображать объекты БОС1810; проверять стек задачи для определения того, какой из системных вызовов выполнялся последним.

Телефон 455-42-75, Москва

Статья поступила 13.04.88

УДК 681.3

Н. И. Васильев, Э. М. Пройдаков, Н. А. Диделева, Г. В. Левитина, И. И. Бабанов, Л. М. Биневич, Е. Я. Винокур, С. Г. Бабаян

ОСОБЕННОСТИ РЕАЛИЗАЦИИ ОС МИКРОС-86

Разработку ОС МИКРОС-86 для СМ1810, начатую в 1984 г., обусловило следующее: широкое распространение в мире ряда ОС, совместимых с CP/M-80,86 и MP/M-80,86, наличие сетевых версий этих систем и появление CCP/M-86; необходимость в системе открытого типа, настройку которой на терминал и нестандартную периферию может выполнять сам пользователь; появление микросхем Интел 80150, содержащих CP/M-86 на кристалле; необходимость в ОС для наиболее дешевых конфигураций СМ1810, рассчитанных на массовое применение в непромышленной сфере.

Поэтому при разработке МИКРОС-86 были поставлены задачи совместимости с операционной средой CP/M-86, функциональной совместимости с ОС1800, расширения уровня сервисного обслуживания, а также предоставления пользователю возможности модификации системы.

МИКРОС-86 — инструментальная, однозадачная, персонально-ориентированная ОС общего назначения, предназначенная для программирования, отладки и выполнения прикладного ПО преимущественно в следующих областях: текстообработке; диалоговой обработке экономической, научно-технической и медицинской информации; интерактивной машинной графике, обучении и играх. МИКРОС-86 — система открытого типа, имеющая средства для настройки (генерации) на работу с нестандартными (или дополнительными) внешними устройствами.

Связь ОС с пользователем осуществляется через алфавитно-цифровой или графический дисплей (ввод команд с клавиатуры и получение ответов системы на экране дисплея). МИКРОС-86 может работать как в диалоговом, так и в пакетном режимах. В пакетном режиме последовательность командных строк записывается в дисковый файл, который затем запускается программой SUBMIT. После выполнения заданной последовательности команд МИКРОС-86 возвращается в диалоговый режим работы.

МИКРОС-86 обеспечивает выполнение следующих функций: форматирование и копирование системных и пользовательских дисков; создание и редактирование исходных текстов программ; интерпретацию программ, написанных на языке БЕЙСИК-86; трансляцию с языков программирования Макроассемблер-86, Фортран-86, ПЛ/М-86, Паскаль-86 и Си-86; компоновку программ из отдельно оттранслированных модулей (получение загрузочных модулей); загрузку, выполнение и отладку программ; управление файлами с динамическим распределением дискового пространства; создание и обслуживание библиотек объектных модулей; поддержку арифметического сопроцессора и псевдодиска (часть ОЗУ, функционально используемая ОС как НГМД, дает повышение производительности в 3...6 раз в зависимости от характеристик дисковой подсистемы микроЭВМ).

МИКРОС-86 обслуживает стандартные логические устройства: два дисплея, до 16 дисковых устройств разных типов, устройства печати, асинхронный последовательный канал. Для экономии дисковой памяти применена одноступенчатая схема загрузки ядра, которое содержится в файле MICROS .SYS. Программы, входящие в состав МИКРОС-86, делятся на резидентные, транзитные и мобильные.

Резидентные программы — записанный в РПЗУ универсальный начальный загрузчик, загружаемая с диска программа эмуляции архитектуры персональной ЭВМ и основной загрузчик, а также занимающее около 16 Кбайт ядро ОС.

Транзитные программы динамически загружаются с диска. С точки зрения пользователя, они являются командами. Команды МИКРОС-86 обеспечивают довольно простой язык взаимодействия с ОС, построенный традиционно: первое слово команды является именем вызываемой программы, последующие — параметры команды.

Мобильные программы разработаны в рамках мобильной системы программирования (МСП). Они могут быть выполнены не только в МИКРОС-86, но и в МДОС1810, БОС1810 и ДОС1810 без перетрансляции исходного текста программ [1, 2].

Основные системные компоненты. Ядро МИКРОС-86 включает в себя процессор консольных команд CCP, систему логического ввода-вывода BDOS и базовую систему ввода-вывода BIOS. *CCP* обеспечивает интерпретацию командной строки, выполнение резидентных (встроенных в ядро) команд и загрузку программ; *BDOS* поддерживает логический уровень ввода-вывода, управление памятью и обработку ошибок; BIOS обеспечивает настройку на конкретные периферийные устройства. BIOS написан на языке ASM86 и поставляется в исходных текстах вместе с набором некоторых программных средств для генерации и «Руководством системного программиста», в котором подробно описана технология генерации. Благодаря этому ОС МИКРОС-86 может быть инсталлирована не только на широкую конфигурацию технических средств СМ1810, но и практически на любую ЭВМ на базе МП К1810ВМ86/К1810ВМ88. Пользователь может самостоятельно изменить состав периферийных устройств и расширить функциональные возможности системы.

При разработке исходного текста BIOS использован механизм условной трансляции, позволяющий создавать не-

сколько версий BIOS в зависимости от условий, задаваемых ключами трансляции. В данной реализации МИКРОС-86 предусмотрено создание версии системы, функционирующей как на СМ1810, так и на ПЭВМ, совместимых с IBM PC XT. Для организации взаимодействия с аппаратными средствами микроЭВМ в BIOS были приняты решения, позволившие реализовать расширение функциональных возможностей клавиатуры, программный способ назначения соответствия физических устройств логическим, блокирование-деблокирование записей, организацию электронного диска в ОЗУ и обработку управляющих ESC-последовательностей.

Расширение функциональных возможностей клавиатуры обеспечивается обработкой нажатия так называемых программируемых функциональных клавиш (ПФК): для каждоя ПФК в служебной зоне BIOS зарезервирован буфер, в котором хранится соответствующая последовательность символов. Функция ввода с консоли реализована так, что в программу пользователя вместо введенной ПФК возвращается соответствующая ей последовательность символов. Переназначение ПФК может выполняться из программы пользователя с помощью управляющей ESC-последовательности.

В BIOS возможна переадресация ввода и вывода назначением логическим устройствам различных физических устройств посимвольного ввода-вывода. С этой целью логическому устройству в служебной зоне выделяется 16-бит слово назначения. Каждый установленный в единицу бит слова назначения определяет физическое устройство, подключенное в данный момент к логическому устройству. В зависимости от установленного слова назначения при обращении к логическому устройству управление передается на тот или иной драйвер. Изменение слова назначения из программ пользователя можно выполнить с помощью управляющей ESC-последовательности.

Блокирование-деблокирование записей улучшает временные характеристики при выполнении операций чтения-записи магнитного диска. В соответствии с этим алгоритмом в служебной области базовой системы ввода-вывода имеется буфер дорожки диска. Первое обращение к записи, находящейся на какой-либо дорожке, приводит к чтению дорожки в буфер. Любое следующее обращение к данной дорожке сводится к обмену с буфером. При переходе на новую дорожку информация из буфера записывается на диск и буфер обновляется. Этот алгоритм позволяет существенно сократить число физических операций доступа к дисковому устройству.

Обработка ESC-последовательностей используется для управления курсором, переадресации ввода-вывода, программирования ПФК и т. д. Код символа ESC(1BH) является признаком начала ESC-последовательности. Далее могут следовать один или несколько символов в зависимости от функции. Символ, следующий непосредственно за символом ESC, указывает функцию, которую надо выполнить. ESC-последовательности должны вводиться из программы пользователя обращением к соответствующим вызовам логической системы ввода-вывода ОС МИКРОС-86.

Электронному диску (M:-диск) в ОЗУ выделяется область памяти с некоторого начального адреса (установленного утилитой SETUP) и до конца доступной области ОЗУ. M:-диск позволяет повысить производительность МИКРОС-86, так как при размещении на M:-диске рабочих и временных файлов, листингов трансляции значительно уменьшается число выполняемых дисковых операций.

Реализация отмеченных возможностей в BIOS послужила основой для многофункциональных, управляемых посредством меню утилит:

ASSIGN — выводит на экран дисплея текущее назначение и переназначает устройства посимвольного ввода-вывода. С помощью этой утилиты можно управлять потоками вводимой и выводимой информации, исходя из потребностей конкретного пользователя;

DSKMAINT — выполняет большинство общих, часто используемых операций по обслуживанию НГМД: форматирование дискет, копирование с диска на диск, проверку и сравнение дисков;

SETUP — устанавливает и сохраняет на диске некоторые параметры, влияющие на режимы работы системы: время шага головок чтения-записи диска, наличие в системе электронного диска (с его объемом), командной строки, автоматически выполняемой после загрузки-перезагрузки системы;

FUNCTION — выводит на экран дисплея и изменяет содержимое буферов ПФК.

Программа RUN обеспечивает выполнение мобильных программ и средств их разработки — трансляторов с языков Макроассемблер-86, Фортран-86, Паскаль-86 и ПЛ/М-86. Языки Си-86, БЕЙСИК-86, RASM86 (перемещающий ассемблер) и ASM86 (абсолютный ассемблер для BIOS) входят в резидентную систему программирования. Программы на языках Си-86 и БЕЙСИК-86 мобильны только на уровне исходных текстов программ. Программа RUN (15 Кбайт) также написана на ассемблере и эмулирует виртуальную операционную среду в рамках предложенного фирмой Интел Универсального программного интерфейса (УПИ). Так как программы МСП имеют собственный формат объектного файла, то RUN включает в себя загрузчик таких файлов.

В МИКРОС-86 входят два отладчика: DDT86 и SID86. Хотя отладчик SID86 перекрывает возможности DDT86 и достаточно иметь только его, такое дублирование было вызвано неуверенностью разработчиков в том, что работы по SID86 будут выполнены в срок.

Организационные мероприятия. Система МИКРОС-86 для СМ1810 разрабатывалась двумя организациями: ИНЭУМ (головная) и НИИ Ленэлектронмаш. В разработке участвовало около 40 чел. Вся документация выполнялась только машинным способом с помощью разработанного в ИНЭУМ пакета DOCFOR. Значительную помощь в разработке оказало сотрудничество с ГДР и ЧССР. Основой взаимодействия всех разработчиков являлся план-график работ, который постоянно уточнялся, а также протокол о распределении работ.

Совместимость с другими ОС. МИКРОС-86 функционально совместима с ОС1800 по организации файловых систем. Набор команд МИКРОС-86 семантически близок набору команд ОС1800, хотя значительно превосходит его функционально. Это позволяет пользователям систем, совместимых с CP/M-80 версии 2.2 и выше (SCP 1715, МикроДОС и др.), легко освоить работу в среде МИКРОС-86.

Кроме того, МИКРОС-86 программно совместима с CP/M-86 (т. е. все программные продукты, функционирующие в среде CP/M-86, могут выполняться под управлением МИКРОС-86), снизу-вверх — с MP/M-86 и CCP/86. По сравнению с функционально аналогичной ОС M86 на ЕС1840 МИКРОС-86 не злоупотребляет полной руссификацией и превосходит ее по инструментальным средствам.

Заключение. В целом при разработке МИКРОС-86 была сделана попытка творчески переосмыслить накопленный для микроЭВМ опыт разработок системного ПО. Наличие МСП позволяет разрабатывать любую ОС для заказных или серийных микропроцессорных систем.

Телефон 283-97-64, Москва

ЛИТЕРАТУРА

1. Бабанов И. И., Пройдаков Э. М. Универсальный программный интерфейс и мобильность компиляторов // Сб. «Модели микроЭВМ семейства СМ1810».— М.: ИНЭУМ, 1987.— С. 109—113.
2. Пройдаков Э. М., Бабанов И. И. Средства обеспечения горизонтальной мобильности прикладных и инструментальных систем // Сб. «Машинно-независимые операционные системы».— М.: МЦНТИ, 1987.— С. 63—68.

Статья поступила 15.06.88

УДК 681.3.06

Г. Г. Кольнер, С. Г. Маслов

КРОСС-СИСТЕМА ПРОГРАММИРОВАНИЯ ДЛЯ ОДНОКРИСТАЛЬНЫХ ЭВМ

Кросс-система программирования Кросс-51 предназначена для разработки программ с целью применения их в системах с использованием однокристальных ЭВМ и микропроцессорных контроллеров, построенных на их основе (например, ОЭВМ КР1816ВЕ51).

Программы Кросс-51 выполняются под управлением дисковых операционных систем ДОС1800 и ДОС1810, БОС1810 и МДОС1810 на микроЭВМ СМ1800 и СМ1810 соответственно. Кросс-51 включает в себя: трансляторы с языков программирования ПЛ/М-51 и ассемблер-51 и пакет программ для обслуживания объектных модулей ОЭВМ.

Пакет программ для обслуживания объектных модулей содержит редактор связей/настройщик RL51 и библиотекарь LIBA51. Программа RL51 выполняет следующие функции: объединяет перемещаемые частичные сегменты с одним именем в один сегмент; распределяет память под объединенные сегменты, полученные на предыдущем шаге, и под все другие завершенные сегменты, перемещаемые из входных модулей; разрешает ссылки на внешние символы между модулями; связывает перемещаемые адреса с абсолютными; создает абсолютный объектный файл и листинговый файл, содержащий краткое описание связей, таблицы символов и сообщения о перекрестных ссылках; выявляет ошибки в редактируемых модулях и сообщает о них.

Программа LIB51 используется для организации работы с библиотечными файлами, представляющими собой наборы объектных модулей, любой элемент которых может быть выделен для разрешения программных ссылок на внешние имена. LIB51 позволяет также изменять содержимое этих файлов добавлением новых модулей или удалением старых. Система команд LIB51 полностью совместима с системой команд библиотекаря LIB из пакета программ обслуживания объектных модулей операционных систем ДОС1800 и ДОС1810.

Процесс разработки модульных программ с использованием программ Кросс-51 представлен на рисунке.

Ввод и редактирование исходных модулей. Текстовый редактор (например, EDIT80) используется для набора исходных модулей на языке ассемблер-51 или ПЛ/М-51 в исходные файлы на дисках.

Ассемблирование и компиляция. Трансляторы ASM51 или PLM51 преобразуют исходные модули в перемещаемый объектный код, порождая объектный файл. Объектный файл после ASM51 является перемещаемым, если хотя бы один входной сегмент — перемещаемый, иначе объектный файл — абсолютный. Объектный файл, получаемый после работы компилятора PLM51, всегда перемещаемый.

Перемещение и установка связей. После трансляции всех модулей программы обработкой файлов занимается программа RL51. Результат ее работы — файл абсолютного объектного модуля и файл листинга, содержащий информацию о работе RL51.

