М.П. Мымрин

Конструкция, применение, программирование и ремонт ПЭВМ «АГАТ»

Москва
·Машиностроение·
1990

ББК 32.973.2я75
М94
УДК 681.3-181.4.001(075.9)

Рецензент инж. А.Г. Бобров

Мымрин М.П.

М94 Конструкция, применение, программирование и ремонт ПЭВМ "Агат". - М.: Машиностроение, 1990. - 304 с.: ил.

ISBN 5-217-00386-3

Рассмотрены устройство и функционирование узлов и блоков персональной ЭВМ (ПЭВМ) "Агат", являющейся одной из базовых ЭВМ для средних общеобразовательных школ. Изложены вопросы применения машины и программирования на ней. Наряду с рекомендациями по обслуживанию и ремонту машины приведено описание блока ее контроля и набор тестовых программ. Даны практические советы по их использованию при выявлении неисправностей ПЭВМ "Агат". Приведено большое число примеров и упражнений, позволяющих получить необходимые навыки по самостоятельному программированию и обслуживанию машины.

Для специалистов, занимающихся вопросами эксплуатации, обслуживания и ремонта персональных ЭВМ, для мастеров и рабочих в этой области в качестве пособия при подготовке и повышении квалификации, а также для учащихся средних школ, изучающих предмет "Информатика и вычислительная техника".

$М\frac{2404000000 - 619}{038(01) - 90}$ 44-89

ББК 32.973.2я75

ISBN 5-217-00386-3

Предисловие

Кажется, совсем недавно появился первый персональный компьютер, а сегодня он входит в нашу жизнь так же прочно, как цветной телевизор и видеомагнитофон.

Ориентированный на индивидуального пользователя, имеющий малые габаритные размеры, персональный компьютер способен проводить самые сложные расчеты, вести учет, прогнозировать, заниматься обучением, играть, предоставлять другие самые разнообразные функциональные возможности. При этом диалог с ним ведется привычными человеку средствами. Персональный компьютер распознает почерк и голос своего владельца, синтезирует человеческую речь, формирует изображение на экране цветного телевизора. Вот почему он завоевывает все большую популярность даже у тех, кто раньше не имел дело с ЭВМ.

При разработке компьютеров наша промышленность ориентировалась на модели, выпускаемые фирмами ИБМ (IBM, США), ДЕС (DEC, США), Эпл (Apple, США).

Началу производству отечественных персональных компьютеров положил выпуск персональной ЭВМ (ПЭВМ) "Агат". Прототипом "Агата" послужил выпускаемый с 1977 г. фирмой "ЭПЛ компьютер" (Apple Computer, США) персональный компьютер "Apple II".

В настоящее время "Apple II" относится к числу наиболее популярных и распространенных во всем мире ПЭВМ из-за простоты обслуживания и эксплуатации, больших функциональных возможностей, математического обеспечения, включающего свыше 20 тыс. пакетов прикладных программ, ориентированных на самые разнообразные сферы применения.

Все выпускаемые в нашей стране персональные машины делятся на пять классов:

ПМ1 - компьютер индивидуального пользователя;

ПМ2 - компьютер ученика;

ПМ3 - компьютер учителя;

ПМ4 и ПМ5 - компьютеры для автоматизации профессиональной деятельности.

К первому классу относятся бытовые компьютеры БК-011, БК-0100 и "Микроша".

Второй и третий класс представляют ПЭВМ "Агат", "Корвет", "Ириша", имеющие емкость памяти до 256К байт, один или два накопителя на гибком магнитном диске, развитую систему цветной графики. Эти машины предназначены в основном для поддержания обучающих систем в школах, ПТУ, техникумах, вузах.

Класс компьютеров для автоматизации профессиональной деятельности представляют дорогие и сложные модели ДВК, "Электроника", "Искра 1030", "Нейрон", ЕС-1842, ЕС-1850. Быстродействие этих машин достигает 4 млн. операций в секунду, емкость памяти 16М байт; в качестве внешней памяти используются дисководы типа "винчестер" емкостью до 20М байт.

Появление персональных компьютеров значительно расширило сложившееся в отечественной практике представление о возможностях автоматизации. Как показывает опыт работы многих организаций, компьютер не только решает уже известные задачи, но, что гораздо важнее, позволяет выявлять, а затем и разрешать ранее не обнаруживавшие себя проблемы (подобно тому, как врач находит скрытые симптомы давней болезни). Важным преимуществом персональных ком-

пьютеров является то, что создаваемые на их базе системы позволяют учесть особенности каждого рабочего места.

Учебное пособие рассчитано в первую очередь на специалистов, использующих или обслуживающих ПЭВМ "Агат", но может быть полезно и тем, кто интересуется современными персональными компьютерами и желает их изучить.

Пособие наряду с многочисленными примерами, иллюстрирующими работу и применение "Агат", снабжено справочными материалами, принципиальными схемами узлов и блоков машины.

Г л а в а 1. ОСНОВЫ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

1.1. ОБЩИЕ СВЕДЕНИЯ

С увеличением выпуска ПЭВМ возрастает ее популярность. Наиболее широкое ее внедрение происходит в таких областях непромышленной сферы, как образование и административное управление.

ПЭВМ можно считать квалифицированным преподавателем, который в состоянии учесть индивидуальные особенности ученика или студента, дать ту информацию, которая будет им освоена полностью. Так, например, компьютерная система обучения (КСО), описываемая в гл. 11, вовлекает учащихся в активный процесс общения с интересующим их предметом в любой области знаний и позволяет каждому самостоятельно организовать этот процесс.

В настоящее время умение обращаться с ЭВМ рассматривается как один из элементарных навыков, которым обязаны овладеть специалисты самых разных профессий, причем приобретать такие навыки, несомненно, следует уже в начальных классах средних школ.

Использование персональных компьютеров дает возможность поднять производительность труда служащих и эффективность работы учреждений. В частности, новые программные средства "деловой графики" позволяют быстро подготовить слайды и печатные материалы для совещания.

Организованные в локальную сеть, компьютеры помогают индивидуальному пользователю анализировать большие информационные массивы, определять направления развития и преодолевать возникающие трудности. Отпадает надобность в введении огромного количества различных деловых бумаг. Все необходимые данные хранятся в памяти машины, их всегда можно вывести в виде приемлемых форм на экран дисплея или на печатающее устройство.

При работе с компьютером применяется алгоритмический язык БЕЙСИК, созданный в 60-х гг. специалистами Дармутского колледжа. Первоначально БЕЙСИК планировали использовать в качестве языка для обучения студентов программированию, теперь это самый распространенный язык общения с персональными компьютерами. В настоящее время имеются десятки других языков высокого уровня, используемых для ведения диалога с компьютерами.

В "Агате", кроме БЕЙСИКА, можно пользоваться языком ЛОГО и завоевавшим особую популярность в последние годы языком ПАСКАЛЬ. Не менее популярна разработанная специально для "Агата" операционная система "Школьница", имеющая в своем составе два языка (РОБИК и РАПИРА).

ПЭВМ "Агат" выпускается в трех модификациях (исполнение 7, исполнение 8, исполнение 9), различающихся наличием внешних устройств и объемом оперативной памяти. Исполнение 7 - базовое исполнение; исполнение 8 имеет по сравнению с базовой моделью печатающее устройство СРА-80; исполнение 9 имеет память объемом 256К байт, СРА-80 и совместима с ПЭВМ "Apple II".

1.2. КОНСТРУКЦИЯ ПЭВМ

ПЭВМ - это сложная вычислительная система, состоящая из функционально законченных устройств. Ее ядро, определяющее базовое исполнение, составляют (рис. 1.1): системный блок, видеоконтрольное устройство (ВКУ) и блок клавиатуры (БК), предназначенные для выполнения основных операций: ввода, обработки и выдачи информации.

Для повышения эффективности использования ПЭВМ используется широкий набор дополнительных средств, обеспечивающих хранение большого количества информации, печатание отчетов, ведомостей и т.п., взаимодействие с другими ЭВМ и множество других функций. К этим устройствам относятся, например, аналого-цифровые пульты, кассетный магнитофон, черно-белый монитор, мозаично-печатающее устройство.

СИСТЕМНЫЙ БЛОК

Основой всей конструкции машины является системный блок. Он объединяет в единую систему все остальные части ПЭВМ, управляя передачей данных от одного устройства к другому, обрабатывает поступающую информацию, хранит ее или выдает в удобном виде. Эти функции выполняют электронные схемы, размещенные на шести платах (*модулях*) внутри системного блока.

Для долговременного хранения данных предназначен встроенный в блок накопитель на гибком магнитном диске (НГМД). НГМД представляет собой устройство, способное записывать и считывать информацию на гибком магнитном диске (ГМД) или, как его обычно называют, *дискете*. Дискета - это гибкий пластик в форме диска диаметром 133 мм ("пятидюймовый диск"), покрытый магнитной пленкой и помещенный в твердый пластмассовый пакет для защиты.

Сменные ГМД позволяют оперативно менять программы или данные при работе с компьютером, что значительно расширяет возможности машины.

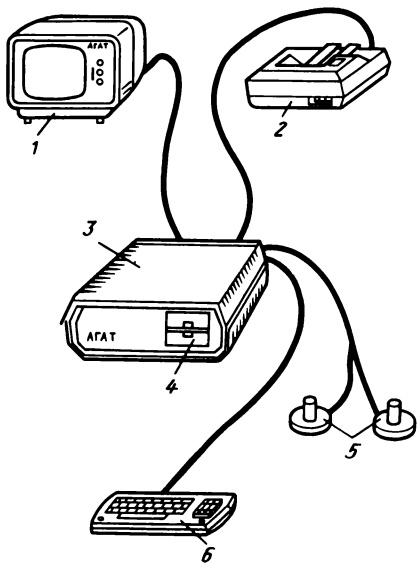


Рис. 1.1. Объединение составных частей ПЭВМ "Агат":
1 - видеоконтрольное устройство; 2 - печатающее устройство; 3 - системный блок; 4 - накопитель на гибком магнитном диске (НГМД); 5 - аналого-цифровые пульты; 6 - блок клавиатуры

Блок питания преобразует сетевое напряжение 220 В, 50 Гц в выходное постоянное стабилизированное напряжение, обеспечивающее работу всех узлов ПЭВМ. Блок выполнен в виде функционально законченного устройства и конструктивно размещен внутри системного блока.

Внешние устройства компьютера подключаются к системному блоку с помощью расположенных на задней стенке группы малоконтактных стандартных разъемов, предназначенных для подключения:

разъем ВИДЕОСИГНАЛ - черно-белого монитора;

разъем RGB - цветного видеоконтрольного устройства;

разъем  - кассетного магнитофона;

разъем ПУЛЬТ - аналого-цифровых пультов;

разъем КЛАВИАТУРА- блока клавиатуры.

КЛАВИАТУРА И ВИДЕОКОНТРОЛЬНОЕ УСТРОЙСТВО

Клавиатура представляет собой устройство, выполненное в виде отдельного блока и предназначенное для ввода команд и данных. Используя 77 клавиш, можно ввести любой из 256 символов, распознаваемых в ПЭВМ, и организовать диалог между пользователем и машиной. Отличительной особенностью ПЭВМ является то, что выполнение большей части программы происходит в диалоговом режиме работы при непосредственном активном участии пользователя. В этом случае блок клавиатуры используется для оперативного ввода указаний или новой информации в машину в ответ на ее запросы.

Вся совокупность клавиш на клавиатуре "Агат" (рис. 1.2) подразделяется на три группы: алфавитно-цифровые клавиши; функциональные клавиши; клавиши управления.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Адрес ячейки
00	00	00	00	00	FA	00	00	02	00	00	00	00	00	00	00	Содержимое ячейки
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	
. . . . .																
FFF0	FFF1	FFF2	FFF3	FFF4	FFF5	FFF6	FFF7	FFF8	FFF9	FFFA	FFFB	FFFC	FFFD	FFFE	FFFF	

Рис. 1.2. Структурная схема памяти компьютера

Работа блока клавиатуры тесно связана с работой ВКУ, который выполнен на базе унифицированного переносного телевизора цветного изображения.

Отображение информации на экране ВКУ может осуществляться в текстовом режиме, когда на экране высвечиваются алфавитно-цифровые символы, или в графическом, когда изображение формируется из отдельных точек (блоков) в виде рисунков, картинок и графиков.

Всего на экране помещается 32 строки, по 32 или 64 символа в строке. Размеры знака в 32-символьной строке вдвое шире по сравнению с 64-символьной строкой (очевидно, что большие по размеру знаки воспринимаются легче).

Разрешающая способность графики определяется числом точек, размещаемых на экране ВКУ. Чем меньше выводится точек, тем меньше разрешающая способность, но тем больше размер самих точек (в этом случае их называют блоками).

В ПЭВМ предусмотрены три режима формирования графического изображения:

графика высокого разрешения - на экран выводится 256 строк по 256 точек в строке;

г***рафика среднего разрешения*** - 128 строк по 128 блоков в строке;

графика низкого разрешения - 64 строки по 64 блока (один блок в 16 раз больше обычной точки; можно сказать, что в графике среднего разрешения один блок состоит из 4 точек, а в графике низкого разрешения - из 16 точек).

В режимах графики среднего и низкого разрешения, а также при выводе 32-символьных строк может использоваться цветное изображение. Для этого компьютер формирует восемь цветов: черный, белый, красный, желтый, зеленый, синий, голубой, фиолетовый.

МИКРОПРОЦЕССОР И ПАМЯТЬ

Внутри системного блока на одной из плат распложен кристалл большой интегральной схемы (БИС) в пластмассовом корпусе - микропроцессор 6502 - мозг всей машины. Именно микропроцессор координирует выполняемые операции и производит арифметические действия, необходимые для работы системы в целом. Микропроцессор 6502 обладает большим быстродействием (около 700 тыс. операций в секунду).

Микропроцессор непрерывно работает, взаимодействуя с другим важным устройством компьютера - ***памятью***. Емкость памяти измеряется в байтах (один байт памяти - одна машинная ячейка - способен хранить только один символ, поэтому можно рассматривать байты как информационные символы).

В ПЭВМ память кратна 1024 байт (1К байт). Всего компьютер может иметь память емкостью 60 - 748К байт (или 61 440 - 766 352 байт).

Часть памяти реализована на ***постоянных запоминающих устройствах (ПЗУ)***. В ПЗУ емкостью 2К байт размещается информация, необходимая для работы ПЭВМ. Именно она определяет, что должен делать микропроцессор, чтобы организовать диалог с пользователем, осуществить обмен данными между устройствами, входящими в состав ПЭВМ. Без этой информации машина работать не будет. Вот почему запись в ПЗУ осуществляется в нестираемой форме в процессе его изготовления и не изменяется, даже если источник питания выключен.

Остальная память машины ***динамическая***, позволяющая многократно записывать

и считывать данные в одни и те же ячейки. При отключении питания, в отличие от ПЗУ, содержимое динамической памяти уничтожается. В базовом варианте ПЭВМ "Агат" содержится 64К байт динамической памяти.

ПЕЧАТАЮЩЕЕ УСТРОЙСТВО

Печатающее устройство (принтер) позволяет печатать на бумаге любую необходимую информацию: документы, сводки, отчеты, тексты программы, таблицы результатов расчета и т.п.

Существует много моделей печатающих устройств, которые можно подключить к компьютеру: DZM–180, ROBOTRON–1156, DARO–1156, D–100, ROBOTRON K–6311, УВВПч–30 и т.д.

ПЭВМ "Агат" девятого исполнения комплектуется принтером СРА–80. На это печатающее устройство можно выводить текстовую и графическую информацию.

АНАЛОГО-ЦИФРОВЫЕ ПУЛЬТЫ И ВСТРОЕННЫЙ ДИНАМИК

Внутри системного блока находится небольшой громкоговоритель для подачи звукового сигнала. Его можно использовать в специальных программах для воспроизведения музыки, звуковых эффектов, человеческой речи.

При желании к ПЭВМ можно подключить аналого-цифровые пульты (АЦП), представляющие собой два потенциометра. Изменяя положение ручек потенциометра, можно управлять объектами на экране ВКУ в программах, предусматривающих использование пультов (например, в играх, графическом редакторе и т.п.).

1.3. ПРЕДСТАВЛЕНИЕ ДАННЫХ В КОМПЬЮТЕРЕ. ДВОИЧНЫЕ ЧИСЛА

СИСТЕМЫ СЧИСЛЕНИЯ

Исторически сложилось, что, производя вычисления, мы пользуемся десятичной системой счисления. Для представления чисел в этой системе используют десять символов - цифры арабского алфавита: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Принцип действия ПЭВМ основан на аналогии цифровых символов электрическим сигналам. Устройства ПЭВМ реагируют на наличие электрического сигнала или на его отсутствие, т.е. имеют только два устойчивых состояния: включено - выключено. Поэтому электрические сигналы поставлены в соответствие только двум цифрам:

0 - отсутствие напряжения;

1 - некоторое фиксированное значение электрического сигнала (например, 5 В).

Таким образом, ПЭВМ осуществляет вычислительный процесс, оперируя только двумя цифрами 0 и 1, т.е. выполняет все действия в ***двоичной системе счисления***.

Числовая величина при записи в двоичной системе занимает больше позиций, чем в десятичной, так как для представления числа в двоичной системе используются всего два символа. Даже сравнительно небольшие числа занимают в

двоичной системе много позиций. Например, двоичное число 11101 соответствует десятичному числу 29.

Для удобства идентификации записи двоичных и десятичных чисел в виде нижнего индекса записывают 2 или 10 соответственно. Так, например, 11101_2 - двоичное число, 11101_{10} - десятичное, причем и выражают они разные величины.

Число 256_{10} можно записать:

$$256_{10} = 2\cdot10^2 + 5\cdot10^1 + 6\cdot10^0.$$

В первой справа позиции размещены единицы, в соседней с ней второй позиции - десятки, в третьей - сотни, в четвертой - тысячи и т.д. Каждую позицию цифры в числе оценивают ***весом***, показателем степени числа 10 (основания). Вес позиции единиц равен 10^0, позиции десятков - 10^1, сотен - 10^2 и т.д. Дробной части чисел соответствуют отрицательные веса. Например:

$$725{,}03 = 7\cdot10^2 + 2\cdot10^1 + 5\cdot10^0 + 0\cdot10^{-1} + 3\cdot10^{-2}.$$

В двоичной системе счисления, как и в десятичной, каждой позиции присвоен определенный вес. Но в отличие от десятичной основание 10 заменяется на основание 2. Так, число 11101_2 запишется:

$$11101_2 = 1\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 = 16 + 8 + 4 + 0 + 1 = 29.$$

Полезно запомнить веса первых 13 разрядов двоичного числа:

4096	2048	1024	512	256	128	64	32	16	8	4	2	1
2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0.

Двоичные цифры 0 и 1 часто называют ***битами***. Так, например, для записи числа 11101_2 используется пять бит.

При вводе десятичных чисел в ПЭВМ необходимо преобразовать их в двоичные эквиваленты, а результат вычисления - двоичное число - преобразовать в десятичное. Следовательно, пользователю нужно владеть приемами преобразования десятичных чисел в двоичные.

Процедура преобразования целых десятичных чисел в двоичные - это частный случай процедуры перевода чисел из одной системы счисления в другую. Разберем один из способов перевода на примере перевода числа 10 в двоичное.

1. При делении любого числа на 2 остаток может быть равен 1 или 0. При делении 10 на 2 частное равно 5, а остаток 0. Значение остатка присваивается младшему значащему разряду искомого числа, т.е. 1-й разряд равен нулю:

$$\begin{array}{r|l} 10 & 2 \\ \hline 10 & 5 \\ \hline 0 & \end{array}$$

1-й разряд = 0

2. Результат деления необходимо еще раз разделить на 2. Остаток (0 или 1) используется в качестве значения следующего по значимости разряда. В примере частное от деления 5 на 2 равно 2, а остаток (значение 2-го разряда) равно 1:

```
 5 | 2
-4 |---
 1 | 2
```

2-й разряд = 1.

3. Результат деления на предыдущем шаге снова поделим на 2, а полученный остаток присвоим очередному разряду:

```
 2 | 2
-2 |---
 0 | 1
```

3-й разряд = 0.

4. Шаги описанной процедуры повторяются до тех пор, пока частное, полученное в результате очередной операции деления, не станет меньше делителя; тогда частное от деления присваивается старшему разряду. В примере на 3-м шаге результат деления стал равным 1. Эта 1 присваивается 4-му разряду:

4 разряд = 1.

Итак, в результате перевода числа 10 из десятичной системы в двоичную получили:

$10_{10} = 1010_2$.

Обратное действие – нахождение десятичного числа по его двоичному коду – выполняется довольно просто. Выпишем двоичные цифры преобразуемого числа, например 1010_2, в строку и над каждой цифрой, начиная с самой первой, выпишем веса разрядов двоичного числа:

8 4 2 1
1 0 1 0.

Теперь сложим те степени 2, которые написаны над 1 в числе 1010_2:

8 + 2 = 10.

Операции двоичной арифметики

Познакомимся, как осуществляется операция с л о ж е н и я двоичных чисел [2, 7, 9].

Сложим, например, два двоичных числа 0001 и 0101.

```
 0101     В крайнем справа столбце (младший разряд значений)
+
 0001     1 + 1 = 0 и перенос 1.
-----
    0     (сумма)
    1     (перенос)
 0101     Во втором справа столбце 0 + 0 + 1 (перенос) = 1.
+
 0001     Переноса в третий разряд нет.
-----
   10
 0101     В третьем столбце справа 1 + 0 = 1 (переноса нет).
+
 0001     Поэтому в сумме получается 1.
-----
  110
 0101     В крайнем левом столбце 0 + 0 = 0 (переноса нет).
+
 0001     Поэтому в сумме получается 0.
-----
 0110
```

Можно проверить этот результат по таблице соответствия (табл. 1.1) двоичных и десятичных чисел: $0101_2 = 5_{10}$, $0001_2 = 1_{10}$, 5+1=6.

Соответствующее представление десятичного числа 6 в двоичной системе 0110_2. Именно такой результат и получился. Попробуйте прибавлять двоичное число 0001 к другим двоичным числам.

Рассмотрим еще два примера на сложение. Сначала сложим числа 0110_2 и 0011_2:

$$\begin{array}{r} 0110_2 \\ +\,0011_2 \\ \hline 1001 \end{array}$$ (ответ).

Затем перейдем к более сложному примеру: сложим числа 0111_2 и 0011_2:

$$\begin{array}{r} 0111_2 \\ +\,0011_2 \\ \hline \end{array}$$

С первым правым столбцом все просто: $1_2 + 1_2 = 0$ и перенос в соседний разряд 1_2. Следующий столбец кажется более сложным, но это только пока не

Таблица 1.1

Шестнадцатеричные числа и их двоичные и десятичные эквиваленты

Шестнадцатеричные числа	Двоичные числа	Десятичные числа	Шестнадцатеричные числа	Двоичные числа	Десятичные числа
0	0	0	13	10011	19
1	1	1	14	10100	20
2	10	2	15	10101	21
3	11	3	16	10110	22
4	100	4	17	10111	23
5	101	5	18	11000	24
6	110	6	19	11001	25
7	111	7	1A	11010	26
8	1000	8	1B	11011	27
9	1001	9	1C	11100	28
A	1010	10	1D	11101	29
B	1011	11	1E	11110	30
C	1100	12	1F	11111	31
D	1101	13	20	10 0000	32
E	1110	14	32	11 0010	50
F	1111	15	40	100 0000	64
10	10000	16	GE	110 1110	110
11	10001	17	80	1000 0000	128
12	10010	18			

привыкли: $1_2 + 1_2$ плюс перенос из соседнего разряда справа 1_2. Известно, что $0001_2 + 0001_2 + 0001_2$ соответствует $1_{10} + 1_{10} + 1_{10} = 3_{10}$ в десятичной системе счисления, но это равно 11_2 в двоичной системе. Поступим следующим образом: в ответе под чертой пишем 1_2 и переносим в следующий разряд 1_2 точно так же, как сделали бы при сложении двух десятичных чисел. Окончательный ответ: 1010_2.

Теперь нетрудно выполнить аналогичным образом любые действия, требующиеся при сложении двух двоичных чисел.

Как видно из примеров, двоичное сложение выполняется по правилам десятичной арифметики. Аналогично выполняется двоичное вычитание.

Заметим, что четырехразрядные числа не нашли применения в ЭВМ. Проще работать с восьмиразрядными двоичными числами, причем это оказалось настолько удобным, что этим числам придумали специальное название - *байт*. Название восьмиразрядного двоичного числа совпадает с названием ячейки памяти. Никакого противоречия в этом нет. В одной ячейке памяти помещается только восемь разрядов двоичного числа, поэтому ее тоже часто называют байтом.

В ПЭВМ все действия производятся над двоичными числами, кратными одному байту, поэтому ограничимся рассмотрением восьмиразрядных чисел.

Диапазон представления всех однобайтовых чисел (от 0 до 255) разбит на два интервала (табл. 1.2): в одном старший разряд равен нулю, в другом единице.

Таблица 1.2

Представление положительных и отрицательных чисел

Положительные числа		Отрицательные числа	
Десятичный код	Двоичный код	Десятичный код	Двоичный код
15	00001111	-1	11111111
14	00001110	-2	11111110
13	00001101	-3	11111101
12	00001100	-4	11111100
11	00001011	-5	11111011
10	00001010	-6	11111010
9	00001001	-7	11111001
8	00001000	-8	11111000
7	00000111	-9	11110111
6	00000110	-10	11110110
5	00000101	-11	11110101
4	00000100	-12	11110100
3	00000011	-13	11110011
2	00000010	-14	11110010
1	00000001	-15	11110001
0	00000000	-128	10000000

Принято считать старший разряд знаковым.

Чтобы избежать трудностей кодирования и работы с имеющими знак двоичными числами, в ПЭВМ их представляют в дополнительном коде (форма дополнения до 2).

Перевод десятичных чисел в двоичный и дополнительный коды представлен ниже.

При переходе к дополнительному коду представление положительных чисел не меняется, а представление отрицательных чисел можно получить путем инвертирования кода положительного числа (нахождение поразрядного дополнения до 1) с последующим прибавлением единицы в младшем разряде.

Пусть, например, требуется получить двоичное представление десятичного отрицательного числа -1_{10} в дополнительном коде:

положительное десятичное число $1_{10} = 00000001_2$;

инвертированный (обратный) код 11111110_2;

после сложения с 1 получим дополнительный код, соответствующий отрицательному десятичному числу -1_{10}:

$$\begin{array}{r} 11111110_2 \\ + \quad 1_2 \\ \hline 11111111_2 \end{array}$$

Попробуйте получить аналогичное представление числа -13_{10} и проверьте результат по приведенной таблице.

Проделаем еще несколько вычитаний по форме **A - B = A + (-B)**.

Вычислим $5_{10} - 2_{10}$ или $5_{10} + (-2_{10}) = 3_{10}$:

5_{10} **есть** 00000101_2 **(уменьшаемое);**

-2_{10} **в дополнительном коде есть** 11111110_2 **(вычитаемое);**

в результате сложения получим 00000011_2 **(разность)**

и 1 переноса из старшего разряда.

Это не является настоящим переполнением, так как остальная часть числа не есть 11111111_2. По приведенной выше таблице десятичных и двоичных чисел в дополнительном коде убеждаемся в том, что числу 00000011_2 есть 3_{10}, т.е. получился правильный ответ.

Вычислим $3_{10} - 5_{10}$ или $3_{10} + (-5_{10}) = -2_{10}$:

3_{10} **есть** 00000011_2 **(уменьшаемое);**

-5_{10} **есть в дополнительном коде** 11111011_2 **(вычитаемое);**

в результате сложения получим 11111110_2 **(разность)**

без переноса из старшего разряда.

По таблице убеждаемся, что полученное число 11111110_2 соответствует

отрицательному десятичному числу -2_{10}, т.е. получился правильный результат.

Из таблицы и приведенных примеров следует, что старший седьмой (счет начинается с нуля) разряд восьмиразрядного числа служит указателем знака этого числа (положительного или отрицательного). Его установка в состояние 0 соответствует положительному числу, а в состояние 1 - отрицательному. Поэтому этот разряд называют ***знаковым***, тогда как остальные семь (с шестого по нулевой) - ***разрядами модуля числа***.

Таким образом, при выполнении операций с восьмиразрядными (однобайтовыми) числами с отведенным для знака старшим разрядом можно оперировать с любыми целыми числами в диапазоне от 01111111_2 (или $+127_{10}$) до 10000000_2 (или -128_{10}).

Рассмотрим пример у м н о ж е н и я двух двоичных чисел:

00001010 множимое
x
00001011 множитель

Сначала множимое умножаем на цифру самого младшего двоичного (правого) разряда множителя и первое промежуточное произведение помещаем под чертой. Затем умножаем множимое на цифру во втором разряде множителя справа и смещаем второе промежуточное произведение влево на один разряд (так же, как и в десятичной арифметике).

Повторяем это с каждой из цифр разрядов множителя по очереди и затем складываем полученные промежуточные произведения:

00001010
x
00001011

00001010 ⎫
0001010 ⎬ промежуточные произведения
000000 ⎪
01010 ⎭

01101110 = окончательное произведение.

Из приведенного примера следует, что умножение - это быстрый способ сложения нескольких одинаковых чисел. Создан простой способ выполнения двоичного умножения, получивший название ***умножение путем сдвига и сложения***. Перечислим основные правила этого способа, основанные на том, что при умножении двоичного числа на 0 получается 0, а результат умножения двоичного числа на 1 есть само это число.

1. ***Формирование первого промежуточного произведения***. Если значение младшего значащего разряда множителя равно 0, то и результат равен 0; если значение этого разряда равно 1, то результат является копией множимого.

2. ***Правило сдвига***. При использовании очередного разряда множителя для формирования промежуточного произведения производится сдвиг множимого на один разряд (позицию) влево.

3. ***Правило сложения***. Каждый раз, когда значение разряда множителя равно

1, к предыдущему промежуточному произведению необходимо прибавить множимое, расположенное в позиции, определенной правилом сдвига.

4. ***Определение результирующего произведения.*** Искомое произведение есть результат выполнения всех операций сдвига и сложения.

Умножение путем сдвига и сложения существенно упрощает двоичное умножение.

ШЕСТНАДЦАТЕРИЧНАЯ СИСТЕМА СЧИСЛЕНИЯ

Оперировать большими двоичными числами неудобно. Записывая, например, двоичное число 1010101110111101, легко ошибиться из-за большого числа нулей и единиц. В то же время в некоторых случаях общаться с персональной ЭВМ на уровне двоичных чисел просто необходимо, например, программируя в машинных командах.

Если перебрать все возможные комбинации из четырех бит, то их будет 2^4 = = 16. Значит, если закодировать каждую из комбинаций отдельным символом, получится удобная сокращенная форма записи двоичного числа. Заметим, что аналогично можно было закодировать группы по два, три, пять бит и т.д., однако кодирование именно по четыре бита вызвано тем, что машинная ячейка (ячейка памяти) хранит один байт, т.е. 8 бит. А восемь разрядов легко разбить на две группы, по четыре бита в каждой.

В качестве символов в таком способе кодирования двоичного числа используются десять арабских цифр и шесть букв латинского алфавита: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F (табл. 1.3).

Приведенное соответствие не что иное, как запись двоичного числа в шестнадцатеричной системе счисления.

Чтобы записать двоичное число в более компактной форме, т.е. в шестнадцатеричной системе, необходимо биты, начиная с младшего, объединить в группы по четыре и каждой группе подобрать соответствующий шестнадцатеричный символ. Например, при преобразовании двоичного числа 10011100101110_2 в шестнад-

Таблица 1.3

Кодирование чисел

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

цатеричное нужно добавить слева два незначащих нуля, чтобы можно было сформировать группы по четыре бита:

0010 0111 0010 1110

Заменив каждую группу битов соответствующим шестнадцатеричным символом (табл. 1.2), получим число

$272E_{16}$.

Шестнадцатеричные числа - это представление двоичных чисел, удобное для интерпретации работы ПЭВМ. В табл. 1.1 приведены некоторые шестнадцатеричные числа, их двоичные и десятичные эквиваленты.

Преобразование десятичного числа в шестнадцатеричное осуществляется аналогично переводу в двоичную систему. Рассмотрим это на примере преобразования числа 634:

Шаг	*Деление*	*Частное*	*Остаток*
1	634/16	39	10
2	39/16	2	7

Результат $634_{10} = 27A_{16}$

Заметим, что процедура перевода в шестнадцатеричную систему выполняется за меньшее число шагов по сравнению с переводом в двоичную систему.

1.4. ПРИНЦИП РАБОТЫ ПЭВМ

МИКРОПРОЦЕССОР И ПАМЯТЬ

Основными функциональными узлами ПЭВМ являются микропроцессор 6502 и память.

Микропроцессор - мозг машины; он обрабатывает данные и управляет работой всех остальных узлов компьютера.

В качестве источника и приемника информации используются специальные ячейки, размещенные вне микропроцессора.

Ячейка - это электронное устройство, позволяющее хранить одно слово (восемь разрядов) данных. Совокупность таких ячеек образует память машины.

Каждой ячейке (или просто байту) в памяти присваивается номер ее положения - адрес. Адрес однозначно характеризует каждый байт (рис. 1.2). Адреса памяти начинаются с нуля и изменяются до максимального числа, к которому может адресоваться микропроцессор. В ПЭВМ "Агат" максимальное число

$64K = 65535_{10} = FFFF_{16} = 1111\ 1111\ 1111\ 1111_2$.

Весь диапазон адресов называется также ***адресным пространством***. Физически в компьютере ячеек памяти больше, чем емкость адресного пространства (максимальная емкость 748К байт).

Основное назначение памяти - хранить данные и инструкции (машинные команды), согласно которым данные обрабатываются микропроцессором. Последовательность команд, необходимых для решения конкретной задачи, составляет программу.

Кроме того, часть памяти используется для создания изображения на экране ВКУ.

Как уже отмечалось, память делится на оперативную и постоянную.

Оперативная память (RAM - RANDOM ACCESS MEMORY) позволяет записывать и считывать информацию, т.е. можно изменять данные, находящиеся в ячейках оперативной памяти. При выключении ПЭВМ информация, хранящаяся в оперативной памяти, теряется.

Постоянная память (ROM - READ ONLY MEMORY), или ПЗУ, используется только для считывания данных; ее содержимое не может меняться пользователем.

Память связана с микропроцессором линиями, по которым передаются адрес (шина адреса) и данные (шина данных).

Адреса и данные попадают (выставляются) на шины с внутренних ячеек памяти микропроцессора (регистров).

Микропроцессор состоит из трех основных блоков (рис. 1.3):
арифметическо-логическое устройство; регистры; схемы управления.

Передача данных между этими устройствами осуществляется по внутренней шине данных [7, 8, 11].

Арифметико-логическое устройство (АЛУ). Главную функцию микропроцессора - обработку данных - осуществляет АЛУ. АЛУ имеет два ввода (порты), куда помещаются обрабатываемые данные, и один вывод (аккумулятор), куда попадает результат обработки.

Функционально порты выполнены в виде ячеек памяти. Каждая ячейка (буфер) может запомнить (и держать на своем выходе) сразу восемь разрядов одновременно - машинное слово, или байт.

АЛУ построено на электронной схеме с жесткой комбинационной логикой. Это

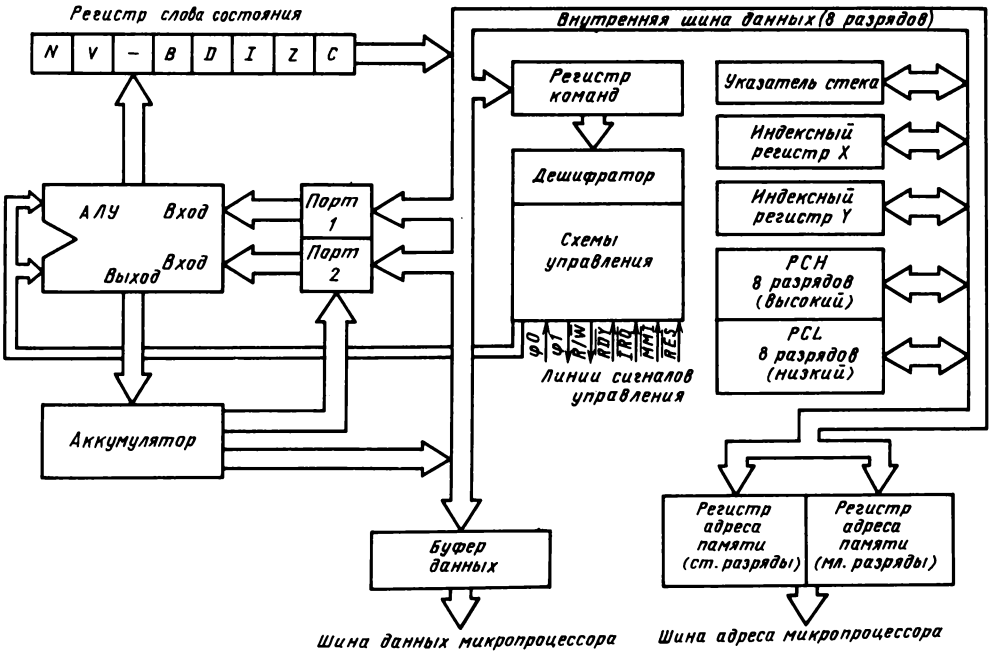


Рис. 1.3. Структурная схема микропроцессора MS6502

означает, что как только на входах есть данные (т.е. электрические сигналы, которым соответствует двоичный 0 или 1) и сигнал, соответствующий выбранной операции (например, сложить), на выходе АЛУ сразу же появляется результат выполненной операции.

Электрические сигналы, соответствующие входной и выходной информации, занимают короткий интервал времени.

Буфера позволяют удержать байт данных на входе и выходе АЛУ в течение времени, необходимого для успешной работы всего микропроцессора.

Над байтами, поступающими на вход, АЛУ выполняет следующие основные операции: ***сложение, приращение, сдвиг, логические функции***. АЛУ может выполнять 56 команд, часть из которых имеет специальное назначение.

Команды АЛУ - это двоичное число, а ***программа*** - последовательность команд (инструкций), хранимых в ячейках памяти. Для реализации программы необходимо выполнить каждую инструкцию программы в строго определенной последовательности. За порядком выполнения команд следит специальный регистр микропроцессора.

Счетчик команд (PC - PROGRAM COLINTER). Счетчик команд содержит адрес очередной команды, подлежащей выборке.

Адрес представляет собой 16-разрядные слова. Поэтому в качестве счетчика команд в микропроцессоре используются два восьмиразрядных регистра. Один из них, называемый нижним,предназначен для хранения восьми младших разрядов адреса. Он обозначается PCL (PROGRAM COUNTER LOW). Другой, называемый верхним, предназначен для хранения восьми старших разрядов адреса и обозначается PCH (PROGRAM COUNTER HIGH).

Программа может располагаться в памяти в любых ячейках, адреса которых изменяются от 0 до $FFFF_{16}$. Когда программа начинает выполняться, первым значением PC является начальный адрес программы. По этому адресу извлекается команда и помещается в другой специальный регистр микропроцессора, называемый ***регистром команд***.

После извлечения начального адреса из памяти, микропроцессор автоматически дает приращение содержимому счетчика команд и приступает к выполнению команды, прочитанной из памяти. Начиная с этого момента и на протяжении всего времени выполнения текущей команды счетчик команд "указывает", где находится следующая команда.

При желании счетчик команд можно загрузить другим содержимым, например, когда нужно многократно повторить одну и ту же часть программы в процессе выполнения всей программы. Вместо того, чтобы писать эту часть программы каждый раз, когда в ней возникает необходимость, такую запись можно сделать один раз и возвращаться к ее повторному выполнению, отступая от заданной последовательности.

Часть программы, выполняемая путем отступления от строгой последовательности команд главной программы, называется ***подпрограммой***. После того, как в счетчик команд записан начальный адрес подпрограммы, счетчик получает приращение по мере выполнения команд этой подпрограммы. Так продолжается до тех пор, пока не встретится команда возврата в главную программу.

Заметим, что при включении ПЭВМ в счетчик команд аппаратно помещается содержимое ячеек памяти с адресами FFFC и FFFD.

Регистр команд. Текущая выполняемая команда хранится в регистре команд микропроцессора. Выход этого регистра является частью дешифратора команд. Рассмотрим, как микропроцессор реализует последовательность выполнения команды, получившую название ***цикл выборки - выполнения***.

Сначала команда извлекается из памяти, затем счетчик команд загружается адресом следующей команды, подлежащей выполнению. При извлечении команда через буфер данных по внутренней шине данных пересылается в регистр команд. После этого начинается подцикл выполнения команды, в течение которого дешифратор команд декодирует содержимое регистра команд, сообщая микропроцессору, что делать для реализации операций команды.

Регистр адреса памяти. Команды микропроцессора, как правило, включают адреса ячеек, содержащие обрабатываемые данные. Эти адреса запоминаются в специальном регистре микропроцессора - регистре адреса памяти. Этот регистр при каждом обращении к памяти ПЭВМ указывает адрес области памяти, которая подлежит использованию микропроцессором. Выход этого регистра связан с шиной адреса микропроцессора.

Регистр слова состояния. Компьютер отличается от простого калькулятора наличием регистра состояния (PCR - PROCESSOR STATUS REGISTER), который хранит результаты проверок, осуществляемых автоматически в процессе выполнения команд.

При наличии в программе перехода выполнение команд начинается с некоторой новой области памяти, т.е. счетчик команд загружается новым числом.

В случае условного перехода такое действие имеет место, если результаты определенных проверок совпадают с ожидаемыми значениями. Результаты таких проверок находятся в регистре состояния.

Возможности программирования с переходами - отличительная характеристика вычислительной машины по сравнению с калькулятором. Регистр состояния предоставляет программисту возможность организовать работу микропроцессора так, чтобы при определенных условиях менялся порядок выполнения команд.

ПЭВМ принимает решение о том или ином продолжении вычислений в зависимости от указанных условий. Калькулятор таких решений принять не может.

Наличие регистра состояния позволило расширить набор команд микропроцессора командами, изменяющими последовательность выполнения программы в зависимости от значения регистра состояния.

Указатель стека. *Стек* (стековая память) - это особым образом организованный участок оперативной памяти, предназначенный для временного хранения содержимого внутренних регистров микропроцессора. Стековая память имеет еще одно название: "последним вошел, первым вышел", или LIFO (LAST - IN, FIRST - OUT). Аналогию можно провести с зарядом и извлечением патронов из магазина автомата.

Такая память необходима в том случае, когда нужно прекратить выполнение реализуемой последовательности команд (например, для немедленного выполнения специальной подпрограммы или в результате прерывания программы) и вернуться к ней позже. Стек занимает 256 ячеек с адресами от 100_{16} до $1FF_{16}$.

Указатель стека - высьмиразрядный регистр - содержит адрес свободной ячейки стека. Данные от микропроцессора поступают в ячейки начиная с адреса 1FF, по 8 разрядов одновременно, а содержимое указателя стека при этом уменьшается на единицу, указывая адрес следующей свободной ячейки стека.

Когда данные считываются из стека (аналогично тому, как выталкиваются патроны из магазина автомата), содержимое указателя стека увеличивается на единицу с каждым выбранным байтом.

Аккумулятор. Выходной регистр АЛУ служит для приема результатов выполнения операции АЛУ.

Схемы управления. Автоматическое поддержание требуемой последовательности функционирования всех узлов микропроцессора осуществляют схемы управления.

Схемы управления соединены со всеми узлами микропроцессора линиями управления. Некоторые линии связаны с внешними линиями сигналов управления ПЭВМ. Например, по сигналу готовности (RDY) микропроцессор заносит в счетчик команд число, находящееся в ячейках памяти с адресами $FFFC_{16}$ и $FFFD_{16}$, инициируя тем самым начало выполнения программы.

Прерывания. Сигналы прерывания (IRQ и NMI) предупреждают микропроцессор о том, что произошло внешнее событие, требующее с его стороны соответствующей реакции. Предположим, что из внешнего источника в ПЭВМ передаются в последовательной форме данные. Каждые восемь двоичных разрядов должны последовательно, по одному, передаваться в порт ввода, после этого микропроцессор считает их. Естественно, что при этом между каждым поступлением нового байта данных будет происходить задержка в их обработке. В течение каждого такого пропуска микропроцессор непрерывно следит за состоянием входного порта, выясняя, поступил ли новый байт. Это приводит к непроизводительной потере времени, когда микропроцессор бездействует в ожидании байта данных.

Механизм прерываний дает возможность микропроцессору использовать свободные интервалы времени для выполнения другой части программы. Когда очередной байт данных готов для обработки, прерывание информирует об этом микропроцессор, и он выполняет ряд действий по обслуживанию прерываний, что позволяет выполнить некоторую обработку данных прежде чем микропроцессор вернется к прерванной части программы.

Прерывания (IRQ) используются также при обработке данных в масштабе реального времени, когда данные для ввода в ПЭВМ вырабатываются внешним устройством так быстро, что информация о том, что они готовы, должна управлять их вводом в ПЭВМ. В случае задержки их ввода данные могут быть утеряны.

Другой тип прерывания (NMI) используется, чтобы вовремя подать сигнал о возникших ненормальных или угрожающих нормальной работе условиях, таких, как отказ питания или неисправности некоторых подсистем. Это немаскируемое (NON MASKable) прерывание, оно определяет немедленную обработку прерывания по окончании выполнения текущей команды.

МНЕМОНИКА КОМАНД

Команда микропроцессора - это такое двоичное число, которое будучи "прочитано" микропроцессором, заставляет его выполнять определенные действия. Другие двоичные числа, отличные от команд, подобных действий вызвать не мо-

гут. Большинство команд осуществляет пересылку или обработку данных, расположенных в памяти или в одном из регистров микропроцессора. Несколько команд предназначено для управления вспомогательными функциями микропроцессора.

Команды имеют разную длину (один, два или три байта) и содержат два вида информации. Во-первых, команда сообщает микропроцессору, что делать (выполнить очистку, сложение, пересылку, сдвиг и т.д.); во-вторых, команда указывает адрес (местоположение) обрабатываемых данных. Например, команды могут заставить микропроцессор добавить к содержимому аккумулятора копию содержимого некоторой ячейки памяти, очистить аккумулятор, перенести данные из индексного регистра X в индексный регистр Y и т.д.

Таким образом, команда состоит из двух частей: кода операции и адреса. Если длина команды составляет два или три байта, то первый из них - код операции, а второй (или, если есть, третий) - адрес. Однако не следует делать вывод, что все команды длиной в 1 байт - безадресные.

Команда микропроцессора - это двоичное число. Но даже однобайтовое двоичное число трудно запомнить. Еще труднее запомнить двух-, трехбайтовые команды. Шестнадцатеричные эквиваленты также трудно запомнить и тем более отождествлять с их фактическим назначением.

Поэтому при работе с компьютером применяют ***мнемоническое*** обозначение - сокращенную форму записи английского названия команд. Для этой цели используются три буквы названия операции выполняемой команды (см. прил. 1). Например, мнемоническое обозначение команды загрузки аккумулятора имеет вид **LDA** (LOAD ACCUMULATOR WITH).

Мнемоническая запись является составной частью языка ассемблера. Ассемблер преобразует мнемоническое обозначение кодов операции в соответствующие двоичные эквиваленты. Сочетание сокращенного буквенного обозначения кода операции с числовой формой записи адреса - наиболее удобная форма записи команды. Длина команды зависит от длины используемого в ней адреса. Тип обращения (адресации) к данным в команде называют ***способом адресации***. Всего микропроцессор насчитывает 16 способов адресации.

Неявная адресация. Однобайтовые команды не адресуются к данным, расположенным в памяти, а оперируют с информацией, загруженной в регистры микропроцессора, или с разрядами регистра состояния. При таком способе адрес как бы встроен в команду, поэтому он и называется неявным.

Однобайтовые команды выполняются быстрее любых других команд (в течение одного-двух машинных циклов).

Непосредственная адресация. Код операции команды с непосредственной адресацией размещается в первом байте. Сразу же за кодом операции следуют данные, занимающие один байт. Эти данные берутся не из памяти, их предоставляет микропроцессору программист при записи команды. При использовании данного способа адресации не требуется указывать адрес памяти, необходимо только записать код операции и данные. Операции, задаваемые первым байтом команды (кодом операции), микропроцессор выполняет над данными, представленными ее вторым байтом. Эти команды исполняются за два машинных цикла.

Прямая адресация. Для размещения или считывания данных из ячейки памяти применяется прямая адресация. Команда с прямой адресацией имеет длину 2 байт (прямая короткая) или 3 байт (прямая длинная). Первый байт предназначен для

кода операции, второй и, если имеется, третий - для адреса. Адрес указывает область памяти, в которой находятся подлежащие обработке данные. Совместное использование второго и третьего байтов команды позволяет адресоваться к любой из 65536 ячеек памяти. При неявной адресации адрес местоположения данных "встроен" в команду, и программист лишен возможности самостоятельно обращаться к данным по их адресу. При непосредственной адресации данные указываются в самой команде сразу за кодом операции. И в этом случае программист не может адресоваться к данным. Только прямая адресация позволяет ему это делать, явным образом задавая адрес используемых данных. Прямая короткая адресация позволяет обратиться только к первым 256 байтам (с 0 по 255) - это адрес так называемой нулевой страницы памяти; прямая длинная охватывает весь диапазон адресов (с 0 по 65535).

Прямая адресация делает программу независимой от обрабатываемых ею данных, т.е. позволяет одну и ту же программу применять с различными данными. Время выполнения прямой длинной команды в 2 раза больше, чем время выполнения команд с непосредственой адресацией, а прямой короткой - в 1,5 раза. Рекомендуется использовать этот тип адресации, только если возникает необходимость размещать данные независимо от программы.

Косвенная адресация. При этом способе во втором байте (косвенная короткая), а если есть, и в третьем (косвенная длинная) указывается адрес ячейки, содержащей адрес (исполнительный) местоположения данных в памяти. Подобные команды отличаются от команд непосредственной адресации, содержащих сами данные, или от команд прямой адресации, которые включают адреса этих данных. Косвенная адресация удобна при обращении к часто используемым областям памяти, особенно в тех случаях, когда данные организованы в виде некоторого списка. Кроме того, этот способ адресации позволяет сделать программу еще более универсальной по отношению к обрабатываемым ею данным, размещая их, например, каждый раз в новой области памяти. Команды с косвенной адресацией выполняются за три и более машинных цикла.

Индексная адресация. Эта адресация удобна при обращении к массивам и таблицам. Для образования исполнительного адреса к адресной части команды прибавляется смещение (шестнадцатеричное число) из индексного регистра. Индексные регистры (X и Y) программно доступны, и их содержимое можно изменить, что позволяет варьировать исполнительные адреса без модификации команды, а значит, и всей программы в целом. Когда индексная адресация используется для доступа к массиву, адрес в команде соответствует базовому адресу массива, а значение индексного регистра - индексу элемента массива.

Относительная адресация. При использовании относительной адресации исполнительный адрес формируется сложением базового адреса с адресным полем команды. В качестве базового адреса служит содержимое регистра счетчика команд (PC). Относительная адресация позволяет строить программы, свободно перемещаемые в памяти за счет того, что в команде всегда указано смещение по отношению к содержимому регистра счетчика команд.

Смещение интерпретируется как знаковое целое число, представленное в дополнительном коде, что обеспечивает переход в любом направлении. Максимальный переход соответствует сдвигу на число адресов в диапазоне от +127 до -128 по отношению к содержимому PC. При этом до прибавления смещения со-

держимое регистра счетчика команд увеличивается для формирования исполнительного адреса следующей команды.

При перемещении программы из одной области памяти в другую смещение не изменяется, так как относительные позиции команд и данных сохраняются. Поскольку содержимое счетчика команд постоянно увеличивается, формируемые исполнительные адреса изменяются для одного и того же смещения, т.е. это динамический способ адресации. Команды с относительной адресацией занимают два байта, но, поскольку для выполнения операции сложения смещения и содержимого счетчика команд используется АЛУ микропроцессора, эти команды выполняются дольше, чем двухбайтовые команды с прямой или косвенной адресацией.

Г л а в а 2. ЭКСПЛУАТАЦИЯ ПЭВМ

2.1. ВКЛЮЧЕНИЕ МАШИНЫ

Базовый вариант ПЭВМ, состоящий из трех узлов (системный блок, ВКУ, клавиатура), объединяется в единую систему двумя семижильными кабелями, входящими в комплект ПЭВМ. При этом ВКУ и блок клавиатуры подключаются к системному блоку через разъемы RGB и КЛАВИАТУРА соответственно.

Блок питания, находящийся внутри системного блока, осуществляет питание всех узлов ПЭВМ, за исключением ВКУ, имеющего свой блок питания. Поэтому ВКУ и системный блок включаются раздельно.

После включения питания микропроцессор начинает выполнять последовательность стартового цикла - "холодный" старт. "Холодный" старт осуществляется микропроцессором в несколько этапов под управлением специальной программы, называемой "Системный монитор". Эта программа хранится в ПЗУ емкостью 2048 байт (2К) и занимает адреса F800 - FFFF в адресном пространстве микропроцессора. Первый этап "холодного" старта начинается с выборки вектора восстановления, хранящегося в ячейках FFFC и FFFD системного монитора. ***Вектор восстановления*** - это начальный адрес программы, осуществляющей дальнейшие начальные установки системы.

Прежде всего по содержимому ячейки индикации питания, имеющей адрес 03F4, программа проверяет, было ли включено питание только что или была нажата клавиша СБРОС. В первом случае микропроцессор продолжает выполнять "холодный" старт, во втором - переходит к выполнению "теплого" старта.

При продолжении "холодного" старта очищается экран ВКУ, и в верхней его части появляется надпись ** АГАТ **. После этого происходит установка ячейки индикации питания (занесение числа А5), сообщающей, что электропитание включено. Завершается "холодный" старт автоматическим включением дисковода и загрузкой программы, находящейся на предварительно вставленным в него ГМД. О начале работы НГМД сигнализирует свечение индикатора на его передней панели. Если дискета в накопитель не вставлена, то прервать работу дисковода можно нажатием клавиши СБРОС.

"Теплый" старт выполняется каждый раз, когда нажимается клавиша СБРОС. При этом повторяется вся последовательность цикла "холодного" старта до

момента проверки ячейки индикации питания. В эту ячейку в цикле "холодного" старта было занесено число A5, и "теплый" старт заканчивается загрузкой в счетчик команды микропроцессора вектора повторного входа (содержимое ячеек 03F2 и 03F3). Таким образом, ПЭВМ начинает выводить данные на экран через несколько секунд после включения машины и ВКУ.

Характер появляющейся на экране информации зависит от формы диалога с ПЭВМ. Но одно условие является обязательным - это наличие символа, приглашающего к диалогу, и курсора. *Курсор* - это мигающая черточка на экране, указывающая позицию расположения на экране очередного символа. Курсор может принимать и иную форму, в том числе белого квадрата, и даже может быть невидимым для глаза. В последнем случае местоположение курсора остается звестным машине, хотя и не обнаруживается визуально.

Вид символа приглашения к диалогу зависит от программы, с которой ведется диалог. При диалоге с системным монитором на экране появляется "*", в случае диалога с интерпретатором языка БЕЙСИК - знак "]".

2.2. КЛАВИАТУРА

Прежде чем приступить к диалогу с компьютером, необходимо изучить клавиатуру.

Клавиатура ПЭВМ сходна со стандартной клавиатурой обычной пишущей машинки и насчитывает 77 клавиш, объединенных в три группы (рис. 2.1). Клавиши, относящиеся к одной группе, распоположены вместе и окрашены в один цвет. Центральное положение занимает самая многочисленная группа алфавитно-цифровых клавиш светлого цвета. С помощью этих клавиш в работе с ПЭВМ можно использовать: 32 символа букв русского алфавита; 26 символов букв латинского алфавита; 10 символов арабских цифр; 28 специальных знаков.

При нажатии каждой клавиши этой группы могут быть воспроизведены на экране два символа латинского или русского набора букв, что задается одновременным нажатием клавиш РЕГ и РУС или РЕГ и ЛАТ.

Крайнюю правую группу образуют 15 темных клавиш, называемых функциональными или программируемыми; их назначение может меняться программным способом.

Особую группу составляют управляющие клавиши. Они выделены темным цветом и расположены слева и справа от основной группы. С помощью этих клавиш реализуются стандартные функции управления.

СБР - клавиша сброса. Нажатие этой клавиши заставляет микропроцессор осуществить "теплый" старт, например, для прерывания выполнения любой программы.

Рис. 2.1. Расположение клавиш на клавиатуре

Чтобы исключить случайный сброс, клавиша СБР защищена от случайного нажатия. Сброс системы происходит только, если клавишу СБР нажать одновременно с клавишей УПР.

↵ - клавиша перевода строки. Обеспечивает ввод информации, набранной на клавиатуре (например, директив). Обычно эта клавиша нажимается последней, после чего ПЭВМ воспринимает все, что было введено до ее нажатия, и отвечает тем или иным образом. В дальнейшем описание клавиши ↵ будет упоминаться достаточно часто при разборе конкретных случаев ввода директив в ПЭВМ.

ПВТ - клавиша повторения. Нажатие этой клавиши обеспечивает автоматическое повторение набора символа, нажатого одновременно с клавишей ПВТ, с частотой 10 раз в секунду. Повторение прекратится, когда будет отпущена клавиша символа или клавиша ПВТ.

РЕГ - клавиша переключения регистра. Используется совместно с другими клавишами для набора символов, изображенных в нижней части клавиш. Например, при нажатии клавиши РЕГ и клавиш цифр на экране будут высвечиваться те символы, которые изображены на соответствующих кнопках снизу.

↑, ↓, →, ← - клавиши управления курсора. Нажатие одной из этих клавиш вызывает перемещение курсора в ту сторону, в которую направлена стрелка, изображенная на данной клавише. С помощью этих клавиш возможно перемещение курсора в пределах экрана вверх, вниз, вправо, влево.

РЕД - клавиша редактирования. Нажатие этой клавиши обеспечивает ввод в режим редактирования, в котором нажатие клавиши → или ← вызывает простое передвижение курсора без изменения буфера входной строки, а нажатие любой другой клавиши выводит из режима редактирования. Более подробно использование этой клавиши описано в гл. 5.

УПР - клавиша управления. Сама никаких действий не производит, используется только совместно с другими клавишами. Одновременное нажатие клавиши УПР с алфавитно-цифровыми клавишами вызывает действие ПЭВМ, определяемое программой "Системный монитор".

УПР Л - очистка экрана. Все, что высвечивалось на экране до нажатия клавиши, стирается; курсор перемещается в левый верхний угол экрана ВКУ.

УПР Щ - очистка до конца строки. Все, что высвечивалось справа за курсором на данной строке, стирается; курсор остается на месте.

УПР Ч - очистка до конца экрана. Все, что высвечивалось ниже и вправо от курсора, стирается; курсор остается на месте.

УПР Г - короткий звуковой сигнал.

УПР Б - выход в интерпретатор языка БЕЙСИК с первоначальными установками. Любая программа или переменные, которые имелись в ПЭВМ ранее (до выхода в программу "Системный монитор"), будут потеряны.

УПР Ц - выход в интерпретатор языка БЕЙСИК с сохранением имеющейся в памяти программы.

УПР Р - инициирование ввода с НГМД. Используется в директиве "X" УПР Р, где X - номер разъема внутреннего системного интерфейса, в который установлен модуль контроллера - НГМД.

УПР М - дублирование действия клавиши ↵.

УПР Н, УПР У, УПР Ы, УПР З - действие клавиш аналогично действию клавиш управления курсором, соответственно влево, вправо, вверх, вниз.

УПР Е - выдача на экран ВКУ содержимого внутренних регистров микропроцессора: A - аккумулятора, X, Y - индексных регистров, P - регистра состояния, S - указателя стека.

УПР Ш - действие, аналогичное действию клавиши РЕД.

УПР Ь - отмена вводимой строки.

2.3. ОРГАНИЗАЦИЯ ДИАЛОГА С ПЭВМ

Как уже отмечалось, диалог пользователя с ПЭВМ "Агат" определяется наличием той или иной системной программы. Основными компонентами системного программного обеспечения являются системный монитор, дисковая операционная система (ДОС), интерпретатор языка БЕЙСИК. Диалог с любой из этих программ организуется по следующему правилу. Наберите на клавиатуре строку, содержащую директивы той системной программы, в диалоге с которой вы работаете, и нажмите клавишу ↵ - перевод строки. Соответствующая системная программа воспринимает то, что набрано, и будет действовать согласно набранным директивам. Строкой при этом считается набор символов (не более 255).

СИСТЕМНЫЙ МОНИТОР

Программа "Системный монитор" выполняет три основные функции:

1) осуществляет загрузку с магнитного диска при включении ПЭВМ "Агат";

2) обеспечивает обмен данными с базовыми устройствами ввода-вывода (ВКУ, БК, магнитофоном, аналого-цифровыми пультами, динамиком);

3) предоставляет пользователю диалоговые возможности для осмотра, сравнения, изменения содержимого памяти и регистров процессора в мнемонике системы команд в шестнадцатеричном формате.

Монитор начинается с адреса FF69. Войти в "Системный монитор" можно прервав "холодный" старт, нажав клавишу СБР или введя соответствующие директивы интерпретатора БЕЙСИКА (если пользователь находится в состоянии диалога с интерпретатором БЕЙСИКА).

Итак, при нажатии сразу же после включения ПЭВМ клавиши СБР (совместно с клавишей УПР) прерывается "холодный" старт до начала загрузки с системного ГМД и происходит выход в "Системный монитор" (о чем свидетельствует появление звездочки и мигающего квадрата).

Монитор воспринимает информацию трех типов: адреса, данные и команды.

Адреса и данные задаются монитору в шестнадцатеричном формате, причем необходимо следить за числом разрядов. Адрес задается только четырьмя разрядами, а данные - двумя. Если разрядов меньше, то вместо недостающих разрядов помещаются нули (слева), а если разрядов больше, то лишние разряды (левые) отсекаются.

Монитор распознает 22 различных символа команд (директив). Это могут быть знаки пунктуации, заглавные буквы или управляющие знаки. Кроме того, монитор запоминает адрес последней опрошенной ячейки (ПОЯ), адрес следующей изменяемой ячейки (СИЯ) и адрес команды (АК).

Рассмотрим некотрые из этих директив.

Просмотр содержимого ячеек памяти:

а) просмотр одной ячейки

*XXXX↵,

где XXXX - шестнадцатеричный адрес просматриваемой ячейки;

б) просмотр интервала памяти

*XXX1·XXX2 ↵,

где XXX1 - адрес первой ячейки памяти, XXX2 - адрес последней ячейки памяти;

в) просмотр нескольких, не идущих подряд ячеек

*XXX1␣XXX2␣XXX3␣ ↵,

где XXX1 - адреса просматриваемых ячеек; ␣ - знак, означающий пробел.

Во всех трех случаях последний набранный адрес запоминается как ПОЯ, а следующий за ним адрес - как СИЯ. Это можно использовать при наборе директив.

Примеры директив:

•↵ - выводит содержимое ячейки СИЯ;

•.XXX2↵ - выводит интервал памяти начиная с ячейки СИЯ и кончая XXX2.

Примеры просмотра ячеек памяти:

	Директива	*Ответ монитора*
а)	*1000↵	1000 - ␣ 00
б)	*1000.100F↵	1000 - 00 FF 00 FF 00 FF 00 FF
		1008 - 00 FF 00 FF 00 FF 00 FF
в)	*1000 1004 1009 ↵	1000 - 00
		1004 - 00
		1009 - FF
г)	*1000 ↵	1000 - 00
	*↵	1001 - FF 00
	*↵	1003 - FF 00 FF 00 FF 00 FF 00

Каждое новое нажатие клавиши вызывает на экране содержимое следующих восьми ячеек памяти и т.д.

Запись данных в ячейки памяти:

а) XXXX:YY↵,

где XXXX - адрес изменяемой ячейки; YY - новое значение, заносимое в ячейку;

б) изменение содержимого последовательности ячеек памяти

XXXX:Y1 Y2...↵,

где Y1, Y2, ... - новое значение, заносимое в ячейки начиная с адреса 1000 и далее.

Максимальное число одновременно изменяемых ячеек, данные которых набираются через пробел, равно 85. Ячейкой СИЯ становится ячейка, следующая за последней измененной.

Примеры изменения ячеек памяти:

а) *Директивы*	*Ответ монитора*
*5 ↵	0005 - 00
*:FF↵ или *5 :FF↵	*
*5 ↵	0005 - FF

Символ двоеточия заставляет монитор записать данные, следуемые после двоеточия в ячейку с набранным адресом или в СИЯ (0005).

б) *Директивы*	*Ответ монитора*
*1000.100F↵	1000 - 00 FF 00 FF 00 FF 00 FF
	1008 - 00 FF 00 FF 00 FF 00 FF
*1000:01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0 0O ↵	
*1000.100F↵	1000 - 01 02 03 04 05 06 07 08
	1008 - 09 0A 0B 0C 0D 0E 0F 0O,

Перемещение интервала памяти (директива MOVE):

а) копирование

*XXX1 < XXX2.XXX3M ↵,

где XXX1 - адрес нового местоположения перемещаемого участка памяти; XXX2 - адрес первой, а XXX3 - адрес последней ячейки старого местоположения перемещаемого участка памяти.

Оба адреса XXX2 и XXX3 должны быть больше или меньше XXX1.

б) дублирование

XXX1 < XXX2.XXX3M↵,

где XXX2 < XXX1, а XXX3 > XXX1; после выполнения директивы содержимое ячеек с адресами от XXX2 до XXX1 дублируется по всему интервалу памяти с адреса XXX1 по адрес XXX3.

Примеры дублирования:

1000.1003↵ 1000 - FF FF FF FF

1004 < 1000.1007M↵

1004.1007↵ 1004 - FF FF FF FF

Сравнение двух участков памяти (VERIFY).

Эта директива используется для проверки правильности выполнения команды MOVE:

XXX1 < XXX2.XXX3V↵

Просмотр программы на экране ВКУ (LIST).

Директива LIST позволяет вывести на экран часть содержимого памяти в виде, упорядоченном согласно мнемонике языка ассемблера ПЭВМ. Одна директива выводит на экран 28 строк текста. Формат директивы

XXXXL,

где XXXX - адрес начала программы. Например:

300L↵

Запуск программы на выполнение (GO).

По директиве системный монитор выполняет программу, записанную в машинных кодах (системе команд микропроцессора 6502), начиная с указанного адреса:

XXXXG,

где XXXX - адрес начала работы программы. Например:

E003G↵

На этом разбор директив монитора закончим.

ИНТЕРПРЕТАТОР С ЯЗЫКА БЕЙСИК И ДОС

В ПЭВМ используются 5-дюймовые ГМД информационной емкостью 140К байт (рис.2.2). Дискета с одной стороны имеет небольшой прямоугольный вырез (6 х 8 мм). Если этот вырез заклеить кусочком плотной бумаги или фольги, то запись информации на ГМД будет невозможна. Обычно вырез заклеивают для исключения случайного стирания информации на ГМД.

Заметим, что если сделать на дискете второй вырез, симметричный первому, то можно использовать вторую сторону диска.

Интерпретатор с языка БЕЙСИК и ДОС размещаются на ГМД. ГМД с записанным на него интерпретатором и ДОС

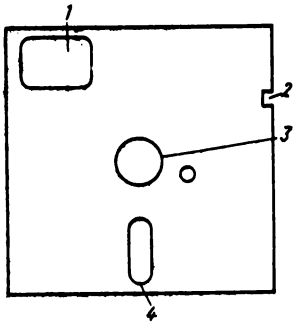


Рис. 2.2. Гибкий магнитный диск:
1 - индивидуальная этикетка; 2 - вырез защиты записи; 3 - отверстие для вала НГМД; 4 - прорезь для головки чтения - записи

в дальнейшем будем называть системным ГМД или просто системной дискетой. Обычно прорезь на системной дискете заклеена.

Загрузка интерпретатора БЕЙСИКА и ДОС осуществляется автоматически. Поэтому перед включением ПЭВМ нужно вставить системный диск в карман НГМД этикеткой вверх, прорезью для головки чтения - записи вперед, опустить планку-замок на НГМД вниз до щелчка.

После включения ПЭВМ выполняет "холодный" старт, и с системного ГМД загружаются ДОС и интерпретатор БЕЙСИКА. В этом случае на экране высвечиваются пригласительный символ "]" и мигающий курсор, после чего системную дискету следует заменить на ту, с которой вам нужно загрузить нужную программу. Системный ГМД, где хранятся ДОС и интерпретатор БЕЙСИКА, рекомендуется использовать только для первоначального ввода в компьютер ДОС и интерпретатора. Кроме того, полезно иметь несколько копий системного ГМД.

Наберите на клавиатуре следующую директиву:

]CATALOG↵

После нажатия клавиши ↵ на передней панели дисковода загорается индикатор, и на экране появляется перечень всех программ, которые хранятся на данном диске. После просмотра оглавления диска можно выполнить выбранную программу. Если программа написана на языке БЕЙСИК (помечена буквой A в оглавлении), она запускается директивой **RUN**; если в двоичных кодах (помеченная B), то директивой **BRUN**:

]RUN имя программы ↵

или

]BRUN имя программы ↵

В обоих случаях после нажатия клавиши ↵ снова загорается индикационная лампа НГМД, и программа загружается в память ПЭВМ. Следует помнить, что все программы и данные, которые находились в памяти ПЭВМ до загрузки, теряются.

Программу на языке БЕЙСИК можно набрать с клавиатуры в диалоге с интерпретатором БЕЙСИКА. Более подробно это рассматривается в гл. V и VI.

Работа с программами в командах микропроцессора разобрана в гл. IV.

Язык БЕЙСИК предоставляет профессиональным программистам и неподготовленным пользователям возможность решать самый широкий круг задач на ПЭВМ.

Для оперативного взаимодействия с НГМД, т.е. использования диска для долговременного хранения программ и данных, предназначена ДОС. Программы, а также совокупность данных, объединенных под одним именем на диске, получили название ***файла***. Другими словами, файл - это любой набор информации под своим именем, записанный на магнитый диск.

ДОС позволяет работать с файлами трех типов:

A - программа на БЕЙСИКе"

B - двоичные данные или программа в командах микропроцессора;

T - текстовая информация.

Заметим также, что в ПЭВМ "Агат" совместно с БЕЙСИКом или другим языком можно использовать только ДОС. Никакую другую операционную систему (например, CP/M, MSX, UNIX и т.д.) "Агат" не понимает. Не будут работать в компьютере и программы, подготовленные и записанные на других ПЭВМ (пусть даже и имеющих аналогичный дисковод). Исключение (с ограничениями, которые будут рассмотрены ниже) составляют программы, записанные на ПЭВМ "Apple II" и "Правец-82".

Более подробно ДОС и интерпретатор БЕЙСИКа разобраны в гл. V и VI.

Кроме ДОС и интерпретатора БЕЙСИКа, на системном диске полезно иметь сервисные программы, обеспечивающие проверку работоспособности дисковода, выявление дефектных участков на ГМД, восстановление "незагружающихся" программ или данных, копирование информации с одного диска на другой. Некоторые такие программы описаны в гл. VI и приведены в прил. 2. К сервисным программам относятся:

SKOR (скорость) - программа, регулирующая частоту вращения дискеты;

POZ (позицирование) - программа, выявляющая дефектные участки на ГМД;

COPY, FID - программы, предназначенные для копирования файлов в пределах одной дискеты или с диска на диск;

COPY2-1, DRW - программы, с помощью которых восстанавливается информация на ГМД.

Эти и подобные им программы, как правило, самодокументируемые, т.е. инструкция, как с ними работать, выводится на экран ВКУ при запуске программы командой RUN или BRUN.

Во время работы некоторые программы требуют указать номер разъема SLOTA и номер привода DRAVE. Для базового исполнения этими номерами являются соответственно 3 и 1.

Вот, например, как можно произвести копирование одного диска на другой с использованием программы COPY:

]RUN COPY ↵

После запуска программы на экране появляется сообщение: "оригинал: разъем:-".

В ответ нужно набрать 3 и ↵, после чего под словом "разъем" появляется слово "драйвер:-".

Затем вводится 1 и ↵. После ввода 1 на экране ниже появляется надпись: "дубликат :"-".

В ответ указывается 3 и ↵, а на следующее слово - 1 и ↵.

На появившийся вопрос: "Приступить к копированию?" необходимо нажать букву D. Затем по указанию программы попеременно менять оригинал и дубликат, подтверждая смену диска нажатием любой клавиши.

ВОЗМОЖНЫЕ СБОИ МАШИНЫ

В заключение отметим некоторые возможные ситуации сбоя, возникающие при включении машины.

1. При включении ПЭВМ на экране ВКУ появляется сетчатое поле с полосой

бегущих квадратов, расположенной на 2-3 см ниже верхней границы экрана. При нажатии на любую клавишу движение полос останавливается. Необходимо выключить ПЭВМ, не выключая ВКУ, и через 10-20 с включить снова. Если сбой повторится, следует проверить соединение в разъеме КЛАВИАТУРА (возможно, отошел контакт или оборван один из проводов, соединяющих клавиатуру с ПЭВМ).

2. ПЭВМ никак не реагирует на нажатую клавишу, т.е. на экране ВКУ соответствующий символ не высвечивается. При нажатии на клавишу ***ПВТ*** на экране начинается многократная печать одного и того же символа. Причиной может быть западание клавиши с символом, распечатываемым на экране. Заметим, что при западании управляющей клавиши выявить ее таким способом нельзя. В этом случае необходима визуальная проверка каждой управляющей клавиши.

3. Загрузка с системного диска интерпретатора БЕЙСИКа не вызывает появления символа "]"; при этом машина не реагирует на клавиатуру ("зависает"), а после "сброса" на экране появляется *.

Одна из возможных причин сбоя - дефект системного диска. В этом случае следует поменять системную дискету и попробовать загрузить интерпретатор БЕЙСИКа снова.

Чтобы избежать сбоев и поломок ПЭВМ, необходимо выполнять следующие простые правила:

не соединять и не разъединять узлы и блоки включенной ПЭВМ;

заземлить системный блок, ВКУ и другие устройства, подсоединяемые к системному блоку;

вставлять системный диск в НГМД до включения и вынимать после остановки дисковода;

включать первым и выключать последним системный блок;

не касаться экрана ВКУ во время работы ПЭВМ руками или другими предметами (ручкой, карандашом и т.п.);

избегать включения электроприборов при работе ЭВМ;

регулярно проводить техническое обслуживание компьютера (тестирование, протирку спиртом контактов и разъемов и т.п.).

Глава 3. АРХИТЕКТУРА И ПРИНЦИП РАБОТЫ ПЭВМ "АГАТ"

3.1. КОНФИГУРАЦИЯ ПЭВМ

ПЭВМ "Агат" построена по модульному принципу. Все основные функциональные блоки выполнены в виде конструктивно законченных устройств - модулей, которые связаны между собой единым каналом обмена информации - внутренним системным интерфейсом.

Модульное построение ПЭВМ позволяет расширить минимальный состав персонального компьютера, т.е. его функциональные возможности. Минимально необходимой конфигурацией, обеспечивающей работоспособность компьютера, является объединительная плата с установленным в нее модулем центрального процессора. Введение дополнительных модулей в любом количестве и в любых сочетаниях, необходимых пользователю [18], обеспечивает простую реконфигурацию

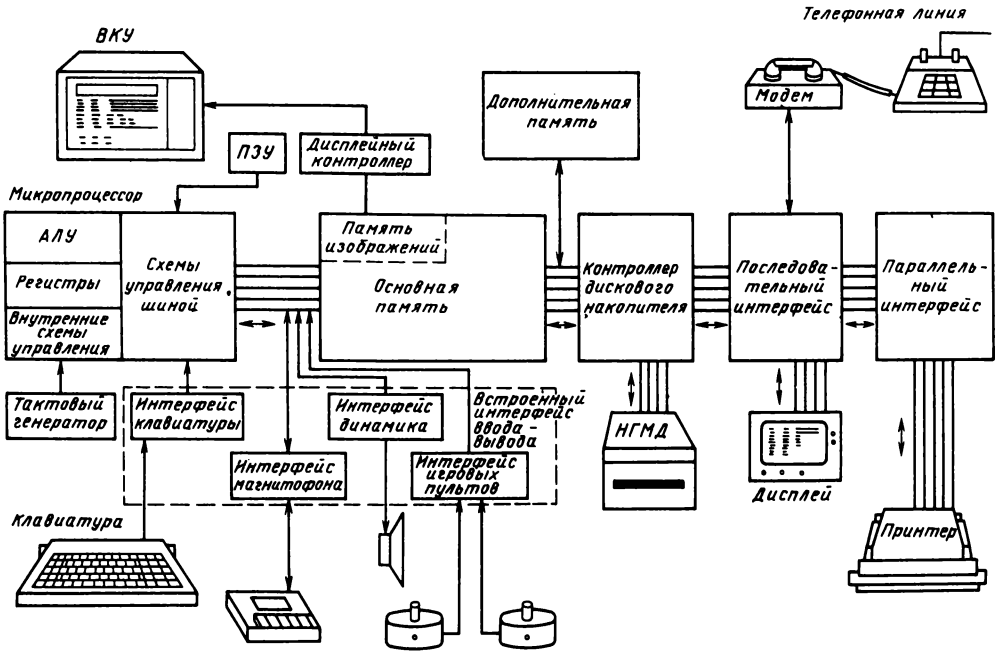


Рис. 3.1. Структурная схема компьютера

ПЭВМ. Таким образом, пользователь может сам определить наиболее оптимальную конфигурацию системы, исходя из конкретных потребностей применения персонального компьютера "Агат".

Рассмотрим работу функциональных узлов компьютера. Как уже упоминалось выше, в машине реализована структура с общей шиной - внутренним системным интерфейсом (рис. 3.1). При такой организации системы шин обмен информацией между процессором, периферийными устройствами и памятью выполняется по единому правилу. Отдельные команды ввода-вывода (как в других ПЭВМ) в системе команд компьютера отсутствуют. Это позволяет повысить эффективность ПЭВМ, так как весь набор команд обращения к памяти может использоваться для передачи и обработки содержимого регистров периферийных устройств. Кроме того, другим важным достоинством является простота структуры шин и минимизация числа связей для обмена информацией между устройствами ПЭВМ.

3.2. РАСПРЕДЕЛЕНИЕ АДРЕСНОГО ПРОСТРАНСТВА

Центральный процессор ПЭВМ с входящим в его состав микропроцессором 6502 (фирма **MOS Technology**), используя 16 адресных линий, может непосредственно адресовать 65 536 ячеек памяти (64К байт). Все ячейки памяти в компьютере поделены на следующие категории (рис. 3.2):

оперативная память;

постоянная память;

ячейки ввода-вывода;

ПЗУ ячеек ввода-вывода.

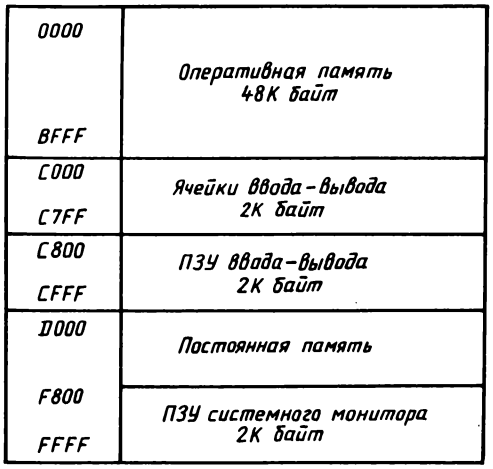


Рис. 3.2. Распределение памяти

РАСПРЕДЕЛЕНИЕ ОПЕРАТИВНОЙ ПАМЯТИ

Оперативная память емкостью 48К байт предназначена для хранения текущих данных и программ пользователя. Как уже отмечалось, оперативная память хранит данные, только пока ПЭВМ включена. При выключении компьютера память теряет свое содержание. При работе с оперативной памятью нужно помнить о том, что некоторые ее области имеют специальное назначение. Это так называемые системные ячейки, используемые системным программным обеспечением.

0000 - 00FF - нулевая страница. Занимает самые младшие 256 байт адресного пространства. Используется для организации короткого способа адресации, когда предполагается, что старший байт исполнительного адреса равен нулю, например (00) - (адресация нулевой странице).

Нулевая страница имеет большое значение для минимизации длины программы и времени ее выполнения. Отдельные ячейки нулевой страницы используются и системным монитором, и дисковой операционной системой, и интерпретатором языка БЕЙСИК.

0100 - 01FF - стек. Занимает следующие 256 байт после нулевой страницы. Стек связан с центральным процессором аппаратно, т.е. при каждой стековой операции микропроцессор автоматически предполагает наличие стека в адресах первой страницы (устанавливает старший байт исполнительного адреса в состояние 01).

0200 - 02FF - входной буфер. Занимает 256 байт второй страницы. Используется большинством программ и языков как входной буфер для ввода информации, поступающей от пользователя. Примером применения входного буфера служит ввод с клавиатуры под управлением системного монитора. При нажатии клавиши, соответствующей определенному символу, его шестнадцатеричное представление в коде КОИ-8 попадает в буфер. При нажатии клавиши $\underline{\lceil}$ накопленная в буфере строка символов передается для анализа в монитор, а буфер освобождается для принятия символов новой строки.

03F0 - 03FF - векторные ячейки монитора. Это специальные 16 ячеек, используемые при диалоге пользователя с ПЭВМ. Приведенное на рис. 3.2 распределение оперативной памяти условное (логическое). Физически все ячейки

памяти (и 0000, и BFFF) равнозначны. Однако нужно помнить, что такому распределению отвечают программные возможности компьютера.

Логически оперативная память разделена еще на один вид страниц. Основную оперативную память можно использовать в качестве экранной памяти (видеоОЗУ). Это означает, что информация, хранимая в этой области, может быть отображена на экране ВКУ в любом из пяти возможных режимов, обеспечиваемых дисплейным контроллером (ДК).

РАСПРЕДЕЛЕНИЕ ПОСТОЯННОЙ ПАМЯТИ

При организации работы с ПЭВМ предполагается, что адресное пространство D000 - FFFF отведено под ячейки ПЗУ. Особенность ПЗУ заключается в возможности долговременно хранить информацию, т.е. при включении компьютера данные в ПЗУ не стираются. Эта особенность делает ПЗУ удобным средством хранения базового системного обеспечения.

Правда, у этого способа хранения системных команд есть недостатки:

микросхемы ПЗУ, способные органически вписаться в персональный компьютер, пока еще дороги и дефицитны;

средства перепрограммирования микросхем ПЗУ недоступны широкому кругу пользователей.

Вот почему память по адресам D000 - FFFF в ПЭВМ "Агат" организована на динамических микросхемах. Отметим, что часть адресов F800 - FFFF реализована в ячейках ПЗУ, где хранятся программы системного программного обеспечения: "Системный монитор".

Распределение адресов постоянной памяти в ПЭВМ следующее:

D000 - DFFF - дисковая операционная система;
E000 - F7FF - интерпретатор языка БЕЙСИК;
F800 - FFF9 - программы "Системный монитор";
FFFA - FFFF - векторы прерывания.

Значение последних шести ячеек играет особую роль в функционировании всей системы ПЭВМ (см. п. 3.4):

FFFA - младший байт векторного указателя для $\overline{NMI}$;
FFFB - старший байт векторного указателя для $\overline{NMI}$;
FFFC - младший байт векторного указателя для $\overline{RES}$;
FFFD - старший байт векторного указателя для $\overline{RES}$;
FFFE - младший байт векторного указателя для $\overline{IRQ}$;
FFFF - старший байт векторного указателя для $\overline{IRQ}$.

ЯЧЕЙКИ ВВОДА-ВЫВОДА

Адреса ячеек C000 - CFFF отведены в компьютере для организации обмена информацией между центральным процессором и внешними устройствами. При этом обмен данными устройств ввода-вывода с контроллерами и интерфейсами осуществляется аналогично обмену центрального процессора с оперативной памятью, т.е. любые из внешних устройств (входные и выходные регистры, регистры управления контроллеров и интерфейсов) - это ячейки памяти, доступные коман-

дам записи и чтения. Как уже отмечалось, такая организация максимально упрощает обмен информацией и управление устройствами ввода-вывода.

Ячеек памяти по этим адресам физически не существует. Используются только линии шины адреса, аппаратно связанные с соответствующими внешними устройствами.

Выделенная под ячейки ввода-вывода область адресного пространства разделена на две части:

C000 - C7FF - собственно ячейки ввода-вывода (2К байт);

C800 - CFFF - адреса, отведенные под программируемые устройства ввода-вывода, имеющие собственные ПЗУ с программой, которую необходимо выполнить микропроцессором при обмене информации с данным уст ройством ввода-вывода (2К байт).

3.3. ОРГАНИЗАЦИЯ ПАМЯТИ

Емкость адресуемой оперативной памяти составляет 48К байт, а емкость ОЗУ, размещенного на объединительной плате, определяется применяемыми микросхемами динамической памяти:

32К байт - исполнение с памятью на К565 РУ6

64К байт - исполнение с памятью на К565 РУ5Д1

128К байт - исполнение с памятью на К565 РУ5(А,Б,В,Г).

В состав ПЭВМ "Агат" могут быть введены модули оперативной памяти, расширяющие оперативную память компьютера. В зависимости от применяемых микросхем динамической памяти емкость ОЗУ, установленного на модуле оперативной памяти, базовых вариантов ПЭВМ может составлять 32, 64 и 128К байт. Таким образом, общая емкость ОЗУ, используемого в ПЭВМ, может изменяться в пределах от 32 до 768К байт (при добавлении 640К байт на пяти модулях оперативной памяти). Для подключения к адресному пространству (емкостью 48К байт) полной емкости ОЗУ используется специальная организация оперативной памяти - страничная.

СТРАНИЧНАЯ ОРГАНИЗАЦИЯ ОПЕРАТИВНОЙ ПАМЯТИ

В дальнейшем ***основной оперативной памятью*** (ООП) будем называть ОЗУ, расположенное на объединительной плате, а ***дополнительной оперативной памятью*** (ДОП) - ОЗУ, расположенное на модулях оперативной памяти.

Страничная организация оперативной памяти предполагает распределение адресного пространства на три области (страницы) емкостью 16К байт каждая [7, 17].

ИСПОЛЬЗОВАНИЕ ООП

ОЗУ, расположенное на объединительной плате, разбито на массивы по 16К байт. При емкости 32К байт таких массивов только два, и они всегда подключены к адресам 0000 - 7FFF (рис. 3.3, *а*).

При емкости ОЗУ 64К байт ООП разделяется на четыре массива по 16К байт. Два из них всегда занимают в адресном пространстве адреса 0000 - 7FFF (рис. 3.3, *б*), остальные два массива являются переключаемыми. Область адресного окна 8000 - BFFF занимает один из этих массивов. Подключение того

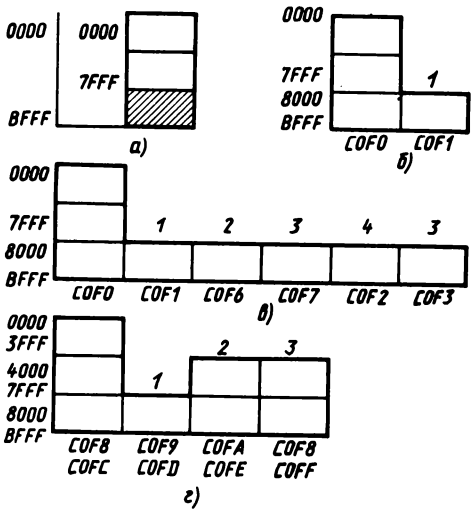


Рис. 3.3. Страничная организация оперативной памяти

или иного массива управляется с помощью специальных ячеек ввода-вывода COFO и COF1. Выполнение микропроцессором операции записи в одну из этих ячеек обеспечивает подключение к адресам 8000 - BFFF соответствующего массива.

При емкости ОЗУ 128К байт ООП разделяется на восемь массивов по 16К байт каждый. В ПЭВМ реализовано два способа использования этих массивов.

Первый способ (рис. 3.3, *в*). Два массива всегда подключены к адресам 0000 - 7FFF. Адреса B000 - BFFF занимает один из шести оставшихся массивов. Для подключения соответствующего массива необходимо выполнить микропроцессором операцию записи в одну из ячеек ввода-вывода: COFO, COF1, COF2, COF3, COF6, COF7.

Второй способ (рис. 3.3, *г*). Один массив всегда занимает область памяти с адресами 0000 - 3FFF. Остальные массивы подключаются к адресам 4000 - BFFF с помощью ячеек COF8 и COFC, COFA и COFE, COFB и COFF или COF9 и COFD. Как и в предыдущих случаях, подключение массивов через соответствующие ячейки ввода-вывода осуществляется при выполнении микропроцессором операции записи в эти ячейки. Отметим еще раз, что при использовании ячеек ввода-вывода COFO - COFF для переключения массивов ООП значащим является только адрес, а данные при этом могут быть произвольными.

ИСПОЛЬЗОВАНИЕ МОДУЛЯ ОПЕРАТИВНОЙ ПАМЯТИ ДЛЯ РАСШИРЕНИЯ ООП

Общая емкость оперативной памяти ПЭВМ может быть увеличена с помощью модулей ДОП. Модуль ДОП, используемый для увеличения оперативной памяти, получил название модуля расширения ООП.

Модуль расширения ООП может быть установлен в любом из свободных разъемов объединительной платы в случае исполнения ПЭВМ "Агат" с объемом ООП в 32К байт.

При других исполнениях ПЭВМ, т.е. при емкости ООП 64 или 128К байт, этот модуль может быть установлен только в разъемы X7, X6 и т.д.

Вся оперативная память модуля разбита на массивы по 16К байт каждый, доступ к которым осуществляется через адресное окно 8000 - BFFF. При этом адреса 0000 - 7FFF всегда подключены к ООП, т.е. к микросхемам памяти, расположенным на объединительной плате.

Управление модулем ДОП осуществляется изменением соответствующих разрядов специального регистра, расположенного на нем. Модуль ДОП имеет специальный регистр - слово состояния модуля. Управление работой модуля заключается в изменении значений соответствующих разрядов этого регистра.

Выбор ДОП в качестве модуля расширения ООП, т.е. подключение к адресному окну 8000 - BFFF, производится установкой четвертого разряда (А3) слова состояния модуля в состояние логической единицы. При этом часть ООП, адресуемая через поле адресов 8000 - BFFF, исключается из адресного пространства процессора.

Отметим, что пока модуль не выбран, доступ к ОЗУ модуля невозможен, и адреса 8000 - BFFF занимает часть ООП.

Номер подключаемого массива емкостью 16К байт определяется тремя младшими разрядами А0 - А2 слова состояния модуля.

При необходимости защиты информации, записанной в модуль оперативной памяти, от случайных программных изменений может быть введена защита записи в модуле расширения ОПП. Это достигается переключением разряда А4 в состояние логической единицы, что запрещает запись в ячейки ОЗУ и сохраняет доступность информации по чтению. Изменение данных в ячейках памяти модуля возможно только после перевода А4 в состояние логического нуля.

Рассмотрим более подробно случаи, когда в качестве модулей расширения ООП используют более одной платы ДОП. Нужно помнить, что разъемы, а значит, и установленные в них модули имеют разный приоритет: младший номер имеет более высокий приоритет.

Выбор модуля с высоким приоритетом обеспечивает подключение ОЗУ, расположенного на этом модуле, к адресам 8000 - BFFF. Все остальные модули с более низким приоритетом переводятся в режим хранения независимо от того, выбраны они или нет.

Чтобы обратиться к оперативной памяти, расположенной на модуле, имеющем низкий приоритет, необходимо предварительно перевести в режим хранения все модули, имеющие более высокий приоритет, чем требуемый модуль. Это достигается переключением разряда А3 слова состояния модулей в логический нуль.

ИСПОЛЬЗОВАНИЕ МОДУЛЯ ОПЕРАТИВНОЙ ПАМЯТИ КАК ПСЕВДОПЗУ

В этом случае базовое исполнение модуля оперативной памяти предназначается не для расширения ООП, а для эмулирования ПЗУ системы. В соответствии с принятым в ПЭВМ "Агат" системным распределением адресного пространства процессора адреса D000 - FFFF всегда занимает ПЗУ ПЭВМ, в котором хранятся программа "Системный монитор" и интерпретатор языка БЕЙСИК. Любое изменение информации в ПЗУ (например, для введения в компьютер других языков программирования) предполагает изготовление новых микросхем ПЗУ, что в процессе эксплуатации ПЭВМ может быть затруднительным. Использование базового исполнения модуля оперативной памяти позволяет решить эту задачу без изготовления новых микросхем ПЗУ.

После окончания загрузки стандартное ПЗУ компьютера блокируется с одновременным подключением на адреса D000 - FFFF центрального процессора микросхем модуля оперативной памяти, которые переводятся в режим работы, аналогичный режиму ПЗУ. Это означает, что для исключения возможности случайного изменения информации, записанной ранее в ОЗУ модуля, вводится блокировка записи. Таким образом, плата оперативной памяти, используемая в качестве псевдоПЗУ, хотя и содержит ОЗУ, но таковым не является, как по адресам,

доступным центральному процессору, так и по режиму доступа: возможна либо только запись во время загрузки с диска, либо только считывание после окончания загрузки. Модуль оперативной памяти, определенный подобным образом, называется в ПЭВМ "Агат" ***модулем псевдоПЗУ***".

Очевидно, что при использовании модуля псевдоПЗУ в составе ПЭВМ, когда существует возможность оперативного изменения системного монитора и языков программирования, любое программно некорректное обращение с модулем оперативной памяти может привести к аварийному завершению работы компьютера.

Отметим еще раз, что операции записи и чтения в ОЗУ модуля являются взаимоисключающими: когда разрешена запись в ОЗУ, то запрещено чтение из него, и наоборот.

Управляет работой модуля псевдоПЗУ пятый разряд А5 слова состояния, который в отличие от всех других разрядов оказывает влияние на работу не только модуля оперативной памяти, но и ПЭВМ в целом.

Два возможных состояния разряда А5 определяют два режима работы модуля псевдоПЗУ и соответственно компьютера "Агат":

1) А5 в состоянии логического нуля. При этом одновременно: разрешена запись в ОЗУ псевдоПЗУ по адресам D000 - FFFF;

запрещено считывание (выдача на шину данных) из ОЗУ модуля псевдоПЗУ;

разрешена нормальная работа стандартного ПЗУ, расположенного на модуле микропроцессора;

2) А5 в состоянии логической единицы. При этом одновременно:

запрещена запись в ОЗУ модуля псевдоПЗУ;

разрешено считывание данных из ОЗУ модуля по адресам D000 - FFFF;

блокировано (запрещено считывание) стандартное ПЗУ.

Как видно из сказанного выше, при логическом нуле в разряде А5 происходит как бы раздвоение адресов D000 - FFFF. Считывание информации по этим адресам происходит из стандартного ПЗУ, а запись - в ОЗУ модуля оперативной памяти. Именно это обстоятельство и позволяет достаточно просто осуществить загрузку ОЗУ модуля для дальнейшей его работы в качестве псевдоПЗУ. Заметим, что на работу модуля в качестве расширения ООП А5 влияния не оказывает.