Версии систем с ПЗУ и РПЗУ. Для версий микроЭВМ с ПЗУ абсолютный объектный модуль «зашивается» в ПЗУ в процессе производства. Программатор РПЗУ загружает абсолютный объектный модуль в микроЭВМ с программным доступом к памяти для исполнения.

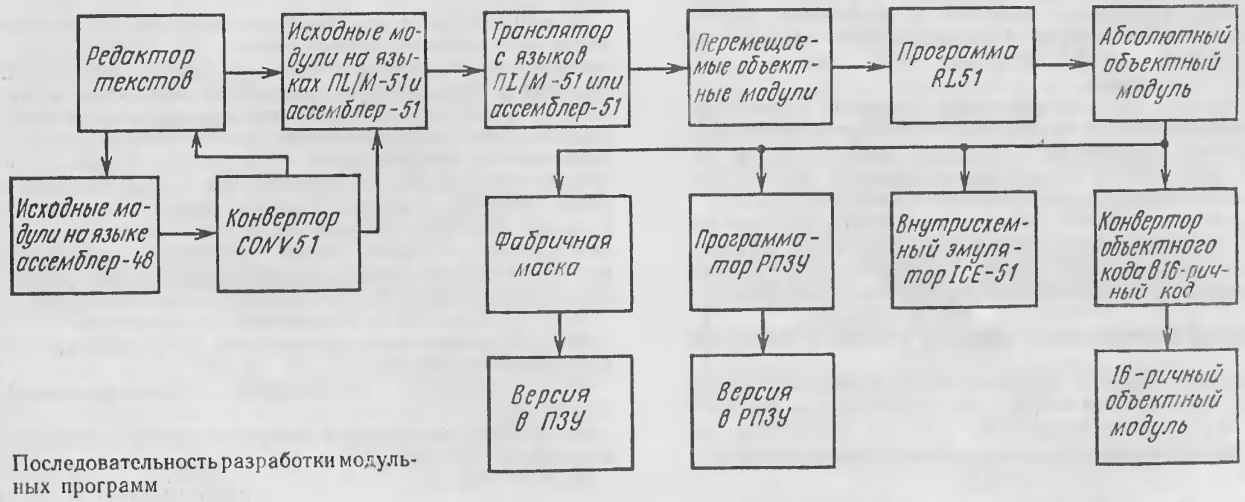
Заключение. Использование Кросс-51 на СМ1800 и СМ1810 позволит: обеспечить разработку ПО интеллектуального устройства связи с объектом для ЭВМ семейства СМ1800; расширить сферу применения СМ1800 и СМ1810, в первую очередь для проектирования и разработки ПО приборов и устройств на базе микропроцессора КР1816ВЕ51; повысить эффективность труда программиста; отказаться от зарубежной техники с аналогичным ПО.

Конвертор CONV51. Полезным дополнением к системе Кросс-51 является программа-конвертор CONV51, которая преобразует исходные тексты, написанные на ассемблере ASM48, в исходные тексты на ASM51. Применение программы CONV51 обусловлено наличием большого программного задела на ассемблере ASM48, а также ситуаций, исключающих возможность отладки программ на ASM51, но позволяющих отлаживать программы на ASM48.

Программа CONV51 в качестве входного файла использует не имеющий ошибок исходный файл на ассемблере ASM48 и необязательные для конвертора директивы и создает файл листинга (необязательный) и выходной (OUTPUT) с исходной программой на ASM51. Файл листинга содержит копии исходной программы на ASM48 и выходной исходной программы на ASM51 с вставленными диагностическими сообщениями.

Требования к аппаратно программному обеспечению: наличие ЭВМ с 64 Кбайт ЗУПВ (запоминающее устройство произвольной выборки), по крайней мере с одним дисководом (CONV51 находится на одной дискете), и функционирование операционных систем ДОС1800 или ДОС1810. Требования к исходному файлу на ASM48: программа на ASM48 должна быть отлажена, не содержать более 450 символических имен; длина строки не превышать 126 символов, исключая символы возврата каретки и перевода строки.

В зависимости от диагностических сообщений конвертора может быть необходимо ручное редакти-



Последовательность разработки модульных программ

рование выходного файла; причем в ряде случаев значительного объема.

Не рекомендуется использовать CONV51 для программ на ASM48: которые работают со стеком и применяют команды, отличные от команд CALL, RET и RETR; работа которых зависит от содержимого стека вне адресов ОЗУ в 8 по 15; нормальное выполнение которых зависит от скорости выполнения команд, числа командных циклов или действительного времени (прерывание по времени); в которых моделируются операции, выполняемые новыми командами ASM51; которые изменяют разряды индикации состояния в ЗУПВ (команды битовой адресации упрощают и обычно укорачивают программу).

Телефон 455-57-81, Москва

Статья поступила 13.04.88

УДК 681.8.06

Н. Н. Божко, В. Г. Каневский, И. С. Мостов, Е. Е. Трилесник

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МЕЖМАШИННОГО ОБМЕНА МИКРОЭВМ СМ1800

Для микроЭВМ СМ1800 разработан ряд программ межмашинного обмена, реализованных в рамках различных ОС [1, 2]. Пакеты программ FTRANS (ДОС1810), ММК СМ (ОС1800), ММК СМ16 (МДОС1810), СЕТЬ МИНИ (ОС1800, МДОС1810) предназначены для микроЭВМ, работающих в режиме интеллектуального терминала или выполняющих функции файлового обмена. В распределенных системах, осуществляющих обмен данными между задачами, доступ к удаленным периферийным устройствам, удаленный файловый доступ (чтение-запись), работают драйвер управления информационным каналом (ОС СФП, ОС СФП1810) и пакет ПО ИЛПС-2 (ОС СФП).

В отдельных случаях разрабатывается специализированное ПО, например универсальный программируемый абонентский пункт (УПАП) [3], созданный на базе СМ1800 для подключения ЕС7920 к рабочим системам сети в протоколах X.25, и неоднородная информационно-вычислительная сеть ГАЛС [4], объединяющая ЕС ЭВМ и СМ ЭВМ.

Программа межмашинного обмена в ДОС1810

Для обмена файлами между микроЭВМ под управлением ДОС1810 разработана программа передачи файлов FTRANS. Программа работает в дуплексном режиме в диалоге с оператором. Протокол обмена данными несимметричен, поэтому программа имеет режимы задатчика и исполнителя.

В режиме задатчика программа принимает и выполняет команды передачи и приема файлов, выхода из программы. В режиме исполнителя — команды, принимаемые по каналу связи. Выход из программы возможен после принятия команды выхода от программы-задатчика.

Программа FTRANS обслуживает многоканальный модуль последовательных интерфейсов МИРПСМ [5]. Скорость обмена до 9600 бит/с. Программу можно также использовать на СМ1800 под управлением ДОС1800.

Пакеты межмашинного обмена в ОС1800 и МДОС1810

Наиболее широкое распространение в непромышленной сфере получили ОС СР/М для 8-разрядных и MS DOS для 16-разрядных микроЭВМ. Для них разработан ряд пакетов прикладных программ, обеспечивающих обмен данными.

Системы ПО для организации распределенных многомашинных комплексов на базе СМ ЭВМ с интерфейсами И41 и ОШ (ММК СМ и ММК СМ16) обладают сходными характеристиками и предназначены для подключения СМ1800 с ОС1800 (ММК СМ) и СМ1810 с МДОС1810 (ММК СМ16) к мини-ЭВМ семейства СМ 3/4. МикроЭВМ подключается к последовательному каналу мини-ЭВМ, обслуживаемому драйвером терминала ОС. Пакеты ММК СМ и ММК СМ16 обеспечивают эмуляцию терминала и файловый обмен.

В режиме эмуляции терминала пользователь микроЭВМ фактически работает за локальным терминалом мини-ЭВМ, ему доступны все команды и ресурсы мини-ЭВМ. Никаких дополнительных требований к ПО мини-ЭВМ в режиме эмуляции терминала не предъявляется.

При вызове программы файлового обмена микроЭВМ на мини-ЭВМ автоматически запускается соответствующая программа. В режиме диалога пользователь задает имена исходного файла, файла назначения и направление обмена. Программы файлового обмена для пакетов ММК СМ и ММК СМ16 реализованы в ОС РВ и РОС РВ для мини-ЭВМ семейства СМ3/4. Пакеты обслуживают коммуникационные модули МИРПС, МСТ, МСМ [5].

Для многомашинных комплексов более сложной структуры можно использовать системное ПО СЕТЬ МИНИ, имеющее функции, аналогичные рассмотренным выше. Преимуществом пакета СЕТЬ МИНИ является то, что под его управлением можно установить канал связи микроЭВМ с любой из ЭВМ, входящих в комплекс. К недостаткам пакета следует отнести более сложную систему команд, затрудняющую работу оператора.

Под управлением системы ПО кольцевой локальной сети СПО КОЛОС могут работать СМ1800 и СМ1810 в рамках ОС1800 и МДОС1810 в составе локальной вычислительной сети кольцевого типа на базе технических средств СЛК СМ [6]. СПО КОЛОС обеспечивает эмуляцию терминала любого узла локальной сети, файловый обмен с любым узлом.

Для подключения микроЭВМ СМ1810 к региональным сетям на базе пакетов программ СЕТЬ СМ ТРАЛ или локальной сети магистрального типа на базе ПО МАГИСТР предназначен пакет программ РЕЛОКС16 (МДОС1810).

Драйвер управления информационным каналом

В системах реального времени часто возникает необходимость в реализации обмена данными между задачами пользователя различных ЭВМ. Такие системы в основном представляют собой однородные многомашинные комплексы, включающие две или более ЭВМ с жестко выделенными связями. Применение всех уровней протоколов эталонной модели открытых систем [7] для них явно избыточно.

В таких системах целесообразно использовать драйвер управления информационным каналом [8], который фактически реализует протокол второго уровня эталонной модели. Драйвер устанавливает и разъединяет канал связи и выполняет обмен данными типа буфер-буфер в прозрачном режиме. Диспетчеризация данных обеспечивается задачами пользователя. Драйвер может быть использован как непосредственно для обмена данными, так и для построения более сложных коммуникационных систем. Он включен в состав ОС СФП [1] для СМ1800 и ОС СФП1810 [2] для СМ1810 и СМ1814. Версии драйвера поддерживают широкий спектр коммуникационных модулей, реализующих различные интерфейсы:

МИРПР — интерфейс радиальный параллельный;

МИРПС — интерфейс радиальный последовательный (токовая петля 20 мА);

МСТ — интерфейс радиальный последовательный (20/40 мА);

МИРПСМ — 4-канальный интерфейс радиальный последовательный (20 мА);

МСМ — стык С2.

Программное обеспечение локальной сети ИЛПС-2

Если в соответствии с требованиями прикладной системы пользователя необходимо объединить несколько микроЭВМ в локальную вычислительную сеть, то такую систему можно построить на базе модулей ИЛПС-2. Программная поддержка осуществляется ПО локальной сети ИЛПС-2 (ПО ИЛПС-2), работающим под управлением ОС СФП.

В состав ПО ИЛПС-2 входят пакеты виртуальной передачи данных и удаленной загрузки программ, прикладные компоненты локальной сети.

Пакет виртуальной передачи данных через ИЛПС-2 (ПВПД) дает возможность процессам, выполняющимся на различных ЭВМ сети, взаимодействовать друг с другом. Он организован по принципу иерархического соединения открытых систем, что позволяет настроить его без существенных затрат на другие существующие и проектируемые сетевые аппаратные средства. В настоящее время заканчивается работа по реализации ПВДП в среде ОС СФП1810 и БОС1810 [2] для 16-разрядных ЭВМ СМ1810 и СМ1814.

Пакет удаленной загрузки программ через ИЛПС-2 (ПУЗП) позволяет загружать программы с дисковых накопителей одной ЭВМ в оперативную память удаленных ЭВМ, связанных с ней модулями ИЛПС-2, и передавать этим программам управление. Возможны следующие режимы ПУЗП.

Режим загрузки (или дозагрузки). В нем к уже существующим процессам на удаленной ЭВМ добавляются новые, которые запускаются и выполняются, конкурируя с существующими за ресурсы на основе приоритетов.

Режим перезагрузки. В этом режиме ранее выполняющиеся на удаленной ЭВМ процессы уничтожаются, а вновь загружаемые запускаются и выполняются. Использование ПУЗП в локальной вычислительной сети позволяет иметь только одну ЭВМ, снабженную дисковыми накопителями.

Компоненты прикладного уровня локальной сети ИЛПС-2 функционируют как следующие серверы виртуальных устройств:

удаленного вывода на терминал;

удаленного вывода на печатающее устройство;

удаленного доступа к дисковому накопителю;

универсального многофункционального сервера обслуживания устройств аналогового и дискретного вводов, позволяющих осуществить прием и первичную обработку технологической информации с удаленных ЭВМ.

Телефон 455-57-81, Москва

ЛИТЕРАТУРА

1. Кравченко В. С. Обзор операционных систем микроЭВМ СМ1800 и СМ1810.— В кн.: Модели микроЭВМ семейства СМ1810.— М.: ИНЭУМ, 1987.— С. 83—89.
2. Азаров А. Д., Кравченко В. С., Новик А. Г., Бабанов И. И. Операционные системы микроЭВМ СМ1810.— См. наст. номер.— С. 73.
3. Богомолов Л. П., Коробков Б. Л., Нестеров М. А., Рацин Ю. В. Программное обеспечение универсального программируемого абонентского пункта на базе СМ1800.— В кн.: Модели микроЭВМ семейства СМ1810.— М.: ИНЭУМ, 1987.— С. 131—140.
4. Кириллов Н. И., Кудинова Н. В., Шемелин С. А. Информационно-вычислительная сеть на базе микроЭВМ семейства СМ1800.— Там же.— С. 12—17.
5. Гревцев В. В. Средства передачи данных микроЭВМ семейства СМ1800 // Микропроцессорные средства и системы.— 1988.— № 2.— С. 41.
6. Колосков М. С., Кузнецов А. Л., Кожевников Ю. Б. Локальная сеть микро- и мини-ЭВМ // Микропроцессорные средства и системы.— 1988.— № 2.— С. 43.
7. Якубайтис Э. А. Архитектура вычислительных сетей.— М.: Статистика, 1980.— 288 с.
8. Божко Н. Н., Каневский В. Г. Реализация протокола управления информационным каналом для СМ1800 // Сб. науч. тр.— Вып. 105.— М.: ИНЭУМ.— 1984.— С. 24—27.
9. Трилесник Е. Е. Базовое программное обеспечение виртуальной передачи данных для локальной сети на базе ЭВМ СМ1800.— В кн.: Вопросы создания единых технико-программных комплексов для АСУ ТП.— М.: ИНЭУМ, 1986.

Статья поступила 13.04.88

УДК 681.3.06

Ю. Е. Квадратов, А. М. Клейнберг, С. В. Припоров

РАСШИРЕНИЕ ДИСКОВОГО ПРОСТРАНСТВА МИКРОЭВМ СМ1800

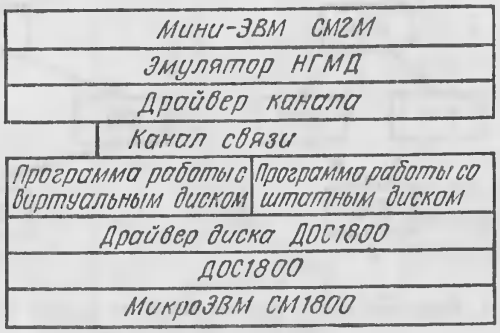
Одно из узких мест эксплуатации микроЭВМ СМ1800 — недостаточный объем внешней памяти на ГМД. Предлагаемое решение этой проблемы состоит в объединении микроЭВМ СМ1800 и мини-ЭВМ СМ2М и увеличении тем самым дискового пространства, одновременно доступного пользователю СМ1800.

Связь СМ1800 и СМ2М выполнена с помощью стандартных аппаратных средств: модуля связи с ИРПР СМ1800.7001 и согласователя интерфейсов ИРПР/2К при незначительной модификации согласователя для обеспечения прозрачности передачи данных по каналу связи.

В ДОС1800 ЭВМ СМ1800 изменен драйвер гибкого диска и добавлена программа работы с виртуальным диском — устройством, эмулирующим работу ГМД в формате файловой системы ДОС1800 и связанным с микроЭВМ каналом связи. Функции такого устройства может выполнять жесткий диск мини-ЭВМ, гибкий диск удаленной микроЭВМ, электронный диск и т. д. В данной системе виртуальные диски организованы в виде файлов на НМД СМ5400 СМ2М. Всего создано восемь таких дисков, так как ДОС1800 допускает работу максимум с десятью дисками (аппаратно к СМ1800 можно подключить только восемь дисков).

Для СМ2М написана программа эмуляции гибких дисков и драйвер канала связи, работающие в операционной системе ДОС АСПО. Общий объем разработанного программного обеспечения не превышает 8 Кбайт (по 4 Кбайт на каждую ЭВМ).

Пользователь в своем распоряжении имеет два диска на штатном НГМД СМ1800 емкостью 0,5 Мбайт с номерами :F0: и :F1: и восемь виртуальных дисков на НМД СМ5400 СМ2М суммарной емкостью 2 Мбайт с номерами :F2:...:F9:. Время доступа к виртуальному диску не больше, чем к гиб-



Взаимодействие программного обеспечения СМ2М и СМ1800

кому, а надежность гораздо выше. Работа с ними ничем не отличается от работы с НГМД.

Командная строка задания на трансляцию программы на Макроассемблере при работе СМ1800 в такой конфигурации имеет вид: :F0:ASM80:F2:FILE. ASM PRINT (:F3:FILE. LST) OBJECT (:F4:FILE. OBJ).

Конфигурацию системы можно изменить, например подключить вторую СМ1800 и расширить ПО коммуникационными функциями. В этом случае два пользователя получают возможность одновременно работать с виртуальными дисками; виртуальное дисковое пространство между ними распределяется соответствующей настройкой ПО ЭВМ СМ2М.

При дальнейшем наращивании числа микроЭВМ, подключаемых друг к другу по шинной топологии, получается простейшая локальная сеть, в которой микроЭВМ будут совместно использовать дисковое пространство СМ2М. Объединять вместе более восьми микроЭВМ нецелесообразно, так как с каждым новым подключением время доступа к виртуальному диску увеличивается.

Программы, обслуживающие виртуальные диски, располагаются в нулевом разделе памяти, поэтому работа с ними не сказывается на работе пользователей СМ2М (см. рисунок). В дальнейшем в состав виртуального дискового пространства предполагается включить диски всех микроЭВМ СМ1800 и создать гибкую распределенную файловую систему. Конфигурация дискового пространства каждой СМ1800 будет динамически определяться ее пользователем.

Телефон 7-54-75, Дзержинск Горьковской обл.

Сообщение поступило 12.05.88

УДК 681.326.34

Б. Г. Резников, А. У. Таратута

ПЕРЕКЛЮЧАТЕЛЬ ИНТЕРФЕЙСА ДЛЯ МИКРОЭВМ СМ1800

В состав комплексов на базе микроЭВМ СМ1800 могут входить блоки расширения (БР) СМ1800.0105, которые подсоединяются к базовой ЭВМ и друг к другу с помощью модуля расширения интерфейса (МРИ), состоящего из двух блоков элементов (БЭ) СМ1800/008 и СМ1800/010, связанных гибким кабелем и устанавливаемых соответственно в основной и добавочной частях интерфейса. Назначение БР — удлинение интерфейса, т. е. увеличение числа мест установки для модулей связи с периферией (одно место — один или два БЭ).

Система подключения БР к базовой ЭВМ может быть последовательной, радиальной или комбинированной, образуя дерево подключения. Одно из средств повышения надежности комплекса — резервирование всей или части вычислительной аппаратуры. Однако конструктивные и системные особенности СМ1800 и МРИ не позволяют подключать к БР более одного вызывающего БР (стоящего ближе к базовой машине, чем вызываемый), поэтому необходимо резервировать весь объем аппаратуры, что расточительно и может быть оправдано лишь в небольшом числе случаев.

Предлагается простое, но весьма эффективное аппаратно-программное средство, позволяющее резервировать практически любую часть дерева подключения. Переключение на резерв осуществляется по команде жизнеспособного процессора. Переключатель интерфейса (ПИ) позволяет подключать к любому БР два вызываемых БР или ЭВМ (рис. 1, 2).

Аппаратная часть ПИ представляет собой коммутатор (К) шин питания (5 В) на блоках элементов СМ1800/010, расположенных в БР и связанных с базовой ЭВМ или другими БР с помощью блока элементов СМ1800/008 и гибкого кабеля. ПИ состоит из двух частей: К1, К2, каждый выход включен последовательно между шиной БР (5 В) и токоведущими проводниками (5 В) двух БЭ СМ1800/010 (рис. 3). Такое исполнение МРИ позволяет отключать напряжение питания (5 В) усилителей-ретрансляторов по командам управления коммутатором. Каждая часть коммутатора, управляющая БЭ СМ1800/010, имеет четыре входа (по два от каждого вызывающего БР или ЭВМ).

Управляющие сигналы формируются программно и через стандартные мо-

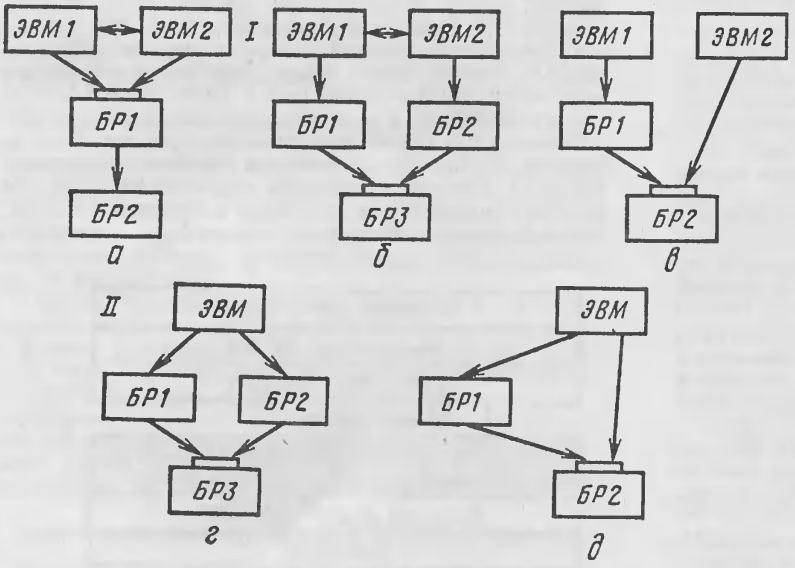


Рис. 1. Основные схемы переключения на резерв: I — полное восстановление функции системы:
а — резервирование только базовых ЭВМ; *б* — ЭВМ и одного из двух функционирующих БР; *в* — поддерживающий режим;
II — резервирование БР, связанное с жизненно важными функциями системы:
г — полное восстановление; *д* — подерживающий режим

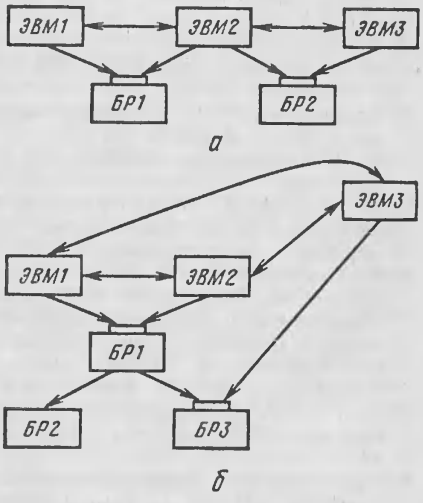


Рис. 2. Примеры схем подключения на резерв:
а — ЭВМ2 резервирует работу ЭВМ1 и ЭВМ2; *б* — система повышенной надежности ЭВМ1 резервирует ЭВМ2, а ЭВМ3 — ЭВМ1, ЭВМ2 и БР1, БР2 в поддерживающем режиме с одним функционирующим БР3

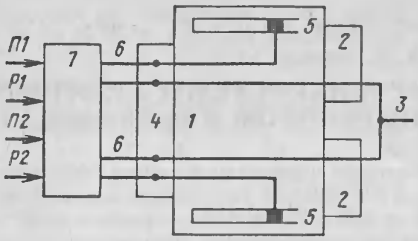


Рис. 3. Принципиальная схема соединения БЭ СМ1800/1810 и части коммутатора:

1 — плата, на которой смонтированы усилители, 2 — внутренние разъемы, 3 — шина, 4 — внешний разъем для подсоединения БР к ЭВМ, 5 — токоведущие проводники, 6 — выходы части коммутатора

дули вывода дискретных сигналов (МВД) СМ1800.9303 подаются на входы коммутатора (рис. 4, 5).

Каждый акт переключения представляет собой последовательность двух сигналов из резервной ЭВМ или БР: подготовительного (П) и рабочего (Р), а каждый сигнал соответствует единичному состоянию бита в выходном байте МВД.

Функции П-сигнала: разрыв цепи питания (5 В) в присоединенном БЭ СМ1800/010, блокирование Р-сигнала из ЭВМ до момента ее переключения, коммутация цепи питания в ранее отсоединенном БЭ СМ1800/010, подготовка цепей к появлению Р-сигнала из резервной ЭВМ (БР).

После выдержки определенной длительности П-сигнал снимается и в этом же МВД формируется Р-сигнал. Время выдержки П- и Р-сигналов определяется временем переходных процессов коммутации в цепях ПИ. Для переключателя, реализованного на реле РЭС-8, это промежуток времени не менее 100 мс, поэтому работа резерва может начаться не ранее, чем через 200 мс от начала процесса переключения.

Функции Р-сигнала: поддержка разрыва цепи питания в присоединенном ранее БЭ СМ1800/010 и ее коммутации в ранее отсоединенном БЭ СМ1800/010, блокирование Р-сигнала

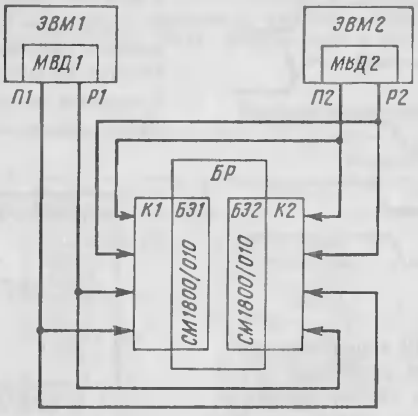


Рис. 4. Логическая сигнальная схема подключения ПИ к БР и двум базовым ЭВМ

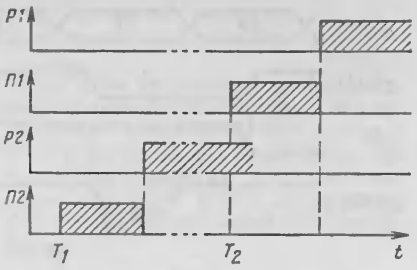


Рис. 5. Временна́я диаграмма сигналов.
T_1, T_2 — начало процесса переключения

из вызывающей ЭВМ (БР) до момента ее переключения; подготовка цепей к появлению П-сигнала из ЭВМ (БР), выходящей в резерв.

Программная часть ПИ для комплексов на базе СМ1800, генерирующая П- и Р-сигналы, реализована в операционной среде реального времени МОС РВ на языке программирования ПЛ/М в виде процедуры, которая может быть вызвана из прикладной части системы командой CALL. Программист может включить (отключить) вывод на дисплей диагностического сообщения о переключении на резерв, задать номер строки вывода на экране дисплея и включить (отключить) звуковое сопровождение переключения (звонок).

252133, г. Киев, Б-р Леси Украинки, 26, институт Гипросельмаш, отдел АСУ ТП; тел. 296-33-25

Статья поступила 24.02.88

КРАТКИЕ СООБЩЕНИЯ

УДК 621.374.334

С. Е. Сыстеров

ОСОБЕННОСТИ АДРЕСНОГО СЕЛЕКТОРА В СТАНДАРТЕ МПИ

При создании внешних устройств микроЭВМ типа «Электроника 60» разработчики часто в селектор адреса устанавливают регистр (триггер) хранения сигнала выборки устройства (ВБ), срабатывающий по фронту сигнала К СИА Н (рис. 1). В программном режиме обмена такой селектор не создает конфликтной ситуации в системной шине (рис. 2, *а*): данные D1 и D2 не могут появиться одновременно, несмотря на то, что сигнал ВБ снимается только в следующем цикле, поскольку канальные сигналы ввода-вывода разнесены во времени со стробом адреса.

Совсем иная ситуация возникает, если устройство с описанным селектором используется в микроЭВМ совместно с контроллерами, способными работать в режиме прерывания программы (рис. 2, *б*). Допустим, что во время обмена с устройством в системную шину направляется сигнал требования прерывания К ТПР Н. В следующем цикле микроЭВМ ответит установкой сигналов К ВВОД Н и К ППР Н, читая тем самым вектор прерывания. Известно, что при вводе вектора сигнал К СИА Н остается пассивным, а, следовательно, сигнал ВБ не сбрасывается. Таким образом, в шину К АД Н одновременно будут выставлены вектор прерывания и данные D1 предыдущего цикла. Поскольку такая ситуация возникает редко, а вектор искажается неоднозначно, то и останов по любому адресу воспринимается как сбой по питанию или помехе.