Поскольку адресное пространство ПЗУ имеет емкость 12К байт, то очевидно, что подключить к ним полностью массив с такой емкостью нельзя. Поэтому 16К байт делятся на основной массив емкостью 8К байт и два массива по 4К байт каждый.

Основной массив всегда подклю чается к адресам E000 - FFFF. Два оставшихся массива подключаются к адресам D000-DFFF в зависимости от состояния шестого разряда А6 слова состояния модуля оперативной памяти (рис.3.4). При этом разряды А0, А1, А2 указывают номер массива из 16К байт, подключаемого к адресам D000 - FFFF. На работу модуля оперативной памяти в режиме псевдоПЗУ разряд А4 не влияет.

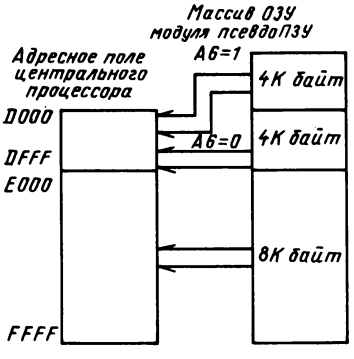


Рис. 3.4. **Схема подключения банков памяти**

СЛОВА СОСТОЯНИЯ МОДУЛЯ ОПЕРАТИВНОЙ ПАМЯТИ

Словом состояния модуля является совокупность из семи младших разрядов регистра слова состояния, расположенного на модуле оперативной памяти. Старший разряд регистра является функциональным признаком модуля:

для модуля расширения ООП равен 0;

для модуля всевдоПЗУ равен 1.

Назначение остальных разрядов приведено в табл. 3.1.

Таблица 3.1

Назначение разрядов слова состояния модуля ООП

Значение разрядов слова состояния								Функция модуля
A7	A6	A5	A4	A3	A2	A1	A0	
x	x	x	x	x	0	0	0	Доступен массив "0"
x	x	x	x	x	0	0	1	Доступен массив "1"
x	x	x	x	x	0	1	0	Доступен массив "2"
x	x	x	x	x	0	1	1	Доступен массив "3"
x	x	x	x	x	1	0	0	Доступен массив "4"
x	x	x	x	x	1	0	1	Доступен массив "5"
x	x	x	x	x	1	1	0	Доступен массив "6"
x	x	x	x	x	1	1	1	Доступен массив "7"
x	x	x	x	0	x	x	x	Модуль расширения ООП не-выбран
x	x	x	x	1	x	x	x	Модуль расширения ООП выб-ран и доступен
x	x	x	0	x	x	x	x	Запись в модуль ООП раз-решена
x	x	x	1	x	x	x	x	Запись в модуль ООП запре-щена
x	x	0	x	x	x	x	x	В модуль псевдоПЗУ разрешена запись, запрещено считывание, разблокировано стандартное ПЗУ

Продолжение табл.3.1

Значение разрядов слова состояния								Функция модуля
A7	A6	A5	A4	A3	A2	A1	A0	
x	x	1	x	x	x	x	x	В модуль псевдоПЗУ запрещена запись, разрешено считывание, блокировано стандартное ПЗУ
x	0	x	x	x	x	x	x	На модуле псевдоПЗУ к адресам D000 - DFFF подключен основной массив емкостью 4К байт
x	1	x	x	x	x	x	x	На модуле псевдоПЗУ к адресам D000 - DFFF подключен дополнительный массив емкостью 4К байт

Примечание. x - состояние разряда может быть любым.

Чтобы записать новое слово состояния в регистр, необходимо выполнить операцию записи произвольной информации по адресу ячейки ввода-вывода:

CNXX,

где N = 1 + 6 - номер разъема, в котором стоит модуль оперативной памяти, слово состояния которого изменяется; XX - код нового слова состояния.

Чтобы прочитать данное слово состояния, необходимо осуществить операцию чтения по адресу:

CNXX.

В результате выполнения этой операции значения разрядов регистра попадают на шину данных микропроцессора.

Пример 1. Для модуля расширения ООП, установленного в разъеме X5 объединительной платы, с помощью команды монитора осуществим запись произвольной информации по адресу C409:

```
*C409:00 ↵
```

Изменившееся при этом слово состояния модуля расширения ООП обеспечит подключение к адресам 8000 - BFFF массива с номером 1 модуля ООП.

Пример 2. Для модуля псевдоПЗУ, установленного в разъеме X3, осуществим запись по адресу C220:

```
*C220:0F ↵
```

Новое слово состояния обеспечивает подключение массива с номером 0 емко-

стью 16К байт к адресам D000 - FFFF и одного массива емкостью 4К байт к адресам D000 - DFFF, блокирует стандартное ПЗУ и разрешает считывание информации из ОЗУ, одновременно запрещая запись в ячейку.

Пример 3. Для модуля ОП выполним:

```
* 1000 :     LDA ¤ C200   AD 00 20
  1003 :     STA ¤ 1006   8D 06 10
* 1000G
* 1004 ⌐
       ↓
       _
```

После выполнения последней команды монитора на экране появится информация 104 - XX, где XX - значение регистра слова состояния модуля оперативной памяти, установленного в разъеме X3.

3.4. ОТОБРАЖЕНИЕ ИНФОРМАЦИИ НА ЭКРАНЕ ВКУ

Отображение информации на экране ВКУ соответствует информации в ячейках памяти. Формируется изображение в одном из режимов (табл. 3.2):

Таблица 3.2

Графические и алфавитно-цифровые режимы ПЭВМ "Агат"

Режим	Формат изображения, элемент разложения	Размер элемента разложения, точка	Число цветов элемента разложения	Емкость видео-ОЗУ, К байт
Графический высокого разрешения	256x256	1x1	2	8
Графический среднего разрешения	128x128	2x2	16	8
Графический низкого разрешения	64x64	4x4	16	2
Алфавитно-цифровой АЦР-32	32x32	7x8	16	2
Алфавитно-цифровой АЦР-64	64x32	7x8	2	2

графическом высокого разрешения (ГВР);
графическом среднего разрешения (ГСР);
графическом низкого разрешения (ГНР);
алфавитно-цифровом (АЦР).

ГРАФИЧЕСКИЙ РЕЖИМ ВЫСОКОГО РАЗРЕШЕНИЯ

Графический режим высокого разрешения обеспечивает отображение в пределах рабочего поля экрана (256 рядов по 256 точек в ряду). Каждой точке изображения на экране соответствует один бит в памяти экрана (видеоОЗУ): "1" в памяти экрана засвечивает точку (белый цвет), "0" гасит (черный цвет).

В пределах одной телевизионной строки (256 точек) на экране отображается 32 байта из памяти экрана. Разряды первого байта в строке отображаются слева направо в верхнем углу экрана. За нулевым разрядом первого байта отображается старший разряд следующего байта, и так до нулевого разряда последнего 32-го байта строки (рис. 3.5).

Таким образом, для отображения изображения, например, из 5584 точек потребуется объем видеоОЗУ в 8К байт.

ГРАФИЧЕСКИЕ РЕЖИМЫ СРЕДНЕГО И НИЗКОГО РАЗРЕШЕНИЯ

Цветные графические режимы среднего и низкого разрешения называются также ***псевдографическими***. Элементом разложения в этих режимах является не точка, а некоторая совокупность точек, засвечиваемая на экране одним цветом и получившая название "блок" (PIXEL). Каждый байт в памяти экрана разделяется

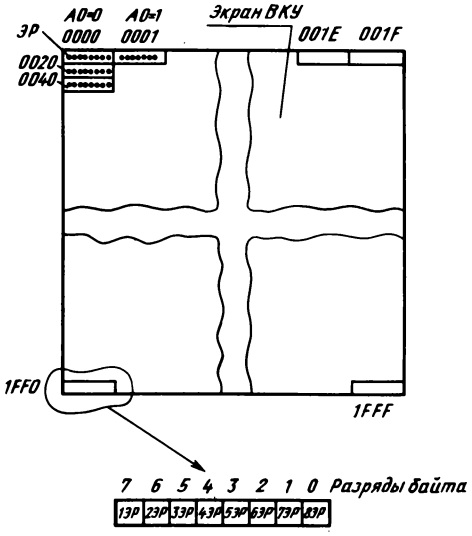


Рис. 3.5. Экран ВКУ в графическом режиме высокого разрешения:
ЭР - элемент разложения; А - разряд байта

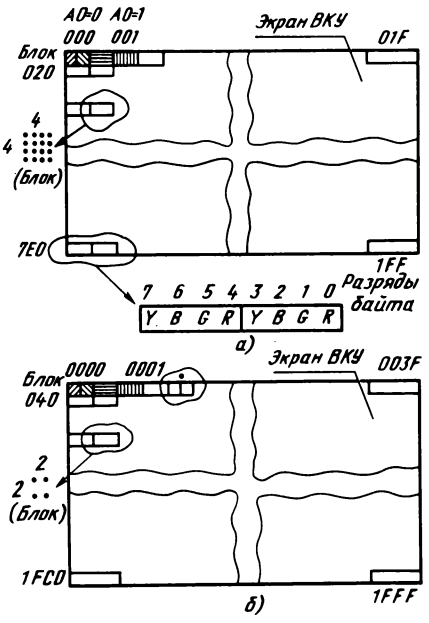


Рис. 3.6. Экран ВКУ в графичес ком режиме низкого (*а*) и среднего (*б*) разрешения

Таблица 3.3

Кодирование цвета

Код цвета		Цвет	Код цвета		Цвет
двоичный	шестнадцатеричный		двоичный	шестнадцатеричный	
0000	0	Черный	1000	8	Черный
0001	1	Красный	1001	9	6 дополнительных цветов
0010	2	Зеленый	1010	A	
0011	3	Желтый	1011	B	
0100	4	Синий	1100	C	
0101	5	Сиреневый	1101	D	
0110	6	Голубой	1110	E	
0111	7	Белый	1111	F	Белый

на две тетрады (по четыре бита в каждой). Блоки в режиме низкого разрешения имеют размеры 4 × 4 точки, а в режиме среднего разрешения 2 × 2 точки (рис. 3.6).

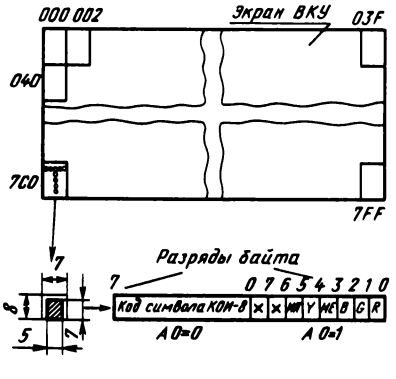
Старшая тетрада первого байта экранной строки (разряды 4 - 7) содержит код цвета блока, расположенного в левом верхнем углу экрана; младшая тетрада (разряды 0 -3) - код цвета следующего по строке блока и т.д. Для отображения одной строки блоков необходимы 32 байта в режиме низкого разрешения и 64 байта в режиме среднего разрешения. Для отображения одного полного кадра потребуется общая емкость видеоОЗУ 2 и 8К байт соответственно для режимов низкого и среднего разрешения. Возможные цвета и их кодировка приведены в табл. 3.3.

АЛФАВИТНО-ЦИФРОВОЙ РЕЖИМ

В алфавитно-цифровом режиме рабочее поле экрана разбивается на 1024 знакоместа (32 строки по 32 знакоместа в строке). Размер знакоместа 7 × 8 точек (8 строк по 7 точек в строке). В знакоместе может быть расположен один алфавитно-цифровой символ размером 5 × 7 (7 строк по 5 точек в строке). Полиграммы (матрицы изображения) символов являются стандартными, хранятся в ПЗУ знакогенератора и располагаются на знакоместе, как показано на рис. 3.7.

Каждому знакоместу экрана соответствуют два байта в видеоОЗУ. Первый (четный) байт, когда младший разряд байта равен нулю (АО = 0), содержит код

Рис. 3.7. Экран ВКУ в алфавитноцифровом режиме



символа (в коде КОИ-8), выводимого на данное знакоместо. Второй (нечетный) байт, когда АО = 1, содержит следующую управляющую информацию:

X - отмечены разряды, оставленные для резерва (принимают любое значение);

R, G, B, Y - разряды кода цвета символов;

ИП - разряд инверсии подсвета, управляет выводом символа на экране в нормальном или инверсном виде. Символ в нормальном виде выводится тем цветом, код которого задан в разрядах цвета, на темном фоне. Символ в инверсном виде выводится черным на фоне; цветного знакоместа, цвет которого задан кодом разрядов цвета;

ME - разряд мерцания, управляет режимом мерцания символа. Мерцающий символ выводится на экран поочередно то нормальным, то инверсным (с частотой переключения 5 Гц).

Коды разрядов ИП и ME приведены в табл. 3.4.

Таблица 3.4

Управление выводом символов на экран

Коды разрядов		Вывод символа на экран
ИП	ME	
0	0	Инверсный
0	1	Мерцающий
1	0	Нормальный
1	1	Нормальный

ДОПОЛНИТЕЛЬНЫЙ АЛФАВИТНО-ЦИФРОВОЙ РЕЖИМ

В этом режиме экран разбивается на 2040 знакоместа (число знакомест в строке увеличивается до 64). При этом соотношение между формами знакоместа и символа остается прежним, как и в режиме с 32 символами в строке, соответственно 7 × 8 и 5 × 7, а точка изображения на экране имеет вдвое меньший размер по строке (рис. 3.8). Это достигается увеличением тактовой частоты выво-

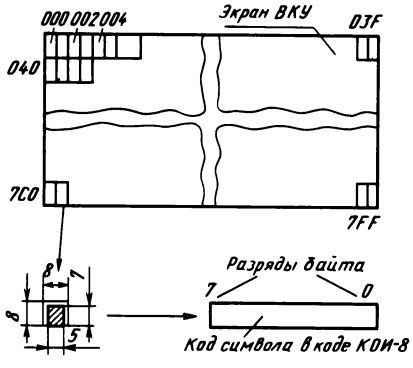


Рис. 3.8. Экран ВКУ в дополнительном алфавитно-цифровом режиме

да символов до 10,5 МГц. Каждому знакоместу на экране соответствует 1 байт в памяти экрана.

Разряды 0 - 7 задают код символа (в коде КОИ-8), выводимого на данном знакоместе.

Емкость памяти экрана сохраняется в тех же пределах, что и для режима с 32 символами в строке - 2К байт. Изображение черно-белое. Возможен вывод на экран мерцающих символов (или их частей, вплоть до отдельной строки).

ПЕРЕКЛЮЧЕНИЕ РЕЖИМОВ И ЭКРАННЫХ СТРАНИЦ

В ПЭВМ любая из областей ООП может быть отображена на экране ВКУ в любом из пяти возможных режимов отображения информации. Для переключения режимов отображения информации и указания отображаемой области ОПП (экранной страницы) отведена часть адресов ячеек ввода-вывода (256 адресов): С700 - С7FF.

Для включения экранной страницы (ЭС) в том или ином режиме достаточно изменить содержимое ячейки памяти по соответствующему адресу из области адресов С700 - С7FF. При этом действует принцип нерелевантности данных: данные могут быть произвольными, важен только сам адрес. В связи с этим ячейки с адресами С700 - С7FF называют ***программными переключателями***,, так как переключение осуществляется в соответствии с адресом. Программные переключатели режимов отображения экранных страниц приведены в табл. 3.5. В пределах одной экранной страницы содержится четыре экранных подстраницы.

Как и экранные страницы, подстраницы имеют самостоятельную нумерацию от 0 до 31. Заметим, что число экранных (текстовых) страниц превышает в 2 раза число экранных подстраниц. В зависимости от исполнения ПЭВМ основная оперативная память, размещенная на объединительной плате, может содержать:

4 экранных страницы и 16 экранных подстраниц - для "Агата" исполнения 7;

16 экранных страниц и 64 экранных подстраницы - для "Агата" исполнения 9.

Кроме того, номер экранной подстраницы задает и режим вывода алфавитно-цифровой информации - АЦР-32 или АЦР-64:

для младших экранных подстраниц (0 - 31), находящихся в пределах экранных страниц 0 - 7, обеспечивается АЦР-32;

для старших экранных подстраниц (32 - 63), находящихся в пределах экранных страниц 8 - 15, обеспечивается АЦР-64. При выводе информации на экран в АЦР-64 все четные экранные подстраницы обеспечивают нормальное отображение символов на экране - белые символы на черном фоне экрана, и все нечетные экранные подстраницы обеспечивают инверсное отображение символов - черные символы на белом фоне.

Для "Агата" исполнения 7, где ООП имеет емкость 32К байт, т.е. не существует ОЗУ для старших экранных подстраниц, в АЦР-64 вместо старших экранных подстраниц выводятся младшие. Таким образом, указание нулевой текстовой страницы выводит на экран нулевую подстраницу в АЦР-32; указание 32-й текстовой страницы отображает на экране нулевую подстраницу в нормальном режиме АЦР-64, а указание 33-й текстовой страницы - нулевую подстраницу в инверсном АЦР-64. Во всех трех случаях на экране отображается нулевая подстраница, но в режиме, определяемом по номеру указанной текстовой страницы.

Переключение экранных страниц

Таблица 3.5.

Отображаемая область ОПП			Программные переключатели экранных режимов и страниц				
№	ЭС	ЭПС	ГНР	ГСР	АЦР-32	АЦР-64	ГВР
0	0000	0000 07FF	C700	C701 C705 C709 C70D	C702	C782	C703 C707 C708 C70F
1	0	0800 0FFF	C704		C706	C786	
2		1000 17FF	C708		C70A	C78A	
3	1FFF	1800 1FFF	C70C		C70E	C78E	
4	2000	2000 27FF	C710	C711 C715 C719 C71D	C712	C792	C713 C717 C718 C71F
5	1	2800 2FFF	C714		C716	C796	
6		3000 37FF	C718		C71A	C79A	
7	3FFF	3800 3FFF	C71C		C71E	C79E	
8	4000	4000 47FF	C720	C721 C725 C729 C72D	C722	C7A2	C723 C727 C72B C72F
9	2	4800 4FFF	C724		C726	C7A6	
10		5000 57FF	C728		C72A	C7AA	
11	5FFF	5800 5FFF	C72C		C72E	C7AE	
12	6000	6000 67FF	C730		C732	C782	

Продолжение табл.3.5

Отображаемая область ОПП			Программные переключатели экранных режимов и страниц				
№	ЭС	ЭПС	ГНР	ГСР	АЦР-32	АЦР-64	ГВР
13	3	6800 6FFF	C734	C731 C735 C739 C73D	C736	C786	C733 C737 C738 C73F
14		7000 77FF	C738		C73A	C78A	
15	7FFF	7800 7FFF	C73C		C73E	C78E	
16	8000	8000 87FF	C740	C741 C745 C749 C74D	C742	C7C2	C743 C747 C73B C73F
17	4	8800 8FFF	C744		C746	C7C6	
18		9000 97FF	C748		C74A	C7CA	
19	9FFF	9800 9FFF	C74C		C74E	C7CE	
20	A000	A000 A7FF	C750	C751 C755 C759 C75D	C752	C7D2	C752 C757 C75B C75F
21	5	A800 AFFF	C754		C756	C7D6	
22		B000 B7FF	C758		C75A	C7DA	
23	BFFF	B800 BFFF	C75C		C75E	C7DE	
24	8000	8000 87FF	C760		C762	C7E2	
25	6	8800 8FFF	C764	C761 C765 C769 C76D	C766	C7E6	C763 C767 C76B C76F
26		9000 97FF	C768		C76A	C7EA	

Продолжение табл. 3.5

Отображаемая область ОПП			Программные переключатели экранных режимов и страниц				
№	ЭС	ЭПС	ГНР	ГСР	АЦР-32	АЦР-64	ГВР
27	9FFF	9800 9FFF	C76C		C76E	C7EE	
28	A000	A000 A7FF	C770		C772	C7F2	
29	7	A800 AFFF	C774	C771 C775 C779 C77D	C776	C7F6	C773 C777 C77B C77F
30		B000 B7FF	C778		C77A	C7FA	
31	BFFF	B800 BFFF	C77C		C77E	C7FE	

Примечание. ГНР, ГСР, ГВР - графические режимы соответственно низкого, среднего и высокого разрешения; АЦР - алфавитно-цифровой режим; ЭС - экранная страница; ЭПС - экранная подстраница.

3.5. ОРГАНИЗАЦИЯ ВВОДА-ВЫВОДА

В данной главе уже упоминалось о ячейках ввода-вывода как части адресного пространства, необходимого для организации обмена информацией между процессором и внешними устройствами.

Функции ввода-вывода в ПЭВМ "Агат" могут быть разделены на две категории:

функции, выполняемые объединительной платой с помощью специальных аппаратно-программных средств, образующих встроенный интерфейс ввода-вывода (ВИ В/В);

функции, выполняемые интерфейсными модулями, образующими внешний интерфейс ввода-вывода и устанавливаемыми в разъемы внутреннего системного интерфейса.

ВНЕШНИЙ ИНТЕРФЕЙС ВВОДА-ВЫВОДА

Под внешний интерфейс ввода-вывода зарезервированы ячейки ввода-вывода адресного пространства микропроцессора C100 - C8FF и C090 - C0FF. Рассмотрим, как распределяются эти адреса.

На объединительной плате расположен ряд из семи разъемов (рис. 3.9), на которых реализован внутренний интерфейс ПЭВМ. В шести из этих разъемов (за исключением X2) могут быть установлены любые из интерфейсных модулей внеш-

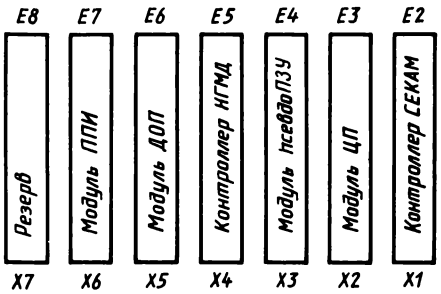


Рис. 3.9. Разъемы системного интерфейса

него оборудования (контроллеры принтера и накопители на магнитных дисках, дополнительная память и т.д.).

На всех разъемах, кроме X2, имеется два контакта для передачи специальных сигналов управления:

$\overline{DS}$ (DIVICE SELECT) - выбор устройства;

$\overline{I/OS}$ (I/O SELECT) - организация ввода-вывода.

Эти сигналы становятся активными (устанавливаются в логический нуль) в разные моменты времени, т.е. не может быть такой ситуации, когда оба сигнала одновременно соответствуют нулю. Наличие активного сигнала $\overline{DS}$ или $\overline{I/OS}$ соответствует обращению процессора в область адресов, зарезервированную за данным разъемом в соответствии с табл. 8.3.

Таким образом, обмен информацией и управление интерфейсными модулями внешних устройств осуществляется микропроцессором через соответствующие области адресов, зарезервированные за данным разъемом, к которому подсоединен модуль.

Более подробно организация работы внешнего интерфейса приведена в гл. 8.

ВСТРОЕННЫЙ ИНТЕРФЕЙС ВВОДА-ВЫВОДА

Встроенный интерфейс ввода-вывода является неотъемлемой частью объединительной платы, обеспечивает управление и обмен информацией со следующими внешними устройствами: громкоговорителями, кассетными магнитофонами, потенциометрическими пультами, блоком клавиатуры.

Под встроенный интерфейс ввода-вывода отводится 128 ячеек ввода-вывода адресного пространства микропроцессора с адресами C000 - C07F.

Распределение функций встроенного интерфейса ввода-вывода между ячейками:

C000 - C009 ..... **Ввод данных с клавиатуры**
C010 - C01F..... **Очистка буфера клавиатуры**
C020 - C02F..... **Управление выходом на кассетный магнитофон**
C030 - C03F..... **Управление громкоговорителем**
C040 - C04F..... **Разрешение таймерных прерываний**
C050 - C05F..... **Сброс таймерных прерываний**
C060..... **Ввод информации с кассетного магнитофона**
C061..... **Ввод состояния кнопки 1**
C062..... **Ввод состояния кнопки 2**
C063..... **Ввод старшего разряда кода нажатой клавиши с клавиатуры**
C064..... **Ввод положения ручки потенциометра пульта 1**
C065..... **Ввод положения ручки потенциометра пульта 2**
C066..... **Резерв**
C067..... **Резерв**
C070 - C07F..... **Включение одновибраторов пультов**

Звуковая индикация компьютера реализуется с помощью громкоговорителя, расположенного внутри корпуса ПЭВМ. Управление громкоговорителем осуществляется программным переключателем C030 - C03F.

При каждом обращении по любому из этих адресов на громкоговоритель поступает импульс, длительность которого равна времени выполнения используемой команды. Обращаясь к адресам переключателя с определенной частотой, микропроцессор позволяет получать звуковой сигнал с любой частотой звукового диапазона.

Интерфейс ввода-вывода кассетного магнитофона преобразует информацию при записи и считывании с кассетного магнитофона программным способом.

Обращения микропроцессора по любому из 16 адресов C020 - C02F, повторяемые с определенной частотой, обеспечивают кодирование информации, записываемой на магнитную ленту. Программа для кодирования данных при записи на магнитную ленту включена в системный монитор. Она кодирует логический нуль сигналом частотой 1 кГц, логическую единицу - сигналом частотой 2 кГц (т.е. используется принцип частотной модуляции). Данные, вводимые с магнитофона, попадают на старший (D7) разряд шины данных микропроцессора при обращении последнего по адресу C060. Таким образом, состояние магнитофонного ввода может быть определено центральным процессором по значению разряда D7 путем просматривания ячейки C060.

Поступающую с магнитофона информацию, например, можно обработать специальной программой, входящей в системный монитор.

Заметим, что подобный способ хранения информации на магнитной ленте применим и для передачи данных по телефонной сети между двумя ПЭВМ. При этом во избежание помех можно применить другую частоту кодирования информации, написав для этого соответствующую программу.

Интерфейс потенциометрических пультов обеспечивает подключение к компьютеру двух пультов, каждый из которых имеет ручку и кнопку на два положения (включено - выключено). Сигналы от кнопок поступают на старший разряд (D7) шины данных при обращении микропроцессора по адресам C061 или C062 соответственно для первого или второго пульта. Если при просмотре разряда D7 ячейки C061 или C062 микропроцессор обнаруживает там нуль, это означает, что кнопка выключена; если единицу - включена.

Потенциометры пультов преобразуют угол поворота ручки потенциометра в длительность импульсов. Чтобы микропроцессор смог считать состояние потенциометров, он должен включить одновибраторы с помощью программного переключателя C070. Таким образом, обращение микропроцессора к ячейкам C070-C07F вызывает пуск одновибраторов, вырабатывающих импульсы, длительность которых определяется сопротивлением потенциометра, соответствующим углу поворота ручки потенциометра.

Микропроцессор может определить временной интервал импульсов между началом и окончанием работы одновибраторов, считывая состояние разряда D7 шины данных при обращении по адресу C064 или C065.

Интерфейс клавиатуры обеспечивает преобразование последовательного кода нажатой клавиши, поступающего с блока клавиатуры с параллельной и последующей передачей его на шину данных центрального процессора (ШДЦП).

При обращении микропроцессора к ячейкам C000-C00F на ШДЦП появляется код

последней нажатой клавиши. Этот код хранится в буфере интерфейса клавиатуры до поступления на него кода следующей нажатой кнопки. Микропроцессор может освободить буфер от принятого кода обращаясь к любому из адресов C010-C01F.

Для программы ввода с клавиатуры, входящей в системный монитор, признаком нажатой клавиши служит единица в старшем разряде кода, принятого на шину данных микропроцессора.

3.6. ПРЕРЫВАНИЯ ЦЕНТРАЛЬНОГО ПРОЦЕССОРА

В ПЭВМ "Агат" имеют место три линии ввода прерываний в МП:

$\overline{RES}$ - сброс;

$\overline{IRQ}$ - запрос прерывания;

$\overline{NMI}$ - немаскируемое прерывание.

Линии INT IN и INT OUT позволяют задавать приоритетность прерывания от внешних устройств по принципу гирлянды.

ЛИНИЯ СБРОСА $\overline{RES}$

Вход $\overline{RES}$ используется для установки микропроцессора в начальное состояние при первом или повторном пуске ПЭВМ ("холодный" или "теплый" старт). Шина $\overline{RES}$ подключена к кнопке СБР и системе автостарта. При нажатии кнопки СБР (блок клавиатуры) микропроцессор заканчивает выполнение текущей программы и заносит в счетчик команд некоторое фиксированное число, передавая управление программе монитора.

ЛИНИЯ ЗАПРОСА ПРЕРЫВАНИЯ IRQ

Требования к правильной работе входа запроса маскируемого прерывания $\overline{IRQ}$ являются более строгими по сравнению с прерываниями к другому входу прерывания $\overline{NMI}$. Это обусловлено тем, что по входу $\overline{IRQ}$ процессор будет прерываться каждый раз, когда сигнал на линии $\overline{IRQ}$ соответствует логическому нулю и сброшен флажок блокировки прерывания (специальный разряд I в регистре слова состояния процессора).

Рассмотрим последовательность операции в ходе обслуживания запроса прерывания:

1) на линии $\overline{IRQ}$ устанавливается уровень логического нуля устройством сопряжения с внешним оборудованием;
2) микропроцессор заканчивает выполнение текущей команды;
3) распознается запрос прерывания;
4) микропроцессор проверяет состояние внутреннего сигнала запрета прерывания (флажок 1). Если внутренний запрет установлен (т.е. I = 1), то прерывание игнорируется до тех пор, пока флажок не будет сброшен (I = 0);
5) микропроцессор начинает обслуживание запроса прерывания, автоматически устанавливается флажок блокировки прерывания (I = 1). Это гарантирует, что точно такое же прерывание не будет распознаваться вновь поелс окончания обслуживания данного запроса прерывания;

6) микропроцессор заносит в стек содержимое своих внутренних регистров, чтобы процессор смог закончить выполнение программы, которая была прервана. В стек заносится содержимое программного счетчика и регистра состояния. Содержимое каких-либо других регистров микропроцессора автоматически не сохраняется;
7) процессор осуществляет выборку вектора прерывания по $\overline{\text{IRQ}}$ из ячеек FFFE и FFFF;
8) микропроцессор начинает выполнять программу обработки запроса прерывания с адреса, указываемого вектором прерывания;
9) микропроцессор заканчивает обработку запроса прерывания; как правило, любая обработка должна заканчиваться одной и той же командой восстановления микропроцессора в состояние, непосредственно предшествующее прерыванию. Этой командой является команда RTI (возврат из прерывания), которая восстанавливает в стеке, счетчике команд и регистре состояния процессора те значения, которые были в них к моменту возникновения прерывания;
10) флажок блокировки прерывания должен быть установлен в исходное состояние, но не раньше, чем будет снят сигнал IRQ (установлен в высокий уровень логический единицы) во избежание дублирования прерывания;
11) микропроцессор приступает к выполнению прерванной программы.

В том случае, если требуется сначала сохранить, а потом восстановить содержимое остальных внутренних регистров микропроцессора (аккумулятора и индексных регистров), в программе обработки прерывания должны быть предусмотрены соответствующие процедуры сохранения и восстановления содержимого этих регистров.

Такая методика совместно с возможностью автоматического прерывания и автоматического возврата из прерывания допускает возникновение множественных прерываний, причем последующие прерывания могут прерывать обработку текущего. Эта возможность является одним из преимуществ использования стека, так как число последовательных прерываний ограничено только длиной стека. В стеке сохраняется 6 байт на каждое прерывание (если сохраняются все регистры) или 3 байт (если сохраняются только счетчик команд и слово состояния). Таким образом, в стеке ПЭВМ "Агат" могут быть сохранены 42 или 84 последовательности прерываний.

ЛИНИЯ НЕМАСКИРУЕМОГО ПРЕРЫВАНИЯ NMI

Немаскируемое прерывание $\overline{\text{NMI}}$ дает возможность прервать обработку прерывания в интересах более приоритетного устройства, которое не может ждать до сброса маски прерывания (флажка I). Последовательность операций при обслуживании немаскируемого прерывания такая же, как и при обслуживании запроса прерывания, за исключением пунктов, связанных с флажком, и адресов, по которым центральный процессор осуществляет выборку вектора.

Вектор прерывания $\overline{\text{NMI}}$ выбирается из ячеек FFFA и FFFB.

Если запрос прерывания и немаскируемое прерывание произойдут одновременно или немаскируемое прерывание произойдет до момента выборки векторов, процессор всегда дает более высокий приоритет немаскируемому прерыванию. Немаски-

руемое прерывание всегда является более приоритетной линией с малой задержкой и может в любой системе использоваться для приписывания более высокого приоритета скоростному устройству.

ЛОКАЛИЗАЦИЯ ВЕКТОРОВ МОНИТОРА

Адреса векторов прерывания входят в состав адресов, закреп ленных за постоянной памятью. Физически это может быть реализовано на микросхемах ПЗУ с зашитым в нее системным монитором или платах оперативной памяти, работающих в режиме псевдоПЗУ. Если в первом случае не представляется возможным изменить что-либо в ячейках FFFA-FFFF, то во втором случае это связано с программными трудностями.

Системный монитор переадресует векторы прерываний микропроцессора на свои векторы, расположенные в области оперативной памяти.

Шестнадцатеричный адрес	Наименование вектора монитора
03F2	Вектор повторного пуска микропроцессора
03F3	Содержит адрес передачи управления после повторного выполнения команды RESET процессором
3FB, 3FC, 3FD	Вектор NMI содержит команду JMP, выполненную центральным процессором после обработки немаксируемого прерывания
3FE, 3FF	Вектор IRQ содержит адрес подпрограммы, на которую передает управление монитор после выполнения обработки прерывания

Г л а в а 4. ПРОГРАММИРОВАНИЕ В МАШИННЫХ КОМАНДАХ

4.1. ВВОДНАЯ ЧАСТЬ

Программирование в машинных командах является наиболее сложной работой на компьютере. Оно требует глубокого знания физических особенностей и принципов действия всех составных частей ПЭВМ, владения в совершенстве приемами программирования. Только научившись программировать на языке машины, можно стать профессиональным программистом. Не случайно создание программ иногда называют искусством.

Умение программировать обязательно придет с опытом, в результате постоянной работы на компьютере. Нужно только регулярно открывать новые и новые возможности персонального компьютера и не останавливаться на достигнутом.

Прежде чем приступить к программированию в машинных командах, следует внимательно изучить команды монитора. Системный монитор предоставляет самые разнообразные возможности, необходимые при программировании в системе ко-

манд: запись программы в память, проверка ее работоспособности (отладка), выполнение и т.п.

Программирование практически невозможно без знания машинных кодов, т.е. системы команд микропроцессора, и архитектуры (логического построения) компьютера. Поэтому в даной главе продолжим рассмотрение архитектуры ПЭВМ, а с техникой программирования читатель может познакомиться по многочисленным публикациям (например [8, 11, 13, 14]).

4.2. РЕГИСТР СЛОВА СОСТОЯНИЯ МИКРОПРОЦЕССОРА

Выше говорилось о регистре слова состояния, как о наиболее важном узле микропроцессора и всего компьютера в целом. Именно использование этого регистра и делает возможной саму работу с компьютером.

Регистр состояния (РСС) представляет собой устройство, способное записать и хранить восемь двоичных разрядов. Каждый такой разряд в РСС иногда называют ***флажком***. Пятый флажок постоянно равен единице, а остальные семь флажков принимают нулевое и единичное значения в зависимости от команды, выполняемой микропроцессором:

C(Carry) - флажок заема-переноса;

Z(Zero) - флажок нулевого результата;

I(IRQ Disable) – блокировка прерывания по входу $\overline{\text{IRQ}}$;

D(Decimal Mode) – флажок десятичного режима;

B(BRK Cammand) флажок исполнения команды программного прерывания BRK;

V(Overflow) флажок переполнения, возникающего в арифметических командах;

N(Negative) - признак отрицательного числа.

Распределение флажков по разрядам РСС:

7	6	5	4	3	2	1	0
N	V	1	B	D	I	Z	C

Ипользование флажков РСС позволяет расширить программные возможности микропроцессора. Используя команды условного перехода, программа может выполнить действия в зависимости от состояния того или иного флажка. Это основная особенность компьютера принимать решение в зависимости от условия.

C(заем-перенос). Этот флажок используется в командах сложения и вычитания. При выполнении команды сложения разряд С становится равным единице, если есть перенос из последнего разряда, т.е. полученный 8-й разряд, не умещающийся в байте, не пропадает и переносится в разряд С.

Рассмотрим пример использования флажка С в команде **ADC** (сложить с учетом переноса) при сложении двух однобайтовых чисел. Последовательность действий при таком сложении:

сбросить флажок С в нуль;

сложить первые (младшие) восемь разрядов обоих чисел, переслать сигнал переноса (0 или 1) в разряд С;

сложить следующие (старшие) восемь разрядов вместе со значением флажка С.

Пример.

Выполним сложение двух двухбайтовых чисел: 00010010 11111011 и 01101000 10000001:

а) сложим младшие восемь разрядов:

```
  11111011
+ 10000001
----------
  01111100,
```

единица переноса пересылается в разряд С;

б) сложим старшие восемь разрядов:

```
  00010010
+ 01101000
         1 (из флажка С)
----------
  01111011
```

В результате получим 01111111 01111100.

При выполнении команды вычитания флажок С устанавливается в нуль, если был заем, и устанавливается в единицу, если займа не было.

Вычитание выполняется путем сложения уменьшаемого с вычитаемым в дополнительном коде, так что перенос при сложении в дополнительном коде служит той же цели, что и заем в обычном вычитании. Представим инвертированный сигнал С ($\bar{C}$) в качестве сигнала заема при выполнении вычитания двух и более однобайтовых чисел.

Так, при вычитании двух 16-разрядных чисел необходимо заполнить разряд переноса при сложении младших восьми разрядов и учесть этот сигнал в качестве сигнала входного переноса при сложении следующих восьми разрядов.

Выполним в качестве примера вычитание следующих 16-разрядных чисел:

01100001 10001001 уменьшаемое

01000001 00010001 вычитаемое.

Сначала находим представление числа 01000001 00010001_2 в дополнительном коде, для чего к обратному коду прибавим единицу:

```
обратный код вычитаемого числа      10111110 11101110
                                                    +1
                                     ----------------
дополнительный код                   10111110 11101111
```

Выполним сложение:

```
          01100001 10001001
        + 10111110 11101111
          ------------------
перенос 1  перенос 1
```

Полученный ответ является положительной разностью двух 16-разрядных чисел, а имеющийся перенос из крайнего левого разряда указывает на то, что знак результата совпадает со знаком уменьшаемого (С=1). Заема не было, так как уменьшаемое было больше вычитаемого.

Если по окончании каждого процесса вычисления сигнал окончательного переноса отсутствует, то С=0, т.е. должен был быть окончательный заем. Это означает, что вычитаемое было больше уменьшаемого.

Флажок C используется в командах сдвига ASL, LSR, ROL, ROR как девятый разряд. Команда SEC устанавливает флажок C в единицу, а команда CLC - в нуль.

Z - **флажок нулевого результата.** Разряд Z устанавливается в единицу, когда результат выполнения команды равен 0000 0000. Если результат отличен от нуля, то флажок устанавливается в нуль.

I - **блокировка прерывания** $\overline{\text{IRQ}}$. Этот флажок устанавливается в единицу командой SEI и в нуль командой CLI. Когда I = 1, запрещается обработка прерывания IRQ; когда I = 0, разрешается прерывание по входу $\overline{\text{IRQ}}$. На вход $\overline{\text{NMI}}$ этот разряд влияния не оказывает.

Начав обработку прерывания $\overline{\text{IRQ}}$, микропроцессор автоматически устанавливает I = 1. Если нужно еще раз повторить обработку $\overline{\text{IRQ}}$, то предварительно необходимо установить I = 0. Команда BRK (программное прерывание) устанавливает I = 1.

D - **флажок десятичного режима.** Микропроцессор 6502 в отличие от других микропроцессоров (6800, 8080, Z80 и т.д.) может работать в двух режимах - двоичном и двоично-десятичном.

Выбор режима определяется состоянием флажка D: при D = 0 микропроцессор складывает и вычитает в двоичном коде; при D = 1 - в двоично-десятичном. Команда CLD сбрасывает флажок D в нуль, команда SED устанавливает D = 1.

B - **флажок исполнения команды** BRK. При исполнении этой команды флажок B = 1.

V - **флажок переполнения.** Используется при сложении и вычитании со знаком. В восьмибитовом числе указателем знака числа служит старший 7-й разряд: нулевое значение определяет положительное число; единичное значение - отрицательное число. Следовательно, для представления числа отводится только семь битов.

Флажок V устанавливается в единицу, когда результатом сложения двух отрицательных чисел является положительное число, или когда при сложении положительных чисел получено отрицательное число.

Флажок V устанавливается в единицу и в следующих случаях:

у результата нет переноса из 6-го разряда в 7-й, но есть перенос из 7-го разряда во флажок C;

есть перенос из 6-го разряда в 7-й и нет переноса из 7-го разряда во флажок C.

Флажки V и C имеют следующие различия:

V используется в арифметических операциях над числами со знаком и показывает перенос из 6-го разряда в 7-й; флажок C используется, в арифметических операциях над числами без знака, а также указывает на перенос из старшего 7-го разряда результата;

V, в отличие от C, не участвует в командах сдвига;

есть команда установки в нуль флажка V(CLV), но нет команды установки V в единицу; флажок C устанавливается в единицу командой CLC;

V может быть установлен в единицу внешним сигналом, подаваемым на вход S.O. микропроцессора 6502; флажок C используется только программно.

N - **отрицательный флажок**. Устанавливается 7-м разрядом (знаковым битом) результата выполнения команды. Команд установки флажка N нет. Существуют две команды BMI и BPL, которые проверяют состояние флажка N и осуществляют переходы по его значениям.

4.3. МЕТОДЫ АДРЕСАЦИИ МИКРОПРОЦЕССОРА 6502 (СМ630)

Число и структура способов адресации определяют возможности микропро цессора по обработке данных.

Исполнительный (действительный) адрес - это реальный номер ячейки памяти, начиная с которой располагаются адресуемые данные или команды.

Микропроцессор 6502 обладает наиболее мощной системой адресации из всех микропроцессоров второго и третьего поколений. Микропроцессор К580 (INTEL 8080/8085) насчитывает четыре способа адресации, MS 6800 (СМ601) - семь, Z80/Z80A (U880) - четыре, К1801 - шесть, а микропроцессор 6502 - 13 способов адресации:

аккумуляторная (**Accumulator Addressing**) - ACC;

неявная (**Implied Addressing**) - IMPL;

непосредственная (**Immediate Addressing**) - IMM;

прямая длинная (**Absolute Addressing**) - ABS;

прямая короткая (**Zero Page Addressing**) - ZP;

индексированная по X, короткая (**Indexed Zero Page Addressing . with Register X**) - ZP,X;

индексированная по Y, короткая (**Indexed zero Pade Addressing wiht Register Y**) - ZP,Y;

индексированная по X, длинная (**Indexed Absolute Addressing with Register X**) - ABS,X;

индексированная по Y, длинная (**Indexed Absolute Addressing with Register Y**) - ABS,Y;

относительная адресация (**Relative Addressing**) - REL;

индексно-косвенная по X (**Indexed Indirect Addressing with Register X**) - IND,X;

косвенно-индексная по Y (**Indexed Indirect Addressing with Register Y**) - IND,Y;

косвенная длинная (**Absolute Indirect Addressing**) - IND.

На рисунках, иллюстрирующих методы адресации, используются следующие сокращения:

KK - код команды;

КСК - код следующей команды;

PC - счетчик команд;

PCH - старший байт PC;

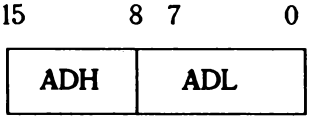
PCL - младший байт PC;

BA - базовый адрес, т.е. адрес, к которому прибавляется смещение (индекс);

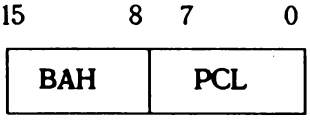
BAL - младший байт базового адреса;

BAH - старший байт базового адреса;

AD - абсолютный адрес КСК или младший байт базового адреса;
ADH:ADL - 16-битовый адрес, образованный **ADH** и **ADL**:



BAH:PCL - 16-битовый адрес, образованный **BAH** и **PCL**:



EA - исполнительный (действительный) адрес;
EAH - старший байт действительного адреса;
EAL - млаший байт действительного адреса;
ADL - младший байт абсолютного адреса;
ADH - старший байт абсолютного адреса.

Аккумуляторная адресация. Этот способ адресации используют четыре однобайтовые команды: **ASL, LSR, ROL, ROR**. Операнд, участвующий в операции, содержится в аккумуляторе.

Команды выполняются за два цикла следования импульсов основной тактовой частоты. Во время первого цикла извлекается код команды и содержимое счетчика команд PC увеличивается на единицу. Во втором цикле PC выставляет на адресную шину адрес следующей выбираемой из памяти команды. Пока текущая команда не выполнится, содержимое счетчика команд на единицу не увеличится.

Неявная адресация. При неявной адресации адрес операндов определяется самим кодом операции.

Команды, использующие этот способ адресации, можно разделить на четыре группы:

1. Команды, работающие с внутренними регистрами и флажками регистра состояния микропроцессора: **CLC, CLD, CLI, CLV, DEX, DEY, INY, INX, NOP, SEC, SED, SEI, TAX, TAY, TSX, TXA, TXS, TYA**. Эти команды выполняются так же, как команды с аккумуляторной адресацией.

2. Команды PHA и PHP записи в стек. Эти команды исполняются за три цикла. Команда PHA записывает содержимое аккумулятора в **MS** (ячейка памяти, адрес которой является номером в указателе стека) и уменьшает содержимое указателя стека на единицу. Команда PHP выполняет аналогичные действия с регистром P.

3. Команды **PLA** и **PLP** чтения из стека. Эти команды исполняются за четыре цикла импульса $\varphi 0$. Команда **PLA** увеличивает содержимое регистра **S** на единицу, затем считывается содержимое стека из ячейки, номер которой указан в регистре указателя стека (S), и загружается в аккумулятор. Команда PHP выполняет аналогичные действия с регистром P.

4. Специальный случай. Три команды **BRK, RTS, RTI** используют неявную адресацию, но выполняются по-разному:

RTS применяется при возвращении из подпрограммы. При ее выполнении микропроцессор читает два байта из стека и загружает младший и старший адрес PC.

При каждом чтении байта из стека содержимое указателя стека увеличивается на единицу;

BRK устанавливает флажок **B** в единицу, заносит содержимое PC и P в стек, а затем загружает в PC содержимое ячеек памяти с адресами **FFFE** и **FFFF**, т.е. вектора запроса прерывания $\overline{\text{INQ}}$;

RTI является командой возврата из прерывания. По этой команде микропроцессор читает три байта из стека и заносит их соответственно в **P**, **PCL** и **PCH**. Продолжение программы начинается с адреса, занесенного в **PC**.

Непосредственная адресация. Команды, использующие этот метод адресации, - двухбайтовые. Первый байт содержит код команды, второй - операнд (рис. 4.1). Имеется всего 11 команд, использующих непосредственную адресацию: **ADC # data, AND #data, CMP # data, CPX # data, CPY # data, EOR # data, LDA # data, LDX # data, LDY # data, ORA # data, SBC # data.**

Эти команды работают с регистрами **A**, **X**, **Y**. Непосредственная адресация на языке ассемблера помечается знаком #, стоящим перед непосредственным операндом (данными).

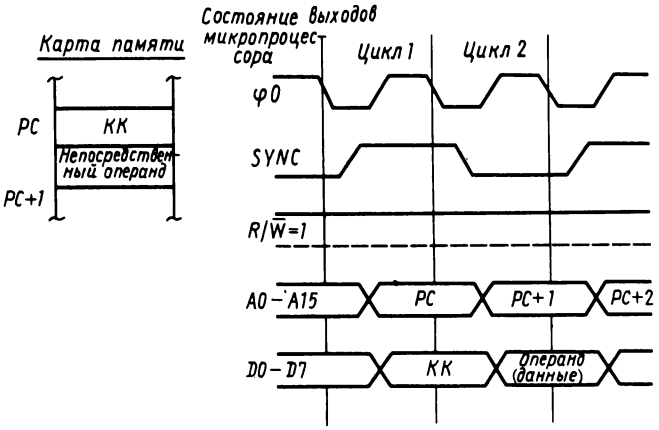
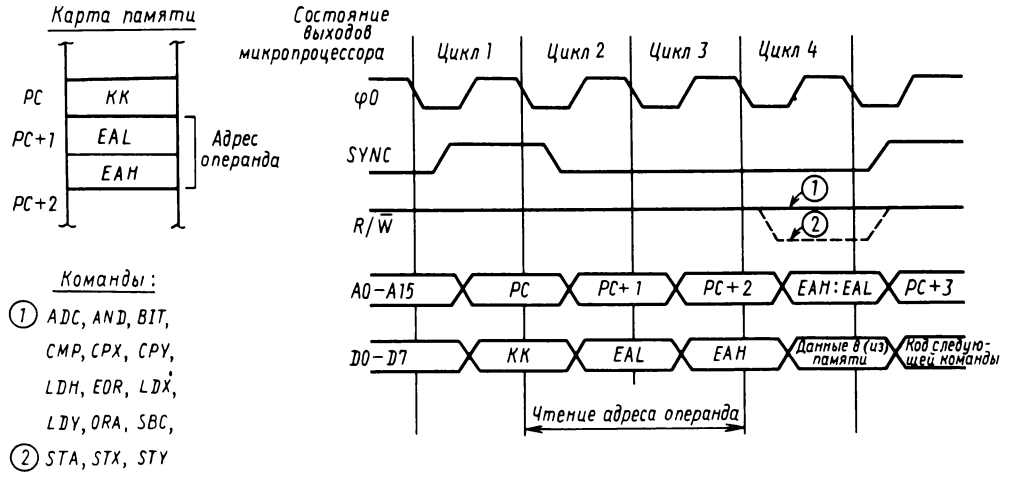


Рис. 4.1. Непосредственная адресация

Рис. 4.2. Прямая длинная адресация (случай 1)



Прямая длинная адресация. Команды, использующие этот метод адресации, - трехбайтовые. Первый байт содержит код команды, второй байт - младший байт действительного адреса **EA** (**Effective Address**) операнда, а третий байт - старший байт **EA**. Этот метод адресации позволяет обратиться к любой из 65 536 ячеек памяти. Используются 23 команды, которые делятся на четыре группы.

1. Логические и арифметические команды (**ADC, AND, BIT, CMP, CPX, CPY, EOR, ORA, SBC**), команды записи во внутренние регистры микропроцессора (**LDA, LDX, LDY**) и команды записи содержимого внутренних регистров в память (**STA, STX, STY**). Перечисленные 15 команд выполняются за четыре цикла (рис. 4.2).

2. Команды сдвига **ASL, SLR, ROL, ROR**, команда **INC** увеличения на единицу и команда **DEC** уменьшения на единицу. Эти шесть команд выполняются за шесть машинных циклов.

3. Команда **JMP** безусловного перехода. Команда исполняется за три цикла. Второй и третий байт содержат адрес, который нужно загрузить в **PC**, т.е. куда нужно перейти.

4. Команда обращения к подпрограмме **JSR** (рис. 4.3). Команда выполняется за шесть циклов.

Прямая короткая адресация. При этом способе команда адресуется к ячейкам нулевой страницы, т.е. к 256 байтам с 0000 по 00FF. Старший байт действительного (исполнительного) адреса всегда равен 00, а младший **EAL** изменяется от 00 до **FF**. Применение этого метода сокращает время выполнения команды. Команды, использующие этот метод, делятся на две группы.

1. Команды, выполняющиеся за три такта. Это арифметические и логические команды: **ADC, AND, BIT, CMP, CPX, CPY, EOR, ORA, SBC**, команды загрузки регистров **LDA, LDX, LDY** и команды запоминания регистров **STA, STX, STY**.

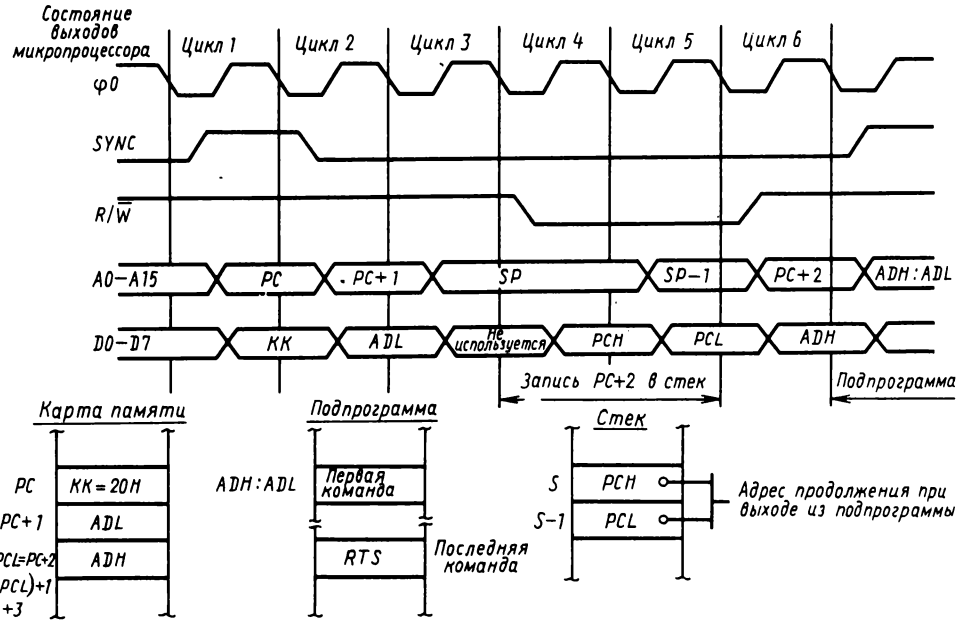


Рис. 4.3. Выполнение команды JSR

2. Команды, исполняющиеся за пять циклов: **ASL, LSR, ROL, ROR, DEC, INC.**

Индексированная по X, короткая. Команды, использующие этот метод адресации, - двухбайтовые. Первый байт содержит код команды, а второй байт указывает на адрес в нулевой странице. Исполнительный адрес формируется как сумма по модулю 2^8 второго байта команды и содержимого индексного регистра X. При суммировании старший (девятый) разряд переноса не учитывается. Таким образом, адресоваться за пределы нулевой страницы (00 - FF) при этом способе нельзя. При попытке адресоваться за пределы нулевой страницы формируется исполнительный адрес: 00.

Команды, использующие этот метод адресации, делятся на две группы.

1. Команды, выполняющиеся за четыре цикла (рис. 4.4). Это арифметические и логические команды (**ADC, AND, CMP, EOR, ORA, SBC**), команды загрузки и выгрузки (**LDA, LDY, STA, STY**).

2. Команды, выполняющиеся за шесть машинных циклов (6 тактовых импульсов $\varphi 0$), - **ALS, LSR, ROL, ROR, DEC, INC.**

Индексированная по Y, короткая.. Этот метод адресации использует индексный регистр Y и выполняется аналогично предыдущему. Используются две команды **LDX** и **STX**, которые выполняются за четыре такта $\varphi 0$.

Индексированная по X, длинная. Все команды, использующие этот метод адресации, - трехбайтовые: первый байт содержит код команды, второй - младший байт базового адреса, третий - старший байт базового адреса. Действительный адрес операнда образуется суммированием базового адреса с содержимым индексного регистра X (называемого также ***смещением***, или ***индексом***). Этот способ адресации используют 15 команд, которые можно разделить на две группы.

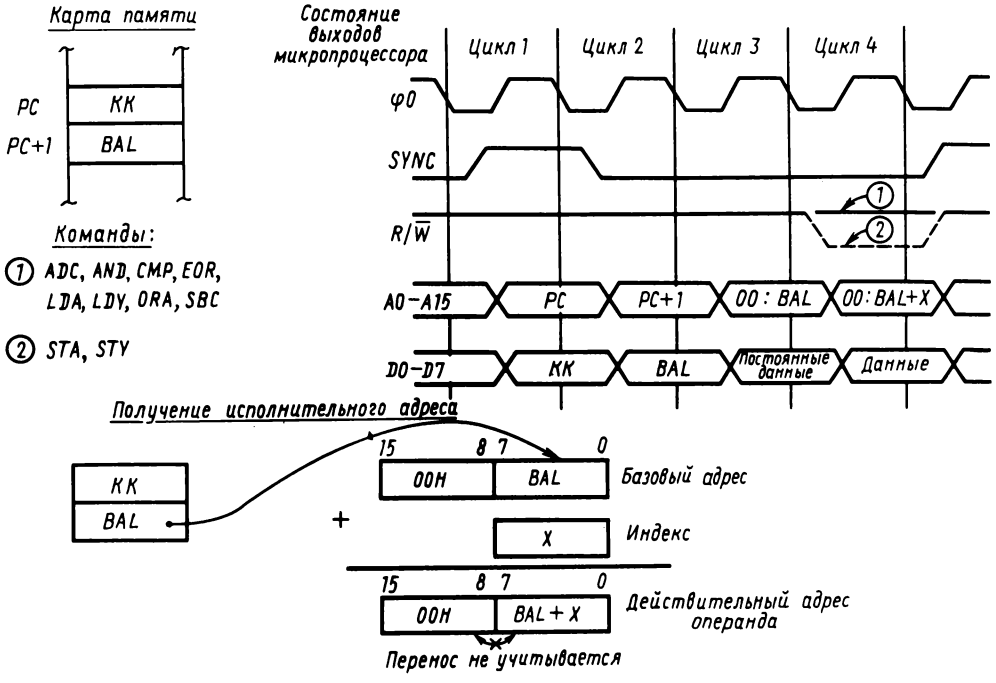


Рис. 4.4. Прямая короткая индексированная по X адресация

1. Команды **ADC, AND, CMP, EOR, LDA, LDY, ORA, SBC, STA**, выполняемые за четыре или пять циклов в зависимости от того, есть или нет переход из одной страницы в другую, т.е. есть или нет перенос от младшего к старшему байту при формировании исполнительного адреса. На рис. 4.5 показана диаграмма выполнения команд, соответствующая второму случаю.

2. Команды **ASL, LSR, ROL, ROR, DEC, INC**, выполняемые за четыре цикла. **Индексированная по Y, длинная**. Этот метод адресации используют девять команд: **ADC, AND, CMP, EOR, LDA, LDX, ORA, SBC, STA**. Он выполняется аналогично предыдущему, с той лишь разницей, что вместо регистра **X** применяется регистр **Y**.

Заметим, что последние два способа адресации учитывают перенос в старший байт, возникающий при сложении младшего байта с содержимым соответствующего индексного регистра.

Относительная адресация. Метод относительной адресации (рис. 4.6) используется в командах перехода. Команды - двухбайтовые:
первый байт содержит код команды, второй - смещение от -128_{10} до $+127_{10}$.

Если смещение отрицательное, то оно указывается в дополнительном коде. Смещение прибавляется к содержимому счетчика команд, в котором имелся адрес первого байта команды, расположенной непосредственно за командой перехода. При этом 7-й (старший) разряд смещения задает фактически направление перехода, а разряды 0 - 6 - значение перехода, т.е. если старший бит смещения равен единице, то дополнительный код смещения вычитается из увеличенного на 2 текущего адреса команды; если же старший бит смещения равен нулю, то смещение прибавляется к увеличенному на 2 текущему адресу команды. Например, размещенная по адресу 1000H команда **BNE** с операндом 03H (1000:D0

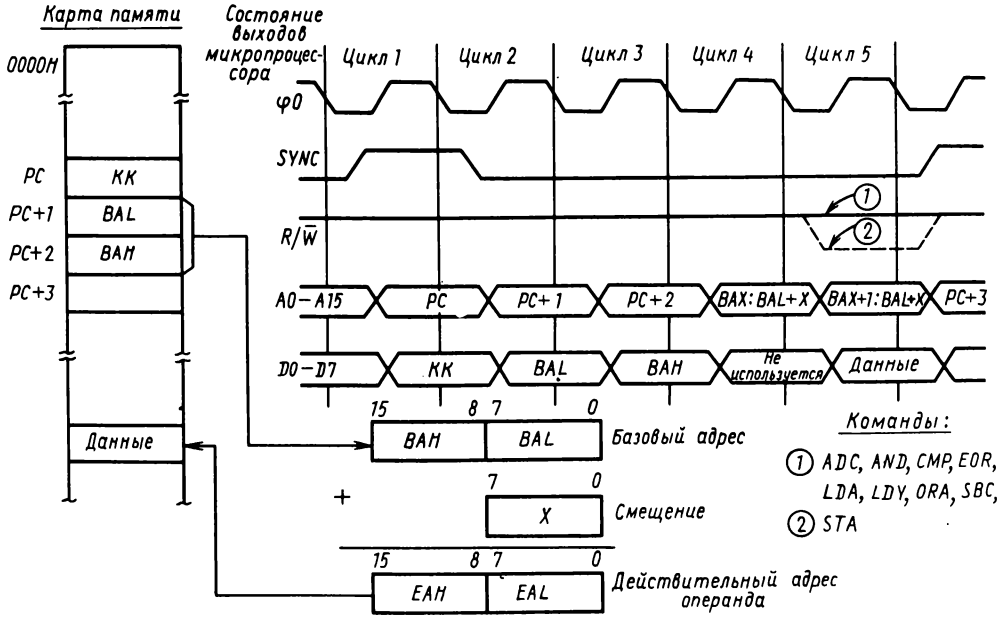


Рис. 4.5. Индексированная по X длинная адресация

03) передает управление по адресу 1005, а при операнде F3H - по адресу FF5H (символ H означает, что число, стоящее перед ним, шестнадцатеричное). Заметим, что разность адреса комады условного перехода и адреса, на который происходит переход, должна быть не более 81H при переходе "вперед" и не более 7EH при переходе "назад".

Относительный метод адресации используют восемь команд: BCC, BCS, BEQ, BMI, BNE, BPL, BVC, BVS. Те из них, которые не совершают перехода, выполняются за два цикла (например, не выполнено условие перехода); за три цикла, если переход осуществляется в рамках текущей страницы; за четыре цикла, если переход происходит за пределы текущей страницы. На рис. 4.6 приведена временная диаграмма для последнего случая.

Индексно-косвенная адресация по X. Команды, использующие этот метод адресации - двухбайтовые. Первый байт содержит код команды, второй - младший байт базового адреса, при этом старший байт базового адреса равен 00H. Младший байт прибавляется к содержимому индексного регистра X, которое рассматривается как смещение. В результате получается младший байт выбираемого адреса, чей старший байт равен 00H, т.е. перенос из младшего байта в старший (при суммировании) не учитывается. Так получается адрес, указывающий на ячейку в нулевой странице памяти. В этой ячейке содержится младший байт действительного адреса операнда, а в следующей ячейке (с адресом на единицу больше) находится старший байт действительного адреса операнда. Эти две ячейки обязательно должны находиться в пределах нулевой страницы. Данные (операнд) могут находиться в любой ячейке памяти.

Так, если сумма смещения и содержимого X равна FFH, то старший байт исполнительного адреса операнда лежит по адресу 0, а младший - по адресу

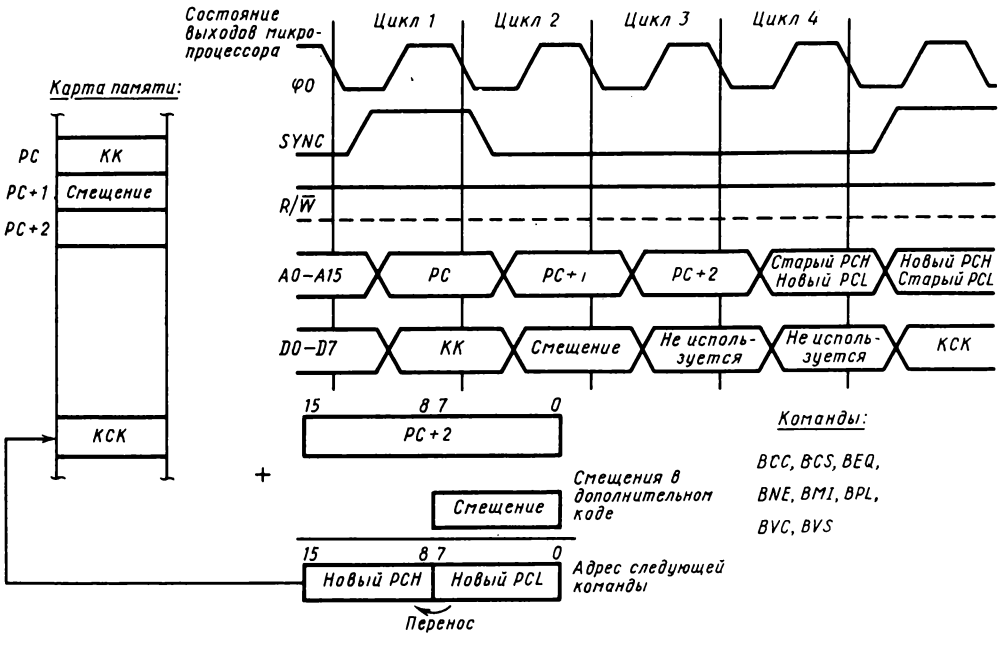


Рис. 4.6. Относительная адресация

FFH. Этот метод адресации используют восемь команд, выполняющихся за шесть циклов: **ADC, AND, CMP, EOR, LDA, ORA, SBC, STA**. Временная диаграмма работы команд приведена на рис. 4.7.

Косвенно-индексная адресация по Y. Этот метод существенно отличается от предыдущего, хотя команды, использующие этот способ адресации, тоже двухбайтовые: первый байт содержит код команды, второй - младший байт адреса в нулевой странице. В ячейке памяти по этому адресу содержится младший байт базового адреса, а в следующей ячейке - старший байт базового адреса.

Содержимое индексного регистра Y прибавляется к младшему байту базового адреса с учетом переноса, добавляемого к старшему байту базового адреса. В результате сложения получается 16-разрядный действительный адрес операнда. Операнд может находиться в любом месте оперативной памяти. Если в команде указывается адрес FFH, то старший байт базового адреса будет равен 00H, а младший байт базового адреса FFH.

Этот способ адресации используют восемь команд: **ADC, AND, CMP, EOR, LDA, ORA, SBC, STA**. Первые семь команд выполняются за пять или шесть циклов в зависимости от того, был или не был перенос от младшего к старшему байту, т.е. были или не были выходы за пределы текущей страницы. На рис. 4.8 показан второй случай. Команда **STA** всегда выполняется за шесть циклов.

Косвенная длинная адресация. С этим методом адресации используется только одна команда **JMP**. При косвенной адресации адрес, лежащий в поле операндов команды **JMP**, указывает на адрес, по которому должен быть произведен переход. Исполнительным адресом указателя адреса перехода является двухбайтовое слово, старшим байтом которого является третий байт команды, а младшим -

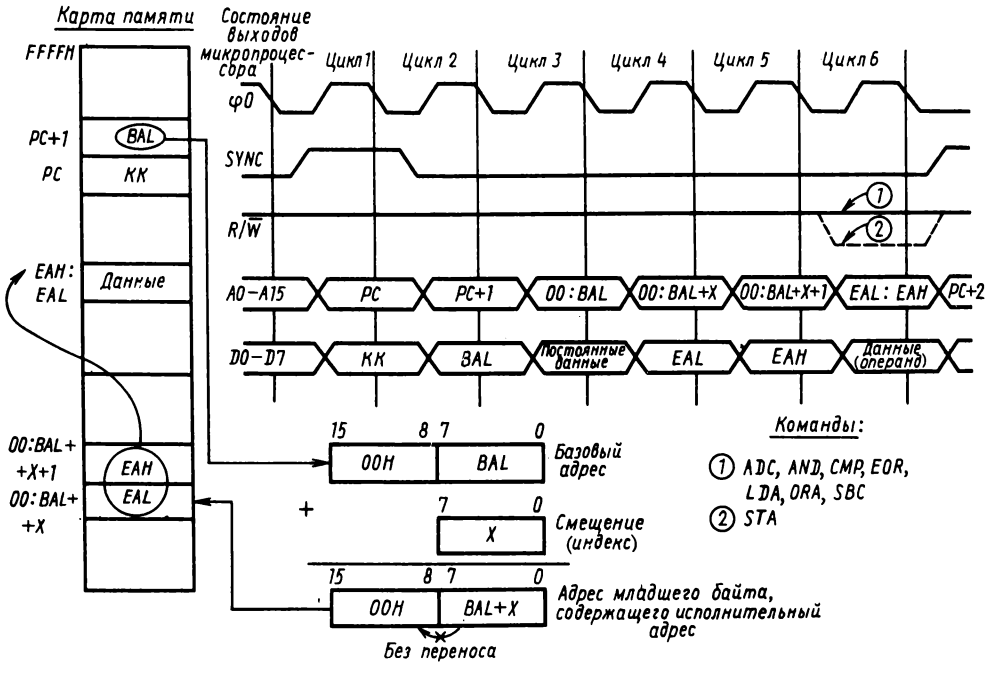


Рис. 4.7. Индексно-косвенная адресация с регистром X

второй байт команды. Адрес, лежащий в памяти, по которому производится переход, также представлен в следующем порядке: младший байт, старший байт исполнительного адреса. Если в команде JMP указывается адрес FFFFH, то старший байт исполнительного адреса перехода будет находиться по адресу 0, а младший - по адресу FFFFH. Диаграмма приведена на рис. 4.9.

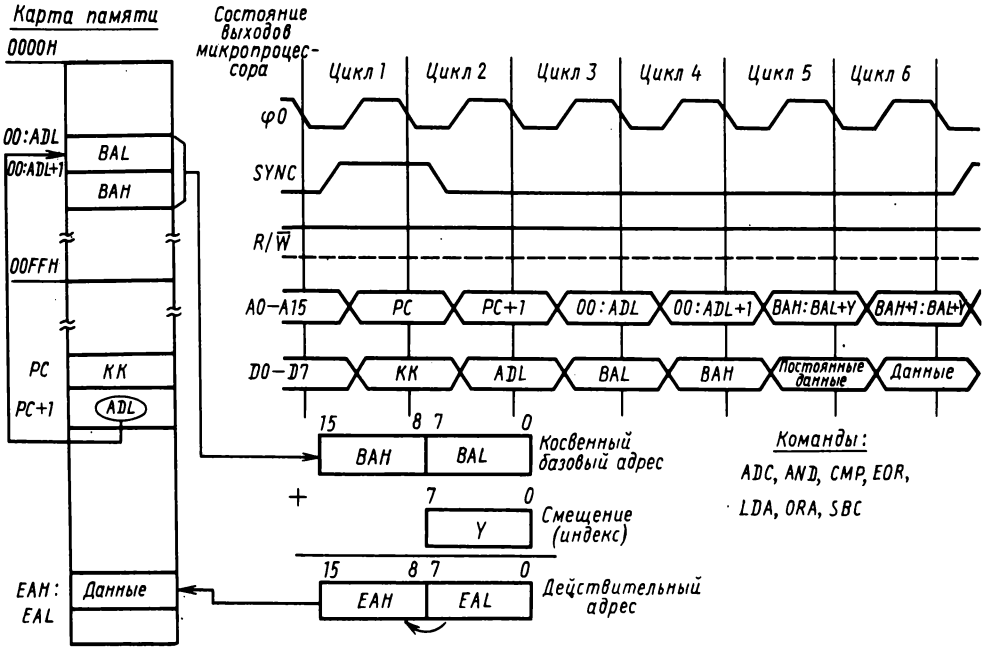


Рис. 4.8. Косвенно-индексная адресация

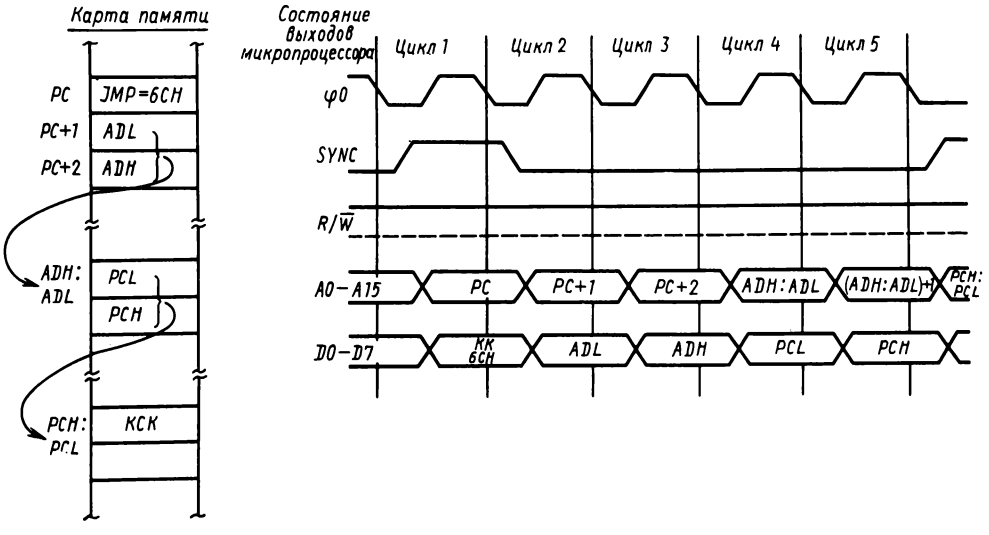


Рис. 4.9. Косвенная прямая адресация

4.4. ПРЕДСТАВЛЕНИЕ ДАННЫХ

В системе команд ПЭВМ "Агат" используются три типа данных (двоичный байт, десятичный байт, битовая строка) и адрес. Нумеруются разряды справа налево; младший разряд имеет номер 0, старший - 7.

Представление в виде двоичного байта. Двоичный байт представляет собой восьмиразрядный набор, который обычно кодируется двумя цифрами в шестнадцатеричном представлении. Соответствие между последовательностями двоичных разрядов и их десятичными и шестнадцатеричными представлениями приведено в гл. 1.

Для указания того, что используется шестнадцатеричное представление, в мнемонике ассемблера ПЭВМ применяется символ ¤ . Так, например, байт, в котором содержится десятичное число 0, в шестнадцатеричном представлении имеет вид ¤ 00 или ¤ 0. Байт, в котором содержится десятичное число 14, имеет вид ¤ F или ¤ 0E. Если описывается последовательность байтов в шестнадцатеричном представлении, то ¤ ставится только перед первым байтом. Например, ¤ 1F3EC4.

При операциях двоичного сложения и вычитания (**ADC** и **SBC**) двоичный байт может интерпретироваться либо как целое число без знака в диапазоне от 0 до 255, либо как целое число со знаком в диапазоне от -128 до +127 с представлением отрицательных чисел в дополнительном коде и использованием для индикации переноса в первом случае флажка C, а во втором - флажка **V**. В командах **ASL**, **CMP**, **CPX**, **CPY**, **LSR**, **ROL**, **ROR** допускается только первая интерпретация двоичного байта.

Представление данных в виде десятичного байта. Десятичный байт представляет собой набор двух двоично-десятичных цифр. При этом старшие и младшие четыре разряда байта кодируют по одной десятичной цифре (т.е. каждые четыре разряда кодируют число в интервале от 0 до 9). В числе, которое кодируется байтом, младшие четыре разряда кодируют младшую цифру десятичного числа, а старшие - старшую цифру. Таким образом, десятичный байт может интерпретироваться как целое число в интервале от 0 до 99. Имеются только две операции **ADC** и **SBC**, которые в десятичном режиме (т.е. при **D** = 1) могут выполнять операции над десятичными цифрами. Для индексации переноса в десятичном режиме используется признак **C**.

Представление данных в виде битовой строки. При представлении данных в виде битовой строки каждый разряд имеет самостоятельное значение.

Примером данных, представленных в виде битовой строки, может служить записываемый в память восьмибитовый регистр состояния **P** или значения однобитовых флажков регистра.

4.5. СИСТЕМА КОМАНД МИКРОПРОЦЕССОРА 6502

Команды персонального компьютера (их всего 56) группируются по следующим признакам [11, 14]:

команды загрузки и сохранения внутренних регистров микропроцессора 6502: **LDA**, **STA**, **LDX**, **STX**, **LDY**, **STY**, **TAX**, **TAY**, **TXA**, **TYA**, **TSX**, **TXS**, **NOP**;

команды увеличения и уменьшения на единицу (**INC**, **DEX**, **INX**, **INY**, **DEX**, **DEY**); команды сложения и вычитания (**ADC** и **SBC**); команды установки и сброса

флажков C и D (CLC, SEC, CLD и SED); команды выполнения логических операций (AND, ORA, EOR) и команда обнуления V (CLV);

команды условного перехода (BCC, BCS, BEQ, BNE, BMI, BPL, BVC, BVS), безусловного перехода (JMP); сравнения (CMP, CPX, CPY); сдвига (ASL и LSR) и циклического сдвига (ROL и ROR); побитного сравнения памяти и аккумулятора (BIT);

команды (JSP и RTS), работающие с подпрограммами; команды работы с прерываниями (CLI, SEI, RTI и BRK); команды записи в стек и чтения из стека (PHA, PHP, PLA, PLP).

Для удобства работы с системой команд 6502 рекомендуется использовать прил. 1, в котором приведена основная информация о них.

Ниже даны объяснения обозначений, используемых в прил. 1 и в этой и последующих главах:

(1) - увеличить на единицу число машинных тактов, за которые выполняется команда;

(2) - в командах, использующих относительный метод адресации (BCC, BCS, BEQ, BMI, BNE, BPL, BYS, BVC), возможны три варианта числа машинных циклов (тактов импульсов $\varphi 0$):

2, если нет перехода (когда не выполняется условие перехода);

2+1, если есть переход в пределах одной страницы;

2+2, если есть переход из одной страницы в другую;

(3) - инверсия разряда C определяет заем (C=заем);

(4) - в десятичном режиме флажок Z не устанавливается, поэтому аккумулятор необходимо проверить на нулевой результат;

X - индексный регистр X;

Y - индексный регистр Y;

A - аккумулятор;

M - ячейка памяти, определяемая исполнительным адресом (EA);

MS - ячейка памяти, адрес которой находится в указателе стека S;

"+" - сложение;

"-" - вычитание;

·AND· (или $\wedge$) - логическое И;

·OR·(или $\vee$) - логическое ИЛИ;

M7 - бит 7 ячейки памяти M;

M6 - бит 6 ячейки памяти M;

5 - число байтов;

· - отмеченный точкой флажок регистра P не используется.

$\longrightarrow$ (или $\longleftarrow$) - передача данных;

OPER - операнд;

- символ непосредственной адресации.

4.6. ПРОГРАММИРОВАНИЕ В СИСТЕМЕ КОМАНД МИКРОПРОЦЕССОРА 6502

Чтобы персональный компьютер стал надежным другом и помощником в решении любых прикладных задач (начиная от простых вычислений и кончая цветными мультфильмами со звуковым сопровождением), необходимо уметь управлять его работой.

Микропроцессор ПЭВМ выполняет ограниченный набор команд над восьмиразрядными данными. Заметим, что микропроцессор не умеет складывать числа, превышающие восемь разрядов, не может их умножать и делить. И уже тем более микропроцессор не в состоянии выполнить более сложные операции: извлечь квадратный корень, возвести в степень, вычислить тригонометрические и логарифмические функции и т.д.

Возможность компьютера решать эти и другие задачи зависит от умения программиста (так обычно называют пользователя, работающего с ЭВМ) точно сообщать микропроцессору абсолютно все, шаг за шагом, что следует делать для решения задачи в рамках тех элементарных команд, которые может выполнять микропроцессор.

Имеется одно очень важное обстоятельство, которое программист должен ясно себе представлять: если не известно, как решать задачу, то программирование последней для решения на ПЭВМ бессмысленно - компьютер не сможет ее решить. Это не означает, что программист должен знать ответ на поставленную задачу, однако он должен понимать, как ее решить. Для программирования работы персонального компьютера требуется описание процесса решения задачи с учетом возможностей ПЭВМ.

Словесное описание процесса решения задачи, которое позволило бы ее решить на компьютере, называется ***алгоритмом*** и является основной частью программирования. Когда алгоритим составлен, его можно закодировать, т.е. представить в виде команд микропроцессора (машинных команд), решающих задачу.

Еще одна проблема, возникающая перед программистом, связана с отладкой программы, т.е. с поиском и устранением ошибок. Понятно, что ошибки быстрее выявляются в процессе отладочного прогона программы.

Рассмотрим пример составления программы. Требуется сложить два однобайтовых числа, представленных в шестнадцатеричном коде: 1A и 7B. Словесное описание алгоритма можно представить следующим образом:

1. Очистить флажок переноса: C = 0.
2. Занести в аккумулятор число 1A : 1A → A.
3. Сложить содержимое A с числом 7B и флажком переноса C: A+7B+C→ A.
4. Конец.

Результатом выполнения этого алгоритма будет сумма чисел 1A и 7B, находящаяся в аккумуляторе. Соответствующая этому алгоритму программа на языке ассемблера микропроцессора 6502 имеет вид:

```
CLC          очистка флажка C;
LDA # $1A    занесение числа 1A в A;
ADC # $7B    A + 7B + C → A;
BRK          конец.
```

Соответствующая программа в машинных кодах запишется следующим образом:

```
18       C = 0;
A9 1A    1A → A;
69 7B    A + 7B + C → A;
00       конец.
```

Чтобы компьютер мог выполнить эту программу, ее необходимо разместить в оперативной памяти. Воспользуемся для этого программой "Системный монитор".

С помощью команд "Системного монитора" запишем приведенную выше программу в машинных кодах в ячейки памяти, начиная с адреса ¤ 300:

```
* 300 : 18 ↵
* 301 : A9 1A ↵
* 303 : 69 7B ↵
* 305 : 00 ↵
```

и выполним ее с помощью команды монитора GO (ВЫПОЛНИТЬ):

```
* 300 G ↵
```

Составим программу сложения двух любых двухбайтовых чисел. В отличие от предыдущего случая, эти числа размещаются в ячейках памяти. Результат располагается также в заданные ячейки памяти. Алгоритм этой программы:

1. Обнулить флажок С: С=0.
2. Содержимое ячейки памяти M1L, хранящей младший байт первого числа, поместить в аккумулятор: M1L → A.
3. Сложить содержимое аккумулятора с содержимым ячейки памяти M2L, хранящей младший байт второго числа:

   M2L + A + C → A.

4. Содержимое аккумулятора поместить в ячейку памяти M3L, предназначенной для хранения младшего байта результата.
5. Содержимое ячейки памяти M1H, хранящей старший байт первого числа, поместить в А: M1H → A.
6. Сложить содержимое аккумултора с содержимым ячейки памяти M2H, хранящей старший байт второго числа, с учетом флажка переноса С: M2H + A + + C → A.
7. Содержимое аккумулятора поместить в ячейку памяти M3H, предназначенной для хранения старшего байта результата.
8. Конец.

Программа на языке ассемблера, реализующая этот алгоритм, имеет вид:

```
CLC
LDA ¤ 1001
ADC ¤ 1003
STA ¤ 1005
LDA ¤ 1000
ADC ¤ 1002
STA ¤ 1004
BRK
```

Занесем программу в машинных кодах в ячейках памяти, начиная с адреса ¤ 300:

```
* 300: 18 ↵
```

```
*301: AD 01 10↵
*304: 6D 03 10↵
*307: 8D 05 10↵
*30A: AD 00 10↵
*30D: 6D 02 10↵
*310: 8D 04 10↵
*314: 00 ↵
```

Эту же программу можно было занести в память следующим образом:

```
*300: 18 AD 01 10 6D 03 10 8D 05 10 AD 00 10
      6D 02 10 8D 04 10 00 ↵
```

Перед пуском программы на выполнение необходимо поместить в ячейки 1000 и 1001 соответственно старший и младшие байты первого числа, а в ячейки 1002 и 1003 аналогичным образом поместить второе число. Например, поместим числа 3A27 и 4BA1:

```
*1000: 3A 27 ↵
*1002: 4B A1 ↵
```

После чего можно выполнить программу:

```
*300 G ↵
```

Результат (число 85C8) будет находиться в ячейках 1004 и 1005:

```
*1004.1005 ↵
1004 - 85 C8
```

Эта программа, в отличие от первой, универсальна по отношению к данным. Если в первой программе данные зашиты в саму программу и их нельзя поменять, не изменив самой программы, то во второй программе данные располагаются независимо от программы. Их можно изменить перед пуском программы, поменяв содержимое соответствующих ячеек.

Рассмотрим пример составления еще более сложной задачи: заполнить произвольным символом весь экран ВКУ. Алгоритм этой задачи:

1. Поместить в ячейку № 300 код символа, а в ячейку № 301 код режима вывода его на экран ВКУ.
2. Поместить в ячейку № 10 младший, а в № 11 старший байты адреса первого байта выбранной текстовой страницы.

3. Поместить в ячейки ¤ 9 и ¤ A соответственно младший и старший байты программного переключателя текстовой страницы.

4. Поместить в индексный регистр X число ¤ 8.

5. Обнулить индексный регистр Y.

6. Переместить код символа из ячейки ¤ 300 в аккумулятор.

7. Занести содержимое аккумулятора по адресу ¤ 10, используя косвенно-индексную адресацию с регистром Y. Исполнительный адрес ячейки памяти, куда будет помещен код символа, сформируется следующим образом: к содержимому ячейки ¤ 10 добавляется содержимое индексного регистра Y (это младший байт исполнительного адреса); старший байт выбирается из ячейки ¤ 11.

8. Увеличить содержимое индексного регистра Y на единицу.

9. Переместить код режима вывода символа из ячейки ¤ 301 в аккумулятор.

10. Занести содержимое аккумулятора по адресу ¤ 10, используя косвенно-индексную адресацию с регистром Y.

11. Увеличить содержимое индексного регистра Y на единицу.

12. Выполнить переход по ненулевому (Z=0) результату на п. 6.

13. Если после выполнения п. 11 результат (содержимое регистра Y) стал нулевым, т.е. Z=1, то увеличить содержимое ячейки ¤ 11 на единицу.

14. Уменьшить содержимое индексного регистра X на единицу.

15. Выполнить переход по ненулевому результату на п. 5.

16. Обратиться к программному переключателю, находящемуся в ячейках ¤ 9 и ¤ A, используя индексно-косвенную адресацию по X.

17. Конец.

При написании алгоритма была использована косвенно-индексная адресация по Y. Это позволяет изменять исполнительный адрес ячеек памяти, не изменяя самой программы. Например, можно записывать код символа в любую текстовую страницу.

В алгоритме использовано еще одно очень важное понятие - цикл. ***Цикл*** - это многократное повторение группы операторов. В приведенном алгоритме необходимо выполнить запись кода символа и кода режима в 2048 ячеек памяти. Используя регистр Y, каждый раз увеличивая его содержимое на единицу, алгоритм осуществляет запись в 256 ячеек. При этом в исполнительном адресе изменяется только младший байт, что и определяет перебор 256 ячеек (0 - FF). Чтобы заполнить все 2048 (0 - 7FF) ячеек, необходимо выполнить восемь таких переборов, каждый раз увеличивая старший байт исполнительного адреса на единицу.

Таким образом, в алгоритме организовано два цикла. Один из них внутренний, другой - внешний. Внутрений цикл изменяет содержимое 256 ячеек в пределах изменения младшего байта исполнительного адреса. Этого недостаточно для засветки всего экрана, поэтому, изменив старший адрес на единицу, следует опять повторить предыдущую процедуру перебора 256 ячеек (сбросив предварительно Y в 0) и т.д. Это и есть внешний цикл, определяющий число повторений внутреннего цикла. Переходы, организующие циклы, построены на операторах перехода по состоянию флажков.

Выбор той или иной команды перехода определяет сам программист.

На рис. 4.10 приведены текст программы на языке ассемблера и в машинных кодах. Программа может быть записана в любые ячейки памяти, лишь бы они не

```
      LDX #08       30A : A2  08
  MET:LDX #00       30C : A0  00
 MET1:LDA ¤30       30E : AD  00 03
      STA (¤10),Y   311 : 91  10
      INY           313 : C8
      LDA ¤301      314 : AD  01 03
      STA (¤11),Y   317 : 91  10
      INY           319 : C8
      BNE MET1      31A : D0  F4
      INC ¤11       31C : E6  11
      DEX           31E : CA
      BNE MET0      31F : D0  EE
      LDA (¤9,X)    321 : A1  09
      BRK           323 : 00
```

Рис. 4.10. Текст программы на ассемблере и в машинных кодах

пересекались с памятью выводимой текстовой страницы. Перед пуском программы нужно поместить в ячейки ¤ 300 и ¤ 301 код символа и режима вывода, в ячейки ¤ 10 и ¤ 11 - начальный адрес выбранной текстовой страницы, в ячейки ¤ 9 и ¤ A - адрес переключателя этой страницы.

Более сложные вопросы программирования в машинных кодах и на языке ассемблера рассмотрены в гл. 6.

Глава 5. БАЗОВОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

5.1. СИСТЕМНЫЙ МОНИТОР

Программа "Системный монитор" управляет работой всей ПЭВМ и одновременно подчиняется ей. Она контролирует все программы, и все программы используют ее.

Вы научились с помощью монитора просматривать одну, две или все ячейки памяти и изменять их содержимое. Теперь познакомимся с тем, как можно писать и выполнять программы не только на машинном языке, но и на языке ассемблера, записывать на магнитные кассеты бесконечные массивы данных и программ и снова их считывать.

Программа "Системный монитор" начинается с адреса $FF69_{16}$ (в десятичном представлении 65385 или -151). Монитор на языке БЕЙСИК может быть вызван командой:

CALL-151 или CALL 65385.

Знак готовности монитора (звездочка *) появляется слева на экране, а справа от него высвечивается мерцающий указатель - знак подчеркивания (_).

В ПЭВМ "Агат" имеются два монитора: один из них записан в ПЗУ компьютера,

а другой является составной частью языка БЕЙСИК. Отличаются мониторы курсором-указателем. Монитор ПЗУ имеет указатель - белый мерцающий квадрат (▫), монитор языка БЕЙСИК - белый мерцающий знак подчеркивания (_).

Запись содержимого памяти на магнитную ленту. В мониторе предусмотрены две специальные команды, которые позволяют переписывать содержимое области памяти на кассету и считывать его для дальнейшего использования. Одна из этих команд, команда WRITE (запись), записывает содержимое от 1 до 65 536 ячеек памяти на стандартные кассеты.

Чтобы записать содержимое области памяти на кассету, монитору необходимо задать начальный и конечный адреса этой области и символ W после них:

* {Начало}. {Конец} W ↵

Чтобы провести запись без ошибок, необходимо установить кассетное звукозаписывающее устройство в режиме "Запись", дать ленте прокрутиться несколько секунд, нажать клавишу ↵.

Десять секунд монитор будет записывать на ленту заголовок (частотой 3,7 кГц) и лишь потом данные. Об окончании работы монитор сообщает коротким сигналом. После записи рекомендуется перемотать ленту и разборчиво написать на кассете данные о содержании хранимых данных и их назначении. Например:

* 0:FFFFAD30C088D004C601F008CAD0F6A6004G020060↵
* 0.14↵

```
0000-FFFFAD30C088D004
C008-C601F008CAD0F6A6
0010-004G020060
```

* 0.14W↵

После десятисекундной записи заголовка для записи на ленту содержимого 4096 ячеек памяти требуется еще 35 с. Скорость записи составляет 1350 бит/с. Если значения всех величин записаны без ошибок, монитор пишет на ленту значение еще одной дополнительной величины - контрольную сумму, которая образована из значений всех величин записываемой области памяти. Команда READ использует потом эту сумму для обнаружения ошибки считывания (контрольная сумма имеет сначала значение ⌘ FF и изменяется посредством операции "исключающего ИЛИ", т.е. сложения по модулю 2, с каждой величиной передаваемой области).

Считывание области памяти с ленты. Если с помощью команды WRITE монитора на ленту записана область памяти, то можно считать ее командой READ монитора. Не обязательно считывать записанные данные на ленту в ту же область памяти.

Формат команды READ такой же, как и команды WRITE, только на месте буквы команды стоит "R":

{Начало}.{Конец} R

Прежде чем нажать ↵ , необходимо установить магнитофон в режим "Воспроизведение" и дождаться воспроизведения заголовка (монотонного звукового сиг-

нала частотой 3,7 кГц). Сигнал служит для настройки команды READ на чтение.

Команда WRITE пишет этот сигнал 10 с, а команда READ настраивается в течение 3 с. Например, запишем в память с адреса 0000 по 0014 данные (см. предыдущий пример):

```
* 0: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ↵
* 0.14 ↵

0000-00 00 00 00 00 00 00 00
0008-00 00 00 00 00 00 00 00
0010-00 00 00 00 00
* 0.14R ↵

* 0.14 ↵
0000-FF FF AD 30 C0 88 D0 04
0008-C6 01 F0 08 CA D0 F6 A6
0010-00 4C 02 00 60
```

После того, как монитор запишет в память все данные, он считывает контрольную сумму и сравнивает ее с собственной контрольной суммой. Если они различаются, то монитор дает звуковой сигнал и выводит на экран ERR. Таким образом, дается предупреждение о том, что при считывании возникли ошибки, и величины не совпадают с теми, которые были записаны на ленте. Если же контрольные суммы равны, то монитор ждет дальнейших инструкций.

Мини-ассемблер. Монитор состоит из совокупности подпрограмм, выполняющих самые разнообразные функции. Написанные пользователем программы на машинном языке могут вызывать многие подпрограммы монитора. Ниже приведена программа, которая выводит на экране буквы от A до Z:

```
* 3ØØ: A9 C1 20 ED FD 18 69 1 C9 DB D0 F6 60 ↵
* 3ØØ. 3ØC ↵

Ø3ØØ-A9 C1 20 ED FD 18 69 01
Ø3Ø8-C9 DB D0 F6 60
```

Ее выполнение задается командой G:

```
* 300G ↵

ABCDEFGHIJKLMNOPQRSTUVWXYZ
*
```

Фрагмент памяти в шестнадцатеричном виде неудобен для того, чтобы читать программы на машинном языке и при этом искать ошибки. Команда LIST позволяет вывести машинные программы на языке ассемблера. Ассемблер распознает 56 команд микропроцессора в мнемоническом коде при 13 способах адресации. По команде LIST на экран выводится программа:

```
* 300L
0300-  A9  C1      LDA # ¤ C1
0302-  20  ED  FD  JSR ¤ FDED
0305-  18          CLC
```

```
0306-  69  01      ADC# ¤ 01
0308-  C9  DB      CMP# ¤ DB
030A-  D0  F6      BNE  0302
030C-  60          RTS
030D-  00          BRK
```

Приведенная выше программа печати на экране букв от A до Z распечатана в формате языка ассемблера в виде отдельных команд.

В мониторе языка БЕЙСИК (в отличие от монитора в ПЗУ компьютера) есть подпрограмма, которая позволяет ввести программу в ПЭВМ в том же формате ассемблера, в котором ее выводит команда LIST, - это мини-ассемблер "Агат". Она называется "мини", так как не может обрабатывать условные адреса, что выполняет развитой ассемблер. Мини-ассемблер запускается следующим образом:

```
* F666G
!
```

Восклицательный знак является знаком готовности. При работе с мини-ассемблером можно задать любую команду монитора, если поставить перед ней ¤ . В противном случае мини-ассемблер использует свой собственный синтаксис и свои собственные команды.

Мини-ассемблер распознает счетчик команд монитора. Прежде чем ввести программу, нужно установить счетчик команд на адрес, в котором будет размещена программа. Для этого необходимо ввести этот адрес и двоеточие. Потом следует первая команда программы, представленная тремя буквами символа команды, пробел, операнд команды. Мини-ассемблер преобразует строку в шестнадцатеричные числа и размещает их в памяти начиная с адреса, на который указывает счетчик команд. Эти шестнадцатеричные числа (их может быть не более трех) переводятся, как при выполнении команды монитора LIST, снова в формат языка ассемблера и выводятся на экран. (Это дает возможность еще раз подробно просмотреть введенную строку.) Затем мини-ассемблер снова выводит восклицательный знак. Если команды следуют одна за другой, нет необходимости при вводе задавать адрес и двоеточие; можно набрать пробел, команду (символ команды, пробел, операнд) и написать клавишу ↵. Эта строка преобразуется, после чего можно ввести следующую строку и т.д.

В случае, если введенная строка содержит ошибку, мини-ассемблер сообщает об этом звуковым сигналом и выводит указатель (^) под ошибочным символом (или по крайней мере вблизи него). В основном это ошибки, возникающие при наборе: неверные символы команд, отсутствующие скобки и т.д. Мини-ассемблер не воспринимает также строки входных данных, если в них пропущен пробел перед или после символа команды или если внесен ошибочный знак в шестнадцатеричную величину или адрес. Если адрес цели относительного перехода расположен слишком далеко от команды перехода (удален более чем на 127 адресов от адреса команды перехода), то мини-ассемблер сообщает об этом как об ошибке. Например:

```
!300:    LDX    02 ↵
0300             A2·      02        LDX      # o 02
!LDA     ¤ 0,X
```

```
0302            B5      00      LDA     ¤ 00,X
!STA    ¤ 10,X
0304            95      10      STA     ¤ 10,X
!DEX
0306            CA              DEX
!STA    ¤ C03   0
0307            8D      30  CO  STA     ¤ C030
```

Чтобы выйти из мини-ассемблера и вернуться в монитор, необходимо нажать клавишу СБР или набрать команду монитора, начинающуюся знаком ¤ например:

```
!¤ FF69G
*
```

Теперь программа на языке ассемблера размещена в памяти. Ее снова можно просмотреть с помощью команды **LIST**.

Для облегчения поиска ошибок в программах, написанных в машинных кодах, в мониторе предусмотрены команды **STER** и **TRACE**:

STER - оценивает, выполняет и индуцирует машинную программу шаг за шагом;

TRACE - автоматически выполняет эти пошаговые действия и останавливается лишь тогда, когда выполнится команда **BRK**.

На следующем примере можно проследить, что происходит при пошаговом выполнении с помощью **STER** введенной программы на мини-ассемблере:

```
*   300S ↵
0   300-    A2      02          LDA     # ¤ 02
    A=0A    X=02    Y=D8    P=3 0   S=F8
*   S ↵
0   302-    B5      00          LDA     ¤ 00,X
    A=0C    X=02    Y=D8    P=3 0   S=F8
*   S ↵
0   304     95      10          STA     ¤ 10,X
    A=0C    X=02    Y=D8    P=3 0   S=F8
*   12 ↵
0   012 -   0C
*
```

После выполнения последней команды было проверено содержимое ячейки памяти о 12. Оно оказалось равным ожидаемому значению. Пошаговое выполнение можно продолжить дальше командой **TRACE**:

```
* T ↵
```

Команды монитора работы с текстом. К этим командам относятся следующие:

NT - вывод на экран ВКУ текстовой страницы (N - номер текстовой страницы);

NO - задание цвета текста, который выводится на экран ВКУ (N = 0 + 7). Каждая цифра кодирует цвет символов, выводимых далее на ВКУ.

С помощью команды **NT** можно вывести на экран ВКУ нужный участок памяти в

виде текста. При этом, если N = 0 + 31, текст на экран ВКУ будет выдаваться в режиме 32×32 символа, если N = 32 + 63 - в режиме 32×64 символа.

Следует помнить, что каждая текстовая страница занимает ▯ 800 байт памяти. Нулевая страница начинается с адреса 0. Адрес тексто вой страницы определяется по формуле

$$A = ▯\ 800 \times N.$$

Страница с N = 32 начинается, как и нулевая, с адреса 0. Попытка использовать страницы с нулевым адресом приводит к искажению содержимого рабочих ячеек системы и "зависанию" монитора.

С помощью команды NO можно задать цвет текста, который выводится на экран ВКУ. Каждая цифра N кодирует цвет выводимых на ВКУ символов.

Одной из возможностей, предоставляемых монитором, является использование его подпрограмм. Для этого (в программе пользователя или отдельно) следует выполнить команду JSR - переход на начальный адрес подпрограммы, задав при необходимости начальные параметры (загрузив требуемые ячейки памяти или регистры микропроцессора).

Ниже перечислены некоторые подпрограммы монитора (в скобках указаны адреса подпрограмм монитора языка БЕЙСИК):

COUT - подпрограмма подготовки вывода символа на экран. Адрес подпрограммы FDDC (FDD4). Выводимый символ должен быть в аккумуляторе.

COUT передает управление по адресу, находящемуся в CSW (ячейки ▯ 36, ▯ 37); обычно это подпрограмма COUT1.

COUT1 - подпрограмма выбода символа на экран. Адрес подпрограммы FDDF (FDD7).

COUT1 выводит символ в текущую позицию экрана. Содержимое всех регистров оставляет без изменений. Интерпретирует следующие управляющие коды:

8 - перевод строки;
88 - возврат на один символ влево;
95 - сдвиг на один символ вправо;
99 - перемещение на одну строчку вверх;
9A - перемещение на одну строчку вниз;
8C - очистка всего экрана;
9D - очистка до конца текущей строки;
9E - очистка до конца экрана;
87 - звуковой сигнал.

SETINV - подпрограмма установки инверсного режима. Адрес подпрограммы FE75 (FE77).

SETINV устанавливает инверсный режим для COUT1. Все символы будут выводиться как черные точки на цветном фоне. Изменяется содержимое регистра A.

SETNORM - подпрограмма нормального режима. Адрес подпрограммы FE79 (FE7B).

SETNORM устанавливает нормальный режим для COUT1. Символы выводятся как цветные точки на черном фоне. Изменяется содержимое регистра A.

CROUT - подпрограмма ввода символа ↓. Адрес подпрограммы FD8E (FD75).

CROUT осуществляет передачу символа (▯ 8D) подпрограмме COUT.

CROUT1 - подпрограмма возврата с чисткой экрана. Адрес подпрограммы FD89 (FD70).

CROUT1 очищает экран от текущей позиции курсора до угла (конца) текстового окна, затем вызывает CROUT вводом в подпрограмму COUT символа ¤ 9E (УПР-И, функциональная "8").

PRBYTE - подпрограмма шестнадцатеричной печати байта. Адрес подпрограммы FDC9 (FDC1).

PRBYTE выводит содержимое аккумулятора в шестнадцатеричной системе. Содержимое регистра A изменяется.

PRHEX - подпрограмма печати шестнадцатеричной цифры. Адрес подпрограммы FDD2 (FDDA).

PREHX выводит последние четыре бита аккумулятора как одну шестнадцатеричную цифру. Содержимое аккумулятора уничтожается.

PRNTAX - подпрограмма печати содержимого регистров A и X. Адрес подпрограммы F932 (F932).

PRNTAX выводит содержимое регистров A и X как четыре шестнадцатеричные цифры. Аккумулятор содержит первый выводимый байт, регистр X - второй. Содержимое аккумулятора не сохраняется.

PRBLNK - подпрограмма печати двух пробелов. Адрес подпрограммы FF5A (FF5A).

PRBLZ - подпрограмма печати нескольких пробелов. Адрес подпрограммы FF5C (FF5C).

PRBLZ - выводит от 1 до 256 пробелов.

В регистре X должно содержаться число выводимых пробелов. Если регистр X=000, то выводится 256 пробелов.

BELL - подпрограмма вывода звукового сигнала (УПР-Г). Адрес подпрограммы FCB4 (FCAE).

Может быть вызвана вводом в COUT символа ¤ 87.

RDKEY - подпрограмма ввода входного символа. Адрес подпрограммы FDOA (FDO4).

RDKEY переходит к подпрограмме ввода, адрес которой лежит в ячейке KSW (¤ 38, ¤ 39); обычно это программа KEYIW.

PDCHAR - подпрограмма РЕД - код. Адрес подпрограммы FB7A (FB66).

RDCHAR - дополнительная подпрограмма ввода, которая вводит символы, обращаясь к RDKEY, и при этом интерпретирует следующие редактирующие клавиши:

"↑" - перемещение курсора на одну позицию вверх;

"↓" - перемещение курсора на одну позицию вниз;

РЕД - переход в режим свободного горизонтального перемещения;

"→" - перемещение на одну позицию вправо (в свободном перемещении);

"←" - перемещение на одну позицию влево (в свободном перемещении).

Интерпретируемые функции выполняются после нажатия соответствующей клавиши и не приводят к завершению программы. Выход из режима свободного перемещения происходит по нажатию любой, отличной от стрелок влево и вправо клавиши.

KEYIN - подпрограмма считывания с клавиатуры. Адрес подпрограммы FDOD (FD07).

KEYIN читает с клавиатуры, ожидая пока нажмут клавишу. После считывания кода символа KEYIN передвигает мерцающий курсор и возвращает код символа в аккумулятор.

GETLN - подпрограмма взять входную строку. Адрес подпрограммы FD6E (FD55).

GETLN выводит приглашение из ячейки о 33. Собирает символы входной строки во входной буфер (начинающийся с адреса о 200, пока не встретится знак "↓").

В регистре X накапливается длина входной строки.

GETLNZ - вход в подпрограмму GETLN. Адрес FD6B (FD4F).

Перед тем, как передать управление GETLN, курсор устанавливается в начало строки.

WAIT - подпрограмма задержки. Адрес подпрограммы FBCB (FB94).

WAIT производит временную задержку, а затем передает управление основной программе. Время задержки зависит от содержимого аккумулятора.

Если в аккумуляторе содержится величина A, то время задержки будет (5*A*A+27*A+26)*2 мкс.

5.2. ДИСКОВАЯ ОПЕРАЦИОННАЯ СИСТЕМА (ДОС)

ЗАГРУЗКА ДОС

Дисковая операционная система ПЭВМ "Агат" предназначена для создания, уничтожения и работы с массивами данных (файлов) пользователя на ГМД. ДОС обрабатывает три типа файлов, при этом оставляя программисту возможность расширить набор типов [15, 17, 19].

Пользователь, работая с компьютером, имеет возможность присвоить любому массиву (набору) данных имя и запомнить его на ГМД. Это может быть, например, программа в шестнадцатеричных командах процессора или в командах языка БЕЙСИК. Каждому такому набору (или просто файлу) при записи на ГМД ДОС автоматически присваивает тип: шестнадцатеричной программе - тип B, программе на языке БЕЙСИК - тип A. Введение типов позволяет индифицировать файлы, упрощает работу с ними.

Различаются файлы и по имени. Имя однозначно характеризует файл, поэтому одинаковых имен для разных файлов задавать не следует - это неизбежно приведет к ошибке.

ДОС может обмениваться данными одновременно с 16 файлами, обслуживая при этом до 10 НГМД (тип ЕС 5088.02). Обмен с накопителем происходит со скоростью 2,5К байт/с.

При начальной загрузке программы ДОС переносится в память автоматически с учетом конфигурации ОЗУ ПЭВМ.

Провести первоначальную загрузку можно включением компьютера в сеть и с помощью следующих команд:

*CX00G - системного монитора;

]PR#X - языка БЕЙСИК;

X УПР P ⌐ - управляющих клавиш БК.

Здесь везде X - номер разъема, в котором установлен контроллер НГМД (в базовом исполнении - 3), например:

*C300 G,]PR#3, 3 УПР P ⌐.

Признаком того, что ДОС загрузилась, служит появившаяся в момент загрузки на экране ВКУ звездочка (*) без мигающего курсора.

ДОС, как и монитор, работает в диалоге с пользователем. Она воспринимает командную строку, введенную с клавиатуры, распознает ее и выполняет. Если в ПЗУ, кроме ДОС загружена система программирования, например, на языке БЕЙСИК, то в этом случае командная строка после анализа в ДОС передается в систему программирования.

Прежде чем рассматривать команды ДОС, введем обозначения:

s - номер разъема, в котором установлен НГМД (2-6);

d - номер привода, под которым подключен НГМД (1-2); в базовом исполнении ПЭВМ с одним НГМД d = 1;

v - номер тома, сформированный при инициализации ГМД.

Параметры команды, заключенные в фигурные скобки, являются необязательными и могут опускаться. При этом используются значения, установленные предыдущей командой.

По умолчанию в ДОС s соответствует номеру разъема, к которому подключен контроллер НГМД, d = 1, v = 254.

ОБЩИЕ КОМАНДЫ ДОС

CATALOG {,Ss} {,Dd} {,Vv} - чтение каталога ГМД. По этой директиве на экран дисплея выводится каталог ГМД.

В каталоге ГМД по каждому из записанных на ней файлов отображается:

тип файла (A, I - программа на языке БЕЙСИК, B - двоичный, T- текстовый);

размер файла в секторах (в одном секторе 256 байт);

имя файла.

INIT **имя файла** {,Ss} {,Dd} {,Vv} - инициализация ГМД. ГМД находится в работе около 1 мин. Если никаких сообщений не выдано, то в каталоге ГМД можно увидеть единственный файл с указанным именем.

При инициализации ГМД следует учитывать предполагаемый характер его использования и конфигурацию ПЭВМ "Агат", на которой ГМД будет эксплуатироваться. Если ГМД должен обеспечивать "холодный" пуск компьютера, в комплект которого не входит модуль ПЗУ (либо когда предполагается загружать с ГМД систему программирования, отличную от размещенной в ПЗУ), следует дополнить описанную процедуру инициализации записью на ГМД двоичного файла, содержащего интерпретатор языка БЕЙСИК или другую систему программирования. Имя этого файла указывается в директиве INIT. Поскольку при разметке записывается файл типа A, для записи интерпретатора на ГМД требуется предварительно удалить этот файл директивой DELETE или переименовать его директивой RENAME.

RENAME **файл 1, файл 2** {,Ss} {,Dd} {,Vv} - переименование файлов. Здесь файл 1 - старое имя файла, файл 2 - новое имя файла.

DELETE **имя файла** {,Ss} {,Dd} {,Vv} - уничтожение файлов.

LOCK имя файла {,Ss} {,Dd} {,Vv} - защита файла на запись. Файл, защищенный от записи, в каталоге помечается звездочкой (*).

UNLOCK имя файла {,Ss} {,Dd} {,Vv} - отменяет защиту от записи в файл.

MON {,C} {,I} {,O} - вызывает отображение на экране команд и информации, выполняемых в программном режиме. Указание C разрешает вывод на экран команд, I - текста, вводимого с ГМД, O - текста, выводимого на ГМД. Хотя бы одно из указаний должно присутствовать.

NOMON {,C} {,I} {,O} - отключает вывод на экран информации, заданной командой MON. Запятые в командах MON и NOMON можно опускать. По умолчанию устанавливается NOMON, C, I, O.

MAXFILESn - резервирует n буферов для активных файлов размером 256 байт каждый. При начальной загрузке n=3.

Работа с файлами типа A (программы на языке БЕЙСИК). Для работы с файлами типа A используются следующие команды:

LOAD имя файла {,Ss} {,Dd} {,Vv} - загружает программу с ГМД. Программа и данные, находящиеся в памяти, теряются.

SAVE имя файла {,Ss} {,Dd} {,Vv} - записывает программу из памяти на ГМД; если на ГМД уже есть файл с таким именем, он теряется и заменяется на новый.

RUN имя файла {,Ss} {,Dd} {,Vv} - загружает и запускает программу с ГМД. Программа и данные в памяти теряются.

CHAIN имя файла {,Ss} {,Dd} {,Vv} - загружает и запускает программу с ГМД. Программа в памяти теряется, данные передаются запускаемой программе.

Работа с файлами типа B (двоичные). Далее A - начальный адрес, L - длина двоичного файла. Требуется указывать десятичное значение либо шестнадцатеричное с признаком ¤ (например, A ¤ 1000, ¤ 4906).

BLOAD имя файла {,Aa} {,Ss} {Dd} {,Vv} - загружает файл с адреса a. Если адрес a не указан, загрузка выполняется с адреса, указанного при записи на ГМД. Адрес и длина загруженного файла остаются в ячейках ¤ AA72-AA73 и ¤ AA60-AA61.

BSAVE имя файла {,Aa} {,Ss} {,Dd} {,Vv} - записывает указанную зону памяти на ГМД.

BRUN имя файла {,Aa} {,Ss} {,Dd} {,Vv} - то же, что и в BLOAD, но с последующей передачей управления на начальный адрес файла.

ПРИМЕНЕНИЕ КОМАНД ДОС В ПРОГРАММАХ

Все команды диалога ДОС предоставляются для программных обращений в виде, описанном выше. Строка, содержащая команду, посимвольно передается на вывод подпрограммой COUT программы "Системный монитор", но перед ней выводится символ с кодом ¤ 84 (CHR ¤ (4) на языке БЕЙСИК или УПР M на клавиатуре). Программы на языке БЕЙСИК передают команды ДОС оператором PRINT. Отображение этой строки на экране выполняется в зависимости от состояния параметра C команды MON/NOMON.

Следует учитывать, что управляющий код должен выводиться в первой позиции строки, гарантией чего может служить вывод перед ним кода КОНЕЦ СТРОКИ, или ¤ 8D(CHR ¤ (13), или PRINT на языке БЕЙСИК.

Пример программного выполнения команды CATALOC в программе на языке БЕЙСИК:

```
110 PRINT:PRINT CHR ¤ (4); "CATALOC"
```

в программе на языке ассемблера:

```
! LDX#0
! ЦИКЛ: LDA КОМАНДА, X
! JSR COUT
! INX
! CPX # > КОНЕЦ - КОМАНДА + 1
! BNE ЦИКЛ
! RTS
! КОМАНДА : ¤ 8D84
! "CATALOC"
! КОНЕЦ: ¤ 8D
```

Кроме команд диалога для программных обращений предоставляется еще набор команд работы с текстовыми файлами. Эти команды имеют аналогичный текстовый формат, но не разрешены в прямом диалоге. Ниже описаны также программные обращения к ДОС, позволяющие проводить обмен с ГМД по физическим адресам.

Работа с текстовыми файлами (типа T). Предусмотрено два вида текстовых файлов: файлы с последовательным доступом, в которых хранится сплошная последовательность символов, и файлы с прямым доступом, содержащие записи фиксированной длины (b - номер байта в файле с последовательным доступом или в записи файла с прямым доступом, r - номер записи в файле с прямым доступом, j - размер записи файла с прямым доступом).

OPEN **имя файла** {,Ss} {,Dd} {,Vv} - открытие файла для последовательного доступа.

OPEN **имя файла**, ,Lj {,Ss} {,Dd} {,Vv} - открытие файла для прямого доступа.

Открытие отсутствующего файла приводит к созданию пустого файла, открытие файла с именем, имеющимся на ГМД, позиционирует файл на начало.

CLOSE **имя файла** {,Ss} {,Dd} {,Vv} - закрытие файла для последовательного доступа.

CLOSE **имя файла** {,Ss} {,Dd} {,Vv} - закрытие файла для прямого доступа.

WRITE **имя файла** {,Bb} - запись в файл для последовательного доступа.

WRITE **имя файла** {,Rr} {,Bb} - запись в файл для прямого доступа.

После выполнения команд OPEN и WRITE любой выход из программы, осуществляемый с помощью подпрограммы COUT или команды PRINT, направляется в открытый текстовый файл. Отображение на экране выводимой информации задается параметром "0" команды MON.

При записи в ранее созданный файл старая запись заменяется новой. Если требуется дополнить файл, следует вместо OPEN пользоваться командой APPEND.

READ **имя файла** {,Bb} - чтение файла для последовательного доступа.

READ **имя файла** {,Rr} {,Bb} - чтение файла для прямого доступа.

После выполнения команд OPEN и READ весь ввод с помощью подпрограммы CROUT, команд INPUT или GET, осуществляемый в программах пользователя, производится из открытого текстового файла. Отображение данных на экране задается состоянием "I" команды MON.

EXEC **имя файла** {,Rr} {,Ss} {,Dd} {,Vv} - исполнение команд из текстового файла.

После выполнения команды весь ввод с помощью программы "Системный монитор" осуществляется из текстового файла до тех пор, пока в нем не встретится команда CLOSE без предшествующей ей команды OPEN, либо до исчерпания файла. Вводимые команды диалога различных подсистем выполняются по мере поступления, ввод запускаемых при этом программ также производится из исполняемого файла (кроме случая, когда запущенная программа читает данные из другого файла).

Команда EXEC может использоваться в прямом диалоге ДОС.

POSITION **имя файла** {,Rr}- позиционирование файла, пропускает r записей в файле (записей заданной длины для прямого доступа). Выключает режим чтения и записи.

APPEND **имя файла** {,Ss} {,Dd} {,Vv} - наращивание файла с последовательным доступом, открывает файл и позиционирует его на последний символ. Последующий WRITE будет наращивать файл. Защита по записи на открытый командой APPEND файл не распространяется.

ОБРАЩЕНИЕ ЗА ИНФОРМАЦИЕЙ ПО ФИЗИЧЕСКИМ АДРЕСАМ

Для программных обращений предоставляется доступ к ГМД по физическим адресам. ГМД размечается на 35 дорожек - треков, на каждом из которых размещается 16 секторов. Нумерация треков ведется с 0 от внешней окружности к центральному отверстию.

В стандартном варианте треки 0, 1, 2 заняты программами ДОС, на треке 17 хранится каталог ГМД. Остальные треки предоставляются пользователю.

Для обращения к подпрограмме чтения - записи трека и сектора (ЧЗТС) необходимо иметь в памяти два массива, форматы которых представлены в табл. 5.1 и 5.2. Адрес табл. 5.1 помещается в регистры A (старший байт) и Y (младший байт), после чего управление передается по адресу o 3D9. Табл. 5.1 размещается по адресу ↑ B7E8, табл. 5.2 - по адресу ↑ B7F8.

Таблица 5.1

Обращение к ГМД по физическим адресам

Номер байта	Назначение байта	Значение байта (шестнадцатеричное)
1	Тип таблицы	01
2	Номер разъема, умноженный на ↑ 10 (16)	00 - FF
3	Номер привода	00 - 01
4	Номер тома ГМД	01- FF (При 00 - номер тома игнорируется)
5	Номер дорожки (трека)	00 - 22
6	Номер сектора	00 - 0F
7-8	Адрес табл. 5.2	0000 - FFFF

Продолжение табл.5.1.

Номер байта	Назначение байта	Значение байта (шестнадцатеричное)
9-10	Адрес буфера данных	0000 - FFFF
11-12	Не используется	-
13	Код режима работы	0 - включение НГМД, подвод головки; 1 - чтение 256 байт в буфер данных; 2 - запись 256 байт из буфера данных; 3 - разметка ГМД.
14	Код завершения работы	0 - нормальное завершение; 10 - ГМД защищен по записи; 20 - не тот номер тома; 40 - ГМД неисправен; 80 - сбой при чтении.
15	Номер тома ГМД в предыдущем обращении	00 - FF
16	Номер разъема в предыдущем обращении, умноженный на ⌘ 10	00 - FF
17	Номер привода в предыдущем обращении	00 - FF

Таблица 5.2

Обращение к НГМД по физическим адресам

Номер байта	Назначение байта	Значение байта (шестнадцатеричное)
1	Тип устройства	00 - FF (для ЕС-5088 - 00)
2	Число обращений на трек	00 - FF (0 - нет обращений 1 - одно обращение и т.д.)
3-4	Код формата записи	0000 -FFFF (для MFM/FM - EFD8)

Пример обращения к ЧЗТС приведен на рис. 5.1.

Эта программа осуществляет чтение сектора 7 дорожки (трека) 9 (без учета номера тома) в буфер по адресу БУФЕР; чтение производится с первого накопителя, подключенного к контроллеру НГМД, установленному в разъем X4 (S-3).

```
]LIST
10  REM ОБРАЩЕНИЕ К ЧЗТС
20  * $1000:
30     ! LDA# < ТАБЛ1
40     ! LDA# > ТАБЛ1
50     ! JSR$309
60     ! LDATАБЛ1 + 13
70     ! STA ANSER
80     ! RTS
90     ! ТАБЛ1:$01300100
100    ! ;ТИПТАБЛИЦЫ,РАЗЬЕМ

110    ! ;ПРИВОД,ТОМ
120    ! $0907;ТРЕК,СЕКТОР
130    !  = ТАБЛ2
140    !  = БУФЕР
150    ! #2;НЕИСПОЛЬЗУЕТСЯ
160    ! $01
170    ! #4;ОСТАЛОСЬОТ
180    ! ;ПРЕДЫДУЩЕГООБРАЩЕ
       НИЯ
190    ! ТАБЛ2:$0001E
200    ! :
210  CALL $1000
220  IF  PEEK ( ANSER) <  >
     0 THEN  PRINT "ОШИБКА
     ": STOP
```

Рис. 5.1. Программа физического обращения к диску

СООБЩЕНИЕ ДОС

Всего ДОС распознает 15 ошибок различного рода, которые распечатываются в развернутом виде. При использовании команды **ONERGOTO** на языке БЕЙСИК код ошибки записывается по адресу 222 (Д DE) и имеет следующий смысл:

01 - язык недоступен системе (нет интерпретатора с языка БЕЙСИК);
02, 03 - ошибка в диапазоне;
04 - ГМД защищен на запись;
05 - чтение после конца данных;
06 - файл не найден;
07 - не тот том (неправильно указан параметр V);
08 - ошибка ввода-вывода;
09 - ГМД полностью использован и дальнейшая запись на него невозможна;
0A - файл защищен на запись;
0B - синтаксическая ошибка;
0C - нет свободных буферов ввода-вывода;
0D - команда не для этого типа файла;
0E - нет конца программы;
0F - команда недоступна в прямом диалоге.

5.3. ИНТЕРПРЕТАТОР ЯЗЫКА БЕЙСИК

Двоичный файл с именем **HELLO**, содержащий интерпретатор языка БЕЙСИК, размещается на системном ГМД [13, 14, 16, 19].

Системный диск формируется таким образом, чтобы при включении питания

происходила автоматическая загрузка интерпретатора в память компьютера (ячейки с адресами E000 + FFFF).

БЕЙСИК занимает 14К байт памяти. Основное отличие интерпретатора от трансляторов (например, с языка ФОРТРАН) заключается в непосредственном переводе командной строки в двоичные коды, которые немедленно выполняются. Таким образом, интерпретатор языка БЕЙСИК позволяет организовать диалог как в программном, так и в непосредственном режиме. Интерпретатор воспринимает командную строку, содержащую операторы на языке БЕЙСИК. В одной строке может поместиться не более 255 символов (что соответствует восьми экранным строкам).

ЭЛЕМЕНТЫ ЯЗЫКА

В отличие от программ в машинных командах, где данные представлялись в двоичном виде, в программе на языке БЕЙСИК (в дальнейшем просто программа на БЕЙСИКе) данные представляются в форме, удобной для пользователя: целые, вещественные, алфавитноцифровые.

Данные на языке БЕЙСИК записываются либо как константы, либо как переменные.

Константа - величина, записанная в виде цифр (число) или символов (строка). Например, целые константы представляются в десятичном виде (диапазон ±32767) или в шестнадцатеричном формате (от ¤ 0 - ¤ FFFF). Числам от ¤ 0 до ¤ 7FFF соответствуют положительные целые величины, от ¤ 8001 до ¤ FFFF - целые отрицательные (от -32767 до -1). Заметим, что поскольку шестнадцатеричному числу ¤ 8000 соответствует целое число 32768 (выходит за диапазон целых констант), то вблизи ¤ 8000 лучше пользоваться вещественными константами.

Примеры целых констант: 29, -100, ¤ 7F,FFFF.

Вещественные константы представляются в естественном виде:

123.456 , -. 001,3.

или в экспоненциальном формате:

1.23456E2 = 123.456 , -I.E-3 = 001, 3.EO.

Вещественные константы изменяются в диапазоне ±9.99999999 или $\pm 10^{\pm 37}$.

Строковые или символьные константы имеют длину не более 237 символов, содержат любые символы, кроме перевода строки (↵), и заключены в апострофы.

Например:

"ПЭВМ АГАТ", "ПРОГРАММА".

Переменная - это такая величина, которая может принимать произвольные значения в зависимости от потребностей пользователя.

В качестве обозначений переменных используют буквенно-цифровые символы. Обозначение переменной служит ее именем и не является количественной характеристикой. В качестве имени переменной можно использовать последовательность из любого числа символов, но первым символом должна быть буква.

Так же, как и константы, переменные могут быть трех типов. Признаком типа служит специальный символ, стоящий после последнего символа имени (% - признак целого типа, ¤ - признак строкового, у вещественных переменных специального символа нет).

Пример.

X - переменная вещественного типа;

X% - переменная целого типа;

X $ - переменная строкового типа;

AX12NAME79ПЮЧ - переменная вещественного типа.

В примере символами X, X%, X $ обозначаются разные переменные (имена совпадают, но тип разный).

Приведенные выше переменные называют еще простыми, так как в памяти им соответствуют одиночные константы соответствующего типа. Так, вещественные числа могут быть присвоены вещественным и целым переменным. В целой переменной запоминается целая часть значения, которое должно принадлежать диапазону представления целых констант.

Строковые константы присваиваются только строковым переменным.

Кроме простых переменных интерпретатор языка БЕЙСИК обрабатывает индексированные переменные или массивы. Число индексов в массиве определяет его размерность и может быть произвольным. Таким образом, массив характеризуется именем, указателем типа, количеством и максимальным значением индексов.

Пример.

M(10,2) - массив вещественных чисел;

M%(2,2) - массив целых чисел;

M $(5,2) - массив строковых символов.

В отличие от простой переменной, в массиве можно запомнить много величин. Например, в массиве M(10,2) можно запомнить 20 вещественных или целых (а возможно, и тех, и других) чисел. Обратиться к каждому числу (элементы массива) можно по имени, за которым в скобках следует указать значения индексов, соответствующих этому элементу. Три массива, приведенные в примере, идентифицируют три матрицы разного типа и размерности. Первый массив - матрица из 10 строк и двух столбцов, содержащая вещественные числа, второй - матрица 2 × 2 целых чисел, третий - матрица 5 × 2 строковых величин.

ОПЕРАЦИИ, ВЫПОЛНЯЕМЫЕ ИНТЕРПРЕТАТОРОМ ЯЗЫКА БЕЙСИК

Арифметические операции:

"=" - присваивание (переменная=выражение);

"-" - взятие с обратным знаком (унарная), вычитание (бинарная);

"+" - сложение;

"*" - умножение;

"/" - деление;

"^" - возведение в степень;

Операции сравнения и логические:

"=" - равно;

"< >" - не равно;

"<" - меньше;

">" - больше;

"<=" - меньше или равно;

">=" - больше или равно;

"NOT" - логическое НЕ;

"AND" - логическое И;

"OR" - логическое ИЛИ;

Логическое значение "истины" тождественно арифметической "1", логическое значение "ложь" - "0". Для строковых применимы отношения "=" и "< >".

КОМАНДЫ ИНТЕРПРЕТАТОРА С ЯЗЫКА БЕЙСИК

Команды интерпретатора (или директивы) подразделяются на четыре типа:

A - команды, начинающиеся с ключевого слова;

B - присваивания, начинающиеся с имени переменной;

C - команды ассемблера, начинающиеся со знака "!";

Д - команды отладочного набора, начинающиеся со знака "*".

Для отличия команд типа A от команд типа B имена не должны начинаться с фрагментов, тождественных ключевым словам. Кроме того, имена, употребляемые в качестве меток текста на языке ассемблер, не могут начинаться с фрагментов, совпадающих с мнемоникой операций системы команд.

В командной строке интерпретатора могут содержаться команды типа A и B, разделенные знаком ":", который отделяет от них также команды типа C. Между собой команды типа C разделяются знаком "!". При записи команд типа Д после других команд разделителем также является знак ":". После команд типа Д знак ":" может быть принят за элемент команды, поэтому располагать его в середине строк не рекомендуется.

Внутри команды слова при необходимости разделяются пробелами.

В непосредственном режиме командная строка начинается с команды, которой могут предшествовать пробелы. В программном режиме команде должно предшествовать целое десятичное число в диапазоне от 0 до 65535. Пронумерованные командные строки запоминаются интерпретатором языка БЕЙСИК начиная с 2000 адреса. Причем интерпретатор автоматически упорядочивает строки в порядке возрастания их номеров.

В непосредственном режиме командная строка нигде не запоминается, а сразу же после нажатия клавиши ↵ выполняется.

Чтобы выполнить командные строки в программном режиме, необходимо запустить их (программу или часть программы) на выполнение специальной командой.

Команда присваивания. Это основная команда интерпретатора языка БЕЙСИК. Кодом команды служит знак "=".

Примеры.

```
1) X = 5.2+3
```

Переменной X присваивается результат сложения двух констант 5.2 и 3.

```
2) X=X+A*4 + 10.2
```

Переменной X присваивается значение суммы старого содержимого этой же переменной X плюс произведение константы 4 и содержимого переменной A плюс константа 10.2. После выполнения этой команды старое содержимое X заменится на новое.

Команды ввода-вывода. Эти команды позволяют организовать обмен информацией в программах.

INPUT LIST (или INPUT "КОНСТАНТА"; LIST) - команда ввода данных. LIST - переменные, разделенные запятой, которым присваиваются значения вводимых с клавиатуры констант. Приглашением к вводу констант служит знак "?" (в первом варианте команды) или КОНСТАНТА (во втором варианте), после чего на экране с помощью клавиатуры нужно набрать столько констант (разделив их запятыми или пробелами), сколько переменных указано в команде (вместо LIST). Нужно помнить о соответствии типов констант типам соответствующих переменных.

GET A$ - ввод одного символа с клавиатуры. Все символы равноправны. На экране введенный символ не отображается.

PRINT LIST - вывод на экран ВКУ значений элементов списка. Если в списке указаны константы, то на экран выводятся эти константы. Разделителем элементов в списке может быть знак ";" или ",". Разделитель ";" задает печать символа вплотную за предыдущим элементом, разделитель "," - в очередную треть экрана. Наличие ";" в конце списка элементов задает для следующего оператора PRINT печатать в ту же строку; при отсутствии этого знака каждый следующий оператор осуществляет вывод с новой строки. Слово PRINT можно заменить символом "?".

Используя оператор PRINT и оператор присваивания, можно пользоваться интерпретатором в качестве калькулятора в режиме непосредственного исполнения сложных вычислений.

DATA - определяет блок данных (констант)

READ X - присваивает X значение очередного элемента списка данных, определенного оператором DATA, например, DATA 1, 5E-7, СТРОКА, "ТЕКСТ, ТЕКСТ" - определяет блок данных из четырех констант. Строки без кавычек в списке не должны содержать запятых.

RESTORE - устанавливает в начальное положение указатель в блоке данных. Следующим оператором READ будет прочитан первый элемент списка блока данных.

Команды и функции, работающие с массивами и строками. К этому типу команд и функций относятся следующие:

DIMA(X, Y, Z) - описывает массив [трехмерный массив A с диапазоном индексов 0-X, 0-Y, 0-Z; под память заняты (X+1)*(Y+1)*(Z+1) вещественных элементов]. Число размерностей ограничивается только общей емкостью памяти;

LEN (A$) - вычисляет число символов в строке-аргументе A$;

STR$ (x) - переводит представление целого или вещественного аргумента в текстовую строку;

VAL (A$) - переводит текстовую строку A$ в вещественное число (сначала и до первого нечислового символа в строке (A$);

CHR$ (X) - переводит в символ число X, заданное в коде КОИ-8;

ASC(A$) - код КОИ-8 первого символа строки-аргумента A$;

LEFT$ (A$, X) - строка из первых X символов аргумента A$;

RIGHT$ (A$, X) - строка из последних X символов аргумента A$;

MID$ (A$, X, Y) - строка из Y символов A$, начиная с X-го.

В качестве примера на рис. 5.2 приведена небольшая программа, с помощью которой можно сортировать массив слов по первому символу латинского алфавита.

Рис. 5.2. Программа сортировки слов

```
100   REM  СОРТИРОВКА МАССИВА
      СЛОВ
110   REM  ПО ПЕРВОМУ СИМВОЛУ
120   REM  ЛАТИНСКОГО АЛФАВИТА
130   FOR  I=2 TO M
150   FOR  J=I-1 TO 1 STEP-1
180   IF ASC(LEFT¤(M¤(I),1))<
      ASC(LEFT¤(M¤(J),1)) THEN
A¤=M¤(I):M¤(I)=M¤(J):M¤(J)=A¤
300   NEXT:NEXT
310   REM M¤ -ИМЯ МАССИВА
```

Пример сложения строк

B$ = "АГАТ": A$ = "БЕЙСИК": A$ = = A$ + "-" B$

В результате строковая переменная A$ содержит строку "БЕЙСИК - АГАТ".

Если теперь набрать директиву PRINT RIGHT$ (A$, 4), то на экране ВКУ будет напечатано: АГАТ. Директива PRINT MID$ (A$, 6,3) приведет к тому, что на экране ВКУ будет напечатано К-А. После выполнения команды A = LEN(A o > переменной A присвоено значение 11.

Следующая программа выводит на экран ВКУ латинский алфавит:

```
10FORI=1TO26
20PRINTI;".";CHR$ (I+ 40)
30NEXT
```

При работе с массивами необходимо помнить о том, что не объявленные в операторе DIM массивы автоматически получают размерность всех индексов от 0 до 10. Объявление в операторе DIM массива должно предшествовать его использованию в программе.

Идентификация идентификаторов осуществляется так, что переменные A,A%, A$,A(I),A%(I),A$ (J) для интерпретатора являются различными, а появление в программе массива A(1,2) и массива A(3,4,5) ведет к ошибке и останову интерпретации.

Команды редактирования. К ним относятся:

LIST - выдача текста программы:

LIST - выдает весь текст;

LIST 100,200 - выдает строки с номерами 100 и 200;

LIST 100, - выдает строки с номерами не меньше 100;

LIST, 200 - выдает строки с номерами не больше 200;

LIST 150 - выдает строку 150.

DEL X, Y - удаляет строки программы от номера X до номера Y включительно;

REM - комментарий до конца строки, при выполнении игнорируется;

VTAB Y - устанавливает курсор на строку экрана с номером Y (O ≤ Y ≤ 31);

HTAB X - перемещает курсор на X-ю позицию текущей экранной строки (нумерация строк и позиций на экране сверху вниз и слева направо);

TAB (X) - применяется в операторе PRINT, перемещает курсор на X-ю позицию;

POS (O) - вычисляет номер позиции от левого края текстового окна. В скобках значение несущественно;

SPC X - выводятся X пробелов (элемент оператора PRINT);

HOME - очищается экран, курсор помещается в верхний левый угол экрана;

CLEAR - переменным присваивается нулевое значение.

В конце текста программы во внутреннем представлении хранятся все имена

переменных, длина которых составляет два и более символа, набранных на клавиатуре в процессе отладки. Среди них могут находиться имена, исключенные из программы, и некоторые ошибочные директивы, которые не влияют на ход исполнения, но "засоряют" память и приводят к неоправданному увеличению длины программы. Директива CLEAR позволяет исключить все лишние имена из внутреннего представления программы. При ее выполнении, кроме того, на экран выдается весь текст программы в формате поиска-замены.

FRE(0) - вычисляет емкость оставшейся памяти и выполняет чистку "мусора" в области хранения строк программы (значение аргумента несущественно);

FLASH - включает мерцающий режим вывода текста;

INVERSE - включает инверсный (черный по светлому фону) режим вывода текста;

NORMAL - включает прямой (светлый по черному) режим вывода текста. Последние три команды в алфавитно-цифровом режиме АЦР-64 не исполняются.

SPEED = X - задает скорость вывода текста на экран, X изменяется от 0 до 255.

Системные команды. К этому типу команд относятся:

LOAD - загружает программу с магнитной ленты (МЛ) (бытовой магнитофон);

SAVE - записывает программы на МЛ;

NEW - устанавливает начальное состояние памяти, стирает программу и данные;

RUN - запускает программы со строки с наименьшим номером; для запуска с промежуточной точки можно указать номер строки (например, RUN200);

STOP - останавливает программу с выдачей текста строки, в которой произошел останов;

END - завершает программу;

<УПР-Ц> - останавливает программу или выдачу текста программы с выдачей текста строки и подсветкой оператора, на котором произошел останов;

CONT - продолжает выполнение программы после STOP или <УПР-Ц>;

TRACE - включает режим вывода номера строки при выполнении каждого оператора;

NOTRACE - отменяет TRACE;

PEEK(X) - вычисляет значение байта по адресу X;

POKE X, Y - заносит Y (0<=Y<=255) по адресу X;

WAIT X, Y, Z - ожидает, пока содержимое байта по адресу X (<X>) будет равно. Например, WAIT $ C000, $ 80,0 ожидает любого нажатия на клавиатуре. CALL X - выполняет вызов кодовой подпрограммы по адресу X;

USR(X) - осуществляет вызов кодовой подпрограммы с передачей значения. Значение X помещается в ячейках $ 9D - $ A2(157 - 164). В ячейках 11-12 находится адрес кодовой подпрограммы в ячейке 10 - код 76 ($ 4C);

HIMEM: - устанавливает верхнюю границу памяти под переменные и строки, используемые программой. Значения переменных теряются. Установленное значение HIMEM увеличивать нельзя;

LOMEM - то же для нижней границы памяти (нельзя уменьшать). Исходные значения HIMEM ($ 9600 с ДОС, $ C000 без ДОС) и LOMEM ($ 808) устанавливаются директивой ДОС "FP" или CALL $ E000.

Команды передачи управления. К этим командам относятся:

GOTO 360 - переход на строку 360;

GOSUB 1100 - переход на строку 1100 с запоминанием адреса возврата;

RETURN - возврат к оператору, следующему за GOSUB;

POP - удаление последнего адреса возврата из списка;

ON X GOTO 100, 200, 300, 400 - переключатель-переход, осуществляет переход на строку с X-м номером из списка;

ON X GOSUB... - переключатель-вызов, вызывает X-ю из перечисленных подпрограмм;

DEF FN SEC(x)=I/SIN(X) - определяет функцию SEC; можно пользоваться оператором Y=1 - FNSEC(.I*A). Число аргументов при описании функций не более одного;

IF X=1 THEN PRINT Y - выполняет последовательность операторов после THEN и до конца строки (только при истинности логического выражения IF; в противном случае выполнение продолжается со следующей строки);

FOR I=1 TO 15 STEP 4...NEXT ЦИКЛ - выполняет операторы между оператором FOR и соответствующим ему NEXT с I=1, затем с I=5, I=9 и т.д., пока не будет I > 15, тогда выполняются операторы,следующие за NEXT; если STEP опустить, будет использован шаг 1; хотя бы одно выполнение операторов цикла происходит всегда;

NEXT - конец цикла; после выхода из цикла его переменная получает первое из значений, не удовлетворяющее условию;

ONERR DOTO 1500 - подключение программ обработки ошибок, диагностируемых интерпретатором и ДОС. Начальный номер строки обработчика 1500. На экран результат диагностики не выдается;

RESUME - используется в программе обработки ошибок. Осуществляет возврат на оператор, при выполнении которого обнаружена ошибка.

При использовании операторов GOTO, GOSUB необходимо помнить, что переход на несуществующую строку по номеру приводит к ошибке и останову интерпретации.

При использовании оператора IF, если последовательность операторов, которые необходимо выполнить при истинности логического выражения, не умещается в одной строке, можно воспользоваться следующим приемом:

IF NOT <логическое выражение> THEN K,

где K - номер строки, следующей за строками, содержащими все операторы, которые необходимо выполнить в случае истинности логического выражения. Операнд K можно заменить на один или несколько операторов.

Пример использования оператора цикла, определяющий, существует ли элемент массива A(10) со значением 5:

```
10 FOR I=0 TO 10
20 IF A(I)=5 THEN I=10:NEXT:GOTO 40
30 NEXT:PRINT "НЕ";
40 PRINT "НАЙДЕМ!"
50 END
```

Наличие оператора END в программе не обязательно!

В операторах-переключателях, если значение переключателя больше числа

номеров строк в списке или равно нулю, переход осуществится на следующий оператор. Если значение переключателя больше 255 или меньше 0, произойдет останов интерпретации и будет выдано сообщение об ошибке с текстом строки, в которой она обнаружена.

Команды графики и игровых датчиков. К этому типу команд относятся:

GR = N - включение графики низкого разрешения (64×64);

здесь N - номер страницы размером 2К байт, при этом 2 ≤ N ≤ 31.

MGR = N - включение графики среднего разрешения (128×128); N - номер страницы размером 8К байт, при этом 1 ≤ N 7.

HGR = N - включение графики высокого разрешения. Параметры такие же, как у оператора MGR. Команды включения графических режимов очищат включенную страницу;

COLOR=X - устанавливает цвет в последующих операторах вывода графической информации; X изменяется от 0 до 7;

PLOT X,Y - помещает точку текущего цвета в X-ю позицию Y-й строки экрана;

PLOT X0, YOTOX1,Y1 - рисует прямую линию текущего цвета из точки X0, Y0 в точку X1, Y1. Если координаты начальной точки не указаны, используется конечная точка последнего по выполнению оператора PLOT;

SCRN (X, Y) - указывает номер цвета точки X, Y;

DRAW I AT X, Y - воспроизводит I-й образ в точке X, Y текущим цветом;

XGRAW J AT X, Y - воспроизводит J-й образ, но цветом, дополнительным к цвету экрана;

SCALE=X - устанавливает масштаб воспроизведения образов (1-255);

ROT=X - устанавливает угол поворота образов по часовой стрелке.

При новом режиме всегда очищается экран. Диапазон допустимых значений координат для GR составляет 0 - 63, для MGR - 0 - 127, для HGR - 0 - 255.

TEXT=N - устанавливает текстовый режим; N - номер текстовой страницы. Номера 0 - 31 соответствуют страницам размера 32×32 символа, режима АЦР-32; номера от 32 до 63 - страницам размера 64×32 символа режима АЦР-64. Страницами 0 и 32 пользоваться нельзя, они находятся на адресах рабочих полей интерпретатора.

Те же замечания относятся и к номерам страниц графики низкого разрешения. Для режимов среднего и высокого разрешения страницы нумеруются от 0 до 3.

PDL(X) - выдает значение, установленное ручкой X-го аналого-цифрового пульта.

МАТЕМАТИЧЕСКИЕ ФУНКЦИИ

В состав языка БЕЙСИК включены математические функции. Наличие функций позволяет производить сложные математические расчеты. Эти функции вычисляют:

SIN(X) - значение синуса (X рад);

COS(X) - значение косинуса (X рад);

TAN(X) - значение тангенса (X рад);

ATN(X) - значение арктангенса (X рад);

INT(X) - значение целой части X;

RND(X) - псевдослучайное значение в интервале (0,1).

При X>0 значение функции RND(X) является элементом случайной последо-

вательности. При X<0 происходит запуск псевдослучайной последовательности (своей для каждого X), при X=0 - повторение последнего результата;

SGN (X) - 1 при X<0; 0 при X=0; 1 при X>0;

ABS (X) - модуль X;

SQR (X) - положительный квадратный корень X;

EXP (X) - экспонента X;

LOG (X) - натуральный логарифм X.

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ АССЕМБЛЕРА

Управление ассемблированием (работой ассемблера). Фрагмент программы

* {СМЕЩЕНИЕ, } АДРЕС:

используется для входа в режим ассемблирования.

АДРЕС - выражение, задающее начальный адрес записи формируемого кода при ассемблировании.

СМЕЩЕНИЕ - выражение, задающее начальное значение адреса, используемого для ссылок на формируемый код (адрес программы при исполнении).

Например, в результате выполнения

* ¤ 1234, ¤ 1100:! МЕТКА: = МЕТКА!:

по адресу ¤ 1100 будет записано слово ¤ 3412.

Если указано одно выражение, адрес и смещение совпадают.

! - сигнал входа в ассемблер и разделитель команд ассемблирования внутри строки; он должен предшествовать каждой команде ассемблера;

!: - пустая команда ассемблера, служит сигналом перехода на второй проход или сигналом конца ассемблирования.

Ассемблирование проводится в два прохода. На первом проходе определяются форматы операндов и значения меток, форматы команд записываются на место объектного кода. Просмотр текста происходит в порядке исполнения строк интерпретатора языка БЕЙСИК, содержащих команды ассемблера. При исполнении на первом проходе пустого оператора управление передается (аналогично GOTO) первому встреченному после входа в режим ассемблирования оператору со знаком "!".

Программа должна обеспечивать второй проход операторов ассемблирования в том же порядке, что и на первом проходе (все операторы интерпретатора языка БЕЙСИК при этом исполняются второй раз). На втором проходе формируется двоичный код, значения меток сохраняются и могут быть в дальнейшем использованы в качестве операндов CALL и USR для исполнения участков кода, а также отладочными директивами для просмотра и модификации кодовых участков.

Команды ассемблера могут быть помечены. Метками служат целые и вещественные переменные языка БЕЙСИК, например:

```
5 KOH = FDDF
10 ! DMI:M2% : LDA  ¤ 55! STA  ¤ 666
20 ! JSR KON RTS
30 !:
```

Команды ассемблера подразделяются на программные команды и константы.

Константы ассемблера. Имеется восемь констант ассемблера:

1)! (выражение) - загружает пять байтов вещественным значением выражения, вычисленным на втором проходе, в представлении интерпретатора;

2) < (выражение) - загружает один байт старшим байтом целого значения выражения;

3) > (выражение) - то же, но загружает младший байт;

4) = (выражение) - загружает два байта целым значением в порядке "младший, старший", т.е. в формате адреса системы команд процессора;

5) @ (выражение) - то же, что и константа =, но в порядке "старший, младший", т.е. в формате данных интерпретатора;

6) ¤ {hex, hex} (четное число шестнадцатеричных цифр) - шестнадцатеричная загрузка области;

7) ! ("строка длиной") - загружает байт кодами КОИ-7 символов строки (старший бит "0");

8) # (выражение) - увеличивает значение счетчиков адреса и смещения на значение выражения.

На рис. 5.3 приведен пример, иллюстрирующий применение констант.

Программные команды ассемблера. Первое поле программной команды - мнемокод операции, затем может следовать операнд.

Мнемокоды и виды адресации записываются так, же как описано п. 5.1.

В качестве содержимого операнда могут применяться константы типа 2) - 4), а в непосредственных операндах - также и типа 7) длиной в один символ.

При использовании в качестве операндов констант типа < и > формат адресации однобайтный, типа = - двубайтный.

В случае отсутствия перед операндом знаков >, <,= формат адресации определяется следующим образом. Если в выражении имеется идентификатор, значение которого не определено на первом проходе к данному моменту, то формат адресации двубайтный. В противном случае формат адресации определяется значением выражения: при значении выражения, меньшем 256, формат адресации однобайтный; в остальных случаях - двубайтный.

Предусмотрены дополнительные возможности введения комментариев в любой команде от символа ";" до конца команды или конца строки. Текст комментария при выполнении игнорируется.

Пример использования комментариев приведен на рис. 5.4.

Средства отладки. Здесь и в дальнейшем под адресом ячейки памяти будем понимать значение выражения. К средствам отладки относятся команды:

* (выражение) - осмотр содержимого ячейки памяти по заданному адресу; например:

```
* ¤ 1000
    1000-BB
```

* (выражение 1).(выражение 2) - осмотр интервала памяти по заданным начальному и конечному адресам; например, после директивы: * ¤ 100. ¤ 127 на экране появится следующая информация:

```
0100 - AA AA AA AA AA AA AA AA
0118 - 10 10 10 10 10 10 10 10
0120 - 1D 1D 1D 1D 1D 1D 1D 1D
```

* (выражение) # (выражение 2)·(выражение 3) M - передвижение интервала памяти;

* (выражение 1) # (выражение 2)·(выражение 3) V - сравнение двух интервалов памяти;

```
]LIST

5  H1 = $100: LNGTH = 1: M2
     % =  M1 -  LNGTH:X =
     1.57
10  * $1000:
     ! ! SIN (X) +  COS (
     X)
     !  =  M1 * 12
20   !  >  M1 + 16 *  LNGTH
     +  M2% + 100
30   ! $AAB6F7FD0001
40   !  OM1 + 256 + 80
50   !  = $FDF4
     ! $81
60   ! "ТАБЛИЦА"
     ! K:
70  REM ПОСЛЕ RUN СФОРМИРУ
     ЕТСЯ КОД
80  REM 1000 -81 00 1A 15
     68 00 0C 45
90  REM 1008 -AA B6 F7 FD
     00 01 02 50
91  REM 1010 -F4 FD 81 74
     61 62 0C 69
92  REM 1018 -63 61
93  REM ГДЕ :
94  REM 1000-1004 -ПРЕДСТА
     ВЛЕНИЕ ВЕЩЕСТВЕННОГО
     1.00079601
95  REM 1005-1006 АДРЕС $C
     00(12*256)
96  REM 1007 -МЛАДШИЙ БАЙТ
      ЗНАЧЕНИЯ $100+$F0+$F
     1+$64=$345
97  REM 1008-100D -ПОСЛЕДО
     ВАТЕЛЬНОСТЬ БАЙТ AAB6
     F7FD0001
98  REM 100E-100F -ЦЕЛОЕ П
     РЕДСТАВЛЕНИЕ 592=$250

100  REM 1010-1011 -АДРЕС
     $FDF4
110  REM 1012       -БАЙТ $
     81
120  REM 1013-1019 -ТЕКСТ
     ТАБЛИЦА К ПОЛУЧИЛА ЗН
     АЧЕНИЕ $101A=4122
```

Рис. 5.3. Использование констант

```
]LIST

5  GOTO 65
10  REM МАКРООПРЕДЕЛЕНИЕ
20    ! PLA
      ! STA10
      ! PLA
      ! STA1
30  RETURN
40  REM МАКРООПРЕДЕЛЕНИЕ Д
     ЛЯ ВОЗВРАТА
50    ! LDA1
      ! PHA
      ! LDA0
      ! PHA
      ! RTS
60  RETURN
65  * 4096:
      !  BEGIN:NOP;ТРЕБУЕТ
     СЯ ДЛЯ ПРАВИЛЬНОГО НА
     ЧАЛА ВТОРОГО ПРОХОДА
70  REM ПОДПРОГРАММА ПЕЧАТ
     И И ВЫБРАСЫВАНИЯ 3-Х
     БАЙТ ИЗ СТЕКА
80  GOSUB 10
90  FOR I = 1 TO 3
100   ! PLA
      ! JSR$FDF4; ПОДПРОГР
     АММА COUT СИСТЕМНОГО
     МОНИТОРА
110  NEXT : GOSUB 40
130  MJ = 0: REM ИСПОЛЬЗОВ
     АНИЕ МАКРООПРЕДЕЛЕНИЯ
      С МЕТКАМИ
140  SR = 5000: DS = 8000:
      LN = 33;ИСТОЧНИК,ПОЛ
     УЧАТЕЛЬ,ДЛИНАПЕРЕПИСИ

150  GOSUB 1000
160  SR = 5100: DS = 7000:
      LN = 99: GOSUB 1000
170   ! : END
1000  MJ =  MJ + 1: REM НА
     ЧАЛО МАКРООПРЕДЕЛЕНИЯ
      ПЕРЕПИСИ
1010  ! LDX#$0:L( MJ): LDA
      =  SR,X;ПРИКАЖДОМНОВ
     ОММАКРОВЫЗОВЕМЕТКОЙБУ
     ДЕТОЧЕРЕДНОЙЭЛЕМЕНТМА
     ССИВА
1020  ! STA =  DS,X
      ! INX: CPX# >  LN
1030  ! BNE = L( MJ);ПЕРЕХ
     ОД ВНУТРИ МАКРООПРЕДЕ
     ЛЕНИЯ
1040  RETURN

]

]

]
```

Рис. 5.4. Применение комментариев

* { (выражение)}L - вывод текста программы в мнемокоде системы команд, с заданного адреса или, если адрес не указан, продолжение распечатки; выводится 28 строк текста программы;

* (выражение 2)·(выражение 2) T - вывод текста в кодировке КОИ-7, в заданном интервале;

* (выражение C) - выполнение программы в кодах системы команд, с заданного адреса;

*{ (выражение I)}S - выполнение программы в кодах системы команд с адреса, определяемого выражением 1, до прерывания по адресу выражения 2.

Если выражение 1 отсутствует, то начальным адресом считается адрес последнего прерывания.

При пользовании директивой S следует учитывать:

1. Исходное включение памяти для отлаживаемой программы отличается от используемого интерпретатором следующим:

модуль-эмулятор ПЗУ включается в состояние PCC= ¤ A0 (¤ E0 в интерпретаторе);

модуль дополнительного ОЗУ включается в состояние PCC = ¤ 08 (¤ 09 в интерпретаторе).

В адресном пространстве отлаживаемой программы на этот момент имеется копия ДОС, соответствующая моменту начальной загрузки ДОС (адреса дополнительного ОЗУ), и интерпретатор языка БЕЙСИК с включенным основным банком ¤ D000 - DFFF.

Переключения банков памяти, выполняемые отлаживаемой программой, запоминаются (значения PCC хранятся по адресам ¤ E8 - ¤ E9) и повторяются при последующих исполнениях "S".

2. Содержимое ячеек ¤ 0... ¤ 3FF отлаживаемой программы хранится в адресах ¤ 8000... ¤ 83FF и не меняется при его работе. При каждом запуске директивы S с нового адреса происходит обратное перезаписывание из 8000... ¤ 83FF в ¤ 0... ¤ 3FF.

3. Содержимое регистров отлаживаемой программы запоминается в ячейках ¤ 45 - ¤ 49, адрес точки останова - в ячейках ¤ 0- ¤ 1.

4. Содержимое текстового экрана отлаживаемой программы хранится по адресу ¤ 8400... ¤ 8BFF. Для его просмотра можно переписать эту зону памяти на адреса текущего экрана.

Предусмотрено использование директивы S для отладки программ, размещенных на адресах ПЗУ. Такую программу следует загружать в массив 1 эмулятора ПЗУ и во время работы следить за сохранностью интерпретатора языка БЕЙСИК, размещенного в массиве 0 эмулятора ПЗУ.

ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ ЯЗЫКА БЕЙСИК

При разработке программного обеспечения на языке БЕЙСИК могут оказываться полезными следующие сведения:

1. При желании для вывода текста на экран ВКУ можно использовать не весь экран, а текстовое окно, которое следует задать. В ячейке памяти 32 лежит левая граница текстового окна, а в ячейке 33 - ширина (в цветном текстовом режиме 32×32 хранятся удвоенные значения, нечетные границы недопустимы).

Директивами POKE 33,A и POKE 32,B можно задавать горизонтальные размеры текстового окна; при этом A должно находиться в диапазоне 1 - 64, а B - в диапазоне 0 - 63. Другие значения могут привести к непредсказуемым последствиям. В ячейках 34 и 35 лежат соответственно верхняя и нижняя границы текстового окна.

Директивами POKE 34,C и POKE 35,D можно задавать вертикальные размеры текстового окна. Переменные C и D должны находиться соответственно в диапазонах 0 - 31 и 1 - 32; при этом D должно быть больше C.

2. Чтобы очистить текстовое окно от символов, можно кроме команды HOME, использовать директиву PRINT CHR (12).

Выполнение директивы PRINT CHR (30) очистит текстовое окно от символов начиная с позиции курсора и до конца текстового окна.

Директива PRINT CHR (31) очищает строки от символов начиная с позиции курсора и кончая правой границей текстового окна.

Одновременное нажатие клавиш УПР и F или выполнение директивы PRINT CHR (7) вызывает появление звукового сигнала.

Можно использовать и другие подпрограммы "Системного монитора", вызывая их с помощью операторов CALL или USR.

3. Переключать страницы памяти можно непосредственно в программах с помощью команд POKE - 14592+KR+NS*Q, 0.

Примечания:

а) KR - код режима работы: 0 - в режиме GR, 1 - в режиме MGR, 2 - в текстовом режиме, 3 - в режиме HGR;

б) NS - номер страницы памяти. NS = 1 + 63 в режимах GR и текстовом, NS=1 + 5 в режимах MGR, NGR.

Номера текстовых страниц с 1 до 31 соответствуют АЦР-32, с 33 до 63 - АЦР-64;

в) Q=4 в режимах GR и текстовом, Q=16 в режимах MGR и NGR.

4. Интерпретатор языка БЕЙСИК имеет четыре оператора, позволяющие манипулировать графическими объектами в режиме высокоразрешающей графики. Это операторы DRAW, XDRAW, ROT, SCALE. Однако, прежде чем эти операторы могут быть использованы, графический объект должен быть описан посредством описателя формы, который состоит из последовательности закодированных векторов, определяющих форму объекта. Один или несколько таких описателей вместе с указателями составляют таблицу образов. Коды этой таблицы могут быть набраны с клавиатуры и записаны на магнитной ленте или магнитном диске для последующего использования.

Каждый байт описателя формы разделен на три секции, каждая из которых описывает вектор, указывающий направление движения воображаемого "пера", рисующего объект.

По команде DRAW или XDRAW интерпретатор просматривает секцию за секцией описателя. Когда встречается байт, все биты которого нули, просмотр описателя заканчивается.

Структура одного байта описателя формы, состоящего из трех секций A, B, C:

	C	B	A
Номер бита	76	543	210
Обозначения битов	DD	PDD	PDD

Каждая пара битов, обозначенная DD, определяет направление движения "пера", а каждый бит P указывает, чертить или не чертить точку, прежде чем "перу" сместиться в указанном направлении.

Если DD=00, "перо" сместится вверх, DD=01 - вправо, DD=10 - вниз, DD=11 - влево.

P=0 вызывает смещение без черчения, а P=1 - смещение с черчением точки.

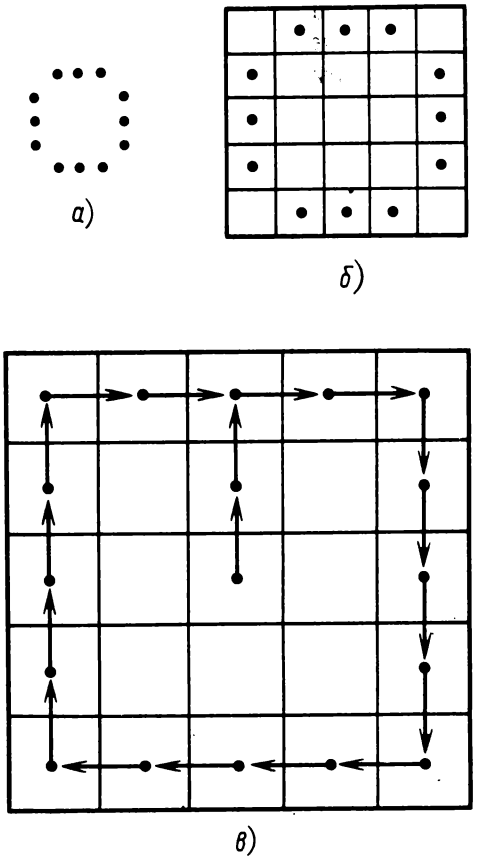
Секция C (два старших бита) не содержит поля P. Считается, что в этой секции P=0, т.е. в ней можно указывать только смещение без черчения.

Таким образом, каждый байт описывает до трех векторов, по одному в секциях A, B и C.

Команды **DRAW** И **XDRAW** обрабатывают секции справа налево (вначале секцию A, затем B, далее C).

Если секции, оставшиеся до конца байта, содержат одни нули, они игнорируются. Например, байт описателя не может заканчиваться секцией C равно 00 (движение вверх без черчения), так как эта секция, содержащая одни нули, будет игнорирована. Аналогично, если секция C содержит 00, секция B не может быть равной 000, так как она тоже будет пропущена. И 000 в секции A прервет описатель, если в секциях B и C нет ни одного бита, равного 1.

Предположим, нужно изобразить объект, показанный на рис. 5.5, *а*. Вначале нанесем его на клетчатую бумагу так, чтобы каждая точка занимала одну клетку (рис. 5.5, *б*). Затем выберем начальную точку. Пусть она находится в центре фигуры. Теперь нарисуем путь воображаемого "пера" через точки нашего объекта, используя только повороты на 90° (рис. 5.5, *в*). Развернем эту последовательность векторов и расположим их в том порядке, в каком "перо" проходит через точки объекта.



Последовательно записываем для каждого вектора его двоичный код в свободную секцию табл. 5.3. Если код не подходит (например, вектор в секции C не может чертить точку) или равен 00 (000) в конце байта, пропускаем эту секцию и переходим к следующей. Преобразуем последовательность нулей и единиц значения вектора описателя в шестнадцатеричный формат (5.3):

12 3F 20 64 2D 15 36 IE 07 00

Эта последовательность байтов теперь должна быть загружена в память компьютера.

Рис. 5.5. Формирование образа

Таблица 5.3

Таблица образов

Номер секции	Вид вектора описателя			Значение вектора описателя			Шестнадцатеричное содержимое секции
	C	B	A	C	B	A	
0		↓	↓		010	010	12
1		←	←		111	111	3F
2		↑	↑		100	000	20
3		↑	↑	01	100	100	64
4		→	→		101	101	2D
5		↓	→		010	101	15
6					110	110	36
7					011	110	1E
8						111	07
9 Конец описания				00	000	000	00

СООБЩЕНИЕ ОБ ОШИБКАХ

Сообщением об окончании исполнения программы и директив языка БЕЙСИК являются сигналы: символ] и курсор в виде мигающей черты.

Интерпретатор сообщает об ошибках в программе или директиве посредством надписей на экране ВКУ либо печати их на печатающем устройстве.

На экране ВКУ выдается, кроме этого, текст строки, при исполнении которой обнаружена ошибка, с выделением местоположения ошибки красным цветом, а при работе в АЦР-64 - знаком "!".

После каждой ошибки происходит останов программы; при этом символ] и курсор в виде мигающей черты вновь появляются на экране ВКУ.

В результате неправильного ввода операторов GET и INPUT в процессе исполнения программы на экране ВКУ появляется сообщение:

ОШИБКА ВВОДА (REENTER)

После этого ввод необходимо повторить.

Возможны следующие сообщения об ошибках:

NEXT БЕЗ FOR - встреченному по ходу исполнения NEXT нет соответствующего FOR;

СИНТ.ОШ. - недопустимый символ в операторе;

RETURN БЕЗ GOSUB - некуда возвращаться в результате, как правило, неверного входа в подпрограмму (GOTO ВМЕСТО GOSUB);

МАЛО ДАННЫХ - выполняется READ, а списки ДАТА исчерпаны (возможно, пропущен RESTORE);

ОШ.ТИП - не согласованы операнды (например, сложение строки с числом);

ПЕРЕПОЛНЕНИЕ - засылка в целую переменную большего числа (шестнадцатеричная константа длиннее четырех цифр и т.д.) ;

МАЛО ПАМЯТИ - нет свободной памяти под переменную или массив (вложенные

GOSUB и FOR не оставили в стеке свободного места под выражение, нет места под буфер поиска замены в директиве LIST и др.);

НЕТ НОМЕРА - отсутствует строка с номером, указанным в GOTO, THEN или GOSUB;

ОШ.ИНДЕКС - значение индекса выходит за пределы размерности, указанной для массива в DIM;

УЖЕ ОПИСАН - при исполнении DIM обнаружен массив с таким же именем; может быть вызвана использованием имени массива раньше DIM;

1/0 - делить на ноль нельзя;

ОШ.ЗНАЧЕНИЕ - неправильный операнд (например, попытка записать с помощью POKE число, большее 255, указать координату за пределами экрана и др.);

СТРОКА ДЛИННА - в результате сложения получена строка длиннее 255 символов;

ОСТАНОВ - одновременно нажаты клавиши УПР, Ц или выполнен STOP;

НАДО RUN - после одновременного нажатия клавиш УПР, Ц или STOP И перед COUT выполнено редактирование программы или нажаты клавиши УПР, СБР - продолжить нельзя;

НЕ В ДИАЛОГЕ - команда только для программного исполнения;

НЕТ МЕТКИ - в выражении ассемблера использовано имя с неопределенным значением, выдается также при использовании отсутствующей функции;

БАЙТ НЕПОЛОН - в шестнадцатеричной загрузке памяти записано нечетное число цифр;

Таблица 5.4

Сообщение об ошибках интерпретатора языка БЕЙСИК

Русская диагностика		Английская диагностика	
Код ошибки	Сообщение	Код ошибки	Сообщение
0	NEXT FOR	0	NO FOR ERROR
12	СИНТ.ОШ	6	SINTAX ERROR
19	RETURN GOSUB	12	NO GOSUB ERROR
35	МАЛО ДАННЫХ	20	NO DATD ERROR
46	ОШ.ЗНАЧЕНИЕ	29	ILLEGAL VALUE
57	ПЕРЕПОЛНЕНИЕ	40	OVVERFLOW ERROR
69	МАЛО ПАМЯТИ	48	OUT OF MEMORY
80	НЕТ НОМЕРА	61	UNDEF STATEMENT
90	ОШ.ИНДЕКС	76	SUBSCRIPT ERROR
99	УЖЕ ОПИСАН	85	REDIM ARRAY
109	X/0	96	DEVISION BY ZERO
124	ОШ.ТИП	126	TYPE ERROR NAME
130	СТРОКА ДЛИННА	130	LONG STRING
165	НЕТ МЕТКИ	157	UNDEF NAME
174	БАЙТ НЕПОЛОН	167	BITE UNCOMPL
186	ОШ. МЕТКА	179	LABEL ERROR
194	ОШ.КОД	184	OPCODE ERROR
200	УЖЕ ЕСТЬ	190	DOOBLE DEF

ОШ.МЕТКА - началю оператора ассемблирования принято за метку, но меткой служить не может, например ! МЕТКА + I:RTS; может быть вызвана неправильной записью кода операции (!TXW);

ОШ.КОД - не удается сформировать команду языка ассемблера, например, недопустимая адресация; выдается также при нарушении условий относительной адресации, когда метка перехода далеко отстоит от команды условного перехода;

УЖЕ ЕСТЬ - в команде языка ассемблера использована метка, значение которой уже определено ко времени первого прохода.

При использовании оператора ONERRGOTO в программе исполнение команды X=PEEK(218)+PEEK(219)*256 приведет к тому, что в X будет номер строки, в которой была ошибка.

Команда POKE 216,0 отключает выполнение оператора ONERRGOTO. После выполнения команды Y=PEEK(222) Y будет равен коду ошибки.

Коды ошибок и их соответствие диагностическим сообщениям русской и английской версиям приведены в табл. 5.4. Если седьмой бит ячейки с адресом 216 равен 1, это означает, что оператор ONERRGOTO уже выполнен. Это можно использовать в программе, например, так:

```
40 IF PEEK (216) 127 THEN GOTO 1000.
```

ОБРАЩЕНИЕ К ИНТЕРПРЕТАТОРУ ЯЗЫКА БЕЙСИК

Для пуска интерпретатора необходимо разместить его в ОЗУ начиная с адреса ↯ F00 и затем передать управление на этот адрес. Все действия выполняются автоматически программой "Дисковая операционная система" (ДОС) при включении ПЭВМ.

После пуска интерпретатора на экране ВКУ появляется символ] и курсор в виде мигающей черты, что определяет готовность интерпретатора к работе. После выполнения программы эти сигналы вновь появляются на экране ВКУ.

Для обращения к интерпретатору из программы "Системный монитор" используются следующие команды:

*E000 - для обращения к интерпретатору без сохранения программы и состояния интерпретатора;

*E003 - для обращения к интерпретатору с сохранением программы и состояния интерпретатора.

Г л а в а 6. ПРОГРАММИРОВАНИЕ НА ПЭВМ "АГАТ"

6.1. ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ БЕЙСИК

ПЭВМ предоставляет большие возможности программисту по созданию графического черно-белого и цветного изображения, проигрыванию на встроенном динамике музыкальных произведений, управлению объектами на экране с помощью аналоговых пультов, созданию баз данных; обработке информации и т.п.

Основным средством программирования в ПЭВМ "Агат" служит интерпретатор языка БЕЙСИК, который расширен до возможности применения в нем ДОС и ассемб-

лера "Агат". Учитывая это при написании программ, рекомендуется использовать следующие соглашения об областях памяти компьютера:

00-FF	- нулевая страница;
100-1FF	- стек МП 6502;
200-2FF	- буфер ввода-вывода;
300-3BF	- программы на ассемблере;
3C0-3FF	- векторы ДОС и монитора;
400-7FF	- страница 1 режима АЦР или ГНР;
800-BFF	- страница 2 режима АЦР или ГНР;
C00-1FFF	- для программ пользователя;
2000-3FFF	- страница 1 режима ГСР или ГВР;
4000-5FFF	- страница 2 режима ГСР или ГВР;
6000-95FF	- для программ пользователя;
9600-BFFF	- ДОС;
C000-CFFF	- ячейки ввода-вывода;
D000-DFFF	- резерв;
F000-F7FF	- интерпретатор языка БЕЙСИК;
F800-FFFF	- системный монитор.

Такое условное разбиение адресного пространства используется при программировании на компьютерах "ПРАВЕЦ-82", "APPLE II", "APPLE II+" и т.п.

Применение соглашения позволит избежать конфликтных ситуаций при переносе программ с одного типа компьютера на другой.

Отметим, что интерпретатор языка БЕЙСИК начинает располагать данные с адреса, определенного оператором LOMEM (по умолчанию 808) или директивой ДОС "FP". Команды (строки программы) интерпретатор располагает в памяти начиная с адреса $801. Последний байт программы определяется адресом, содержащимся в $AF и $B0. Кроме того, при программировании на языке БЕЙСИК могут быть полезны следующие советы:

используйте по возможности одноцифровые числа;

используйте однобуквенные имена в переменных;

избегайте использовать целочисленные (помеченные символом %) и строковые (помеченные символом $) переменные;

команду PRINT заменяйте на символ ?;

опускайте имя счетчика цикла на одиночном операторе NEXT; объединяйте два последовательных оператора NEXT в один (например, вместо NEXT I:NEXT J напишите NEXT I,J);

опускайте незначащие нули в дробных числах (.5 вместо 0.5);

когда это можно, оператор GOTO, как начинающий программу, заменяйте на RUN.

Рассмотрим примеры применения некоторых операторов интерпретатора языка БЕЙСИК. Используя оператор DATA в программе, можно запомнить константы, которые в ходе выполнения программы могут быть присвоены любым переменным.

Проиллюстрируем это на примере:

```
10  FOR N=1 TO 10 : READ A : PRINT A : NEXT
20  FOR K=1 TO 8 : READ B$ : PRINT B$ : NEXT
30  DATA 1,2,3,4,5,6,7,8,9,0, "A", "B", "C", "D", "E", "F", "G", "H"
```

В строке 10 организовано десятикратное повторение операторов **READ A** и **PRINT A**. Такое многократное повторение одной и той же группы операторов называется ***циклом***. Цикл строится командой **FOR** и параметрами этой команды: N - счетчик цикла, 1 - начальное значение счетчика цикла, 10 - конечное значение счетчика цикла, **NEXT** - конец цикла. Все операторы (**READ A** и **PRINT A**), стоящие между **FOR N=1 TO 10** и **NEXT**, будут повторены 10 раз, так как N будет менять свое значение от 1 до 10 тоже 10 раз.

В строке 30 записан оператор **DATA**, содержащий последовательность констант, разделенных запятыми.

При первом выполнении оператора **READ A** первая константа (1) из **DATA** будет прочитана в переменную **A**; при втором выполнении команды **READ A** вторая константа (число 2) из **DATA** будет снова прочитано в **A** (при этом старое содержимое в **A** стирается, но в операторе **DATA** константа не теряется) и т.д.

После очередного чтения из **DATA** в переменную **A** значение **A** отображается оператором **PRINT A** на экране ВКУ.

Применение других команд интерпретатора языка БЕЙСИК будет рассмотрено ниже в соответствующих разделах.

6.2. ПРОГРАММИРОВАНИЕ ВСТРОЕННОГО ИНТЕРФЕЙСА

Встроенный интерфейс ПЭВМ "Агат" является программно доступным.

Управление звуковым каналом. Управление динамиком основывается на возможности извлечь короткий звуковой сигнал при обращении к одной из ячеек C030 - C03F.

Для создания музыкальной мелодии нужно организовать многократное обращение к этим ячейкам. На рис. 6.1 приведена программа на языке ассемблера, воспроизводящая звуковой сигнал, длительность и тональность которого опреде-

```
*$300L

0300-  A0 C9      LDY  #$C9
0302-  98         TYA
0303-  AA         TAX
0304-  CA         DEX
0305-  D0 FD      BNE  $0304
0307-  AD 30 C0   LDA  $C030
030A-  88         DEY
030B-  D0 F5      BNE  $0302
030D-  60         RTS
030E-  00         BRK
```

Рис. 6.1. Программа воспроизведения мелодии

```
302: LDY #$00  ;загрузка 0
     в регистр Y
304: LDX $300  ;загрузка из 300
     ячейки  в X
307: LDA $C030 ;посылка импульса
     на   динамик
30A: DEY       ;Y=Y-1
30B: BNE $312  ;переход по
     ненулевому  результату
30D: DEC $301  ;содержимое ячейки
     301  уменьшили на 1
310: BEQ $317  ;переход по
     нулевому  результату
312: DEX       ;X=X-1
313: BNE $30A  ;переход по
     ненулевому результату
315: BEQ $304  ;переход по
     нулевому  результату
317: RTS ;возврат на точку входа
```

Рис. 6.2. Программа воспроизведения ноты

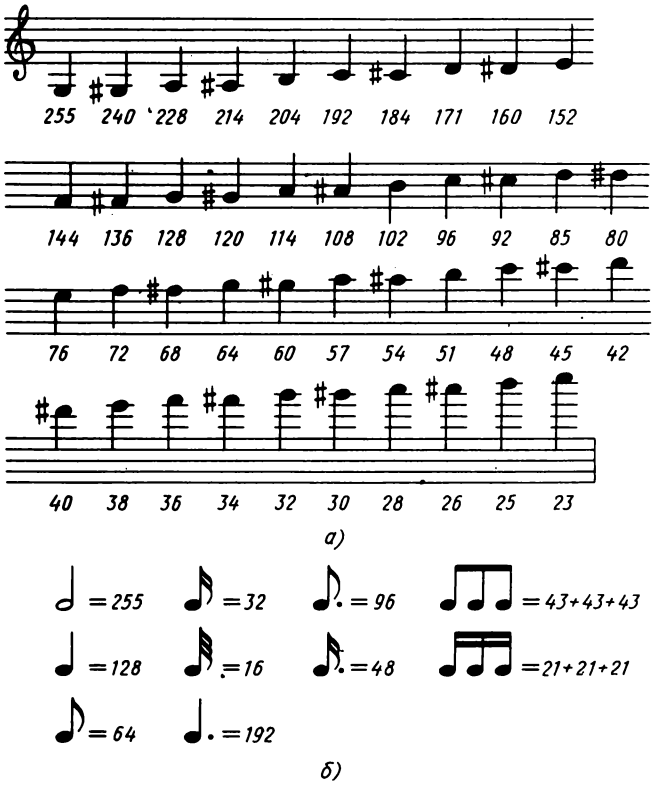


Рис. 6.3. Числовые характеристики нот

ляются числом ↑ C9, содержащимся в ячейке ↑ 301. Команда LDA ↑ C030 непосредственно извлекает один импульс, длительностью 5 мкс.

Для создания звукового сигнала, различимого человеческим ухом, в программе используются два цикла, построенные на двух командах BNE. Задав многократное повторение этой программы, можно получить музыкальный гудок. Если же при этом многократном обращении еще и изменять содержимое ячейки ↑ 301, получится случайная музыкальная мелодия.

Используя язык БЕЙСИК, можно написать программу, организующую такое многократное обращение:

```
10 FOR I=1 TO 255 : POKE 769,1 : CALL 768 : NEXT
```

Программа, состоящая из одной строки, выводит на динамик сигнал, напоминающий сирену. Конечно, перед запуском этой программы в память начиная с адреса ↑ 300, необходимо записать приведенную на рис. 6.1 программу. Это можно сделать, используя программу "Системный монитор" или операторы языка БЕЙСИК. Переведем шестнадцатеричные коды программы на языке ассемблера в десятичные эквиваленты (A0-160, C9-201 и т.д.):

160,201,152,170,202,208,253,173,48,192,136,208,245,96.

Эту последовательность на языке БЕЙСИК можно запомнить оператором DATA. Используя совместно с оператором DATA команды READ (чтение из DATA) и команды POKE (занесение байта по адресу), можно организовать запись программы, приведенной на рис. 6.1, непосредственно в память начиная с адреса 768 (↑ 302):

```
1 DATA160,201,152,170,202,208,253,173,48,192,136,208,245,96
2 FOR I=770    TO 781:  READ A : POKE I,A : NEXT
```

На рис. 6.2 приведена программа на языке ассемблера, извлекающая звук заданной длительности и тональности. Используя эту программу, можно заставлять компьютер исполнять любую мелодию. На рис. 6.3, *а* изображены нотные

Рис. 6.4. SIRENA-программа, воспроизводящая мелодию

```
LIST

10  GOSUB 210
20  INPUT "ВВЕДИ NC,P1,P2,
    DP,L-"; NC, P1, P2, DP
    ,L
30  REM NC-КОЛИЧЕСТВО ЦИКЛ
    ОВ
40  REM P1-НАЧАЛО ЦИКЛА ТО
    НАЛЬНОСТИ
50  REM P2-КОНЕЦ ЦИКЛА ТОН
    АЛЬНОСТИ
60  REM P -ТОН КАЖДОЙ НОТЫ
65  REM DP-ШАГ ИЗМЕНЕНИЯ Т
    ОНАЛЬНОСТИ НОТ
70  REM L -ПРОДОЛЖИТЕЛЬНОС
    ТЬ ЗВУЧАНИЯ ОДНОЙ НОТ
    Ы
80  GOSUB 120
90  END
100  POKE 768,P: POKE 769,
    L
110  CALL 770: RETURN
120  FOR J = 1 TO  NC
130  FOR P =  P1 TO  P2 STEP
     DP
140  GOSUB 100: NEXT P
150  FOR P =  P2 TO  P1 STEP
     -  DP
160  GOSUB 100: NEXT P,J: RETURN
200  DATA  160,0,174,0,3,1
     73,48,192,136,208,5,2
     06,1,3,240,5,202,208,
     245,240,237,96,256
210  AD = 770
220  READ X: IF X = 256 THEN
     RETURN
230  POKE  AD,X: AD =  AD +
     1: GOTO 220
```

знаки с десятичным числом под каждой нотой, определяющим ее тональность. На рис. 6.3, *б* приведены десятичные константы, характеризующие длительность звучания. Программа использует две ячейки (↑ 300 и ↑ 301). Перед запуском программы в ячейку ↑ 300 необходимо поместить число, эквивалентное высоте (тональности), а в ячейку ↑ 301 - число, эквивалентное продолжительности, воспроизводимой динамиком ноты.

На рис. 6.4 приведена программа SIRENA, организующая самые разнообразные музыкальные заставки. Ими можно разнообразить программы. Программа использует оператор GOSUB.

GOSUB означает переход на строку с номером, указанным в этом операторе. Как только в последующих операторах будет встречен оператор RETURN, происходит автоматический возврат на оператор, следовавший за GOSUB. В программе в строке 10 оператор GOSUB 200 передает управление на строку 210. Группа операторов, стоящая в строках 210,220,230, организует запись в память начиная с адреса 770 (↑ 302), констант, записанных в DATA. Запись в память осуществляется до тех пор, пока не будет прочитано число 256, по которому происходит возврат (оператор RETURN) на строку 10.

В прил. 2 приведена программа: ВАЛЬС, использующая программу, представленную на рис. 6.3 и записываемую в память аналогично тому, как это делается в операторах 210, 220, 230 программы SIRENA.

В заключение отметим еще раз, что.

770 = ↑ 302, 769 = ↑ 301, 768 = ↑ 300

Ввод данных с клавиатуры. Ввод данных с клавиатуры можно организовать, например, так:

```
100  WAIT ↑ C000, ↑ 80,0 : X ↑ = CHR ↑ (PEEK( ↑ C000))
200  POKE ↑ C010,0
```

В результате работы этой программы в переменной X ↑ будет находиться код последней нажатой клавиши. Поскольку признаком нажатой клавиши служит единица в старшем разряде, то для получения действительного кода (в КОИ-8) необходимо эту единицу исключить. Следующая программа распечатывает на экра-

не зеленым цветом код нажимаемой клавиши, сопровождая это сообщение звуковым сигналом:

```
 5  RIBBON = 3; X=PEEK( ¤ C000): POKE ¤ C010,0
10  IF X=0 THEN PRINT : GOTO 10 : HTAB 15
20  X=X-128 : PRINT CHR ¤ (7) : PRINT X : GOTO 5
```

Программирование пультов. Пульты позволяют управлять положением символов или объектов на экране ВКУ, что открывает широкие возможности по написанию динамических программ.

Считывая состояние ручки потенциометра пульта, можно связывать с ним координату воспроизводимого символа или объекта на экране.

Изменение (при вращении) положения ручки потенциометра приводит к изменению координаты, а значит, и положения объекта на экране. Так, например, в программе "Теннис", приведенной в прил. 2, в операторах с номерами 1100-1300 и других определяется положение ручек пультов и в соответствии с этим осуществляется управление положением ракеток на экране. Программа

```
10  X=PEEK( ¤ C062) : Y=PEEK( ¤ C061):X = INT(X/255)
20  Y=INT(Y/255) : X1 = PDL(1) : Y1 = PDL(0)
30  RIBBON=2: PRINT X1;"-"; RIBBON=3 : PRINT X; "␣";
40  RIBBON=4:PRINT Y1; " - "; RIBBON=7:PRINT Y : GOTO 10
```

Вычисляет координаты положения ручек потенциометров обоих пультов в виде целых чисел от 0 до 255 (отображает зеленым и синим цветом на экране ВКУ). Кроме того, программа считывает состояние кнопок пультов. При нажатии кнопок изменяется соответствующее им число на обратное (0 на 1 или 1 на 0).

Эту программу можно использовать как тест проверки работоспособности пультов.

Таймерные прерывания. Таймерные прерывания открывают широкие возможности по использованию компьютера в режиме реального времени. Например, с помощью таймерных прерываний легко реализуется программа "Часы". После обращения по адресу C040 по линиям $\overline{IRQ}$ и $\overline{NMI}$ на соответствующие входы микропроцессора начинают поступать импульсы, прерывающие его работу. Так как прерывания поступают с фиксированной частотой, в качестве программ обработки прерываний можно организовать счетчики, подсчитывающие число поступавших импульсов, соответствующее интервалу времени.

Этот принцип используется в программе CLOCK, приведенной в прил. 2. Число поступивших импульсов по входу $\overline{NMI}$ подсчитывается в четырех ячейках: 7083 - накапливаются часы, 7082 - минуты, 7081 - секунды, 7080 - пятидесятые доли секунд.

Адреса векторов прерывания лежат в области, отведенной под ПЗУ. Поэтому в программе предусмотрено изменение соответствующих ячеек векторов $\overline{NMI}$ и $\overline{IRQ}$, находящихся в псевдоПЗУ. Запуск часов осуществляется обращением по адресу C040-C04F, выключение - по адресу C050 - C05F. Посмотреть, как "идут" часы можно обратившись по адресу C7BA. После включения 46-й текстовой страницы в режиме АЦР 64 первые четыре символа в третьей строке сверху изменяются в соответствии с назначением ячеек. В четвертой позиции символы изменяются с

частотой 50 символов в секунду, в третьей - 1 символ в секунду, во второй - 1 символ в минуту и в первой - 1 символ в час.

Программа функционирует независимо от интерпретатора языка БЕЙСИК и располагается в памяти с адреса Ч 7000 по адрес Ч 7080.

Этот же принцип использован в программе "Часы", приведенной в прил. 2.

Рассмотрим в качестве примера программы, работающей в режиме реального времени (использующей таймерные прерывания), программу SOVM, приведенную в прил. 2. Программа позволяет совмещать на экране ВКУ графику с четырьмя текстовыми строками.

SOVM поочередно, с привязкой к таймерному прерыванию, отображает на экране часть графической и текстовой страницы.

В заключение заметим, что после обращения по адресу C220 псевдоПЗУ стало доступным по записи в него, а стандартное ПЗУ - по чтению. Это дает возможность изменить содержимое, например, векторов $\overline{\text{IRQ}}$ и $\overline{\text{NMI}}$. После записи изменений в псевдоПЗУ его необходимо включить в режим чтения обращением по адресу C200.

Магнитофонный вход. ПЭВМ предоставляет пользователю возможность не только хранить программы и данные на магнитной ленте. Используя интерфейс магнитофона, можно, например, организовать обмен данными между двумя и более компьютерами по кабелю, имеющему длину не более 10 м. Для этого магнитофонный выход первого компьютера соединяется с магнитофонным входом другого и наоборот, после чего по команде SAVE на первом компьютере и команде LOAD на втором происходит передача-прием программы на языке БЕЙСИК между двумя ПЭВМ. Уровень передаваемого сигнала порядка 200-250 мВ.

Использование интерфейса магнитофона этим не ограничивается. Вопросы, связанные с передачей данных посредством телефонного канала и организации локальной сети, рассмотрены в гл. 11.

6.3. ПРОГРАММИРОВАНИЕ ГРАФИКИ

Изображение на экране ВКУ формируется цветовым пером (курсором). Толщина пера зависит от выбранного графического режима.

В состав языка БЕЙСИК включены операторы управления этим пером. По желанию программиста можно рисовать либо поднятым, либо опущенным пером. На рис. 6.5 приведена программа, рисующая на экране зеленую елочку, на вершине которой красным цветом мигает звезда. В строке 10 программы задается графический режим среднего разрешения (толщина пера - 2×2 точки) и выбирается зеленый цвет (COLOR=2). Командой PLOT осуществляется перемещение пера по точкам координатной сетки 128 × 128 точек.

Начало координат находится в левом верхнем углу. Координаты X изменяются слева направо, соответственно от 0 до 128. Координаты Y изменяются сверху вниз, соответственно от 0 до 128. Таким образом, чтобы нарисовать линию, необходимо в операторе PLOT задать координаты начальной и конечной точки. Так, в программе в строке 50 формируется положение зеленого треугольника с вершиной в точке, имеющей координату X=64, Y=19. Основание треугольника определяется координатой Y=40, а боковые стороны зависят от переменной X;

```
LIST

5  REM РИСУНОК ЕЛОЧКИ
10  MGR= 2
12  HOME
20  COLOR= 2
30 X = 43
40  FOR I = 1 TO 43
50  PLOT 64,19 TO X,40
60 X = X + 1
70  NEXT I
80 X = 30
90  FOR I = 1 TO 66
100  PLOT 64,32 TO X,70
110 X = X + 1
120  NEXT I
130 X = 16
140  FOR I = 1 TO 86
150  PLOT 64,43 TO X,100
160 X = X + 1
170  NEXT I
180  COLOR= 5
190  PLOT 62,100 TO 62,110
      TO 63,110 TO 63,100 TO
     64,100 TO 64,110 TO 6
     5,110 TO 65,100
195 K = 1
200  FOR I = 1 TO 10000
210  COLOR= 1
230  PLOT 64,10 TO 60,19 TO
     70,15 TO 58,15 TO 68,
     19 TO 64,10
231  COLOR= 0
232  PLOT 64,10 TO 60,19 TO
     70,15 TO 58,15 TO 68,
     19 TO 64,10
240  NEXT I
250  COLOR= K
260  IF K = 2 THEN K = 7
270  PLOT X,Y:X = 02:Y = X
      ^ 2
```

Рис.6.5. Программа использования графических операторов

изменяющейся в цикле от 43 до 86. Операторы в строках 195-240 задают изображение на вершине елки мигающей звезды.

В состав базового обеспечения ПЭВМ "Агат" входит графический редактор GRED. Редактор выводит на экран курсор, соответствующий среднему разрешению. Управление курсором осуществляется стрелками → , ←, ↑, ↓; выбор цвета - нажатием начальной буквы цвета. Опускается и поднимается перо нажатием на клавишу "Пробел". Редактор позволяет заштриховать область любым цветом. Для этого нужно перевести курсор в эту область, поднять перо, выбрать цвет закрашивания и нажать клавишу "*".

Пример графического редактора, написанного на языке БЕЙСИК, приведен на рис. 6.6. Представленная программа работает в графическом режиме высокого разрешения. Разобраться в работе этого редактора предлагается самостоятельно. Заметим только, что для расшифровки назначения клавиш, участвующих в управлении программой, полезно воспользоваться операторами, рассмотренными в п. 6.1 в программе ввода с клавиатуры.

Другой возможностью графического изображения, предоставляемо компьютером, является воспроизведение образов на экране ВКУ. Программа, управляющая положением образов на экране, приведена на рис. 6.7. Для ее функционирования необходимо составить двоичную таблицу, кодирующую вид образа, и разместить ее в памяти. Младший адрес таблицы заносится в ячейку)(E8, старший с - в ячейку)(E9. В программе предполагается наличие такой таблицы в виде двоичного файла на магнитном диске. Загружается файл (строка 5) в память начиная с адреса 1DFC.