Сбои подобного рода можно устранить, если для запоминания сигнала ВБ использовать потенциальные регистры К155ТМ5, КМ555ТМ7. Можно применять также регистры, срабатывающие по перепаду импульса на тактовом входе, но для этого сигнал ВБ необходимо стробировать ак-

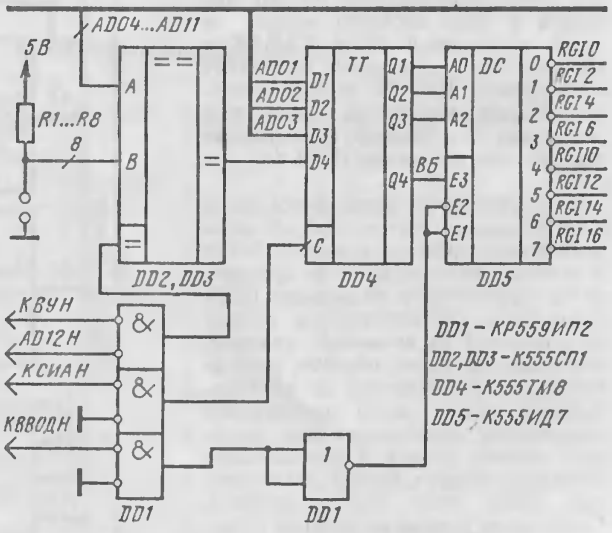


Рис. 1. Схема селектора адреса

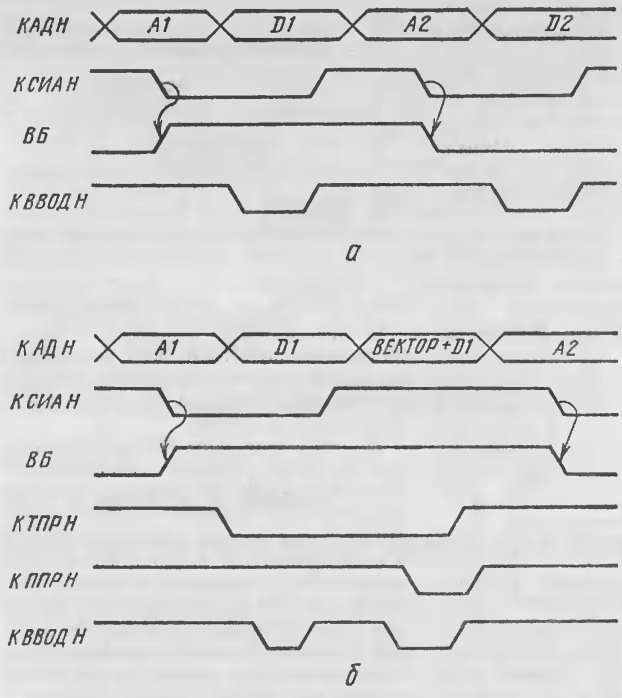


Рис. 2. Временны́е диаграммы цикла ВВОД в программном режиме обмена (*а*) и при возникновении конфликта (*б*)

тивным уровнем К СИА Н или обнулять регистр пассивным уровнем К СИА Н.

Телефон 49-32-59, Пермь

Сообщение поступило 1.11.88

УДК 681.325

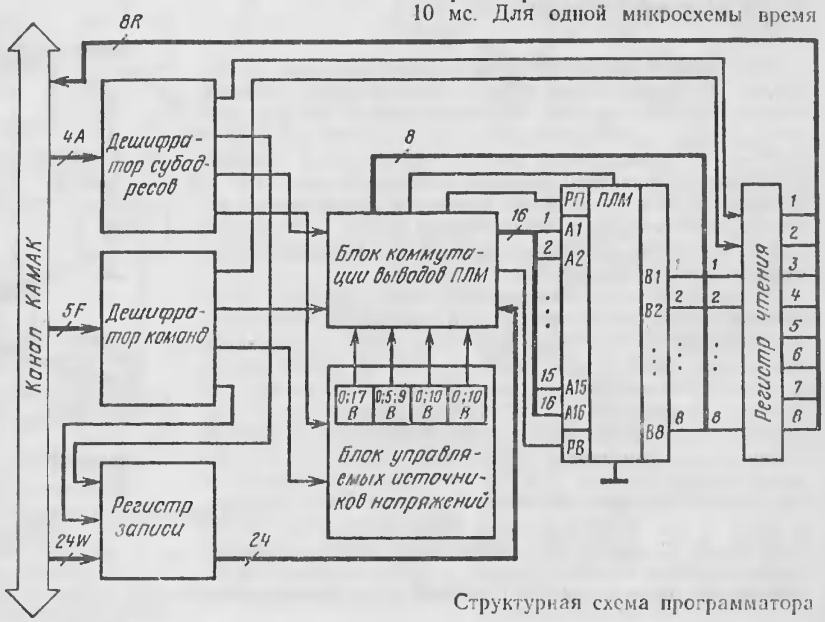
А. Ф. Хворостухин, Т. И. Кокшарова

ПРОГРАММАТОР ЛОГИЧЕСКИХ МАТРИЦ

Программатор предназначен для программирования ПЛМ К556РТ1, автоматического контроля «чистых» и чтения запрограммированных матриц. Выполнен в виде двойного модуля на одной стандартной плате КАМАК и работает под управлением микроЭВМ «Электроника 60». На передней панели модуля размещены контактное устройство и индикатор правильности подключения микросхем ПЛМ (см. рисунок).

В программаторе используется набор КАМАК-команд, достаточный для автоматического прожига и чтения ПЛМ. В ответ на каждую команду программатор устанавливает на выводах ПЛМ напряжения, соответствующие одному из положений на временны́х диаграммах прожига. Таким образом, для совершения элементарного прожига достаточно задать номер прожигаемой конъюнкции, дизъюнкции или активного низкого уровня и определенную последовательность команд, число которых равно числу этапов прожига. Содержание команд не зависит от номера прожигаемой конъюнкции, дизъюнкции или активного низкого уровня, что позволяет выделить разрешенные команды из набора КАМАК-команд и аппаратно запретить выполнение неразрешенных.

Время прожига составляет не более 10 мс. Для одной микросхемы время



Структурная схема программатора

УДК 681.323

П. Н. Васильев, В. И. Рябков

СОПРЯЖЕНИЕ МИКРОСХЕМЫ КР580ВН59 С МИКРОПРОЦЕССОРОМ КР580ВМ80А

Отсутствие информации о совместной работе БИС контроллера прерываний КР580ВН59, работающей в режиме запроса прерывания, с МП КР580ВМ80А, вызывает некоторые трудности при их одновременном применении.

При отсутствии микросхемы КР580ИР82 (КР580ИР83) необходимо на вход $\overline{RD}$ микросхемы КР580ВН59 подать сигнал $\overline{DBIN}\cdot INP$ (INP — разряд 6 байта состояния), на вход $\overline{INTA}$ — сигнал $\overline{DBIN}\cdot INTA$, где INTA — разряд 0 байта состояния. Сигнал на выходе INT сбрасывается в нулевое состояние по окончании второго сигнала $\overline{INTA}$. Это совпадает по времени с фронтом импульса Ф2. Анализ наличия сигнала прерывания в МП КР580ВМ80А осуществляется по спаду этого же импульса, что приводит к прекращению выдачи сигналов $\overline{INTA}$. В результате на вход $\overline{INTA}$ вместо трех импульсов поступают два.

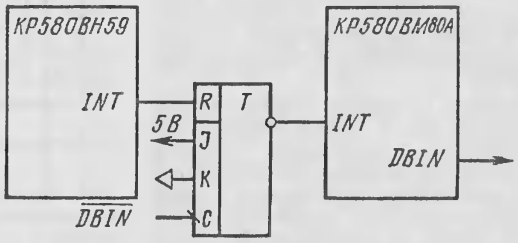


Схема удлинения сигнала INT

Для восстановления правильной работы необходимо удлинить либо сигнал $\overline{INTA}$, либо сигнал INT на время, равное или большее длительности импульса Ф2.

Телефон 33-51-35, Минск *Сообщение поступило 20.09.88*

прожига не превышает 5 мин, время чтения — 10 с. Средняя рассеиваемая мощность — не более 13 Вт. Для работы программатора используются следующие напряжения питания крейта КАМАК: 6 В, 0,8 А; 12 В, 110 мА; —12 В, 3 мА; 24 В, 75 мА. Дополнительных источников питания не требуется.

Неправильно подключенная матрица не прожигается и не повреждается. Предусмотрена защита от сбоев питания и программы.

Работой программатора управляют дешифратор команд и дешифратор субадресов. Дешифратор команд преобразует КАМАК-команду в управляющие сигналы, а дешифратор субадресов определяет, для какого блока предназначена команда. Блок коммутации выводов ПЛМ подключает соответствующие выводы ПЛМ к блоку управляемых источников питания, который, в свою очередь, вырабатывает необходимые напряжения прожига. В регистре записи запоминается номер конъюнкции, дизъюнкции или программируемой функции. При поэлементном контроле информация о состоянии выходов ПЛМ запоминается в регистре чтения, а затем передается в канал КАМАК.

Программное обеспечение программатора реализовано на языках Паскаль и ассемблер, имеет оверлейную структуру из четырех подуровней и работает под управлением ОС RT-11, ОС РАФОС. Программно обеспечиваются ввод исходных данных, формирование и редактирование таблицы истинности логических функций, запись функций в файл, поэтапный (три этапа) прожиг ПЛМ, чтение ПЛМ, печать таблиц.

На этапе 1 прожига проверяется целостность пережигаемых перемычек в ПЛМ, сравниваются данные таблицы истинности логических функций с состоянием перемычек ПЛМ. На терминал выдаются координаты дефектной перемычки. При успешном завершении входного контроля микросхема считается годной для программирования.

На этапе 2 в процессе каждого элементарного прожига осуществляется контроль результата программирования. Если результат отрицательный, то прожиг повторяется (до семи попыток) с увеличением времени воздействия. Результатом успешного завершения прожига микросхемы будет создание в ОЗУ таблицы истинности запрограммированной ПЛМ. На этапе 3 проверяется соответствие таблиц истинности логических функций и запрограммированной ПЛМ. Программатор успешно эксплуатируется в течение двух лет.

660049, Красноярск, пр. Мира, 53, СКТБ «Наука» СО АН СССР; тел. 27-49-16, 27-72-62

Статья поступила 28.12.88

ПИСЬМО В РЕДАКЦИЮ

В журнале «Радио» был цикл статей, посвященных микроЭВМ на основе МП К580ИК80А. Особенно мне понравилась табличка с полным перечнем команд МП и шестнадцатеричными машинными кодами.

У меня к Вам просьба: поместите, пожалуйста, на страницах своего журнала похожую табличку с тем же «обрамлением» из кодов машинных команд, но с другой «начинкой». В клетках таблицы должны быть проставлены: время выполнения соответствующей команды, измеренное, скажем в тактах ГТИ, имена регистров (в том числе аккумулятора МП), информация в которых после выполнения команды становится отличной от той, что была в них до начала выполнения этой команды. И то и другое весьма существенно при разработке программ в мнемо- или машинных кодах; и то и другое не во всех случаях легко выводится «умозрительно».

В. Н. Ружинский, Москва

По просьбе читателей мы публикуем таблицу команд МП К580ИК80А, подготовленную по заказу редакции нашим консультантом Ю. И. Торговым.

Таблица 1

Команды микропроцессора КР580ИК80А

Команды пересылки

	Мнемокод	Операция	Ц	Т	Ф	Признаки	Регистры
1	MOV R1, R2	(R1)<—(R2)	1	5	1	Нет	R1
2	XCHG	(HL)<>(DE)	1	4	1	»	H, L, D, E
3	SPHL	(SP)<—(HL)	1	5	1	»	SP
4	MOV R, M	(R)<—M(HL)	2	7	1	»	R
5	MOV, M, R	M (HL)<—(R)	2	7	1	»	Нет
6	LDAX RP'	(A)<—M(RP)	2	7	1	»	A
7	STAX RP'	M(RP)<—(A)	2	7	1	»	Нет
8	ЛDA A16	(A)<—M(A16)	4	13	3	»	A
9	STA A16	M(A16)<—(A)	4	13	3	»	Нет
10	LHLD A16	(HL)<—M(A16)	5	16	3	»	H, L
11	SHLD A16	M(A16)<—(HL)	5	16	3	»	Нет
12	MVI R, D8	(R)<—D8	2	7	2	»	R
13	LXI RP, D16	(RP)<—D16)	3	10	3	»	RP
14	MVI M, D8	M(HL)<—D8	3	10	2	»	Нет
15	PUSH RP"	M(SP—1)<—(RPH) M(SP—2)<—(RPL) (SP)<—(SP)—2	3	11	1	»	Нет
16	POP RP"	(RPL)<—M(SP) (RPH)<—M(SP+1) (SP)<—(SP)+2	3	10	1	»	RP
17	XTHL	M(SP)<>(L) M(SP+1)<>(H)	5	18	1	»	H, L
18	IN PORT	(A)<1(POPT)	3	10	2	»	A
19	OUT PORT	0(PORT)<—(A)	3	10	2	»	Нет

Команды арифметических операций

20	ADD R	(A)<—(A)+(R)	1	4	1	S, Z, AC, P, CY	A
21	ADC R	(A)<—(A)++(R)+CY	1	4	1	S, Z, AC, P, CY	A
22	SUB R	(A)<—(A)—(R)	1	4	1	S, Z, AC, P, CY	A
23	SBB R	(A)<—(A)——(R)—CY	1	4	1	S, Z, AC, P, CY	A
24	INR R	(R)<—(R)+1	1	5	1	S, Z, AC, P	R
25	DCR R	(R)<—(R)—1	1	5	1	S, Z, AC, P	R
26	DAD RP	(HL)<—(HL)++(RP)	3	10	1	CY	H, L
27	INX RP	(RP)<—(RP)+1	1	5	1	Нет	RP
28	DCX RP	(RP)<—(RP)—1	1	5	1	Нет	RP
29	ADD M	(A)<—(A)++M(HL)	2	7	1	S, Z, AC, P, CY	A
30	ADC M	(A)<—(A)++M(HL)+CY	2	7	1	S, Z, AC, P, CY	A
31	SUB M	(A)<—(A)——M(HL)	2	7	1	S, Z, AC, P, CY	A
32	SBB M	(A)<—(A)——M(HL)—CY	2	7	1	S, Z, AC, P, CY	A