Двоичная таблица имеет вид:

01 00 04 00 12 3F 20 64 2 15 36 1E 07 00 5F 46
11 8A 2A A6 B8 0 02 C6 B8 C6 B8 A0 1B 85 89

Подобную таблицу можно составить самостоятельно, руководствуясь принципами ее построения, изложенными в гл. 5. Тогда программа воспроизведет закодированный образ. Строка 10 задает графическую страницу 3 в режиме высокого разрешения. В строке 15 оператором SKALE задается масштаб воспроизводимого символа, а в строке 20 - угол поворота образа на экране. Воспроизведение образа осуществляется оператором XDRAW.

```
]LIST
100  REM * ГРАФИЧЕСКИЙ РЕД
     АКТОР:REM * РЕЖИМ 256
     *256
110  REM ПЕРО УПРАВЛЯЕТСЯ
     КЛАВИШАМИ НА БК
120  INPUT "ВВЕДИТЕ НОМЕР
     ГРАФИЧЕСКОЙ СТРАНИЦЫ"
     ; GP
130  HGR=  GP: CLEAR
140 X = 128:Y = 128:M = 1
1000  REM ПРОВЕРКА КОДА НА
     ЖАТОЙ КЛАВИШИ
1005 I =  PEEK ($C000): IF
     I > 127 THEN I = I -
     128
1006  IF I = 87 THEN 4000
1007  IF I = 91 THEN M = 1
1008  IF I = 82 THEN 5000
1010  IF I = 48 THEN C = 0
1015  IF I = 49 THEN C = 1
     5
1020  IF I = 8 THEN 3000
1025  IF I = 21 THEN 3000
1030  IF I = 25 THEN 3000
1035  REM ПЕРЕМЕЩЕНИЕ ПЕРА
1037  COLOR= 10: PLOT X,Y:
      PLOT  X1, Y1
1040  IF I = 78 THEN  X1 =
     X: Y1 = Y
1045  IF I = 76 THEN I = 0
     : GOSUB 2000
1050  IF I = 80 THEN I = 0
     : GOSUB 2005
1055  IF I = 75 THEN 2030
1060  IF I = 15 THEN  X1 =
     0: Y1 = 0: HGR= 1
1063  IF I = 79 THEN  GOSUB
     2800
1065  COLOR= 0: PLOT X,Y
1070  COLOR= C: PLOT X,Y
1080  GOTO 1000
2000  COLOR= C: PLOT  X1, Y1
      TO X,Y: RETURN
2005  COLOR= C
2010  FOR J =  X1 TO X
2015  PLOT J,Y TO J, Y1
2020  NEXT J
2025  RETURN
2030  TEXT= 0: HOME : RIBBON=
     3: END
2800  COLOR= C:R =  ABS (X
      -  X1)
2805  FOR Z = 0 TO 6.28 STEP
     0.03
2810 X =  X1 + R *  COS (Z
     )
2815 Y =  Y1 + 1.33 * (R *
      SIN (Z))
2820  PLOT X,Y
2825  NEXT Z
2830  RETURN
2900  IF X =  < 0 THEN X =
     255: GOTO 2915
2905  IF Y =  < 0 THEN Y =
     255: GOTO 2915
2910  IF X >  = 255 THEN X
      = 0
2911  IF Y >  = 255 THEN Y
      = 0
2915  COLOR= 15: PLOT X,Y:
      PLOT  X1, Y1
2920  COLOR= 0: PLOT X,Y: PLOT
      X1, Y1
2990  COLOR= C: PLOT X,Y
2997  IF M = 1 THEN  GET A
     $:I =  ASC (A$): GOTO
     3005
3000 I =  PEEK ($C000) - 1
     28
3005  IF I = 8 THEN X = X -
     1: GOTO 2900
3010  IF I = 21 THEN X = X
      + 1: GOTO 2900
3015  IF I = 25 THEN Y = Y
      - 1: GOTO 2900
3020  IF I = 26 THEN Y = Y
      + 1: GOTO 2900
3025 M = 0
3030  GOTO 1000
4000  TEXT=  TXT: HOME
4005  INVERSE : PRINT "ЗАП
     ИСЬ НА МД": NORMAL
4010  INPUT "ИМЯ ИЗОБРАЖЕН
     ИЯ?";A$
4015  PRINT  CHR$ (4);"BSA
     VE";A$;",A$4000,L$1FF
     F"
4020  PRINT "ЗАПИСЬ ЗАКОНЧ
     ЕНА": GOTO 4025
4025  GET A$
4030  GOTO 1000
5000  TEXT=  TXT: HOME
5005  HOME : PRINT : PRINT
      CHR$ (4);"CATALOG":
5007  NORMAL : INPUT "ИМЯ
     ИЗОБРАЖЕНИЯ?";A$
5010  HGR= 2: PRINT  CHR$
     (4);"BLOAD";A$
5015  GET A$
5020  GOTO 1000
9000  GET A$: PRINT A$;"-"
     ; ASC (A$): GOTO 9000
```

Рис. 6.6. Графический редактор

Рис. 6.7. Программа управления положением образа на экране

```
]LIST

2  REM УПРАВЛЕНИЕ ОБРАЗОМ
     НА ЭКРАНЕ ВКУ
10  REM ТАБЛИЦА X :
20  REM 1DFC: 01 00 04 00
     12 3F 20 64 2D 15
30  REM 1E05: 36 1E 07 00
     5F 46 11 8A 2A A6
40  REM 1E0F: B8 D0 02 C6
     B9 C6 B8 A0 1B 85
50  REM 1E1A: 89
60  POKE $E8, FC: POKE $E9
     ,$1D
70  PRINT  CHR$ (4);"BLOAD
     X,A$1DFC"
80  HGR= 3: FOR M = 1 TO 2
     0: SKALE = M
90  FOR K = 0 TO 64: ROT=
     K: COLOR= 7
100  XDRAW 1 AT 127,127: COLOR= 0
110  XDRAW 1 AT 127,127: NEXT
     : END
```

6.4. РАБОТА С НГМД

ОРГАНИЗАЦИЯ ЗАПИСИ НА МАГНИТНЫЙ ДИСК ПОД УПРАВЛЕНИЕМ ДОС

В ПЭВМ "Агат" пользователю предоставлены две возможности работы с НГМД:
под управлением операционной системы ДОС;
программируя на физическом уровне в машинных кодах.

Как и любая дисковая система, ДОС "Агат" состоит из двух основных частей:

подсистемы распределения памяти и организации массивов, которая обеспечивает программисту быструю и удобную работу над массивами. Эта часть ДОС реализована программно;

подсистемы контроллер - диск, которая обеспечивает физически процесс записи информации на магнитный носитель и ее считывание.

Управление подсистемой осуществляется с помощью ДОС специальными программами, выдающими обобщенные команды типа: ЧТЕНИЕ СЕКТОРА, ЗАПИСЬ В СЕКТОР, ФОРМАТИРОВАНИЕ и т.п. Таким образом, в ДОС "Агат" дисковый контроллер реализован программно, т.е. упомянутые команды реализуются программным способом, а физический контроллер НГМД выполняет только элементарные команды типа: ЗАПИСЬ БАЙТА, ЧТЕНИЕ БАЙТА и т.д., причем под непосредственным контролем ДОС.

ГМД для записи на него информации делится (форматируется) на дорожки и секторы. ДОС разбивает диск на 35 дорожек по 16 секторов на каждой дорожке (рис. 6.8). В каждый сектор можно записать до 256 байтов данных. Нормальный объем информации, которую можно записывать на одном диске:
35 дорожек × 16 секторов × 256 байт = 143 360 байт, или 142К байт

Форматирование. При записи информации на ГМД в компьютере применяются два способа записи (или кодирования) данных. В обоих способах вместе с данными записываются сигналы синхронизации. Объясняется это сравнительно широким

допуском на частоту вращения диска, что приводит к изменению моментов появления (или "плаванию") считываемых сигналов. Последовательный поток данных разделяется на битовые элементы, каждый из которых кодирует двоичный нуль или единицу.

Различие способов записи заключается в формировании сигналов собственно даных DB (DATA BIT) и синхронизации CB (CLOCK BIT). Биты информации записываются на ГМД в точно определенных интервалах времени, которые фиксируются тактовым импульсом CB. Интервал времени между двумя соседними CB определяет время, за которое записывается или считывается один бит информации DB.

Рассмотрим временную диаграмму записи двоичного числа 10101 (рис. 6.9).

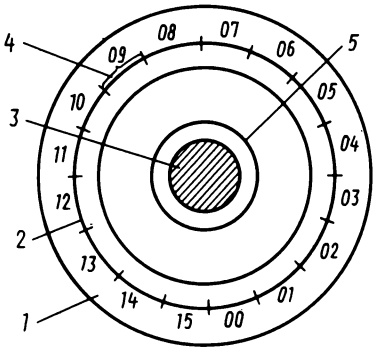
В случае, когда бит информации имеет значение единицы, между импульсами CB появляется информационый импульс (DB=1), а когда бит информации равен нулю, информационного импульса нет (DB=0).

Этот метод записи называется методом частотной модуляции или FM-методом. В литературе его иногда называют методом записи с удвоенной частотой (ЧМ). Нетрудно заметить, что при записи или считывании одних нулей формируется поток импульсов с частотой f, а при записи или считывании одних единиц - поток с частотой 2f, что и объясняет название способа записи.

В FM-методе более половины переходов магнитного потока вызываются битами синхронизации (импульсами CB), и для увеличения плотности записи можно применять более эффективные способы кодирования. В ДОС "Агат" наряду с FM-методом используется и метод записи, описание которого приведено ниже.

Каждый сектор делится ДОС на адресное поле и поле данных.

Адресное поле. Адресное поле содержит адресную информацию, что делает доступным для ДОС любой из 560 секторов магнитного диска. Адресное поле идентифицирует каждый сектор, для чего в нем размещены друг за другом указатель начала (пролог), поле номера тома (VOLUME), поле номера дорожки



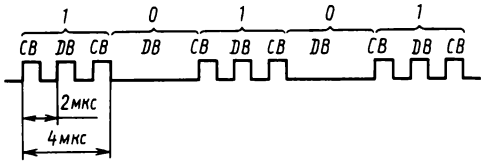


Рис. 6.8. Структура ГМД в ДОС "Агат":
1 - магнитная поверхность диска; 2 - дорожка №-00; 3 - отверстие для вала НГДМ; 4 - сектор №-09; 5 - дорожка №-34

Рис. 6.9. Временная диаграмма записи двоичного числа

(TRACK), поле номера сектора (SECTOR), поле контрольной суммы (CHECK SUM) и указатель конца (эпилог):

D5	AA	96	XX, YY	XX, YY	XX, YY	XX, YY	DE	AA	EB

Пролог VOLUME TRACK SEKTOR CHECK SUM Эпилог

Адресная информация кодируется по FM (ЧМ)-методу, и поэтому каждый записанный байт данных в этом поле занимает два байта.

Кодирование каждого байта адресной информации выполняется программным способом следующим образом:

нечетный полубайт получается при сдвиге основного байта вправо на один бит и выполнении операции ИЛИ с байтом AA_{16};

четный полубайт получается в результате операции ИЛИ основного байта и байта AA_{16}.

Суть кодировки состоит в том, что все нечетные разряды содержат единицу (биты синхронизации), а все четные являются значимыми (см. табл. 6.1).

Таблица 6.1

Кодировка адресной информации

Назначение байта	Разряды байта							
Байт, поступивший на запись	D7	D6	D5	D4	D3	D2	D1	D0
Записанный нечетный байт (XX)	1	D7	1	D5	1	D3	1	D1
Записанный четный байт (YY)	1	D6	1	D4	1	D2	1	D0

Например, для дорожки 21 (вспомним, что $21_{10} = 15_{16}$ и т.п.):

Основной байт 00010101 = 15_{16};

Нечетный байт 10101010 = AA_{16};

Четный байт 10111111 = BF_{16}.

т.е. байт 15_{16}, соответствующий номеру дорожки 21, кодируется в поле TRACK как два байта AA BF.

Аналогично кодируются в двух байтах номер тома, номер сектора, контрольная сумма.

Поле данных. Это поле по структуре такое же, как адресное, но содержит 256 байт действительно пользовательской информации.

Начинается поле с пролога, за которым следуют данные пользователя, контрольная сумма и эпилог. Пролог состоит из трех байтов D5, AA, AD. Эпилог также состоит из трех байтов DE, AA, EB.

Информация в поле данных, подобно адресной, также кодируется, но использованный метод кодирования позволяет достичь значительно более высокой плотности записи.

FM-метод, используемый при кодировании адресной информации, приводит к возрастанию в 2 раза числа байтов, т.е. если необходимо записать в одном секторе 256 байт, то действительное число записанных байтов будет 512 за счет записи в каждом байте нечетных битов, равных единице (это и есть синхроимпульсы СВ).

Для увеличения количества записываемой информации при записи данных в поле данных число СВ-импульсов уменьшено. Бит СВ (синхроимпульс) записывается только тогда, когда текущий битовый элемент содержит нуль, а в предыдущем битовом элементе не было изменения потока ни от данных, ни от синхронизации. Этим достигается запись 342 байт вместо 512.

Способ получил название модифицированное FM-кодирование или MFM-метод. По сравнению с FM в MFM-методе наблюдается увеличение плотности записи на 66 %.

Поле данных можно представить следующим образом:

D5	AA	AD	342 байт	xx, xx	DE	AA	E8
Пролог			Данные	CHECK SUM	эпилог		

При MFM-кодировании каждый байт, присланный компьютером для записи на магнитный диск, обрабатывается следующим образом.

1. Приводится к виду 00XXXXXX, т.е. преобразуется в "укороченный" байт (шестибитовый). При этом два старших бита сохраняются и комбинируются с другими подобными парами до получения "короткого" (шестибитового) байта. Видно, что из каждых трех байтов формируются четыре "коротких" байта ("короткими" эти байты называтся потому, что два старших бита равны нулю). Таким образом, из 256 нормальных байтов информации получается:

$(256 \times 4):3 = 341.3 \approx 342$ "коротких" байта.

Возможное число коротких байтов составляет $2^{16} = 64$ и изменяется от 00_{16} до $3F_{16}$.

2. Полученные 342 байта заменяются (кодируются) на 342 полных байта с соблюдением следующих условий:

старший байт (D7) всегда равен единице;

байт не содержит более одной пары соседних нулей;

байт не содержит более одной пары несоседних единиц.

Эти условия отвечают MFM-методу кодирования.

Ниже приведена кодировка коротких байтов в полные по MFM-методу:

00-96	10-B4	20-D6	30-ED
01-97	11-B5	21-D7	31-EE
02-9A	12-B6	22-D9	32-EF
03-9B	13-B7	23-DA	33-F2
04-9D	14-B9	24-DB	34-F3
05-9E	15-BA	25-DC	35-F4
06-9F	16-BB	26-DD	36-F5
07-A6	17-BC	27-DE	37-F6
08-A7	18-BD	28-DF	38-F7
09-AB	19-BE	29-E5	39-F9
0A-AC	1A-BF	2A-E6	3A-FA
0B-AD	1B-CB	2B-E7	3B-FB
0C-AE	1C-CD	2C-E9	3C-FC
0D-AF	1D-CE	2D-EA	3D-FD
0E-B2	1E-CF	2E-EB	3E-FE
0F-B3	1F-D3	2F-EC	3F-FF

Байты AA и D5 исключены из таблицы, так как они используются при записи эпилога и пролога.

Прежде чем записать даные на ГМД, необходимо осуществить их перекодировку - каждые шесть битов информации пользователя в восемь битов информации, записываемой на дискету.

Все разрешенные значения байтов находятся в диапазоне 96-FF. Считанную с диска информацию нужно перекодировать обратно (каждые восемь битов в шесть битов информации) и упаковать в полные байты.

С точки зрения пользователя ДОС на одном секторе умещается 256 байт информации. Контрольная сумма (CHECK SUM) кодируется так же, как в адресном поле.

Самосинхронизация и разделительные поля. Поле адреса и поле данных ограничены специальными разделительными промежутками (участками), GAP, заполненными определенным числом специальных байтов. Необходимость в их применении заключается в следующем:

обеспечивают при записи информации в данном поле сохранение информации в соседних полях;

обеспечивают компьютеру время для дешифрации адресной информации до того, как искомое поле данных достигло головки чтения-записи.

Очередность различных типов полей и промежуточных участков на одной дорожке (рис. 6.10) одинакова для всех дорожек магнитного диска и определяется в процессе форматирования (команда INIT в ДОС "Агат"). Одинакова также структура всех секторов.

Информацию, записанную на одну дорожку мини-диска, можно рассматривать как непрерывный поток импульсов, и на первый взгляд кажется невозможным выделение отдельных байтов, поскольку нет указателя начала и конца одного байта. Сделать это позволяет использование разделительных участков, в которых записаны специальные самосинхронизирующиеся байты. Каждый самосинхронизирующийся байт является нормальным байтом, равным FF_{16}, к которому

добавляются два бита с нулевыми значениями, т.е. 1111111100. Эти два нулевых бита записываются на диск автоматически контроллером НГМД.

Подпрограмма СЧИТЫВАНИЕ в ДОС, которая следит за заполнением регистра сдвига контроллера, принимает решение, что операция закончена только тогда, когда старший бит обращается в единицу.

На рис. 6.11 показана синхронизация контроллера в режиме считывания информации с промежуточных участков; при этом синхронизация достигается в рамках пяти последовательных синхробайтов.

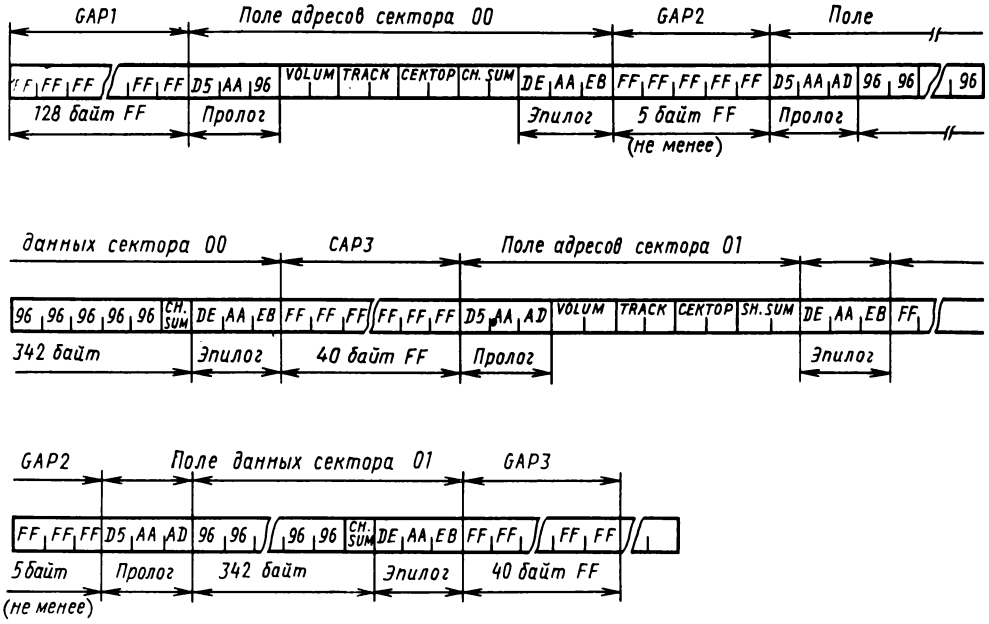


Рис. 6.10. Форматирование дорожки магнитного диска

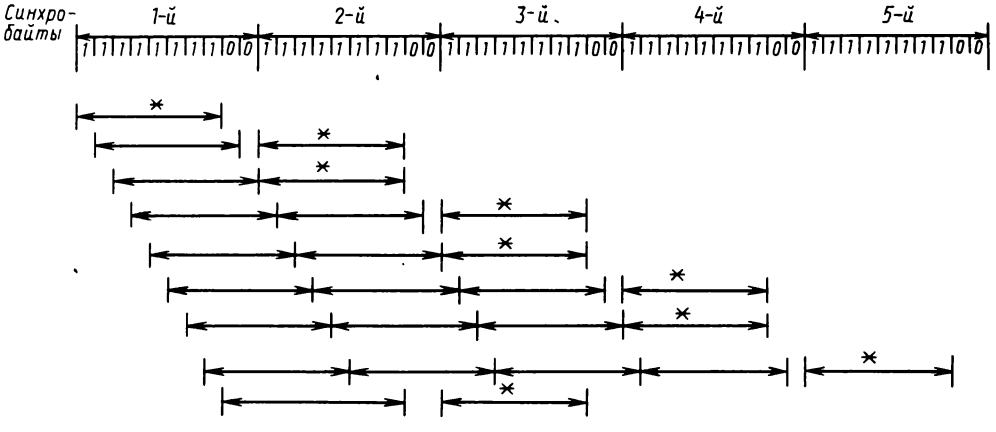


Рис. 6.11. Возможные случаи начала самосинхронизации (символом * отмечен байт, в котором произошла самосинхронизация контроллера с данными)

Возможны девять различных начальных моментов процесса синхронизации, которые определяются моментом запуска подпрограммы СЧИТЫВАНИЕ. Этот метод самосинхронизации контроллера по информации не требует, чтобы программа следила за синхробайтами. Достаточно после старта подпрограммы СЧИТЫВАНИЕ отыскать соответствующий указатель начала поля (ПОЛЯ АДРЕСА или ПОЛЯ ДАННЫХ). При этом можно гарантировать, что в рамках одного оборота (рис. 6.11) мини-диска этот указатель будет найден.

На каждой дорожке размещены три типа разделительных участков (см.рис. 6.10), заполненные синхробайтами; эти участки различаются только по длине и предназначению:

GAP1 представляет собой последовательность из 128 синхробайтов и генерируется только один раз во время форматирования дорожки. Синхробайты в некоторой степени компенсируют изменение частоты вращения и деформацию диска. **GAP1** предназначен для индикации начала каждой дорожки, т.е. после него следует сектор 00, который выполняет функцию буфера между адресным полем сектора 00 и полем данных последнего сектора 15. Из-за непостоянной частоты вращения диска часть данных сектора 15 может перейти в участок **GAP1**, и при этом адресное поле сектора 00 не разрушается;

GAP2 разделяет ПОЛЕ АДРЕСА от следующего за ним ПОЛЯ ДАННЫХ и имеет длину от 5 до 10 байт. Этот промежуточный участок обеспечивает дисковой системе время для дешифрации адресного поля. При записи новых данных в ПОЛЕ ДАННЫХ нельзя предсказать начальный момент команды ЗАПИСЬ, и разрушение хотя бы одного синхробайта в **GAP2** вполне возможно. Поэтому подпрограмма ЗАПИСЬ начинается генерацией пяти синхробайтов, с последующей генерацией указателя начала ПОЛЯ ДАННЫХ, и только тогда записывается действительная информация. Это гарантирует самосинхронизацию при считывании информации (рис. 6.11);

GAP3 разделяет ПОЛЕ ДАННЫХ от ПОЛЯ АДРЕСА следующего сектора и состоит из 40 синхробайтов, обеспечивающих время, необходимое компьютеру для обработки уже считанных данных. Если компьютер не успевает в течение времени прохождения этого участка обработать полученные данные и обратиться к следующему сектору, то он это сделает при следующем обороте диска.

Заметим, что пролог и эпилог позволяют управляющим программам определять положение головки чтение - запись относительно соответствующей дорожки.

Каталог. Для организации и ведения файловых структур ДОС размещает свой каталог, начиная с дорожки 17. На дорожке 17 каталог начинается с сектора 15 и кончается сектором 0. В начале ПОЛЯ ДАННЫХ каждого сектора ДОС записываются три байта служебной информации:

первый байт - признак продолжения или ноль;

второй байт - дорожка, продолжение каталога;

третий байт - сектор, продолжение каталога.

После трех служебных байтов идут восемь байтов, не задействованных (резерв), за которыми начинается поле файлов. Если каталог, заведенный под файлы на дискете, помещается на одной дорожке (17), то в этом случае на всех секторах в ПОЛЕ ДАННЫХ второй байт ссылается на дорожку 17, а третий байт старшего сектора - на младший (например, 15 на 14, 14 на 13 и т.д.). При этом нулевой сектор ссылается на сектор 15, т.е. каталог как бы закольцовывается. При невозможности разместить информацию о файлах на ГМД на одной

17-й дорожке, в ПОЛЕ ДАННЫХ сектора 00 второй и третий байт укажут, где продолжается каталог.

Каждому файлу в каталоге (на секторе в ПОЛЕ ДАННЫХ отводится поле файла, имеющее фиксированную длину 36 байт. Структура поля:

TR	SP	T	NAME	QS	R
1 байт	1 байт	1 байт	30 байт	2 байт	1 байт

Здесь:

TR - номер дорожки, SR - номер секторов, на которых начинают размещаться данные файла; QS - число секторов, занимаемых файлом; T - тип файла: 2 - файл на языке БЕЙСИК; 4 - двоичный файл; 0 - текстовый файл. Кроме того, байт T определяет, защищен или нет файл по записи. Если файл защищен, этот байт увеличивается на 128 (o 80). Байт R оставлен для резерва.

Под имя файла NAME отведено 30 байт. Отметим, что TR и SR могут содержать ссылку на ссылку, где размещаются данные файла. В этом случае они указывают на дорожку и сектор, содержащие такую ссылку в байтах 13 и 14 соответственно. Таким образом, секторы, отведенные под данные файлов, имеют ссылки в байтах 13 и 14 на участки, где эти данные начинаются или продолжаются.

При удалении файла в каталоге соответствующий байт TR, указывающий начало дорожки, увеличивается на 160 (ϕ A0). Данные файла при этом сохраняются.

В прил. 2 приведена программа DRW. Программа работает в диалоге, задавая следующие вопросы:

V? - указать номер диска (как правило, указывается 254, но если указать 0, будет работать с любым диском);

R/W? - читать или писать, ответ: 1 - для чтения, 2 - для записи;

T? - трек, ответ - номер дорожки (0 - 34);

S? - сектор (0 - 15).

При задании режима чтения программа считывает указанный далее сектор и записывает его в буфер (ϕ 2000 - ϕ 20FF), после чего переводит данные в "видимые" символы и распечатывает на экране ВКУ по следующим правилам:

коды 32 - 127 отображаются в режиме NORMAL (белым по черному);

коды 128 + 32 + 128 + 127 отображаются в режиме INVERSE (черным по белому);

коды 0 - 31 (контрольные байты) отображаются на экране красным цветом (по черному), т.е. @ - байт ϕ 00; A - байт ϕ 01, ...; Z - ϕ 1A и т.д.

Коды 128 + 127 + 31 (контрольные байты) отображаются черным по красному.

Таким образом, любой байт от ϕ 00 до ϕ FF становится видимым. Использование этой программы может быть полезным при работе с ДОС. Заметим, что если указать режим записи (2 в ответе на вопрос R/W?), то в указанный сектор будет записана информация из ячеек ϕ 2000 - ϕ 20FF. Это можно использовать для корректировки сбойных участков на ГМД, включая каталог.

УПРАВЛЕНИЕ РАБОТОЙ НАКОПИТЕЛЯ

Управление дисководом осуществляет контроллер НГМД. Включение контроллера в работу производится сигналом $\overline{DS}$. Этот сигнал вырабатывается при обращении

центрального процессора по любому из адресов COXO-COXF. Здесь X = 8 + N, где N - номер разъема, в который установлен контроллер. В формировании сигналов управления дисководом участвуют следующие адреса:

C0X1 - включение фазы φ0;
C0X3 - включение фазы φ1;
C0X5 - включение фазы φ2;
C0X7 - включение фазы φ3;
C0X0 - выключение фазы φ0;
C0X2 - выключение фазы φ1;
C0X4 - выключение фазы φ2;
C0X6 - выключение фазы φ3;
C0X8 - выключение двигателя вращения дискеты;
C0X9 - включение двигателя вращения;
C0XA - включение привода № 1 (первого дисковода);
C0XB - включение привода № 2 (второго дисковода);
C0XC - чтение данных из регистра чтения;
C0XD - запись данных в регистр записи;
C0XE - включение режима чтения даных с ГМД в регистр чтения;
C0XF - включение режима записи данных из регистра записи на ГМД.

Дисковый контроллер обеспечивает управление двумя дисководами. В каждый момент времени контроллер может управлять только одним НГМД, который выбирается инструкциями:

LDA ☼ C08A , X - выбор дисковода 1 ;
LDA ☼ C08B , X - выбор дисковода 2,

где X = N0 (т.е. X = NX ☼ 10).

Двигатель вращения, приводящий в движение ГМД, включается на выбранном дисководе после выдачи команды включения LDA ☼ C089, X. Управляющая программа ДОС реализует необходимое запаздывание до достижения номинальной частоты вращения (300 об/мин).

Устройство выключается при выдаче микропроцессором команды LDA ☼ C088, X. При этом контроллер осуществляет задержку на 5 - 6 оборотов.

Во время вращения дискеты на нее можно записать или считать с нее информацию с помощью магнитной головки. Перемещение (позиционирование) магнитной головки осуществляет шаговый двигатель. Шаговый двигатель можно представить в виде ротора с четырьмя устойчивыми состояниями по углу поворота, присоединенного через редуктор к магнитной головке. Включая одну из фаз (φ0 - φ3), можно повернуть ротор в соответствующее положение. Целесобразным является только последовательное включение фаз. Прямая последовательность φ0, φ1, φ2, φ3, φ0,... перемещает головку к центру диска; обратная последовательность φ0, φ3, φ2, φ1, ... - к внешнему краю.

Перемещение головки на одну дорожку осуществляется переключением фаз и остановкой в несоседней фазе, т.е. φ0 → φ1→ φ2 или φ2 → φ3 → φ0 и т.д.

После включения питания ДОС не имеет информации о текущем положении головки относительно поверхности мини-диска. Поэтому требуется возвращение головки в исходное положение (дорожка 00), т.е. выполняется операция ВОЗВРАЩЕНИЕ К ДОРОЖКЕ 00 (RECALIBRATE). Операцию реализует ДОС генерацией 80 импульсов в направлении перемещения "назад". Такое число импульсов

гарантирует возвращение головки в крайнее (исходное) положение, которое обозначено как дорожка 00. После возвращения к дорожке 00 шаговый двигатель останавливается с включенной фазой φ0, что определяет положение последующих дорожек. Выполнив эту операцию, ДОС получает информацию о текущем положении головки, и позиционирование любой дорожки осуществляется вычислением разности между текущей и желаемой дорожкой. Эта разность преобразуется в число переключений фаз посредством выполнения инструкций типа:

LDA C080,X - выключение фазы;

LDA C081,X - включение следующей фазы.

При движении головки необходимо формировать задержки приблизительно 1 мс при проходе фазы и не менее 2 мс для фиксации головки в конечном положении.

После выбора соответствующего дискового устройства и установки (позиционирования) головки относительно желаемой дорожки ДОС получает возможность выполнять основные операции ЧТЕНИЕ и ЗАПИСЬ. Эти операции рассматриваются как макрооперации типа ЗАПИСЬ ПОЛЯ АДРЕСА; ЗАПИСЬ ПОЛЯ ДАННЫХ; ЧТЕНИЕ ПОЛЯ АДРЕСА; ЧТЕНИЕ ПОЛЯ ДАННЫХ.

Каждая из этих операций может рассматриваться как ЗАПИСЬ или ЧТЕНИЕ последовательности из определенного числа байтов, кодированных указанными методами. Работая под непосредственным управлением ДОС, контроллер выполняет только элементарные операции: считывание байта; проверка защиты; запись байта; ввод байта.

На каждом ГМД имеется вырез защиты записи. Он расположен на левом краю ГМД. Если вырез защиты записи заклеен, вырабатывается сигнал "Защита записи" (WRITE PROTECT), и запись на ГМД невозможна.

Контроллер работает либо в режиме чтения, либо в режиме записи информации. Режим чтения включается путем выполнения любой операции процессора с адресом o C0XE. После этого можно считывать байты с фиксированной дорожки. Затем контроллер воспринимает последовательные данные с диска и посредством регистра сдвига преобразует их в параллельный байт D7 - D0. ДОС следит за состоянием регистра посредством инструкции.

```
        LDA C08E, X
RDBYTE: LDA C08C, X
        BRL RDBYTE
```

Проверяется регистр чтения ¤ COXC до тех пор, пока старший бит по адресу ¤ COXC не станет равным единице. При установлении D7 в единицу считается, что очередной байт уже воспринят контроллером и ДОС его может обрабатывать.

Процесс преобразования последовательной информации в параллельную управляется микропрограммным автоматом контроллера НГМД. Он вырабатывает необходимые интервалы времени, на протяжении которых "захватываются" поступающие с дискового устройства импульсы. Если за 4 мкс на входной шине не появится импульс, то в регистр сдвига записывается нуль, а при появлении импульса - единица. После этого значение регистра чтения обнуляется, т.е. считывание байта из регистра чтения возможно только один раз.

Для примера предположим, что нам нужно считать три байта с фиксированной дорожки ГМД. Будем считать, что значение регистра X равно номеру разъема, к которому подключен контроллер, умноженному на 16 (¤ 10), т.е. в старшей тетраде регистра находится номер разъема, а в младшей - 0:

X = N × 16 или N × ⌘ 10.

Программа имеет вид:

```
        LDY     ⌘ 00
        LDA     ⌘ C08E,X ;   включить режим чтения
READ:   LDA     ⌘ C08C,X ;   считать байт
        BRL READ         ;   если D7 ≠ 1
        STA TABL,Y       ;   поместить байт в таблицу
        INY
        CPY ≠  03
        BNE READ         ; считывание следующего байта
        RTS
```

Режим записи включается выполнением любой операции процессора с ячейкой ввода-вывода: ⌘ COXF. Запись байта данных возможна только через 100 мкс после включения режима записи. Чтобы записать один байт данных на ГМД, необходимо записать байт данных в регистр записи (⌘ COXD) и выполнить любую операцию процессора с адресом COXE. Записать байт данных на ГМД можно, например, так:

```
LDA     TABL
STA     ⌘ C08D, X ; запись в регистр записи
ORA     ⌘ C08F, x ; запись на ГМД из регистра записи
```

После выполнения любой операции с адресом COXF значение регистра записи не меняется, т.е. одно и то же значение можно записывать последовательно на ГМД, не загружая его предварительно в регистр записи. При записи нескольких байтов программа должна обеспечивать интервал 32 мкс между обращениями к ⌘ CO8F.

Процесс записи четко ограничен во времени, и каждый байт должен быть записан за 32 мкс. Микропрограммный автомат фиксирует интервалы по 4 мкс каждый, за время которых записывается соответствующий бит (0 или 1).

Рассмотрим, например, каким образом ДОС реализует операцию ЗАПИСЬ В ПОЛЕ АДРЕСА - ДАННЫХ.

Программа, используемая ДОС, выполняется за 32 мкс (между обращениями к ячейке ⌘ CO8F):

Время, мкс	*Код операции*		*Операция*
-	LDA	CO8F , X	Включение режима записи
2 } 8	LDA	⌘ DATA 1	Запись в аккумулятор
6 }	JSR	WRUTE	первого байта данных
2 } 8	LDA	⌘ DATA 2	Запись второго байта данных
6 }	JSR	WRITE	
2	WRITE: CLC		Сброс флажка С
3	PHA		
4 } 24	PLA		
5	STA	⌘ C08D, X	Ввод байта в регистр
4	ORA	CO8C, X	Запись байта
6	RET		

Чтобы считать сигнал WRITE PROTECT, необходимо выполнить любую операцию процессора с адресом o C0XD и считать значение по адресу o C0XE. Если старший разряд считанного значения равен единице, то запись невозможна. Пример программы:

```
LDA     )( C08D, X; X = N * 16
LDA     )( C08E, X; считывание сигнала
BMI     WRERR     ; запись невозможна
```

На контроллере НГМД в ПЗУ записана специальная программа - драйвер. ПЗУ подключено к адресным шинам таким образом, что эта программа занимает адреса CN00 - CNFF, где N = 3 (номер разъема, в котором размещен контроллер).

После включения компьютера управление передается на программу-загрузчик, (входящую в состав драйвера НГМД), которая осуществляет запись в память машины (начиная с адреса)(800) информации, считываемой с нулевого сектора нулевой дорожки магнитного диска (как правило, это программа загрузки ДОС). Драйвер записан в ПЗУ котроллера НГМД.

ДОС, если она есть на ГМД, расположена на дорожках 0 - 2.

Перед записью в память осуществляется перекодировка прочитанных с диска данных.

В дальнейшем драйвер работает под управлением ДОС, выполняя макрокоманды записи-считывания полей АДРЕСА - ДАННЫХ. Распечатать программу-драйвер можно в мониторе с помощью команды L: * C300 LLLLLL.

ОРГАНИЗАЦИЯ БАЗ ДАННЫХ

Организация баз данных сводится к написанию программ, обеспечивающих хранение, пополнение и корректировку информации на ГМД. Такие программы не должны зависеть от типа и объема данных, т.е. должны составить стандартный набор (пакет).

Рассмотрим возможности, которые предоставляет пользователю ДОС для ведения баз данных.

В состав ДОС входят операторы, работающие с текстовыми файлами.

На рис. 6.12 приведена программа, создающая текстовый файл последовательной структуры и записывающая в него данные, вводимые непосредственно с клавиатуры. Оператор INPUT в строке 10 в начале работы программы выводит первый вопрос. В ответ с блока клавиатуры вводится последовательность символов - имя создаваемого файла. Следующий оператор INPUT в строке 20 запрашивает ввод данных с клавиатуры. Информация запоминается в символьной переменной A)(. Операторы в строках 30 и 40 создают текстовой файл с именем, указанным в переменной NAME)(, и переводят его в режим записи. Следующий оператор в строке 50 записывает содержимое переменной A)(в созданный файл.

На рис. 6.13 представлена программа, осуществляющая обратные действия: чтение данных из последовательного текстового файла. Первый оператор в строке 10 выводит звуковой сигнал, второй распечатывает на экране ВКУ каталог ГМД, вставленный (в данный момент) в накопитель. Файл с указанным именем открывается по чтению операторами в строке 30; чтение при этом начнется с первого байта данных.

```
 5   REM ?-ИМЯ ТЕКСТОВОГО ФАЙЛА
 6   REM ??-ЧТО ПИСАТЬ,ВВОД
     С КЛАВИАТУРЫ
10   INPUT NAME¤
20   INPUT A¤
30   PRINT CHR¤(4);''OPEN'';
     NAME¤
40   PRINT CHR¤(4);''WRITE'';
     NAME¤
50   PRINT A¤
60   PRINT CHR¤(4);''CLOSE'';
     NAME¤
```

Рис. 6.12. Программа создания последовательного файла и запись в него

```
 5   REM ?-ИМЯ ФАЙЛА
10   PRINT CHR¤(7) : PRINT CHR¤
     (4);''CATALOG''
20   INPUT NAME¤ : PRINT
30   PRINT CHR¤(4);''OPEN'';NAME¤
   : PRINT CHR¤(4);''READ'';NAME¤
40   N=0 : ONERR GOTO 90
50   GET A¤
60   IF ASC(A¤)<32 THEN 80
70   PRINT  '' '';A¤; : N=N+1 :
     GOTO 50
80   RIBBON=1 : PRINT '' '';
   : RIBBON=7 : N=N+1 : GOTO 50
90   PRINT CHR¤(7);PRINT CHR¤(7) :
     CHR¤(ASC(A¤)+64); :
     ***************'':
     PRINT '' *****************
     PRINT ''EOF '';N;'' SYM.''
```

Рис. 6.13. Программа информации данных из последовательного файла

Оператор **GET A** ¤ в строке 50 считывает один символ, распечатываемый затем операторами в строке 70. Чтение осуществляется до тех пор, пока файл не иссякнет (второй оператор в строке 40).

Для дальнейшего понимания излагаемого материала введем понятие записи. ***Запись*** - это совокупность данных, записываемая в файл одним оператором **PRINT**. В одной записи могут находиться несколько переменных (т.е. в один оператор **PRINT** можно включить любое число переменных). Признаком конца записи может служить код ⌐ (число 13).

В программе предусмотрен поиск кода ⌐ (конца записи) и печать его на экране в виде красной буквы М.

Отметим, что в операторе **INPUT** код клавиши ⌐ служит признаком конца читаемой записи, т.е. за одно обращение **INPUT** читает константы от начала очередной записи до конца ⌐ включительно.

Примером использования этого оператора служит программа, приведенная на рис. 6.14.

Особенность последовательного файла заключается в том, что записи создаются последовательно. Нельзя прочитать вторую запись, не прочитав первую, и т.д.

Во всех рассмотренных выше программах оператор **OPEN** позиционирует головку чтения-записи на начало файла. Таким образом, чтобы записать в файл дополнительную информацию, необходимо повторить все предыдущие записи. Это неудобно, а иногда просто невозможно. Применение оператора **APPEND** дает возможность исключить лишние команды записи при дописывании информации в

```
 5   ?-ИМЯ ФАЙЛА
10   PRINT CHR¤(7) : PRINT CHR¤
     (4);''CATALOG''
20   INPUT NAME¤ : PRINT
30   PRINT CHR¤(4);''OPEN'';
     NAME¤
   : PRINT CHR¤(4);''READ'';
     NAME¤
40   N=0 : ONERR GOTO 70
50   INPUT A¤
60   PRINT A¤ : N=N+1 : GO TO 50
70   PRINT CHR¤(7) : PRINT
     CHR¤(7)
   : PRINT '' ***************
     *******'';
   : PRINT '' ************''
   : PRINT '' EOF'';N;''LIN.''
```

Рис. 6.14. Печать последовательного файла

```
10   REM ?-ИМЯ ФАЙЛА
20   REM ??-ДОБАВЛЯЕМАЯ В ФАЙЛ
     ИНФОРМАЦИЯ
30   INPUT NAME¤
40   INPUT A¤
50   PRINT CHR¤(4);''APPEND'';
     NAME¤
60   PRINT CHR¤(4);''WRITE'';
     NAME¤
70   PRINT A¤
80   PRINT CHR¤(4);''CLOSE'';
     NAME¤
```

Рис. 6.15. Программа добавления к файлу информации

текстовой файл (рис.6.15). Пропустить ненужные записи можно и оператором **POSITION.**

Конечно, все рассмотренные программы систему базы данных не составляют. Для разработки подобной системы необходимо реализовать следующие требования:

создать последовательный файл, содержащий номера логических записей и ссылки, где они находятся на ГМД;

логические записи помещать в одну или несколько физических записей в файле с прямым доступом.

Примером реализации этих требований служат программы, представленные на рис. 6.16 и 6.17.

```
10   REM  СОЗДАНИЕ
     ПОСЛЕДОВАТЕЛЬНОГО
20   REM  ФАИЛА КЛЮЧЕЙ
30   DIM  K(400) : CLEAR
40   PRINT CHR¤(4);''OPEN KLY''
50   PRINT CHR¤(4);''WRITE KLY''
60   FOR I=1 TO 400 : PRINT K(I):
     NEXT
70   PRINT CHR¤(4);''CLOSE KLY''
80   REM   ФАЙЛ МОЖЕТ
     СОДЕРЖАТЬ НЕ
90   REM   МЕНИЕ 400 КЛЮЧЕЙ
100  REM   НОМЕР КЛЮЧА - НОМЕР
     ЗАПИСИ
110  REM   В ПРЯМОМ ФАЙЛЕ ДАННЫХ
```

Рис. 6.16. Создание последовательного файла ключей

```
10   REM   СОЗДАНИЕ ПРЯМОГО
     ФАЙЛА,
20   REM   СОДЕРЖАЩЕГО ДАННЫЕ.
25   DIM  A¤(400) : CLEAR
30   PRINT CHR¤(4);''OPEN ZAP,
     L255''
40   FOR K=1 TO 400
50   PRINT CHR¤(4);''WRITE
     ZAP,R''K
60   PRINT  A¤(K) : NEXT
70   PRINT CHR¤(4);''CLOSE
     ZAP''
80   REM  НОМЕР ЗАПИСИ
     СОВПАДАЕТ С
90   REM  НОМЕРОМ КЛЮЧА
     В ФАЙЛЕ KLY
```

Рис. 6.17. Создание прямого файла данных

Программы формируют два текстовых файла. Один из них последовательный, другой - прямой. В последовательный файл записываются ключи - числовые величины. Номеру записи в последовательном файле соответствует номер записи в файле с прямым доступом. Таким образом, в последовательный файл помещаются ключи, а в файл с прямым доступом - данные, которые эти ключи идентифицируют. Число ключей в одной ключевой записи можно увеличить. Такое разбиение данных на ключевые и информационные делает программу более рациональной и универсальной.

В прил. 2 приведена программа "Личный секретарь", работающая с текстовым файлами.

6.5. ПРОГРАММИРОВАНИЕ ПОСЛЕДОВАТЕЛЬНОГО И ПАРАЛЛЕЛЬНОГО ИНТЕРФЕЙСА (ППИ)

Модуль ППИ позволяет управлять и обмениваться информацией с периферийными устройствами, печатающим устройством, графическим дисплеем, графопостроителем, алфавитно-цифровым дисплеем, модемом телефонного канала, устройством ввода с голоса, оптическими устройствами ввода-вывода и т.д.

При подключении перечисленных устройств к ПЭВМ "Агат" необходимо пользоваться таблицами назначения контактов выходного разъема ППИ.

Функционально модуль ППИ состоит из параллельного и последовательного интерфейса.

ПРОГРАММИРОВАНИЕ ПАРАЛЛЕЛЬНОГО ИНТЕРФЕЙСА

В адресном пространстве ячеек ввода-вывода для управления программируемым параллельным интерфейсом (ППРИ) отведены ячейки ввода-вывода с адресами **COY0-COY7**; при этом **Y** - шестнадцатеричная цифра, которая вычисляется по формуле:

$$Y = 8 + N,$$

где **N** - номер разъема, к которому подключен модуль последовательного и параллельного интерфейса.

Таким образом, в младшей тетраде адреса разряд A3 всегда равен нулю (буквой A и цифрой за ней будем обозначать разряды адреса). Значение разряда A2 не является существенным. Разряды A1 и A0 используются для адресных портов.

В ППРИ имеется 24 линии (разряда) ввода-вывода. Эти 24 линии делятся на две группы (A и B) по 12 линий в каждой. Каждая группа может быть использована в одном из трех режимов.

В ППРИ имеется три порта (A, B и C) по 8 разрядов каждый. Порт A и четыре старших разряда порта C составляют группу A. Порт B и четыре младших разряда порта C составляют группу B.

Для управления работой ППРИ используется специальный регистр R управления.

Адресация портов в ППРИ:

A1	A0	Порт
0	0	A
0	1	B
1	0	C
1	1	R

В регистр управления данные можно только записывать. Чтение из регистра управления невозможно.

Чтение и запись в порты осуществляется командами чтения и записи процессора по адресам соответствующих ячеек ввода-вывода.

ППРИ может работать в одном из следующих трех режимов:

режим "0" - основной режим ввода-вывода;

режим "1" - режим стробированного ввода-вывода;

режим "2" - режим двунаправленной передачи.

Для задания нужного режима работы ППРИ необходимо в регистр управления записать управляющее слово режима. Формат управляющего слова режима приведен ниже. Разряды данных обозначаются D0, D1 и т.д.

Формат управляющего слова:

D0 - порт C (младшие разряды): 1 - ввод, 0 -вывод;

D1 - порт B: 1 - ввод, 0 - вывод;

D2 - режим группы B: 0 - режим "0", 1 - режим "1";

D3 - порт C (старшие разряды): 1 - ввод, 0 - вывод;

D4 - порт A: 1 - ввод, 0 - вывод.

Разряды D5, D6 определяют режимы работы ППРИ:

00 = режим "0";
01 = режим "1";
10 = режим "2";
11 = режим "2"

Разряд D7 = 1.

Режим "0". В этом режиме данные считываются или записываются в адресуемый порт. Управляющие сигналы между абонентами при этом не используются. Функционально в ППРИ использовать два восьмиразрядных и два четырехразрядных порта, любой из которых может быть вводным или выводным. В управляющих словах D2=0, D5=0, D6=0, D7=1. Остальные разряды определяют работу портов в режиме "0" в соответствии с приведенным выше форматом управляющего слова.

Режим "1". Этот режим обеспечивает передачу данных между периферийным устройством и адресуемым портом в соответствии с управляющими сигналами взаимодействия абонентов. В режиме "1" порты A и B используют соответствующие четырехразрядные доли порта C для передачи или приема этих сигналов. Таким образом, в режиме "1" используются группы A и B, причем каждая группа состоит из восьмиразрядного порта данных и четырехразрядного порта управляющих сигналов.

Каждый из портов может быть запрограммирован либо как вводной, либо как выводной.

Сигналы управления в режиме "1" (ввод):

1) строб A - ПС4 (четвертый разряд порта C),

строб B - ПС2.

Наличие нуля в этих разрядах означает, что данные загружены во входные регистры (порт А или порт В);

2) Вх БПА - ПС5,

Вх БПВ - ПС1.

Выходные сигналы Вх БПА, Вх БПВ (входной буфер полон) вырабатываются при нулевом стробирующем сигнале и сбрасываются после считывания данных из порта процессором. Используются сигналы Вх БПА, Вх БПВ в качестве сигнала подтверждения приема данных;

3) ПРА - ПС3,

ПРВ - ПС0.

Выходные сигналы ПРА и ПРВ (прерывания А и В) используются в качестве сигналов запроса прерывания микропроцессора. Эти сигналы вырабатываются при условии, если строб=1, Вх БП=1, INTE=1, и сбрасываются после считывания данных из соответствующего порта процессором;

4) INTEA - ПС4,

INTEB - ПС2.

Единица в соответствующем разряде ПС4 или ПС2 (маска прерывания А,В) позволяет вырабатывать сигналы прерывания от порта А или порта В.

Сигналы управления в режиме "1" (вывод):

1) Вых БПА - ПС7,

Вых БПВ - ПС1.

Наличие нуля в соответствующем разряде входного сигнала указывает, что процессор произвел запись в выбранный порт. Сигнал устанавливается в результате записи в порт и сбрасывается сигналом "подтверждение";

2) Подтв. А - ПС6,

Подтв. В - ПС2.

Входной сигнал "подтверждение" извещает ППРИ, что данные получены устройством;

3) ПРА - ПС3,

ПРВ - ПС0.

Назначение сигналов то же, что и в режиме "1" (ввод). Установка в единицу ПРА, ПРВ происходит при условии, если Подтв.=1, INTE=1.

Сброс ПРА, ПРВ происходит после записи данных в порты А, В;

4) INTE A - ПС6,

INTE B - ПС2.

Назначение сигналов то же, что и в режиме "1" (ввод).

Режим "2". Это режим стробируемого двунаправленного обмена.

Режим "2", который может быть использован только с портом группы А, обеспечивает возможность приема и передачи данных по одним и тем же восьми разрядам в режиме двунаправленной шины.

Управляющие сигналы, обеспечивающие взаимодействие абонентов при обмене, генерируются и воспринимаются пятью разрядами порта С.

Сигналы управления в режиме "2":

1) ЗПРА - ПС3 (выходной сигнал запроса прерывания)

2) Вых БПА - ПС7 (выходной сигнал буфер А полон)

3) Подтв.А - ПС6 (входной сигнал подтверждения порта А)

4) INTE 1 - ПС6. Маска прерывания по сигналу Вых БПА. Управляет непосредственной записью в даный разряд.

5) Строб А - ПС4

6) Вх БПА - ПС5

7) INTE 2 - ПС4. Маска прерывания по сигналу Вх БПА управляется записью в соответствующий разряд.

Функциональное назначение сигналов в режиме "2" полностью соответствует их назначению в режиме "1", за исключением сигнала Подтв.А. Наличие нулевого сигнала Подтв.А разрешает передачу данных из ППРИ к внешнему устройству.

В режиме "2" пять старших разрядов порта С используются для генерации и приема управляющих сигналов порта А, а три младших разряда (ПС0-ПС2) могут быть запрограммированы как вводные или выводные, если порт В запрограммирован для работы в режиме "0".

Если порт В запрограммирован для работы в режиме "1", разряды ПС0-ПС2 используются для генерации и приема управляющих сигналов, соответствующих режиму "1".

В режимах "1" и "2" имеется возможность считывать слово состояния, определяющее статус обмена. Считывание осуществляется операцией чтения из порта С.

Формат слова состояния в режимах "1" и "2" приведен соответственно в табл. 6.3 и 6.4.

Таблица 6.3

Слово состояния ППРИ в режиме "1"

Наименование группы	Разряд	Назначение разряда	
		при вводе	при выводе
	D0	ПРВ	ПРВ
	D1	Вх БПВ	Вых БПВ
Группа В	D2	INTE B	INTE B
	D3	ПРА	ПРА
	D4	INTE A	Ввод-вывод
Группа А	D5	Вх БПА	То же
	D6	Ввод-вывод	INTE A
	D7	то же	Вых БПА

Таблица 6.4

Слово состояния ППРИ в режиме "2"

Наименование группы	Разряд	Назначение разряда
Группа В	D0-D2	Зависит от выбора режима ("0" или "1")
Группа А	D3	ЗПРА
	D4	Вх БПА
	D5	Вых БПА
	D6	INTE 1
	D7	Вых БПА

В режиме "1" (ввод) разряды ПС6 и ПС7 группы А могут быть запрограммированы независимо друг от друга для использования в качестве вводных или выводных. Программирование этих разрядов зависит от разряда D3 управляющего слова, записанного в регистр управления. В режиме "1" (вывод) оставшиеся свободными разряды группы А также можно запрограммировать как вводные или выводные.

ПРОГРАММИРОВАНИЕ ПОСЛЕДОВАТЕЛЬНОГО ИНТЕРФЕЙСА

Под управление программируемым последовательным интерфейсом (ППСИ) отве дены ячейки ввода-вывода COY8 - COY9, разряд адреса A3 всегда должен содержать единицу. Разряд адреса A0 служит для адресования к регистру команд или к регистру данных. Значение разрядов адресов A2 и A1 несущественно.

Наличие единицы в разряде A0 служит для адресования к регистру команд; наличие нуля - для адресования к регистру данных. Для простоты можно считать, что адрес регистра команд - COY9, а регистра данных - COY8. Если модуль параллельного и последовательного интерфейса подключен к разъему 5, адрес регистра данных COD8, а регистра команд - COD9.

Обмен с внешним устройством осуществляется простым считыванием или записью данных в регистр данных. Команды задаются записью в регистр команд. Слово состояния считывается из регистра команд. ППСИ получает от процессора данные в параллельной форме (побайтно), преобразует их и передает внешнему устройству в виде последовательного потока бит. ППСИ может принимать от внешнего устройства данные в последовательном формате и через регистр данных передавать их процессору в виде байтов.

Команды ППСИ делятся на два вида: команды выбора режима и управляющие команды.

Команда выбора режима записывается в регистр комад, после чего ППСИ воспринимает команды как управляющие.

Следующий режим может быть задан только после внутреннего (задается управляющей командой) или внешнего (осуществляется за счет сигнала RESET) сброса.

ППСИ может работать в двух режимах (синхронном или асинхронном) на прием или на передачу данных внешнему устройству. Модуль ППСИ, установленный в "Агате", выполняет все функции синхронного режима и частично асинхронного. Для организации полного обмена в асинхронном режиме требуется выполнить доработку модуля для формирования сигналов DTR, DSR, CTS, RTS, TXC, RXC.

В асинхронном режиме на передачу с ППСИ нужно работать по следующему алгоритму:

1) осуществить сброс (внутренний или внешний);
2) задать режим (записать команду режима в регистр команд);
3) задать управляющую команду (записать команду в регистр команд);
4) передать байт данных (записать байт в регистр данных);
5) считать слово состояния из регистра команд;
6) в зависимости от слова состояния перейти к выполнению алгоритма начиная с одного из пп. 3 - 5, 7;
7) в случае сброса ППСИ прекращает работу в данном режиме и ожидает задания нового режима.

В асинхронном режиме на прием алгоритм работы следующий:

1) осуществить сброс;
2) задать режим;
3) задать управляющую команду;
4) считать слово состояния из регистра команд;

5) в зависимости от слова состояния перейти к выполнению алгоритма начиная с одного из пп. 3, 4, 6, 7;

6) осуществить прием байта от внешнего устройства (считать байт из регистра данных);

7) в случае сброса ППСИ прекращает работу в данном режиме и ожидает команду выбора режима.

Алгоритм работы ППСИ в синхронном режиме аналогичен алгоритму работы в асинхронном режиме. Отличие состоит в том, что после команды выбора режима необходимо загрузить символы синхронизации в регистр команд.

Выбор режима работы зависит от самого внешнего устройства.

Формат комады выбора режима. ***Синхронный режим:***

D0, D1 - определяют выбор режима (код 00);

D2, D3 - служат для кодирования длины символов (длина символа, бит: код 00 - 5, код 10 - 6, 01 - 7, код 11 - 8);

D4 - предназначен для формирования бита четности (код 1 - бит четности передается, код 0 - бит четности отсутствует);

D5 - предназначен для контроля переданного байта на четность в соответствии с битом четности (код 1 - контроль, код 0 - контроль отсутствует);

D6 - определяет вид синхронизации (код 1 - внешняя синхронизация, код 0 - внутренняя синхронизация);

D7 - определяет число синхросимволов (код 0 - два синхросимвола, код 1 - код синхросимвол).

Асинхронный режим:

D0, D1 - определяют скорость обмена. При коде 10 скорость обмена равна частоте тактового генератора. При коде 01 скорость обмена равна 1/16 частоты тактового генератора; при коде 11 скорость обмена равна 1/64 частоты тактового генератора;

D2 - D5 - имеют такое же назначение, что и соответствующие разряды в синхронном режиме;

D6, D7 - задают число стоп-бит (код 00 - стоп-битов нет, код 10 - 1 стоп-бит, код 01 - 1,5 стоп-бита, код 11 - 2 стоп-бита).

После выбора режима необходимо ввести символы синхронизации, если режим синхронный, и управляющую команду. Управляющая команда реализует следующие функции: разрешение приема передачи, сброс ошибки и управление модемом.

Формат управляющей команды. Назначение разрядов:

D0, сигнал TxEN - определяет разрешение (запрещение) передачи (1 - разрешение, 0 - запрещение);

D1, сигнал DTR - запрашивает (при D I=I) готовность внешнего устройства к приему информации;

D2, сигнал RxE - определяет разрешение (запрещение) приема (1 - разрешение, 0 - запрещение);

D3 - управляет прерыванием (0 - отсутствие прерывания, 1 - формирование запроса на прерывание устройства - приемника данных);

D4 - сброс ошибки, сброс флажков ошибки PE, OE, FE;

D5 - запрос на передачу;

D6 - внутренний сброс (1 - возвращает состояние ППСИ в режим ожидания команды выбора режима);

D7 - режим ожидания символов синхронизации.

Назначение сигналов ППСИ. ППСИ формирует следующие команды управления:

TxRDY - канал запрашивает символ данных для передачи (передача предыдущего символа, возможно, продолжается);

TxEMPTY - канал пуст, готов к работе на передачу;

DSR - канал принял порцию данных от внешнего устройства;

DTR - готовность данных к передаче (запрос порции данных у внешнего устройства передатчика);

CTS - разрешение передачи, сигнал приходит от внешнего устройства приемника и позволяет передавать последние данные;

TxC - управление скоростью передачи;

SYNDET - данный сигнал в режиме приема высоким уровнем сообщает, что ППСИ принял символ синхронизации;

RxRDY - канал принял от внешнего передатчика порцию данных, которая готова к считыванию из регистра данных процессором.

ФОРМАТ СЧИТЫВАЕМОГО СЛОВА СОСТОЯНИЯ

Назначение разрядов:

D0, сигнал TxRDY - буфер шины данных пуст. Вырабатывается при CTS=0, TxEN=1;

D1, D2, D6 - соответствуют сигналам ППСИ RxRDY, TxEMPTY, SYNDET;

D2, флажок PE - ошибка четности. Устанавливается при обнаружении ошибки четности и не запрещает работы ППСИ;

D4, флажок OE - ошибка перебегания. Устанавливается в том случае, если процессор не считал символ до поступления нового, и не запрещает работы ППСИ;

D5, флажок FE - ошибка формы. Устанавливается в том случае, если в конце каждого символа не будет обнаружен запрограммированный стоп-бит;

D7, сигнал DSR - набор даных готов.

Глава 7. ЭЛЕМЕНТНАЯ БАЗА ПЭВМ

7.1. ДИСКРЕТНЫЕ ЭЛЕМЕНТЫ

ПЭВМ "Агат" относится к ЭВМ четвертого поколения,использующих большие и средние интегральные схемы.

Узлы и блоки компьютера в основном построены на логических элементах, триггерах, счетчиках, регистрах [3, 4].

Триггерные устройства разделяются по виду логического функционирования, способу записи информации, числу ступеней построения.

По способу записи информации триггеры делятся на ***несинхронные (асинхронные)*** и ***синхронные***. Последние, в отличие от первых, имеют специальный синхронизированный С-вход, сигнал которого разрешает триггеру принять новую информацию. Этот сигнал называют тактирующим или командным.

По числу ступеней различают ***одноступенчатые*** и ***двухступенчатые*** триггерные

устройства. Двухступенчатый триггер позволяет получить эффект задержки информации. Такие триггеры называют также *MS*-триггерами, так как одна из ступеней (*slave*) повторяет состояние другой ступени (*master*).

Особенно широко применяют триггеры с разным логическим функционированием: ***с установочным запуском*** (типа RS), с ***задержкой*** (типа D), ***универсальные*** (типа JK, V, T) и др.

Регистром называется устройство, предназначенное для приема, хранения и передачи двоичных чисел. Каждому разряду числа соответствует разряд регистра, выполненный на основе триггерной схемы. Регистры позволяют производить логическое сложение и умножение, сдвиг числа на определенное число разрядов, а также преобразование параллельного кода в последовательный, и наоборот.

Основными видами регистров являются ***параллельные***, предназначенные для записи и хранения числа в двоичном параллельном коде, и ***последовательные*** (регистры сдвига).

Регистр состоит из нескольких триггеров, число которых соответствует числу разрядов кода числа. В регистрах сдвига добавлены управляющие цепи связи старших разрядов с младшими. Подачей соответствующего управляющего импульса регистр позволяет сдвигать код числа влево или вправо. Регистр, осуществляющий сдвиг кода числа влево и вправо, называется ***реверсивным***.

Счетчик импульсов - устройство, предназначенное для подсчета числа импульсов, поступивших на его вход. В ПЭВМ применяются суммирующие, вычитающие и реверсивные (т.е. выполняющие и сложение, и вычитание) счетчики. По виду связи между разрядами различают счетчики с непосредственными связями, переносом и комбинированными связями. По коэффициенту счета счетчики бывают двоичные и с произвольным коэффициентом счета. Наибольшее распространение получили двоичные счетчики, которые состоят из последовательно соединенных триггеров, работающих в режиме счета. Быстродействие счетчика характеризуется временем переключения и максимальной частотой поступления счетных импульсов.

В п. 7.5 описаны микросхемы серий 155 и 555, реализующие функции триггеров, счетчиков и регистров, составляющих основу построения компьютера.

7.2. ШИННЫЕ ФОРМИРОВАТЕЛИ

Назначение шинных формирователей. Шинный формирователь представляет собой усилитель, сопрягающий выходы микропроцессора с магистралями передачи информации компьютера. Вследствие этого шины адреса и данных можно непосредственно подключать к выводам БИС микропроцессора, а все остальные элементы компьютера (память, порты ввода-вывода) подключать сразу же к соответствующим шинам. Система управления ЭВМ в этом случае имеет небольшое число элементов и получается очень простой.

Объем вычислительной системы является тем ограничивающим фактором, который связан с нагрузочной способностью выходных буферов микропроцессора и подсоединенных к ним шин. Нагрузочная способность шины данных для состояния логического нуля составляет 1,8 мА, а в состоянии логической единицы - 0,15 мА. Нагрузочная способность всех остальных выводов составляет 0,75 мА

для логического нуля и только 0,1 мА для логической единицы. Эти значения гарантированы разработчиками микропроцессорного комплекса, и, хотя на практике их фактически завышают, все же они указывают те границы, которых следует придерживаться при эксплуатации компьютера.

Если суммарная нагрузка какой-либо линии шины адреса или шины данных будет превосходить указанные выше значения, то в такую шину надо вводить буфера-усилители, которые называют ***шинными формирователями***.

При проектировании ПЭВМ "Агат" была оценена токовая нагрузка, которую будет испытывать каждая линия шин адреса и данных. Адресные линии в микропроцессорной системе управления связаны со множеством входов различных модулей, подключенных параллельно. Модули памяти обычно потребляют ток очень малой силы (0,02 - 0,4 мА), но в состав системы входят те или иные логические устройства, для реализации которых применяются интегральные схемы серии К155. Сила тока, потребляемого на входах этих микросхем, составляет 1,6 мА в состоянии логического нуля и 0,04 мА в состоянии логической единицы, т.е. практически полностью расходуется токовый ресурс микропроцессора 6502.

Чтобы сократить потребление тока в сигнальных линиях и тем самым избежать введения буферных усилителей-формирователей, логические преобразования выполнены на интегральных схемах серии К555. Эти схемы потребляют токи меньшей силы: в состоянии логического нуля - 0,36 мА, а в состоянии логической единицы - 0,02 мА. Таким образом, микропроцессор 6502 может непосредственно (без применения буферов) выдержать нагрузку пяти интегральных схем серии К555.

В общем случае нагрузка по току на адресных линиях может превысить возможности микропроцессора, тогда возникает необходимость во введении шинных формирователей. Введение буфера адреса требуется и потому, что адресные линии работают на значительную емкостную нагрузку, например, на внешний интерфейс. Для поддержания электрических характеристик сигнала на конце линии его необходимо усиливать перед передачей по интерфейсу, т.е. применять шинный формирователь.

Включение буфера в шину данных. Когда микропроцессор работает в режиме вывода, к шине данных параллельно подключается много различных модулей. При этом может оказаться, что общая нагрузка от всех устройств будет для шины данных чрезмерной. Эта ситуация подобна той, которая обсуждалась при рассмотрении нагрузки на шину адреса, и для ее предотвращения нужно также вводить буферформирователь.

Однако, рассматривая шину данных, необходимо учитывать то обстоятельство, что в отличие от адресной шины, в которой сигналы проходят только в одном направлении (от микропроцессора к другим модулям системы), шина данных двунаправленная.

Для использования в качестве двунаправленного буфера создана специальная интегральная схема К589АП16 [3, 4] - четырехразрядный сдвоенный шинный формирователь (см. п. 7.8). Эта схема состоит из набора усилителей с тремя состояниями на выходе и логических схем на входе для управления направлением передачи (***ВН*** - выбор направления) и организации доступа. Отличительной особенностью этих микросхем является малая сила потребляемого тока на входе (0,25 мА) и высокая нагрузочная способность: первая группа усилителей дает

50 мА при нулевом и 10 мА при единичном уровне, а вторая группа - 15 и 1 мА соответственно. При использовании интегральной схемы К589АП16 в качестве двунаправленного буфера шины данных выводы ***A*** и ***C*** соединяются, но включенным всякий раз может быть только один из усилителей, так как выход другого переводится в высокоомное состояние.

При подаче на вход ***ВН*** логической единицы информация от микропроцессора передается к другим модулям системы (запись), а при подаче логического нуля информация передается от модулей к микропроцессору (чтение). В третье (высокоомное) состояние шинные формирователи переводятся путем подачи на вход ***ВМ*** уровня логической единицы. В реальной системе этот сигнал может быть взят с выхода $\overline{DMA}$ интерфейса. Если режим передачи управления в системе не используется, то на вход ***ВМ*** нужно постоянно подавать уровень логического нуля, соединив, например, вывод ***1*** микросхемы с корпусом.

На практике шинные формирователи К589АП16 применяют в качестве буфера не только шины данных, но и шины адреса, хотя одно состояние при этом оказывается избыточным. Для обслуживания всей шины адреса требуется четыре микросхемы.

7.3. ДЕШИФРАТОРЫ

Дешифраторы адреса используются для реализации принципа адресации к соответствующим устройствам.

На вход дешифратора подаются сигналы с линий шины адреса, а на выходе образуются ***сигналы выбора*** (***селекции***). Эти сигналы имеют много разных наименований и обозначений. Довольно часто используется обозначение ***CS*** (***Chip Select*** - выбор кристалла), поэтому иногда используется его русский эквивалент ***ВК***. Часто применяется также обозначение ***ВМ*** (выбор микросхемы или выбор модуля). Кроме того, сигналы селекции называют также входом разрешения или входом доступа ***CE*** (***Chip Enable***), так как какая-либо операция в данной микросхеме может выполняться только тогда, когда на вход доступа поступает сигнал соответствующего уровня, т.е. логическая единица, если вход ***ВМ*** прямой, и логический нуль, если вход ***ВМ*** инверсный (кстати сказать, чаще используется именно инверсный вход). Такой вход доступа (выбора) имеют все микросхемы памяти, шинные формирователи и буферные регистры.

Многие микросхемы имеют не один, а несколько входов ***ВМ***, каждый из которых требует различных логических уровней разрешающих сигналов. Образно можно сказать, что микросхема имеет запирающий ее замок, ключом к которому является соответствующая кодовая комбинация на входах разрешения $ВМ_1$, $ВМ_2$ и т.д.

Схема селекции микросхем может быть собрана из отдельных логических элементов, но в ПЭВМ "Агат" эти схемы реализованы в виде микросхем среднего уровня интеграции К155ИД4 [3 - 5].

7.4. ОПЕРАТИВНЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Оперативные запоминающие устройства (ОЗУ) часто выполняются на МОП-транзисторах. Особенностью этих транзисторов является очень высокое входное сопротивление, что позволяет использовать в качестве элемента памяти кон-

денсатор, функции которого выполняет паразитная емкость. В результате получается так называемая ***динамическая память***.

Схема, иллюстрирующая принцип работы динамической памяти, приведена на рис. 7.1. Входной сигнал поступает на затвор транзистора через ключ. Конденсатор заряжается входным сигналом до напряжения этого сигнала. При размыкании ключа заряд на конденсаторе сохраняется, поддерживая транзистор либо в открытом, либо в закрытом состоянии в зависимости от логического уровня входного сигнала. Вследствие очень медленного разряда конденсатора через сопротивление затвора уровень входного сигнала сохраняется, т.е. информация "запоминается", до следующего замыкания ключа. Выходной сигнал снимается со стока МОП-транзистора, так что конденсатор практически никогда не шунтируется нагрузкой.

Однако практически нельзя сохранить информацию в течение длительного времени. Ключ представляет собой МОП-транзистор, имеющий некоторое конечное сопротивление, шунтирующее конденсатор. Изоляция затвора транзистора также обладает конечным сопротивлением. Поэтому максимальное время запоминания, или минимальная рабочая частота схемы, определяется емкостью конденсатора и этими двумя сопротивлениями (главным образом сопротивлением ключа).

На рис. 7.1 изображена реальная схема динамической ячейки памяти с запоминанием на кондесаторе. Такая ячейка управляется двухтактной серией синхронизирующих импульсов $\varphi 1$ и $\varphi 2$. Запоминающие конденсаторы в интегральной схеме специально не формируются, а в качестве их используются паразитные емкости. Предположим, что на вход схемы подан логический нуль. В течение действия импульса $\varphi 1$ транзисторы ***VT2*** и ***VT3*** открыты, а ***VT1*** закрыт. На выходе ***VT3*** потенциал будет близок к напряжению питания и зарядит конденсатор ***C***. В следующий период (во время действия импульса $\varphi 2$) транзистор ***VT4*** проводит ток, а транзистор ***VT5*** выполняет роль нагрузочного сопротивления. На выходе будет уровень логического нуля. Если же на вход подать логическую единицу, то транзистор ***VT1*** открывается, напряжение питания падает практически полностью на ***VT2***, и заряда конденсатора через ***VT3*** не происходит. В следующем такте, во время действия импульса $\varphi 2$, на выходе будет уровень напряжения питания, т.е. уровень логической единицы.

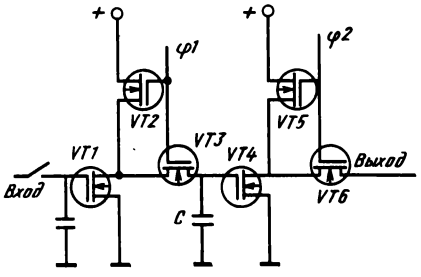


Рис. 7.1. Динамическая память

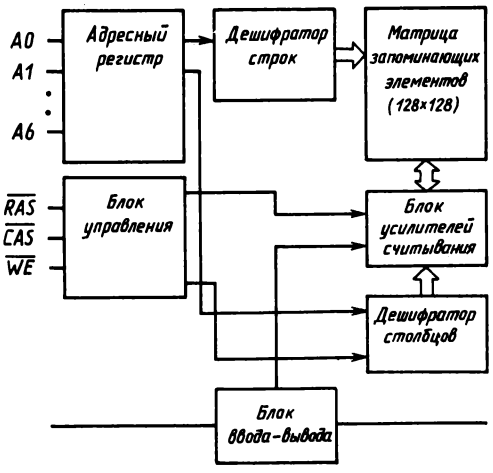


Рис. 7.2. БИС памяти емкостью 16 К бит с мультиплексированием выводов

Пользуясь схемами динамической логики, удается выполнить многие функции, применяя меньшее число компонентов по сравнению с обычными схемами. В результате этого для реализации данной функции расходуется меньшая площадь кристалла, или на той же площади реализуется большее число функций.

Ниже приведены основные характеристики БИС ОЗУ динамического типа [3 - 5].

	К507РУ1	К535РУ3	К565РУ1	К565РУ3	К565РУ6
Емкость, бит...	1024	512	4К	16К	16К
Организация памяти	1024x1бит	64x8 бит	4096x1 бит	16384x1 бит	16384x1 бит
Время цикла выборки, ... мкс	0,7	0,4	0,4	0,4	0,4

Однако схемы динамической памяти обладают и определенными недостатками, прежде всего необходимостью периодической перезаписи информации. Обращение к матрице запоминающих элементов для записи или чтения данных вызывает подключение к усилителям считывания одной строки матрицы; при этом автоматически происходит подзаряд конденсаторов всех ячеек выбранной строки. Этот процесс называют ***регенерацией памяти***. Для предотвращения потери информации необходимо обращаться к каждой строке матрицы не реже чем через 2 мс. При выполнении микропроцессором реальной программы это условие не всегда соблюдается, так как к одним ячейкам обращаются часто, а к другим очень редко. Поэтому в состав ПЭВМ входит специальный блок регенерации памяти, автоматически выполняющий периодическую перезапись содержимого всех ячеек динамической памяти.

Рассмотрим мультиплексированные БИС памяти - микросхемы К565РУ6 [3, 4]. Несмотря на большую информационную емкость (16К бит), эта схема размещена в корпусе с 16 выводами, хотя только для обращения к 16К ячейкам памяти необходимо иметь 14-разрядный адрес. Поэтому в данной микросхеме имеется специальный адресный регистр (рис. 7.2), информация в который заносится в два приема:

сначала поступают адреса семи младших разрядов, а затем семи старших. Первые сопровождаются управляющим сигналом $\overline{RAS}$ (***Row Address Select*** - выбор строки), а вторые - сигналом $\overline{CAS}$ (***Column Address Select*** - выбор столбца). Запись информации в ячейку памяти происходит с вывода ***DI*** (***Data In***) в момент действия сигнала $\overline{CAS}$, если предварительно уже сделан выбор строки путем подачи сигнала $\overline{RAS}$. Одновременно должен действовать сигнал $\overline{WE}$ (***Write Enable***). Микросхемы динамической памяти в течение всего цикла записи характеризуются высоким выходным сопротивлением; таким образом, отпадает необходимость в специальном выходном буфере для связи с линиями шины данных. Считывание данных происходит также в момент действия сигнала выбора столбца $\overline{CAS}$ после предварительной установки сигнала выбора строки $\overline{RAS}$.

Для ввода-вывода данных корпус БИС должен содержать столько же выводов, сколько битов имеется в слове, хранимом в памяти. Если длина слова памяти равна машинному слову микропроцессора, то организация системы памяти значительно упрощается, так как каждый вывод корпуса подключается к соответ-

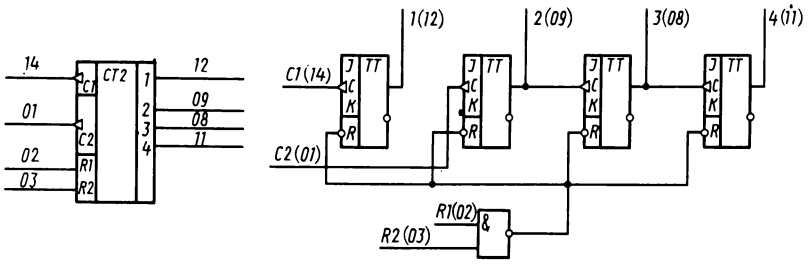
ствующей линии шины данных. Но подобная организация БИС памяти имеет место только в некоторых ПЗУ. Гораздо чаще организация микросхемы такова, что слово памяти и соответственно число выводов корпуса меньше, чем разрядность машинного слова. В частности, большинство БИС ОЗУ имеет организацию 256x1, 512x1, 1024x1 и т.д. Иными словами, обычно в каждой ячейке памяти хранится лишь один бит.

При однобитной организации БИС памяти необходимая длина машинного слова достигается параллельным включением соответствующего числа БИС. В микропроцессорах с байтовой (восьмиразрядной) организацией машинного слова нужно параллельно включить восемь одинаковых БИС памяти(К565РУ6 или К565РУ5). Адресные входы всех этих БИС подключаются к разрядам шины адреса через мультиплексор адреса. Все выводы ***ВМ*** и ***ЧТЗП*** подключаются также параллельно к соответствующим линиям шины управления.

7.5. СПРАВОЧНЫЕ СВЕДЕНИЯ ПО МИКРОСХЕМАМ[1*]

К155ИЕ5. Быстродействующий счетчик К155ИЕ5 представляет собой асинхронный делитель частоты на триггерах типа *J-K* (рис. 7.3).

Первый триггер, имеющий тактовый вход ***C1*** и изолированный от других триггеров прямой выход ***1***, представляет собой счетчик-делитель на 2. Три остальных триггера образуют счетчик-делитель на 8. Счетные импульсы при этом должны подаваться на вход ***C2***, а частота, деленная на 8, снимается с выхода 4. Счетчики работают самостоятельно, однако выбор их производится по входам ***R1*** и ***R2***, которые обеспечивают два режима работы. Значения сигналов на этих входах для различных режимов работы счетчиков приведены в таблице на рис. 7.3.



Режим работы счетчика	Состояние установочного входа	
	R1	R2
Установка в „0"	1	1
Счет	0	X
	X	0

Рис. 7.3. Условное обозначение микросхемы К155ИЕ5:
x - любое состояние; * - 3-е состояние (состояние высокого сопротивления)

---

[1] **Описание схем различных типов микросхем приводится в соответствии с техническими условиями на них.**

Переключение триггеров происходит по каждому срезу входных тактовых импульсов. Путем внешнего соединения выхода *1* с входом *C2* образуется двоичный счетчик-делитель на 16. Счетные импульсы при этом подаются на вход *C1*, а частота, деленная на 16, снимается с выхода *4*. В режиме счета состояния выходов триггеров меняются в последовательности двоичного счета от 0 до 15.

К155ИЕ8. Счетчик К155ИЕ8 представляет собой шестиразрядный двоичный делитель частоты с постоянным и переменным коэффициентом деления (рис. 7.4). Схема имеет тактовый счетный вход *C*; вход *R* установки в нуль; стробирующий вход *V1*, который управляет дешифратором коэффициента; разрешающий буферный вход *V2*, служащий для управления входами; шесть информационных входов *D1* - *D6* для выбора коэффициента деления частоты; вход *V3* одиночно-каскадного счета, используемый для управления выходом; выходы *A1* и *A2* для счета импульсов в прямом и инверсном коде; разрешающий выход B, используемый при увеличении разрядов счетчика. Режим счета обеспечивается подачей на входы *V1*, *V2*, *R* уровня логического нуля.

В зависимости от кодовой комбинации на информационных входах *D1*, *D6* счетчик из 64 входных импульсов выделяет *M* импульсов, определяемых по формуле

$$M = A\cdot 2^0 + B\cdot 2^1 + C\cdot 2^2 + D\cdot 2^3 + E\cdot 2^4 + F\cdot 2^5,$$

где *A*, *B*, *C*, *D*, *E*, *F* - состояния информационных входов *D6* - *D1* соответственно.

При этом частота на выходе счетчика

$$f_{вых} = \frac{Mf_{вх}}{64}.$$

Число *M* определяет количество импульсов за цикл, равный 64 периодам входного тактового импульса.

Если на всех информационных входах представлен логический нуль, то на выходе *A2* устанавливается состояние логической единицы. Если один из информационных входов находится в состоянии логической единицы, то *M* оказывается числом, кратным 2, и счетчик осуществляет деление на целое постоянное число. Во всех остальных случаях коэффициент деления - число дробное. Состояние (или число импульсов на выходах счетчика) в зависимости от состояний входов приведено в таблице на рис. 7.4.

Когда входные схемы И дешифратора коэффициента закрыты стробом, вход *V3* одиночно-каскадного счета, будучи подсоединенным к счетному входу, может быть использован для передачи счетной частоты в инверсном коде на выход *A1*.

Для получения 12-разрядного делителя частоты разрешающий выход *B* должен быть подсоединен к стробирующему и разрешающему входам следующего каскада. Выход *A2* подсоединяется к входу *V3* следующего каскада счетчика, частота промежуточного деления снимается с выхода *A1*.

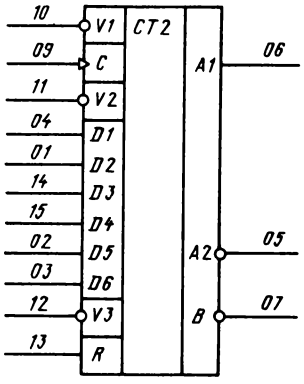
К155ИР1. Схема представляет собой четырехразрядный регистр сдвига с входами последовательного и параллельного занесения. Функциональная схема регистра (рис. 7.5) содержит четыре счетных триггера, пять схем И-ИЛИ-НЕ и шесть инверторов.

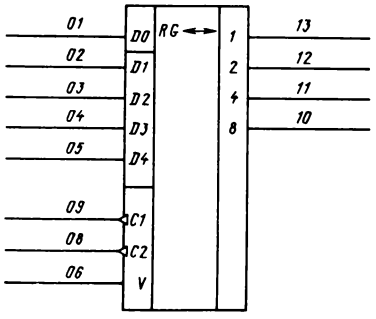
К155ИР1 имеет два тактовых счетных входа *C1*, *C2*, управляющий вход *V* предварительной установки (выбор режима работы), один информационный вход *D0* для занесения информации в последовательном коде, четыре входа *D1* - *D4* для занесения информации в параллельном коде и четыре выхода (*1*, *2*, *4*, *8*) с каждого разряда регистра.

Схема может работать в четырех режимах, в которых можно выполнить: сдвиг информации вправо; сдвиг информации влево; параллельное занесение; хранение.

Выбор режима работы осуществляется подачей уровня логического нуля или логической единицы на вход *V* предварительной установки. При работе схемы в режиме преобразования последовательного кода в параллельный со сдвигом вправо на вход *V* подается уровень логического нуля. При этом отключаются параллельные входы, разрешается занесение в регистр информации в последовательном коде, устанавливается связь выхода каждого разряда с входом последующего, а также разрешается прохождение тактовой серии через вход *C1*. Сдвиг вправо на один разряд осуществляется при каждом отрицательном перепаде тактового импульса, поданного на вход *C1*. Четырехразрядная информация в параллельном коде появляется на выходах *1*, *2*, *4*, *8* через четыре такта входного счетного импульса.

Для использования схемы в качестве преобразователя последовательного кода в параллельный с сдвигом влево необходимо выполнить внешние соединения





Вход										Выход		
управляющий			информационный						строб	A1	A2	B
R	V1	V2	D1	D2	D3	D4	D5	D6	V3			
1	1	X	X	X	X	X	X	X	1	0	1	1
0	0	0	0	0	0	0	0	0	1	0	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	0	0	1	0	1	2	2	1
0	0	0	0	0	0	1	0	0	1	4	4	1
0	0	0	0	0	0	1	0	1	1	5	5	1
0	0	0	0	0	0	1	1	0	1	6	6	1
0	0	0	0	0	0	1	1	1	1	7	7	1
0	0	0	0	0	1	0	0	0	1	8	8	1
0	0	0	0	1	0	0	0	0	1	16	16	1
0	0	0	1	0	0	0	0	0	1	32	32	1
0	0	0	1	1	1	1	1	1	1	63	63	1
0	0	0	1	1	1	1	1	1	0	0	63	1
1	1	X	X	X	X	X	X	X	0	1	1	1

Режим работы	Состояние входа		
	V	C1	C2
Сдвиг вправо	0	↓	X
Параллельное занесение (сдвиг влево)	1	X	↓
Хранение при сдвиге вправо	0	1	X

Рис. 7.4. Условное обозначение микросхемы К155ИЕ8

Рис. 7.5. Условное обозначение микросхемы К155ИР1

выходов каждого разряда регистра с входом параллельного занесения предыдущего разряда. При этом на вход *V* предварительной установки должен подаваться уровень логической единицы, который отключает вход последовательного занесения *D0*, разрывает связь выхода каждого разряда с входом последующего, разрешает занесение информации через входы параллельного занесения с выхода каждого разряда на вход предыдущего и прохождение тактовой серии через тактовой серии через вход *C2*. Информация в последовательном коде заносится через вход параллельного занесения последнего каскада регистра *D4*. Сдвиг влево осуществляется при каждом отрицательном перепаде тактового импульса, поданного на вход *C2*.

Параллельное занесение информации в регистр происходит через параллельные входы *D1* - *D4* при наличии на управляющем входе *V* уровня логической единицы и с приходом на вход *C2* отрицательного перепада напряжения. При этом входы последовательного занесения *D0* и тактовой частоты *C1* отключаются.

Состояния управляющего и счетных входов, а также входа последовательного занесения при различных режимах работы регистра сдвига приведены в таблице на рис. 7.15.

Соединив последовательно несколько регистров описываемого типа, можно получить *n*-разрядный регистр со сдвигом вправо или влево.

Для организации сдвига влево необходимо выполнить внешние соединения.

К155КП2. Схема представляет собой два коммутатора, подключающие четыре входа к одному выходу со стробированием (рис. 7.6).

Схема имеет два управляющих входа *V1* и *V2*, являющиеся общими для обоих коммутаторов, два стробирующих входа *R1* и *R2*, восемь информационных входов

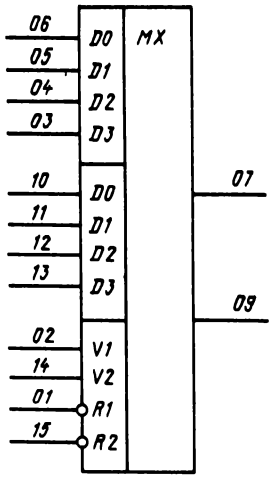


Рис.7.6. Условное обозначение микросхемы К155КП2

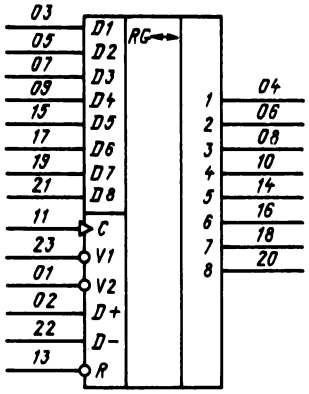


Рис.7.7. Условное обозначение микросхемы К155ИР13

Режим работы	V1	V2	R	D_+	D_-	C
Установка в „0"	X	X	0	X	X	X
Последовательное занесение, сдвиг вправо	0	1	1	0/1	0	⎍
Последовательное занесение, сдвиг влево	1	0	1	0	0/1	⎍
Параллельное занесение	1	1	1	0	0	⎍
Хранение	0	0	1	0/X	X/0	X

D0 - *D4* и два выхода. В зависимости от кодовой комбинации на управляющих входах открывается по одной схеме совпадения И каждого коммутатора. Сигнал с соответствующего информационного входа будет передан в прямом коде на выход того коммутатора, стробирующий вход которого находится в состоянии логического нуля (см. табл. на рис. 7.16).

Уровень логической единицы на стробирующем входе *R* запрещает коммутацию любого входа на выход, на котором будет сохраняться уровень логического нуля независимо от состояний информационных входов.

Схема может быть использована также в качестве преобразователя параллельного кода в последовательный.

К155ИР13. Схема представляет собой восьмиразрядный двунаправленный регистр сдвига с последовательным и параллельным занесением информации (рис. 7.7).

Схема имеет входы последовательного занесения D_+ при сдвиге вправо и D_- при сдвиге влево, восемь входов параллельного занесения *D1* - *D8*, тактовый счетный вход *C*, управляющие входы *V1* и *V2* для выбора режима работы, вход *R* установки в нуль и восемь выходов (*1* - *8*) каждого разряда.

В зависимости от состояний установочных входов *V1*, *V2* и *R* схема может работать в различных режимах: последовательного занесения со сдвигом вправо; последовательного занесения со сдвигом влево; параллельного занесения; хранения; установки в нуль.

При работе в режиме последовательного занесения со сдвигом информации вправо на вход *V1* подается уровень логического нуля, а на вход *V2* - уровень логической единицы.

Информация в последовательном коде поступает на информационный вход D_+. Сдвиг вправо на один разряд выполняется синхронно при каждом положительном перепаде тактового импульса.

Если на управляющий вход *V1* подать уровень логической единицы, а на вход *V2* - уровень логического нуля, то на каждый положительный перепад тактового импульса будет выполняться сдвиг влево. При этом вновь поступающая информация подается на последовательный вход сдвига влево *D* .

Параллельное занесение информации осуществляется через информационные входы *D1* - *D8*, когда оба установочных входа находятся в состоянии логической единицы. На выходах регистра информация в параллельном коде появится синхронно по положительному перепаду тактового импульса.

При отсутствии тактового импульса информация будет храниться в регистре до прихода следующего положительного перепада тактового импульса.

Режимы работы, в которых может работать регистр, в зависимости от состояний входов приведены в таблице на рис. 7.7.

Состояние управляющих входов *V1*, *V2* может меняться только тогда, когда тактовый вход находится в состоянии логической единицы.

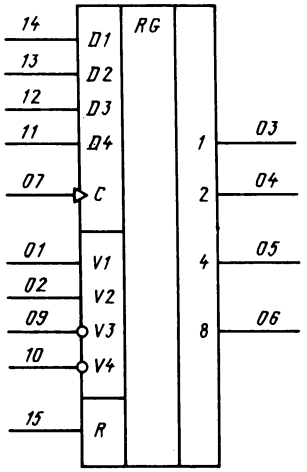
К155ИР15. Схема представляет собой четырехразрядный регистр, построенный на триггерах *D*-типа, с выходами, принимающими три устойчивых состояния (рис. 7.8).

Схема имеет четыре информационных входа *D1* - *D4* для занесения информации

в параллельном коде, тактовый вход *C*, вход *R* установки в нуль, управляющие входы *V3*, *V4* для разрешения параллельной записи, входы *V1*, *V2* управления выходами и четыре выхода *1*, *2*, *4*, *8*.

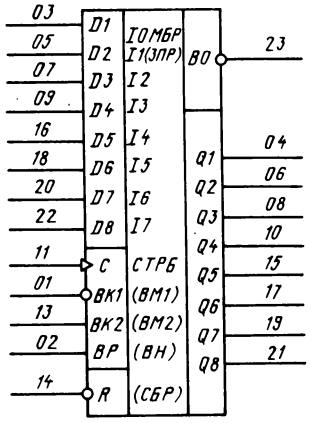
Состояние триггеров меняется в соответствии с состоянием информационного входа в момент перехода тактового импульса *C* в состояние логической единицы. Уровень логической единицы на входе *R* переводит все выходы триггера в состояние логического нуля. Уровень логической единицы, поданный на один из входов *V1* и *V2*, переводит выходы в третье состояние. Состояние выходов регистра в зависимости от состояния входов приведено в таблице на рис. 7.8.

К589ИР12. Схема представляет собой универсальный восьмиразрядный регистр, состоящий из *D*-триггеров и входных буферных схем с тремя устойчивыми состояниями. Схема имеет встроенную селективную логику и отдельный независимый *D*-триггер для формирования запроса на прерывание центрального процессора. Регистр (рис. 7.9) имеет восемь информационных входов *D1* - *D8*, два входа *ВК1*, *ВК2* выбора кристалла, вход *ВР* выбора режима, тактовый вход *C*, вход *R*



Вход					Выход
R	C	V3	V4	D	0
1	X	X	X	X	0
0	1	X	X	X	*
0	0	X	X	X	*
0	⊓	1	X	X	*
0	⊓	X	1	X	*
0	⊓	0	0	0	0
0	⊓	0	0	1	1

Рис. 7.8. Микросхема К155ИР15: x — состояние



Состояние входа			Состояние выхода	Режим
C	ВР	$\overline{ВК1 \cdot ВК2}$	Q_n	
1	0	0	Z	Запись информации при закрытом выходе
0	0	0	Z	Хранение
X 0	1 0	0 1	Q_{n-1}	Выдача информации при закрытом входе
X 1	1 X	1 1	d	Передача информации с входа на выход

Входы			Состояние триггера Тр Зп	Состояние выхода ЗП	Режим
R	$\overline{ВК1 \cdot ВК2}$	C			
0	0	X	1	1	Установка в исходное состояние
X 1 1	1 1 0	0 X ⊓	1 1 0	0	Запрос прерывания
1	0	0	0	0	Хранение предыдущего состояния Тр Зп

Рис. 7.9. Условное обозначение микросхемы К589ИР12

установки в нуль, восемь выходов ***Q1 - Q8*** каждого разряда регистра и выход ***ЗП*** запроса прерывания.

Информационные ***D***-триггеры повторяют входную информацию при высоком логическом уровне на тактовом входе ***C***. При переходе тактового входа в состояние логического нуля происходит запоминание входной информации. Выходы каждого информационного триггера соединены с выходными буферными ключами с тремя устойчивыми состояниями.

Внутренняя шина выдачи информации (***ВД***) стробирует каждый выходной буфер. При напряжении логической единицы на шине ***ВД*** выходные буфера разблокированы, и данные поступают на выход соответствующей линии выходных данных (***Q1 - Q8***).

Входы ***ВК1*** и ***ВК2*** управляют выборкой кристалла. При напряжении логического нуля и логической единицы выборка кристалла разрешена соответственно на входах ***ВК1*** и ***ВК2***.

Состояние входа ***ВР*** определяет один из двух режимов работы регистра. При напряжении логического нуля на входе ***ВР*** устройство работает во входном режиме, и управление записью осуществляется сигналом по входу ***C***. Однако состояние выходных буферов зависит от состояния входов ***ВК1*** и ***ВК2***. Когда устройство выбрано, т.е. $ВК1 \cdot ВК2 = 1$, то выходные буфера открыты. В противном случае все выходы переходят в третье состояние.

При напряжении логической единицы на входе ***ВР*** устройство работает в выходном режиме. В этом случае выходные буфера открыты независимо от выборки устройства. Таким образом, запись в регистр и выдача информации из регистра происходят по следующим формулам:

$З = C \cdot \overline{ВР} \vee ВР \cdot \overline{ВК1} \cdot ВК2$ (запись информации);

$ВД = ВР \vee \overline{ВК1} \cdot ВК2$ (выдача информации).

Состояние выходов микросхемы в зависимости от состояний управляющих и стробирующего входов представлено в таблице на рис. 7.9.

Уровень логического нуля на входе ***R*** переводит все информационные триггеры в нулевое состояние. Триггер запроса прерывания при этом устанавливается в состояние логической единицы, т.е. данное устройство не требует прерывания. Выход триггера запроса прерывания ***ТЗП*** объединен с выходом логической схемы выбора устройства по схеме ИЛИ.

При работе схемы в режиме ввода информации (на входах ***ВР*** и $ВК1 \cdot ВК2$ уровень логического нуля) входной сигнал на входе ***C*** по срезу производит установку триггера запроса прерывания в нуль.