Продолжение таблицы 1

33	INR M	M(HL)<—M(HL)+1	3	10	1	S, Z, AC, P	Нет
34	DCR M	M(HL)<—M(HL)—1	3	10	1	S, Z, AC, P	Нет
35	ADI D8	(A)<—(A)+D8	2	7	2	S, Z, AC, P, CY	A
36	ACI D8	(A)<—(A)+D8+CY	2	7	2	S, Z, AC, P, CY	A
37	SUI D8	(A)<—(A)—D8	2	7	2	S, Z, AC, P, CY	A
38	SBI D8	(A)<(A)—D8—CY	2	7	2	S, Z, AC, P, CY	A
39	DAA	Десятичная коррекция	1	4	1	S, Z, AC, P, CY	A

Команды логических операций

40	ANA R	(A)<--(A) AND(R)	1	4	1	S, Z, P, AC', CY=0	A
41	XRA R	(A)<—(A)XOR(R)	1	4	1	S, Z, P, AC=CY=0	A
42	ORA R	(A)<—(A)OR(R)	1	4	1	S, Z, P, AC=CY=0	A
43	CMP R	(A)<—(R)	1	4	1	S, Z, P, CY	Нет
44	RLC	Сдвиг влево цикл. (An+1)<—(An) (AO)<—(A7)	1	4	1	CY<--A(7)	A
45	RRC	Сдвиг вправо цикл. (An)<—(An—1) (A7)<--(A0)	1	4	1	CY<—A(0)	A
46	RAL	Сдвиг влево цикл. через перенос (An+1)<—(A) (A0)<—CY	1	4	1	CY<—A(7)	A
47	RAR	Сдвиг вправо цикл. через перенос (An)<—(An+1) (A7)<—CY	1	4	1	CY<—A(0)	A
48	CMA	(A)<—INV(A)	1	4	1	Нет	A
49	ANA M	(A)<—(A)AND M(HL)	2	7	1	S, Z, P, AC, CY=0	A
50	XRA M	(A)<—(A)XOR M(HL)	2	7	1	S, Z, P, AC=CY=0	A
51	ORA M	(A)<—(A)OR M(HL)	2	7	1	S, Z, P, AC=CY=0	A
52	CMP M	(A)<—M(HL)	2	7	1	S, Z, P, AC, CY	Нет
53	ANI D8	(A)<—(A) AND D8	2	7	2	S, Z, P, AC', CY=0	A
54	XRI D8	(A)<—(A) XOR D8	2	7	2	S, Z, P, AC=CY=0	A
55	ORI D8	(A)<—(A) ORI D8	2	7	2	S, Z, P, AC=CY=0	A
56	CPI D8	(A)<—D8	2	7	2	S, Z, P, AC, CY	Нет
57	CMC	(CY)<—INV(CY)	1	4	1	CY	Нет
58	STC	(CY)<—1	1	4	1	CY=1	Нет

Команды передачи управления

59	PCHL	(PCH)<—(H) (PCL)<—(L)	1	5	1		PCH, PCL
60	IMP A16	(PC)<—A16	3	10	3	Все признаки сохраняют свои значения	
61	I(COND)	A16.Если условие выполняется, то (PC)<—A16, иначе переход к след. ком.	3	10	3		
62	CALL A16	M(SP—1)<—(PCH) M(SP—2)<—(PCL) (SP)<—(SP)—2 (PC)<—A16	5	17	3		

УДК 681.3.06

Н. И. Бубело

ЯЗЫК ПРОГРАММИРОВАНИЯ ДЛЯ МИКРОЭВМ

Для самых массовых 8-разрядных микропроцессоров (К580ВМ80 и Z80) реализовано около двух десятков различных языков. Такое богатство инструментальных возможностей находится в ошеломляющем противоречии с тем фактом, что абсолютное большинство сколько-нибудь «серьезных» программ пишется на языке ассемблера. Чем же объяснить массовое применение ассемблера, этого «динозавра», упорно не желающего исчезать из джунглей программирования? Разумеется, высокой эффективностью, прежде всего, и малым объемом занимаемой памяти. Программисты идут на это, жертвуя преимуществами языков высокого уровня. Наиболее близок к ассемблеру из таких языков ПЛ/М, но и он не всегда удовлетворяет программистов из-за значительного расхода памяти. Для некоторых задач программы на ПЛ/М занимают в 3,5...5 раз бо́льший объем памяти, чем соответствующие программы на ассемблере. Таким образом, между ассемблером и ПЛ/М просматривается свободная «экологическая ниша» для еще одного языка программирования.

Исходные требования при разработке языка — приблизить его к ассемблеру по эффективности; сохранить удобство записи и чтения, предоставляемые ПЛ/М и другими языками высокого уровня.

Источник неэффективного использования памяти — пакеты библиотечных подпрограмм, подключаемые к скомпилированной программе на этапе компоновки из-за того, что набор операций и данных исходного языка не соответствует системе команд процессора. Следовательно, **для исключения библиотечных подпрограмм необходимо ограничить набор операций языка лишь теми, которые имеют аналоги в системе команд целевого процессора.** Но язык не должен быть всего лишь структурированным ассемблером: необходимо обеспечить выполнение реализуемых операций для всех используемых типов данных, а также сохранить четкие и ясные управляющие структуры. Эти положения — основа данной разработки.

Реализовать требуемый язык можно было бы на основе старой идеи «макро», но при таком подходе язык мог бы оказаться недостаточно надежным. Поэтому был написан полномасштабный компилятор, причем в качестве целевого был выбран процессор КР580ВМ80.

В языке используются данные двух типов: байты и двухбайтовые слова. Есть возможность инициализации данных, литерального объявления констант

(Продолжение см. на с. 89)

Окончание таблицы 1

63	C(COND) A16	Если условие выполняется, то см. ком. 62, иначе переход к след. ком.	5 3	17 11	3 3	
64	RST N	M(SP—1)<—(PCH) M(SP—2)<—(PCL) (SP)<—(SP)—2 (PC)<—8*(N)	3	11	1	
65	RET	(PCL)<—M(SP) (PCH)<—M(SP+1) (SP)<—(SP)+2	3	10	1	
66	R(COND)	Если условие выполняется, то см. ком. 65, иначе переход к след. ком.	1	5	1	

Специальные команды

67	EI	Разрешить прерывание (триггер РПР)<—1	1	4	1	Все признаки сохраняют свои значения	Нет
68	DI	Запретить прерывание (триггер РПР)<—0	1	4	1		
69	HLT	Останов	1	7	1		
70	NOP	Пустая операция	1	4	1		

Принятые обозначения:

< — операция пересылки, < > — операция обмена, AND — конъюнкция (И); OR — дизъюнкция (ИЛИ); XOR — сложение по модулю 2 (исключающее ИЛИ); INV — инверсия;
R — один из семи регистров: 7/A, 0/B, 1/C, 2/D, 3/E, 4/H, 5/L;
RP — одна из регистровых пар: 0/B, 1/D, 2/H или 3/SP;
RP' — одна из регистровых пар: 0/B или 1/D;
RP" — одна из регистровых пар: 0/B, 1/D, 2/H или 3/PSW;
RPH, RPL — старший и младший регистры в регистровой паре;
M — память, адресуемая косвенно через HL;
PORT — восьмиразрядный адрес порта ввода-вывода;
N — один из восьми уровней PESTART: 0, 1, 2, 3, 4, 5, 6, 7;
D8 — восьмиразрядный непосредственный операнд;
A16 — шестнадцатиразрядный адрес;
(R), (RPi, M(RP) — содержимое регистра, регистровой пары и ячейки памяти по адресу, хранящемуся в регистровой паре RP соответственно;
I(PORT), O (PORT) — содержимое портов ввода и вывода с адресом PORT;
COND — одно из восьми условий: 0—NZ — ненулевой результат (Z=0); 1 — Z — нулевой результат (Z=1); 2—NC — отсутствие переноса из старшего разряда или заема в старший разряд (CY=0); 3—C — наличие переноса или заема (CY=1); 4—PO — нечетность числа единиц в результате (P=0); 5—PE — четность числа единиц в результате (P=1); 6—P — «плюс» (S=0); 7—M — «минус» (S=1);
Ц — число машинных циклов, Т — число машинных тактов, Ф — формат команды в байтах;
D16 — шестнадцатиразрядный непосредственный операнд.

Таблица 2

Регистр признаков микропроцессора КР580ИК80А

Признак	S	Z	O	AC	0	P	1	CY
Разряд	7	6	5	4	3	2	1	0

S — признак «знака» (принимает значение старшего разряда результата);
Z — признак нуля (если результат равен нулю, то Z=1, иначе Z=0);
AC — признак вспомогательного переноса (если есть перенос между тетрадами байта, то AC=1, иначе AC=0);
P — признак четности (если число единиц в байте результата четно, то P=1, иначе P=0);
CY — признак переноса (если при выполнении команды возник перенос из старшего разряда или заем в старший разряд, то CY=1, иначе CY=0).
Примечание. Для команд логического умножения признак вспомогательного переноса AC принимает значение четвертого разряда результата (AC)<—A (3).

(Окончание. Начало см. на с. 88)

и портов ввода-вывода. Основные управляющие структуры — разветвление и цикл. Есть оператор выхода из цикла в любой точке. Отметим, что в языке отсутствует блочность и нет составных операторов, поэтому введены явные ограничители циклов и разветвлений. Базовый набор операций соответствует системе команд процессора и включает в себя сложение, вычитание, логические операции (И, ИЛИ, исключающее ИЛИ, отрицание), операции инкрементирования и декрементирования, преобразования типов, а также арифметические и циклические сдвиги. Можно работать с адресами и указателями. Все операции, за исключением операций сравнения и присвоения, записывают в префиксной форме в виде вызовов функций. Присвоения, как и вызов процедуры, могут быть отдельными операторами или входить в состав выражений.

Дополнительное средство — в языке предусмотрена возможность написания процедур на ассемблере (вызываемых как обычные процедуры), а также вставок ассемблерного текста в любом месте программы. Для сохранения доступа программиста к тексту программы на любом уровне компилятором в качестве объектного файла формируется текстовый файл на ассемблере.

Компилятор написан на языке Си в операционной среде CP/M. Значительный объем его (32 Кбайт) подтверждает неэффективность программ, написанных на таком языке, как Си, для 8-разрядного процессора с недостаточно развитой системой команд.

В серии экспериментов **по сравнению реализованного языка с ассемблером и ПЛ/М** для исследования были выбраны программы сортировки, поиска, обмена с внешними устройствами, простейшей интерпретации команд. Полученные программы превышали по объему их тщательно оптимизированные ассемблерные аналоги в среднем в 2...2,5 раза. Для ПЛ/М это же соотношение оказалось лежащим в диапазоне 3...4. Оценка уровня языка по критерию Холстеда дала значения, в 1,5...2 раза превосходящие этот показатель для ассемблера и практически не отличающиеся от значений уровня ПЛ/М.

Сфера применения разработанного языка — системное программирование: драйверы внешних устройств, фрагменты ОС, ПО микроконтроллеров и небольших систем реального времени, а также некоторые другие задачи. Нет необходимости создания подобных языков для 16-разрядных процессоров, на которых достаточно эффективно реализуются, например, Си или ПЛ/М. Но для 8-разрядных МП и однокристальных микроЭВМ такой язык достаточно удобен при создании понятных и эффективных программ.

Телефон 30-83-74, Днепропетровск

Ю. И. Торгов

Сообщение поступило 25.08.88

ОПЕРАТИВНОЕ ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО К565РУ7

Микросхема К565РУ7 представляет собой ОЗУ динамического типа с произвольной выборкой емкостью 262144 и организацией 262144×1 разряд, выполненное по n-канальной МОП-технологии. Число элементов на кристалле 831550. Условное графическое обозначение микросхемы дано на рис. 1, назначение выводов приведено в табл. 1, электрическая структурная схема представлена на рис. 2.

Накопитель ОЗУ организован в виде матрицы, содержащей по 512 строк и столбцов. Кроме того, в состав накопителя входят две сдвоенные четверки резервных строк и четыре пары резервных столбцов (всего 8256 резервных ячеек).

Схема управления представляет собой две цепочки тактирующих генераторов, которые включаются сигналами $\overline{RAS}$ и $\overline{CAS}$. Для выбора любой из 262144 ячеек памяти требуется 18-разрядный код. Адресный код подается на 9-разрядный адресный регистр в мультиплексном режиме: сначала девять

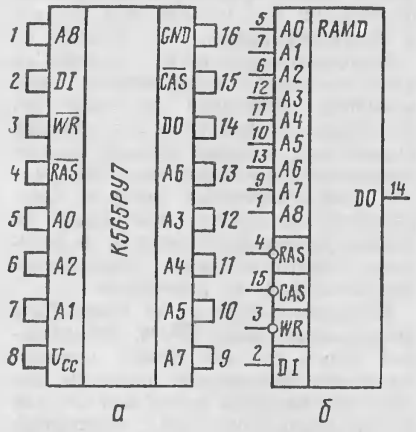


Рис. 1. Условное графическое обозначение СБИС К565РУ7 по порядку расположения (*а*) и функциональному назначению (*б*) выводов

Таблица 1

Назначение выводов

Вывод	Обозначение	Назначение
1	A8	Вход адреса
2	DI	Вход данных
3	WR	Запись
4	$\overline{RAS}$	Выборка строк
5...7	A0, A2, A1	Входы адреса
8	U_{CC}	Напряжение источника питания, 5 В
9...13	A7, A5...A3, A6	Входы адреса
14	DO	Выход данных
15	$\overline{CAS}$	Выборка столбцов
16	GND	Общий