Многорежимный буферный регистр находится в состоянии прерывания тогда, когда выход ***ЗП*** соответствует состоянию логического нуля, что позволяет обеспечить прямое соединение выхода ***ЗП*** со входами запроса блока приоритетного прерывания.

Логическая формула условия прерывания:

$\overline{ЗП} = \overline{ВК1 \cdot ВК2 \vee C \cdot R}$.

Логическая формула сброса сигнала прерывания:

$ЗП = (ВК1 \vee ВК2) \cdot \overline{R}$.

Состояния выхода запроса прерывания ***ЗП*** и выхода ***ТЗП*** в зависимости от состояний управляющих и стробирующего входов представлены в таблице на рис. 7.9.

К589АП16. Схема является параллельным двунаправленным формирователем

сигналов для управления магистралями в цифровых вычислительных устройствах и просто шинным формирователем (ШФ).

Схема (рис. 7.10) представляет собой четырехканальный комму татор, имеющий в каждом канале одну шину ***A*** для приема информации, одну шину ***C*** для выдачи и одну двунаправленную шину ***B*** для приема и выдачи информации.

Схема имеет четыре входа информации (***A1*** - ***A4***), вход ***ВК*** выборки кристалла, вход ***УВ*** управления выдачей информации, четыре входа-выхода реверсивной передачи информации (***B1*** - ***B4***) и четыре выхода информации (***C1*** - ***C4***). Все выходы могут принимать три устойчивых состояния.

Для управления режимом работы и направлением передачи информации в ***ШФ*** предусмотрена логика управления выдачей, выполненная на двух логических элементах И с двумя входами. Формирователь обеспечивает передачу информации при уровне логического нуля на входе ***ВК***. При этом управление выдачей информации на шины ***C*** и ***B*** осуществляется сигналом на входе ***УВ***. Если на входе ***УВ*** уровень логического нуля, то разрешена передача информации с входа ***A*** на выход ***B***. Выход ***C*** при этом переходит в состояние высокого выходного сопротивления. Если на входе ***УВ*** уровень логической единицы, то разрешена передача информации с входа ***B*** на выход ***C***. Выход ***B*** при этом переходит в состояние высокого выходного сопротивления.

Уровень логической единицы на входе ***ВК*** запрещает передачу информации, и оба выхода ***B*** и ***C*** переходят в состояние высокого выходного сопротивления.

На выход ШФ информация передается в прямом коде.

К555ИЕ6. Схема представляет собой синхронный четырехразрядный реверсивный двоично-десятичный счетчик, который состоит из четырех двухступенчатых триггеров, дешифратора счета и логической схемы предварительной установки

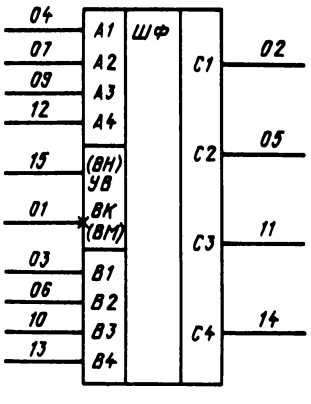
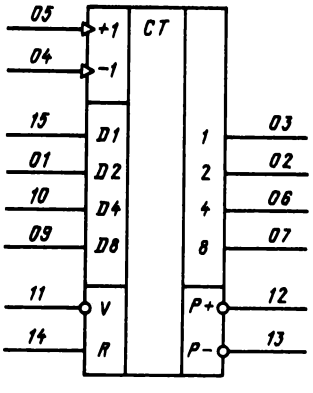


Рис. 7.10. Микросхема К589АП16



Режим работы счетчика	Вход						Выход	
	R	V	D1	D2	+1	-1	1	2
Установка в „0"	1	X	X	X	X	X	0	0
Запись	0	0	1	0	X	X	1	0
Хранение	0	1	X	X	–	–	1	0
Счет в режиме суммирования	0	1	X	X	⎍	1	*	*
Счет в режиме вычитания	0	1	X	X	1	⎍	*	*

Рис. 7.11. Условное обозначение микросхемы К555ИЕ6

(рис. 7.11). Отличительной особенностью схемы является возможность параллельной записи информации в счетчик.

Схема имеет два счетных входа (вход *+1* в режиме суммирования и вход в режиме вычитания *-1*); четыре входа параллельной записи *D1* - *D4*; управляющий вход *V*, разрешающий параллельную запись информации; вход *R* установки в нуль; выходы *1*, *2*, *4*, *8* четырех разрядов счетчика; выходы прямого P_+ и обратного P_- переносов, позволяющие осуществлять каскадное соединение счетчиков без дополнительной логики.

В зависимости от состояния установочных и управляющих входов возможны три режима работы счетчика: режим установки в логический нуль; режим параллельной записи; режим хранения, режим счета.

Режим установки в нуль обеспечивается подачей на вход *R* уровня логической единицы. При этом отключается вход, разрешающий запись, и входы параллельной записи.

Режим параллельной записи обеспечивается подачей на входы *V* и *R* уровня логического нуля. При этом импульсы, поданные на информационные входы *D1* - *D4*, появляются на выходах триггеров независимо от состояния входного тактового импульса.

Режим хранения обеспечивается подачей на вход *V* уровня логической единицы, а на вход *R* - уровня логического нуля. При этом запрещается запись новой информации, и до прихода тактового импульса предыдущая информация хранится в счетчике. Поступление тактового импульса приводит к изменению состояния счетчика на следующее в последовательности двоично-десятичного счета. Дальнейший счет осуществляется по каждому положительному перепаду тактового импульса, когда на втором счетном входе имеется уровень логической единицы.

Состояния входов и выходов счетчика при различных режимах его работы приведены в таблице на рис. 7.11[2]

На выходе прямого переноса P_+ импульс отрицательной полярности формируется при переполнении счетчика, т.е. при появлении в нем максимального числа, и при условии, что тактирующий импульс, поданный на вход *+1*, находится на уровне логического нуля. На выходе обратного переноса P_- импульс формируется при появлении на выходах всех разрядов счетчика логического нуля, когда тактовый импульс, поданный на вход *-1*, находится в состоянии логического нуля. Длительность импульсов на выходах P_+ и P_- равна длительности отрицательного импульса на счетном входе.

Каскадное соединение счетчиков образуется соединением выхода прямого переноса со счетным входом *+1* следующего счетчика, а также соединением выхода обратного переноса P_- со счетным входом *-1* следующего счетчика.

---

[2] **В таблице для примера приведены состояния двух информационных входов и соответствующих им выходов.**

Схема может быть использована во многих устройствах, когда первоначальное число должно быть занесено в счетчик и многократно просчитано.

К555КП13. Схема представляет собой четырехразрядный регистр с четырьмя двухвходовыми мультиплексорами на входах (рис. 7.12).

Схема содержит четыре пары информационных входов *D1* - *D8*, управляющий вход *V* выбора слова, тактовый вход *C* и четыре выхода *Q1* - *Q4* каждого разряда слова. В зависимости от состояния управляющего входа *V* в регистр заносится одно из двух выбираемых слов. Если вход *V* находится на уровне логического нуля, то по отрицательному фронту тактового импульса в регистр заносится информация со входов *D1*, *D3*, *D5*, *D7*. Для выбора другого слова вход *V* должен быть переведен в состояние логической единицы.

При соединении входов и выходов нескольких схем друг с другом можно получить регистр сдвига четырехразрядного двоичнодесятичного кода. Сдвиг из регистра в регистр будет осуществляться по каждому отрицательному фронту тактового импульса. В регистре сдвига сохраняется возможность параллельной загрузки.

К555КП11. Схема представляет собой четыре коммутатора двух входов на один выход, принимающий три устойчивых состояния (рис. 7.13).

Каждый коммутатор имеет два информационных входа *D1*, *D2* и один выход, а также общие для всех коммутаторов управлящий вход *V* и вход *Z* управления выходом. В зависимости от состояния входа *V* сигнал с информационного входа *D1* или *D2* передается в прямом коде на выход коммутатора. При этом вход *Z* должен находиться в состоянии логического нуля. При подаче на вход *Z* уровня

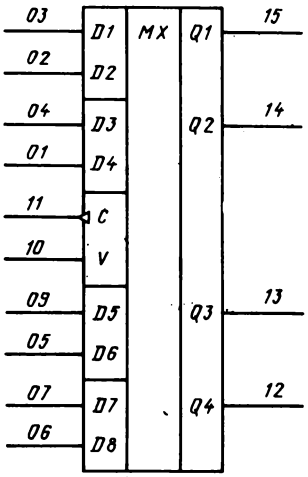
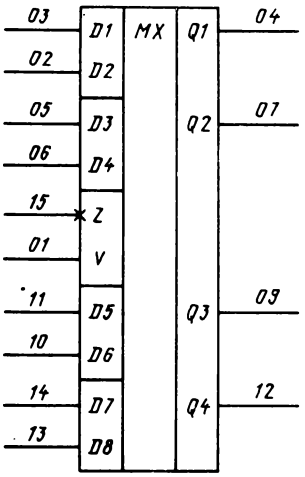


Рис. 7.12. Условное обозначение микросхемы К555КП13



Вход				Выход
D1	D2	Z	V	
X	X	1	X	Z
0	X	0	0	0
1	X	0	0	1
X	0	0	1	0
X	1	0	1	1

Рис. 7.13. Условное обозначение микросхемы К555КП11

логической единицы выходы переходят в третье состояние независимо от состояния информационных входов.

Состояние выхода в зависимости от состояния информационных и управляющих входов приведено в таблице на рис. 7.13.

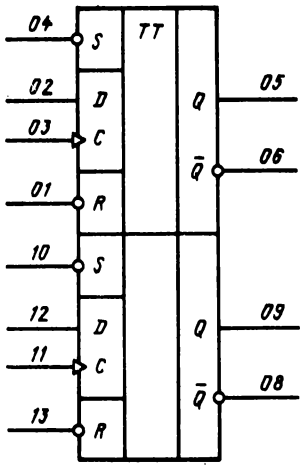
Способность данных схем принимать состояние, характеризуемое высоким выходным сопротивлением, позволяет использовать их для организации ПРОВОДНОГО ИЛИ объединением по выходу нескольких схем.

К555ТМ2. Схема (рис. 7.14) представляет собой два триггера типа *D*, переключающиеся по положительному фронту.

Каждый триггер имеет информационный вход *D*, установочные входы *R* и *S*, тактовый вход *C*, прямой *Q* и инверсный $\overline{Q}$ выходы.

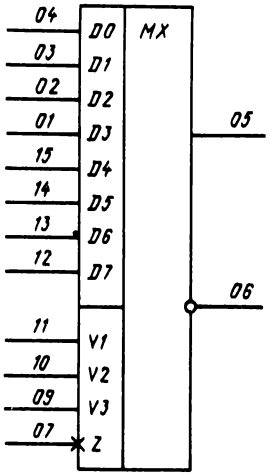
Уровень логического нуля на входе устанавливает триггер в единичное состояние. Информация со входа передается на выход в момент перехода тактового импульса *C* из состояния логического нуля в состояние логической единицы. При этом на вход *R* должен быть подан уровень логической единицы.

Когда на тактовом входе *C* постоянный уровень логического нуля или логи-



Вход				Выход	
S	R	C	D	Q	$\bar{Q}$
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	1*	1*
1	1	↓	1	1	0
1	1	↓	0	0	1
1	1	0	X	H	H
1	1	1	X	H	H

Рис. 7.14. Условное обозначение микросхемы К555ТМ2



Вход				Выход	
V1	V2	V3	Z	прямой	инверсный
X	X	X	1	Z*	Z*
0	0	0	0	D0	$\overline{D0}$
1	0	0	0	D1	$\overline{D1}$
0	1	0	0	D2	$\overline{D2}$
1	1	0	0	D3	$\overline{D3}$
0	0	1	0	D4	$\overline{D4}$
1	0	1	0	D5	$\overline{D5}$
0	1	1	0	D6	$\overline{D6}$
1	1	1	0	D7	$\overline{D7}$

Рис. 7.15. Условное обозначение микросхемы К555КП15

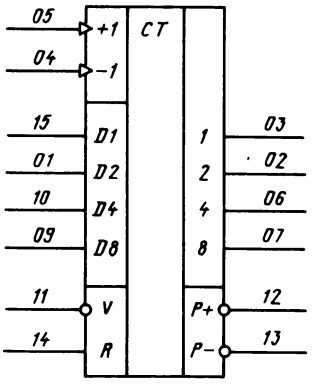
ческой единицы, то сигнал с информационного входа *D* на выход не передается.

Состояние выходов триггера в зависимости от состояния входов приведено в таблице на рис. 7.14.

К555КП15. Схема представляет собой коммутатор восьми входов на один выход с тремя состояниями (рис. 7.15).

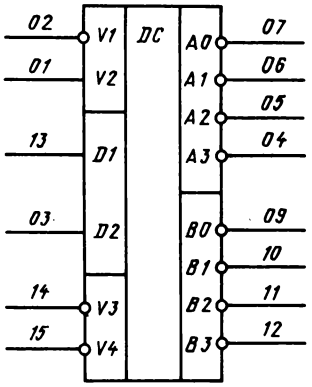
Схема имеет три управляющих входа *V*1 - *V*3, восемь информационных входов *D*0 - *D*7, один стробирующий вход *Z*, два выхода. В зависимости от кодовой комбинации на управляющих входах разрешается передача информации с одного из информационных входов на выходы через одну из восьми схем совпадения И. Стробирующий вход *Z* при этом должен находиться в состоянии логического нуля.

Уровень логической единицы на стробирующем входе переводит выходы коммутатора в третье состояние. Таблица на рис. 7.16 поясняет, с какого из восьми входов информация передается на выход в зависимости от кодовой комбинации на управляющих входах.



Режим работы счетчика	Вход						Выход	
	R	V	D1	D2	+1	−1	1	2
Установка в „0"	1	X	X	X	X	X	0	0
Запись	0	0	1	0	X	X	1	0
Хранение	0	1	X	X	–	–	1	0
Счет в режиме суммирования	0	·1	X	X	⎍	1	*	*
Счет в режиме вычитания	0	1	X	X	1	⎍	*	*

Рис. 7.16. Условное обозначение микросхемы К555КП12



Состояние входа				Состояние выхода			
D1	D2	V1	V2	A0	A1	A2	A3
X	X	1	X	1	1	1	1
0	0	0	1	0	1	1	1
1	0	0	1	1	0	1	1
0	1	0	1	1	1	0	1
1	1	0	1	1	1	1	0
X	X	X	0	1	1	1	1

Состояние входа				Состояние выхода			
D1	D2	V3	V4	B0	B1	B2	B3
X	X	1	X	1	1	1	1
0	0	0	0	0	1	1	1
1	0	0	0	1	0	1	1
0	1	0	0	1	1	0	1
1	1	0	0	1	1	1	0
X	X	X	1	1	1	1	1

Рис. 7.17. Условное обозначение микросхемы К555ИД4

Схема может быть использована в качестве преобразователя параллельного кода в последовательный.

К555КП12. Схема представляет собой два коммутатора четырех входов на один выход со стробированием (рис. 7.16). Выходы коммутатора могут принимать три устойчивых состояния.

Каждый коммутатор имеет по четыре информационных входа, два управляющих входа *V1* и *V2*, являющиеся общими для обоих коммутаторов, стробирующий вход *Z* и по одному выходу. В зависимости от кодовой комбинации на управляющих входах разрешается передача информации через одну из схем совпадения И каждого коммутатора. Сигнал с соответствующего информационного входа будет передан в прямом коде на выход того коммутатора, стробирующий вход которого находится в состоянии логического нуля. Состояние выходов коммутатора в зависимости от состояния входов приведено в таблице на рис. 7.16.

Уровень логической единицы на стробирующем входе *Z* запрещает коммутацию любого соответствующего входа на выход, который при этом переходит в третье состояние, характеризуемое высоким выходным сопротивлением. Наличие третьего состояния позволяет использовать данную схему при работе на магистраль.

Схема может быть использована в качестве преобразователя параллельного кода в последовательный.

К555ИД4. Схема представляет собой два двухразрядных дешифратора (рис. 7.17). Дешифраторы имеют два общих управляющих входа *D1* и *D2*, по два стробирующих входа *V1*, *V2* и *V3*, *V4*, а также по четыре выхода *A0* - *A3* и *B0* - *B3* на каждый дешифратор.

В зависимости от кодовой комбинации на управляющих и стробирующих входах сигнал, соответствующий уровню логического нуля, появляется на выходе в соответствии с таблицей на рис.7.17.

Схема К555ИД4 может использоваться для дешифрации трехразрядного кода. В этом случае необходимо объединить входы *V2* и *V4*, а также *V1* и *V3* между собой. В качестве управляющих входов в этом случае используются входы *D1*, *D2* и *V2*(*V4*), а вход *V1*(*V3*) - в качестве стробирующего.

Схему К555ИД4 можно также использовать в качестве сдвоенного демультиплексора с одного входа на четыре выхода. В таблице приведены сведения, характеризующие работу каждого из них, если принять, что входы *V2* и *V4* используются для передачи данных соответственно для первого и второго демультиплексора, входы *V1* и *V3* - стробирующие, а общие входы *D1* и *D2* используются для выбора (подключения) необходимых выходов.

Схема трехразрядного дешифратора является одновременно и демультиплексором с одного входа на восемь выходов. Данные подаются на вход *V1*(*V3*), выбор необходимого выхода осуществляется подачей кодовых комбинаций на входы *D1*, *D2* и *V2*(*V4*).

К555ИР16. Схема представляет собой четырехразрядный регистр сдвига (рис. 7.18) с входом последовательного и параллельного занесения и с выходами, принимающими три устойчивых состояния.

Схема имеет один информационный вход *D* для занесения информации в параллельном коде, четыре входа *D1* - *D4*, управляющий вход *V* выбора режима

работы, тактовый счетный вход *C*, вход управления *Z* выходом, четыре выхода *1*, *2*, *4*, *8* с каждого разряда регистра.

В зависимости от состояния управляющего *V* и счетного *C* входов регистр может работать в режимах: параллельного занесения; последовательного занесения со сдвигом вправо; последовательного занесения со сдвигом влево; хранения.

Режим параллельного занесения обеспечивается подачей на управляющий вход *V* уровня логической единицы. При этом вход последовательного занесения отключается, и информация с параллельного входа заносится в соответствующий триггер и появляется на выходе с приходом отрицательного перепада тактового импульса.

Режим последовательного занесения со сдвигом вправо, т.е. от младшего разряда к старшему, обеспечивается подачей на управляющий вход *V* уровня логического нуля. Информация в последовательном коде подается на вход D_+ последовательного занесения. Сдвиг вправо на один разряд осуществляется по каждому отрицательному перепаду тактового импульса. Режим последовательного занесения со сдвигом влево обеспечивается при выполнении внешних соединений выходов каждого разряда регистра с входами параллельного занесения предыдущего разряда. При этом управляющий вход *V* должен находиться в состоянии логической единицы, а информация в последовательном коде должна подаваться на вход *D4* параллельного занесения старшего разряда. Сдвиг информации влево на один разряд осуществляется по каждому отрицательному перепаду тактового импульса.

При работе регистра во всех указанных режимах управляющий вход *Z* должен находиться в состоянии логической единицы. При подаче уровня логического нуля на вход управления выходом Z все выходы переходят в третье состояние, характеризуемое высоким выходным сопротивлением независимо от состояния других входов.

Состояние управляющих и счетного входов при различных режимах работы регистра приведено в таблице на рис. 7.18.

Заметим, что для организации сдвига влево необходимо выполнить внешние соединения. Соединив последовательно несколько регистров описываемого типа, можно получить n-разрядный регистр со сдвигом вправо или влево.

К555ИЕ7. Схема представляет собой синхронный четырехразрядный реверсивный двоичный счетчик (рис. 7.19).

Схема имеет два счетных входа (вход в режиме суммирования *+1* и вход в режиме вычитания *-1*); четыре информационных входа *D1*, *D2*, *D4*, *D8* параллельной записи, управляющий вход *V*, разрешающий параллельную запись информации; вход *R* установки в нуль; выходы *1*, *2*, *4*, *8* четырех разрядов счетчика; выход P_+ прямого переноса и выход P_- обратного переноса, позволяющие осуществить каскадное соединение счетчиков без дополнительной логики.

В зависимости от состояния установочных и управляющих входов возможны четыре режима работы счетчика: режим установки в логический нуль; режим занесения предварительной информации; режим хранения информации; режим счета.

Режим установки в логический нуль обеспечивается подачей на вход *P* уровня

логической единицы. При этом отключается вход, разрешающий запись, и входы параллельной записи.

Режим параллельной записи обеспечивается подачей на входы ***R*** и ***V*** уровня логического нуля. При этом импульсы, поданные на информационные входы ***D1***, ***D2***, ***D4***, ***D8***, появляются на выходах триггеров независимо от состояния входного тактового импульса.

Режим хранения информации обеспечивается подачей на вход ***V*** уровня логической единицы, а на вход ***R*** - уровня логического нуля. При этом запрещается запись новой информации, и до прихода тактового импульса предыдущая информация будет храниться в счетчике. Поступление тактового импульса приводит к изменению состояния счетчика на следующее в последовательности двоичного счета.

Дальнейший счет осуществляется по каждому положительному перепаду тактового импульса при наличии на втором счетном входе уровня логической единицы.

Состояние входов и выходов счетчика при различных режимах его работы приведено в таблице на рис. 7.19.

На выходе P_+ прямого переноса импульс отрицательной полярности форми-

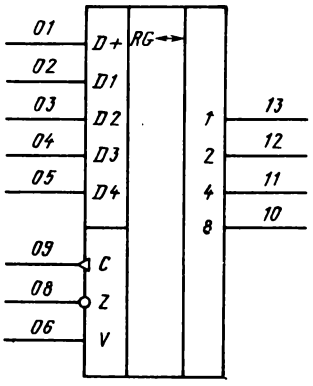
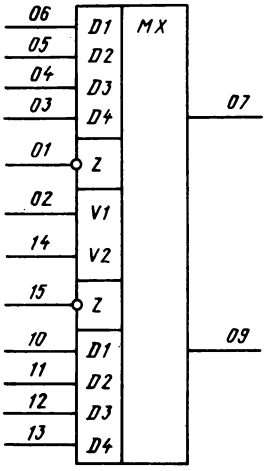


Рис. 7.18. Условное обозначение микросхемы К555ИР16

Режим работы	Состояние входов		
	Z	V	C
Параллельное занесение (сдвиг влево)	1	1	↓
Сдвиг вправо	1	0	↓
Перевод выходов в состояние с высоким сопротивлением	0	X	X
Хранение при сдвиге вправо	1	0	1



Управляющий вход		Информационный вход				Установочные входы	Выход
V1	V2	D1	D2	D3	D4	Z	
X	X	X	X	X	X	1	Z
0	0	0	X	X	X	0	0
0	0	1	X	X	X	0	1
0	1	X	0	X	X	0	0
0	1	X	1	X	X	0	1
1	0	X	X	0	X	0	0
1	0	X	X	1	X	0	1
1	1	X	X	X	0	0	0
1	1	X	X	X	1	0	1

Рис. 7.19. Условное обозначение микросхемы К555ИЕ7

руется при переполнении счетчика, т.е. при появлении в нем максимального числа 15 и при условии, что на вход *+1* подан уровень логического нуля. На входе P_- обратного переноса импульс формируется при появлении на всех разрядах счетчика логического нуля и при условии, что на вход *-1* подан уровень логического нуля.

Длительность импульсов на выходах P_+ и P_- равна длительности отрицательного импульса на счетном входе.

Каскадное соединение счетчиков образуется соединением выхода прямого переноса со счетным входом *+1* следующего счетчика, а также соединением выхода обратного переноса *P* со счетным входом *-1* следующего счетчика.

Глава 8. СИСТЕМНЫЙ БЛОК

8.1. ВНУТРЕННИЙ СИСТЕМНЫЙ ИНТЕРФЕЙС

СОСТАВ ШИН ВНУТРЕННЕГО ИНТЕРФЕЙСА

В ПЭВМ "Агат" реализована архитектура с общей шиной. Обмен информацией между всеми блоками и устройствами компьютера осуществляется по шинам системного интерфейса, образующими единую магистраль.

Все внешние устройства связаны с системной шиной через специальные электронные согласователи - контроллеры внешних устройств. Физически эта связь реализуется через шесть разъемов, к которым подведены линии системного интерфейса. Устройство подключается к соответствующему контроллеру, подсоединенному к любому разъему. Управление осуществляется по линиям внутреннего интерфейса. Сигналы управления и разъемы общей шины образуют внешний интерфейс ввода-вывода.

В состав ПЭВМ включены контроллеры громкоговорителя, магнитофона, клавиатуры, игровых пультов. Электронные схемы размещены на основной плате и являются неотъемлемой частью машины. Управление осуществляется сигналами шин системного интерфейса, образующими встроенный интерфейс ввода-вывода.

В табл. 8.1 приведена структура внутреннего интерфейса, причем линии и соответствующие им сигналы имеют одинаковое название. Сигналы внутреннего интерфейса обеспечивают управление всеми режимами ввода-вывода. Кроме состава сигналов интерфейса указана категория линий - однонаправленного (1н) или двунаправленного (2н) действия.

ФУНКЦИОНАЛЬНО-ВРЕМЕННЫЕ ХАРАКТЕРИСТИКИ ВНУТРЕННЕГО ИНТЕРФЕЙСА

Существует два типа передачи данных шинам внутреннего интерфейса - чтение и запись.

Чтение данных. Активное устройство помещает адрес на шине адреса A0 - A15

Таблица 8.1

Структура магистрали системного интерфейса ПЭВИ "Агат"

Мнемоника шин	Название шин	Число шин	Направленность линии
A0-A15	Адрес	16	2н
D0-D7	Данные	8	2н
R/$\overline{W}$	Чтение-запись	1	2н
$\overline{RES}$	Сброс	1	2н
$\overline{NMI}$	Немаскируемое прерывание	1	2н
$\overline{IRQ}$	Запрос прерывания	1	2н
$\overline{RDY}$	Готовность	1	2н
$\overline{DMA}$	Прямой доступ к памяти	1	2н
INT OUT	Выход прерывания	1	1н
INT IN	Вход прерывания	1	1н
DMA OUT	Выход прямого доступа к памяти	1	1н
DMA INT	Вход прямого доступа к памяти	1	1н
$\overline{INH}$	Блокировка ПЗУ	1	2н
USER1	Потребитель 1	1	2н
$\overline{I/OS}$	Выбор устройства ввода-вывода	1	1н
$\overline{I/O\ STR}$	Выбор ПЗУ	1	1н
$\overline{DS}$	Выборка внешних устройств	1	1н
φ1	Фаза 1-й тактовой частоты	1	1н
φ0	Фаза 1-й тактовой частоты	1	1н
2MHZ	2 МГц	1	1н
7MHZ	Частота 1	1	1н
14MHZ	Частота 2	1	1н
GND	Общая шина	1	1н
+12 V	Питание плюс 12 В	1	1н
+5 V	Питание плюс 5 В	1	1н
-12 V	Питание минус 12 В	1	1н

в течение 300 нс после начала импульса φ1; затем пассивное устройство помещает байт данных на шину данных D0 - D7 в течение 350 нс после начала импульса φ0 и удерживает их не менее 10 нс после окончания φ0. Изменение адреса во время действия сигнала недопустимо (см. рис. 8.1).

Запись данных. Активное устройство помещает адрес данных A0 - A15 и сигнал записи R/$\overline{W}$ в течение 300 нс после начала импульса φ1; затем активное устройство помещает байт данных, предназначенный для записи в пассивное устройство, на шину данных D0 - D7 в течение 200 нс после начала импульса φ0 и удерживает их не менее, чем на 10 нс после окончания φ0 (рис. 8.2).

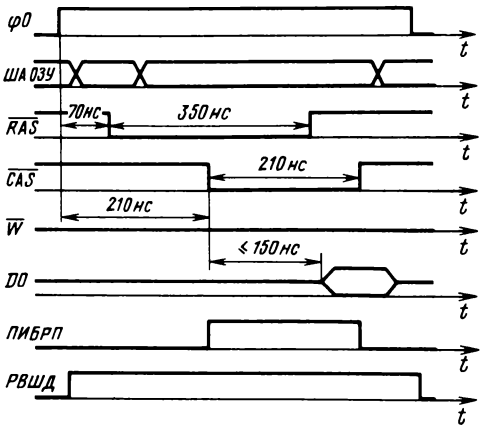


Рис. 8.1. Временные диаграммы операции чтения

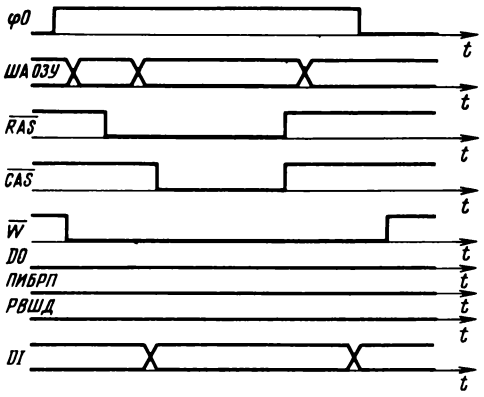


Рис. 8.2. Временные диаграммы операции записи

ФИЗИЧЕСКАЯ РЕАЛИЗАЦИЯ ВНУТРЕННЕГО ИНТЕРФЕЙСА

Физически внутренний интерфейс ПЭВМ "Агат" представляет собой набор информационных и служебеных шин, подключенных к стандартному разъему типа ОНп-КС-23-Р, в который может быть установлен модуль контроллера внешнего устройства. В табл. 8.2 приведено назначение контактов разъема.

Таблица 8.2

Структура разъема интерфейса ПЭВМ "Агат"

Контакт	Наименование сигнала	Направление
A1	+12 В	
A2	+12 В	
A3	D0	
A4	D1	
A5	D2	Вход-выход
A6	D3	
A7	D4	
A8	D5	
A9	D6	
A10	D7	
A11	$\overline{DS}$	
A12	φ0	
A13	USERI	
A14	φ1	Выход
A15	2 МГц	
A16	7 МГц	
A17	14 МГц	
A18		

Продолжение табл. 8.2

Контакт	Наименование сигнала	Направление
А20	$\overline{\text{INY ROM}}$	Выход
А21	RESET	Вход-выход
А22	$\overline{\text{IRQ}}$	Вход
А23	NMI	
А24	INTIN	
А25	DMA IN	
А26	–	–
А27	–	–
А28	GND	Общий
А29	GND	
А30	GND	
Б1	$\overline{\text{I/OS}}$	Выход
Б2	$\overline{\text{I/OS}}$	
Б3	A0	Вход-выход
Б4	A1	
Б5	A2	
Б6	A3	
Б7	A4	
Б8	A5	
Б9	A6	
Б10	A7	
Б11	A8	
Б12	A9	
Б13	A10	
Б14	A11	
Б15	A12	
Б16	A13	
Б17	A14	
Б18	A15	
Б19	$\overline{\text{R/W}}$	
Б20		
Б21	I/O STR	Выход
Б22	$\overline{\text{RDU}}$	Вход
Б23	$\overline{\text{DMA}}$	
Б24	INT OUT	
Б25	DMA OUT	

Продолжение табл. 8.2

Контакт	Наименование сигнала	Направление
Б26	-12 В	Выход
Б27	-	
Б28	+5 В	
Б29	+5 В	
Б30	+5 В	

Шины адреса A (0:15) и данных D (0:7) могут находиться в одном из трех возможных состояний: пассивном, активном и высокого импе данса. Остальные шины имеют только два состояния.

Нормальное значение уровней сигналов:

логический нуль ..... 0,4 В
логическая единица ..... 2,4 В

При подключении к шинам системного интерфейса используются интерфейсные усилители, предназначенные для передачи и приема информации.

Номенклатура микросхем магистральных источников (ИСТ) и приемников (ПРМ), а также число подключаемых к каждому периферийному контроллеру устройств приведены в табл. 8.3. Допускается применение микросхем, имеющих аналогичные или близкие временные параметры, значение входной емкости и силу тока, соответствующую низкому уровню на выходе.

Таблица 8.3

Наименование шины сигнала	Микросхемы		Максимальное число устройств для одного периферийного модуля
	Источник (ИСТ)	Приемник (ПРМ)	
A0-A15	К589АП16	К589АП16 К589ИР12 Серии 155, 555	2 2 2
D0 - D7	К589АП16 К589ИР12	К589АП16, К589ИР12, К531КП11	2 2 2
$\overline{R/W}$	К559ЛП8		2
$\overline{RES}$	К559ЛП8	Серии 155, 555	2
RDY, $\overline{IRQ}$	К559ЛА8		2
$\overline{NMI}$, $\overline{DMA}$	К155ЛП8	Серии 155, 555	2

Продолжение табл. 8.3

Наименование шины сигнала	Микросхемы		Максимальное число устройств для одного периферийного модуля
	Источник (ИСТ)	Приемник (ПРМ)	
INH, USERI 2MHZ φ0, φ1 I/O STR	-	Серии 155, 555	10
$\overline{I/OS}$ $\overline{DS}$	–		
INT OUT DMA OUT INT IN DMA IN	-	Серии 155, 555	10

Нагрузочные резисторы источников сигналов $R/\bar{W}$, $\bar{R}\bar{E}\bar{S}$, $\bar{R}\bar{D}\bar{Y}$, $\bar{I}\bar{R}\bar{Q}$, $\bar{N}\bar{M}\bar{I}$, $\bar{D}\bar{M}\bar{A}$, $\bar{I}\bar{N}\bar{H}$, $\bar{U}\bar{S}\bar{E}\bar{R}1$ установлены в ячейке процессора ПЭВМ и подключены одним концом к указанным сигналам, а другим - к источнику питания +5 В.

ПОСТРОЕНИЕ МОДУЛЕЙ ВНЕШНЕГО ИНТЕРФЕЙСА

Управление контроллерами внешних устройств. В каждый из шести интерфейсных разъемов компьютера (2-й разъем используется для модуля центрального процессора) может быть установлен модуль (ячейка) внешнего интерфейса, осуществляющий те или иные функции ввода-вывода и обработки информации. Для каждого периферийного модуля в поле адресов ввода-вывода отведено 272 адреса, из которых 16 адресов стробирует сигнал $\bar{D}\bar{S}$, а 256 адресов - сигнал $\overline{I/O}\bar{S}$. Кроме того, имеется дополнительная область емкостью 2048 К байт, доступная для всех ячеек внешнего интерфейса и стробируемая сигналом $\overline{I/O}$ $\bar{S}\bar{T}\bar{R}$. Распределение адресов приведено в табл. 8.4. Сигнал $\bar{D}\bar{S}$ формируется низким уровнем в течение действия низкого уровня фазы φ1 при обращении микропроцессора по адресам CONO - CONF, где N = K + 8, а K = 0 + 7 - номер соответствующего разъема. В табл. 8.4 приведено распределение адресов. Из таблицы видно, что сигнал формируется для 16 адресов по каждому из разъемов, кроме X2.

Сигнал $\bar{D}\bar{S}$ и адресные шины A00 - A0F могут быть использованы для адресации внутренних схем периферийной платы или для выработки различных сигналов управления периферийного устройства. Совместное использование их с линией $R/\bar{W}$ удваивает число сигналов $\bar{D}\bar{S}$. Более подробно эти сигналы рассмотрены в п. 8.3.

Постоянная память модуля. Наличие сигнала $\overline{I/O}\bar{S}$, адресующего 256-байтовую страницу памяти для каждого периферийного модуля, позволяет использовать ее

Таблица 8.4

Внешний интерфейс

Диапазон адресов	Сигнал	Номер разъема с сигналом
C090 - C09F	$\overline{DS}$	X1
C0A0 - C0AF	$\overline{DS}$	X3
CB00 - C0BF	$\overline{DS}$	X4
C0C0 - C0CF	$\overline{DS}$	X5
C0D0 - C0DF	$\overline{DS}$	X6
C0E0 - C0EF	$\overline{DS}$	X7
C100 - C1FF	$\overline{I/OS}$	XI
C200 - C2FF	$\overline{I/OS}$	X3
C300 - C3FF	$\overline{I/OS}$	X4
C400 - C4FF	$\overline{I/OS}$	X5
C500 - C5FF	$\overline{I/OS}$	X6
C600 - C6FF	$\overline{I/OS}$	X7
C800 - CFFF	$\overline{I/O\ STR}$	XI - X7, кроме X2

для размещения управляющих программ или подпрограмм (драйверов), предназначенных для управления этим модулем.

Сигнал $\overline{I/OS}$ формируется аналогично сигналу $\overline{DS}$ при обращении микропроцессора по адресам C100 - C6FF (см. табл. 8.4).

Возможны различные случаи использования этого сигнала. При организации на контроллере периферийного устройства памяти сигнал $\overline{I/OS}$ используется как сигнал ВМ (ВК), а адресные сигналы A00 - AFF служат для выбора конкретного адреса. При этом $R/\overline{W}$ определяет соответствующую операцию. На плату можно добавлять небольшое ПЗУ, содержащее программу управления. В некоторых случаях, например при обслуживании высокоскоростных устройств, место ПЗУ занимает буферная память (до 256 байт).

Для построения постоянной памяти рекомендуется использование ПЗУ или перепрограммируемого ПЗУ (ППЗУ) с временем выборки не более 450 нс. Шину данных ПЗУ необходимо подключать к шине данных интерфейса ПЭВМ через выходные буфера в соответствии с табл. 8.3.

Расширение постоянной памяти модуля. Зона памяти 2К байт (2048 байт) с адресами C800 - CFFF зарезервирована за периферийными устройствами для ПЗУ или ППЗУ, предназначенными для хранения больших программ или подпрограмм - драйверов.

Подключение этой памяти осуществляется сигналом $\overline{I/O\ STR}$, который формируется аналогично сигналу $\overline{DS}$ (в соответствии с табл. 8.3) при обращении микропроцессора по адресам C800 - CFFF.

На рис. 8.3 приведена схема усложненной адресации, позволяющая подключить дополнительную память на нескольких разъемах.

Реализация прямого доступа к памяти ($\overline{DMA}$). Передача данных с внешних запоминающих устройств в оперативную память ПЭВМ обычно выполняется побайтно под управлением микропроцессора. При организации высокоскоростных информационных систем управления передачей данных осуществляется отдельной периферийной платой. При этом внешнее устройство получает прямой доступ к ОЗУ ПЭВМ, причем центральный процессор должен быть полностью остановлен. Периферийный модуль получает доступ к шине чтения-записи, а также к адресной шине и шине данных. На рис. 8.4 приведена фукциональная схема, осуществляющая блокировку процессора, шин адреса, данных и R/$\overline{W}$. Запросом на предоставление прямого доступа является низкий уровень сигнала $\overline{RDY}$. Сформированный на D-триггере низкий уровень сигнала $\overline{DMA}$ является признаком готовности шины данных, адреса и R/W центрального процессора к операции прямого доступа.

Сигнал $\overline{DMA}$ должен выдаваться только во время положительной фазы импульсов $\varphi 1$. Он может прервать работу микропроцессора после окончания текущего цикла.

Для организации системы приоритета используются сигналы DMA IN и DMA OUT. Шина DMA OUT обеспечивает выход цепи приоритета к разъемам на низших иерархических уровнях. В рабочем режиме этот вход установлен в состояние логической единицы. В нерабочем режиме шина подсоединена к входу DMA IN.

Аналогично используется шина DMA IN, обеспечивающая вход цепи приоритета к разъемам на высших иерархических уровнях. Если сигнал не используется, шина DMA IN соединена с шиной DMA OUT.

Шины DMA IN и DMA OUT позволяют организовать прямой доступ к памяти по принципу "гирлянды": внешнее устройство с более высоким приоритетом может заблокировать прямой доступ к памяти устройству с низким приоритетом. Для устранения конфликтных ситуаций, возникающих при работе различных источников информации на общую область памяти, каждый периферийный модуль должен иметь устройство коммутации, подключающее его в нужный момент к шине данных ПЗУ. Это устройство включается сигналом I/OS, т.е. дополнительное ПЗУ на любом модуле будет частично включаться после первого обращения программы к устройству, на котором оно находится. Вторым сигналом, включающим допол-

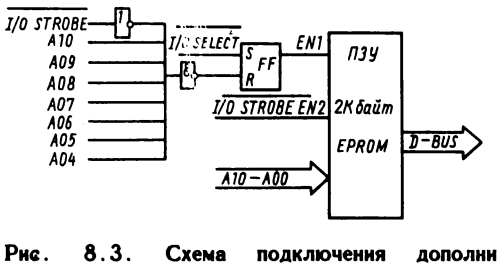


Рис. 8.3. Схема подключения дополнительной памяти

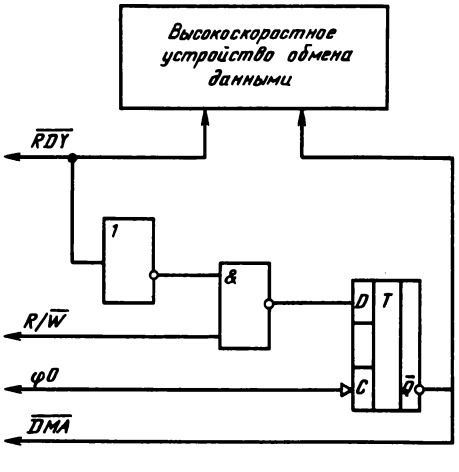


Рис. 8.4. Прямой доступ к памяти

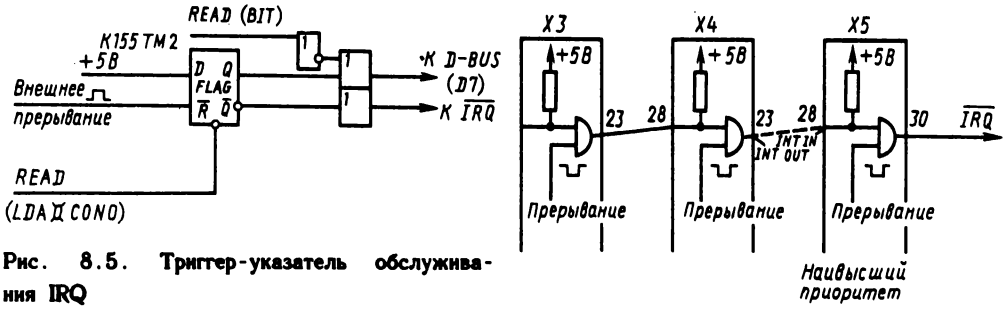


Рис. 8.5. Триггер-указатель обслуживания IRQ

Рис. 8.6. Организация приоритета обработки прерывания

нительное ПЗУ, является сигнал I/O STR. Признаком выключения дополнительного ПЗУ может быть программное обращение к ячейке CFFF.

Система прерываний. Микропроцессор 6502 допускает два типа прерывания: маскируемое прерывание ($\overline{IRQ}$) и немаскируемое (NMI). Шины прерываний - общие для всех периферийных разъемов.

Чтобы программа обработки прерывания смогла найти периферийное устройство, вызвавшее прерывание, каждый контроллер внешнего устройства должен иметь триггер-указатель (рис. 8.5), который устанавливается в единицу при выработке сигнала прерывания.

Состояние этого триггера анализируется командой BIT. После обслуживания прерывания триггер должен быть сброшен.

Когда в системе имеется несколько устройств, работающих с прерыванием, обслуживающая программа должна проверять периферийные устройства в определенном порядке, чем фиксируется приоритет каждого устройства. Первое проверяемое устройство будет иметь наивысший приоритет.

Сигналы INT IN и INT OUT (аналогично DMA IN и DMA OUT) дают возможность реализовать простую систему приоритета (рис. 8.6), в которой запрос на прерывание данного устройства блокирует запросы на прерывание устройств низших иерархических уровней.

8.2. МОДУЛЬ ЦЕНТРАЛЬНОГО ПРОЦЕССОРА

Модуль центрального процессора представляет собой функционально законченный узел, размещенный на отдельной плате (размер 240x120 мм) в разъеме X2 объединительной платы системного блока. Модуль работает в составе ПЭВМ и без него компьютер неработоспособен.

Модуль центрального процессора выполняет следующие функции: выдачу адресов чтения-записи информации по 16-разрядной шине адреса (ША); прием, обработку и выдачу информации по 8-разрядной шине данных (ШД); формирование сигналов дешифратора адреса, необходимых для работы всех периферийных модулей компьютера; организацию диалога с пользователем посредством системного монитора, записанного в ПЗУ модуля.

Технические данные модуля

Напряжение питания, В .....-5 ±0,5

Сила потребляемого тока, А 1,2

Время выполнения операции типа регистр, мкс ... 2

Емкость внутреннего ПЗУ, К байт 2-12

Рассмотрим функциональную схему модуля (рис. 8.7).

Микропроцессор 6502 представляет собой регистровое АЛУ с жесткой логикой. Имеет отдельную шину адреса и шину данных, которые связаны с шинными формирователями адреса и данных. Тактовая система микропроцессора воспринимает задающие импульсы $\varphi 0$, определяющие рабочую частоту работы микропроцессора 1 МГц, и вырабатывает переменный сигнал R/$\overline{W}$, задающий время считывания - записи в каждый цикл работы АЛУ (рис. 8.8).

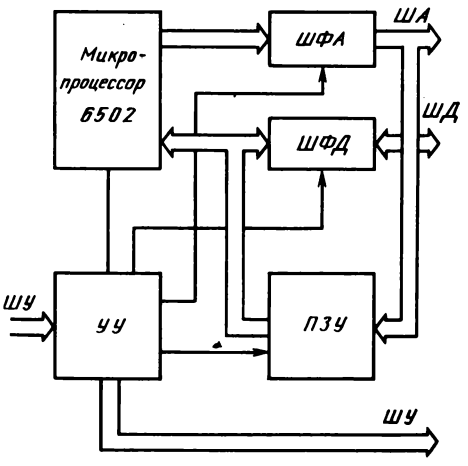
Микропроцессор **6502** - один из семейства микропроцессоров 650X и 651X, разработанных фирмой MOS TEchnology; он аппаратно совместим с микропроцессором 6800 фирмы "Моторола" (Motorola, США), т.е. периферийные контроллеры одной из этих серий могут использоваться и для другой. В системе команд у обоих процессоров есть различия.

Микропроцессор 6502 размещается в корпусе, имеющем 40 выходов (рис. 8.9). Напряжение питания +5 В ±5 % поступает по входу 08. Выполнен по NMOS- и CMOS-технологии, являясь совместимым с элементами ТТЛ-технологии. Назначение входов и выходов:

A0 - A15 (***Address Bus*** - адресная шина) - 16 выходов, по которым микропроцессор передает адрес в память и периферийные устройства.

D0 - D7 (***Data Bus*** - цина данных) - образует двунаправленную восьмиразрядную шину данных, по которой микропроцессор обменивается данными и командами с памятью и периферийными устройствами.

$\overline{RDY}$ (***Ready*** - готовность) - вход, позволяющий остановить работу микропроцессора в любой момент, за исключением цикла записи. Если 6502 производит запись, то она завершается до конца при любом состоянии входа ***$\overline{RDY}$***. Если во время положительной фазы $\varphi 1$ на вход ***$\overline{RDY}$*** поступает перепад напряжения с высокого на низкий уровень, то микропроцессор останавливается, выставляя на ***ША*** текущий адрес. Если нуль пришел во время цикла записи, то запись завершается, а в следующем цикле микропроцессор остановится.



$\overline{IRQ}$ (***Interrupt Reguest*** - запрос прерывания) - вход маскируемого прерывания, чувствительного к низкому уровню напряжения (логическому нулю).

Рис. 8.7. Функциональный состав ячейки процессора:
ШФА, ШФД - шинные формирователи адреса и данных; ША, ШД, ШУ - шины адреса, данных и управления; УУ - устройство управления; ПЗУ - постоянное запоминающее устройство

$\overline{NMI}$ (*Non Maskable Interrypt* - немаскируемое прерывание) - вход, чувствительный к перепаду напряжения высокого на низкий уровень.

Оба эти прерывания стробируются импульсами $\varphi 2$, микропроцессор начинает обслуживать прерывание с начала первой фазы $\varphi 2$ после завершения текущей команды.

S.O.(*Stt Overflow* - устанавливает в логическую единицу флажок V переполнения регистра слова состояния микропроцессора) - вход, устанавливающий V в логическую единицу по перепаду напряжения с высокого на низкий уровень. Стробируется импульсами $\varphi 1$. В ПЭВМ "Агат" этот вход не используется.

SYNC - выход, который находится в состоянии логической единицы во время положительной фазы $\varphi 1$ с начала подцикла выборки микропроцессора и до начала фазы $\varphi 2$. Если на входе $\overline{RDY}$ подан нуль, то выход *SUNC* находится в состоянии единицы до тех пор, пока на вход *RDY* не будет подана логическая единица. В базовом варианте компьютера "Агат" этот вход не задействован.

$\overline{RES}$ (*$\overline{RESET}$* - начальная установка) - вход для установки начального состояния микропроцессора при включении напряжения или после ликвидации ситуации сбоя. Пока на входе $\overline{RES}$ имеется логический нуль, микропроцессор находится в состоянии ожидания. При появлении положительного фронта на входе $\overline{RES}$ микропроцессор начинает выполнять последовательность команд начальной установки. До момента начала работы микропроцессор пропускает шесть тактов, затем устанавливает флажок блокировки прерывания (*I*) в единицу и. извлекает адрес первой выполняемой команды. Заметим, что в момент включения ПЭВМ в сеть линия $\overline{RES}$ находится в состоянии логического нуля два такта; за это время напряжение на входе V достигает +4,75 В и сигналы *SYNC* и $R/\overline{W}$ стабилизируются.

$R/\overline{W}$ (*Read/Write* - чтение - запись) - выход (рис. 8.9), который используется для управления процессами обмена информацией между микропроцессором и памятью или периферийным устройством. При $R/\overline{W}$ = 1 микропроцессор читает данные в памяти или внешнем устройстве, а при $R/\overline{W}$ = 0 - записывает данные.

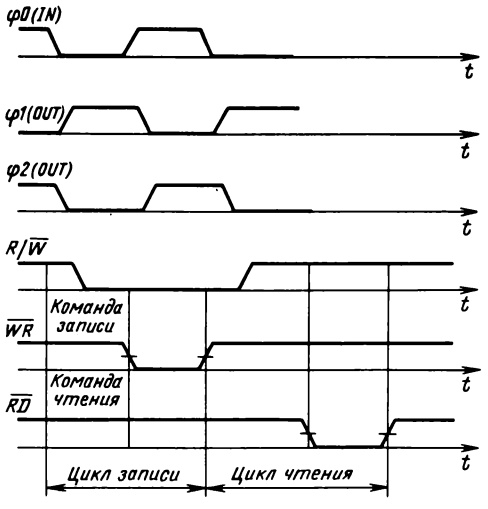


Рис. 8.8. Тактовая система микропроцессора 6502

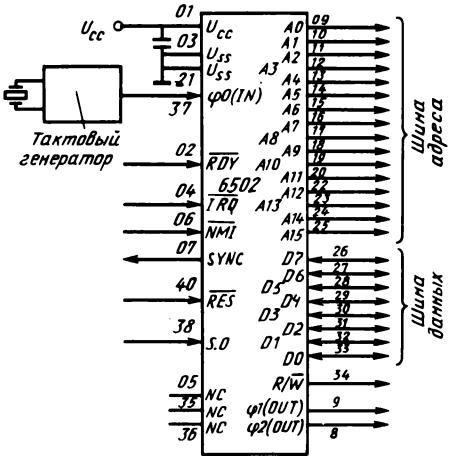


Рис. 8.9. Обозначение выводов микропроцессора 6502

$\varphi 0$ (*IN*) - вход импульсов тактовой частоты 1 МГц; этими импульсами осуществляется синхронизация работы микропроцессора.

$\varphi 1$ (*OUT*), $\varphi 2$ (*OUT*) - выходы тактовой частоты 1 МГц, применяются для синхронизации и управления памятью и внешними устройствами. В ПЭВМ "Агат" эти выходы не используются.

U - напряжение питания +5 В ±5 %.

U_{SS} - общий.

Ns - вывод не используется.

Отметим, что тактовая частота микропроцессоров серии 650X различна для разных микросхем этой серии.

Если в ПЭВМ "Агат" используется 6502A или 6502B, то быстродействие компьютера можно увеличить, увеличив тактовую частоту тактового генератора до 6 МГц.

Сигнал $\overline{RESET}$ по входу $\overline{RES}$ инициирует работу микропроцессора по соответствующей пусковой последовательности "холодного" или "теплого" старта.

После прохождения этапов "холодного" или "теплого" старта дальнейшая работа микропроцессора состоит в поочередной выборке команд и их реализации.

Перед выбором команды микропроцессор опрашивает вход $\overline{RDY}$, проверяя готовность внешних устройств к работе:

если $\overline{RDY}$ = 0, то микропроцессор переходит в режим останова с постоянным опросом входа $\overline{RDY}$;

как только $\overline{RDY}$ = 1 (внешние устройства включены, или закончили предыдущую работу, или в них устранена аварийная ситуация), микропроцессор переходит к циклу выборки и выполнения очередной команды;

если внешние устройства не вызывали останова микропроцессора, то он опрашивает свой вход $\overline{NMI}$;

если вход $\overline{NMI}$ = 0, то микропроцессор прекращает обработку своей программы и начинает выполнение подпрограммы прерывания;

если вход $\overline{NMI}$ = 1, то микропроцессор переходит в следующий цикл и опрашивает свой вход $\overline{IRQ}$; если вход $\overline{IRQ}$ = 0 и флажок блокировки прерывания сброшен (*I* = 0), то микропроцессор переходит к подпрограмме прерывания, прекращая выполнять текущую программу;

если $\overline{IRQ}$ = 0 и *I* = 1 (маска установлена), то микропроцессор игнорирует это прерывание и продолжает свою работу.

Принципиальная схема модуля приведена на рис. 8.10.

Шинный формирователь адреса (ШФА) построен на четырех микросхемах D9 - D12 (К589АП16). Микросхема К589АП16 - элемент с тремя состояниями, управляемый по входам *BK* (выбор кристалла) и *BH* (выбор направления). На вход *BH* постоянно подается нуль, что открывает шину адреса для микропроцессора в прямом направлении. В зависимости от состояния входа *BK* микросхемы либо передают данные от микропроцессора на шину адреса (*BK* = 0), либо находятся в состоянии высокого импеданса (*BK* = 1), т.е. отключают микропроцессор от шины адреса в режиме прямого доступа к оперативной памяти. В этом случае устройство, осуществляющее режим прямого доступа оперативной памяти, формирует

сигнал $D\overline{M}\overline{A}$ (с низким уровнем на линии $D\overline{M}\overline{A}$). Инвертированный на D2.6 этот сигнал логической единицы отключает ШФА от микропроцессора (***ВК*** = 1).

Шинный формирователь данных (ШФД) построен на двух элементах ***D13 и D14*** (К589АП16). ШФД (в отличие от ШФА) может находиться в трех режимах. Управление осуществляется по входам ***ВК*** и ***ВН*** сигналами $\overline{DMA}$, $R/\overline{W}$, поступающими с устройства управления модуля центрального процессора. Логическая единица на входе ***ВК*** переводит элементы ***D13*** и ***D14*** в состояние высокого импеданса, тем самым отключая ***ШФД*** от шины данных микропроцессора.

Состояние входа ***ВН*** определяет работу ***ШФД*** в прямом или обратном направлении. Определяется этот вход сигналом ***ВНШД***, формируемым на элементах ***D6.1***, ***D3.1***, ***D2.2***, входящих в состав устройства управления (УУ), а также сигналами $R/\overline{W}$ и $\varphi 1$.

При обращении к памяти или к внешним устройствам в режиме чтения микропроцессор формирует (см. рис. 8.8) высокий уровень напряжения на линии $R/\overline{W}$. Этот сигнал поступает на УУ, где формируется высокий уровень сигнала ***ВНШД***. Логическая единица, поступившая на вход ***ВН***, разрешает прохождение данных на вход микропроцессора.

В режиме записи микропроцессор формирует низкий уровень напряжения на выходе $R/\overline{W}$, который, поступая на вход ***ВН***, открывает ***ШФД*** для передачи данных от микропроцессора в память или на внешнее устройство.

Устройство управления (УУ) реализовано на элементах комбинационной логики *D*1, *D*2, *D*3, *D*6 и дешифраторе *D*4.

Кроме описанных выше сигналов УУ вырабатывает сигналы $\overline{I/OSTRI}$ и $\overline{I/OSTR}$.

$\overline{I/OSTRI}$ - выбор внешних устройств. Работая с внешними устройствами, центральный процессор использует диапазон адресов C000 - C7FF, характеризующийся постоянным состоянием разрядов шины адреса центрального процессора:

A14 = 1, A15 = 1, A13 = 0, A12 = 0, A11 = 1.

Таким образом, эта комбинация на адресной шине, инициирующая обращение микропроцессора к внешнему устройству, поступает на вход дешифратора ***D4***. С его выхода ***9*** логический нуль проходит через ***D2.4*** на ***D5.1***, где стробируется импульсом тактовой частоты $\varphi 0$, т.е. элемент ***D5.1*** формирует переменный сигнал $\overline{I/O\ STR\ I}$, повторяющий фактически $\varphi 0$. Заметим, что на входе ***3*** элемента ***D5.1*** постоянно присутствует нуль

Сигнал $\overline{I/O\ STRI}$ поступает через разъем ***X2*** на объединительную плату, где и происходит формирование сигнала $\overline{DS}$ для каждого из остановившихся разъемов (***X1, X3 - X7***).

$\overline{I/O\ STR}$ - строб ввода/вывода. В отличие от сигнала $\overline{DS}$, который присутствует на одном из выбранных разъемов, $\overline{I/O\ STR}$ является общим для всех разъемов (кроме ***X2***). Он определяет предпочтительность работы внешнего устройства. Формируется строб на выходе ***10*** дешифратора *D*4 при комбинации 11001 старших разрядов ***A15 - A11***, что соответствует обращению микропроцессора к диапазону адресов ***C800 - C8FF***.

УУ усиливает сигнал чтения - записи ($R/\overline{W}$), поступающий от микропроцессора. С выхода ***D6,1*** сигнал $R/\overline{W}$ подается на разъем ***X1.A***.

Как уже отмечалось выше, модуль центрального процессора может иметь 1 - 6 микросхем ПЗУ, с записанным в них системным программным обеспечением. В

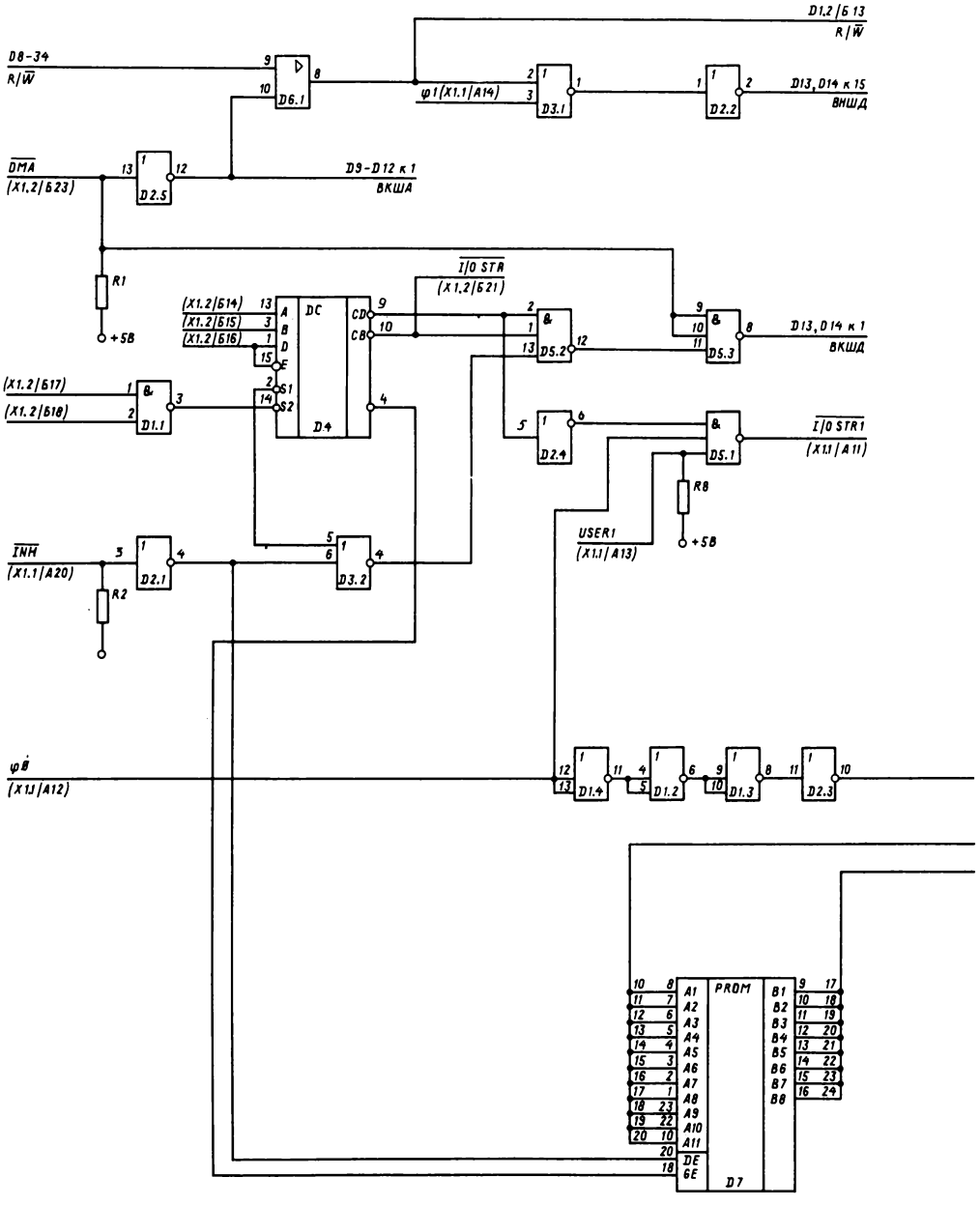
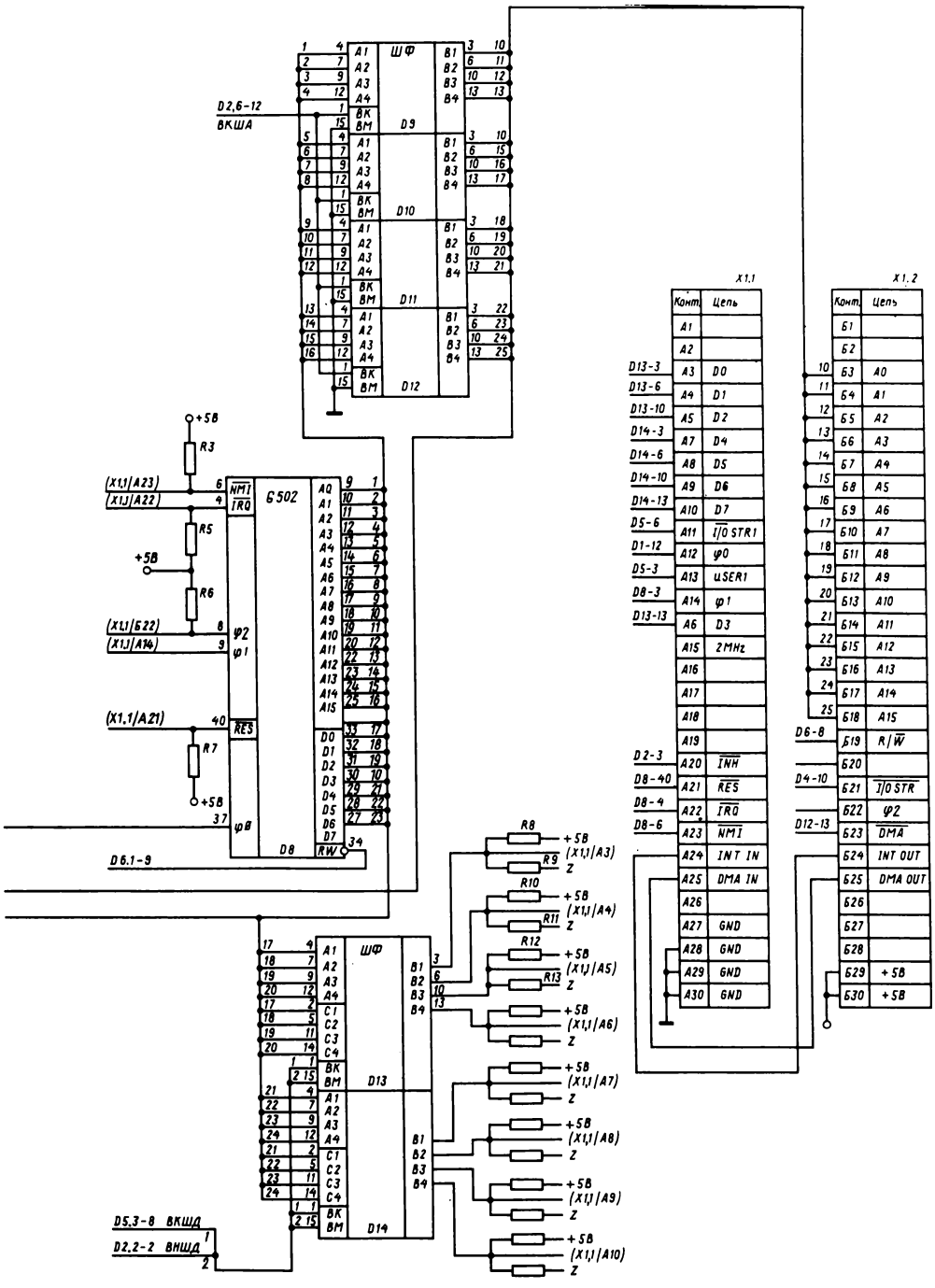


Рис. 8.10. Принципиальная схема модуля центрального процессора


ШФ
D9
D10
D11
D12
D2,6-12
ВКША
+5В
R3
R5
R6
R7
G502
(X1,1/A23)
(X1,1/A22)
(X1,1/A34)
(X1.1/A21)
D6.1-9
D8
D13
D14
D5,3-8 ВКШД
D2.2-2 ВНШД
R8
R9
R10
R11
R12
R13
(X1,1/A3)
(X1,1/A4)
(X1,1/A5)
(X1,1/A6)
(X1,1/A7)
(X1,1/A8)
(X1,1/A9)
(X1,1/A10)
X1.1
X1.2
Конт.
Цепь
D13-3
D13-6
D13-10
D14-3
D14-6
D14-10
D14-13
D5-6
D1-12
D5-3
D8-3
D13-13
D2-3
D8-40
D8-4
D8-6
D6-8
D4-10
D12-13
INH
RES
IRQ
NMI
INT IN
DMA IN
GND
I/O STR
DMA
INT OUT
DMA OUT

базовом варианте ПЭВМ на плате центрального процессора имеется только одна микросхема ПЗУ - ***D7***, которая хранит программу "Системный монитор". Для использования системных программ следует от микропроцессора отключать шину данных центрального процессора. При обращении микропроцессора по адресам этих программ необходимо подключить его шины данных к одной из микросхем ПЗУ. Рассмотрим, как это происходит в базовом варианте ПЭВМ.

Как известно, "Системный монитор" располагается в диапазоне адресов F800 - FFFF. При обращении по любому из этих адресов пять старших разрядов A15 - A11 оказываются в состоянии логической единицы. Это означает, что на выходе ***4*** дешифратора ***D4*** будет находиться логический нуль, как только на его адресный вход поступит комбинация 11111. Этот нулевой сигнал подается на вход ***GE*** (ножка ***18***) микросхемы ***D7***. Другой нулевой сигнал поступает с выхода схемы ***D2.1***, на вход которой подается высокий уровень сигнала $\overline{INH}$. Оба нулевых сигнала включают D7 в работу. На адресные входы D7 микропроцессор выставляет адрес нужной ячейки, а на выходах этой микросхемы появляются считываемые данные, поступающие на ШД микропроцессора. Адрес на ***D7*** поступает через ***ШФА***. Для блокировки ***ШФД*** на время работы ***D7*** используется сигнал ***ВКШД*** = 1. Формирование сигнала осуществляется на ***D5.3*** нулевыми сигналами, поступающими с выходов ***9*** и ***10*** дешифратора, и инверсным высоким уровнем ***INH***.

При обращении центрального процессора к псевдоПЗУ по адресам системного монитора на шине ***INH*** выставляется логический нуль, который, инвертируясь на D2.1, блокирует работу по входу ***DE*** (ножка ***20***). Этот же сигнал на шине $\overline{INY}$ формирует сигнал ***ВКШД***-1, блокирующий работу ***ШФД***.

Список применяемых микросхем модуля центрльного процессора приведен ниже.

Обозначение на схеме	*Тип элемента*
D1	К555ЛА3
D2	К555ЛН1
D3	К555ЛЕ1
D4	К555ИД4
D5	К555ЛА4
D6	К155ЛП8
D7	К573РФ2
D8	МП6502
D9 - D14	К589АП16

На рис. 7.17 приведена функциональная схема дешифратора D4.

Рассмотрим, каким образом элемент ***D7*** используется в качестве носителя "Системного монитора". Для выбранных адресов этой микросхемы памяти указан код информации записи (для программирования и контроля) в виде двух шестнадцатеричных цифр. Каждая цифра шестнадцатеричного кода информации записи соответствует двоичному числу на выводах ***9, 10, 11, 13*** (правая цифра), на выводах ***14, 15, 16, 17*** (левая цифра) микросхемы ***D7***. При этом выводы ***5 - 8*** соответствуют младшим разрядам кода адреса записываемой информации, а выводы ***19, 22, 23*** - коду старших разрядов адреса информации записи. Таким образом, в микросхему записывается 2К байт данных. Перепрограммирование микросхемы возможно после стирания старой информации ультрафиолетовым излучением, например, на установке ТФМЗ.857209.

8.3. МОДУЛЬ ИНТЕРФЕЙСА И ПАМЯТИ

УЗЛЫ МОДУЛЯ

На модуле интерфейса и памяти расположены три функциональных блока машины (рис. 8.11): оперативная память *ОП*; дисплейный контроллер *ДК*; встроенный интерфейс ввода-вывода *ВИ В/В*. Кроме того, на этом модуле размещены разъемы, на которых реализован внешний интерфейс ПЭВМ. Обмен информацией между центральным процессором и функциональными блоками модуля обеспечивается шиной данных и шиной адресов центрального процессора (соответственно *ШД ЦП* и *ША ЦП*).

Оперативная память. Оперативная память связана с *ДК* так же, как и с центральным процессором, шиной адреса (*ША ДК*) и шиной данных (*ША ДК*). По *ША ДК* от *ДК* в *ОП* передаются 16-разрядные коды адреса, формируемые *ДК*, а по *ШД ДК* из *ОП* в *ДК* поступают 16-разрядные коды данных, которые после соответствующей обработки *ДК* выводит на экран видеоконтрольного устройства *ВКУ*. Следует отметить, что *ДК* только считывает информацию из *ОП* и поэтому *ША ДК* и *ШД ДК* являются однонаправленными (рис. 8.12).

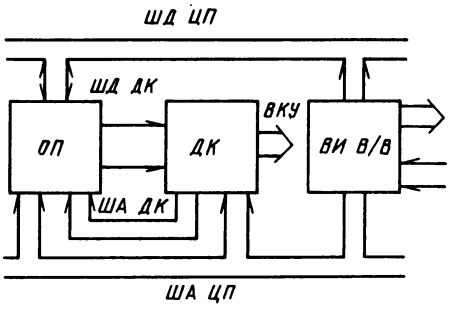


Рис. 8.11. Функциональная схема ячейки памяти и интерфейса

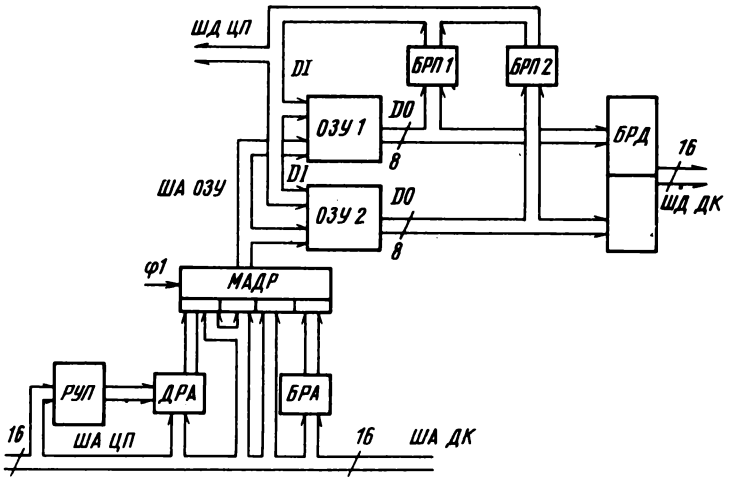


Рис. 8.12. Функциональная схема оперативной памяти:
ШД ЦП - шина данных центрального процессора; ША ОЗУ - шина адреса оперативного запоминающего устройства; БРП1, БРП2 - буферные регистры памяти; ОЗУ1, ОЗУ2 - оперативные запоминающее устройства; БРД - буферный регистр данных; ШД, ДК - шина данных дисплейного контроллера; МАДР - мультиплексор адреса; РУП - регистр управления памятью; ДРА - логика формирования дополнительных разрядов адреса; БРА - буферный регистр адреса; DO - шина данных "чтения"; DI - шина данных "записи"; ША ЦП - шина адреса центрального процессора; ШАДК - шина адреса дисплейного контроллера

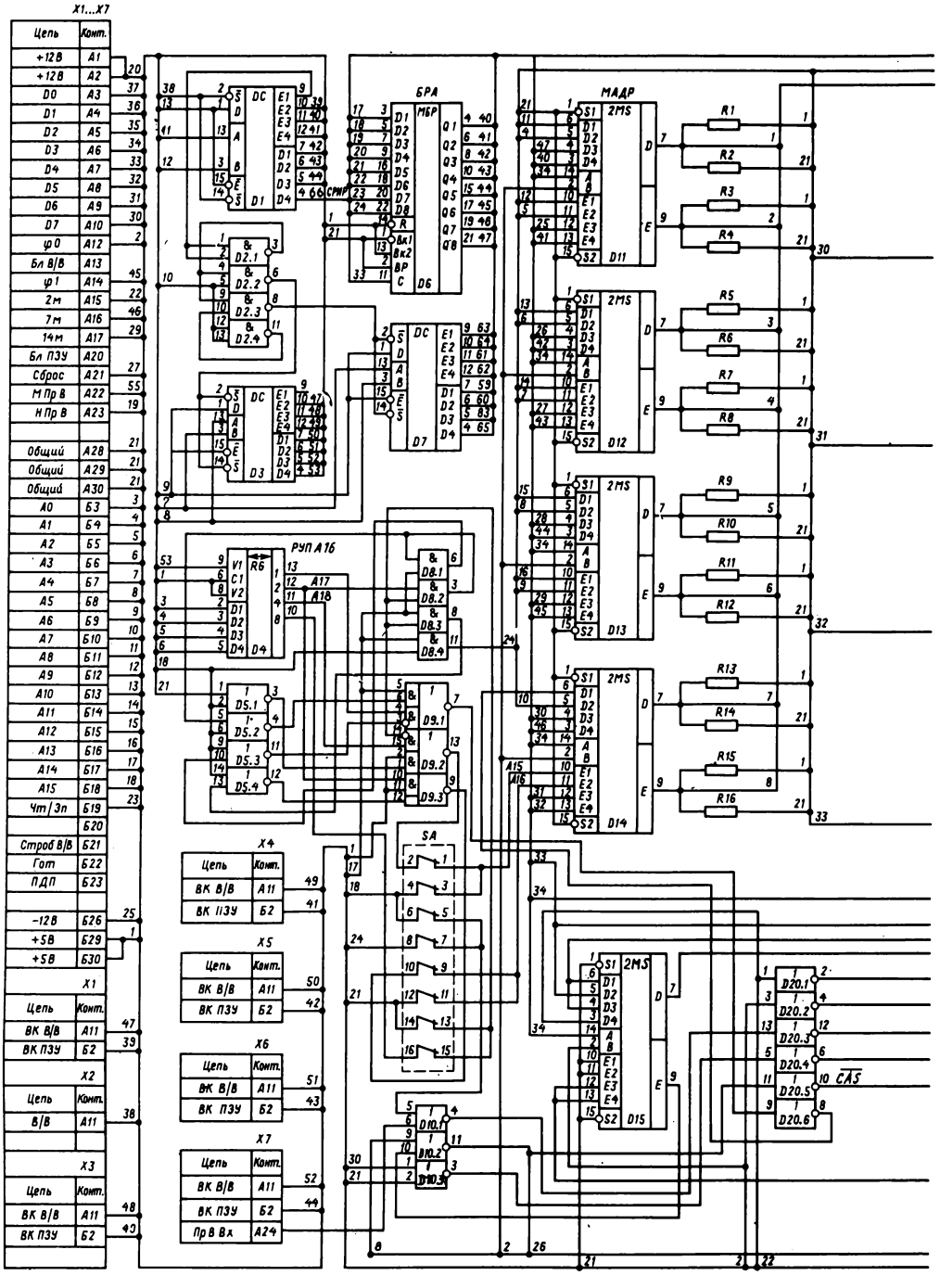
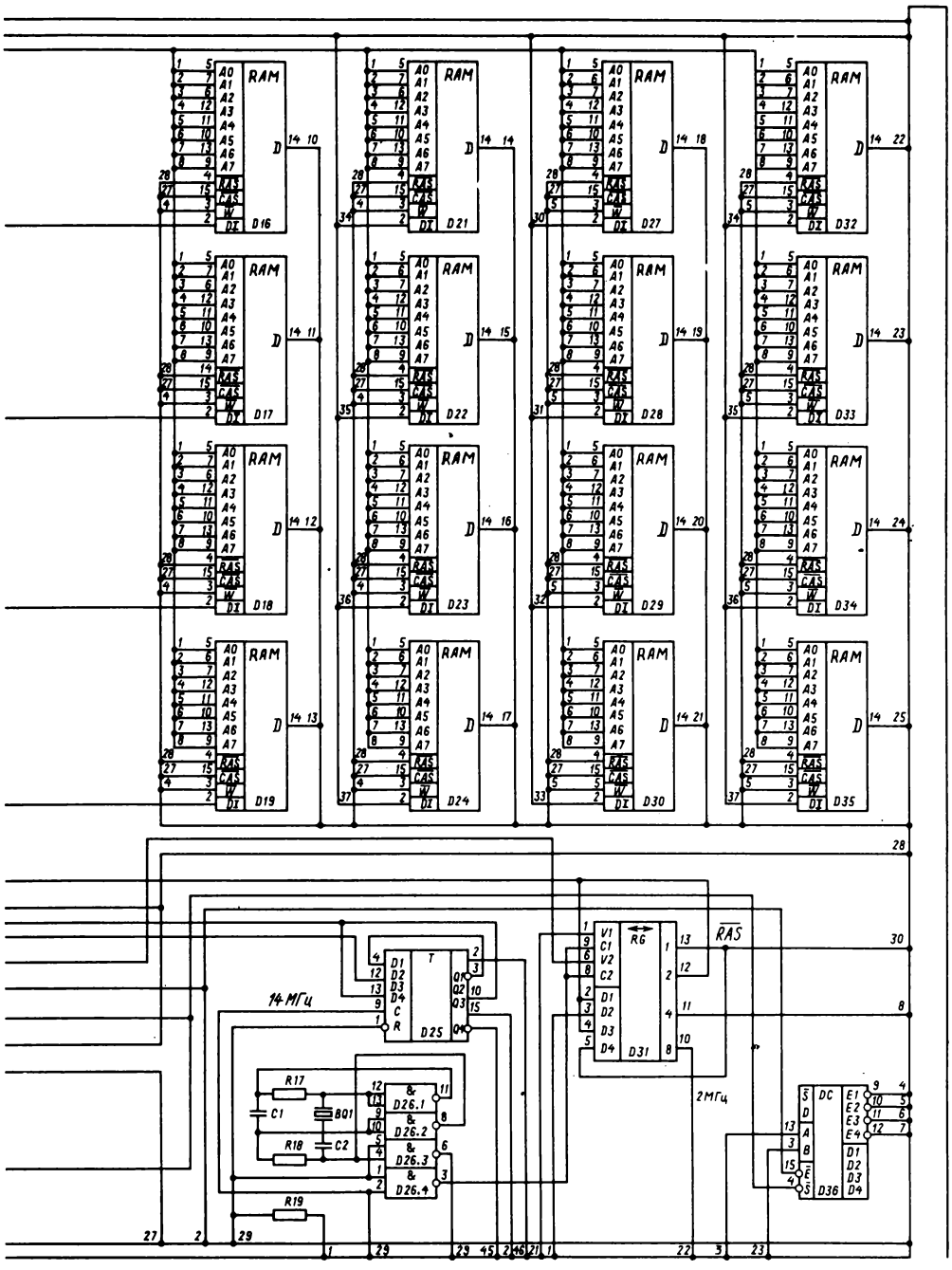


Рис. 8.13. Принципиальная схема модуля памяти и интерфейса


RAM
D16
D17
D18
D19
D21
D22
D23
D24
D27
D28
D29
D30
D32
D33
D34
D35
14 МГц
T
D25
RG
D31
RAS
2 МГц
R17
R18
R19
C1
C2
BQ1
D26.1
D26.2
D26.3
D26.4
D36

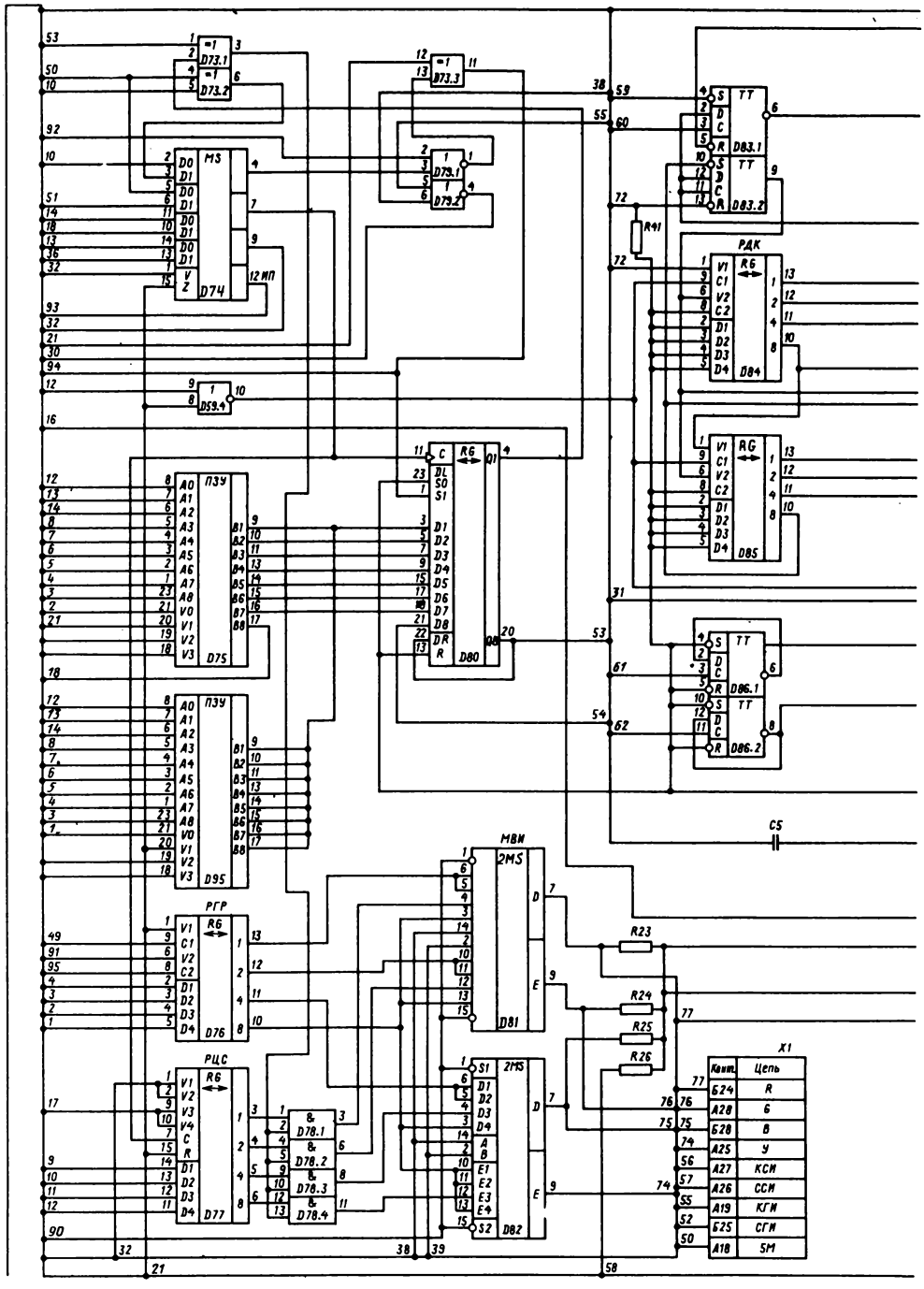
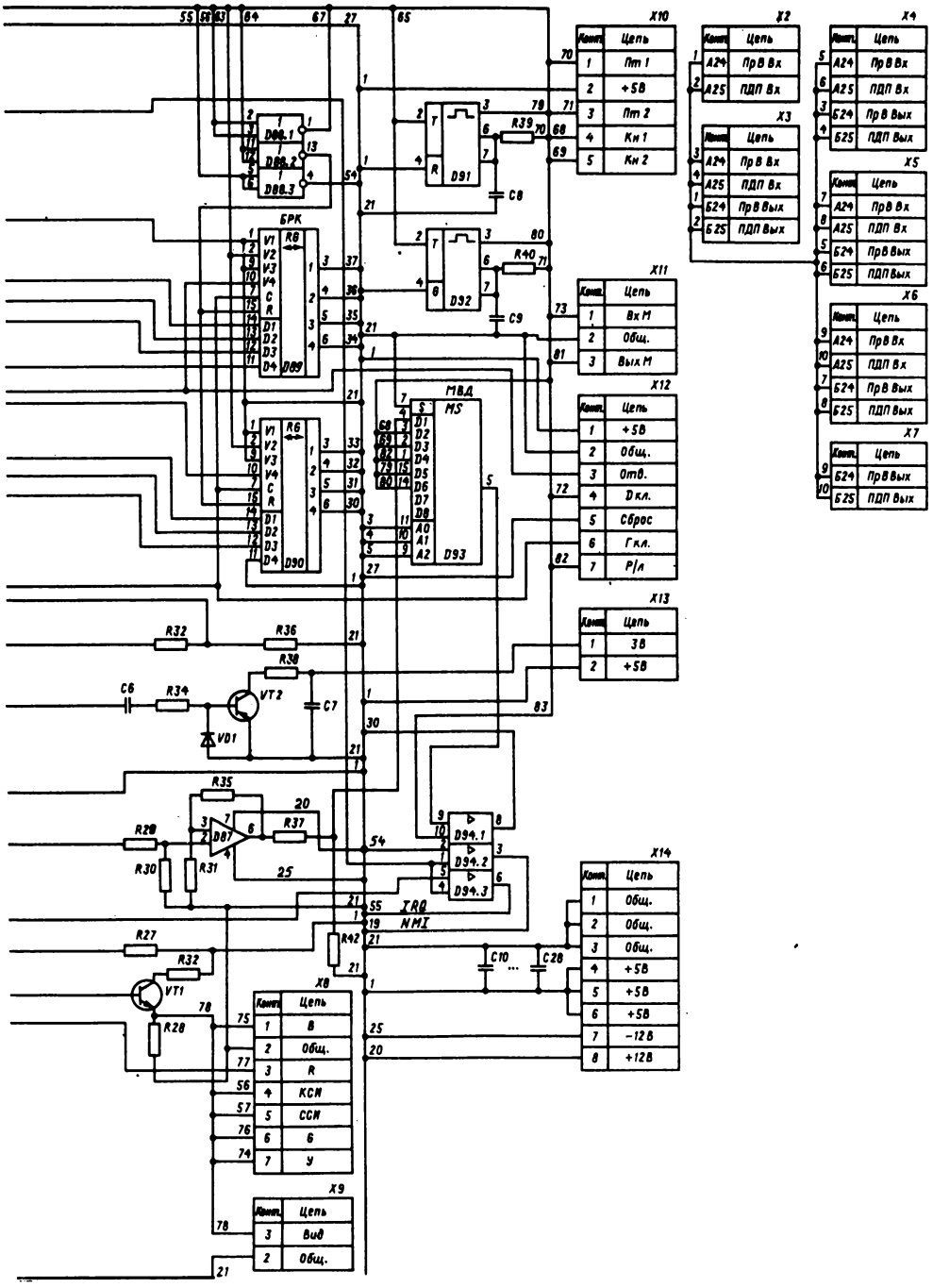


Рис 8.13. Продолжение


X10
Конт. Цепь
1 Пт 1
2 +5В
3 Пт 2
4 Кн 1
5 Кн 2
X2
Конт. Цепь
A24 ПрВ Вх
A25 ПДП Вх
X3
Конт. Цепь
A24 ПрВ Вх
A25 ПДП Вх
Б24 ПрВ Вых
Б25 ПДП Вых
X4
Конт. Цепь
A24 ПрВ Вх
A25 ПДП Вх
Б24 ПрВ Вых
Б25 ПДП Вых
X5
Конт. Цепь
A24 ПрВ Вх
A25 ПДП Вх
Б24 ПрВ Вых
Б25 ПДП Вых
X6
Конт. Цепь
A24 ПрВ Вх
A25 ПДП Вх
Б24 ПрВ Вых
Б25 ПДП Вых
X7
Конт. Цепь
Б24 ПрВ Вых
Б25 ПДП Вых
X11
Конт. Цепь
1 Вх М
2 Общ.
3 Вых М
X12
Конт. Цепь
1 +5В
2 Общ.
3 Отв.
4 Dкл.
5 Сброс
6 Гкл.
7 Р/л
X13
Конт. Цепь
1 3В
2 +5В
X14
Конт. Цепь
1 Общ.
2 Общ.
3 Общ.
4 +5В
5 +5В
6 +5В
7 −12В
8 +12В
X8
Конт. Цепь
1 В
2 Общ.
3 R
4 КСИ
5 ССИ
6 G
7 У
X9
Конт. Цепь
3 Вид
2 Общ.
БРК
МВД
D88.1
D88.2
D88.3
D89
D90
D91
D92
D93
D94.1
D94.2
D94.3
D87
VT1
VT2
VD1
R6
R8
R20
R27
R28
R30
R31
R32
R34
R35
R36
R37
R38
R39
R40
R42
C6
C7
C8
C9
C10 ... C28
IRQ
NMI

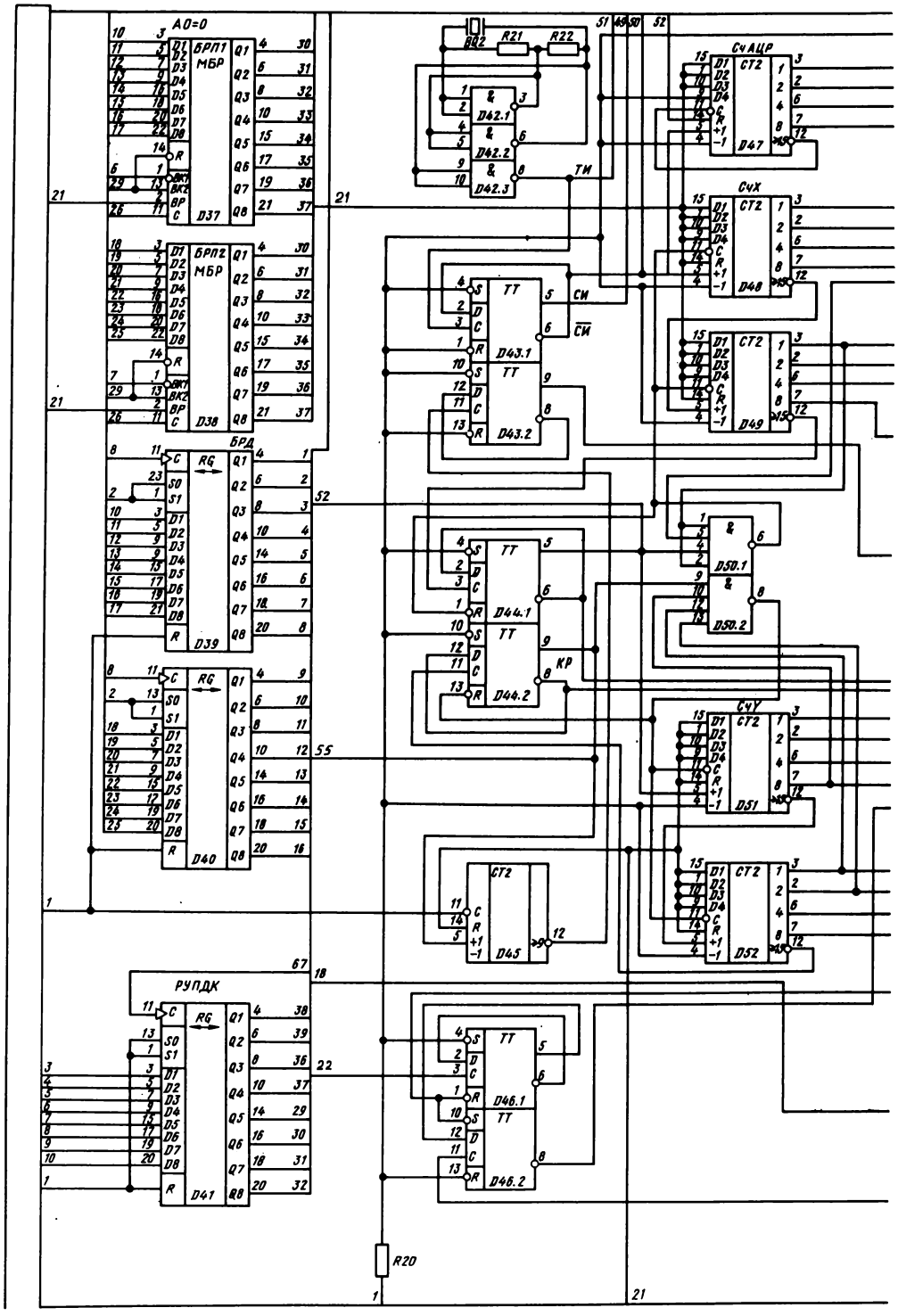
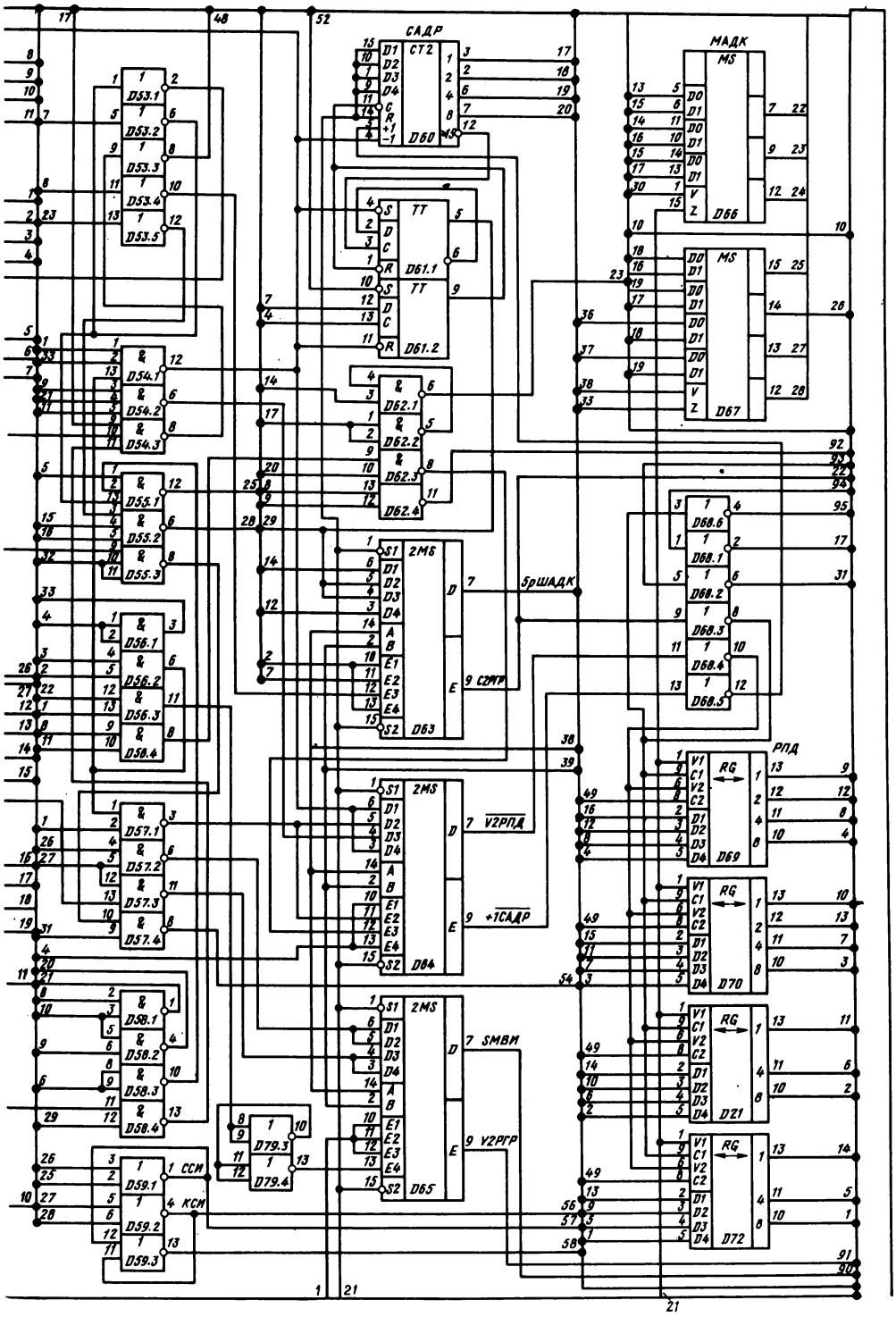


Рис 8.13. Продолжение


САДР
МАДК
РПД
D53.1
D53.2
D53.3
D53.4
D53.5
D54.1
D54.2
D54.3
D55.1
D55.2
D55.3
D56.1
D56.2
D56.3
D56.4
D57.1
D57.2
D57.3
D57.4
D58.1
D58.2
D58.3
D58.4
D59.1
D59.2
D59.3
D60
D61.1
D61.2
D62.1
D62.2
D62.3
D62.4
D63
D64
D65
D66
D67
D68.1
D68.2
D68.3
D68.4
D68.5
D68.6
D69
D70
D21
D72
D79.3
D79.4
5ршадк
9 ссмр
7 V2рпд
9 +1САДР
7 SMBM
9 V2PGP
1 ССМ
4 КСМ

Оперативная память модуля обеспечивает обслуживание двух пользователей (дисплейного контроллера и центрального процессора), организуется попеременно на основе разделения фаз. Микропроцессор в течение действия положительного уровня импульсов $\varphi 0$ (фазы процессора получает доступ к оперативной памяти, а в течение действия отрицательного уровня импульсов $\varphi 1$ осуществляет внутреннюю обработку и на это время отключается от оперативной памяти. Это позволяет дисплейному контроллеру во время действия низкого уровня импульсов $\varphi 0$ (фазы ***ДК***) получить доступ к оперативной памяти. Принципиальная схема модуля приведена на рис. 8.13.