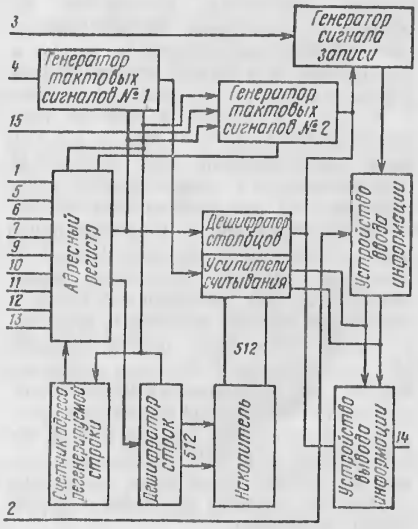


Рис. 2. Электрическая структурная схема ОЗУ К565РУ7

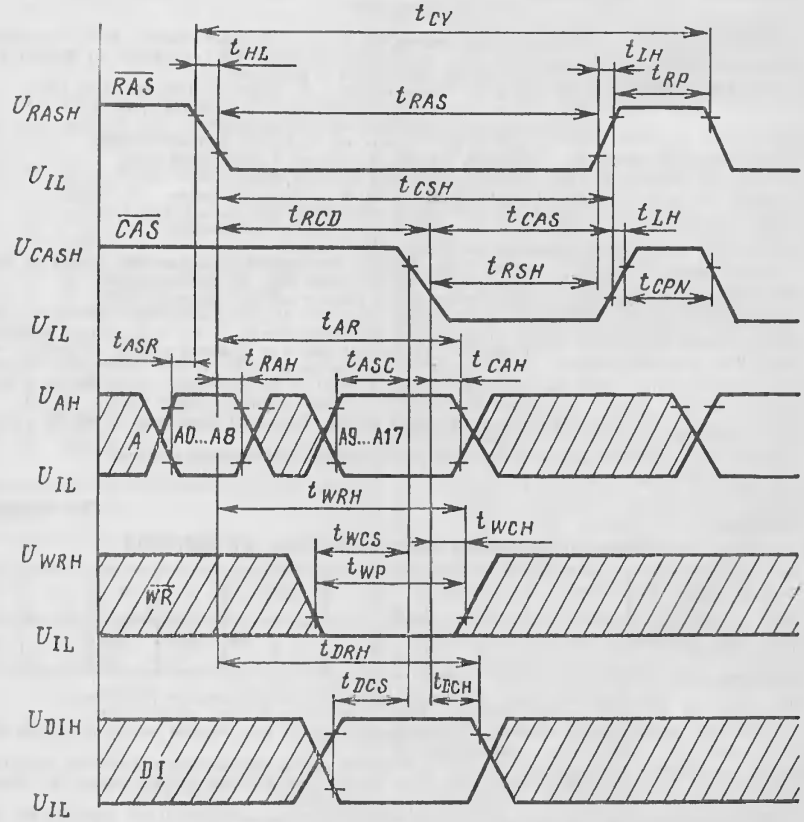


Рис. 3. Режим записи: заштрихованные поля соответствуют сигналам, имеющим произвольные состояния

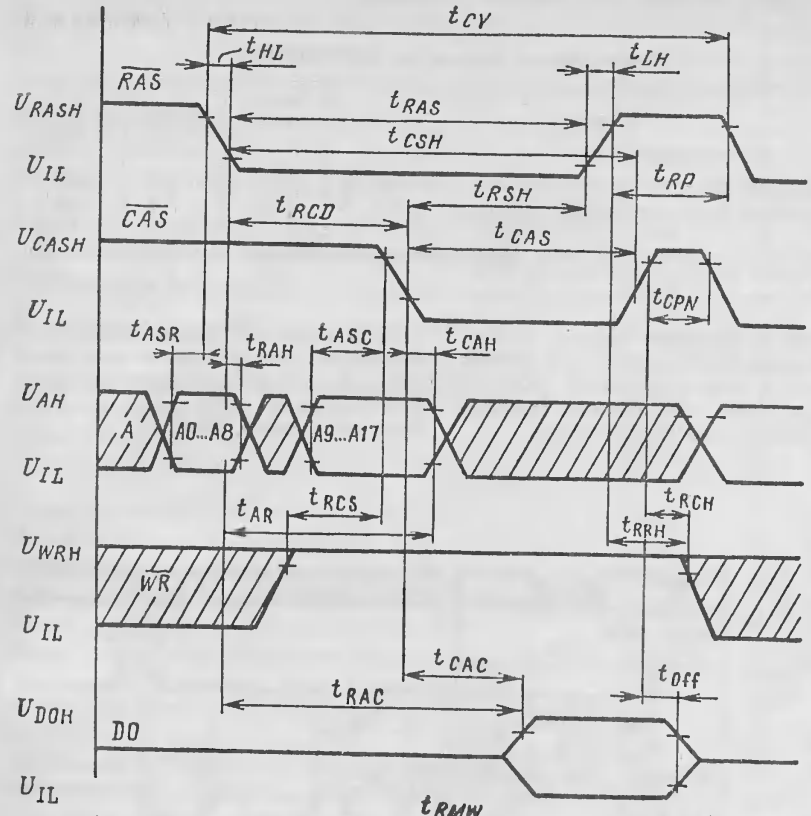


Рис. 4. Режим чтения

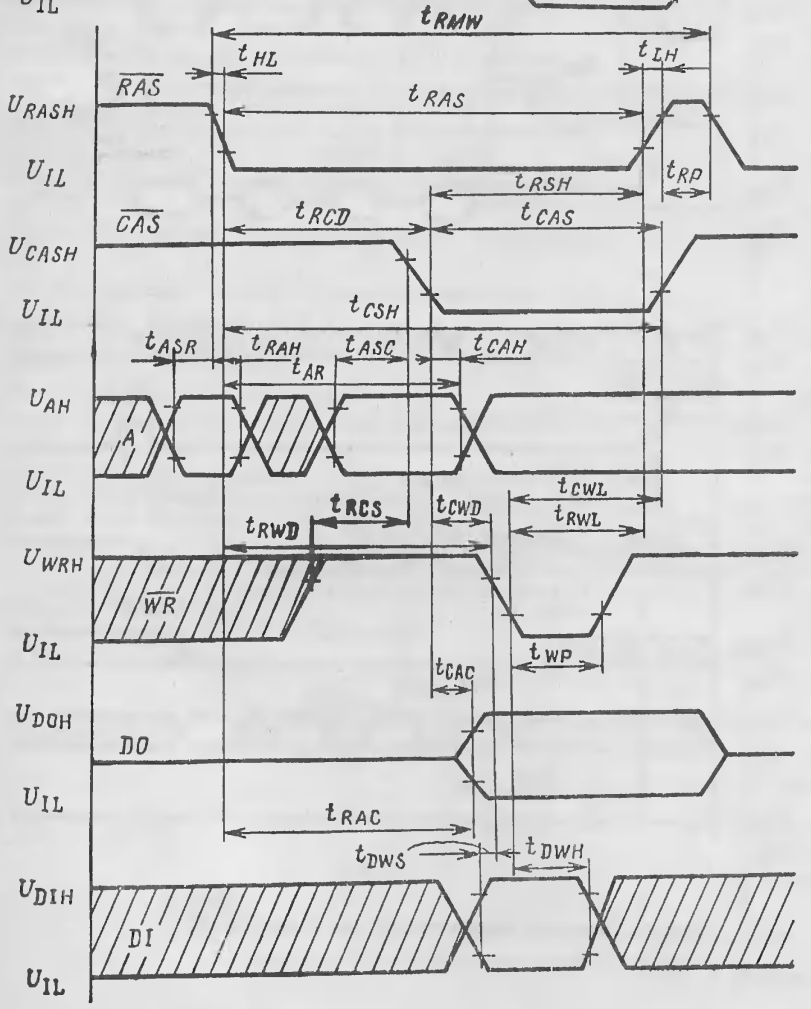


Рис. 5. Режим чтения-модификации-записи

младших разрядов адреса для выбора нужной строки, затем на те же выводы — девять старших разрядов, соответствующих адресу столбца. Код адреса строки фиксируется на адресном регистре с помощью сигнала $\overline{RAS}$ при переходе на нижний уровень, а код адреса столбца — с помощью сигнала $\overline{CAS}$. К моменту приема адресным регистром адреса столбца на выходе дешифратора строк фиксируется выбранная строка. Фиксация обеспечивается, пока сигнал RAS находится в активном состоянии (на нижнем логическом уровне).

Временные диаграммы работы СБИС К565РУ7 в основных режимах приведены на рис. 3...6, электрические режимы и временные параметры сигналов — соответственно в табл. 2 и 3, состояния управляющих входов показаны в табл. 4.

ОЗУ К565РУ7 работает в следующих режимах: чтения, записи (ранней записи), чтения-модификации-записи, записи полубайта (слоговом), чтения полубайта (слоговом), регенерации сигналом $\overline{RAS}$, регенерации при «$\overline{CAS}$ раньше $\overline{RAS}$».

В режиме чтения после срабатывания дешифратора столбцов на предварительные усилители чтения-записи одновременно выводятся четыре числа. Информация с усилителя, адрес которого определяется адресами А3, А6, выводится на выходной буфер.

В режимах чтения и записи время t_{RCD} имеет два критических значения (рис. 4, 5). Если оно меньше $t_{RCD\,min}$, то микросхема не работает, если больше $t_{RCD\,max}$, то правильное функционирование микросхемы не нарушается, однако время выборки t_{RAC} увеличивается на разницу между t_{RCD} и $t_{RCD\,max}$. Время выборки t_{CAC} при этом минимально.

При $t_{RCD\,min} \leqslant t_{RCD} \leqslant t_{RCD\,max}$ время выборки t_{RAC} остается неизменным. Время выборки t_{CAC} увеличивается на разность между $t_{RCD\,max}$ и t_{RCD}. В режиме записи входная информация DI воспринимается входным регистром-защелкой при переходе в активное состояние (Лог. 0) сигналов WR и $\overline{CAS}$, если при этом $\overline{RAS}$ находится на нижнем уровне.

Если сигнал $\overline{WR}$ переходит в активное состояние раньше $\overline{CAS}$ (ранняя запись), то выход микросхемы DO будет сохранять состояние высокого импеданса (выход имеет плавающий потенциал).

В режиме чтения сигнал $\overline{WR}$ поддерживается на высоком логическом уровне с момента подачи сигнала $\overline{CAS}$ до окончания режима. Считанная информация появляется на выходе с задержкой, соответствующей времени выборки, которое, в свою очередь, опре-

деляется задержками внутренних управляющих сигналов. При считывании вход DO переходит из состояния с высоким импедансом в активное состояние и сохраняет его до тех пор, пока сигнал $\overline{CAS}$ не перейдет в состояние с высоким логическим уровнем. Это обстоятельство позволяет использовать сигнал $\overline{CAS}$ в качестве сигнала выборки кристалла в блоке памяти.

Слоговый режим или режим ускоренного выбора полубайта позволяет считывать и записывать от двух до четырех последовательных битов информации. Первый бит считывается обычным способом. Адрес задается девятью строчными и девятью столбцовыми адресами, причем адреса столбцов A3 и A6 определяют выборку первого бита в четверке. Каждое последующее тактирование сигналом $\overline{CAS}$ при активном $\overline{RAS}$ (перевод $\overline{CAS}$ в пассивное состояние — Лог. 1, а затем в активное — Лог. 0) вызывает приращение внутренних адресов A3, A6 с помощью встроенного в микросхему счетчика. Первым получает приращение адрес A6, т. е. он будет младшим. Другие адреса остаются неизменными.

Если $\overline{CAS}$ переходил в активное состояние более четырех раз, то последовательность адресов A3, A6 повторяется и считывается та информация, которая записана в предыдущем обращении по данному адресу.

Слоговый режим стал возможным благодаря организации одновременного вывода из матрицы накопителя 4 бит на внутренний регистр. Цикл считывания слогового режима можно сократить, так как данные из внутреннего регистра на выходной буфер выводятся за два такта.

В режиме записи информация с входного буфера, стробируемого одновременным переходом в активное состояние сигналов $\overline{CAS}$ и $\overline{WR}$, к моменту появления сигнала выборки столбцов подается на один из четырех усилителей чтения-записи (в зависимости от кода адресов A3 и A6) и по шинам ввода-вывода через ключи столбцов записывается в матрицу памяти.

Информация, которую необходимо записать в выбранную ячейку, запоминается входным регистром при подаче на внешние выводы схемы сигналов $\overline{CAS}$ и $\overline{WR}$ при активном состоянии сигнала $\overline{RAS}$. Регистр входной информации функционирует по принципу защелки. Его состояние, соответствующее входной информации, фиксируется сигналами $\overline{CAS}$ и $\overline{WR}$ в момент прихода последнего по времени сигнала.

Схема управления мультиплексней построена таким образом, что на операцию мультиплексирования практически не требуется дополнительного времени в отличие от параллельной подачи кода адреса. Применение принципа мультиплексирования позволило при сохранении высокого быстродействия получить максимально высокую информационную плотность и разместить кристалл в стандартном 16-выводном корпусе. Устройства ввода-вывода служат для приема входной и выдачи считанной информации, а также обеспечивают стыковку ОЗУ с внешними устройствами.

Регенерация ОЗУ осуществляется обращением с интервалом 8 мс к каждой из 512 строк путем перебора строчных адресов A0...A8 в любом цикле работы микросхемы. Наиболее целесооб-

Таблица 2

Электрические параметры ОЗУ К565РУ7

Параметр	Обозначение	Норма	
		не менее	не более
Напряжение низкого уровня входных сигналов, В	U_{IL}	−0,6	0,6
Напряжение высокого уровня входных сигналов, В	U_{IH}	2,4	5,5

Примечание. Допустимые уровни управляющих сигналов указаны с учетом амплитудных значений помех в полосе частот не менее 100 МГц.

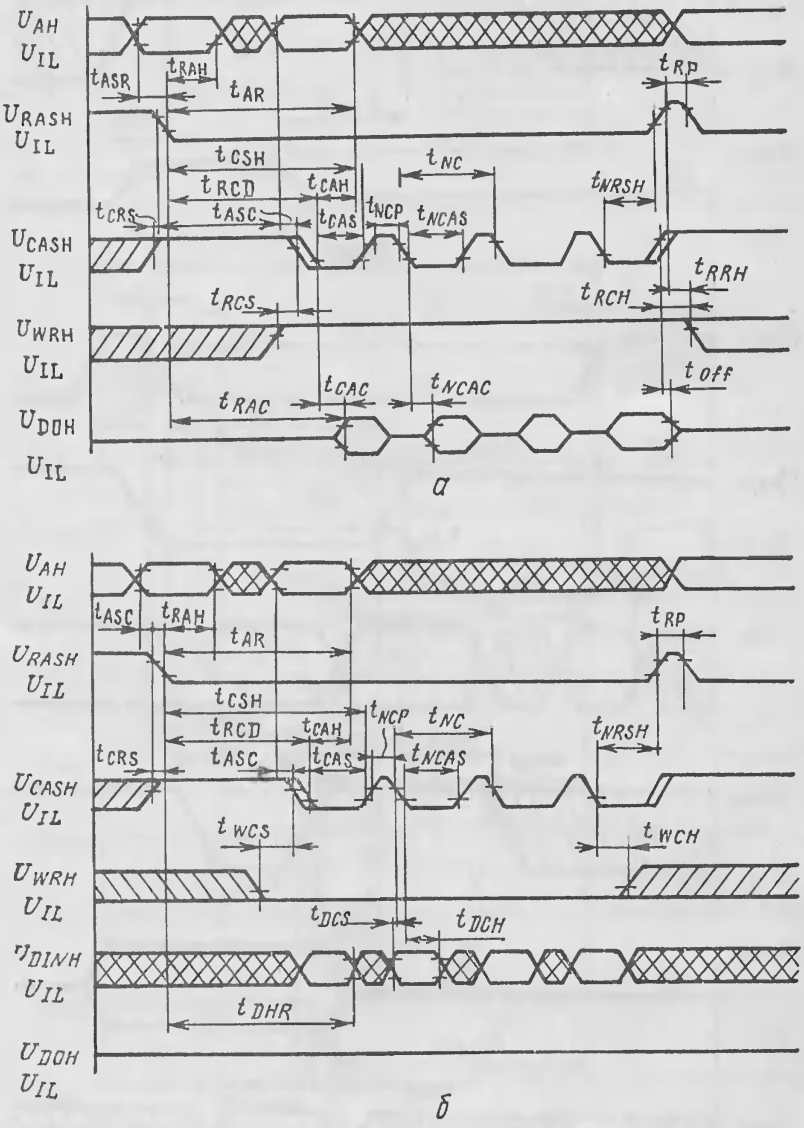


Рис. 6. Слоговый режим чтения (*а*) и записи (*б*)

Временны́е параметры сигналов на управляющих входах Таблица 3

Параметр	Обозначение	Норма (нс) для микросхемы (не менее)			Примечание
		К565РУ7В	К565РУ7Г	К565РУ7Д	
Время установления сигнала адреса строк относительно $\overline{RAS}$	t_{ASR}	0	0	0	
Длительность сигнала $\overline{RAS}$	t_{RAS}	150	200	250	Не более 5000 нс (4)
Время удержания сигнала адреса строк относительно $\overline{RAS}$	t_{RAH}	20	25	45	
Длительность интервала между сигналами $\overline{RAS}$	t_{RP}	180	200	240	
Длительность сигнала $\overline{CAS}$	t_{CAS}	75	100	125	Не более 5000 нс (4)
Время установления $\overline{RAS}$ относительно $\overline{CAS}$	t_{RCD}	50 (75)	60 (100)	70 (125)	1,4
Время удержания $\overline{RAS}$ относительно $\overline{CAS}$	t_{RSH}	75	100	125	4
Время установления $\overline{WR}$ относительно $\overline{CAS}$	t_{WCS}	0	0	0	2
Время записи по $\overline{CAS}$	t_{CWL}	65	75	105	4
Время записи по $\overline{RAS}$	t_{RWL}	65	75	105	4
Длительность фронта	t_{LH}, t_{HL}	3	3	3	Не более 35 нс
Время удержания $\overline{CAS}$ относительно $\overline{RAS}$	t_{CSH}	150	200	250	
Время удержания DI относительно $\overline{RAS}$	t_{DRH}	130	170	250	
Время удержания $\overline{WR}$ относительно $\overline{RAS}$	t_{WRH}	130	170	215	
Время удержания сигнала считывания относительно $\overline{RAS}$	t_{RRH}	50	70	90	
Время установления $\overline{CAS}$ относительно $\overline{WR}$	t_{CWD}	75	100	125	
Время установления $\overline{RAS}$ относительно $\overline{WR}$	t_{RWD}	150	200	250	
Длительность интервала между сигналами $\overline{CAS}$	t_{CPN}	50	60	90	
Время удержания сигнала адреса столбцов относительно $\overline{RAS}$	t_{AR}	130	170	215	
Длительность сигнала $\overline{CAS}$ в слоговом режиме	t_{NCAS}	50	60	80	
Длительность интервала между сигналами $\overline{CAS}$ в слоговом режиме	t_{NCP}	60	70	90	
Время удержания $\overline{RAS}$ относительно $\overline{CAS}$ в слоговом режиме	t_{NRSH}	60	70	95	
Время установления $\overline{CAS}$ относительно $\overline{RAS}$	t_{CRS}	50	70	95	
Время установления сигнала адреса столбцов относительно $\overline{CAS}$	t_{ASC}	0	0	0	4
Время удержания сигнала адреса столбцов относительно $\overline{CAS}$	t_{CAH}	40	50	60	
Время установления сигнала считывания относительно $\overline{CAS}$	t_{RCS}	0	0	0	
Время удержания сигнала считывания относительно $\overline{CAS}$	t_{RCH}	20	25	35	
Время установления сигнала DI	t_{DCS}, t_{DWS}	0	0	0	
Время удержания сигнала DI	t_{DCH}, t_{DWH}	55	70	90	
Длительность сигнала $\overline{WR}$	t_{WP}	40	65	90	
Время удержания $\overline{WR}$ относительно $\overline{CAS}$	t_{WCH}	55	70	90	
Время удержания $\overline{CAS}$ относительно $\overline{RAS}$ в режиме «$\overline{CAS}$ раньше $\overline{RAS}$»	t_{CRH}	50	70	95	
Время цикла чтения-записи	t_{CV}	340	410	500	
Время цикла чтения-модификации-записи	t_{RMW}	405	490	610	
Время цикла в слоговом режиме	t_{NC}	120	140	180	

Примечания: 1. Если $t_{RCD} > t_{RCD\,max}$, то время выборки t_{RAC} будет возрастать на величину $t_{RCD} - t_{RCD\,max}$.
2. При $t_{WCS} \geqslant t_{WCS\,min}$ информационный выход остается в состоянии высокого импеданса в течение всего цикла записи.
3. Временные интервалы и фронты управляющих сигналов измеряются относительно уровней входных сигналов: $U_{IL} = 0{,}6$ В; $U_{AH} = U_{DIH} = U_{WRH} = U_{RASH} = U_{CASH} = 2{,}4$ В.
4. Параметры обеспечиваются при $t_{LH\,(HL)} \leqslant 5$ нс.

Таблица истинности Таблица 4

Входы				Выход	Режим работы
$\overline{RAS}$	$\overline{CAS}$	$\overline{WR}$	DI	DO	
1	1	X	X	Z	Схема не выбрана
1	0	X	X	Z	—»—
0	1	X	X	Z	Регенерация
0	0	0	0 или 1	Z	Запись
0	0	1	X	0 или 1	Чтение

Примечание. X — любое состояние; Z — высокий импеданс.

Таблица 5

Основные электрические параметры

Параметр, единица измерения	Обозначение	Норма для микросхемы (не более)		
		К565РУ7В,	К565РУ7Г	К565РУ7Д
Время выборки относительно сигнала выбора адреса столбцов, нс	t_{CAC}	75	100	125
Время выборки относительно сигнала $\overline{CAS}$ в слоговом режиме, нс	t_{NCAC}	50	60	70
Период генерации, мс	T_{REF}	8	8	4
Ток утечки на входах A0...A8, $\overline{WR}$, DI, $\overline{RAS}$, $\overline{CAS}$, мкА	I_{LI}	10	10	10
Ток утечки на информационном выходе, мкА	I_{LDO}	10	10	10
Ток потребления динамический, мА	I_{CCH}	65	65	65
Напряжение высокого уровня, В, при $I_{DOL} \leqslant 4$ мА, $I_{DOH} \leqslant \lvert -2{,}0 \rvert$ мА	U_{DOH}	2,4	2,4	2,4
Напряжение низкого уровня, В, при $I_{DOL} \leqslant 4$ мА, $I_{DOH} \leqslant \lvert -2{,}0 \rvert$ мА	U_{DOL}	0,4	0,4	0,4
Время сохранения сигнала выходной информации после $\overline{CAS}$, нс	t_{off}	60	70	80
Емкость вывода сигнала $\overline{WR}$, пФ	C_{WR}	10	10	10
Емкость информационного входа и выходная емкость, пФ	C_I, C_O	10	10	10
Емкость вывода $\overline{RAS}$, пФ	C_{RAS}	12	12	12
Емкость вывода адресных сигналов A0...A8, пФ	C_A	12	12	12
Емкость вывода $\overline{CAS}$, пФ	C_{CAS}	15	15	15
Время выборки относительно $\overline{RAS}$, нс	t_{RAC}	150	200	250

Примечания. 1. t_{CAC} измеряется при емкостной нагрузке на выходе 50 пФ (с учетом паразитных емкостей). 2. Регенерация осуществляется по сигналу RAS за 512 циклов перебором адресов A0...A8. 3. $t_{RAC}=t_{RCD}+t_{CAC}$. Если $t_{RCD\,min}<t_{RCD}<t_{RCD\,max}$, то t_{RAC} будет оставаться min, а t_{CAC} — увеличиваться на величину $t_{RCD}-t_{RCD\,min}$. Если $t_{RCD}>t_{RCD\,max}$, то t_{RAC} возрастет на величину $t_{RCD}-t_{RCD\,max}$, а t_{CAC} останется минимальным.

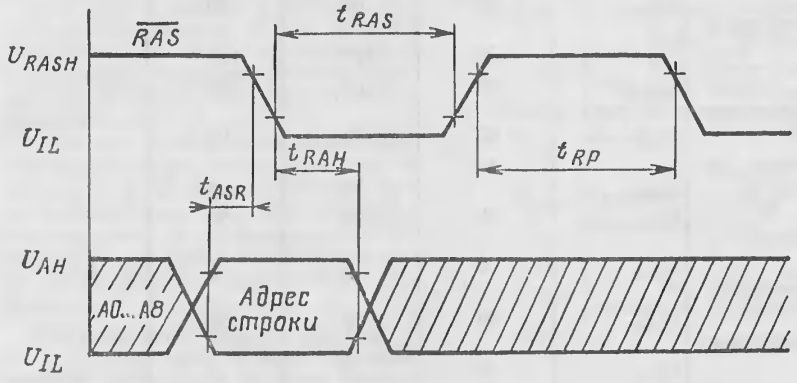


Рис. 7. Режим регенерации сигналом $\overline{RAS}$

режиме работает внутренний счетчик, отсчитывающий регенерируемые строки и срабатывающий от сигнала $\overline{RAS}$. Содержимое счетчика увеличивается на единицу в каждом такте регенерации. На всех адресных входах и выходах $\overline{WR}$, DI — произвольные уровни. Регенерация в процессе работы микросхемы несколько снижает эффективное быстродействие.

разно выполнять регенерацию сигналом $\overline{RAS}$ при высоком уровне сигнала $\overline{CAS}$, когда $\overline{CAS}$ приходит раньше $\overline{RAS}$.

При этом за время, равное периоду регенерации, тактируются сигналы $\overline{RAS}$ и перебираются все строчные адреса (рис. 7). Регенерация в режиме «$\overline{CAS}$ раньше $\overline{RAS}$» выполняется при низком уровне сигнала $\overline{CAS}$ (рис. 8). В этом

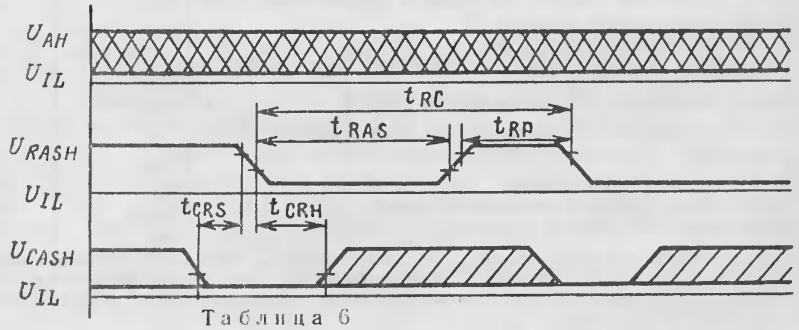


Рис. 8. Режим «$\overline{CAS}$ раньше $\overline{RAS}$»

Таблица 6

Предельно допустимые и предельные значения электрических параметров

Параметр	Обозначение	Предельно допустимый режим	Предельный режим
Напряжение питания, В	U_{CC}	4,75 (5,25)	6,0
Входное напряжение высокого уровня, В	U_{IH}	2,4 (5,5)	6,5
Входное напряжение низкого уровня, В	U_{IL}	—0,6 (0,6)	—1
Емкость нагрузки, пФ	C_L	50	100

Примечание. Допускается превышение в течение 10 мс U_{CC} и U_{IH} величины 7 В, U_{IL} — минус 2 В.

Основные электрические параметры микросхем приведены в табл. 5, предельно допустимые и предельные значения этих параметров — в табл. 6. Номинальное значение напряжения питания микросхем 5 В, допустимое отклонение от номинала ±5 %. Наработка на отказ составляет 15000 ч, интенсивность отказов не более $1 \cdot 10^{-6}$ 1/ч.

Г. Глушкова

УДК 681.323

Мутанов В. И., Гришина Ю. П., Черняв ский Ю. Г., Чернышев В. И. **Процессор цифровой обработки сигналов с векторной системой команд** // Микропроцессорные средства и системы.— 1989.— № 4.— С. 2.

Рассмотрен специализированный процессор цифровой обработки сигналов, имеющий архитектуру, ориентированную на применение системы команд векторного типа. Процессор реализован на базовом матричном кристалле, обеспечивает высокую производительность при малой потребляемой мощности.

УДК 681.327.28—529

Сидоренко В. П., Яровой С. И., Хоружий А. А. **БИС РПЗУ с УФ-стиранием информации серии К573** // Микропроцессорные средства и системы.— 1989.— № 4.— С. 5.

Приведены технические характеристики микросхем РПЗУ К573РФ5, К573РФ6, К573РФ7. Даны временные диаграммы режимов работы БИС.

УДК 681.322.1

Кравчук В. Г., Некрасов А. А., Флорентьев В. В. **Опыт работы с профессиональными персональными ЭВМ ЕС1840** // Микропроцессорные средства и системы.— 1989.— № 4.— С. 7.

Сравниваются технические аспекты реализации функционально близких ПК — IBM PC и ЕС1840. Проводится оценка надежности, ремонтопригодности, разбор имевших место отказов оборудования ЕС1840 и ЕС1940.05, описываются некоторые аппаратные добавки. Рассматриваются вопросы постановки и развития общесистемного и прикладного ПО для ЕС1840.

УДК 681.327.5.072.2

Пинчук Н. И., Тищенко В. Д., Шалугин С. С., Школяренко А. К. **Программно-аппаратный комплекс для подключения печатающих устройств типа ЕС7040 к ПЭВМ ЕС1840, ЕС1841** // Микропроцессорные средства и системы.— 1989.— № 4.— С. 22.

Описан комплекс, обеспечивающий в составе ПЭВМ ЕС1840, ЕС1841 вывод информации на печатающее устройство типа ЕС7040. Состоит из модуля интерфейса ввода-вывода, программы вывода файлов на печать, теста устройства печатающего ЕС7040.

УДК 519.725

Лупенко А. Н. **Программа преобразования кода Грея в двоичный код** // Микропроцессорные средства и системы.— 1989.— № 4.— С. 26.

Рассмотрена программа преобразования 8-разрядного кода Грея в двоичный код для микроЭВМ на базе МП КР580ИК80А.

УДК 681.3.06

Саркисов И. П. **Система автоматизированной разработки алгоритмов и программ для микроЭВМ «Искра 226» (Система АРАП)** // Микропроцессорные средства и системы.— 1989.— № 4.— С. 27.