Генератор тактовых импульсов. Генератор собран на кварцевом генераторе ***BQ1***, усилительных элементах ***D15, D20, D25, D31, D26*** и элементах ***D10, D36***, синхронизирующих циклы чтения и записи.

Генератор тактовых импульсов задает тактовые импульсы всех рабочих частот. Кварцевый генератор ***BQ1*** вырабатывает сигнал постоянной частоты 14 МГц, который усиливается четырьмя элементами ***D26***. С выхода ***6*** элемента ***D26*** сигнал поступает на вход ***9*** триггера ***D25***, который работает как делитель частоты в число раз, кратное двум. Таким образом, с выхода ***2*** элемента ***D25*** снимается импульс частотой 7 МГц, а с выходов ***15*** и ***14*** соответственно импульсы основной тактовой частоты $\varphi 0$ (1 МГц) и $\varphi 1$ (800 кГц). С выхода ***3*** триггера ***D26*** импульсы частоты 14 МГц поступают на управляющие входы регистра сдвига ***D31***. Этот элемент формирует импульсы, управляющие работой микросхем памяти ($\overline{CAS}$ - строб выбора столбца, $\overline{RAS}$ - строб выбора строки), которые усиливаются двумя инверторами ***D10*** (выходы ***3*** и ***11***) и ***D20*** (выходы ***10*** и ***6***) и заводятся на входы ***15*** и ***4*** всех микросхем памяти.

Регистр сдвига ***D31*** на выходе ***10*** формирует сигнал вспомогательной частоты 2 МГц. Коммутатор ***D15*** пропускает сигналы смешанной частоты, задавая режим работы ***D10*** и ***D25***.

Регистр управления памятью. Это четырехразрядный регистр, к информа-

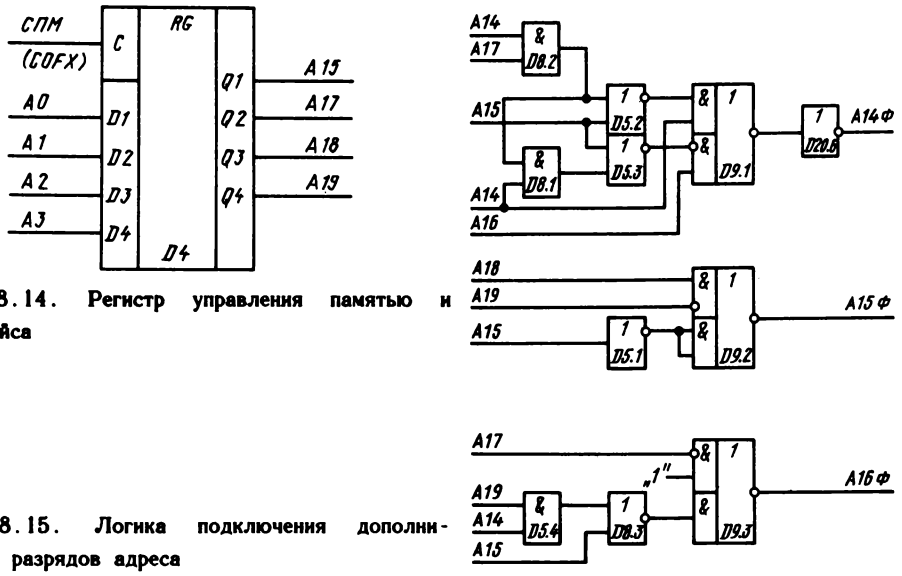


Рис. 8.14. Регистр управления памятью и интерфейса

Рис. 8.15. Логика подключения дополнительных разрядов адреса

ционным входам которого подключены младшие разряды адреса ***A0 - A3 ША ЦП*** (рис. 8.14). Управление приемом информации в регистр осуществляется с помощью специального сигнала ***СПМ***. Сигнал ***СПМ*** вырабатывается на ***D3*** (выход ***4***, см. рис. 8.13) в случае, когда на остальных разрядах адреса ***A4 - A15 ША ЦП*** будет установлена комбинация COF (1100 0000 1111), т.е. при обращении центрального процессора по адресу программного переключателя ***COFX*** младшие четыре разряда фиксируются в регистре управления памятью. Назначение выходных сигналов ***A16 - A19*** следующее:

A16 - дополнительный разряд адреса, обеспечивает подключение первого дополнительного массива емкостью 16К байт, а также управляет подключением 2-го или 3-го и 4-го или 5-го дополнительных массивов;

A17 - дополнительный разряд адреса, обеспечивает подключение дополнительного ОЗУ емкостью 64К байт (2-го - 5-го дополнительных массивов), в случае варианта с общей емкостью - ОЗУ в 128К байт;

A18 - дополнительный разряд адреса, обеспечивает подключение соответствующих дополнительных массивов (2-го и 3-го или 4-го и 5-го) в случае использования одного из способов (***A19*** = 1) подключения дополнительных массивов; при другом способе (***A19*** = 0) подключения значение A18 безразлично;

A19 - определяет способ подключения дополнительных массивов при использовании ОЗУ емкостью 128, 32 и 16К байт.

Дополнительные разряды адреса ***A17*** и ***A18*** для варианта ОЗУ емкостью 64 и 32К байт не используются, и их состояние в этом случае может быть произвольным.

Соответствующие программные переключатели, задающие дополнительные разряды адреса и управляющие подключением дополнительных массивов, указаны в табл. 8.5.

Логические схемы, обеспечивающие подключение соответствующих дополнительных разрядов адреса (ДРА) к системе адресации ОЗУ, показаны на рис. 8.15.

Таблица 8.5

Подключение дополнительной памяти

Способ подключения дополнительных массивов	Программные переключатели	Состояние выходов РУП	Подключаемые массивы памяти	
		A19 A18 A17 A16	Адрес 4000 - 7FFF	Адрес 8000 - BFFF
	C0F0	0000		Основной
	C0F1	0001		1-й дополнительный
Первый	C0F2	0010		4-й дополнительный
способ	C0F3	0011		5-й дополнительный
подключения	C0F4	0100	Основной	Основной

Продолжение табл. 8.5

Способ подключения дополнительных массивов	Программные переключатели	Состояние выходов РУП	Подключаемые массивы памяти	
		A19 A18 A17 A16	Адрес 4000 - 7FFF	Адрес 8000 - BFFF
	C0F5	0101		1-й дополнительный
	C0F6	0110		2-й дополнительный
	C0F7	0111		3-й дополнительный
	C0F8	1000	Основной	Основной
	C0F9	1001	Основной	1-й дополнительный
Второй	C0FA	1010	2-й дополнительный	4-й дополнительный
способ	C0FB	1011	3-й дополнительный	5-й дополнительный
подключения	C0FC	1100	Основной	Основной
	C0FD	1101	Основной	1-й дополнительный
	C0FE	1110	2-й дополнительный	4-й дополнительный
	C0FF	1111	3-й дополнительный	5-й дополнительный

ФУНКЦИИ МОДУЛЯ

Операция "Чтение". "Чтение" данных из ОЗУ в центральный процессор осуществляется во время положительной фазы импульсов $\varphi 0$ (см. рис. 8.1), которые переключают мультиплексор адреса ***МАДР*** (элементы ***D11 - D14***, см. рис. 8.13) на адреса, поступающие от микропроцессора. В начале фазы процессор выставляет на ***ША ЦП*** адрес строки (младший байт адреса) и адрес столбца (старший байт адреса) опрашиваемой ячейки памяти. Мультиплексор подключает адресную шину (ША) ОЗУ к разрядам ***A1 - A7 ША ЦП*** и разряду ***Ф16Ф***, поступающего с выхода ***D9.3*** (рис. 8.15). Байт адреса поступает на все 16 микросхем ОЗУ и записывается в них по отрицательному перепаду сигнала $\overline{RAS}$. Сигналом регистра сдвига ***D31*** (выход ***12***, см. рис. 8.13) ***МАДР*** переключается на восприятие старшей части адреса, поступающего от центрального процессора, и пропускает разряды ***A8*** -

A13 ЩА ЦП и *A14Ф* (*D9.1*; *D9.2*, см. рис. 8.15 на *ША* ОЗУ. Как и в предыдущем случае, байт адреса столбца поступает на адресные входы всех микросхем памяти и записывается в них по отрицательному перепаду сигнала $\overline{CAS}$ с генератора тактовых импульсов. На этом адресная часть цикла чтения заканчивается.

Высокий уровень сигнала $R/\overline{W}$, поступающего от центрального процессора через дешифратор *D36* (см. рис. 8.13) на все микросхемы памяти (оба сигнала *W1* и *W2* на ножках *9* и *10 D36* имеют высокий уровень), обеспечивает их работу в режиме "чтения". В этом случае через интервал времени, равный времени выборки относительно сигнала $\overline{CAS}$, содержимое адресуемой ячейки установится на выходах всех 16 микросхем ОЗУ. Выбранная из памяти информация поступает на два буферных регистра памяти (*БРП1* и *БРП2*, соответственно *D37* и *D38*), являющиеся буферами между микросхемами памяти и *ШД ЦП*. Буферные регистры имеют постоянный нуль на входе *ВР*, поэтому управление регистрами осуществляется по входам *ВК1* и *С*, которые определяют режим приема или выдачи хранящейся в буфере информации. *БРП* принимают данные от микросхем памяти в течение действия высокого уровня сигнала приема информации в буферный регистр памяти (*ПИБРП*), поступающего на вход *С* с генератора тактовых импульсов (инвертированный сигнал $\overline{CAS}$).

Управление выдачей информации производится высоким уровнем сигнала разрешения выдачи на ШД (*РВШД*), формирующимся на *D36* (выходы *11* и *12*) в соответствии с состоянием младшего разряда адреса выбираемой ячейки памяти. *РВШД* подается на *БРП2*, если *А*0 = -1, и на управляющий вход *БРП1*, если А0 = 0. Таким образом, на шину данных центрального процессора поступает информация из адресуемой центральным процессором ячейки памяти одного из двух массивов ОЗУ (*ОЗУ1* или *ОЗУ2*). На этом цикл чтения заканчивается. Заметим, что 0*ЗУ1* реализовано на *D16 - D24* и хранит информацию в ячейках с четными адресами; *ОЗУ2* реализовано на *D27 - D35* и хранит информацию ячеек памяти с нечетными адресами.

Операция "Запись". Адресная часть цикла "Запись" аналогична адресной части цикла "Чтение". При этом наличие низкого уровня сигнала $R/\overline{W}$ во время процессорной фазы переводит один из массивов ОЗУ (*ОЗУ1* или *ОЗУ2*) в режим "записи". Выбор нужного массива осуществляется в зависимости от состояния разряда адреса А0 *ША ЦП*. Как и в предыдущем случае, младший разряд поступает на вход дешифратора *D36* и на него же приходит низкий уровень сигналов *R/W*,формируемого центральным процессором. На выходе *D36* формируется либо сигнал записи в *ОЗУ1* ($\overline{W}_1$ = 0), либо сигнал записи в *ОЗУ* ($\overline{W}_2$ = 0), соответственно при нулевом или единичном состоянии разряда *А*0. Запись байта данных в ОЗУ производится по сигналу $\overline{CAS}$ непосредственно с *ШД ЦП*. При этом сигнал РВШД на регистры *БРП1* и *БРП2* не поступает, в результате чего они находятся в третьем состоянии, т.е. выходы *БРП1* и *БРП2* отключены от *ШД ЦП* и микросхем памяти. По окончании адресной части цикла байт данных, выставленных центральным процессором на *ШД*, записывается в *ОЗУ1* или *ОЗУ2*. Другое ОЗУ в это время находится в режиме чтения, что необходимо для его регенерации.

Обращение дисплейного контроллера к оперативной памяти. ДК получает доступ к оперативной памяти во время отрицательной фазы импульсов $\varphi 0$ (рис. 8.16). При этом все микросхемы памяти массивов *ОЗУ1* и *ОЗУ2* переводятся в ре-

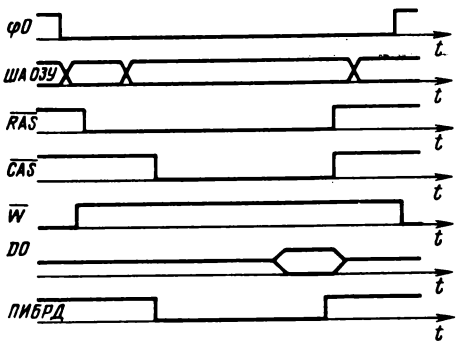


Рис. 8.16. Временные диаграммы обращения дисплейного контроллера

жим считывания, при котором дешифратор ***D36*** (см. рис. 8.13) заблокирован высоким уровнем φ0 и ДК только считывает информацию из оперативной памяти. Единичное состояние выходов ***D36*** также отключают ***БРП1*** и ***БРП2*** от шины данных. Высокий уровень φ1 переключает ***МАДР*** на адреса, поступающие от ***ДК***. Сначала к ***ША ОЗУ*** подключаются восемь разрядов ***ША ДК***, соответствующих сигналу $\overline{RAS}$. Эти разряды адреса ***ДК*** поступают на входы ***МАДР*** через промежуточный буферный регистр ***БРА*** (***D6***). Включение ***БРА*** позволяет исключить возможные изменения адресов ***ДК*** на время приема этих разрядов адреса во внутренний адресный регистр микросхем памяти. После этого к ***ША ОЗУ*** подключаются остальные восемь разрядов адреса ***ША ДК***, которые принимаются во внутренний адресный регистр по срезу сигнала $\overline{CAS}$ (рис. 8.16). Через время, равное времени выборки относительно сигнала $\overline{CAS}$, 2 байт информации из адресуемых ячеек обоих массивов ***ОЗУ*** поступают на 16-разрядный регистр ***БРД*** (***D39*** - ***D40***, см. рис. 8.13) и запоминаются в нем по фронту сигнала ***ПИБРД***, поступающего с элемента ***D31*** генератора тактовых импульсов. На этом цикл обращения ДК к оперативной памяти заканчивается, в результате чего в регистре ***БРД*** хранится 16-разрядное слово, которе поступает в ***ДК*** для дальнейшей обработки. На вход ***S0*** и ***S1 БРД*** приходит положительный уровень импульсов φ1, разрешающий в него запись.

Заметим, что частота стробирующего сигнала ПИБРД составляет 2 МГц.

Регенерация памяти. Микросхемы динамической памяти требуют 128 циклов регенерации. Это выражается перебором 128 возможных вариантов на адресных входах микросхем памяти по сигналу $\overline{RAS}$, причем время перебора должно быть не более 2 мс. Особенности растрового принципа развертки изображения, а также особенности работы ДК при формировании изображения на экране ВКУ позволяют совместить регенерацию памяти с обращением ДК к оперативной памяти.

ДК, формируя изображение на экране ВКУ, последовательно просматривает содержимое всех ячеек памяти выбранного блока видеоОЗУ. При этом счетчик адреса ДК генерирует все 128 комбинаций на разрядах ***A0*** - ***A6 ША ОЗУ***. Таким образом, не требуется никаких специальных аппаратных и временных затрат для организации регенерации микросхем памяти.

ВАРИАНТЫ ИСПОЛНЕНИЯ МОДУЛЯ

Базовая модель компьютера имеет в своем составе модуль памяти и интерфейса с ОЗУ емкостью 32К байт. Память этого модуля построена на микросхемах К565РУ6, имеющих организацию 16Кх1 бит. Кроме микросхем памяти с такой организацией выпускаются элементы К565РУ5Д1 по 32Кх1 бит и К565РУ5 (А, Б, В, Г, Д) по 64Кх1 бит. Применение подобных микросхем позволяет организовать ОЗУ емкостью 64К байт или 128К байт.

Чтобы обеспечить страничную организацию памяти, соответствующую данному объему ОЗУ, необходимо осуществить определенные соединения между контактами движкового выключателя, расположенного на модуле справа от массива микросхем памяти. Варианты замыканий и контактов для каждого исполнения модуля представлены ниже.

Тип применяемых микросхем памяти:	*Замыкание между контактами выключателя* SA
К565РУ6	5 - 6; 13 - 14
К565РУ5Д1	3 - 4; 7 - 8; 11 - 12; 13 - 14
К565РУ5	1 - 2; 7 - 8; 9 - 10; 15 - 16

Другое расположение переключателей может привести к сбою компьютера.

ДИСПЛЕЙНЫЙ КОНТРОЛЛЕР

ПЭВМ предназначена для оперативного вывода информации на экране телевизионного приемника и обеспечивает формирование изображения в одном из следующих режимов: ГВР, ГСР, ГНР, АЦР (см. п. 3.3).

Основные характеристики всех режимов приведены в табл. 3.2.

Управление работой ДК осуществляется с помощью специального регистра РУП ДК, содержимое которого может изменять центральный процессор.

Основные функциональные блоки ДК показаны на рис. 8.17 и описаны в последующих подразделах. Можно выделить две основные составные части ДК: видеогенератор и блок развертки изображения. Видеогенератор обеспечивает прием информации из оперативной памяти и ее преобразование для последующего вывода на телевизионный приемник. Блок развертки изображения осуществляет общее управление функциональными блоками ДК, вырабатывает все необходимые для этого синхронизирующие импульсы, а также импульсы кадровой и строчной синхронизации. Кроме того, блок развертки изображения обеспечивает формирование последовательности адресов ячеек оперативной памяти, перебираемых ДК при развертке изображения на экране телевизионного приемника.

Регистр управления ДК. Регистр управления ***ДК*** (***РУП ДК***) - это восьмиразрядный регистр, к информационным входам которого подключены разряды адреса ***А0 - А7 ША ЦП*** (рис. 8.18). Управление приемом информации в регистр осуществляется с помощью сигнала ***СРИР***. Сигнал ***СРИР*** формируется на выходе ***12*** дешифратора ***D7*** (см. рис. 8.13) в случае, когда на восьми старших разрядах адреса ***А8 - А15*** будет установлена комбинация ***С7*** (11000111), т.е. при обращении центрального процессор по адресу ***С7ХХ*** младшие восемь разрядов адреса фиксируются в РУП ДК. Распределение функций между разрядами РУП ДК следующие:

разряды РВИ обеспечивают управление переключением режимов вывода информации на экран;

разряды ЭПС указывают номер отображаемой экранной подстраницы емкостью 2К байт в пределах экранной страницы емкостью 8К байт при ГНР и АЦР;

разряды ЭС указывают номер отображаемой экранной страницы емкостью 8К байт в пределах оперативной памяти при ГСР и ГВР.

Соответствующие программные переключатели указаны в табл. 8.6; ОЗУ, раз-

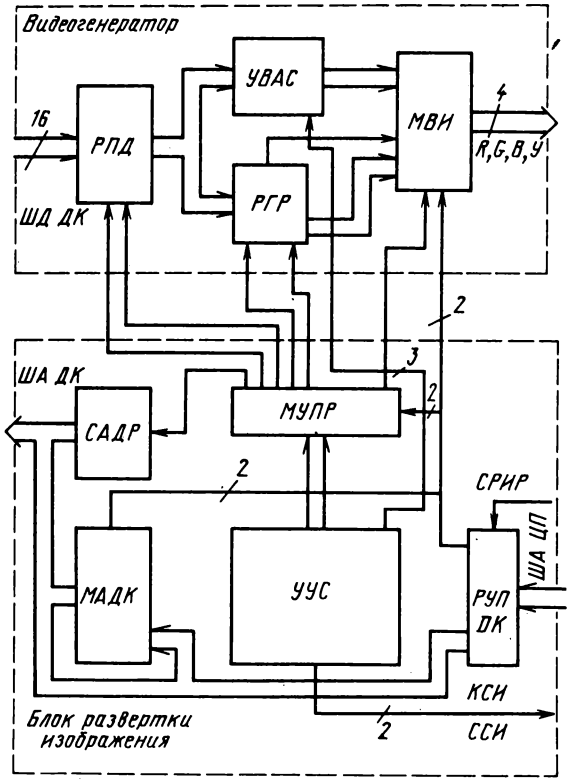


Рис.8.17. Функциональная схема дисплейного контроллера:
РПД - регистр приема данных; РГР - регистр графических режимов; МВИ - мультиплексор выводной информации; УВАС - узел вывода алфавитно-цифровых сиволов; САДР - счетчик адреса; МУПР - мультиплексор управления; МАДК - мультиплексор адреса ДК; УУС - устройство управления синхронизацией; СРИР - сигнал разрешения изменения регистров; РУПДК - регистр управления дисплейным котроллером; КСИ - кадровый синхроимпульс; ССИ - строчный синхроимпульс

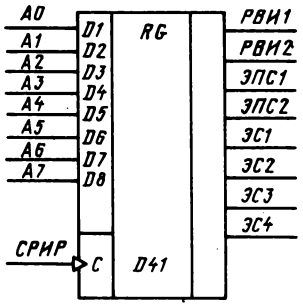


Рис. 8.18. Регистр управления дисплейного контроллера

мещенное на модуле, в зависимости от исполнения модуля подразделяются на виды:

4 ЭС и 16 ЭПС для исполнения с емкостью ОЗУ 32К байт;

8 ЭС и 32 ЭПС для исполнения с емкостью ОЗУ 64К байт;

16 ЭС и 64 ЭПС для исполнения с емкостью ОЗУ 128К байт.

Таким образом, из сказанного следует, что ДК обеспечивает вывод на экран (в том или ином виде) информации, хранящейся в любой области ОЗУ, размещенного на ячейке. Область памяти, высвечиваемая в данный момент времени на экране, определяется состоянием разрядов ЭС для режимов ГСР и ГВР и состоянием разрядов ЭС и ЭПС для режимов ГНР и АЦР. Расположение экранных страниц в пределах оперативной памяти показано на рис. 8.19.

Номер экранной подстраницы (экранной страницы) задает также режим вывода алфавитно-цифровой информации - АЦР32 или АЦР64: для младших экранных подстаниц (ЭПС0 - ЭПС31), находящихся в пределах ЭС0 - ЭС7, обеспечивается режим АЦР32, а для страших экранных подстаниц (ЭПС32 - ЭПС63), находящихся в пределах экранных страниц ЭС8 - ЭС15, режим АЦР64 (см. табл. 8.6). При выводе информации на экран в режиме АЦР64 все четные экранные подстраницы обеспечивают нормальное отображение символов на экране (белые символы на черном фоне экрана), а все нечетные - инверсное отображение символов (черные символы на белом фоне). Для исполнения модуля с емкостью ОЗУ 32 и 64К байт, в которых не существует ОЗУ для старших экранных подстраниц ЭПС32 - ЭПС63 (ЭС8 - ЭС15), в режиме АЦР64 вместо старших экранных подстраниц выводятся младшие. Таким образом, для этих исполнений модуля младшие экранные подстра

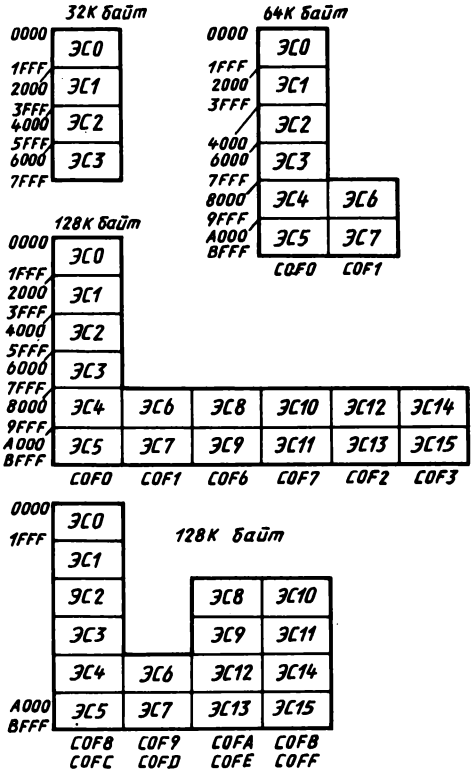


Рис. 8.19. Расположение экранных страниц

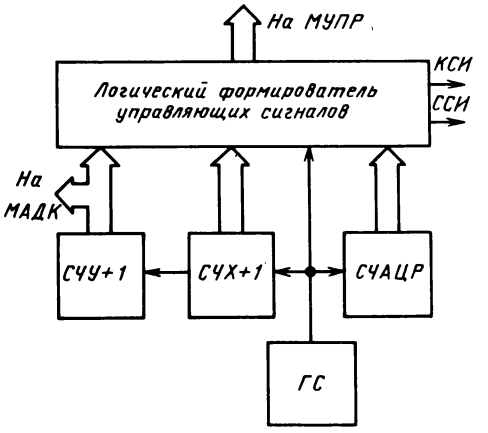


Рис. 8.20. Функциональная схема узла управления и синхронизации

Таблица 8.6

Включение экранных страниц

Режим вывода информации	Программные переключатели экрана	РВИ		ЭПС	ЭС
		РВИ 2	РВИ 1		
ГНР	C7X0 C7X4	0	0	0 1	X = 0+F
	C7X8			2	
	C7XC			3	
ГСР	C7X1	0	1	-	X = 0+F
	C7X5				
	C7X9				
	C7X0				
ДЦР-32	C702 - C772	1	0	0	
	C706 - C776			1	
	C70A - C77A			2	
	C70E - C77E			3	
АЦР-64	C782 - C7F2	1	0	0	Нормальные символы
	C78A - C7FA			1	
	C786 - C7F6	1	0	2	Инверсные символы
	C78E - C7FE			3	
ГВР	C7X3	1	1	-	X = 0+P
	C7X7				
	C7XB				
	C7XF				

ницы можны выводить как в режиме АЦР32, когда ЭС4 = 0, так и в режиме АЦР64, когда ЭС4 = 1.

Узел управления и синхронизации. Узел управления и синхронизации (УУС) является основным функциональным блоком (рис. 8.20), с помощью которого ДК осуществляет развертку изображения на экране телевизионного приемника. При этом УУС вырабатывает:

тактовые импульсы (ТИ) основной частоты и синхроимпульсы (СИ);

сигналы, необходимые для управления и синхронизации остальных функциональных блоков ДК;

кадровый и строчный синхроимпульс;

часть разрядов ША ДК.

Генератор синхроимпульсов (ГС) состоит изх задающего генератора, представляющего собой высокочастотный автогенератор с кварцевой стабилизацией (ВК-2), и делителя частоты на основе *D*-триггера (*D*43, см. рис. 8.13). Задающий генератор обеспечивает непрерывную генерацию ***ТИ*** с частотой следования 10,5 МГц, а триггер (делитель частоты на 2) вырабатывает синхроимпульсы СИ и $\overline{\text{СИ}}$ с частотой следования 5,25 МГц (выводы ***6*** и ***5*** триггера ***D43***). Один период СИ (190 нс) равен времени продвижения луча по строке на расстояние, равное размеру одного элемента изображения (точки).

Из последовательности СИ с помощью двух перечисленных схем СчХ (***D43*** - ***D49***) и СчY (***D51*** - ***D52***) вырабатываются импульсы строчной и кадровой разверткой. Следует отметить, что качество изображения в сильной степени зависит от вида винхронизации стройчной развертки. Используемая в телевидении черезстрочная развертки обычно приводит к заметному эффекту мерцания при воспроизведении изображений, составленных из линейных элементов. Поэтому для повышения качеств изображения в ДК черезстрочной была использована более прогрессивная развертка с удвоенной частотой строк (построчная). Это позволило практически полностью исключить утомительный для глаза эффект мерцания (фликкер - эффект), а яркость и контрастность изображения повысить.

Счетчик горизонтальной дискретизации (СчХ) (горизонтального перемещения луча). Представляет собой девятиразрядный счетчик с коэффициентом пересчета равным 336 (256 импульсов на прямой ход развертки строки и 80 на обратный ход импульсов гашения). СчХ яавляется основным устройством, с помощью которого осуществляется горизонтальная развертка изображения. Код, получаемый в каждый данный момент на выходах СчХ, соответствует положению луча на телевизионной строке. На выходе девятого разряда СчХ вырабатывается последовательностьимпульсов, аналогичных телевизионным строчным гасящим импульсам (СГИ), с периодом строчной развертки 64 мкс.

Счетчик вертикальной дискретизации (СчY, см. рис. 8.13) или вертикального перемещения луча. Это девятиразрядный счетчик, но с коэффициентом пересчета равным 312 (256 + 56). СчY является, в свою очередь, основным устройством, с помощью которого осуществляется вертикальная развертка изображения. Код на выходе СчY равен номеру телевизионной строки, воспроизводимой в данный момент на экране. На выходе девятого разряда СчY вырабатывает последовательность импульсов, аналогичных телевизионным кадровым гасящим импульсам (КГИ), с периодом кадровой развертки 20 мс.

Счетчик алфавитно-цифрового режима (**СчАЦР**, см. рис.8.13) представляет

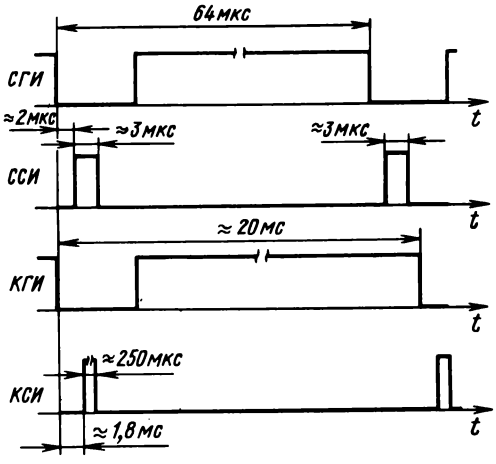


Рис. 8.21. Временные диаграммы сигналов с СИ и КСИ

обой счетчик-делитель с коэффициентом деления, равны 7, и обеспечивает горизонтальное форирование знакомет для алфавитно-цифровых символов.

Логический формирователь управляющих сигналов объединяет схемы комбинированной логики, реализованные на основе традиционных логических элементов (И-НЕ, ИЛИ-НЕ, И и др.), используя сигналы, вырабатываемые ГС, чХ, СчY, СчАЦР, формируют все основные сигналы, управляющие работой остальных устройств ДК. Кроме того, в логическом формирователе вырабатываются два сигнала высшей синхронизации КСИ и ССИ, предназначенных для синхронизации ДК с телевизионным приемником (рис. 8.21).

Мультиплексор управления. Мультиплексор управления (МУПР) (***D63, D65,*** см. рис. 8.13) служит для согласованного переключения сигналов, управляющих работой устройств ДК. Смена режима вывода информации осуществляется центральным процессором, который для этого должен изменить состояние соответствующих разрядов выбора информации (РВИ) в регистре управления ДК. Изменение разрядов РВИ вызывает переключение трех мультиплексоров, входящих в ДК: мультиплексоров управления, адреса и выходной информации (МУПР, МАДК и МВИ), переводя тем самым ДК в соответствующий режим вывода информации. При этом МУПР осуществляет переключение следующих сигналов, управляющих регистром приема данных (РПД), регистром графических режимов (РГР), счетчиком адреса (САДР) и МВИ:

SBMИ - сигнал, стробирующий МВИ, обеспечивает выключение МВИ на время обратного хода строчной и кадровой разверток для исключения воспроизведения случайной информации за пределами рабочей части экрана. Сигнал формируется на выходе *7* триггера ***D65*** (см. рис. 8.13) импульсами КГИ и СГИ, поступающими через ***D57*** на информационные входы ***D1 - D4*** элемента ***D65;***

C2РГР - сигнал, по которому происходит прием информации с ***РПД*** на ***РГР***. Вырабатывается на выходе *9* элемента ***D63***, далее сигнал усиливается и поступает на ***РГР*** (вход ***C2***);

V2РГР - сигнал, разрешающий прием информации на РГР, а также управляющий режимом работы РГР (сдвиг информации вправо либо прием и хранение параллельного кода). Этот сигнал формируется элементом ***D65*** (выход 9);

V2РПД - сигнал, разрешающий прием информации с ШД ДК на РПД и управлящий режимом работы ***РПД,*** формируется на выходе *7* триггера ***D64***.

+1САДР - сигнал, задающий скорость перебора адресов дисплейным контроллером при выводе изображения в пределах телевизионной строки (выход *9* элемента ***D64***).

Счетчик адреса. Счетчик адреса (САДР) (***D60***, см. рис. 8.13) обеспечивает формирование младших разрядов ША ДК (***A1 - A4***) при любом режиме работы ДК. С

Таблица 8.7

Назначение разрядов ША ДК

Разряды ША ДК	Режим вывода информации			
	ГНР РВИ1 РВИ2 0 0	ГСР РВИ1 РВИ2 1 0	АЦР РВИ1 РВИ2 0 1	ГВР РВИ1 РВИ2 1 1
A5	3р СчY	5р САДР	5р САДР	1р СчY
A6	4р СчY	2р СчY	4р СчY	2р СчY
A7	5р СчY	3р СчY	5р СчY	3р СчY
A8	6р СчY	4р СчY	6р СчY	4р СчY
A9	7р СчY	5р СчY	7р СчY	5р СчY
A10	8р СчY	6р СчY	8р СчY	6р СчY
A11	ЭПС 1	7р СчY	ЭПС 1	7р СчY
A12	ЭПС 2	8р СчY	ЭПС 2	8р СчY

помощью САДР осуществляется последовательный перебор адресов памяти экрана, содержащих информацию, необходимую для формирования изображения в пределах телевизионной строки. Инкрементирование (увеличение на единицу) САДР происходит по сигналу *+1САДР*. Частота следования импульсов сигнала *+1САДР* определяется режимом работы ДК. Обратный ход строчной развертки усматривает САДР в нулевое состояние.

Мультиплексор адреса ДК. Мультиплексор адреса ДК (МАДК) предназначен для коммутации разрядов адреса *A5 - A12* ША ДК в соответствии с режимом работы ДК. Разряды *A5 - A12* ША ДК определяют последовательность перебора адресов памяти экрана, содержащих информацию, необходимую для вертикальной развертки изображения на экране (по строкам). В зависимости от режима вывода информации МАДК подключает к *A5 - A12* соответствующие разряды СчY схемы управления и синхронизации (табл. 8.7). Заметим, что 5-й разряд формируется на МУПР коммутацией сигналов, поступающих с СчY и САДР.

Регистр приема данных. Регистр приема данных (РПД) (*D*69 - *D*72, см. рис. 8.13) предназначен для приема информации, поступающей с ШД ДК. Как известно, информация на ШД ДК поступает из ОЗУ через регистр БРД. Так как организация оперативной памяти предусматривает хранение информации, представленной восьмиразрядными словами центрального процессора, существует взаимосвязь между разрядами слова центрального процессора, разрядами БРД (ШД ДК) и разрядами РПД (табл. 8.8).

Информция о ШД ДК принимается в РПД по срезу сигнала СИ при наличии вы-

Таблица 8.8

Соответствие между разрядами ШД центрального процессора и БРД

№ байта	1-й байт (четный, A0=0	2-й байт (нечетный, A0=1
Разряды слова центрального процессора	D7 D6 D5 D4 D3 D2 D1 D0	D7 D6 D5 D4 D3 D2 D1 D0
Разряды БРД (ШД ДК)	1 2 3 4 5 6 7 8	9 10 11 12 13 14 15 16
Разряды РПД	1 5 9 13 2 6 10 14	3 7 11 15 4 8 12 16

сокого уровня сигнала (V2РПД, см. рис. 8.13), разрешающего прием данных. На вход ***C1*** через ***D68.3*** поступает инвертированный синал управления с мультиплексора ***D63***. В режиме приема данных вход ***C1*** = 0.

Регистр графических режимов. Регистр графических режимов (РГР) (***D76***,. см. рис. 8.13) используется для формирования изображения только в графических режимах. При этом во всех режимах работы ДК регистр ***РГР*** всегда принимает информацию с разрядов 1, 5, 9 из 13 РПД по срезу сигнала ***C2РГР***. При работе ДК в режимах среднего и низкого разрешения РГР работает как четырехразрядный регистр и передает принятую с РПД информацию параллельным кодом на соответствующие входы МВИ.

В графическом режиме с высоким разрешением ***РГР*** работает как регистр сдвига с параллельным приемом информации. При наличии высокого уровня сигнала ***V2РГР*** регистр принимает информацию параллельным кодом. При смене высокого уровня сигнала ***V2РГР*** на низкий уровень РГР переводится в режим двига принятого кода вправо. Сдвиг оуществляется по срезу сигнала ***СИ***. Таким образом, в режиме с высоким разрешением РГР обеспечивает преобразование параллельного кода в последовательный при передаче информации с ***РПД*** на ***МВИ*** с каждым импульсом СИ. В этом случае код выхода 10 РГР поступает на входы ***D4*** и ***E4*** МВИ.

Мультиплекор выходной информации. МВИ (D81, D82, см. рис. 8.13) подключает выходные контакты ДК (***R***, ***G***, ***B***, ***Y***) к различным иточникам выходной информации в зависимости от режима работы ДК (табл. 8.9).

На выходе МВИ формируется четыре сигнала (R, GF, B, Y), которые вместе с

Таблица 8.9

Формирование изображения

Режим вывода информации	ГНР	ГСР	АЦР	ГВР
Состояние сигнала РВИ1	0	1	0	1
Состояние сигнала РВИ2	0	0	1	1
Источник выходной информации	РГР (парал-лельный	РГР (параллель-ный	УВАС	РГР (последо-тельный

КСИ и ССИ подаются либо на контроллер СЕКАМ (разъем X1), либо на телевизор, имеющий соответствующие входы, либо на ВКУ. С помощью игнала **SMBИ** мультиплексор включается (на всех выходах низкий уровень) на время обратного хода строчной и кадровой разверток. В алфавитно-цифровом режиме время выключения МВИ по строкам несколько больше, чем в графичеких режимах, так как число отображаемых на строке точек и режиме АЦР равно 224 (32 знакомест по 7 точек в троке), т.е. изображениеалфавитно-цифровых символов по строкам заниает не 256 точек, как в графических режимах, а на 32 точки меньше, чем обратный ход строчной развертки.

Узел вывода алфавитно-цифровых символов. Узел вывода алфавитно-цифровых символов (УВАС) обеспечивает отображение на экране фиксированного набора символов - букв, цифр, специальных знаков. УВАС используется дисплейным контроллером только в алфавитно-цифровом режиме. Важнейшей частью этого узла является ПЗУ знакогенератора ***D73*** (см. рис. 8.13), содержащее точечную матрицу (полиграмму) размерностью 7 x 8 каждого отображения символа. При растровом методе формирования изображения символов на экране происходит построчная развертка полиграммы символа, хранимой в ПЗУ знакогенератора. С этой целью на адресные входы ПЗУ знакогенератора, кроме кода символа, поступающего с РПД, подаются разряды 1 - 3 СчY (рис. 8.22).

Выбранная из ***ПЗУ*** знакогенератора строка символа поступает параллельным кодом на регистр сдвига ***РС*** (***D80***, см. рис. 8.13), который сдвигает ее последовательным кодом частотой СИ при 32 символах в строке и частотой ТИ при 64 символах в строке. Последовательный код строки символа поступает на логическую схему инверсии подсвета и мерцания ***СИПМ***, где в зависимости от кодов ***ИП*** и ***МЕ*** (переключение осуществляет коммутатор ***D74***, происходит либо инвертирование, либо повторение кода. С выхода СИПМ последовательный код строки поступает на логическую схему ***D78***, где происходит поразрядное логическое умножение с четырехразрядным кодом цвета данного символа, поступающим с регистра цвета символа ***РЦС*** (***D77***). С выходов с элемента D78 четырехразрядный код последовательности цветовой засветки точек строки символа поступает на соответствующий вход МВИ.

Код цвета данного символа заносится с РПД на РЦС одновременно с приемом параллельного кода строки отображаемого символа на РС. Данные с РПД разделяются: часть разрядов 2 байта как сигнал режима отображения цвета поступает на РЦС, два разряда ИП и МЕ поступают на D74, управляя его переключе нием. В течение отображения семи точек строки данного символа информация на РЦС остается неизменной. В режиме АЦР64 код, задающий мерца-

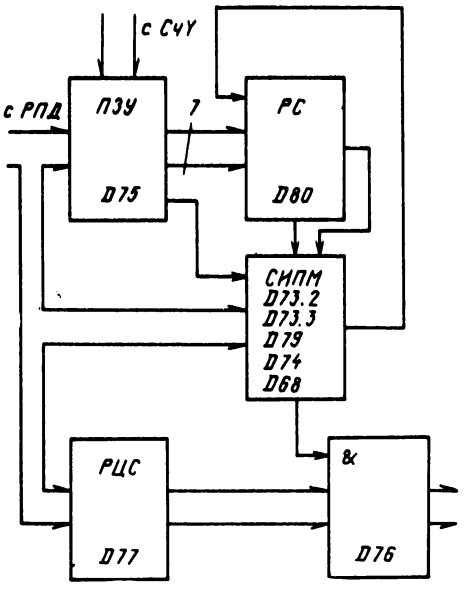


Рис. 8.22. Узел вывода алфавитно-цифровой информации

ние символа, хранится в восьмом разряде ПЗУ знакогенератора, вместе с полиграммами символов. Это дает возможность задавать режим мерцания для отдельных частей изображения, вплоть до отдельных строк символов.

Формирование изображения дисплейным контроллером. Чтобы обеспечить формирование изображения на экране, ДК независимо от режима работы должен осуществить последовательный перебор адресов видеоОЗУ в соответствии с перемещением луча по экрану телевизора. В режимах ГНР и АЦР контроллер при формировании одного кадра изображения последовательно перебирает адреса 000 - 7FF, что соответствует емкости видеоОЗУ 2К байт. В режимах ГСР и ГВР емкость видеоОЗУ возрастает до 8К байт, что требует увеличения адресов последовательного перебора ДК до 0000 - 1FFF. В пределах телевизионной строки перебор адресов осуществляется с помощью САДР. При перемещении луча по строке через определенные интервалы времени САДР по сигналу +1САДР каждый раз увеличивает свое состояние на единицу. Временной интервал между соседними импульсами сигнала +1САДР определяется режимом вывода информации на экран и размером элемента разложения (знакоместа) по строке.

Перебор остальных адресов видеоОЗУ осуществляется с помощью СчY, который увеличивает свое состояние на единицу с каждым перемещением луча на новую телевизионную строку. МАДК в соответствии с режимом вывода информации определяет подключение разрядов СчY в качестве разрядов ША ДК, тем самым обеспечивая размеры элемента разложения (знакоместа) по вертикали.

Из адресуемой контроллером ячейки видеоОЗУ информация во время фазы ДК поступает на БРД, где хранится до тех пор, пока ДК не сменит адрес. После этого во время очередной фазы ДК (низкий уровень сигнала $\varphi 0$) на БРД поступит новая информация.

Из БРД информация по ШД ДК поступает на регистр приема данных ДК. Регистр независимо от режима работы ДК после приема информации переключается на режим сдвига. При этом выходной информацией РПД являются четырехразрядные слова, поступающие на РГР, который в режимах ГНР и ГСР сохраняет принятую информацию до следующего приема, а в режиме ГВР осуществляет сдвиг принятой информации. В режимах ГНР и ГСР информацией, хранимой на РГР, является код цвета блока, выводимого в данный момент на экран. Размер блока по строке определяет время сохранения кода цвета данного блока на РГР. В течение этого времени на РПД происходит сдвиг информации, и к моменту очередного приема информации на РГР на выходах ***4, 3, 2*** и ***1*** разрядов РПД будет присутствовать код цвета следующего по строке блока. В режиме ГВР информация, выдвигаемая с последовательного выхода РГР, определяет засветку четырех последовательно расположенных по строке точек формируемого изображения.

СТРУКТУРА ВВОДА-ВЫВОДА

Функции ввода-вывода в ПЭВМ "Агат" могут быть разделены на две категории: функции, которые выполняются ячейками памяти и интерфейса с помощью специальных аппаратно-программных средств, образующих построенный интерфейс ввода-вывода (ВИ В/В), и функции, выполняемые интерфейсными платами, устанавливаемыми в разъемы ячейки памяти и интерфейса (***X1, X3 - X7***) и образующими внешний интерфейс ввода-вывода. Обе эти категории функций, используя

внутренний интерфейс ПЭВМ, обеспечивают связь с центральным процессором и могут быть программно доступны через 4096 ячеек в адресном пространстве процессора (адреса C000 - CFFF).

Внешний интерфейс ввода-вывода. На ячейке памяти и интерфейса расположен ряд из семи 60-контактных разъемов для подключения контроллеров внешних устройств. В шесть из семи этих разъемов могут быть установлены любые из интерфейсных плат внешнего оборудования (контроллеры принтера, накопителя на магнитных дисках и т.п.). Каждый разъем на ячейке имеет свой собственный номер: самый близкий к краю платы разъем имеет обозначение ***X7***, а самый дальний - ***X1***. Разъем ***X2*** предназначен для установки платы центрального процессора и не может быть использован для других целей. Разъем ***X1***, кроме стандартных шин внутреннего интерфейса ПЭВМ, имеет несколько контактов, предназначенных для подключения платы контроллера СЕКАМ.

Во всех разъемах, кроме разъема ***X2***, имеются два контакта для передачи специальных сигналов управления. Такими сигналами являются:

$\overline{DS}$ (DEVICE SELECT- контакт А11);

$\overline{I/OS}$ ($\overline{I/O\ SELECT}$ - контакты Б1, Б2).

Эти сигналы являются активными (устанавливаются в логический нуль) в разные моменты времени, и никогда не может быть такой ситуации, когда на уровень логического нуля переходит более, чем один сигнал. Наличие того или иного сигнала $\overline{DS}$ или $\overline{I/OS}$ соответствует обращению центрального процессора в область адресов внешних устройств, зарезервированную за данным разъемом (см. табл. 8.3).

Для формирования этих сигналов используется схема, показанная на рис. 8.23. Сигналы $\overline{DS}$ и $\overline{I/OS}$ позволяют выбрать для работы тот или иной модуль внешнего интерфейса. Сигнал $\overline{I/OS}$ формируется дешифратором ***D1*** (см. также рис. 8.13). Этот же дешифратор вырабатывает сигналы управления дешифраторами ***D3*** и ***D7***. С разъема ***X2*** на вход ***S*** подается вспомогательный сигнал ***I/O STR1*** (см. рис. 8.10), вырабатываемый дешифратором ***D4*** платы центрального процессора, при обращении микропроцессора по адресу C000 - CFFF. На входы ***A, B, C*** дешифратора ***D1*** поступают разряды ***A8, A9, A10***, определяющие номер разъема. На выходах ***10, 11, 12, 7, 6, 5*** дешифратора ***D1*** формируются сигналы $\overline{I/OS}$, поступающие на соответствующие разъемы согласно обращению центрального процессора по одному из 256 адресов в диапазоне CN00 - CNFF, где N - номер разъема.

На выходе ***9*** дешифратора ***D1*** формируется сигнал, управляющий работой дешифратора ***D3***, вырабатывающего импульсы $\overline{DS}$. Разряд ***7*** (***A7***) ША ЦП определяет выработку сигнала $\overline{DS}$ или сигналов выбора встроенного интерфейса ввода-вывода.

Если ***A7*** = 1, то сигнал $\overline{I/O\ STR1}$, инвертированный ***D2.2***, поступает на вход ***S*** элемента ***D3***. На информационные входы этого дешифратора поступают разряды ***A4, A5, A6***, определяющие номер разъема, на который приходит сигнал $\overline{DS}$.

Если ***A7*** = 0, сигнал $\overline{I/O\ STR1}$ поступает на ***D7*** (через ***D2.3***), где происходит формирование по разрядам ***A4, A5, A6*** сигналов выборки встроенного интерфейса ввода-вывода (В и В/В).

Встроенный интерфейс ввода-вывода (В и В/В). Встроенный интерфейс ввода-вывода (рис. 8.24) обеспечивает подключение к внутреннему интерфейсу ПЭВМ "Агат" следующих устройств ввода-вывода: клавиатуры, игровых пультов, магни-

тофона и громкоговорителя. Управление функциями ввода-вывода В и В/В осуществляется с помощью 128 ячеек адресного пространства центрального процессора, которое начинается с ячейки С000 и простирается до ячейки С07F. Информация поступает на ШД ЦП через мультиплексор входных данных ***МВД***.

Интерфейс звуковой индикации ПЭВМ "Агат" реализуется с помощью громкоговорителя, расположенного внутри корпуса ПЭВМ и управляемого схемой звуковой индикации (рис. 8.25). Основой схемы является D-триггер, включенный в счетном режиме. Выходной сигнал с триггера усиливается на транзисторе ***VT2***, в коллекторную цепь которого включается громкоговоритель. Переключение триггера осуществляется с помощью программного переключателя С030 - С03F. Каждый раз, когда происходит обращение центрального процессора к одному из 16 указанных адресов, состояние триггера изменяется на противоположное. Обращаясь к адресам С030 - С03F с определенной частотой, процессор может обеспечить переключение триггера с любой частотой звукового диапазона.

Следует учесть, что данные, которые при этом могут считываться или записываться, могут быть любыми, поскольку именно адрес является переключателем.

Интерфейс кассетного магнитофона (рис. 8.26) служит для преобразования информации при записи и считывании с кассетного магнитофона программным способом. Способ записи информации на магнитофон частотно-модулированный.

Схема для записи информации выполнена аналогично схеме звуковой индикации: на D-триггере, включенном в счетном режиме (рис. 8.26, *а*). Выходной

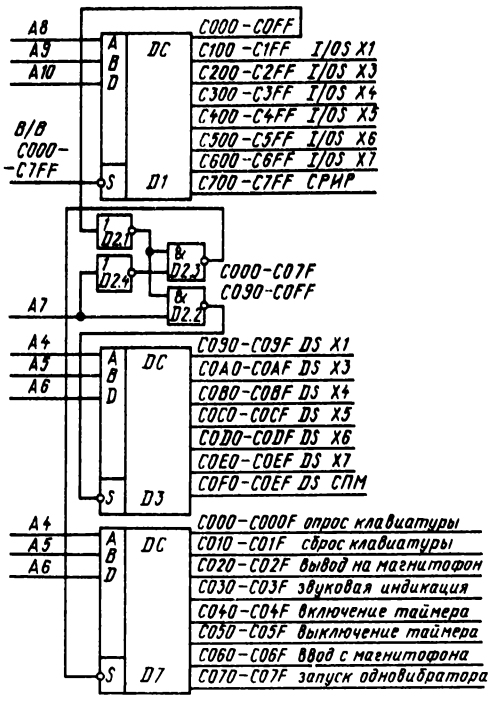


Рис. 8.23. Дешифрация сигналов ввода-вывода

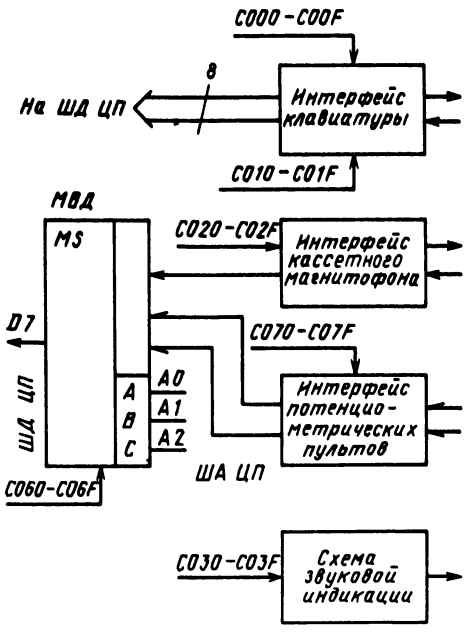


Рис. 8.24. Встроенный интерфейс ввода-вывода

сигнал с триггера через резистивный делитель подается на выходной контакт разъема магнитофона *X11* (см. рис. 8.13).

Резистивный делитель ограничивает амплитуду выходного сигнала на уровне 200 мВ. Для переключения триггера используется программный переключатель C020 - C02F. Часто повторяемое обращение центрального процессора к любому из этих 16 адресов обеспечивает переключение триггера с определенной частотой. Изменяя частоту и продолжительность переключения триггера, можно тем самым закодировать информацию, записываемую на магнитную ленту. Программа для кодирования данных при записи на магнитную ленту включена в системный монитор. Логический нуль кодируется частой в 1 кГц, логическая единица - частотой 2 кГц, т.е. использован принцип частотной модуляции. Такое кодирование позволяет компенсировать изменение скорости движения ленты.

Для ввода программ с магнитофона в ПЭВМ применяется схема для приема информации с магнитофона (рис. 8.26, *б*). Поступающий с линейного выхода магнитофона сигнал подается на вход операционного усилителя *D87*, включенного по схеме усилителя-ограничителя с коэффициентом усиления 100.

С выхода операционного усилителя усиленный сигнал поступает на МВД (*D93*), который обеспечивает ввод данных от магнитофона на разряд *D7* шины данных центрального процессора при его обращении по адресу C060. Таким образом, состояние схемы магнитофонного ввода может быть определено центральным процессором по значению разряда *D7* путем просмотра ячейки C060 адресного пространства. Дальнейшую обработку поступающей с магнитофона информации процессор осуществляет под управлением специальной программы, которая также входит в системный монитор.

Интерфейс потенциометрических пультов (ИПП) обеспечивает подключение к ПЭВМ двух пультов, в состав которых входят потенциометр и кнопка. Сигналы от кнопок поступают непосредственно на МВД, который обеспечивает их передачу на *D7* ШД ЦП при обращении центрального процессора по адресам C061, C062.

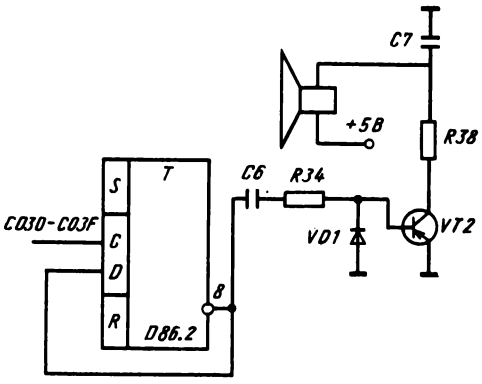


Рис. 8.25. Интерфейс звуковой индикации

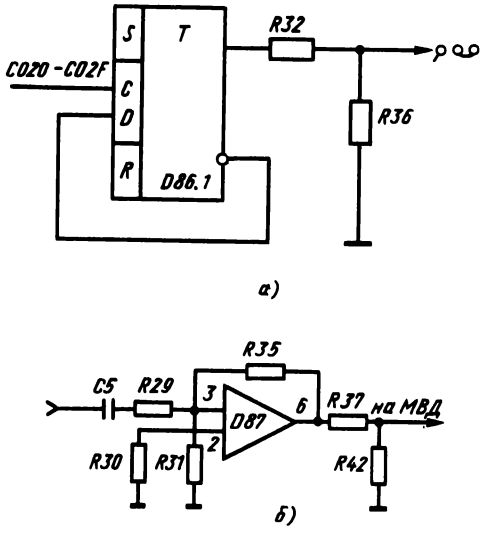


Рис. 8.26. Интерфейс кассетного магнитофона:
а - запись; б - чтение

Состояние кнопок может быть считано процессором по специальной программе таким же образом, как это делается для магнитофонного ввода: путем просматривания разряда D7 ячеек C061, C062 адресного пространства центрального процессора.

Потенциометры пультов подключаются к одновибраторам, собранным по схеме, показанной на рис. 8.27. Одновибраторы преобразуют угол поворота потенциометров во временной интервал. Чтобы центральный процессор смог считать установку потенциометров, он должен запустить одновибраторы с помощью программного переключателя C070. Обращение центрального процессора к ячейкам C070 - C07F вызывает пуск одновибраторов. Временной интервал между вырабатываемыми импульсами определяется сопротивлением, соответствующим углу поворота потенциометров.

Центральный процессор под управлением программы может определить относительное значение временного интервала между запуском и окончанием импульсов одновибраторов, считывая состояние разряда D7 ШД ЦП при обращениях по адресам C064, C065.

МВД (*D93*) постоянно находится в рабочем состоянии (на входе *7* - логический нуль). По заданной адресной комбинации *A0* - *A2* он пропускает на прямой выход один входной сигнал, соответствующий двоичному коду адреса. Выход МВД (*D93*) связан с шиной *D7* ШД ЦП через буферный элемент *D94.1*. Ножка *10* элемента *D94.1* является управляющей: процессор, опрашивая разряд *D 7* шины данных, обращается к диапазону адресов C060 - C06F.

Интерфейс клавиатуры обеспечивает преобразование последовательного кода, поступающего от блока клавиатуры, в параллельный код и передает его на ШД ЦП (рис. 8.28).

Признаком готовности информации к передаче из блока клавиатуры является

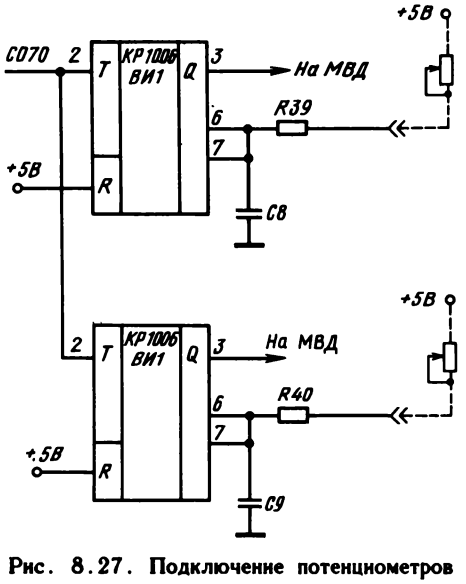


Рис. 8.27. Подключение потенциометров пульта

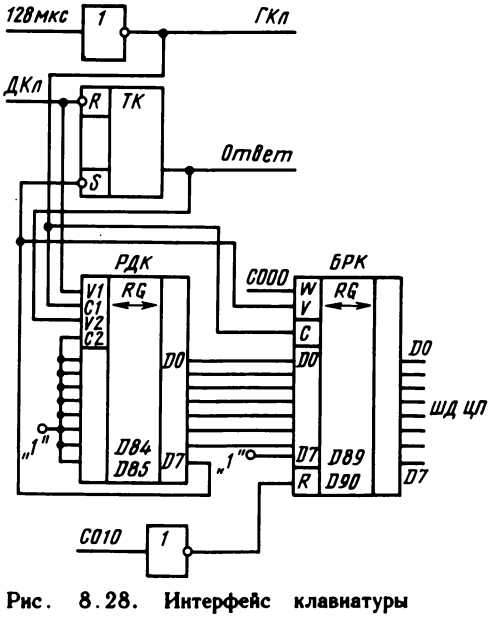
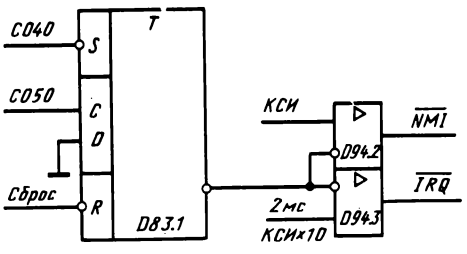


Рис. 8.28. Интерфейс клавиатуры

Рис. 8.29. Узел формирования сигналов прерывания процессора



появление уровня логического нуля на информационной линии данных с клавиатуры (***ДКл***), подключенной к входу сброса триггера ***ТК*** клавиатуры и к последовательному входу регистра ***РДК*** данных клавиатуры.

Логический нуль на входе сброса ***ТК*** устанавливает на прямом выходе триггера логический нуль, который по линии "***Ответ***" подтверждает готовность интерфейса клавиатуры к приему информации и переводит ***РДК*** в режим последовательного приема информации. По каждому срезу импульсов на линии ***ДКл*** происходит прием очередного бита информации с линии ***ДКл*** на последовательный вход ***РДК*** до тех пор, пока начальный нулевой бит не окажется принятым в старший разряд ***РДК***. Появление низкого уровня на выходе старшего разряда ***РДК*** вызывает переброс ***ТК*** в противоположное состояние (на прямом выходе устанавливается высокий уровень), прекращая тем самым прием информации с линии ***ДКл***. Регистр ***РДК*** переводится в режим параллельного приема информации, и по очередному срезу импульсов на линии ***ГКл*** осуществляется сброс ***РДК*** (во всех разрядах появляется логическая единица). Принятая перед этим на ***РДК*** информация передается на буферный регистр клавиатуры (***БРК***) при наличии низкого уровня на старшем разряде ***РДК***. Информация на ***БРК*** является программно доступной, т.е. при обращении центрального процессора к адресам C000 - C00F БРК с сигналами дешифратора ***D7*** переводится в режим выдачи. Входы ***БРК*** при этом закрыты. На ШД ЦП поступает код нажатой клавиши.

Признаком нажатой кнопки на клавиатуре для центрального процессора является наличие логической единицы в старшем разряде считываемого с БРК байта при обращении к адресам C000 - C00F. После того, как центральный процессор примет информацию, поступившую от клавиатуры, он может освободить БРК от хранимой на нем информации. Для этой цели программа обработки данных, принимаемых с клавиатуры, должна предусматривать обращение процессора к любой ячейке C010 - C01F, тем самым обеспечивая формирование сигнала сброса содержимого ***РДК***.

Узел формирования сигналов прерывания процессора позволяет выработать два сигнала прерывания процессора ($\overline{NMI}$ и $\overline{TRQ}$), каждый из которых имеет определенную фиксированную частоту следования импульсов запроса прерываний (рис. 8.29). Частота следования импульсов сигнала $\overline{NMI}$ равна частоте телевизионной кадровой развертки (50 Гц), а частота следования импульсов сигнала $\overline{TRQ}$ в 10 раз выше (500 Гц). Формирование сигналов прерывания центрального процессора осуществляется с помощью программного переключателя C040 - C04F. Обращение процессора по этим адресам переводит триггер в состояние логической единицы (на инверсном выходе логический нуль), тем самым разрешая прохождение импульсов запроса прерываний на линиях $\overline{NMI}$ и $\overline{TRQ}$.

Прекратить формирование сигналов прерываний центрального процессора можно двумя способами:

1) с помощью программного переключателя С050. Обращение процессора по адресам С050 - С05F переводит триггер в нулевое состояние (на инверсном выходе - единица), тем самым запрещая прохождение импульсов запроса прерывания на линиях $\overline{NMI}$ и $\overline{IRQ}$;

2) с помощью общесистемного сигнала "Сброс", который подключен к входу установки триггера в нуль.

После прекращения формирования сигналов прерываний на линиях $\overline{NMI}$ и $\overline{IRQ}$ поддерживается высокий уровень, что позволяет центральному процессору работать без прерываний.

8.4. МОДУЛЬ ОПЕРАТИВНОЙ ПАМЯТИ

Модуль оперативной памяти в зависимости от исполнения может быть использован либо для расширения основной оперативной памяти, расположенной на плате памяти и интерфейса, либо в качестве псевдоПЗУ ПЭВМ.

Основные функциональные блоки модуля оперативной памяти показаны на рис. 8.30, принципиальная схема - на рис. 8.31.

Обмен информацией между центральным процессором и функциональными блоками модуля обеспечивается шиной данных (ШД ЦП) и шиной адреса ША ЦП.

Работа модуля оперативной памяти организуется на основе способа разделения фазы. При положительном уровне импульсов (рабочая фаза) происходит обмен информацией между центральным процессором и оперативной памятью

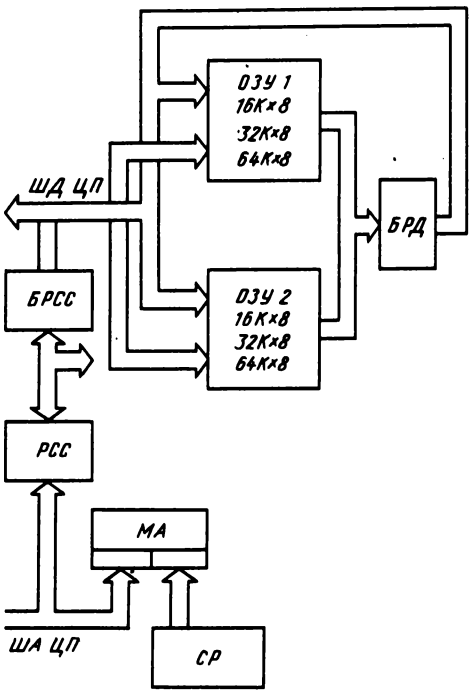


Рис. 8.30. Функциональная схема модуля оперативной памяти:
БРСС - буфер регистра слова состояния: РСС - регистр слова состояния; МА мультиплексор адреса; СР - счетчик регенерации; БРД - буферный регистр данных

модуля; при отрицательном (фаза регенерации) на модуле осуществляется регенерация ОЗУ.

Регистр слова состояния (рис. 8.32). Регистр слова состояния формирует сигналы, управляющие работой модуля. Ниже приведено назначение этих сигналов:

A16, A17, A18 - дополнительные разряды адреса;

ВЯ - сигнал выбора модуля при использовании его для расширения памяти;

Бл $\overline{W}$ - сигнал блокировки записи в модуль при использовании платы в качестве расширения памяти;

БлПЗУ - сигнал, обеспечивающий блокировку стандартного ПЗУ ПЭВМ и одновременно управляющий записью и чтением модуля оперативной памяти при использовании его в качестве псевдоПЗУ;

A19 ПЗУ - дополнительный разряд адреса, управляющий подключением дополнительного массива памяти емкостью 4К байт при использовании модуля в качестве псевдоПЗУ.

Дополнительные разряды адреса *A16, A17, A18*. Независимо от исполнения модуля оперативной памяти они всегда указывают номер массива памяти, подключаемого к соответствующим адресам доступа в
адресном поле процессора. В исполнениях модуля для расширения ОЗУ этими адресами являются 8000 - BFFF, которые обеспечивают адресацию массива емкостью 16К байт. При использовании модуля с объемом ОЗУ в 128К байт дополнительные разряды адреса ***A16, A17, A18*** указывают номер одного из восьми возможных массивов, подключаемых к адресам доступа центрального процессора. При использовании модуля с объемом ОЗУ в 64К байт состояние A18 безразлично, а ***A16, A17*** указывают соответственно номер одного из четырех возможных для доступа массивов. Для ячейки с объемом ОЗУ в 32К байт соответственно используется лишь ***A16***, а состояние ***A17*** и ***A18*** безразлично. Для модуля оперативной памяти, используемого в качестве псевдоПЗУ, назначение разрядов ***A16, A17, A18*** остается прежним, но при этом они указывают номер массива, подключаемого к адресам D000 - FFFF.

Дополнительный разряд *A19*. Как можно заметить, адреса D000 - FFFF обеспечивают адресацию массива памяти объемом 12К байт. Дополнительные разряды ***A16, A17, A18*** адреса указывают номер массива объемом 16К байт, поэтому необходимо ввести еще один дополнительный разряд ***A19***, который в модуле псевдоПЗУ управляет подключением оставшегося дополнительного массива памяти емкостью 4К байт на адреса D000 - DFFF.

Сигнал выбора модуля (ВЯ). Служит для перевода модуля расширения основной оперативной памяти (ООП) из режима хранения (когда доступ к ОЗУ модуля со стороны центрального процессора невозможна) в рабочий режим. Для этого, используя соответствующий программный переключатель, необходимо перевести ВЯ в состояние логической единицы, при котором разрешается доступ в модуль через адреса 8000 - BFFF. При этом отключается часть центрального ОЗУ, адресуемая через те же адреса. Перевод в состояние логического нуля запрещает доступ в модуль и одновременно включает на адреса 8000 - BFFF соответствующую часть ООП.

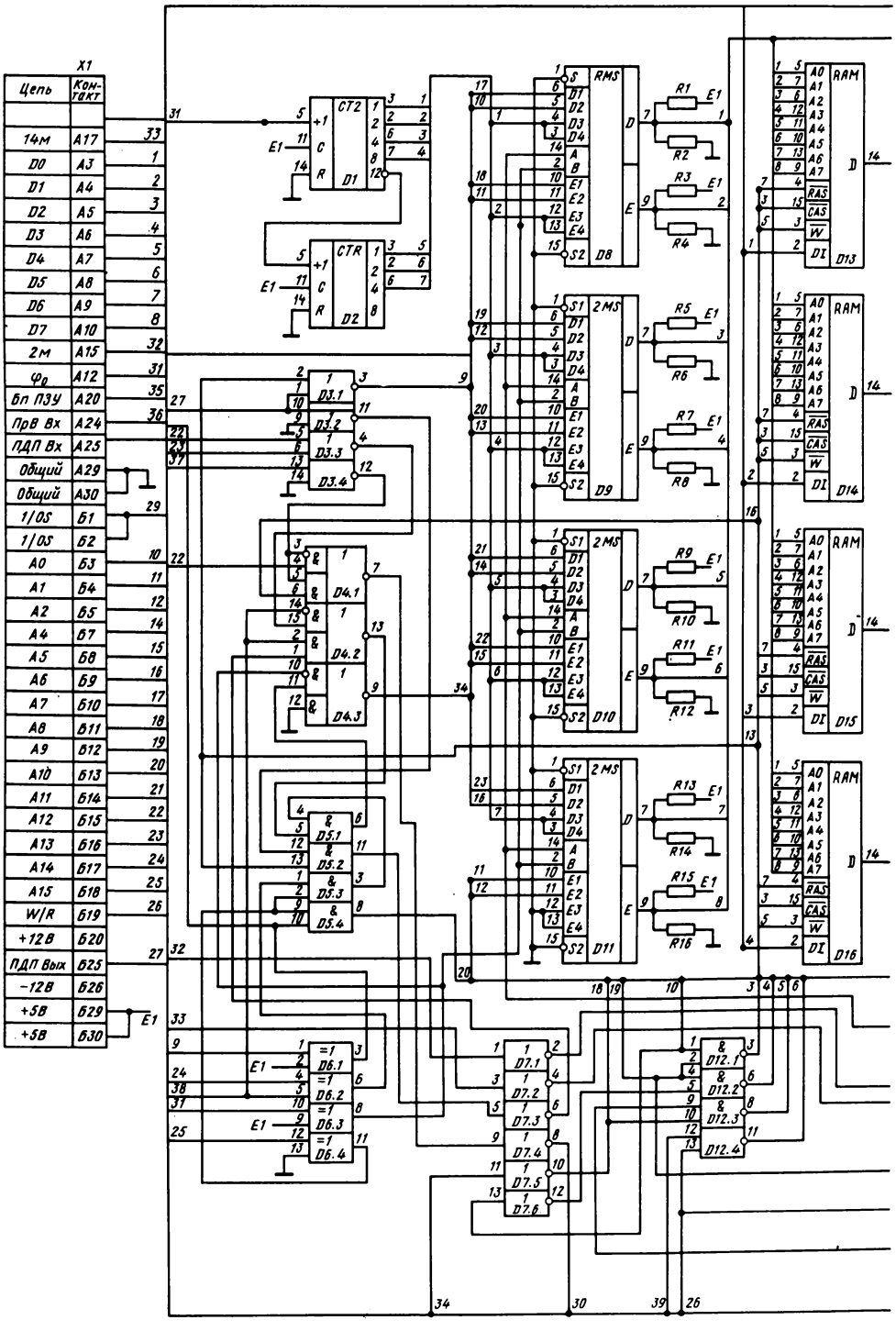
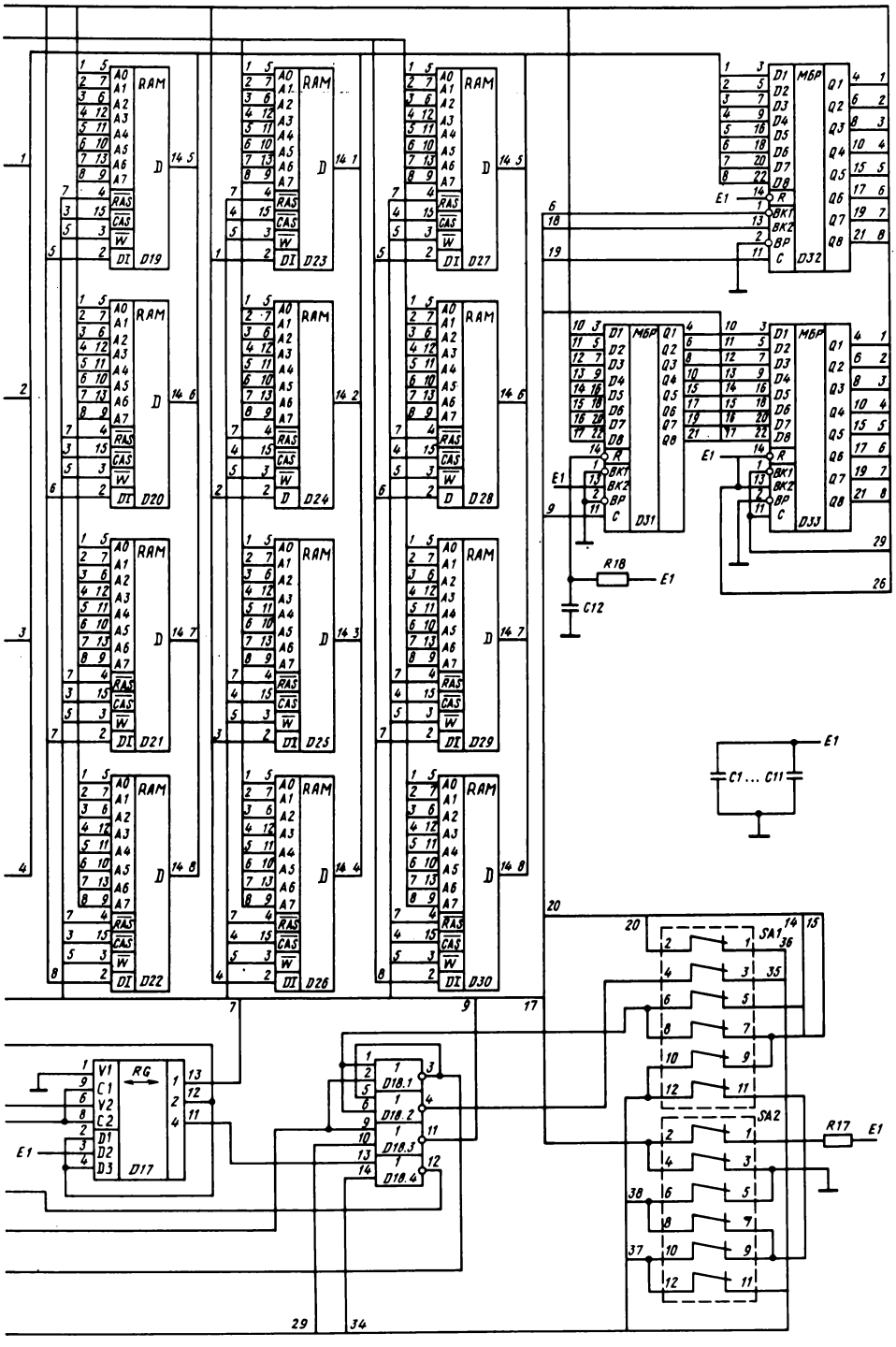


Рис. 8.31. Принципиальная схема модуля оперативной памяти


RAM
D19
D20
D21
D22
D23
D24
D25
D26
D27
D28
D29
D30
МБР
D31
D32
D33
R18
C12
C1 ... C11
E1
RG
D17
D18.1
D18.2
D18.3
D18.4
SA1
SA2
R17

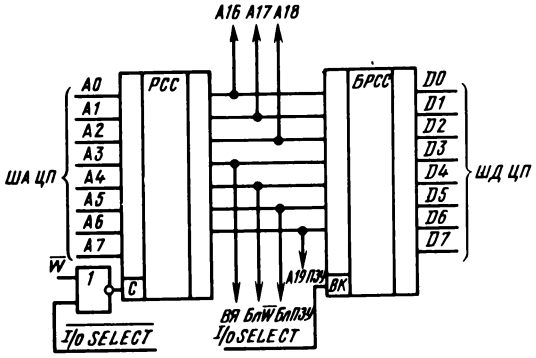


Рис. 8.32. Регистр слова состояния

Обязательным условием нормальной работы сигнала ВЯ является правильная установка модуля расширения ООП, что особенно важно для тех исполнений модуля памяти и интерфейса, в которых объем памяти превышает 32К байт (в случае применения нескольких модулей ООП). Один модуль расширения ООП может быть установлен только в разъем *X7* платы памяти и интерфейса, второй - только в соседний разъем *X6*, третий - в разъем *X5* и т.д. Максимальное число установленных таким образом плат расширения ООП равно пяти, и соответственно максимальный объем дополнительной оперативной памяти может достигать 640К байт (в случае использования памяти исполнения К565 РУ5). Приоритетность сигналов следующая: у модуля, установленного в разъем *X7*, сигналы имеют наиболее низкий приоритет; чем дальше от крайнего разъема *X7* располагается очередной модуль, тем выше приоритет его сигнала; модуль, установленный в разъем *X3*, имеет сигналы с наивысшим приоритетом. Это означает следующее:

установка сигналов ВЯ в логическую единицу на модуле с более высоким приоритетом обеспечивает подключение на адреса 8000 - BFFF соответствующего массива памяти именно с этого модуля и независимо от сигнала ВЯ модуля с более низким приоритетом он переводится в режим хранения;

чтобы обратиться к ОЗУ, расположенному на модуле с более низким приоритетом, необходимо предварительно установить в нуль все сигналы ВЯ модулей, имеющих более высокий приоритет по отношению к данному модулю.

Для исполнения платы памяти и интерфейса с минимальным объемом ООП в 32К байт модуль расширения ООП может быть установлен в любой из свободных разъемов. Все сказанное относительно приоритета модуля остается в силе в том случае, если более чем один модуль расширения ООП установлены в соседние разъемы.

Сигнал ВЯ не влияет на работу платы оперативной памяти в режиме псевдоПЗУ.

Блокировка записи. При необходимости защиты информации, записанной в ОЗУ модуля оперативной памяти, от случайных программных изменений на плате расширения памяти может быть введена защита записи. Это достигается переключением сигнала "Блокировка записи" (Бл $\overline{W}$) в состояние логической единицы, что запрещает запись в ОЗУ модуля, сохраняя доступность информации по чтению. Изменение информации в модуле возможно только после перевода Бл $\overline{W}$ в состояние логического нуля. На работу платы оперативной памяти в режиме псевдоПЗУ сигнал Бл $\overline{W}$ не влияет.

Блокировка ПЗУ. В отличие от всех других разрядов слова состояния модуля, блокировка ПЗУ (БлПЗУ) оказывает влияние не только внутри модуля оперативной памяти, но и на работу ПЭВМ в целом. Можно выделить два основных состояния модуля псевдоПЗУ и компьютера в зависимости от двух возможных состояний разряда БлПЗУ:

состояние логического нуля БлПЗУ одновременно разрешает запись в модуль псевдоПЗУ по адресам D000 - FFFF; запрещает считывание (выход в шину данных) из модуля псевдоПЗУ; разрешает нормальную работу стандартного ПЗУ;

состояние логической единицы БлПЗУ одновременно: запрещает запись в модуль псевдоПЗУ; разрешает считывание из модуля псевдоПЗУ по адресам D000 – FFFF; блокирует (запрещает считывание) стандартное ПЗУ.

Как видно, существует как бы раздвоение адресов D000 - FFFF при логи ческом нуле БлПЗУ: считывание из этих адресов происходит из стандартного ПЗУ, а запись - в ОЗУ модуля оперативной памяти. Именно это обстоятельство и позволяет достаточно просто осуществить загрузку ОЗУ модуля оперативной памяти для дальнейшей его работы в качестве псевдоПЗУ. На работу модуля для расширения памяти ООП разряд БлПЗУ не влияет.

Запись нового слова состояния модуля. Изменение информации, записанной в регистр слова состояния (РСС), как и изменение информации модуля оперативной памяти, не может быть случайным, так как может привести к аварийному завершению работы компьютера. Поэтому в отличие от всех остальных используемых в ПЭВМ программных переключателей программные переключатели модуля оперативной памяти действуют только при наличии записи по ним. При этом сохраняется правило нереливантности данных: в любом программном переключателе значащим является только адрес, т.е. чтобы осуществить ввод нового слова состояния, необходимо записать произвольную информацию по адресу соответствующего программного переключателя. При этом в РСС запишутся младшие восемь разрядов ША ЦП (*A0* - *A7*).

Все программные переключатели модуля оперативной памяти лежат в пределах диапазона адресов сигнала $\overline{I/O\ SELECT}$ (256 адресов) того разъема внутреннего интерфейса ПЭВМ, в который установлен модуль оперативной памяти, и указаны в табл. 8.4.

Таким образом, например, для модуля расширения ООП запись произвольной информации по адресу, соответствующему программному переключателю C60A, обеспечивает подключение к адресам 8000 - BFFF массива "2" модуля оперативной памяти. А для модуля псевдоПЗУ, установленного в разъем *X5* платы памяти интерфейса, запись по адресу C420 вызывает подключение массива емкостью 16К байт к адресам D000 - FFFF и части этого массива емкостью 4К байт к адресам D000 - DFFF, а также блокирует стандартное ПЗУ и разрешает считывание информации из ОЗУ, одновременно запрещая запись в модуль.

Чтение слова состояния центральным процессором. Слово состояния модуля является программнодоступным для центрального процессора, что облегчает использование модуля оперативной памяти в составе ПЭВМ. Чтобы прочитать слово состояния, центральный процессор должен обратиться по любому из 256 адресов сигнала $\overline{I/O\ SELECT}$ разъема, в который установлен модуль оперативной памяти. При этом к шине данных центрального процессора подключается буферный регистр слова состояния (БРСС), тем самым обеспечивая передачу слова состояния по ШД ЦП в центральный процессор.

Кроме слова состояния, в его старшем разряде считываемого байта содержится информация о функциональном исполнении модуля оперативной памяти - признак исполнения модуля. Для модуля расширения ООП признак исполнения модуля равен логическому нулю, а для модуля псевдоПЗУ - логической единице.

Установка РСС в нулевое состояние. Чтобы исключить неоднозначность первоначальной установки РСС после включения питания, вводится принудительная начальная установка РСС: с помощью цепочки ***R17, C11*** (см. рис. 8.31) после включения питания происходит обнуление РСС. Это означает: для модуля расширения ОПП - работа в режиме хранения до тех пор, пока модуль не будет выбран; для модуля псевдоПЗУ - готовность к загрузке информации с ГМД и гарантированная работа ПЭВМ со стандартным ПЗУ, пока вся необходимая информация не будет загружена в псевдоПЗУ.

ОПЕРАЦИЯ ЧТЕНИЯ

Операцию чтения осуществляет центральный процессор во время положительной фазы импульсов $\varphi 0$ (рабочей фазы), в течение которой процессор получает доступ к ОЗУ модуля при условии, что он выбран (см. рис. 8.1).

Состояние дополнительного разряда ***A16*** определяет один из массивов ОЗУ (ОЗУ1 или ОЗУ2), в котором находится адресуемая центральным процессором ячейка памяти. В этом случае один из массивов ОЗУ переходит из режима хранения в рабочий режим, а второй остается в режиме хранения. Мультиплексор адреса обеспечивает передачу кода адреса с ША ЦП на адресные входы микросхем ОЗУ. Выбранная из адресуемой ячейки памяти информация поступает с выходов микросхем памяти на буферный регистр данных (РБД), который передает поступившие данные на ШД ЦП. Мультиплексор адреса управляется по входу ***B*** импульсами $\varphi 0$, а по входу ***A*** - сигналом ***C D17***, соответствующим сигналу $\overline{RAS}$, вырабатываемому во время отрицательной фазы $\varphi 0$. Сигнал $\overline{RAS}$ формируется на выходе ***13*** элемента D17 (см. рис. 8.31) в соответствии с импульсами 2 и 14 МГц.

ОПЕРАЦИЯ ЗАПИСИ

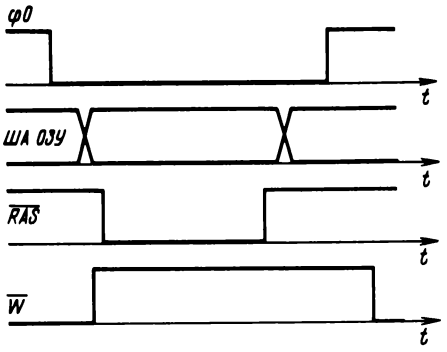
Как и операция чтения, операция записи осуществляется во время положительной фазы импульсов $\varphi 0$ (см. рис. 8.2). Низкий уровень сигнала $\overline{W}$ переводит в режим записи микросхемы одного из массивов ОЗУ (***ОЗУ1*** и ***ОЗУ2***) в зависимости от состояния дополнительного разряда ***A16***. По сигналу $\overline{CAS}$ (см. рис. 8.31) производится запись байта данных, поступающего с ШД ЦП в соответствующий массив (***ОЗУ1*** или ***ОЗУ2***) по адресу, определяемому кодом адреса, поступающим через мультиплексор адреса с ША ЦП. Во время операции записи РБД всегда переводится в состояние высокого импеданса, тем самым отключаясь от ШД ЦП.

Отметим, что выбор ОЗУ1 или ОЗУ2 осуществляется не сигналом $R/\overline{W}$, а 16-м дополнительным разрядом, формируемым на выходе 4 РСС. Этим сигналом на выходе ***3*** или ***6*** элемента ***D12*** формируется сигнал, соответствующий выбранному ОЗУ1 или ОЗУ2.

РЕГЕНЕРАЦИЯ ПАМЯТИ

Для используемых в модуле оперативной памяти микросхем динамической памяти необходимо обеспечить 128 циклов регенерации не более чем за 2 мс перебором адресов ***A0*** - ***A7***, соответствующих сигналу $\overline{RAS}$. С этой целью на

Рис. 8.33. Временные диаграммы регенерации



модуле оперативной памяти используется метод принудительной регенерации, при котором в течение каждого отрицательного уровня импульсов φ0 (фазы регенерации) осуществляется один цикл регенерации (рис. 8.33). На это время мультиплексор переключается на передачу адресов, вырабатываемых счетчиком регенерации (СР). Переброс счетчика осуществляется после каждого очередного цикла регенерации по фронту импульсов φ0.

ПРАВИЛА ИСПОЛЬЗОВАНИЯ МОДУЛЯ ОПЕРАТИВНОЙ ПАМЯТИ

Правила использования модуля расширения ООП

1. Модуль расширения ООП может быть установлен в любой из свободных разъемов (*X1*, *X3* - *X7*) модуля памяти и интерфейса с минимальным объемом ООП в 32К байт (микросхемы К565РУ6).

2. Модуль расширения ООП может быть установлен только в разъем *X7* платы памяти и интерфейса с объемом ООП более 32К байт [исполнение с микросхемами К565РУ5 (А, Б, В, Г, Д, Д1)].

3. Доступ к ОЗУ модуля осуществляется только через поле адресов 8000 - BFFF. Пока модуль не выбран, доступ к ОЗУ модуля невозможен, и адреса 8000 - BFFF занимает соответствующая часть ООП.

4. Выбор модуля осуществляется переключением в состояние логической единицы третьего разряда (*A3*) слова состояния модуля. При этом часть ООП, адресуемая через поле адресов 8000 - BFFF, исключается из адресного пространства процессора.

5. Вся оперативная память модуля разбита на массивы по 16К байт, и к адресам 8000 - BFFF подключается один из них. Номер подключаемого массива определяется тремя разрядами *A0* - *A2* слова состояния.

6. Блокировка записи в ОЗУ модуля осуществляется переключением в состояние логической единицы четвертого разряда (*A4*) слова состояния.

7. При использовании более одного модуля расширения ООП необходимо устанавливать платы в соседние разъемы [для исполнений с К565РУ5 (А - Д1)], начиная от разъема *X7*; при этом обеспечивается выполнение приоритетных правил выбора модуля:

модули, устанавливаемые в разъемы с более младшим номером, имеют более высокий приоритет;

выбор модуля с более высоким приоритетом обеспечивает подключение к адресам 8000 - BFFF массива оперативной памяти, расположенного на этом модуле;

все остальные платы с более низким приоритетом независимо от того, выбраны они или нет, переводятся в режим хранения;

чтобы обратиться к оперативной памяти, расположенной на модуле, имеющем более низкий приоритет, необходимо предварительно перевести в режим хранения все модули, имеющие более высокий приоритет, чем требуемый модуль.

Правила использования модуля псевдоПЗУ:

1. Модуль может быть установлен в любой разъем платы памяти и интерфейса, кроме разъема ***X2***. (В базовом варианте ***X3***.)
2. Модуль доступен по адресам D000 - FFFF. При этом возможны два режима работы в зависимости от состояния пятого разряда (***A5***) слова состояния модуля. В случае, если ***A5*** переключен в состояние логического нуля, считывание из модуля запрещено, а разрешена запись в модуль по адресам D000 - FFFF. Соответственно стандартное ПЗУ системы разблокировано. Переключение разряда ***A5*** в логическую единицу разрешает считывание из ОЗУ модуля по адресам D000 - FFFF, блокирует запись в модуль и стандартное ПЗУ системы.

Основной массив подключается к адресам E000 - FFFF. На адреса D000 - DFFF возможно подключение оставшегося дополнительного массива емкостью 4К байт. Для этого необходимо установить в логическую единицу шестой разряд (***A6***) слова состояния.

Правила работы со словом состояния

1. Словом состояния модуля является совокупность из семи младших разрядов программного переключателя, соответствующего сигналу ***I/O SELECT*** разъема, в которой установлена плата. Старший разряд программного переключателя является функциональным признаком модуля: равен нулю для модуля расширения ОПП, равен единице для модуля псевдоПЗУ.
2. Чтобы записать новое слово состояния в модуль, необходимо записать произвольную информацию по адресу программного переключателя, соответствующему сигналу ***I/O SELECT*** с необходимым значением восьми младших разрядов адреса, т.е. по адресу CNXX, где N = 1 + 6.
3. Чтобы прочитать данное слово состояния модуля, необходимо осуществить чтение по тому же адресу (CNXX).
4. После включения питания происходит автоматическая установка нулевого слова состояния.

8.5. РАБОТА КОНТРОЛЛЕРА НГМД

Контроллер НГМД (рис. 8.34) обеспечивает работу с компьютером одного или двух накопителей типа ЕС5088, ЕС5088.01, ЕС5089 и т.п. со скоростью 250 бит/с.

Функционально модуль содержит четыре узла:

управления работой ячейки;

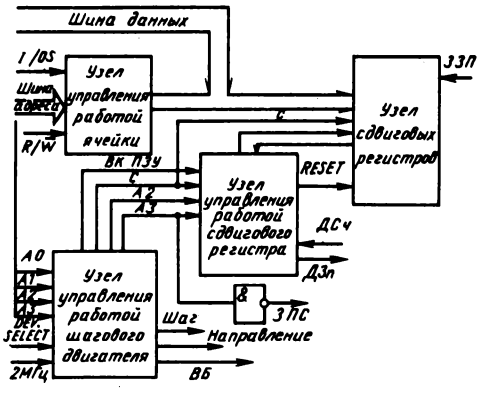
управления работой шагового двигателя;

управления работой сдвигового регистра;

сдвигового регистра.

Узел управления работой ячейки контроллера (рис. 8.35) содержит два

Рис. 8.34. Структурная схема модуля контроллера диска



шинных формирователя (*D18*, *D19*), и ПЗУ (*D1*), хранящее драйвер контроллера НГМД. Узел начинает работу по сигналу $\overline{I/OS}$ (низкий уровень) выборкой команды из ПЗУ, соответствующей адресу на восьми младших разрядах шины адреса центрального процессора. Узел управления работой шагового двигателя построен на элементах *D2* - *D9*, *D13*, *D14*, *D21*.

Дешифратор *D7* включается в работу импульсом длительностью 80 - 100 нс, сформированному элементом *D3* (одновибратор) из сигнала *DS*. В соответствии с кодом, выставленным на разрядах *A0*, *A1*, *A2* шины адреса центрального процессора дешифратор вырабатывает сигналы выбора одного из триггеров: соответственно *D4*, *D5*, *D6*, *D21*.

При поступлении на контроллер адреса C0X9 включается двигатель вращения ГМД, а при поступлении адреса C0X8 - выключается. Под X здесь и ниже понимается шестнадцатеричная цифра, которая вычисляется по формуле

$$X = 8 + N,$$

где N - номер разъема, к которому подключен контроллер.

Триггер *D4* вырабатывает сигнал, который, проходя через одновибратор D32 и микросхемы *D9.4*, *D8.4*, формирует сигнал включения - выключения двигателя. Этот же сигнал открывает микросхему *D13.1* для прохождения сигналов частотой 2 МГц на синхровходы регистров сдвига *D14* и *D16* и инверторы *D8*, обеспечивая прохождение сигналов, появляющихся на выходе триггера *D4*, при поступлении адреса C0XA (включение привода *1*) и C0XB (включение привода *2*), которые являются сигналами выбора НГМД.

В каждый момент времени контроллер может управлять только одним НГМД. Выбор НГМД осуществляется сигналами выбора привода. При включении одного из приводов другой отключается.

В случае использования НГМД с фазовым управлением шагового двигателя триггеры *D6* и *D21* вырабатывают четыре сигнала $\varphi 0$, $\varphi 1$, $\varphi 2$, $\varphi 3$ при поступлении адресов C0X0 - C0X7.

Контроллер может работать в режиме чтения или в режиме записи.

Режим записи организуется выполнением любой операции процессора по адресу C0XF. В этом случае на выходе *8* триггера *D5* формируется сигнал уровня логической единицы - *A3*, который поступает на вход *5* ПЗУ и на входы *1* и *2* *D17*, формируя на выходе *3* этой микросхемы уровень логического нуля сигнала "ЗПС", идущего в НГМД, что соответствует режиму записи.

Сигнал $\overline{DS}$, поступая на первые выводы микросхем *D10* и *D11*, открывает их. Сигнал $R/\overline{W}$ на выходе *13* микросхемы *D2.4* имеет уровень логической единицы и, поступая на входы *15* схем *D10* и *D11*, переводит их в соответствующий режим работы, т.е. шинные формирователи передают данные, идущие из центрального процессора в параллельном коде, на входы регистра сдвига *D15*.

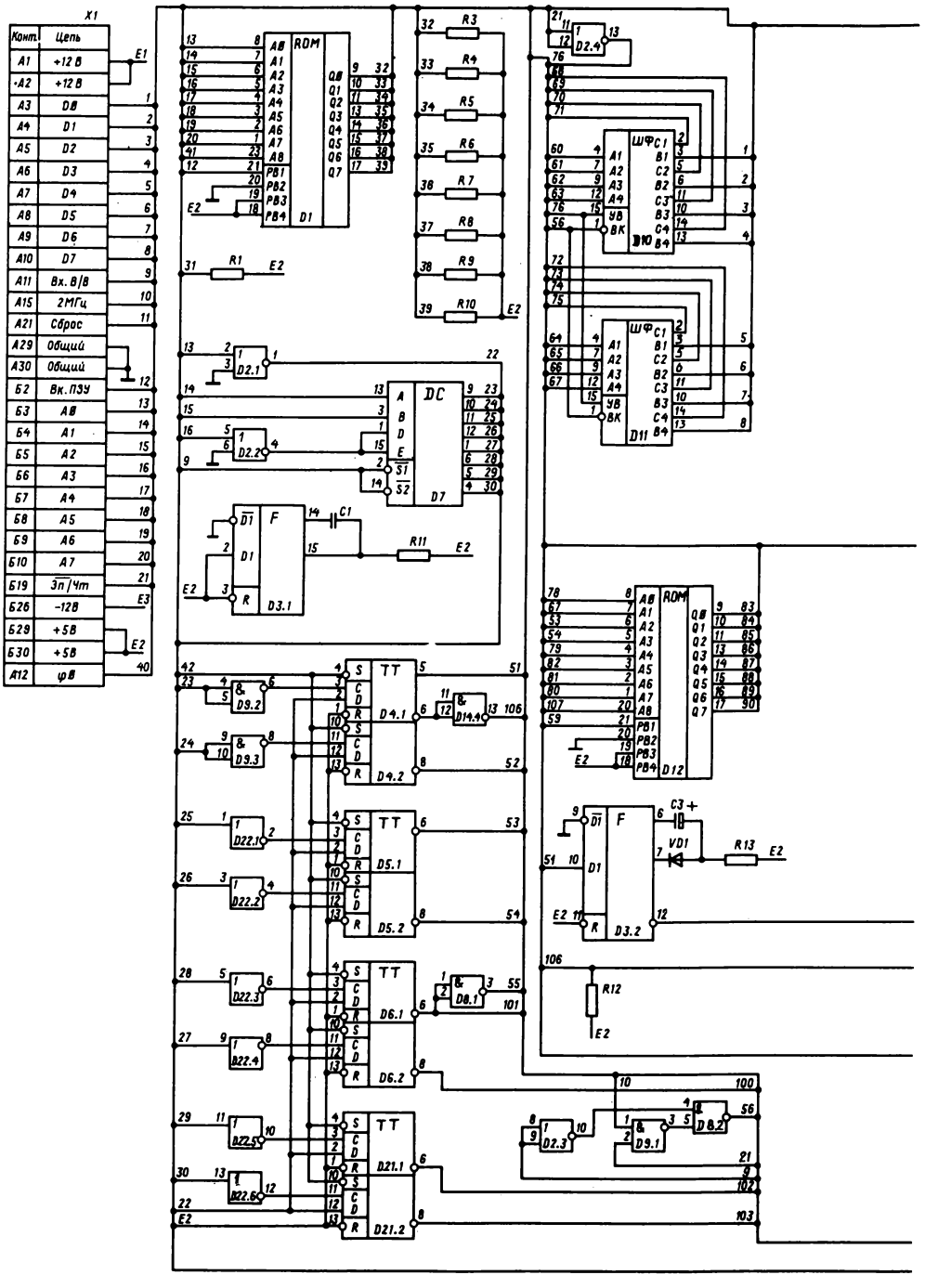
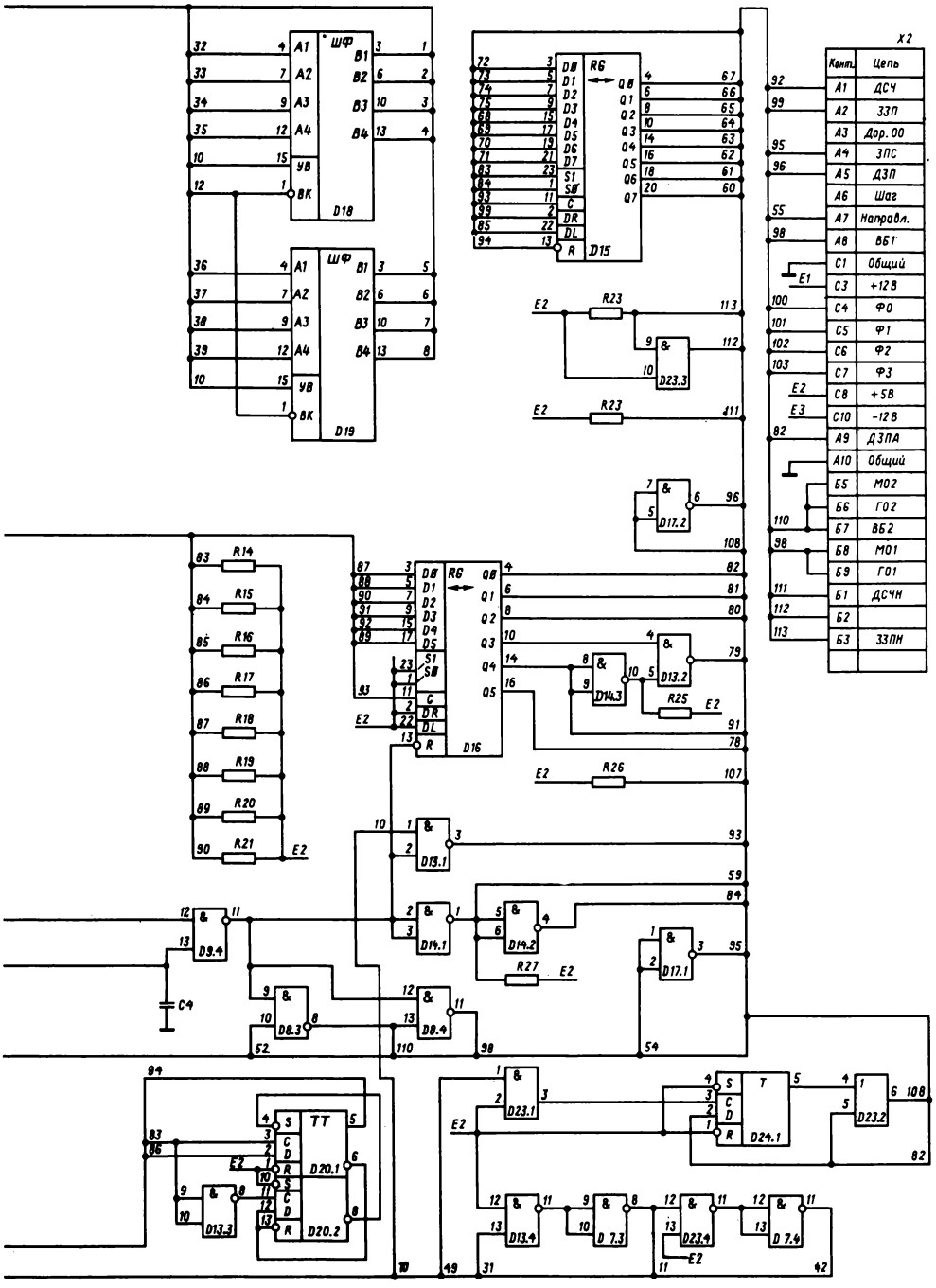


Рис. 8.35. Принципиальная схема контроллера НГМД


ШФ
D18
D19
RG
D15
D16
R23
R25
R26
R27
R14
R15
R16
R17
R18
R19
R20
R21
C4
TT
D20.1
D20.2
T
D24.1
X2
Конт.
Цепь
A1 ДСЧ
A2 ЗЗП
A3 Дор. 00
A4 ЗПС
A5 ДЗП
A6 Шаг
A7 Направл.
A8 ВБ1
C1 Общий
C3 +12В
C4 Ф0
C5 Ф1
C6 Ф2
C7 Ф3
C8 +5В
C10 -12В
A9 ДЗПА
A10 Общий
Б5 М02
Б6 Г02
Б7 ВБ2
Б8 М01
Б9 Г01
Б1 ДСЧН
Б2
Б3 ЗЗПН

Регистр сдвига сигналами, поступающими с выходов *9* и *10* ПЗУ *D12*, устанавливается в режим параллельного ввода. Затем этими же сигналами регистр переводится в режим сдвига вправо. В этом случае регистр сдвига выдает информацию побитно с выхода *4* на вход *7* ПЗУ (*D12*). С ПЗУ информация подается на вход элемента *D16*, постоянно включенного в режим параллельного приема. Далее через *D23* и *D24* данные поступают на НГМД. Сигнал *ЗЗП*, поступающий с НГМД на выход *2* элемента *D15*, закрывая его, указывает на то, что на данный НГМД нельзя записывать информацию.

Режиму чтения соответствует низкий уровень сигнала *A3* и соответственно высокий уровень сигнала *ЗПС*, идущего в НГМД. Режим чтения организуется выполнением любой операции процессора с адресом о C0XE. После этого можно читать байты с фиксированной дорожки.

Чтение осуществляется из регистра чтения (Д C0X0) до тех пор, пока старший разряд считанных данных не станет равным единице. После этого значение регистра чтения обнуляется, т.е. считывание байта из регистра чтения возможно только один раз. Данные с разъема, подключенного к НГМД, поступают в последовательном коде на выход *15* элемента *D16*, затем через ПЗУ на выход *22* регистра сдвига *D16*, который включен сигналами *83* = 1 и *84* = 1 (поступающими с ПЗУ) в режим сдвига влево. Поскольку данные на ГМД записаны таким образом, что старший бит является логической единицей, то признаком заполнения регистра сдвига является наличие логической единицы на выходе *4* ПЗУ. Принимая этот сигнал, ПЗУ формирует сигналы: *83* = 1 и *84* = 1, переключающие регистр сдвига в режим параллельной выдачи информации, которая через шинные формирователи *D10* и *D11*, включенные сигналами $R/\overline{W}$ = 1 и $\overline{DS}$ = 0, поступает в шину данных. Одновременно с сигналами *83* и *84* ПЗУ формирует сигнал *86*, поступающий через *D20* на вход *R* регистра сдвига, сбрасывая его в нуль.

8.6. МОДУЛЬ ПАРАЛЛЕЛЬНОГО И ПОСЛЕДОВАТЕЛЬНОГО ИНТЕРФЕЙСА

Модуль параллельного и последовательного интерфейса (МППИ) работает в составе ПЭВМ и предназначен для управления и обмена информацией с периферийными устройствами (принтером, графопостроителем, графическим и алфавитно-цифровым дисплеем, модемом телефонного канала и т.д.).

МППИ в компьютере может быть установлен в любом из разъемов *X3* - *X7*. МППИ имеет ППЗУ емкостью 256 байт, предназначенное для хранения драйверных программ. Схема параллельного интерфейса позволяет обмениваться данными по двум восьмиразрядным шинам и двум четырехразрядным шинам. Нагрузочная способность одной выходной шины параллельного интерфейс по уровню логического нуля не более 1,8 мА. Емкость нагрузки не более 100 пф. Обмен информацией по последовательному интерфейсу осуществляется со скоростью 300 - 9600 бит/с. Длина передаваемых и принимаемых посылок 5 - 8 бит. Входные и выходные уровни сигналов этого канала составляют ±12 В. Протокол обмена по последовательскому каналу соответствует интерфейсу обмена RS 232 с.

СОСТАВ МППИ

МППИ содержит следующие узлы (рис. 8.36):
формирования сигналов *УФС*;

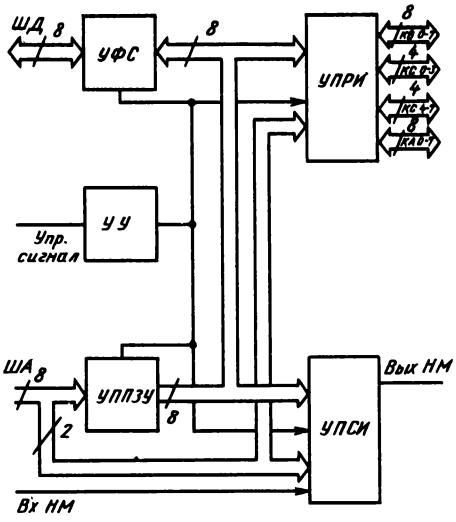


Рис. 8.36. Функциональная схема параллельного последовательного интерфейса

перепрограммируемой постоянной памяти ***УППЗУ***;

управления ***УУ***;

параллельного интерфейса ***УПРИ***;

последовательного интерфейса ***УПСИ***.

УФС согласует шину данных ПЭВМ с внутренней шиной ячейки; реализован на двух интегральных микросхемах К589АП16.

УУ предназначен для формирования сигналов, управляющих вводом информации в ***УПРИ***, ***УПСИ*** и выводом из них; реализован на микросхемах серии К555.

УППЗУ предназначен для хранения специальных программ, определяемых типом ВУ и требованиями пользователя; реализован на интегральных микросхемах КР556РТ5, емкость программно доступной памяти составляет 256 байт.

УПРИ реализован на микросхеме КР580ВВ55; функциональная конфигурация ***УПРИ*** программируется с помощью системного математического обеспечения.

УПСИ реализован на микросхеме КР580ВВ51 и представляет собой программируемое устройство для преобразования параллельного восьмиразрядного кода в последовательный и наоборот.

РЕЖИМ РАБОТЫ МОДУЛЯ

Рассмотрим два режима работы модуля:
записи (ЗП) информации из ШД в ВУ;
чтения (ЧТ) информации из ВУ в ШД.

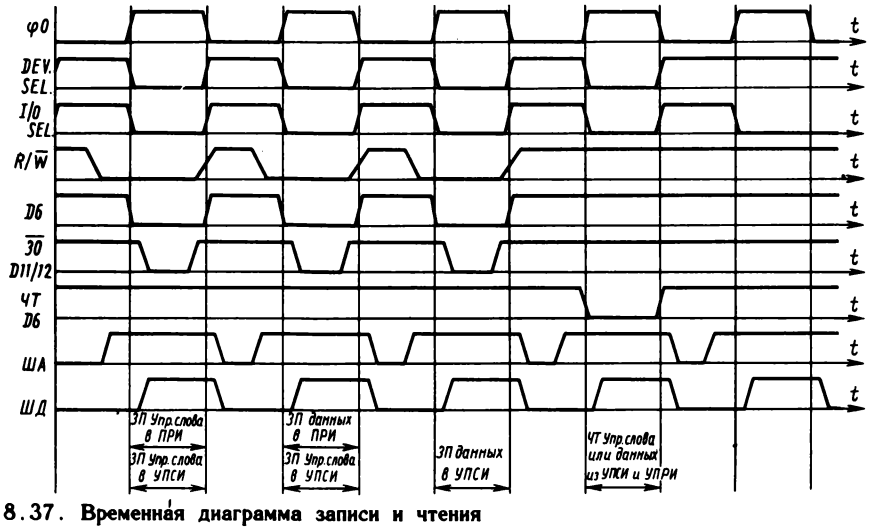
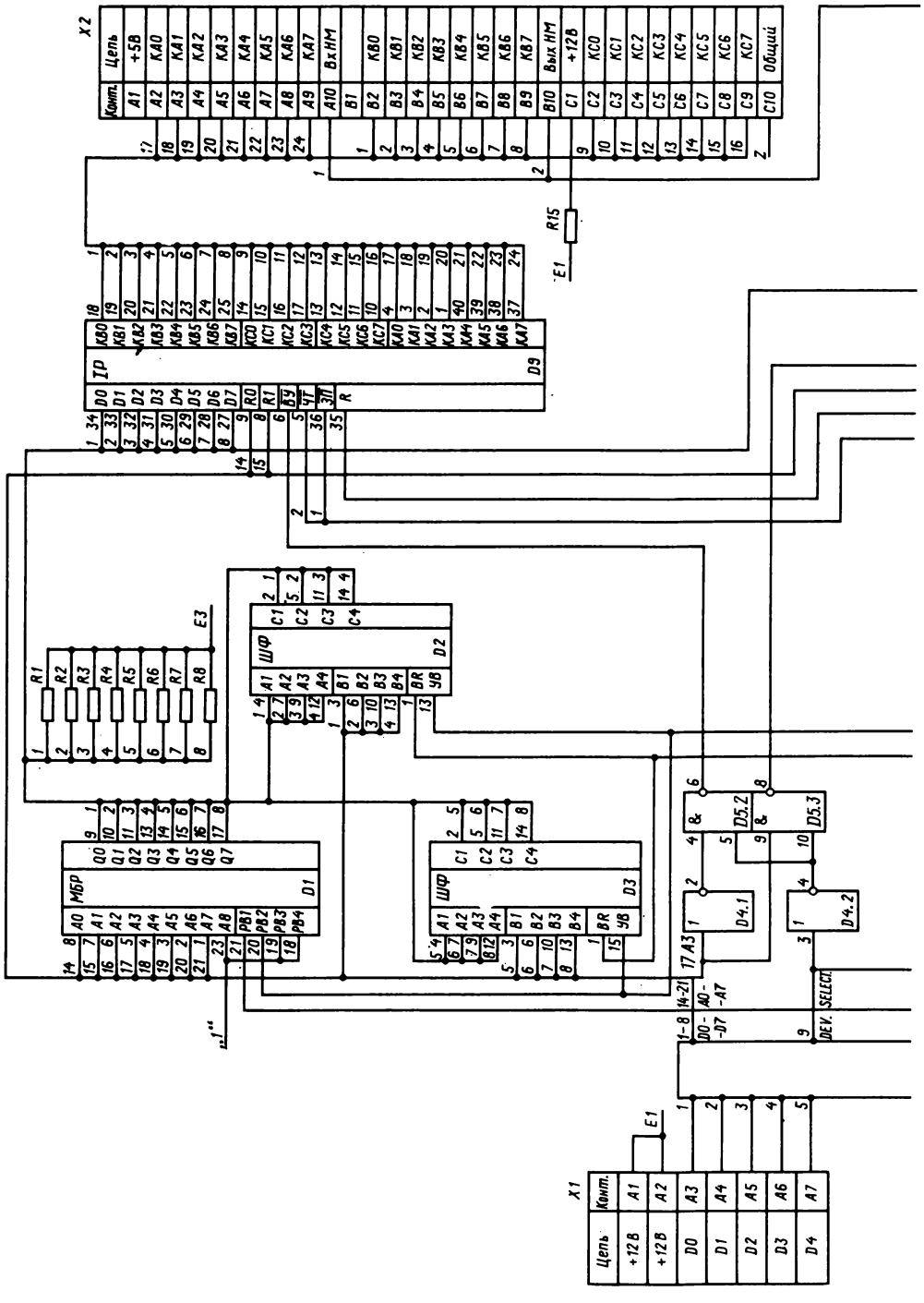


Рис. 8.37. Временная диаграмма записи и чтения


X2
Конт.
Цепь
+5В
КА0
КА1
КА2
КА3
КА4
КА5
КА6
КА7
Вх НМ
КВ0
КВ1
КВ2
КВ3
КВ4
КВ5
КВ6
КВ7
Вых НМ
+12В
КС0
КС1
КС2
КС3
КС4
КС5
КС6
КС7
Общий
R15
E1
D9
ТР
D0
D1
D2
D3
D4
D5
D6
D7
R
D2
ШФ
D3
МБР
D1
D5.2
D5.3
D4.1
D4.2
R1
R2
R3
R4
R5
R6
R7
R8
E3
X1
Конт.
Цепь
+12В
DO
D1
D2
D3
D4
SELECT
DEV.

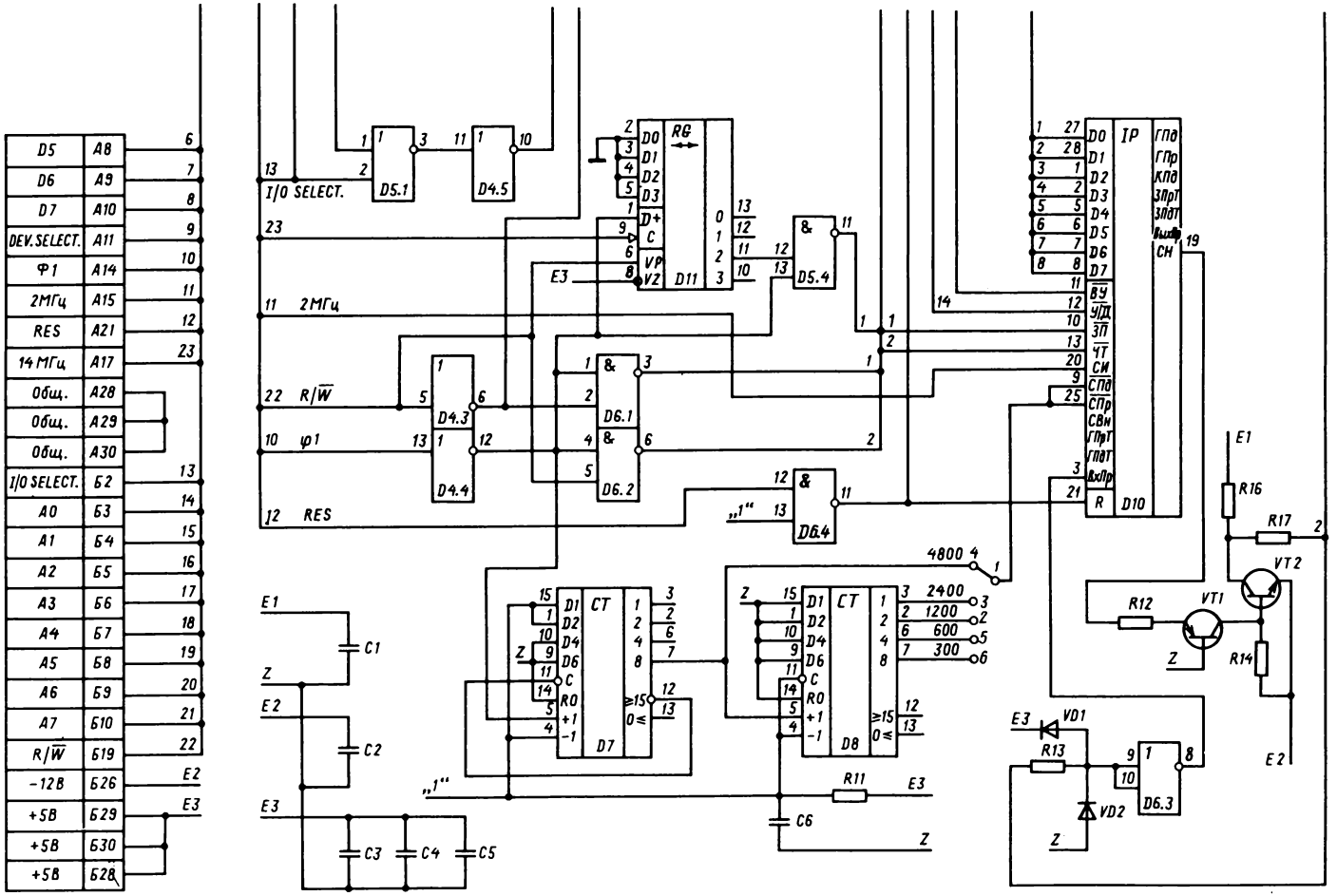


Рис. 8.38. Принципиальная схема модуля параллельного и последовательного интерфейса

Режим записи. Для обеспечения режима ЗП на микросхемы УПРИ и УПСИ необходимо подать следующие управляющие сигналы:

ВУ - выбор устройства;
ЗП - сигнал записи;
ЧТ - сигнал чтения;
А0 - разряд 0 адресной шины;
А1 - разряд 1 адресной шины;
D0 - D7 - разряды 0 - 7 шины данных.

Сигналы ***ВУ*** для ***УПРИ*** и ***УПСИ*** различны. Они определяются состоянием адресной шины ***А3***. Сигнал ***ЗП*** поступает на микросхемы в режиме записи, а сигнал ЧТ - в режиме чтения.

Разряды ***А0***, ***А1*** определяют работу УПРИ. Режим работы УПСИ определяет разряд ***А0*** [8].

Шины данных ***D0*** - ***D7*** предназначены для записи управляющего слова или данных УПРИ или УПСИ и чтения из УПРИ и УПСИ слова состояния или данных.

Временные соотношения управляющих сигналов приведены на рис.8.37.

Следует обратить внимание на то, что данные в УПРИ и в УПСИ записываются после программирования режима этих микросхем. Причем в УПРИ программирование режима производится в первом такте $\varphi 0$, а данные записываются во втором такте $\varphi 0$.

В УПСИ в первом такте $\varphi 0$ выполняет программирование инструкции режима, во втором такте $\varphi 0$ - программирование команды и только в третьем такте $\varphi 0$ данные из ШД передаются на выход УПСИ.

В режиме чтения в зависимости от состояния адресных шин ***А0***, ***А1*** производится чтение либо слова состояния микросхемы, либо информации с шин данных D0 - D7.

Рассмотрим формирование управляющих сигналов (рис. 8.38).

Сигнал $\overline{ВУ}$ для УПРИ (D9) формируется схемой И (***D5.2***), на входы которой подаются $\overline{DEV.SELECT}$ и $\overline{A3}$. Для микросхемы УПСИ (***D10***) сигнал ВУ формируется также схемой И (D5.3), на входы которой подаются $\overline{DEV.SELECT}$ и ***А3***.

Формирование сигнала ЗП производится микросхемами ***D11.1***, ***D11.2***. Сигнал $R/\overline{W}$ поступает на вход ***D4.3*** (в режиме записи сигнал $R/\overline{W}$ имеет уровень логического нуля). Инверсный $R/\overline{W}$ стробируется фазой $\varphi 1$ и поступает на одновибратор ***D11.1***. На вход ***ЗП*** микросхем УПРИ и УПСИ подается сигнал ***ЗП***, сформированный одновибратором ***D11.2***.

Информация с шины данных ПЭВМ поступает на входы ВКУФС через схему ИЛИ D5.1), куда поступают сигналы $\overline{I/O\ SELECT}$ и $\overline{DEV.\ SELECT}$.

Информация на входы ***D0*** - ***D7*** микросхем УПРИ и УПСИ может поступать и из ППЗУ (***D1***) по заданному центральным процессором "Агат" адресу.

Режим чтения. Сигналы $\overline{ВУ}$ для УПРИ и УПСИ в режиме чтения формируются так же, как и в режиме записи (см. рис. 8.37); при этом сигнал ***R/W*** имеет уровень логической единицы.

8.7. ИМПУЛЬСНЫЙ БЛОК ПИТАНИЯ

Импульсный блок питания предназначен для питания всех узлов и блоков ПЭВМ. Блок питания формирует три напряжения: +5, +12, -12 В. Характеристики каждого выходного канала приведены в табл. 8.10.

Таблица 8.10

Характеристики блока питания

Нормальное выходное напряжение, В	Сила тока нагрузки, А		Нестабильность выходных напряжений, В	Пульсация выходных напряжений, В
	минимальная	максимальная		
+5	3	6,5	±0,1	±0,05
+12	0	0,9	+1,20 -0,55	±0,1
-12	0	0,2	+1,20 -0,55	±0,1

Импульсный блок питания выполнен на базе транзисторного преобразователя частоты. Функциональная схема блока приведена – на рис. 8.39, принципиальная на рис. 8.40.

Схема основана на принципе обратного преобразования, сущность которого заключается в том, что в момент открытого состояния силового транзисторного ключа (прямого хода инвертора) в магнитоприводе трансформатора (зазоре между пластинами) накапливается энергия. В момент закрытия транзистора (обратный ход инвертора) накопленная энергия поступает на выходные выпрямители и далее через сглаживающие фильтры в нагрузку.

Регулируемый инвертор представляет собой однотактный (на одном такте транзисторного ключа) мощный автогенератор с глубоко насыщенной трансформаторной обратной связью. Выходная мощность регулируемого инвертора прямо пропорциональна силе тока, протекающего в первичной обмотке трансформатора в момент перед началом обратного хода. Таким образом, управляя моментом переключения силового транзисторного ключа из состояния насыщения в состояние отсечки, можно менять количество энергии, запасенной в магнитоприводе трансформатора, т.е. регулировать выходную мощность.

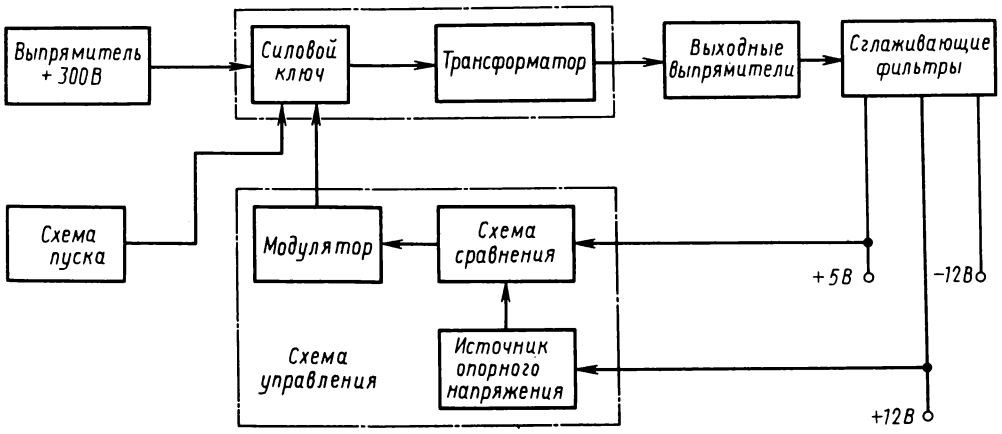


Рис. 8.39. Функциональная схема блока питания ПЭВМ

Регулирование осуществляется по цепи обратной связи сравнением выходного напряжения с опорным (поступающим с источника опорного напряжения) и заданием в зависимости от полученного возмущения порогового значения силы коллекторного тока силового транзистора. Таким образом, схема управления в зависимости от изменения выходного напряжения задает и регулирует момент переключения инвертора с прямого хода на обратный.

Для питания регулируемого инвертора используется выпрямитель.

Выпрямитель сетевого напряжения (рис. 8.39) собран на высокочастотных диодах ***VD1 - VD4*** (KD209). Диодный мост выпрямляет напряжение в сети до +300 В. Для ограничения силы тока импульса, протекающего через диоды выпрямителя в момент включения блока питания в сеть, в схему включен резистор ***R1*** мощностью 2 Вт.

Выпрямленный импульс фильтруется конденсаторами ***C5 - C7***. Фильтр высоких частот (на конденсаторе ***C7***) служит для устранения перегрева электролитов ***C5*** и ***C6***, образующих фильтр низких частот.

Другой фильтр низких частот, образованный катушкой ***L1*** и конденсаторами ***C1*** и ***C2***, уменьшает высокочастотную пульсацию, передаваемую блоком питания в сеть.

Выпрямленное и подфильтрованное напряжение +300 В поступает на схему регулируемого инвертора.

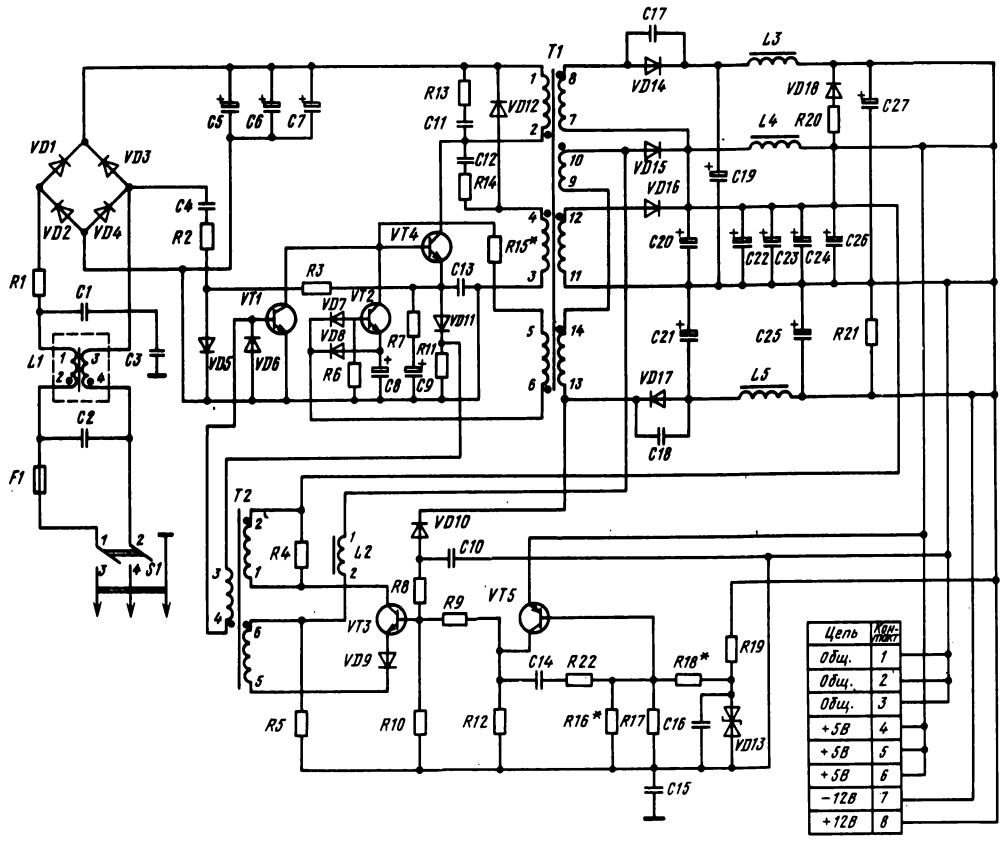


Рис. 8.40. Принципиальная схема блока питания ПЭВМ

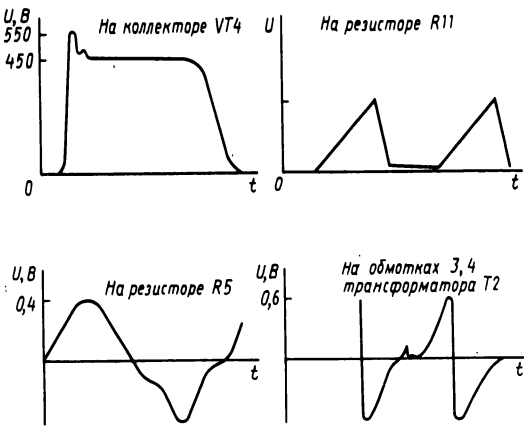


Рис. 8.41. Опоры напряжения на элементах блока питания

Для пуска автогенератора (инвертора) служит схема первоначального пуска, включенная в схему источника питания.

Схема первоначального запуска работает только в момент подключения блока питания к сети и содержит выпрямитель на ***VD5***, ограничительный резистор ***R3*** и конденсатор ***C4*** (рис. 8.39). Для запуска автогенератора необходимо открыть силовой ключ, собранный на ***VT4***. Для включения ***VT4*** необходимо на базу подать положительный относительно эмиттера потенциал. Это условие равносильно подаче на эмиттер ***VT4*** отрицательного импульса.

Диод ***VD5*** пропускает положительные полуволны переменного сетевого напряжения. Отсеченные таким образом отрицательные импульсы через ***R3*** поступают на ***VT4***. Для уменьшения амплитуды импульсов до 3 В в схему включен электролит ***C4***.

"Мягкое" включение силового ключа (***VT4***) обеспечивается ***R*** - ***C***-цепочкой, собранной на ***R7*** и ***C9***. Диод ***VT11*** запирает резистор ***R11***, имеющий маленькое сопротивление (0,25 Ом), обеспечивая тем самым отрицательный потенциал 3,4 В на эмиттере ***VT4***.

Таким образом, в момент включения блока питания в сеть открывается ***VT4***, обеспечивая начало генерации регулируемого инвертора.

Регулируемый инвертор представляет собой однотактный мощный автогенератор с трансформаторной обратной связью (обмотки 5 - 6), который вырабатывает прямоугольные импульсы с частотой 16 - 30 кГц.

В момент прямого хода транзистор ***VT4*** открыт, и в первичной обмотке ***1 - 2*** трансформатора ***T1*** протекает линейно нарастающий ток. В цепи вторичной обмотки ***9 - 10*** тока нет, что обеспечивается включением в цепь этой обмотки диода ***VD15***. Таким образом, пока ***VT4*** находится в насыщенном состоянии, в магнитоприводе трансформатора ***T1*** происходит накопление энергии.

По сигналу со схемы управления в момент времени t_1 положительный прямоугольный импульс открывает транзистор ***VT1***. Он насыщается и шунтирует переход база - эмиттер силового ключа ***VT4***, закрывая его. Это вызывает прекращение тока в обмотке ***1 - 2*** трансформатора ***T1***.

По закону электромагнитной индукции напряжение на обмотке ***9 - 10*** трансформатора меняет свой знак на противоположный.

Диод ***VD15*** открывается, и начинается процесс передачи энергии из сердечника трансформатора в нагрузку (это и есть обратный ход преобразователя); сила тока и напряжение изменяются по закону, приведенному на рис. 8.41.

ЭДС нагрузки изменяется по закону

$$E = -L\,\Delta I_н/\Delta t,$$

где L - индуктивность; $I_н$ - сила тока нагрузки.

Энергия будет передаваться в нагрузку до тех пор, пока ток на обмотке 9 - 10 не станет равным нулю. Это обеспечивается обратной связью через обмотку 5 - 6.

Выходное напряжение пропорционально силе тока в первичной обмотке трансформатора. Поэтому изменение тока приводит к изменению выходного напряжения. Такая регулировка осуществляется схемой управления. Рассасывание неосновных носителей VT4 осуществляет схема рассасывания.

Схема рассасывания осуществляет форсированное запирание силового транзистора ***VT4*** путем подачи на его базу кратковременного отрицательного импульса в момент подачи обратного хода. Форсированное закрытие ***VT4*** необходимо для уменьшения потерь и повышения КПД автогенератора.

В насыщенном состоянии силового транзистора ***VT4*** через обмотку ***5 - 6*** трансформатора ***T1*** и базу ***VT4*** протекает ток, сила которого ограничена резистором ***R15***. Этот резистор является подстроечным элементом при настройке блока питания. Увеличение сопротивления резистора ***R15*** приводит к уменьшению силы тока базы ***VT4***. Ток протекает по следующей цепи: верхняя точка обмотки ***5 - 6*** трансформатора ***T1*** → ***R15*** → база - эмиттер открытого ***VT4*** → ***C13*** → → ***C8*** → открытый ***VD8*** и параллельно ***R6*** → открытый ***VD7***. Далее оба тока объединяются и поступают в обмотки ***5 - 6*** трансформатора ***T1***. Конденсатор ***C8*** при этом заряжается таким образом, что на верхней пластине накапливается отрицательный заряд.

С начала обратного хода напряжение на первичной обмотке ***5 - 6*** трансформатора ***T1*** уменьшается до нуля, а затем меняется на противоположное, что приводит к открыванию ***VT2***, базовый ток которого определяется ***R6*** и ***C8***. Диоды ***VD7*** и ***VD8*** запираются, к эмиттеру ***VT2*** приложен отрицательный потенциал от ***C8***, а к базе ***VT2*** - положительный потенциал от ***C8*** через ***R6***. В результате к базе ***VT4*** подключается отрицательная пластина ***C8*** и происходит интенсивное рассасывание неосновных носителей заряда в области базы транзистора ***VT4*** и его форсированное запирание.

Обмотка ***4 - 3*** трансформатора ***T1*** является рекуперационной, ограничивающей выброс напряжения на силовом ключе коллектора ***VT4*** (рис. 8.41).

Выброс на силовом ключе возникает из-за индуктивности рассеивания первичной обмотки; в этом случае напряжение на обмотке ***4 - 3*** превышает напряжение источника питания (+300 В). Тогда диод ***VD12*** открывается, и излишки напряжения отдаются обратно источнику. Таким образом, напряжение на коллекторе ***VT4*** ограничивается на уровне +600 В.

Элементы ***C11***, ***R13***, ***C12***, ***R14*** формируют траекторию рабочей точки так, чтобы при открытом состоянии ***VT4*** на коллекторе был нуль, при закрытом - максимум. Конденсатор C8 устраняет отрицательную обратную связь шунтируя диоды ***VD11*** и резистор ***R11***.

Выходные выпрямители выполнены по однопериодной схеме на высокочастотных диодах КД 213А и КД 212А. Обмотки ***10 - 9*** и ***12 - 11*** соединены параллельно через диоды ***VD15***, ***VD16***. Обмотка ***8 - 7*** является вольтодобавочной, соединенной последовательно с ли- нией +5 В.

В канале +12 В установлена дополнительная стабилизация на ***VD18*** и ***R20***, работающих в режиме холостого хода.

Выходные фильтры образованы высокочастотными конденсаторами (до 100 кГц)

C20 - C24 и высокочастотными дросселями *L3, L4, L5 (L3 - 12* МГц, *L4* - 15 МГц и т.д.).

Параллельно диодам *VD14* и *VD17* включены высокочастотные конденсаторы, служащие для уменьшения так называемых столбовых помех в каналах +12 и -12 В. В цепи *VD15* и *VD16* стоят мощные сглаживающие электролиты *C22 - C26*.

Схема защиты предусматривает защиту от короткого замыкания, при котором сила тока нагрузки достигает максимального значения. Дроссели имеют малое сопротивление (несколько Ом), поэтому они накоротко замыкаются. Общий коэффициент положительной обратной связи, охватывающей силовой транзистор через трансформатор *T1*, становится меньше единицы, колебательный процесс в схеме срывается, и генерация прекращается, а цепь *C9*, *R7* ограничивает силу тока. Схема пуска пытается запустить автогенератор, кратковременно открывая транзистор *VT4*. При этом слышен характерный рокот, треск сердечника трансформатора *T1* с частотой 50 Гц. Устранение короткого замыкания приводит к автоматической установке нормальной работы.

Схема управления содержит модулятор на транзисторе *VT3* и трансформаторе *T2*, а также схему сравнения на *VT5*. Эта схема постоянно сравнивает опорное (эталонное) напряжение базы *VT5* с выходным напряжением +5 В, заведенным на эмиттер *VT5*. В качестве источника опорного напряжения используется параметрический стабилизатор на кремневом стабилитроне *VD13* и резисторе *R18*. Конденсатор *C16* уменьшает пульсацию выходного напряжения стабилизатора.

Делитель на резисторах *R17* и *R18* создает на базе *VT5* постоянный потенциал, не зависящий от изменения выходных напряжений. Следовательно, состояние *VT5* зависит только от изменения потенциала эмиттера, т.е. от выходного напряжения +5 В.

В момент закрытия ключа *VT4* диод *VD10* открыт, и конденсатор *C10* заряжается до -12 В. Это отрицательное напряжение через делители *R8* и *R10* поступает на базу *VD3*, закрывая его. В то же время линейно нарастающее напряжение закрывает диод *VD9*.

С начала работы ключа *VT4* напряжение на обмотках трансформатора *T1* меняет полярность, закрывая тем самым диод *VD10*. Конденсатор *C10* начинает разряжаться по цепи *R8* → *R9* → *R10* → *R12* и уменьшает отрицательный потенциал на базе *VT3*. В то же время на резисторе *R5* увеличивается отрицательное пилообразное напряжение, открывающее диод *VD9*.

В момент, когда потенциал базы транзистора *VT3* по отношению к эмиттеру уменьшится (т.е. на эмиттере минус больше, чем на базе) и достигнет некоторой пороговой величины, зависящей от сопротивления резистора *R12*, транзистор *VT3* открывается и в его коллекторной цепи через обмотку *1 - 2* трансформатора *T2* протекает ток, наводящий во вторичной обмотке (*5 - 6*) трансформатора *T2* импульс такой полярности, при которой еще больше (лавинообразно) открывается *V13*.

Это, в свою очередь, наводит короткий положительный импульс во вторичной обмотке *3 - 4* трансформатора *T2*, открывающий управляющий транзистор *VT1*. Длительность и амплитуда импульса определяются параметрами обмоток *1 - 2* и *5* - 6.

Предположим, что напряжение канала +5 В увеличилось. Это приводит к увеличению напряжения на коллекторе *VT4*; растущее отрицательное напряжение

откроет ***VT3*** несколько раньше по сравнению с номинальным режимом. Сформированный модулятором положительный импульс раньше откроет ***VT1*** и закроет ***VT4*** силового ключа, что приведет к понижению выходного напряжения в канале +5 В.

Этот процесс происходит непрерывно. Силовой ключ ***VT4*** работает на частоте 16 кГц при максимальной нагрузке, 30 кГц - при минимальной, 5 кГц - в режиме холостого хода.

Резистор ***R4*** определяет более быстрое реагирование на изменение выходного напряжения в канале +5 В. Если его убрать, то напряжение пульсации по входу +5 В будет не ±0,05 В, а ±0,1 В.

Резисторы ***R16***, ***R17*** определяют регулировку выходного напряжения в канале +5 В.

Регулировка в блоке питания осуществляется следующим образом:

выходное напряжение +5 В регулируется подбором ***R16*** (грубо) и ***R17*** (точно);

увеличение емкости конденсатора ***C16*** приводит к уменьшению пульсации выходного напряжения стабилизатора;

конденсаторы ***C17***, ***C18*** служат для уменьшения импульсных помех в каналах +12 и -12 В;

конденсатором ***C11*** регулируется мощность инвертора;

резистором ***R1*** ограничивается бросок силы тока через диоды моста в момент включения блока питания;

резистором ***R15*** регулируется сила тока базы силового ключа ***VT4*** увеличение сопротивления приводит к уменьшению силы тока базы VT4).

Г л а в а 9. ВНЕШНИЕ УСТРОЙСТВА

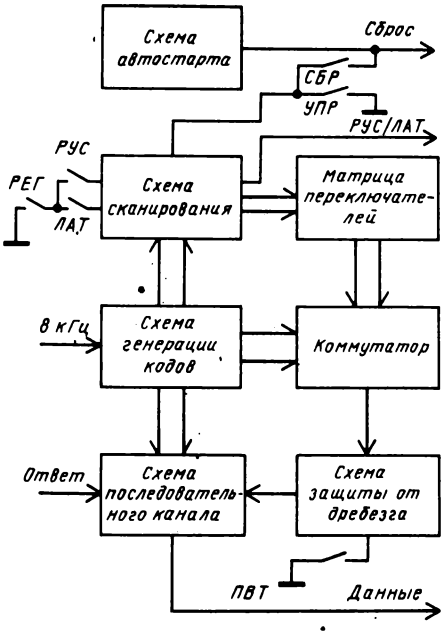
9.1. БЛОК КЛАВИАТУРЫ

Блок клавиатуры (БК), предназначенный для ввода информации в компьютер, организован по принципу матричного шифратора, в узлах которого размещены коммутационные элементы - клавиши. В исходном состоянии (если клавиша не нажата) происходит непрерывное сканирование узлов матрицы. При нажатии клавиши сканирование останавливается, а на счетчике фиксируется код строки и столбца нажатой клавиши, который передается на схему последовательного канала.

На рис. 9.1 приведена функциональная схема блока клавиатуры.

Тактовые импульсы частотой 8 кГц поступают на вход схемы генерации кодов, представляющей собой два четырехразрядных двоичных счетчика ***D11*** и ***D12*** (рис. 9.2). Счетчики преобразуют импульсы тактовой частоты, поступающие от системного блока, в двоичный код. Младшие разряды данного кода управляют схемой сканирования, реализованной на дешифраторе ***D13***, а старшие разряды - коммутатором, собранным на интегральных схемах ***D14***, ***D15***. На одном из выходов дешифратора ***D13*** устанавливаются сигнал логического нуля, сканирующий поле переключателей. Поле переключателей - это набор кнопочных переключателей,

Рис. 9.1. Функциональная схема блока клавиатуры



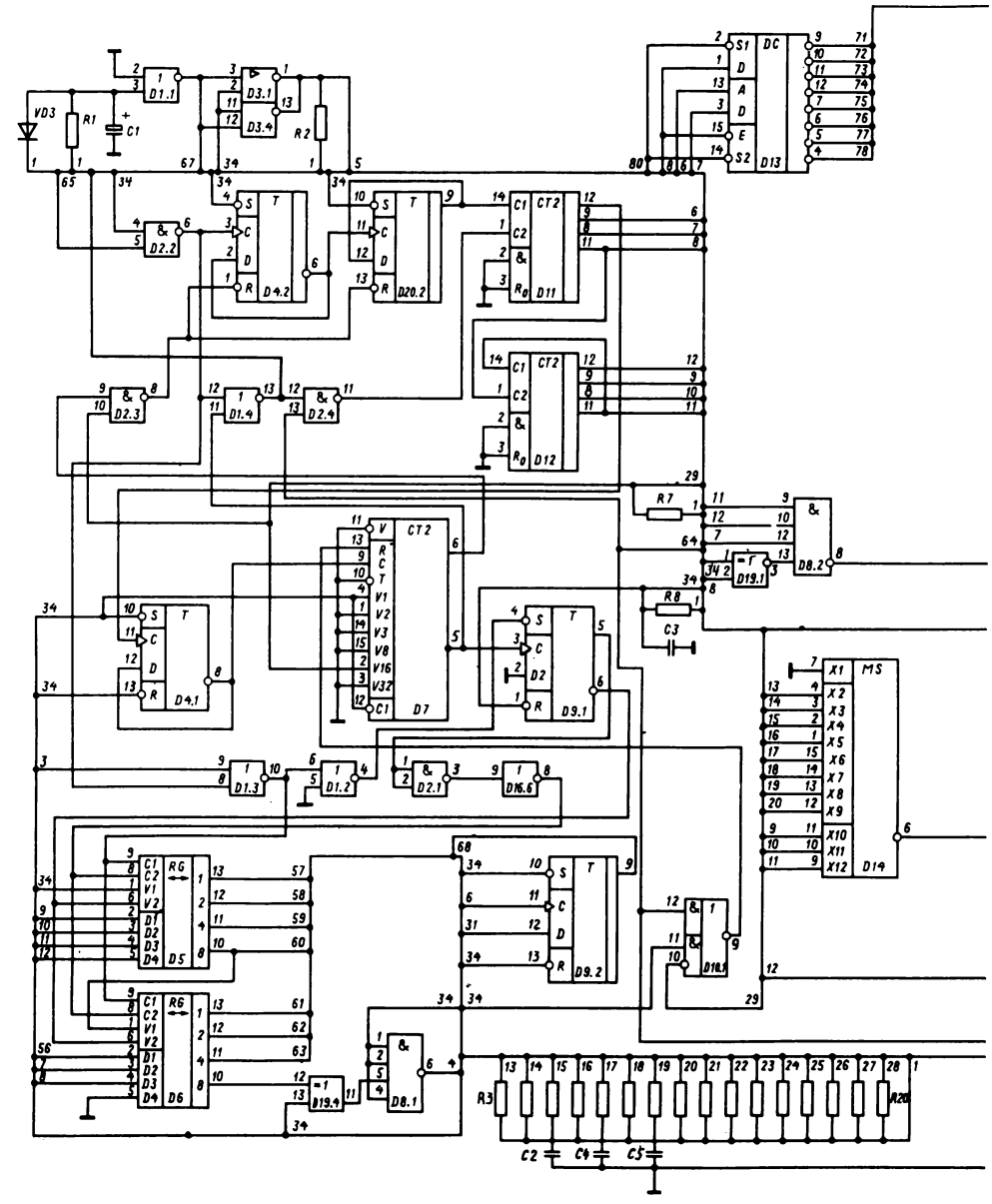
установленных в узлах матрицы размером 16 × 8. Когда сигнал логического нуля "обежит" все восемь выходов дешифратора, переключается счетчик ***D12***. Выходами счетчика управляет коммутатор, к входам которого подключены контакты клавиш.

При нажатии одной из клавиш сигнал логического нуля проходит с выхода дешифратора ***D13*** на вход одного из мультиплексоров ***D14*** или ***D15*** и через ***D10*** на вход ***D*** триггера ***D9***, который блокирует прохождение тактовых импульсов на двоичные счетчики ***D11*** и ***D12***. Таким образом, на выходах счетчика будут зафиксированы коды столбца и строки нажатой клавиши. Чтобы исключить влияние дребезга контактов клавиш на работу схемы, в схему введен счетчик ***D7***, который формирует время задержки 40 мс до начала очередного сканирования при отпускании клавиши.

При нажатии клавиши сбрасывается счетчик ***D7*** (схемы защиты от вибрации), на выходе триггера ***D9*** формируется строб, по которому код клавиши, зафиксированный на счетчиках ***D11*** и ***D12***, параллельно вписывается в восьмиразрядный регистр схемы последовательного канала, реализованной на регистрах ***D5*** и ***D6***, откуда при разрешающем сигнале "Ответ" подается в последовательном коде на шину данных. При отпускании клавиши новое сканирование не начнется до тех пор, пока счетчик ***D7*** не досчитает до конца (пока не кончится вибрация контактов, сигнал с ***D10*** всякий раз будет сбрасывать счетчик ***D7*** на начало отсчета).

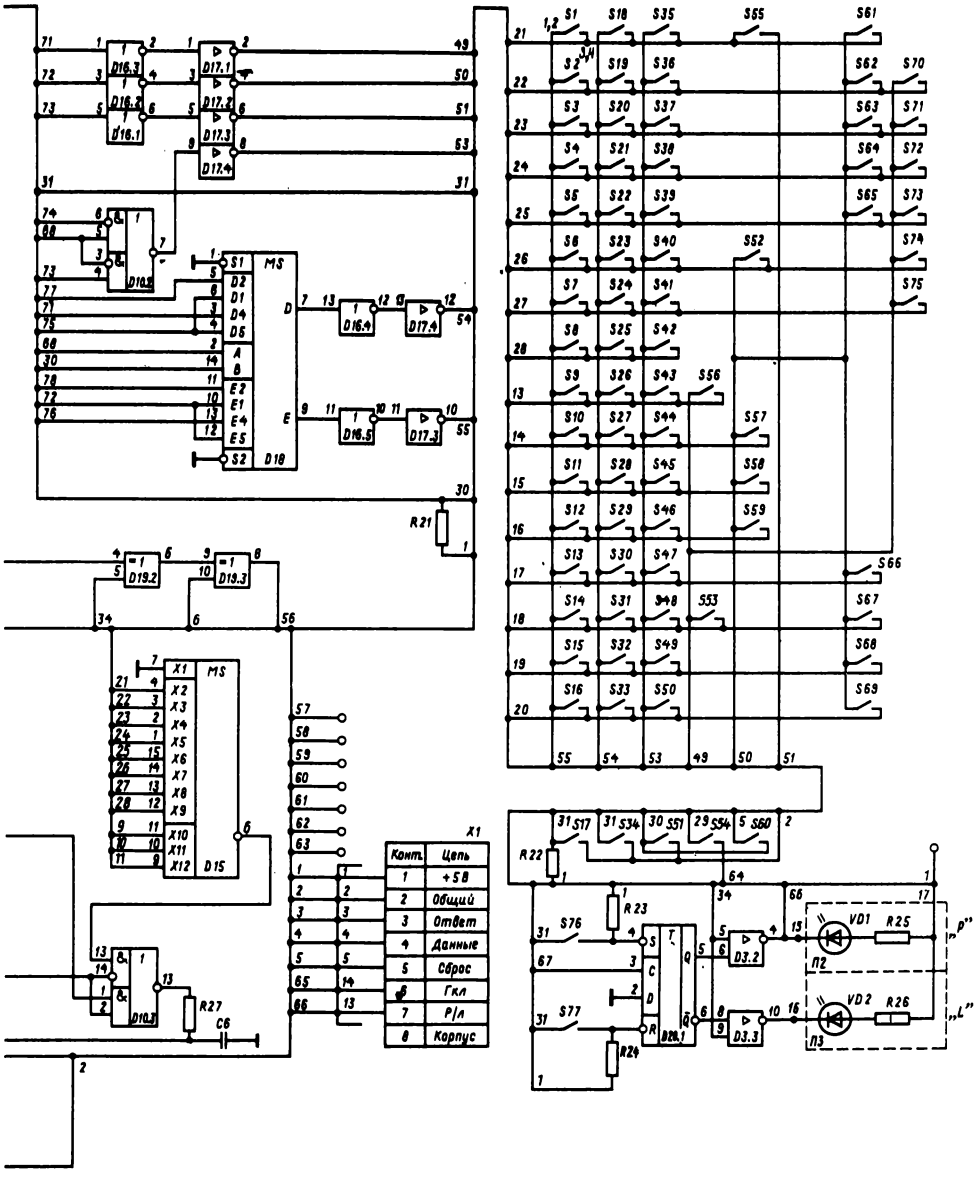
Схема автостарта удерживает сигнал "Сброс" в состоянии логического нуля после включения питания до тех пор, пока все питающие напряжения и синхроимпульсы не будут стабилизированы (пока не зарядится конденсатор ***C1***). Связанная с этой схемой клавиша СБР защищена от случайного нажатия, т.е. сигнал "Сброс" можно получить лишь одновременно нажав клавиши УПР и СБР.

В блоке клавиатуры предусмотрена световая индикация режимов РУС и ЛАТ. Переход из одного режима в другой можно осуществить путем одновременного нажатия клавиш РЕГ и РУС либо РЕГ и ЛАТ, что вызовет соответствующее изменение состояния триггера ***D20*** и зажигание соответствующего светоуказателя. С выхода ***D20*** код, соответствующий режимам "РУС" или "ЛАТ" поступает на линию ***Р/Л***.



S1	S2	S3	S4	S5	S6	S7	S8	S9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
П	Я	Р	С	У	У	Ж	В	Ь	Ы	З	Ш	Э	Щ	Ч		РЕГ	Ю	А	Б	Ц	Д	Е	Ф	Г	Х	И	Й	К	Л	М	Н	О		РЕГ	8	1	2	3	4
P	Q	R	S	U	u	V	W	x	Y	Z	[	\	]	^	ъ		D	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O			!	//	#	¤

Рис. 9.2. Принципиальная схема блока клавиатуры



40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77
5	6	7	8	9	:	;	<	=	>	?	ИНР	→	⊏	ПВТ		←	↑	↓	РЕД	СБР	1	2	3	4	5	6	7	8	9	0	·	=	F1	F2	F3		РУС	LAT
%	&	'	(	)	*	+	,	−	.	/																												

9.2. НАКОПИТЕЛЬ НА ГИБКОМ МАГНИТНОМ ДИСКЕ

В состав компьютера "Агат" входит НГМД ЕС 5088.01 или ЕС 5088.02. Оба накопителя созданы на базе серийного ЕС 5088. Кроме ПЭВМ "Агат", ими оснащаются и другие персональные компьютеры. В накопителе применяются пятидюймовые ГМД.

НГМД состоит (рис. 9.3) из корпуса *16*, платы "Логика" *1*, передней лицевой панели *9* с замыкающим устройством *8*; рамы *3* с прижимной шайбой *4*; чашки *5* шпинделя; суппорта *13* с головкой записи-чтения *11* и несущего устройства *2* для загрузки головки; электродвигателя постоянного тока *18* для привода ГМД *6*; шагового электродвигателя *15* для позиционирования головки записи-чтения *11*; микропереключателя *10* для установления режима "Защита записи"; микропереключателя *7* для предварительного развертывания шпинделя; головки записи-чтения *17*.

Питание - от двух источников постоянного тока:

1) напряжением +12 ±0,6 В; номинальная сила тока 0,9 А, максимальная - 1,8 А;

2) напряжением +5 ±0,25 В; номинальная сила тока 0,25 А, максимальная - 0,90 А.

Электронная часть НМГМД расположена на двух печатных платах "Регулятор" (РЕГЛ) *17* и "Логика" (ЛОГК). На плате РЕГЛ размещена схема регулирования частоты вращения вала электродвигателя постоянного тока в определенном диапазоне.

На плате ЛОГК находятся схемы:

управления позиционированием головки записи-считывания по сигналам персонального компьютера;

формирования сигнала "Защита записи";

генерирования сигналов записи и туннельного стирания, подаваемых головке записи-чтения;

чтения и воспроизведения считываемых сигналов и подачи этих сигналов ПЭИМ;

первоначального развертывания.

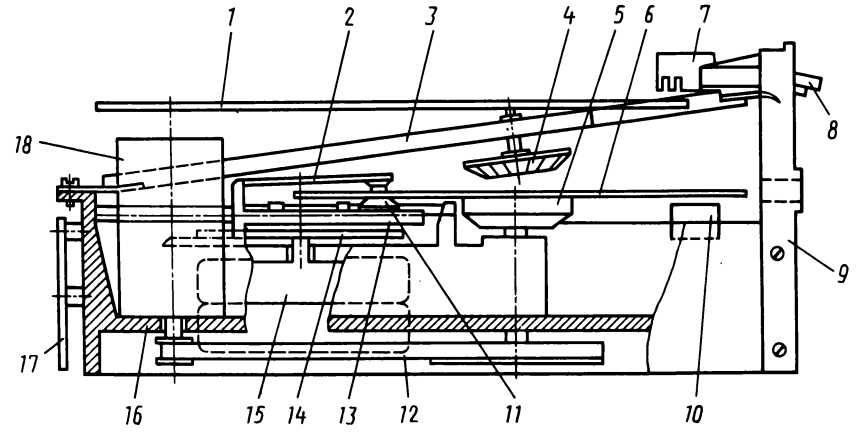


Рис. 9.3. НГМД ЕС5088.02(01)

Накопитель осуществляет три основные функции - поиск, запись и воспроизведение. Для их выполнения следует сообщить дискете необходимую скорость, позиционировать головку записи-чтения на предварительно определенную дорожку и осуществить операцию записи или чтения (рис. 9.4).

С помощью кабеля накопитель соединяется с ПЭВМ. По этому кабелю передаются интерфейсные сигналы и питающие напряжения (рис.9.5).

Двигатель *12* (см. рис. 9.3) питается напряжением +12 В постоянного тока и посредством ременной передачи вращает шпиндель с частотой 300 об/мин.

ГМД *6* центрируется и прижимается к чашке *5* шпинделя посредством прижимной шайбы *4*, расположенной на несущей раме *3*. Рама поджимается и опускается с помощью переключателя *7*, расположенного на лицевой панели накопителя.

Механизм позиционирования головки записи чтения. Головка чтениязаписи позиционируется на требуемую дорожку с помощью шагового электродвигателя *15* (см. рис. 9.3) и шайбы *14* со спиральным желобом. Шаговый двигатель вращает шайбу в направлении по часовой стрелке или против нее в зависимости от поступающих из ПЭВМ сигналов, причем для перемещения головки на соседнюю дорожку вал двигателя совершает два шага.

Головка чтения-записи накопителя НГМД ЕС 5088.02 (ЕС 5088.01) находится в непосредственном контакте с носителем информации ГМД *6*. Стеклокерамическая поверхность головки сконструирована таким образом, чтобы обеспечить расположение максимального числа байтов информации на магнитный носитель при минимальном изнашивании поверхности головки и ГМД. Для осуществления лучшего контакта с головкой во время обмена информацией диск дополнительно прижимается к головке (при отсутствии записи зазор составляет 1 - 2 мм).

Выбор накопителя. Выбор накопителя производится подачей высокого уровня напряжения по интерфейсной линии выбора. Это приводит к подключению напря-

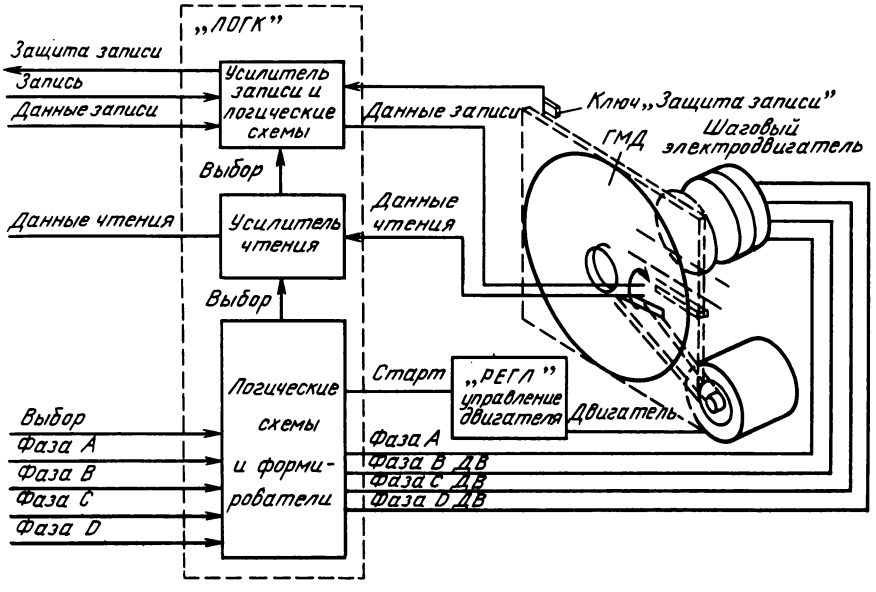


Рис. 9.4. Функциональная схема НГМД

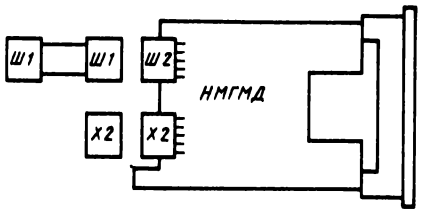


Рис. 9.5. Схема подключения НГМД ЕС5088.02 к контроллеру

Входные и выходные линии			
Контакт разъема Ш1	Наименование сигнала	Контакты разъема Ш2	Вид сигнала для НМГМД
А8	Выбор	В07(в07)	Входной
С7	Фаза А	В04(в10)	Входной
С6	Фаза В	В03(в11)	Входной
С5	Фаза С	В02(в12)	Входной
С4	Фаза D	В01(в13)	Входной
А9	Данные записи	В09(в05)	Входной
А4	Запись	В05(в09)	Входной
А2	Защита записи	В10(в04)	Выходной
А1	Данные чтения	В08(в06)	Выходной
С3	Напряжение +12В	А07(а07), А08 А09(а03), А10(а04)	–
С8	Напряжение +5В	А06 (а 08) В06 (в08)	–
А10	Напряжение 0В	А01(а13), А02(а12) В03(а11), А04(а10)	–

жения 12 В к электронным схемам, двигателям и индикатору на лицевой панели, кроме того, разрешается прохождение остальных интерфейсных сигналов.

Включение прямоточного электродвигателя. Включение двигателя, обеспечивающего вращение гибкого магнитного диска с необходимой частотой, осуществляется по сигналу "Старт". Чтобы обеспечить номинальную частоту вращения ГМД, необходимо подавать команды записи или чтения не раньше, чем через 1 с после включения электродвигателя постоянного тока.

Выключение двигателя осуществляется снятием напряжения с линии выбора накопителя.

Поиск необходимой дорожки. Позиционирование головки записи-чтения на необходимую дорожку осуществляется включением четырех фаз шагового двигателя в определенном порядке. Число и последовательность подаваемых импульсов определяется компьютером (рис. 9.6).

Операция ЗАПИСЬ (рис. 9.7). Порядок выполнения операции ЗАПИСЬ следующий:

1. Активируется линия ВЫБОР накопителя.
2. Головка записи-чтения устанавливается на определенную дорожку.
3. Активируется линия записи.
4. Выключатель защиты записи устанавливает на линии ЗАЩИТА ЗАПИСИ высокий уровень.
5. По линии ДАННЫЕ ЗАПИСИ подается информация, которую необходимо записать на ГМД.

Операция ЧТЕНИЕ (рис. 9.8). Порядок выполнения операции ЧТЕНИЕ следующий:

1. Активируется линия выбора накопителя.
2. Головка записи-чтения позиционируется на необходимую дорожку.
3. Линия записи переводится в высокий уровень.
4. По линии ДАННЫЕ ЧТЕНИЯ на ПЭВМ подаются считываемые данные.

Интерфейс (см. рис. 9.5). Интерфейс НМГМД обеспечивает передачу данных сигналов управления от ПЭВМ и к ней, а также устанавливает последовательность передачи сигналов управления, обеспечивающих выполнение операции ввода и вывода данных.

Рис. 9.6. Временная диаграмма сигналов при операции ПОИСК

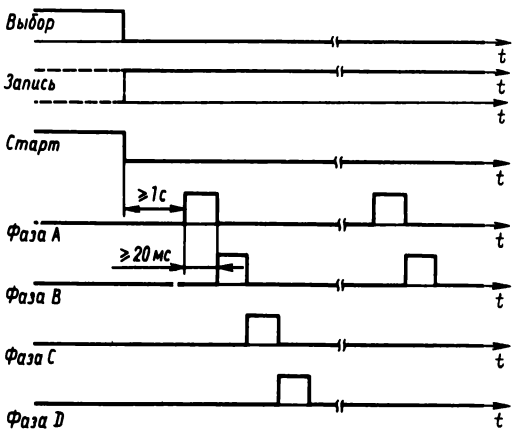
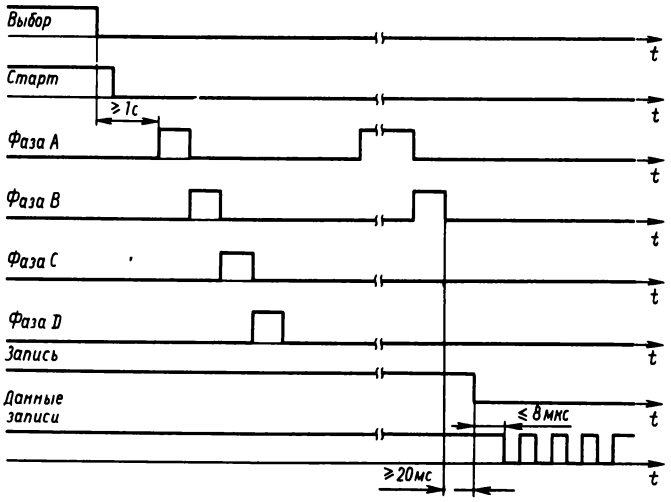


Рис. 9.7. Временная диаграмма сигналов при операции ЗАПИСЬ



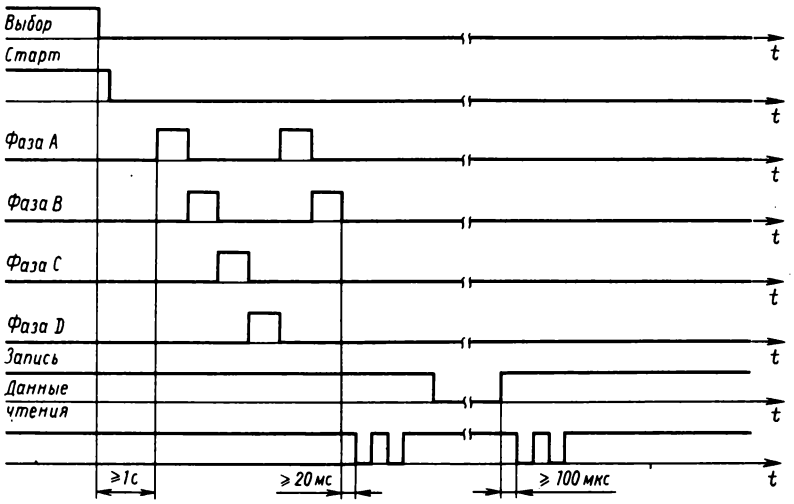


Рис. 9.8. Временная диаграмма сигналов при операции ЧТЕНИЕ

Низкий уровень напряжения на линии соответствует появлению сигнала, а высокий - его отсутствию.

Все управляющие линии - логические и задают сигнал к накопителю (входные) и к ПЭВМ (выходные). Линии передачи данных передают данные к накопителю (при записи) или к ПЭВМ (при чтении).

Питающий интерфейс обеспечивает необходимые напряжения питания для работы НМГМД от ПЭВМ.

Входные интерфейсные линии имеют следующие параметры:

Низкий уровень

$U^0_{вх}$, В ..... 0 - 0,4

$I_{вх}$, мА 40

Высокий уровень

$U^1_{вх}$, В ..... 2,4 - 5,25

$I_{вх}$, мА 250

Управляющие входные линии терминированы резисторами сопротивлением 1 кОм, расположенными на плате "Логика".

Выбор - линия выбора накопителя (рис. 9.9, *а*).

ЗАПИСЬ - линия определения вида производимой накопителем на ГМД операции ЗАПИСЬ или ЧТЕНИЕ (рис. 9.9, *а*). Низкий уровень этого сигнала определяет операцию запись, т.е. дает возможность записать на дискету данные, поступающие по линии ДАННЫЕ ЗАПИСИ. Высокий уровень определяет операцию ЧТЕНИЕ, т.е. осуществляет передачу информации из накопителя в ПЭВМ.

ЗАЩИТА ЗАПИСИ - интерфейсная линия, указывающая, можно ли записывать информацию на установленный ГМД.

ДАННЫЕ ЗАПИСИ - линия, по которой в накопитель поступает информация, которая будет записываться на ГМД. Разрешается активирование линии ЗАПИСЬ. Временные соотношения импульсов по ли- нии приведены на рис. 9.9, *б*.

ДАННЫЕ ЧТЕНИЯ - линия передачи данных из НМГМД в ПЭВМ. По этой линии передаются считанные и сформированные данные с ГМД (рис. 9.9, *в*).

Интерфейс питания представляет собой совокупность электрических линий, посредством которых накопитель подключается к системе питания и обеспечиваются необходимые напряжения (табл. 9.1).

Связь между ГМД и персональным компьютером осуществляется посредством разъема X01, на который подаются управляющие сигналы, данные записи и чтения.

Таблица 9.1

Параметры питания НГМД

Напряжение, В	Допустимое отклонение, В	Допустимые пульсации, МВ	Сила тока нагрузки, А	
			максимальная	номинальная
+5	±0,25	50	0,9	0,25
+12	±0,6	100	1,8	0,9

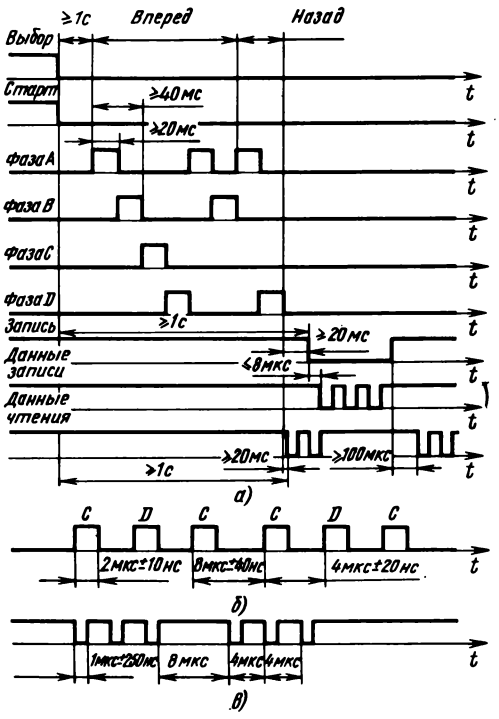


Рис. 9.9. Временные диаграммы:
а - обобщенная интерфейсных сигналов; б - сигнала "Данные чтения"; в - сигнала "Данные записи"

Плата "Логика". Принципиальная схема приведена на рис. 9.10.

Управление шаговым двигателем. При выбранном накопителе сигнал "Выбор" посредством инвертора ***D01.01*** насыщает транзистор VT6. Обеспечивается передача напряжения 12 В к шаговому двигателю и к тракту записи-чтения.

Включение каждой фазы осуществляется ПЭВМ, причем в накопителе сигналы проходят через буферные схемы ***T107***, управляющие транзисторами ***VT1 - VT4***, каждый из коллекторов которых соединен с одной фазовой обмоткой двигателя.

Формирование сигнала "Защита записи". Этот сигнал поступает от микропереключателя. Если прорезь на дискете (ГМД) не заклеена, после установления ее в накопитель микропереключатель не срабатывает и на разъем ***X02***, контакт ***B05*** (***X02***, ***B09***) подается низкий уровень, вследствие чего при выбранном накопителе не формируется сигнал "+Защита записи" и не блокируется сигнал "-Запись". На дискету можно записывать информацию.

Если в накопитель установить дискету (ГМД) с заклеенной прорезью, микропереключатель сработает, на вывод 02 схемы D02.01 поступит высокий уровень, который блокирует команду "-Запись", а на разъем ***X01***, контакт ***B04*** (***X01***, ***B10***) поступит сигнал "+Защита записи". С этой дискеты можно считывать записанную информацию, но невозможно произвести запись новой информации. Контакты разъема, указанные в скобках, относятся к варианту НГМД ЕС 5088.01.

Генерирование сигналов записи и туннельного стирания. Запись информации на дискету (ГМД) возможна только после выбора накопителя (сигнал "Выбор") при наличии сигнала "-Запись" и при неактивированном сигнале "+Ключ защита записи".

На вывод ***01*** элемента ***D01*** поступает высокий логический уровень сигнала "-Выбор", транзисторы ***VT6*** и ***VT5*** насыщаются и подается напряжение +12 В к тракту записи-чтения.

Сигнал "-Запись" через элемент ***D02.02*** отпирает последовательно соединенные транзисторы ***VT10***, ***VT11*** и ***VT7***. Через последний из них осуществляется питание конечной ступени записи и туннельного стирания. Выходом D02.02 разрешается переход сигнала "Данные записи" через ***D02.03***, установление триггера в состояние логи- ческой единицы, в результате чего транзистор ***VT12*** открывается и к головке записи-чтения подается логический нуль. Данные с выхода ИС ***D02.03*** формируются в противофазе на выходах буферных схем ***D03.05*** и

D03.06 и управляют конечными переключающими транзисторами *VT8* и *VT9*, которые соединены с головкой записи-чтения через диоды *VD9* и *VD10*.

Цепь туннельного стирания включается в работу с опозданием по отношению команды ЗАПИСЬ и осуществляется пропусканием постоянного тока через обмотку стирания головки, диод *VD7* и токоограничивающий резистор *R20*.

ФОРМИРОВАНИ СИГНАЛОВ "ДАННЫЕ" И ЗАПИСИ-ЧТЕНИЯ

При отсутствии сигнала "-Запись" на средний вывод обмотки записи-чтения головки подается напряжение 6 В от делителя, выполненного на резисторах *R24*

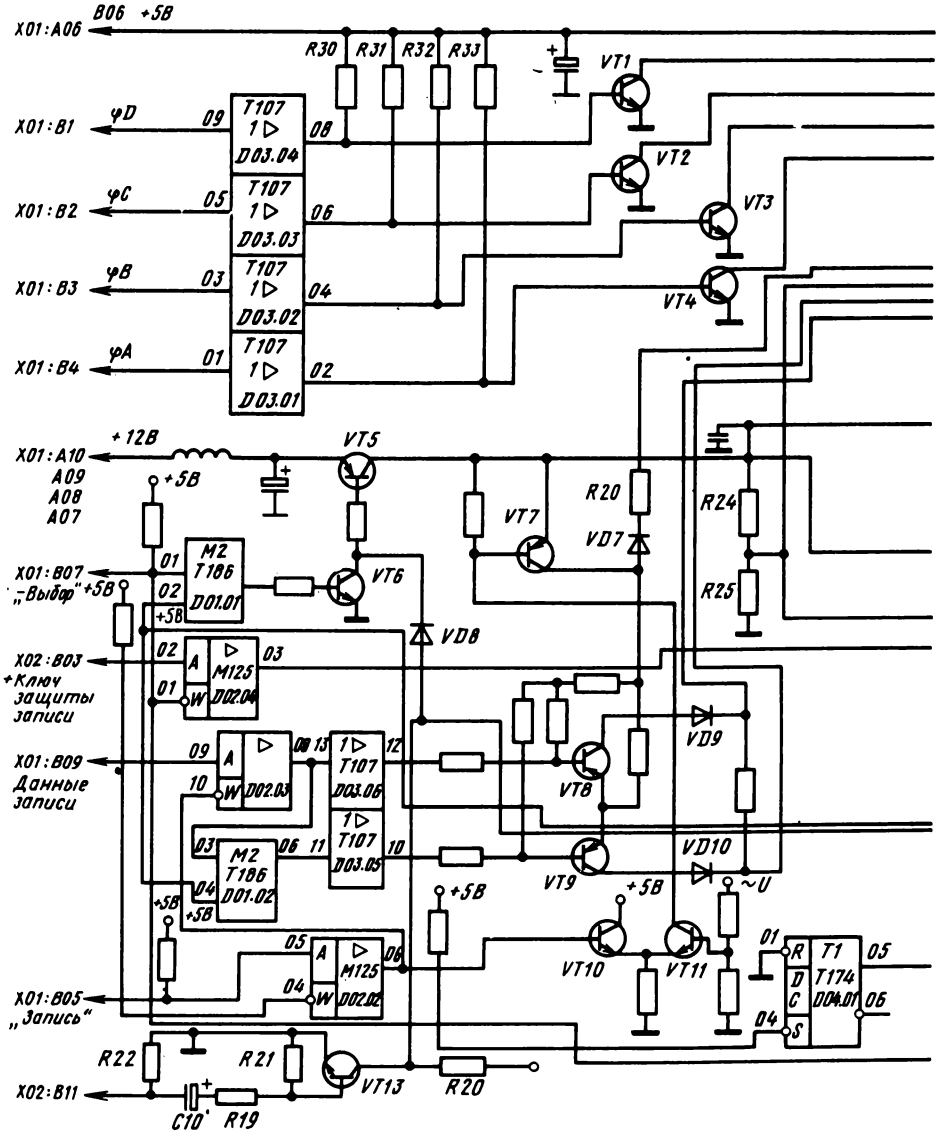
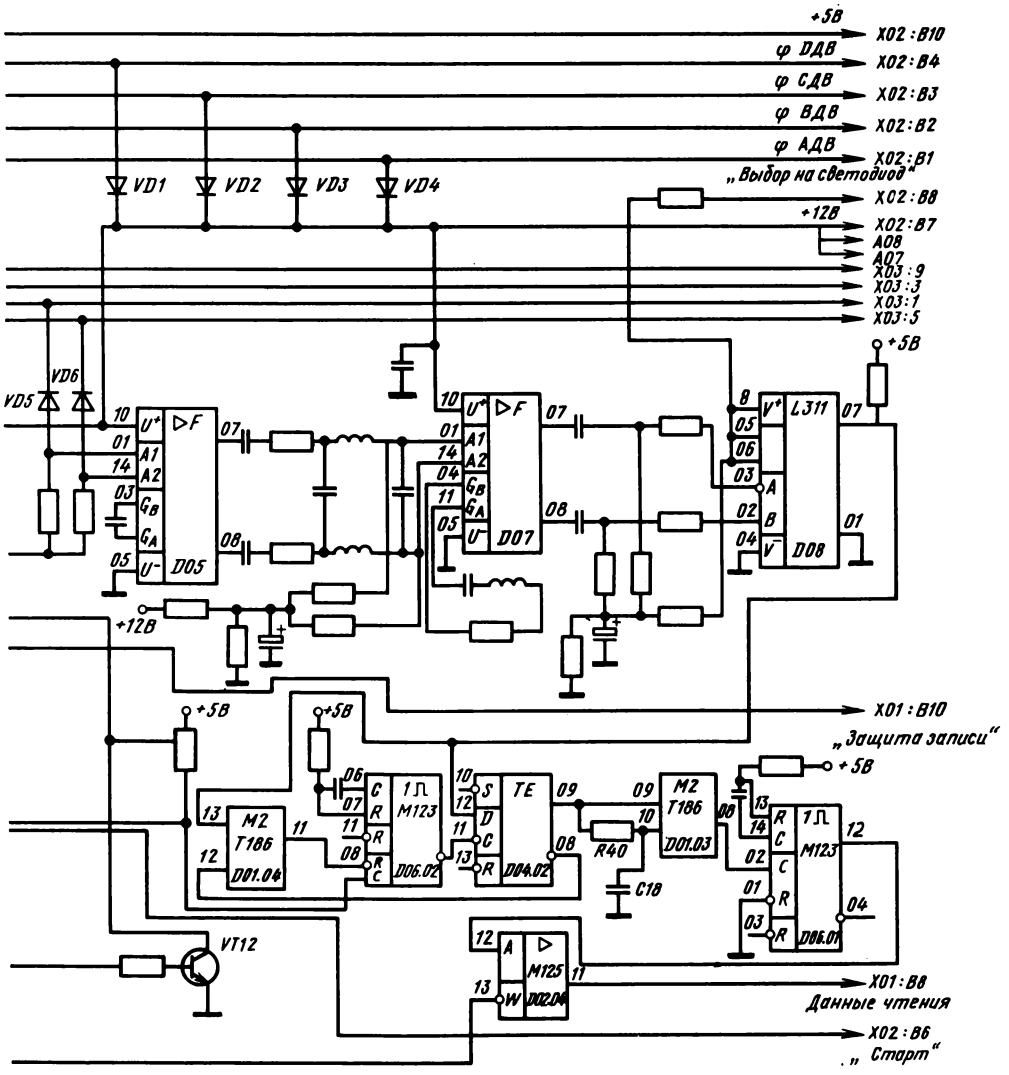


Рис. 9.10. Принципиальная схема платы ЛОГК (обозначение разъемов X01 и X02

и ***R25***. Диоды ***VD7 - VD10***, а также транзисторы ***VT7***, ***VT8***, ***VT9*** и ***VT12*** заперты. Индуцированный в обмотке записи-чтения сигнал через диоды ***VD5*** и ***VD6*** поступает на вход предусилителя, выполненного на интегральной схеме Р592 (***D05***), фильтруется низкочастотным ленточным фильтром с линейной фазочастотной характеристикой и подается на второй усилитель, выполненный на интегральной схеме Р592 (***D07***). Усиленный сигнал дифференцируется и подается на входы компаратора Р311 (***D08***), с выхода которого подается на формирующую группу (схемы ***D01.04*** и ***D06.02***; ***D04.02***), дифференцируется цепочками ***R40***, ***C18*** и ***D01.03*** и задерживается по времени (схема ***D06.01***). Выход данных разрешается сигналом "Выбор", формируемым элементом ***D02.04***.



соответствует разъему типа ELTRA (ПНР))

Схема первоначального развертывания. Предназначена для раскручивания шпинделя накопителя непосредственно после защелкивания защелки на лицевой панели. Действие схемы обеспечивает хорошую центровку гибкого магнитного диска. Непосредственно перед защелкиванием конденсатор ***C10*** разряжается через резисторы ***R19***, ***R21***, ***R22***, причем на выводы X02, B03 и X02, B11 не поступает сигнал от микропереключателя "Дверь закрыта", так как цепь разомкнута. Транзистор ***VT13*** заперт, и сигнал "-Старт" генерируется только от цепи сигнала "-Выбор".

Перед появлением сигнала "+Дверь закрыта" конденсатор заряжается, причем протекающий через резисторы ***R21*** и ***R22*** ток заряда отпирает транзистор ***VT13*** и формирует сигнал "-Старт" до момента зарядки конденсатора.

Плата "Регулятор" (РЕГЛ) (рис. 9.11). Регулирование частоты вращения производится после сравнения напряжения, пропорционального частоте вращения, с опорным напряжением. Напряжение тахогенератора формируется компаратором ***D03*** и включает ожидающий мультивибратор ***D01***. Этот выходной сигнал после формирования по амплитуде через резистор ***R08*** заряжает конденсатор ***C05***. На микросхеме ***D04*** выполнен сравнивающий интегратор, который управляет транзисторами ***VT1*** и ***VT2***. Опорное напряжение неинвертирующего входа D04 определяется делителем ***R03***, ***R14*** и ***VD3***. Постоянная времени интегратора задается параметрами R09 и C02. Выходное напряжение интегратора через ***VT1*** и ***VT2*** управляет двигателем.

Резистор ***R17*** служит ограничителем тока. Сигнал "Старт" управляет схемой через элемент ***D02.4***. Дроссель L01 и конденсатор C03 образуют фильтр помех.

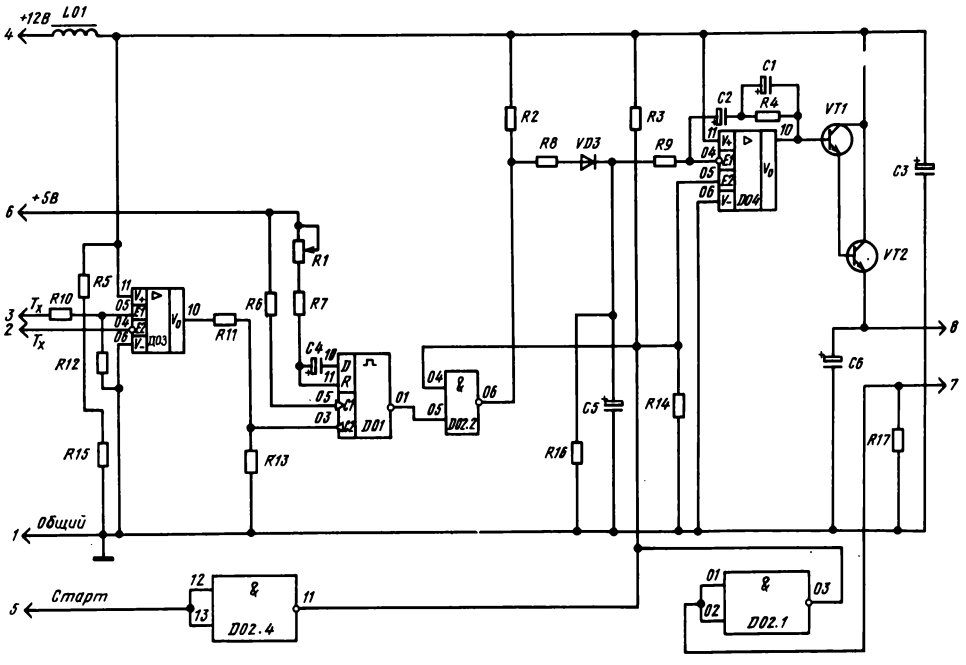


Рис. 9.11. Принципиальная схема платы РЕГЛ

9.3. ВИДЕОКОНТРОЛЬНОЕ УСТРОЙСТВО

ВКУ выполнено на базе унифицированного полупроводниково-интегрального модульного телевизионного приемника цветного изображения "Юность Ц-404" (УПИЦТ-32-10) с размером экрана по диагонали 32 см и углом отклонения электронных лучей 90°. Питание ВКУ осуществляется от сети переменного тока.

В состав ВКУ входят следующие устройства:

1. Блок разверток (БР), который содержит модуль кадровой развертки МЗ-2-IV; модуль синхронизации и задающего генератора строчной развертки МЗ-1-IV; модуль коррекции и гашения; плату блока разверток БР.
2. Блок питания БП, состоящий из платы преобразователя (ППР) модуля управления МУ-1, платы выпрямителя ПВ.
3. Плата панели кинескопа М6-1.
4. Плата позистора.
5. Блок обработки сигналов (БОС), содержащий плату блока обработки сигналов (платы БОС); модуль обработки сигналов цветности и опознавания (УМ2-1-1); три модуля М2-4-1 выходного видеоусилителя; модуль сопряжения; перемычку; панель управления (ПУ).

Рассмотрим работу перечисленных блоков ВКУ. При изучении модулей следует руководствоваться принципиальными схемами.

Блок разверток. Он вырабатывает: отклоняющие токи кадровой и строчной частоты; высокое напряжение 18 кВ и 3,5 кВ для питания второго анода кинескопа и цепи фокусировки; напряжение питания 600 В ускоряющих электродов и 150 В выходных видеоусилителей R, G, B; ряд импульсных напряжений, а также формирует в отклоняющих катушках токи центровки растра по вертикали и горизонтали.