Описана система АРАП, обеспечивающая инструментальную поддержку на этапах разработки алгоритма, программной реализации алгоритма и оптимизации программы. Рассмотрен язык описания логики алгоритмов, положенный в основу системы АРАП.

УДК 681.324

Белицкий Р. И. **Ядро операционной системы мультипроцессора с магистральной структурой** // Микропроцессорные средства и системы.— 1989.— № 4.— С 30.

Рассматриваются структура и функции ядра операционной системы экспериментального образца мультимикропроцессорной системы с магистральной структурой, построенного на основе микропроцессоров К1810ВМ86.

UDC 681.323

Mutanov V. I., Grishina Yu. P., Chernyavsky Yu G., Cernyshev V. I. **Digital signal processor with vektor instruction set** // Microprocessor devices and systems.— 1989.— N 4.— P. 2.

A specialized processor for digital signal processing having architcture designed for vector instruction set is described. The processor is a customized ULA chip and provide high data flow rate combined with l ow power consumption.

UDC 681.327.28—529

Sidorenko V. P., Yarovoy S. I., Choruzhij A. A. **UV-erasible EPROM LSIs of K573 family** // Microprocessor devices and systems.— 1989.— N 4.— P. 5.

The technical data on EPROM chips types K573PF5, K573RF6, K573RF7 are listed. The timing diagrams for these LSIs are shown.

UDC 681.322.1

Kravchuk V. G., Nekrasov A. A., Florentiev V. V. **The experience of EC1840 professional computers usage** // Microprocessor devices and systems.— 1989.— N 4.— P. 7.

The technical aspects of functionally equivalent computers usage of IBM PC and EC1840 families are discussed. The reliability estimation, common troubles and serviceability problems of EC1840 are listed. Some useful additional units are shown. The system and applied software installation for EC1840 are described.

UDC 681.327.5.072.2

Pinchuk N. I., Tishenko V. D., Shalugin S. S., Shkolyaenko A. K. **Hardware / software combination helps to connect EC7040 printer to EC1840 computer** // Microprocessor devices and systems.— 1989.— N 4.— P. 22.

The hardware and software suport used for connection of EC7040 printer to EC1840 and EC1841 computer are described. The printer adapter is supported by file printing utility and printer test program.

UDC 519.725

Lupenko A. N. **Program converts GRAY code into binary** // Microprocessor devices and systems.— 1989.— N 4.— P. 26.

The program which converts 8-bit Gray code into binary ranning on 8080 microprocessor is described.

UDC 681.3.06

Sarcisov I. P. **The ARAP system for automated algorithm and program development on ISKRA 226 computer** // Microprocessor devices and systems.— 1989.— N 4.— P. 27.

The ARAP system which gives instrumental support on the stages of algorithmic design, program implementation and optimization is featured. The language for algorithm logic description used in ARAP system is explained.

UDC 681.324

Belitsky R. I. **The Kernel of the multiprocessor operating system for bus-based processor structure** // Microprocessor devices and systems.— 1989.— N 4.— P. 30.

The structure and OS services of experimental operating system Kernel for a bus-based modular multiprocessor system wich 8086 processors is described.

УДК 681.3.06

Барковский Д. В., Жарков А. П., Креславский И. Г. **Программный пакет МЕТАКС — инструмент для генерации ассемблеров** // Микропроцессорные средства и системы.— 1989.— № 4.— С. 34.

Рассматривается машинно-независимый пакет программ, позволяющий генерировать новые и использовать имеющиеся языки ассемблера и вести трансляцию на любом из них. Приведены примеры работы с языками, у которых нерегулярные системы команд. Описывается механизм работы пакета и его техническая реализация.

УДК 681.3.06

Креславский И. Г., Барковский Д. В., Жарков А. П., Синицын Н. В. **Микропрограммный ассемблер МАСС** // Микропроцессорные средства и системы.— 1989.— № 4.— С. 37.

Представлен микропрограммный ассемблер для автоматизации проектирования ПО высокопроизводительных вычислительных и управляющих устройств, выполненных на основе секционированных микропроцессоров. Сформулированы требования к современным микроассемблерам, рассмотрена общая схема работы, функциональные возможности и технические характеристики микроассемблера МАСС. Приведен пример разработки ПО специализированного процессора.

УДК 621.327—529

Погосян М. О., Агинян Г. С., Арутюнян В. Г. **Варианты построения генераторов прямоугольных импульсов с программируемой длительностью** // Микропроцессорные средства и системы.— 1989.— № 4.— С. 49.

Характеризуются генератор прямоугольных импульсов с программируемыми периодом $T_{прог}$=2...65536 мкс и длительностью τ=1,0...65535 мкс, а также генератор с T=2...131 мкс и τ=100 нс...1,5 мкс.

УДК 681.3.049

Гуськов В. Д., Еремеев А. Л., Чучкалов П. Б. **Центральный процессор микроЭВМ СМ1810** // Микропроцессорные средства и системы.— 1989.— № 4.— С. 65.

Описано вычислительное ядро всех комплексов СМ1810 — 16-разрядный модуль центрального процессора, выполненный на основе микропроцессора КМ1810ВМ86.

УДК 681.3

Бобков Г. М. **Видеоконтроллер СМ1810.7006** // Микропроцессорные средства и системы.— 1989.— № 4.— С. 67.

Рассмотрены технические характеристики, принципы работы и программирования цветного видеоконтроллера для отображения символьной и графической информации.

UDC 681.3.06

Barcovsky D. V., Zharkov A. P., Kreslawsky I. G. **The METAX software toolbox: the instrument which generates assemblers** // Microprocessor devices and systems.— 1989.— N 4.— P. 34.

Machine-independants software paskage designed for assembly-language compilers generation is described. Newly generated assemblers as well as already existing ones may be used for user program compilation. The examples of implementation of assembly languges wich non-regular instruction sets are given. Thr details of software algorithms and technical implementation are discussed.

UDC 681.3.06

Kreslawsky I. G., Barkovsky D. V., Sharkov A. P., Sinitsin N. V. **The microprogram assembler MASS** // Microprocessor devices and systems.— 1989.— N 4.— P. 37.

The microprogram assembler suitable for target software design for high-performance computers and controllers built of bit-slice microprocessor chips is described. Main features of modern microassemblers are formulated. The general algorithm and functional features of MASS microassembler are explained. The sample program development process for specilized microprocessor is shown for illustration.

UDC 621.327—529

Pogosyan M. O., Aginyan G. S., Arutunyan V. G. **Some variants of programmable-duration pulse generators** // Microprocessor devices and systems.— 1989.— N 4.— P. 49.

Two rectangular pulse generators producing pulses with programmable period are described. The first outputs pulses with period of 2...65536 mks and duration of 1...65535 mks while the second gives the period of 2...131 mks and the duration of 0.1—1.5 mks.

UDC 681.3.049

Guskov V. D., Eremeev A. L., Chuchkalov P. B. **The central processor module for CM1810 microcomputer** // Microprocessor devices and systems.— 1989.— N 4.— P. 65.

The main data processing unit of CM1810 — 16-bit CPU module built around 8086 microprocessor is described.

UDC 681.3

Bobkov G. M. **CM1810.7006 videocontroller** // Microprocessor devices and systems.— 1989.— N 4.— P. 67.

The technical characteristics, operation principles and programming of colour videocontroller which displays alphanumeric and graphic information are explained.

Заместитель главного редактора С. М. Пеленов

Номер подготовили:
Е. И. Бабич, Г. Г. Глушкова,
В. М. Ларионова
Корректор Е. М. Кучерявенко
Техн. редактор Г. И. Колосова

Адрес редакции журнала:
103051, Москва, Малый
Сухаревский пер., д. 9-а
Тел.: 208-73-23; 208-19-94

Сдано в набор 21.04.89. Т — 09997
Подписано к печати 06.06.89.
Формат 84×108 1/16
Офсетная печать.
Усл. печ. л. 10,08. Уч.-изд. л. 14,6

Тираж 108.530
Заказ № 5642
Цена 1 р. 10 к.

Орган Государственного комитета СССР по вычислительной технике и информатике

Набрано в ордена Трудового Красного Знамени Чеховском полиграфическом комбинате Государственного комитета СССР по печати. 142300, г. Чехов Московской обл.

Отпечатано в Московской типографии № 13 ПО «Периодика» Государственного комитета СССР по печати.
107005, г. Москва, Денисовский пер., 30.
Заказ 76.

вую микроЭВМ, совместимую с IBM PC XT/AT, пакет программ, содержащий экспертную систему управляемого биостимулирования, систему учета базы данных и информации, программы координации и контроля специализированных микрокалькуляторов. Последние предназначены для сбора и обработки данных, получаемых от модуля биоэлектрических сензоров и для управления и контроля электролазерного и ультразвукового биостимулирований.

Эта система предоставляет врачу абсолютно безвредный метод лечения, индивидуально для каждого больного выбирает вид биостимулирования, сокращает среднее время госпитализации, расширяет возможности исследования механизмов биоэлектрического регулирования и т. д.

В разделе республики Куба высокой оценки заслуживают разработанные с применением вычислительной техники медицинское диагностическое оборудование, ряд алфавитно-цифровых и графических видеотерминалов, широкий набор клавиатур для ПЭВМ. В секторе медицинское приложение — система анализа клинических данных пациента, система токсикологии, психологическая система и другие. Достойны внимания обучающие системы, системы международного финансового учета, спортивной статистики и т. д., позволяющие повысить эффективность и производительность общественного труда. Оригинальная разработка — программное обеспечение для преобразования статистической информации в графическую форму.

Различные пакеты прикладных программ разработаны на базе ПЭВМ ЕС1841.

Системы речевого диалога (СРД) линии «Речь» (модели «Речь-1», «Речь-1001», «Речь-111», «Речь-121») предназначены для речевого ввода и вывода информации в/из ЭВМ. Они позволяют вести диалог человека и ЭВМ в наиболее привычной форме — голосом. Как диалоговые средства, СРД линии «Речь» используются в различных человеко-машинных системах сбора, обработки информации и управления: системах автоматизированного проетирования, связи, АСУ, АСУТП; информационно-справочных системах, гибких автоматизированных производствах, робототехнике, диспетчерских службах и т. п.

Применение СРД повышает производительность труда и эффективность использования человеко-машинных систем.

Актуальность разработок обусловлена острой необходимостью в принципиально новых средствах взаимодействия человека и ЭВМ.

Системы речевого диалога линии «Речь» состоят из четырех основных частей: встроенной микроЭВМ, устройства анализа речевых сигналов, процессора распознавания слов и слитной речи, синтезатора речи и устройства визуальной информации.

Системы распознают отдельно произносимые слова и слитные фразы, составляемые из слов выбранного словаря объемов до 600 слов на любом индоевропейском языке, синтезируют (озвучивают) речь неограниченного словаря по цифробуквенной информации на русском и украинском языках. Модель «Речь-121» осуществляет многоязычный синтез речи. Модель «Речь-1001» дополнительно реализует смысловую интерпретацию квазисянтной (с паузами между словами) речи, а модель «Речь-121» — слитной речи.

Надежность распознавания и смысловой интерпретации — 95...99 %; словесная разборчивость синтезированной речи — 98 %. Для работы системы необходимо предварительное обучение (настройка) на словарь, голос диктора и предметную область. Скорость обучения не более 35 слов в минуту. Время выдачи ответа распознавания и интерпретации не зависит от продолжительности слов или фраз и составляет 0,3 с после окончания произнесения.

В системах линии «Речь» применены методы распознавания речи, основанные на сравнении сигналов с эталонами с помощью динамического программирования. Модели СРД «Речь» имеют оригинальную архитектуру аппаратных и программных средств. Используются как автономно, так и в комплексе с персональными компьютерами; отличаются низкой стоимостью и простотой пользования.

Приоритет в разработке и применении этих методов принадлежит Институту кибернетики им. В. М. Глушкова АН УССР.

Система речевого диалога «Речь-121»

В заключение следует отметить, что данная выставка будет способствовать дальнейшему развитию средств вычислительной техники стран социалистического содружества.

1р. 10к. Индекс 70588

ПРОГРАММНО-ТЕХНОЛОГИЧЕСКИЙ КОМПЛЕКС ПОДДЕРЖКИ ИНФОРМАЦИОННОЙ ТЕХНОЛОГИИ НА БАЗЕ ЭВМ ЕДИНОЙ СИСТЕМЫ

Комплекс предназначен для тех, кто желает в сжатые сроки (1—2 квартала):

— революционным образом усовершенствовать технологию обработки данных, не останавливая промышленного функционирования системы;

— создать и внедрить общедоступные базы данных коллективного пользования, функционирующие в реальном масштабе времени;

— разработать и ввести в эксплуатацию специализированные высокопроизводительные прикладные системы учета и распределения ресурсов в реальном масштабе времени;

— осуществить эволюционный переход от устаревших к современным системам управления базами данных (СУБД) и операционным системам технических средств.

Комплекс обеспечивает пользователю:

— ввод данных по коммутируемым и выделенным телеграфным и телефонным линиям связи;

— подготовку данных на базе терминалов ЕС-7929, телетайпов и персональных ЭВМ, а также перфо- и магнитных лент;

— контроль данных (логический и по базам данных) с помощью критериев достоверности, определяемых пользователями — непрограммистами в диалоговом режиме;

— загрузку и корректировку баз данных под управлением различных СУБД (ДИСОД, Ока-ВС, СЕТОР и т. п.) как в диалоговом, так и в пакетном режимах;

— разработку прикладных задач на языках четвертого поколения типа ДИОД, позволяющих использовать данные, находящиеся под управлением СУБД различного типа, а также различных методов доступа операционной системы (последовательный, прямой, индексно-последовательный, виртуальный);

— обмен данными между базами данных центральных ЭВМ типа ЕС и персональных ЭВМ типа IBM PC/XT и PC/AT.

Комплекс превосходит аналогичные системы по таким показателям, как:

степень настраиваемости на конкретные условия применения,

мобильность по отношению к операционным системам и СУБД,

производительность при выполнении основных технологических операций по обработке данных,

а также обеспечивает полностью интерактивные режимы управления обработки данных и настройкой на конкретную область применения.

Организация-изготовитель комплекса обеспечивает заказчику поставку программных средств и эксплуатационной документации, а также ряд дополнительных научно-технических услуг:

— установку комплекса;

— настройку комплекса на конкретные условия применения;

— установку компонентов системного программного обеспечения функционирования комплекса (виртуальная и операционная системы, система телеобработки данных Кама-ВС, виртуальные методы доступа, эмуляционная программа для процессора телеобработки ЕС-8371);

— консультации по применению установленных средств.

За справками обращаться по адресу: 125083, Москва, ул. Юннатов, 18, ВНИИТ, отдел 61, телефон: 212-85-39.