Нормально работающий модуль синхронизации задающего генератора строчной развертки МЗ-1-IV (рис. 9.12) выделяет кадровый синхроимпульс из полного видеосигнала, поступающего на контакт *7* разъема 2х2, генерирует и автоматически подстраивает частоту и фазу строчной развертки и формирует на контакте *1* разъема 2х3 импульсы для управления работой буферного и выходного транзисторов строчной развертки.

Контроль исправности и ремонт блока разверток. Чтобы убедиться в исправной работе модуля, необходимо с помощью осциллографа С1-64 измерить амплитуду импульсов на контактах разъема 2х3 блока развертки. Исправную работу характеризуют следующие параметры:

амплитуда импульсных сигналов синхронизации на контакте *7* - не менее 2 В;

амплитуда выходного сигнала задающего генератора строчной развертки на контакте *1* составляет 2 - 3 В;

амплитуда кадрового синхронизирующего импульса на контакте *5* составляет 5 - 9 В.

Далее осциллографом проверяют на контакте *X2N* модуля длительность и амплитуду строчных синхроимпульсов. Длительность импульсов должна быть 5,1 ±0,1 мкс при амплитуде не менее 10 В.

Наличие синхронизации между импульсами модуля и управляющими импульсами, поступающими с системного блока ПЭВМ, проверяют также с помощью осциллографа, подключенного к контакту *1* разъема 2х3 блока развертки. Изменяя

сопротивление резистора ***R24*** регулировки частоты строчной развертки, убеждаются в том, что импульс на экране осциллографа неподвижен.

При устранении неисправностей в модуле необходимо руко водствоваться результатами измерений режимов работы транзисторов по постоянному току, приведенных в документации к телевизору "Юность Ц-404".

В состав модуля кадровой развертки МЗ-2-IV входят задающий генератор и бестрансформаторный выходной каскад, нагрузкой которого служат кадровые отклоняющие катушки (рис. 9.12).

Неисправности, возникающие в этом модуле, приводят к отсутствию развертки по вертикали (узкая горизонтальная полоса в центре), нарушению размера и нарушению линейности.

При ремонте модуля следует предельно убавить яркость, так как появление в

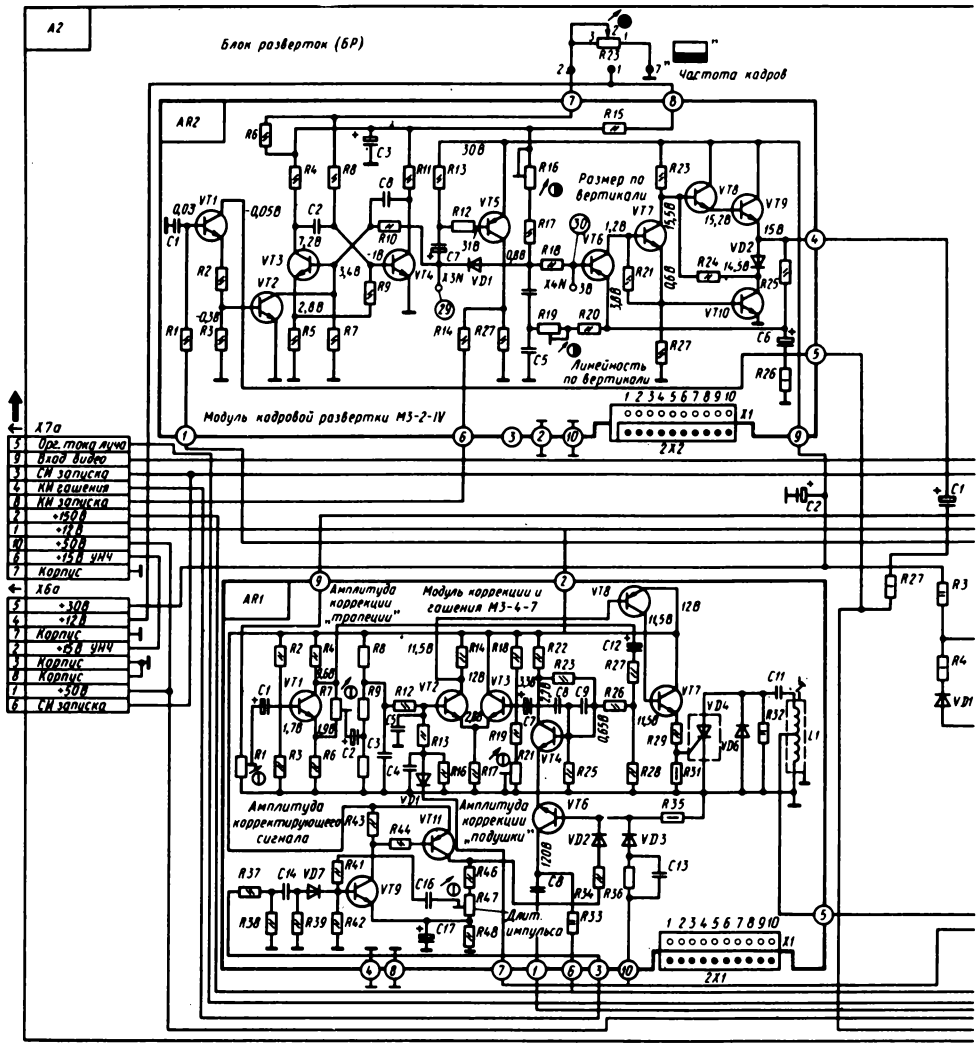


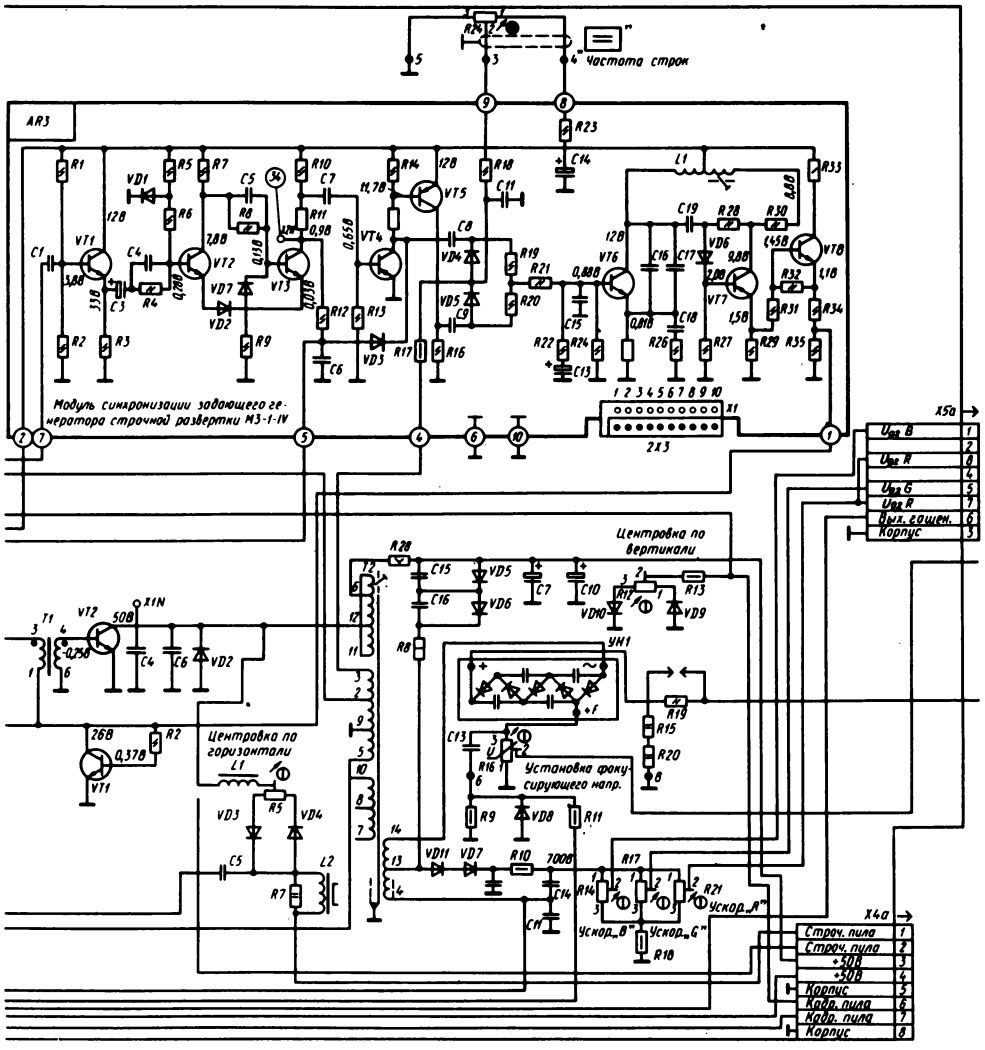
Рис. 9.12. Принципиальная схема блока разверток

центре экрана горизонтальной полосы может вызвать выгорание люминофора и деформацию теневой маски кинескопа.

Определение неисправностей начинают с изменения напряжения питания $30^{+0,5}_{-3,0}$ В на контакте *9* разъема 2х2 блока развертки, так как отклонение напряжения от номинального значения сказывается на размере и линейности изображения.

Исправно работающий модуль формирует на контакте *4* разъема 2х2 пилообразно-импульсное напряжение в соответствии с рис. 9.13. Амплитуда импульсов кадровой частоты, измеренная на контакте *X3*, должна быть не менее 11 В при длительности импульса 0,7 ± 0,1 мс.

Наличие синхронизации между импульсами модуля и управляющими импульсами, поступающими с системного блока ПЭВМ, проверяют с помощью осциллографа,



подключенного к контакту *X3N* модуля. Изменяя сопротивление резистора *R23* регулировки частоты кадровой развертки, убеждаются в том, что импульс на экране осциллографа неподвижен.

Наиболее частой причиной отказа модуля является выход из строя транзисторов *VT9* и *VT10*. При замене транзисторов следует обратить внимание на достаточную затяжку винтов, с помощью которых транзисторы крепятся к корпусу радиаторов.

Измерение режимов работы по постоянному току производят комбинированным прибором Ц-4353.

Модуль коррекции и гашения М3-4-7 осуществляет коррекцию геометрических искажений растра и гашение обратного кода луча по горизонтали.

Для устранения "подушкообразных" искажений растра по горизонтали (искривление вертикальных линий) в модуле предусмотрена модуляция отклоняющего тока строчной частоты параболическим током кадровой частоты. Формирование необходимого корректирующего сигнала осуществляется дифференциальным усилителем на транзисторах *VT2*, *VT3*, а модуляция отклоняющего тока - за счет изменения индуктивности контура *L1*.

Схема гашения выполнена на транзисторе *VT6*, с коллекторной нагрузки которого сигнал поступает на модулятор кинескопа.

Для оценки работоспособности модуля следует убедиться в том, что при изменении сопротивления переменных резисторов *R1*, *R7* и *R21* удается регулировать "подушкообразные" и "трапецеидальные" искажения растра. В противном случае необходимо проверить с помощью осциллографа С1-64 наличие входных и выходных сигналов на контактах разъема 2х2 блока разверток. Параметры сигналов должны быть следующими:

амплитуда кадровой пилы на контакте *9* - не менее 20 В;

амплитуда строчных импульсов на контакте *7* - не менее 65 В;

амплитуда кадровых импульсов на контакте *3* - не менее 8 В;

размах выходного сигнала коррекции параболической формы на контакте *5* - 40 - 85 В;

размах выходных импульсов гашения отрицательной полярности на контакте *1* - не менее 110 В.

Осциллограмма среза импульса на выходе тиристора *VT4* должна соответствовать рис. 9.14.

Измерение режимов работы по постоянному току выполняют комбинированным прибором Ц-4353.

На кросс-плате блока развертки расположены буферный (транзистор *VT1*) и

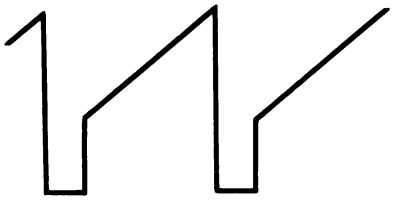


Рис. 9.13. Эпюры напряжений импульсов кадровой развертки

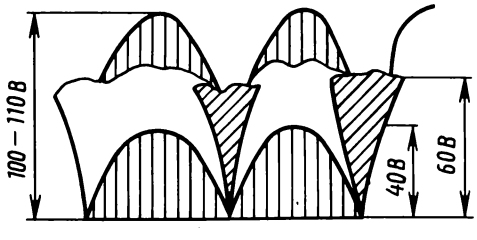


Рис. 9.14. Эпюры напряжений импульсов на выходе тиристора VT4 (см. рис. 9.12)

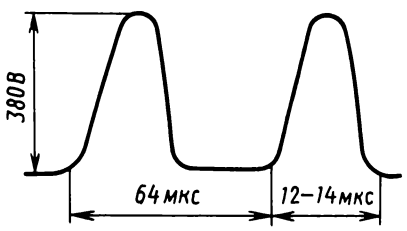


Рис. 9.15. Эпюры напряжений импульсов строчной развертки

выходной (транзистор *VT2*) усилительные каскады строчной развертки. Нагрузкой транзистора *VT2* является *ТВС*, с выхода которого снимаются импульсы высокого напряжения 6 кВ, поступающие на умножитель УН-8,5/25-1,2-А. С выхода умножителя "+" напряжение 18 кВ через резистор *R19* подается на второй анод кинескопа, а с выхода "+ F" - на фокусирующий электрод кинескопа.

Параллельно отклоняющей системе включена схема центровки растра по горизонтали. Она состоит из резисторов *R5*, *R6* и двух диодов *VD3*, *VD4*. С помощью резистора R5 можно изменять силу постоянного тока через отклоняющую систему, а от силы тока зависит перемещение изображения по горизонтали на кинескопе.

Признаком исправной работы выходного каскада строчной развертки является наличие всех выпрямленных напряжений, указанных на принципиальной схеме. Постоянное напряжение до 800 В измеряют комбинированным прибором Ц-4353.

Для контроля режима работы транзистора *VT2* необходимо подключить измерительный вход осциллографа к контакту *X1N* и убедиться в соответствии осциллограмме, представленной на рис. 9.15.

Наиболее вероятной причиной неработоспособности выходного каскада является выход из строя транзитора *VT2*, *ТВС* и умножителя напряжения.

БЛОК ПИТАНИЯ

Питание всех узлов ВКУ осуществляется стабилизированным бестрансформаторным блоком питания, работающим синхронно с частотой строчной развертки (15 625 Гц).

Для успешного отыскания неисправностей блока питания приведем краткие сведения о блоке.

Блок питания (рис. 9.16) представляет собой импульсный обратноходовой преобразователь, выполненный по однотактной схеме на силовом транзисторе *VT2*. На базу этого транзистора поступают импульсы положительной полярности с модуля управления *МУ-1*, усиленные буферным каскадом на транзисторе *VT1*.

Эти импульсы формируются микросхемой *D1* (К174ГФ1), содержащей автогенератор и предварительный усилитель. Для синхронизации на ее вывод *13* подаются импульсы строчной частоты.

Стабилизация выходных напряжений осуществляется за счет изменения времени открытого состояния силового транзистора *VT2*, которое определяется длительностью импульса в его базе, и зависит от режима работы микросхемы *D1*. Так, при увеличении напряжения сети происходит сужение импульса на выходе микросхемы, что приведет к снижению выпрямленных вторичных напряжений, т.е. постоянные напряжения останутся постоянными. Требуемые выходные напряжения устанавливаются с помощью подбора сопротивления резистора R1.

Пуск блока питания осуществляется в момент включения его в сеть за счет тока заряда конденсатора *C13* от выпрямленного напряжения сети через резистор

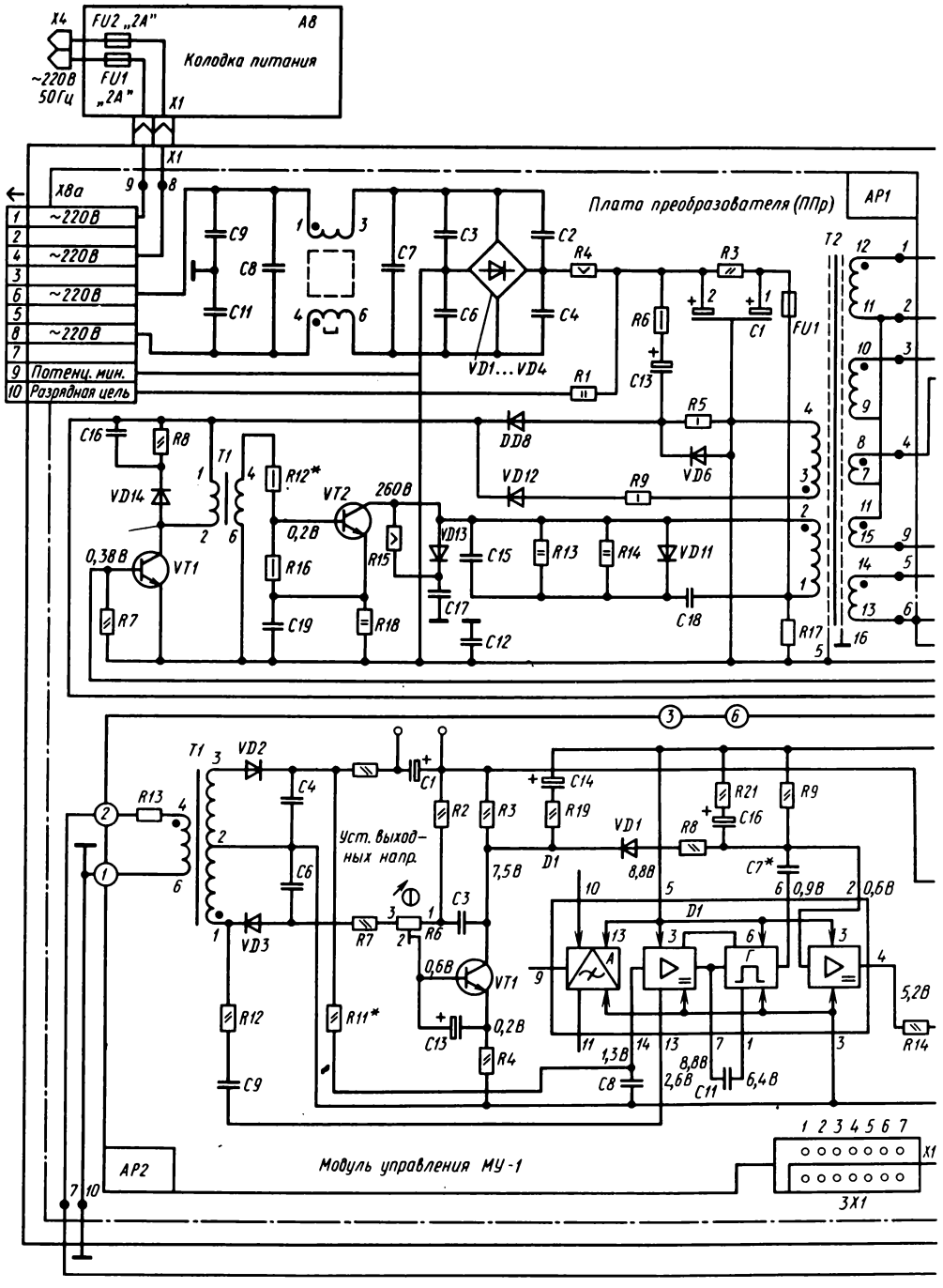
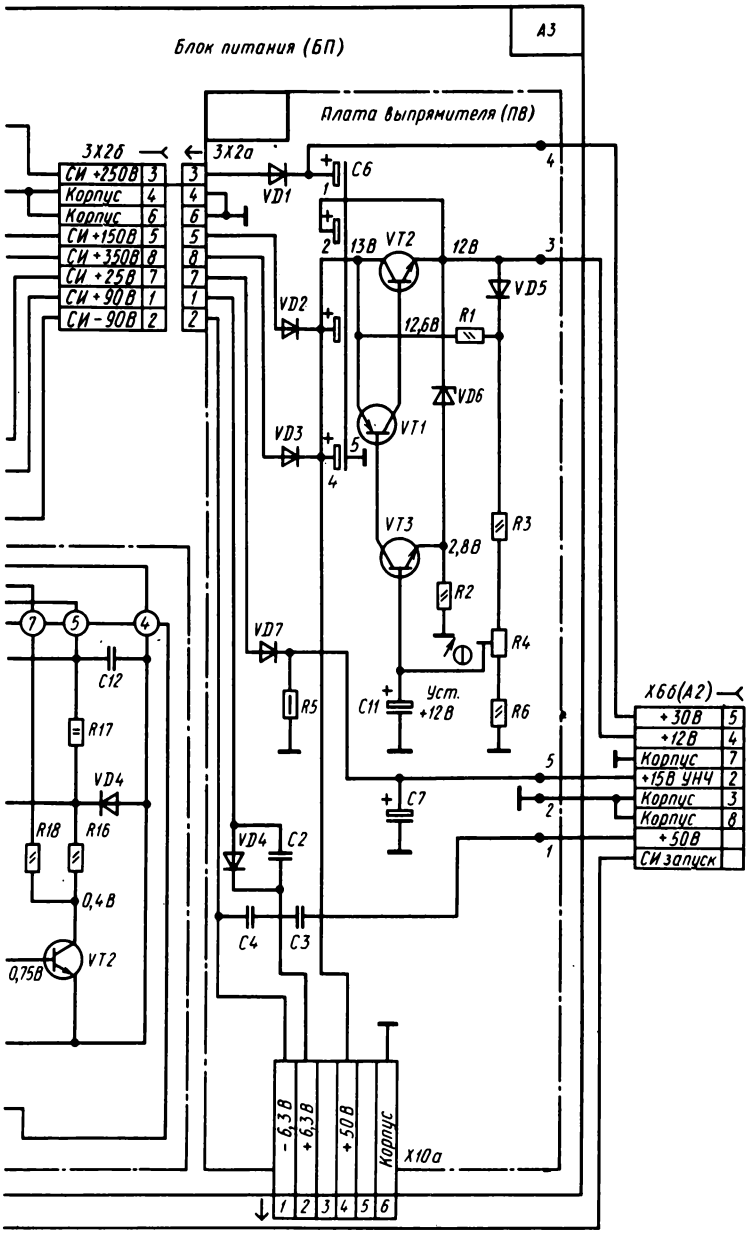


Рис. 9.16. Принципиальная схема блока питания ВКУ


Блок питания (БП)
А3
Плата выпрямителя (ПВ)
3Х2б
3Х2а
СИ +250В
Корпус
Корпус
СИ +150В
СИ +350В
СИ +25В
СИ +90В
СИ -90В
VD1
С6
VD2
VD3
13В
VT2
12В
VD5
12,6В
R1
VD6
VT1
VT3
2,8В
R3
R2
R4
VD7
R5
С11
Уст. +12В
R6
С12
R17
VD4
R18
R16
0,4В
0,75В
VT2
С7
VD4
С2
С4
С3
- 6,3В
+ 6,3В
+ 50В
Корпус
Х10а
Х66(А2)
+ 30В
+ 12В
Корпус
+15В УНЧ
Корпус
Корпус
+ 50В
СИ запуск

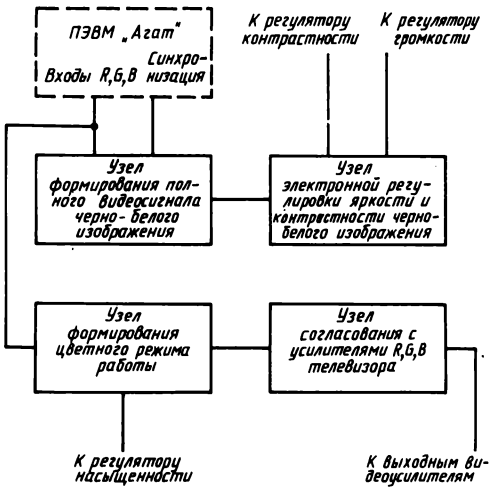


Рис. 9.17. Структурная схема модуля сопряжения

R6. При этом на конденсаторе *C1* образуется положительное напряжение, которое поступает на микросхему и буферный каскад. В дальнейшем, после пуска блока питания, питание микросхемы *D1* и буферного каскада осуществляется от дополнительной обмотки импульсного трансформатора *T2*.

Блок питания имеет защиту от коротких замыканий. При коротком замыкании напряжение на выводе *14* микросхемы *D1* отсутствует и она прекращает работать, что приводит к выключению блока питания.

Признаком исправной работы блока питания является наличие всех выпрямленных напряжений (см. п. 10.6).

БЛОК ОБРАБОТКИ СИГНАЛОВ

Блок преобразует импульсные сигналы, поступающие из системного блока ПЭВМ, в сигналы основных цветов для подачи их на катоды кинескопа.

Формирование полного видеосигнала из импульсов ТТЛ-логики дисплейного контроллера осуществляет модуль сопряжения. Этот модуль позволяет регулировать яркость и контрастность чернобелого изображения, а также насыщенность цветного изображения.

Модуль сопряжения функционально состоит из четырех узлов (рис. 9.17):

формирования полного видеосигнала черно-белого изображения;

электронной регулировки яркости и контрастности черно-белого изображения;

формирования цветного режима работы;

согласования с усилителями красного (R), зеленого (G) и синего (B) цвета.

В дальнейшем при изучении схемы работы модуля сопряжения следует руководствоваться принципиальной схемой, представленной на рис. 9.18.

Узел формирования полного видеосигнала формирует полный телевизионный сигнал, состоящий из сигналов изображения и синхронизации, с амплитудой, необходимой для нормальной работы микросхемы *D4*. Узел выполнен на элементах *D1.1*, *D2.1*, *D3.1*, *D3.2*, *R3*, *R4*, *R5*, *VD1*. Делитель, состоящий из резисторов *R4*, *R5*, определяет уровень черного, а резистор *R3* и диод *VD1* - уровень белого в сигнале изображения.

Сигналы кадровой и строчной синхронизации, объединенные в микросхеме *D2.1*, инвертируются микросхемой *D3.1*. Элемент *D3.1* (вывод 2) управляет работой диода VD1, формируя сигнал черно-белого изображения, когда сигналы *R*, *G*, *B* по длительности равны между собой.

Дроссели *L1*, *L2* и конденсаторы *C1*, *C2* образуют фильтр нижних частот, способствующий получению более устойчивой синхронизации в условиях воздействия помех.

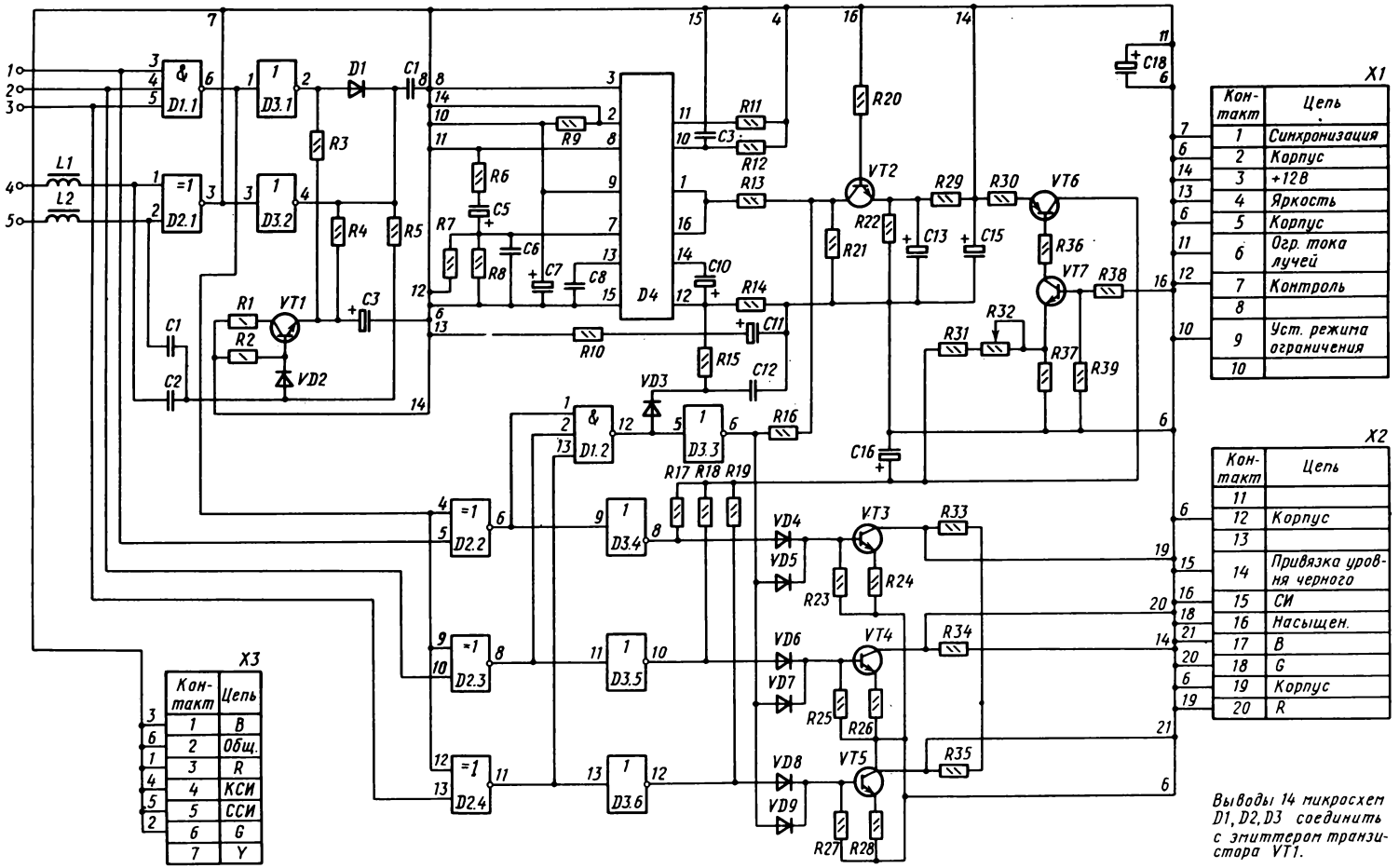


Рис. 9.18. Принципиальная схема модуля сопряжения

Узел электронной регулировки яркости и контрастности черно-белого изображения выполнен на микросхеме ***D4*** (К174УП1). В этом узле помимо указанных регулировок осуществляется привязка уровня черного изображения к фиксированному уровню и ограничение силы суммарного тока кинескопа ниже предельно допускаемого значения.

Коэффициент усиления, определяющий контрастность изображения, регулируется изменением постоянного напряжения на выводе ***7*** элемента ***D4***. Регулятором контрастности телевизора можно изменять размах сигнала черно-белого изображения не менее чем в 2,5 раза.

Информация о яркости изображения содержится в разнице между уровнем черного изображения и опорным уровнем. При этом информация не будет утрачена при потере постоянной составляющей сигнала. Привязка уровня черного изображения к некоторому фиксированному уровню осуществляется в микросхеме ***D4***. Для этого на вывод ***11*** микросхемы подаются отрицательные импульсы обратного хода строчной разертки, а на вывод ***10*** - эти же импульсы, продифференцированные цепочкой ***C9***, ***R12***. Внутри содержится активный элемент управления уровнем привязки. Максимальное значение уровня черного изображения определяется делителем ***R14***, ***R15***.

Опорный уровень создается с помощью транзистора ***VT2***, который открывается прямоугольным импульсом обратного хода строчной развертки до полного насыщения. При этом на делителе ***R29***, ***R22*** создается опорное напряжение 1,5 В, которое не зависит от формы передаваемого сигнала черно-белого изображения. С помощью микросхемы ***D4*** осуществляется также ограничение силы суммарного тока кинескопа ниже предельно допустимого значения. Для этого на вывод ***8*** из блока разверток телевизора поступает положительное постоянное напряжение, пропорциональное силе суммарного тока кинескопа. Это напряжение сравнивается внутри микросхемы с постоянным напряжением, подаваемым на вход ***9*** микросхемы.

Узел формирования цветного режима работы в зависимости от комбинаций сигналов ***R***, ***G***, ***B*** (выводы ***3***, ***4***, ***5*** микросхемы ***D1.1***) управляет работой блока согласования с усилителями ***R***, ***G***, ***B*** телевизора. В узле формирования цветного режима производится также регулировка насыщенности цветного изображения. Узел представляет собой коммутатор, выполненный на микросхемах ***D2.2***, ***D2.4***, ***D1.2***, ***D3.3***, ***D3.4***, ***D3.5***, ***D3.6***.

В зависимости от комбинации сигналов ***R G***, ***B*** на выводе ***8*** микросхемы ***D3.3*** вырабатывается признак цвета: 1 - черно-белое изображение; 0 - цветное изображение.

Если на выводе ***8*** микросхемы ***D3.3*** сигнал логической единицы, а на выводах ***8***, ***10***, ***12*** - сигнал логического нуля, то диоды ***VD5***, ***VD7***, ***VD9*** открыты, а диоды ***VD4***, ***VD6***, ***VD8*** закрыты. В этом случае на экране кинескопа наблюдается черно-белое изображение. Если хотя бы один из сигналов ***R***, ***G***, ***B*** не равен двум другим по длительности, вырабатывается признак цвета - 0. Тогда диоды ***VD5***, ***VD7***, ***VD9*** закрываются, а один или два из диодов ***VD4***, ***VD6***, ***VD8*** открываются. При этом на экране кинескопа наблюдается цветное изображение.

Цветная насыщенность изображения регулируется изменением постоянного напряжения на коллекторе транзистора ***VT6***. Транзистор ***VT7*** является управляющим. При уменьшении напряжения на его базе регулятором контрастности телевизора уменьшается проводимость участка коллектор - эмиттер транзистора

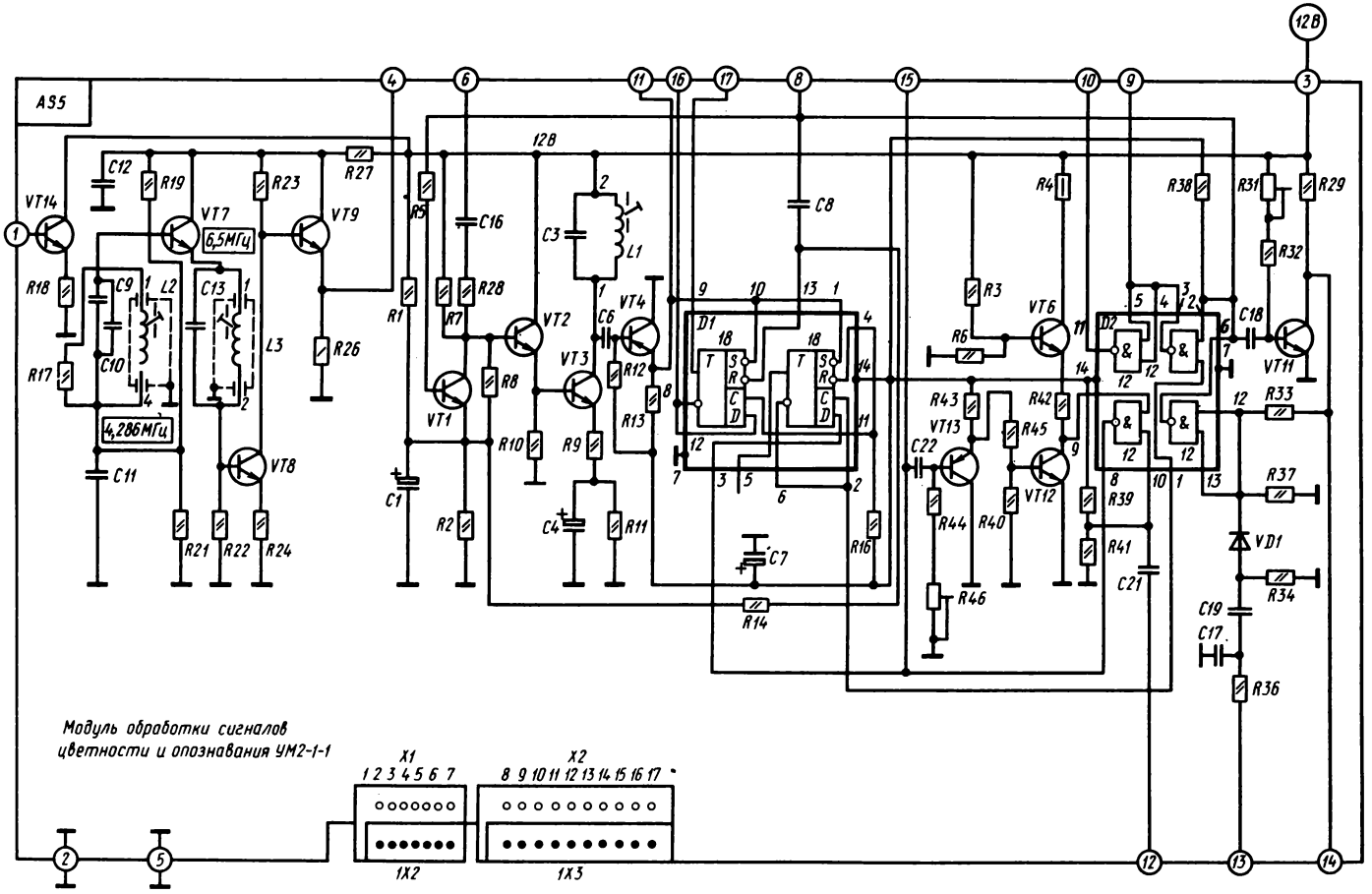


Рис. 9.19. Принципиальная схема УМ2-1-1

Рис. 9.20. Осциллограммы ВКУ

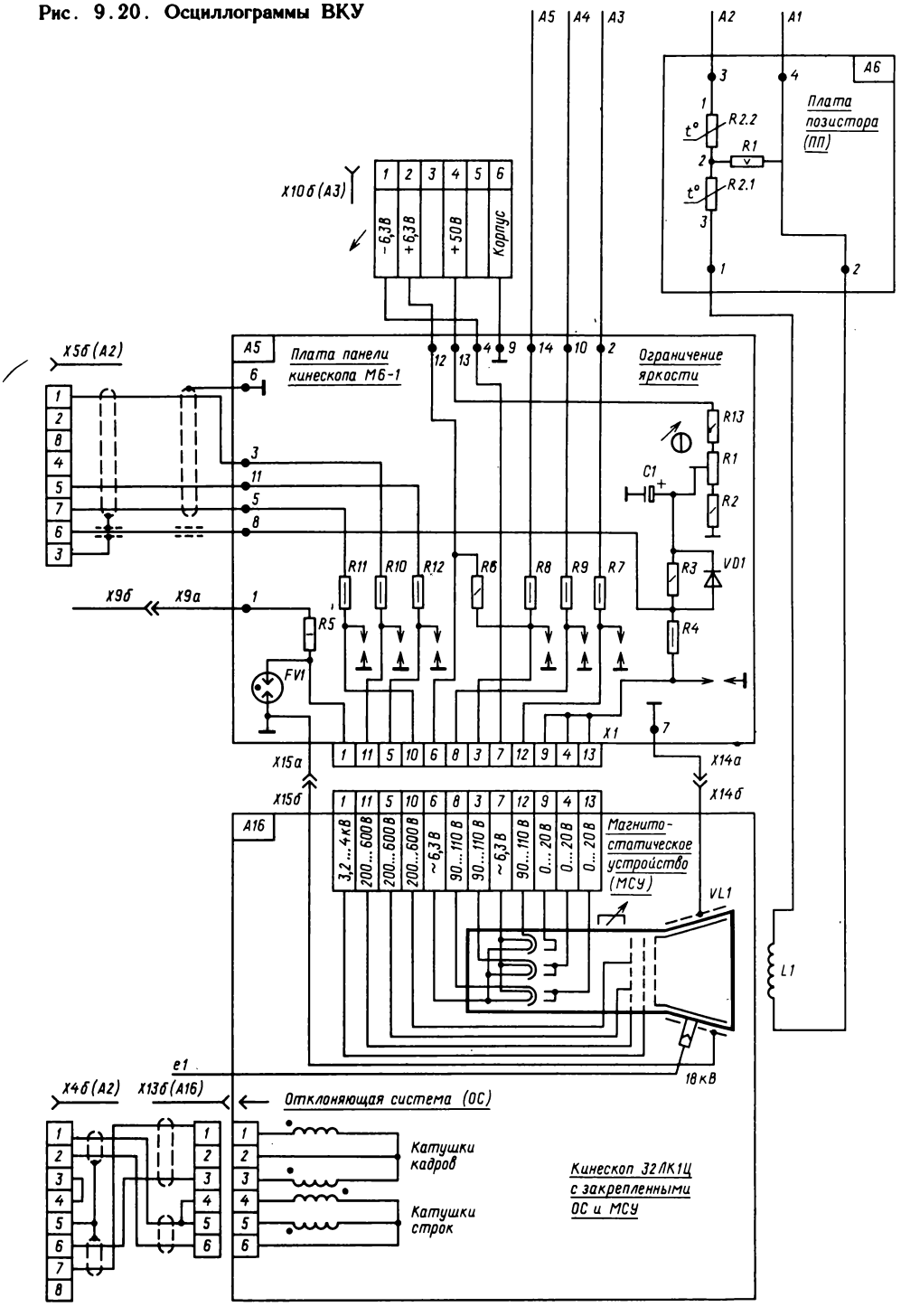
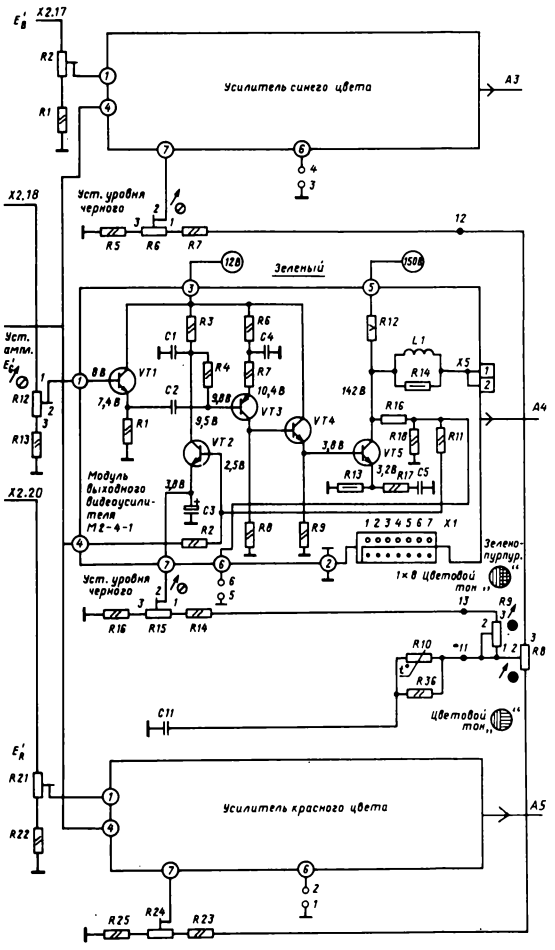


Рис. 9.21. **Принципиальная схема видеоусилителей**



VT7, а следовательно, и напряжение смещения на базе транзистора *VT6*. Это приводит к уменьшению напряжения на коллекторе транзистора *VT6*. Размах сигналов на выводах *8*, *10*, *12* микросхемы *D3* уменьшается, т.е. уменьшается цветовая насыщенность.

Подстроечный резистор *R32* определяет максимальное значение постоянного напряжения на коллекторе транзистора *VT6*.

Узел согласования с усилителями *R*, *G*, *B* телевизора поднимает уровень сигналов черно-белого или цветного изображения до уровня, необходимого для нормальной работы модулей выходных видеоусилителей (контакты *17*, *18*, *20* разъема *X2*). Узел выполнен на транзисторах *VT3*, *VT4*, *VT5*. Резисторы *R24*, *R26*, *R28*, включенные в эмиттеры соответствующих транзисторов, определяют размах сигналов черно-белого или цветного изображения (рис. 9.19).

Чтобы убедиться в исправной работе модуля сопряжения, необходимо подать от системного блока ПЭВМ "Агат" сигнал испытательного теста "Цветные полосы" и измерить с помощью осциллографа постоянные и импульсные напряжения на штырьках разъемов 1х14 и 1х15.

Амплитуда импульсных сигналов на контактах *17*, *18*, *20* разъема 1х15 при установке регулятора контрастности в крайнее правое положение должна быть не менее 1 В.

Постоянное напряжение на коллекторе транзистора *VT6* при установке регулятора насыщенности в крайнее правое положение должно быть не менее 6 В.

Оценка работоспособности модуля обработки сигналов цветности и опознания УМ2-1-1 (рис. 9.19) заключается в измерении постоянных напряжений на контактах разъемов 1х2 и 1х3 блока развертки и проверке с помощью осциллографа С1-64 амплитуды кадрового импульса на контакте *15* разъема 1х3. Амплитуда импульса должна быть не менее 4 В, а постоянные напряжения, измеренные комбинированным прибором Ц-4353, должны соответствовать напряжениям в соответствии с принципиальной схемой рис. 9.20.

Усиление сигналов *R*, *G*, *B* до требуемого размаха на катодах кинескопа (не

менее 40 В) осуществляется тремя идентичными схемами (рис. 9.21) для каждого из этих сигналов, конструктивно выполненных в виде одинаковых модулей (М2-4-1). Поэтому оценить работоспособность модулей можно путем их взаимной замены.

В случае обнаружения неисправного модуля, что может проявиться в отсутствии какого-либо основного цвета, следует произвести дефектацию его путем измерения режимов работы транзисторов модуля по постоянному току.

Г л а в а 10. РЕМОНТ ПЭВМ "АГАТ"

10.1. ТЕХНИЧЕСКОЕ ОБСЛУЖИВАНИЕ КОМПЬЮТЕРА

Техническое обслуживание заключается в профилактическом осмотре ПЭВМ и выявлении дефектов модулей и узлов в целях их последующего ремонта. Особенностью ЭВМ четвертого поколения, к которым относится компьютер, является наличие комплекта специальных программ - тестов. С помощью тестов гораздо проще выявить неисправные модули и узлы, локализовать неисправность до отдельных элементов.

Нужно отметить, что с ростом парка компьютеров увеличивается число и повышается качество тестовых программ. Обычно для удобства работы тесты располагаются на специально выделенной одной или нескольких дискетах. Рекомендуем при диагностике пользоваться "Полным тестом", приведенным в прил. 2.

10.2. ДИАГНОСТИКА ОПЕРАТИВНОЙ ПАМЯТИ

Неисправные микросхемы оперативной памяти выявляются с помощью теста оперативной памяти. При этом нужно руководствоваться сообщениями, выводимыми тестом на экран ВКУ.

Например, при тестировании ОЗУ на экране появилось сообщение:

МАРШ: БАНК *F*0 : 1000 = 00 (04)

Из этого сообщения можно сделать вывод, что в ячейку ОЗУ по адресу 1000 был записан ноль, а прочитано число 04. Это объясняется тем, что в разряде А2 постоянно присутствует логическая единица. Таким образом, можно сделать вывод, что элемент ***D25*** платы памяти и интерфейса, соответствующий этому разряду в ОЗУ1 (четные адреса), неисправен.

Обычно в микросхеме памяти выходит из строя вся строка или столбец. В таком случае на экране ВКУ появляются сообщения об ошибке по 128 адресам строк (младший байт адреса) или 128 адресам столбцов (старший байт адреса). Заметим, что ошибка может быть и иного характера, когда на выводе неисправной микросхемы постоянно присутствует не логическая единица, а логический нуль.

Аналогичным образом определяются неисправности микросхем на модуле ДОП и псевдоПЗУ.

10.3. ДИАГНОСТИКА ДИСПЛЕЙНОГО КОНТРОЛЛЕРА

Неполадки в дисплейном контроллере связаны с несправностями в видео генераторе и блоке развертки изображения. Вышедший из строя генератор синхроимпульсов (ГС) приводит к отсутствию растра на экране ВКУ. Потеря одного из импульсов кадровой (КСИ) или строчной (ССИ) развертки приводит к нарушению кадровой или строчной развертки.

Отсутствие цвета только в ГСР и ГНР сигнализирует о неисправности регистра графических режимов (РГР), отсутствие цвета только в алфавитно-цифровых режимах - неисправности РАЦР.

Для выявления неисправностей в видеогенераторе используется программа вывода цветных полос (см. гл. 6) или символов. Так, если при выводе цветных полос на экране появились черные или цветные точки (вкрапины), это говорит о том, что либо вышли из строя микросхемы ОЗУ, либо буферный регистр памяти (БРП).

При невозможности определить, что вызвало неисправную работу ПЭВМ - дисплейный контроллер или другой узел, необходимо провести диагностику контроллера по следующей методике:

1. На экране ВКУ получить изображение белого поля:

```
* 15 T ↓
* УПР Л ↓
* 2000 : F ␣N␣2001<2000.3FFFM␣N␣C713↓
```

2. Отсоединить кабель, идущий к ВКУ, от разъема RGB системного блока. На осциллографе С1-64 заземляющий контакт щупа подключить к контакту *2* разъема RGB компьютера. Установить длительность строчного синхроимпульса на контакте *5* разъема RGB (его длительность должна быть в пределах 3,0 ±0,1 мкс). Установить длительность развертки осциллографа 50 мкс на одно деление и измерить длительность кадрового синхроимпульса на контакте *4* разъема RGB (его длительность должна быть в пределах 256 ±10 мкс).

3. Установить длительность развертки осциллографа 20 мкс на одно деление и, последовательно подключая измерительный вход к контактам *3*, *6*, *1*, *7* разъема RGB, измерить уровни сигналов *R*, *G*, *B*, *Y*. Уровни всех сигналов (в том числе КСИ и ССИ) должны соответствовать уровню ТТЛ-логики (логический нуль меньше 0,4 В, логическая единица больше 2,4 В). Сигналы на выходах должны соответствовать уровню логической единицы.

В случае возникновения неисправностей обратитесь к п. 8.3.

10.4. ДИАГНОСТИКА ВСТРОЕННОГО ИНТЕРФЕЙСА

Проверка интерфейса пультов. Аналого-цифровые пульты подключаются к разъему "ПУЛЬТ". Каждый пульт имеет переменный резистор типа СП3-30а с максимальным сопротивлением 150 кОм и кнопочные замыкающие переключатели. Переменные резисторы в разъеме подключены к контактам *1*, *3* и *2*, кнопочные переключатели - к контактам *4*, *5* и *6* бытового разъема СГ7.

Для проверки работоспособности интерфейса пультов наберите на блоке клавиатуры в режиме диалога с БЕЙСИК программу, приведенную на рис. 10.1. На очищенном экране загорается одиночный символ "1".

Вращением рукоятки пультов достигается перемещение символа по горизонтали (пульт *1*) и по вертикали (пульт *2*). Нажатие кнопок пультов приводит к увеличению (пульт *1*) или уменьшению (пульт *2*) значения изображаемого на экране числа в пределах от 0 до 9. Если при вращении одной из рукояток символ на экране не перемещается, это свидетельствует о выходе из строя данного пульта. Если оба пульта не работают - неисправен мультиплексор входных данных (МВД) или дешифратор D7 платы памяти и интерфейса.

Проверка интерфейса видеосигнала. Это проверка в режиме диалога с интерпретатором языка БЕЙСИК. Предварительно на разъем "Видео" установить перемычку (СГ5), входящую в состав ПЭВМ "Агат". Затем с помощью БК набрать программу:

TEXT =15:HOME:CALL-151 ⌐

2000:FF␣N␣2001<2000.3FFFM␣N␣C713 ⌐

После нажатия на клавишу ⌐ на экране ВКУ наблюдается изображение белого поля. Подключить заземляющий вход осциллографа к контакту *2* перемычки, а измерительный вход - к контакту *3*. Установить длительность развертки осциллографа 20 мкс на одно деление и измерить амплитуду сигнала (рис. 10.2):

$T_{стр}$ (период строчной развертки), мкс 64

$T_{кадр}$ (период кадровой развертки), мкс 16969

$\tau_{стр}$ (длительность строчного синхроимпульса), мкс ... 3

$\tau_{кадр}$ (длительность кадрового синхроимпульса), мкс ... 256

Амплитуда импульсов от уровня черного изображения до уровня белого может принимать любое из восьми дискретных значений (восемь градаций яркости), определяемых комбинацией основных цветов в соответствии с кодами цвета.

Сопротивление выходной нагрузки составляет 75 Ом; емкость нагрузки - 100 пФ.

Магнитофонный выход. Магнитофонный выход проверяется комплексным тестом. В случае появления ошибки необходимо проверить работоспособность интерфейса

```
NEW
5   A=1 : HOME
10  HTAB PDL(0)/8
20  VTAB PDL(1)/8
30  IF PEEK(¤C061)<127
    THEN A=A+1
40  IF PEEK(¤C062)<127
    THEN A=A-1
50  IF A<0 THEN A=8
60  IF A>9 THEN A=0
80  PRINT A; GOTO 5
RUN
```

Рис. 10.1. Программа проверки пультов

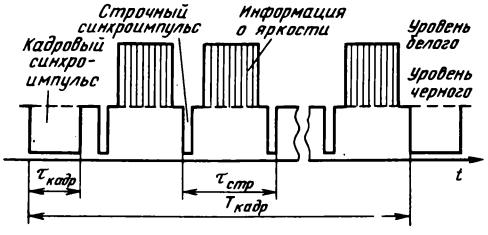


Рис. 10.2. Характеристика сигнала "ВИДЕОСИГНАЛ"

магнитофона. В режиме диалога с интерпретатором языка БЕЙСИК набрать на блоке клавиатуры программу:

```
]TEXT=15:HOME:CALL-151 ↵
*1000:A9 00 20 F0 FE A9 FF 20 F0 FE AD 00 C0 10
00 F1 60 ↵.
*1000 G ↵
```

Затем подключить измерительный вход осциллографа к контакту *3* разъема и измерить амплитуду импульсов. Амплитуда сигналов должна быть в диапазоне 0,25 - 0,50 В при частотном диапазоне от 0,5 до 10,0 кГц.

После окончания измерения нажать любую клавишу на блоке клавиатуры.

10.5. ДИАГНОСТИКА МОДУЛЯ ПАРАЛЛЕЛЬНОГО И ПОСЛЕДОВАТЕЛЬНОГО ИНТЕРФЕЙСА (ППИ)

При проверке и ремонте модуля ППИ следует руководствоваться описанием его работы, приведенной в гл. в 8 и в табл. 10.1. Порядок проверки ППИ следующий.

1. В режиме диалога с интерпретатором языка БЕЙСИК набрать на клавиатуре программу

```
TEXT=15:HOME:CALL-151 ↵
*C0D8:80 N C0D9:4E N C0D0:AA AA AA N
C0D9:5 ↵.
*1000:AD D9 C0 4A 90 FA A9 55 8D D8 CD
AD 00 C0 10 F0 8D 10 C0 60 ↵
*1000G ↵
```

Таблица 10.1

Назначение контактов разъема X2 модуля ППИ

Контакт	Цель	Контакт	Цель
A1	+5 В	B6	KB4
A2	KA0	B7	KB5
A3	KA1	B8	KB6
A4	KA2	B9	KB7
A5	KA3	B10	Вых НМ
A6	KA4	C1	+12 В
A7	KA5	C2	KC0
A8	KA6	C3	KC1
A9	KA7	C4	KC2
A10	Вх НМ	C5	KC3
B1		C6	KC4
B2	KB0	C7	KC5
B3	KB1	C8	KC6
B4	KB2	C9	KC7
B5	KB3	C10	Общий

2. Подключить осциллограф последовательно к контактам ***A2***, ***A9***, ***B2***,..., ***B10***, ***C2***,..., ***C9*** разъема ***X2*** модуля ППИ, установленного в разъеме ***X5*** системного интерфейса (окошко ***E6***), и измерить уровни логического нуля и логической единицы. Уровень измеряемых сигналов на контакте ***B10*** составляет: для нуля меньше -11,6 В, для единицы - больше +11,6 В, а на остальных контактах соответствует уровням ТТЛ-логики (логический нуль - меньше 0,4 В, логическая единица - больше 2,4 В).

Контакты ***A2*** - ***A***9, ***B2*** - ***B9***, ***C2*** - ***C9*** разъема - входы-выходы каналов ***A***, ***B***, ***C*** параллельного интерфейса. Контакт ***A10*** - вход последовательного канала. Контакт ***B10*** - выход последовательного канала.

Нагрузочная способность одной линии параллельного интерфейса составляет 1,6 мА по уровню логического нуля и 0,2 мА по уровню логической единицы. Нагрузочная способность последовательного канала по уровню логического нуля - не более 10 мА.

10.6. ДИАГНОСТИКА ВКУ

При поиске неисправностей ВКУ необходимо уточнить узел или блок, в котором оказалась данная неисправность. Перечень возможных неисправностей приведен ниже.

Внешние признаки неисправности	Блок, модуль, подлежащий проверке
Нет растра, светодиод не светится ...	Блок питания (БП), сетевые предохранители, сетевой шнур
На экране узкая горизонтальная полоса	Модуль кадровой развертки (МКР), отклоняющая система
Нарушение синхронизации по вертикали	Модуль строчной развертки (МСР), ячейка сопряжения
На изображении отсутствует один из первичных цветов	Мультиплексор выходной информации (МВИ), ячейка сосопряжения
Нет цветного, есть черно-белое изображение	Ячейка сопряжения
Искривление вертикальных и горизонтальных линий	Модуль коррекции и гашения

Если путем анализа внешних признаков не удается определить участок схемы, где произошла неисправность, необходимо измерить постоянные и импульсные напряжения на штырьках соединителей со стороны печати, руководствуясь описанием работы ВКУ п. 9.3.

Измеренные напряжения не должны отличаться от указанных на схеме более чем на ±15 %. Исключение составляет напряжение питания микросхем, допустимые отклонения которых не должны превышать 5 %. При отыскании неисправностей необходимо прежде всего проверить наличие всех выпрямленных напряжений блока питания.

Место измерений в блоке развертки	Измеряемое напряжение, В:
Контакт *1* разъема *X*6а	50 ±0,2
Контакт *5* разъема *X*6а	$30^{+0,5}_{-3,0}$
Контакт *4* разъема *X*6а	12 ±0,1
Контакты *12* и *13* разъема *X*5а	6,45 ±0,15

Одним из признаков наличия высокого напряжения на втором аноде является появление поверхностного заряда на экране кинескопа, выражающееся в легком потрескивании при включении ВКУ, или кратковременное появление линии или несфокусированного трехцветного пятна в центре экрана при выключении ВКУ. Если на штырьках разъемов постоянные и импульсные напряжения соответствуют указанным на принципиальной схеме, то неисправность следует искать в модулях.

Дальнейшее определение работоспособности модулей производить, руководствуясь описанием работы ВКУ, приведенным в гл. 9.

Для проверки работоспособности модуля сопряжения необходимо вывести на ВКУ цветные полосы и измерить осциллографом постоянные и импульсные напряжения на разъемах 1х14 и 1х15, руководствуясь описанием модуля сопряжения.

После проведения ремонта необходимо произвести настройку ВКУ. Подключить ВКУ к сети через разделительный трансформатор. Проверить наличие напряжения 12 ±0,1 В на контакте *3* разъема 1х14 (если напряжение отличается от номинального, то изменением сопротивления установить напряжение в заданных пределах); наличие напряжения 50 ±0,2 В на контакте *1* разъема X6A блока разверток и напряжения $30^{+0,5}_{-3,0}$ В на контакте *5* этого же разъема; наличие напряжения 6,45±0,15 В на плате панели кинескопа М6-1 на контактах *12* и *13* разъема X5A блока разверток. Осциллограммы имульсов разъема 1х15 блока обработки сигналов на контакте *14* должны соответствовать указанным на рис. 10.3,*а*, на контакте *15* - указанным на рис. 10.3,*б*.

Напряжение на эмиттере транзистора *VT1* модуля сопряжения должно находиться в пределах 5±0,15 В, напряжение на коллекторе *VT2* этого же модуля - в пределах 1,6±0,15 В. Подбором сопротивления резистора *R10* добиться, чтобы осциллограмма импульсов на коллекторе *VT2* соответствовала указанным на рис. 10.3,*в*. При этом экран кинескопа (регулятор яркости в среднем положении) должен иметь белое или серое свечение.

При необходимости отрегулировать баланс "белого" следует установить регулятор контрастности в крайнее левое положение против часовой стрелки и добиться подбором *R7* едва заметного изображения на экране кинескопа. Установить регулятор насыщенности ВКУ в крайнее правое положение и регулировкой резистора *R32* добиться, чтобы ярко-

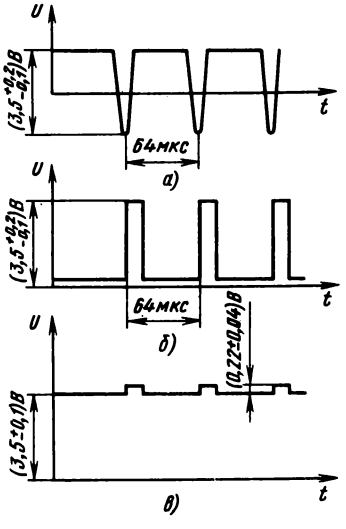


Рис. 10.3. Параметры импульсных сигналов

сти изображения цветных полос и белой полосы были равны. Заполнить экран изображением буквы Г. Размер изображения символов по вертикали в интервале 155±5 мм при минимальной яркости зависит от сопротивления резистора *R18* модуля КР. Размер изображения символов по горизонтали в интервале 215±5 мм при минимальной яркости устанавливается вращением сердечников РЛС2 на блоке развертки.

После установки ручки ЧАСТОТА СТРОК в среднее положение провести центровку изображения по горизонтали резистором *R5* платы блока развертки. Расстояние по горизонтали между левым краем растра и символом должно составлять 12 - 15 мм. Центровка изображения по вертикали выполняется с помощью резистора *R12*, установленного на плате блока развертки. Расстояние по вертикали между верхним краем растра и символом должно составлять 13 - 16 мм.

Запустить испытательный тест "Сетчатое поле". Если нелинейные искажения растра больше допустимых по горизонтали на ±12%, а по вертикали на ±10 %, то изменением сопротивления резистора *R20* на модуле кадровой развертки добиться нужных допусков нелинейных искажений по вертикали, а вращением магнита РЛС2 на плате блока развертки - по горизонтали.

Если геометрические искажения больше нормы, то необходимо устранить искривления вертикальных линий регулировкой в модуле М3-4-7. Для этого *R21* установить в среднее положение. Резистором *R*1 добиться минимальных искажений вертикальных линий в верхней и нижней части растра. Резистором *R7* установить минимальные искажения вертикальных линий типа "трапеция". Фокусировка осуществляется резистором *R16*.

10.7. ДИАГНОСТИКА НГМД

Для выявления неисправностей в накопителе необходимо использовать тесты РАЗМЕТКА, СКОРОСТЬ и ПОЗИЦИОНИРОВАНИЕ. Загружаются указанные тесты с исправного накопителя, который после загрузки отсоединяется от шины контроллера и к разъему подсоединяется дефектуемый НГМД.

Все тесты - самодокументируемые и работают в диалоговом режиме.

Тестом СКОРОСТЬ (SKOR) производится точная регулировка частоты вращения дискеты. После запуска теста СКОРОСТЬ в тестируемый дисковод необходимо вставить чистый ГМД и, вращая потенциометр, расположенный на плате "РЕГЛ", добиться, чтобы частота вращения была в пределах 200±0,45 мс. Информация о частоте вращения выводится на ВКУ. Если частота не регулируется в указанных пределах, а на экране ВКУ появляется случайный набор цифр, необходимо определить работоспособность платы "РЕГЛ". Для этого нужно НГМД извлечь из системного блока, вывести на экран ВКУ белое поле, запустить тест в работу и аккуратно приблизить накопитель стробоскопом к экрану ВКУ (стробоскоп - это диск с размещенными на нем черными полосками). Устойчивое положение (стоят или медленно двигаются) черных полос на диске стробоскопа говорит о нормальной работе платы "РЕГЛ". Если потенциометром не удается зафиксировать неподвижное положение черных полос, следовательно, плата "РЕГЛ" неисправна.

Определение работоспособности платы "ЛОГК" НГМД осуществляется тестами РАЗМЕТКА и ПОЗИЦИОНИРОВАНИЕ. Так, тест РАЗМЕТКА позволяет выявить неисправности в каналах чтения или записи. В дальнейшем при поиске неисправностей нужно руководствоваться описанием работы НГМД, приведенным в гл. 9.

10.8. НЕИСПРАВНОСТИ БЛОКА ПИТАНИЯ ПЭВМ

Определение работоспособности блока питания начинается с анализа внешнего проявления отказа. Характерными внешними проявлениями отказов блока являются:

отсутствие выходных напряжений;
отсутствие стабилизации выходных напряжений;
несоответствие выходных напряжений номинальным значениям;
завышенная амплитуда пульсаций выходных напряжений.

Перед тем как приступить к поиску неисправностей, необходимо снять крышку блока и разрядить конденсаторы *C5*, *C6* фильтра выпрямителя путем соединения

Таблица 10.1

Возможные неисправности блока питания

Наименование неисправностей	Вероятная причина	Способ устранения
При включении блока в сеть на выходе нет напряжения	1. Перегорел предохранитель	1. Заменить предохранитель
	2. Неисправны транзисторы VT1, VT4	2. Заменить транзисторы
	3. Неисправность в цепи схемы рассасывания	3. Проверить элементы VT2, VD7, VD9, R6; неисправные заменить
	4. Неисправны диоды VD1 - VD4	4. Проверить диоды; неисправные заменить
	5. Большая сила тока утечки конденсатора C9	5. Заменить C9
	6. Низкий коэффициент передачи тока транзистора VT4	6. Заменить VT4
	7. Неисправность в цепи схемы запуска	7. Проверить элементы C4, C13, VD5, VD11; неисправные заменить
	8. Неисправен трансформатор T1	8. Заменить T1
Слышен характерный рокот, потрескивание, напряжений на выходе нет	1. Короткое замыкание в цепи выходных выпрямителей	1. Проверить диоды VD14 - -VD17, конденсаторы C17 - C27; неисправные заменить
	2. Замыкание в нагрузке	2. Устранить причину замыкания
Блок выдает нестабилизированное напряжение	1. Неисправность в схеме управления	1. Проверить элементы VT3, VT5, VD13, R19, L2; неисправные заменить

Продолжение табл. 10.1

Нанменование неисправностей	Вероятная причина	Способ устранения
	2. Неисправен транзистор VT1, диод VD6	2. Заменить транзистор или диод
	3. Неисправен трансформатор T2	3. Заменить трансформатор
Завышенная амплитуда пульсаций выходных напряжений	1. Неисправность в цепи фильтров выходных выпрямителей	1. Проверить элементы L3, L5, C19 - C27; неисправные заменить
	2. Самовозбуждение блока (при этом слышен свист частотой 2 - 5 кГц)	2. Проверить элементы C14, VT5, C16, C27, C10, R22; неисправные заменить
	3. Пульсация в канале +5 В превышает норму	3. Проверить R1; если он неисправен, заменить
Несоответствие выходных напряжений номинальным значениям	1. Неисправен стабилитрон VD13	1. Заменить стабилитрон
	2. Несоответствие сопротивления резисторов R16 - - R19 номинальному значению	2. Заменить неисправный резистор

их корпусов с коллектором транзистора *VT4* проводом марки ПВ длиной 10 см. Проделать это 2 - 3 раза через 5 - 10 с.

Проверить установку электрорадиоэлементов, исключающую их соприкосновение между собой, и надежность механического крепления радиаторов транзистора *VT4*, диодов *VD15*, *VD16*.

Проверить наличие выходных напряжений в соответствии с табл.10.1. Выходные напряжения можно измерить, например, универсальным вольтметром В7-22А.

Перечень характерных неисправностей блока и методы их устранения приведены в табл. 10.1. При определении работоспособности блока необходимо руководствоваться осциллограммами, приведенными на рис. 8.40.

Г л а в а 11. ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ ПЭВМ "АГАТ"

11.1. ПОДКЛЮЧЕНИЕ ПЕЧАТАЮЩИХ УСТРОЙСТВ

ПЭВМ "Агат" может комплектоваться матричным печатающим устройством (МПУ) D-100 или СРА-80. В этом случае на плате ППИ необходимо наличие микросхемы ПЗУ с программой-драйвером печатающего устройства. Текст программы приведен в прил. 2. Драйвер можно загрузить (если нет ПЗУ) в оперативную память,

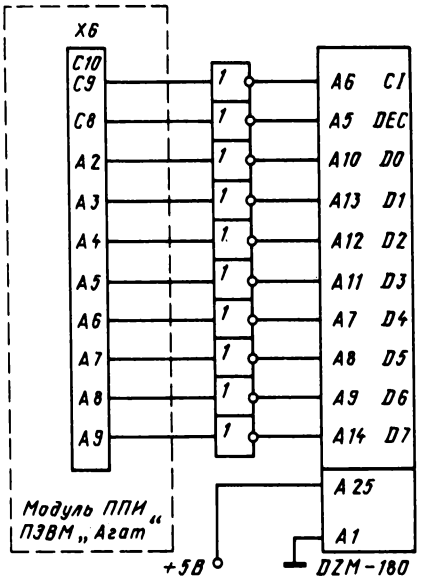


Рис. 11.1. Подключение к ПЭВМ "Агат" печатающего устройства "DZM-180"

Рис. 11.2. Драйвер устройства DZM-180

```
0300-  A9 03     LDA  #$03
0302-  8D 2D A2  STA  $A22D
0305-  A9 0B     LDA  #$0B
0307-  8D 2C A2  STA  $A22C
030A-  60        RTS
030B-  29 07     AND  #$07
030D-  D0 03     BNE  $0312
030F-  4C 90 FE  JMP  $FE90
0312-  0A        ASL
0313-  0A        ASL
0314-  0A        ASL
0315-  0A        ASL
0316-  8D 2D 03  STA  $032D
0319-  AA        TAX
031A-  A9 AB     LDA  #$AB
031C-  9D 83 C0  STA  $C083,X
031F-  A9 8D     LDA  #$8D
0321-  20 2F 03  JSR  $032F
0324-  A9 03     LDA  #$03
0326-  85 37     STA  $37
0328-  A9 2F     LDA  #$2F
032A-  85 36     STA  $36
032C-  60        RTS
032D-  50 01     BVC  $0330
032F-  48        PHA
0330-  8E 2E 03  STX  $032E
0333-  AE 2D 03  LDX  $032D
0336-  C9 E0     CMP  #$E0
0338-  90 02     BCC  $033C
033A-  49 80     EOR  #$80
033C-  C9 8D     CMP  #$8D
033E-  D0 02     BNE  $0342
0340-  A9 8A     LDA  #$8A
0342-  49 7F     EOR  #$7F
0344-  48        PHA
0345-  BD 82 C0  LDA  $C082,X
0348-  10 FB     BPL  $0345
034A-  68        PLA
034B-  9D 80 C0  STA  $C080,X
034E-  AE 2E 03  LDX  $032E
0351-  68        PLA
0352-  4C DF FD  JMP  $FDDF
0355-  00        BRK
```

например с адреса д300. Тогда передача ему управления должна производиться оператором CALL или POKE.

Передача управления на драйвер ПЗУ производится PR#N - в интерпретаторе языка БЕЙСИК или CN00G - в мониторе (N - номер разъема, в который установлен модуль ППИ).

На рис. 11.1 приведена схема подключения к ПЭВМ "Агат" DZM-180.

DZM-180 может управляться и драйвером D-100, тогда работа с ним аналогична работе с D-100. На рис. 11.2 приведен текст программы, управляющей печатью DZM-180.

11.2. РАСШИРЕНИЕ ГРАФИЧЕСКИХ ВОЗМОЖНОСТЕЙ

В базовом варианте ПЭВМ "Агат" бит Y (яркости) не используется. Несложная доработка модуля сопряжения (рис. 11.3) дает возможность использовать этот бит для указания яркости или ее половинного значения, что позволяет наблюдать на экране ВКУ две градации каждого из шести цветов. Задание Y = 1 выводит закодированный в битах R, G, B цвет полной яркостью. При Y = 0 яркость цвета уменьшается наполовину.

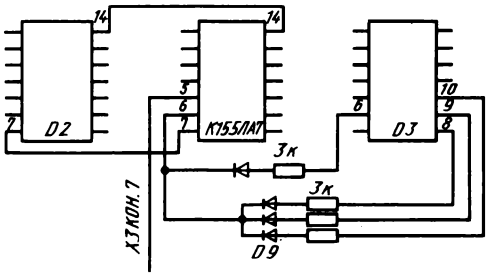


Рис. 11.3. Схема формирования полутона

11.3. ЗАДАНИЕ ЦВЕТА ФОНА И ПЕРА В ГРАФИЧЕСКОМ ИЗОБРАЖЕНИИ РАЗМЕРА 256Х256

На рис. 11.4 приведена принципиальная схема узла, преобра зующего черно-белую графику с разрешением 256х256 в цветную.

Схема собирается на плате стандартного размера и устанавливается в любой

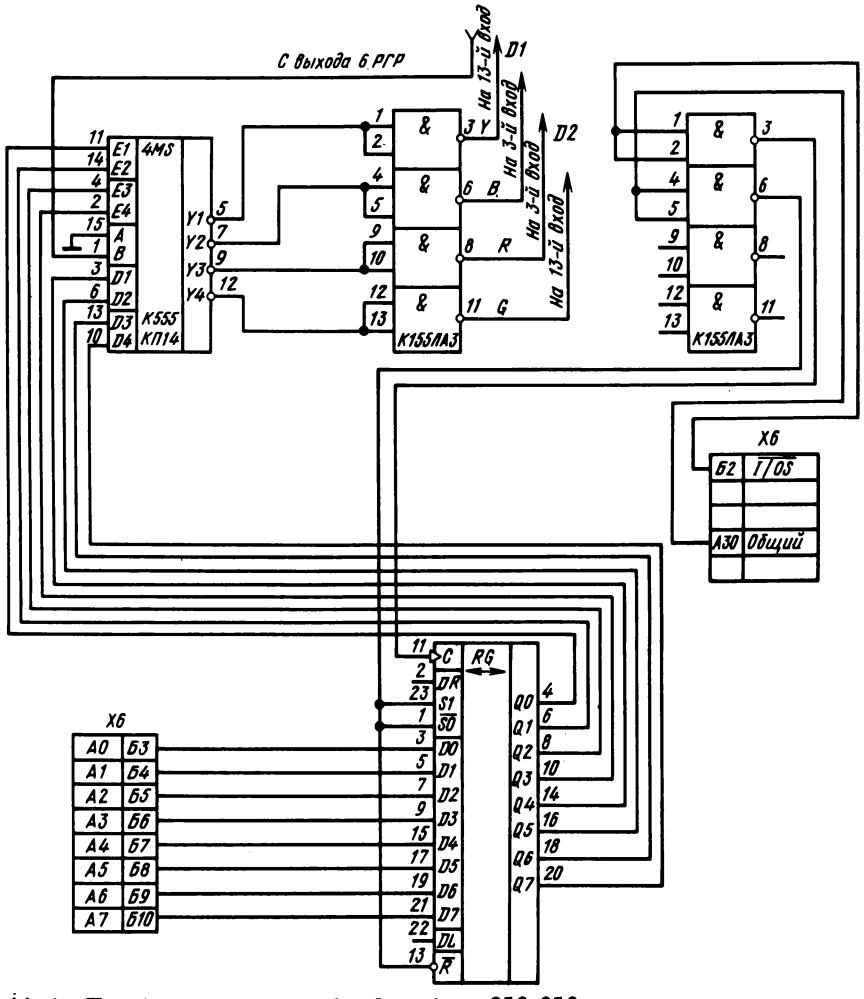


Рис. 11.4. Преобразование черно-белой графики 256х256 в цветную

свободный разъем. После этого навесным монтажом соединяются элементы, расположенные на плате памяти и интерфейса, с элементами установленного модуля; выход в РГР, входы *3* и *13 D1* и входы *3* и *13 D2* соединяются с элементами модуля согласно схеме на рис. 11.4.

Узел имеет в своем составе регистр К155ИР13, запоминающий соответственно четыре бита кода цвета фона и четыре бита кода цвета рисуемых точек (для режима ГВР). Коды цвета заносятся в регистр по сигналу $\overline{I/OS}$, соответствующему разъему, в который установлена плата. Сделать это можно, например, командой монитора С112.

Затем в регистр будет занесен код фона - 1 (что соответствует красному цвету) и код пера - 2 (что соответствует зеленому цвету). В режиме ГВР информация будет выводиться зеленым цветом по красному фону.

Отметим, что, используя узел, можно выводить на экран ВКУ двухцветную палитру из диапазона 256 цветов.

11.4. СОЗДЕНИЕ ЛОКАЛЬНОЙ СЕТИ ИЗ НЕСКОЛЬКИХ ПЭВМ

В гл. 6 уже упоминалось о возможности создания локальной сети и обмена по телефонному каналу.

Простейший способ создания локальной сети - соединение двух ПЭВМ через магнитофонный вывод. Более сложные варианты требуют разработки модуля (аналогичного показанному на рис. 11.4), осуществляющего коммутацию между несколькими компьютерами. Обмен данными по-прежнему удобнее выполнять через магнитофонный вход-выход, а сигналы на запрос должны непосредственно поступать на коммутатор добавочного модуля. Коммутатор может быть снабжен регистром, запоминающим запрос от соответствующей ПЭВМ. Это позволяет избавить от многократных повторений запроса в случае, если запрашиваемый компьютер за-

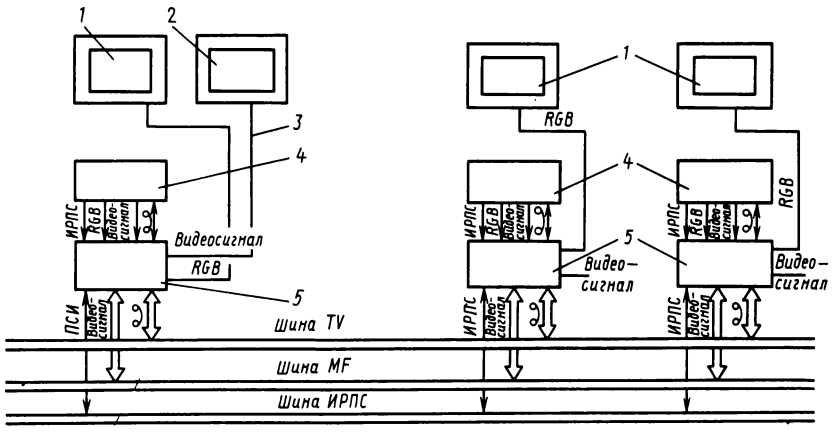


Рис. 11.5. Организация локальной сети:
1 **- ВКУ;** ***2*** **- дополнительный черно-белый монитор;** ***3*** **- шина видеосигнала;** ***4*** **- системный блок;** ***5*** **- программируемое коммутационное устройство и преобразователь RGB-видеосигнал и видеосигнал-RGB; ПСИ - последовательный сигнал**

нят. Подобное устройство несложно изготовить. При этом можно руководствоваться схемой, приведенной на рис. 11.5.

Несколько сложнее написать программу, обслуживающую этот модуль, в соответствии с потребностями диалога в постоянной локальной сети.

11.5. ПЕРЕДАЧА ДАННЫХ ПО ТЕЛЕФОННОМУ КАНАЛУ

Модуль ППИ организует обмен информацией (байтовыми посылками) в синхронном или асинхронном режиме. Это позволяет ПЭВМ "Агат" обмениваться данными с самыми разнообразными внешними устройствами. Однако уровень ТТЛ-логики трудно поддерживать неизменным на магистрали большой протяженности. К тому же существуют индуктивные и фоновые помехи, формирующиеся в проводнике значительной длины. Вот почему даже при наличии мощных шинных формирователей передача информации по линиям, превышающим 800 м, нецелесообразна. Простое повышение выходного напряжения также не решает эту проблему, так как приводит к значительному удорожанию связи.

Наиболее простой выход - воспользоваться телефонно-телеграфными каналами. В телефонной магистрали в момент разговора присутствует напряжение порядка 60 В, модулируемое микрофоном в соответствии с частотой говорящего. Если воспользоваться этим и подать на телефонный канал частотно-модулируемый сигнал с выхода ПЭВМ, то он будет передан аналогично тому, как передается разговорная речь. Однако, если при разговоре по телефонному каналу можно пренебречь помехами, то при передаче данных от компьютера каждая такая помеха вызывает ошибку.

Программа **WRITE**, входящая в состав системного монитора, управляя работой интерфейса магнитофона, преобразует информа- ционный бит в частотно-модулируемый выходной сигнал (логическая единица кодируется частотой 1 кГц, логический нуль - частотой 2 кГц).

Обработка прерываний выполняется программой **READ**. Таким образом, использование интерфейса магнитофона с программами **WRITE** и **READ** (на языке БЕЙСИК **SAVE** и **LOAD**) позволяет путем несложных доработок организовать обмен данными между несколькими ПЭВМ "Агат" по телефонно-телеграфным каналам. В этом случае роль телефонного микрофона выполняет интерфейс магнитофона. Для обратного преобразования в момент приема данных необходимо наличие аппаратного преобразователя высокого уровня (60 В) в низкий (порядка 250 мВ). Декодирование частоты можно выполнить программой **READ**. Использовать телефонный аппарат для преобразования напряжения не рекомендуется, так как он порождает помехи именно на частоте 1 - 2 кГц. Как показывает опыт, частоты 1 - 2 кГц - самые помехонасыщенные в телефонном канале. Для устойчивости передачи нужно либо передавать информацию малыми порциями (байт, два байта и т.д.), либо изменить частоту модулирования, снизив ее до 400 - 600 Гц, написав соответствующую программу.

11.6. КОМПЬЮТЕРНАЯ СИСТЕМА ОБУЧЕНИЯ

Компьютерная система обучения (КСО) предназначена для обучения и контроля полученных знаний. Она может использоваться в учебных процессах школы, техникума, училища, института, факультетов: КСО с успехом может применяться

на предприятиях для освоения и поддержания на высоком уровне профессиональных навыков, знаний по технике безопасности, пожарной безопасности и т.п.

Обязательным условием функционирования системы обучения является наличие локальной сети. КСО может быть реализована не только на компьютере "Агат", но и на любых других ПЭВМ, входящих в состав локальной сети.

КСО написана большей частью на языках высокого уровня (БЕЙСИК, ПАСКАЛЬ, Си), поэтому ее легко адаптировать для многих операционных систем (ДОС, СР/М, РАФОС, RSX, MSX и т.п.).

Для организации материала учебного процесса в КСО сформированы следующие базы: уроков; знаний; терминов; объектов; задач; образов.

База уроков содержит набор стандартных уроков по данному курсу обучения. По желанию пользователя эту базу можно расширить, включив в нее оригинальные уроки. База уроков формируется в соответствии с методологическими разработками по применению КСО в данном курсе обучения (например, использование принципа опорных сигналов В.Ф. Шаталова в школьном курсе обучения) разработчиками системы.

База знаний содержит информацию, характеризующую различные факты, события, процессы, свойства и т.п. (например, Блез Паскаль родился в 1623 г.; закон Паскаля: "Давление, производимое на жидкость или газ, передается без изменения в каждую точку жидкости или газа"; 1 Па = 1 $Н/м^2$ и т.п.). База знаний открыта для пользователей КСО, т.е. может изменяться и пополняться в процессе эксплуатации КСО.

База терминов состоит из совокупности терминов, их определений и характеристик, используемых для описания данных терминов. Например, термин "жидкость" связан и характеризуется терминами "текучесть", "плотность", "поверхностное натяжение", "коэфициент преломления", "давление", "объем" и т.д.

База объектов содержит элементы базы знаний, используемые в различных предметных областях, в том числе и в разных курсах обучения. Примером этого может служить межпредметная связь в школьном курсе обучения (химия - физика - математика - биология - география и т.д.). Использование элементов базы объектов позволяет расширить возможности обучения за счет установления (систематизации) разных точек зрения на один и тот же объект. Например, на жидкость как на объект существует множество точек зрения - химика, физика, биолога и т.д. С помощью базы объектов можно организовать изучение одного и того же объекта в разных предметных областях, формируя таким образом системное представление о данном объекте.

В базе задач все задачи квалифицируются по направлениям:

прямое использование полученных знаний;

предварительное выяснение возможности применения имеющихся знаний (обучаемому приходится абстрагироваться от условий задачи);

привлечение дополнительного материала из других разделов курса обучения;

преодоление логических и психологических сложностей в решении.

Пользователю предоставляется возможность расширить имеющуюся систему признаков.

Задачи в базе задач не являются жестко связанными с каким-либо предметом, что позволяет использовать их в разных предметах обучения.

База образов представляет собой набор графических изображений, иллюстрирующих элементы перечисленных выше баз.

Образы могут масштабироваться, вращаться, деформироваться. Управление этими функциями, а также извлечение образов из базы осуществляется манипулятором образов. Манипулятор представляет собой сканирующий по экрану курсор - указатель (управляемый с клавиатуры или специального устройства типа "джойстик", "мышь" и т.п.). Подводя курсор под выбранный образ, пользователь, выбрав соответствующий режим (масштабировать, переместить, изменить цвет, развернуть, деформировать по выбранному закону), манипулирует образом.

База образов содержит необходимый минимум стандартных образов, используемых в процессе обучения. База открыта для пользователя. Используя манипулятор образов, можно дополнить ее новыми образами.

С точки зрения пользователя в КСО реализовано два самостоятельных режима работы: подготовки урока (получения новых знаний; контроля и закрепления полученных знаний), демонстрация урока.

Режим подготовки урока предусматривает диалоговый режим работы с учителем. В ходе диалога происходит формирование содержания и организация материала будущего урока с учетом индивидуальных особенностей учителя, аудитории и содержания предмета обучения. При этом выделяются новые, неизвестные понятия, термины; устанавливаются логические связи между освоенными и новыми терминами; определяется последовательность подачи новых терминов; проводится членение изучаемого материала на блоки и определяется использование примеров и пояснений.

В режиме подготовки урока пользователь (учитель) может сформировать свой индивидуальный урок на основе стандартного урока, взятого из базы уроков. Учитель имеет возможность дополнять урок элементами, извлеченными из баз КСО, а также составить урок, полностью отличающийся от стандартного. Именно этот, подготовленный им урок, учитель будет использовать на своих занятиях (в режиме проведения урока), со временем накапливая собственную базу уроков.

Формирователь урока позволяет многократно моделировать ход урока в любом направлении. Он выводит на экран компьютера общую схему урока и его составные части (блоки). Пользуясь манипуляторами формирователя урока, учитель включает в блоки элемента из баз КСО, устанавливает связи между ними. При этом вся необходимая информация выводится на экран, а выбор и включение ее производится простым манипулированием образами. Система фиксирует и информирует о продолжительности отображения на экране всех составных частей урока. Кроме того, пользователю предоставляется возможность задать продолжительность любого фрагмента урока.

Работа в режиме подготовки урока позволяет совершенствовать процесс обучения, давая учителю возможность моделировать будущий урок и за счет этого развивая самого учителя.

Моделирование хода урока и его содержания позволяет студентам (будущим преподавателям) без аудитории, а на модели готовиться к проведению занятий, в том числе и в классах, не имеющих КСО.

Формирование урока контроля происходит по аналогичной схеме формирователем урока. В базе уроков содержатся стандартные уроки контроля по каждой

изучаемой теме, кроме того, используя базу задач, в урок контроля могут быть включены другие задачи по выбору учителя. В распоряжение учителя предоставлен логический наборник, используя который можно моделировать новые задачи, включая их в базу задач и конкретные уроки. Логический наборник располагает образами логических элементов по каждому разделу обучения, и, манипулируя ими, учитель по трафаретам создает условие задачи и основные варианты ее решения.

Используя заложенную в формирователь схему продвижения по уровням сложности задач, учитель заполняет уровни схемы соответствующими задачами и решениями, а также вводит баллы системы оценки решений.

Работа КСО ***в режиме получения новых знаний*** позволяет совместить рассказ преподавателя с иллюстрацией этого рассказа на экране монитора (для каждого ученика). Изображение, сопровождающее рассказ, - динамическое (мультипликационное), что позволяет моделировать в ходе рассказа опыты, явления, схемы, характерные ситуации, ошибочные приемы и т.п.

Использование временных характеристик позволяет сжимать или растягивать иллюстрации во времени (например, одну и ту же информацию можно дать за 5 или 10 мин в зависимости от уровня подготовки аудитории). Возможности этого режима делают процесс обучения более управляемым, позволяют учителю задавать его динамику.

Ученик, используя подготовленную к уроку информацию из базы знаний, терминов, образов, расширяет связи изучаемого материала с предыдущими уроками, заимствуя их фрагменты; связи со свойствами и особенностями изучаемого объекта или явления в других областях знаний, используя для этого базу объектов.

В ***режиме контроля и закрепления полученных знаний*** каждый ученик получает на свой монитор условие задачи и подготовленные учителем к этой задаче элементы из логического наборника. Ученик, манипулируя этими элементами, формирует на экране монитора решение задачи и вводит его для проверки. В зависимости от результатов проверки решения КСО подводит ученика к задачам большей или меньшей сложности, предоставляя ученику комментарии или разъяснения.

Параллельно КСО по заданной схеме ведет учет результатов контроля обучения, формируя соответствующий протокол.

Локальная сеть позволяет учителю эффективно и своевременно контролировать действия ученика; в свою очередь, ученик может обратиться к учителю за разъяснениями либо вызвать и повторно изучить необходимый ему материал.

Компьютерная система обучения, реализованная на объединенных в сеть ПЭВМ, позволяет значительно ускорить процесс обучения, углубить знания в различных предметных областях и сформировать системное понимание изучаемых объектов и явлений.

Таблица команд микропроцессора 6502

Мнемоника команды	Функция команды	Методы адресации	Запись команды на языке ассемблера	Число машинных циклов	Шестнадцатеричный код	Число байтов	Используемые флажки регистра P 76543210 NV.BDIZC
1	2	3	4	5	6	7	8
ADC	A + M + C →A,C	IMM	ADC# OPER	2	69	2	VN ... ZC
		ZP	ADC OPER	3	65	2	
		ZP,X	ADC OPER,X	4	75	2	
		ABS	ADC OPER	4	6D	3	
		ABS,X	ADC OPER,X	4(1)	7D	3	
		ABS,Y	ADC OPER,Y	4(1)	79	3	
		(IND,X)	ADC (OPER,X)	6(5)	61	2	
		(IND,Y)	ADC (OPER),Y	5(1)	71	2	
AND	АЛМ → A	IMM	AND# OPER	2	29	2	N ... Z
		ZP	AND OPER	3	25	2	
		ZP,X	AND OPER,X	4	35	2	
		ABS	AND OPER	4	2D	3	
		ABS,X	AND OPER,X	4(1)	3D	3	
		ABS,Y	AND OPER,Y	4(1)	39	3	
		(IND,X)	AND (OPER,X)	6(5)	21	2	
		(IND),Y	AND (OPER),Y	5(1)	31	2	
ASL	7 0 C ←[байт]← O	ACC	ASL A	2	0A	1	N ... ZC

Мнемоника команды	Функция команды	Методы адресации	Запись команды на языке ассемблера	Число машинных циклов	Шестнадцатеричный код	Число байтов	Используемые флажки регистра P 76543210 NV.BDIZC
1	2	3	4	5	6	7	8
		ZP	ASL OPER	5	06	2	
		ZP,X	ASL OPER,X	6	16	2	
		ABS	ASL OPER	6	0E	3	
		ABS,X	ASL OPER,X	7	1E	3	
BCC	ПЕРЕХОД ПРИ C = 0	REL	BCC OPER	2 2 + 1 2 + 2	90	2	
BCS	ПЕРЕХОД ПРИ C = 1	REL	BCS OPER	2 2 + 1 2 + 2	B0	2	
BEQ	ПЕРЕХОД ПРИ Z = 1	REL	BEQ OPER	2 2 + 1 2 + 2	F0	2	
BIT	АЛМ M7 → N M6 → V	ZP ABS	BIT OPER BIT OPER	3 4	24 2C	2 3	M7M6 ...Z

Мнемоника команды	Функция команды	Методы адресации	Запись команды на языке ассемблера	Число машинных циклов	Шестнадцатеричный код	Число байтов	Используемые флажки регистра P 76543210 NV.BDIZC
1	2	3	4	5	6	7	8
BMI	ПЕРЕХОД ПРИ N = 1	REL	BMI OPER	2 2 + 1 2 + 2	30	2	
BNE	ПЕРЕХОД ПРИ Z = 0	REL	BNE OPER	2 2 + 1 2 + 2	D0	2	
BPL	ПЕРЕХОД ПРИ N = 0	REL	BPL OPER	2 2 + 1 2 + 2	10	2	
BRK	ПРОГРАММНОЕ ПРЕРЫВАНИЕ	IMPL	BRK	7	00	1	...1...
BVC	ПЕРЕХОД ПРИ V = 0	REL	BVC OPER	2 2 + 1 2 + 2	50	2	
	ПРИ V = 1			2 + 1 2 + 2	70	2	

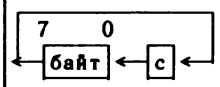
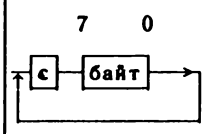
Мнемоника команды	Функция команды	Методы адресации	Запись команды на языке ассемблера	Число машинных циклов	Шестнадцатеричный код	Число байтов	Используемые флажки регистра P 76543210 NV.BDIZC
1	2	3	4	5	6	7	8
CLC	0 ⟶ C	IMPL	CLC	2	18	1	0
CLD	0 ⟶ D	IMPL	CLD	2	D8	1	...0...
CLI	0 ⟶ I	IMPL	CLI	2	58	1	...0...
CLV	0 ⟶ V	IMPL	CLV	2	B8	1	.0.....
CMP	A ⟶ M	IMM	CMP # OPER	2	C9	2	N.....ZC
		ZP	CMP OPER	3	C5	2	
		ZP,X	CMP OPER,X	4	D5	2	
		ABS	CMP OPER	4	CD	3	
		ABS,X	CMP OPER,X	4(1)	DD	3	
		ABS,Y	CMP OPER,Y	4(1)	D9	3	
		(IND,X)	CMP (OPER,X)	6(5)	C1	2	
		(IND),X	CMP (OPER),Y	5(1)	D1	2	
CPX	X - M	IMM	CPX # OPER	2	E0	2	N.....ZC
		ZP	CPX OPER	3	E4	2	
		ABS	CPX OPER	4	EC	3	

Мнемоника команды	Функция команды	Методы адресации	Запись команды на языке ассемблера	Число машинных циклов	Шестнадцатеричный код	Число байтов	Используемые флажки регистра P 76543210 NV.BDIZC
1	2	3	4	5	6	7	8
CPY	Y - M	IMM	CPY # OPER	2	C0	2	N.....ZC
		ZP	CPY OPER	3	C4	2	
		ABS	CPY OPER	4	CC	3	
DEC	M - 1 → M	ZP	DEC OPER	5	C6	2	N.....Z.
		ZP,X	DEC OPER,X	6	D6	2	
		ABS	DEC OPER	6	CE	3	
		ABS,X	DEC OPER,X	7	DE	3	
DEX	X - 1 → X	IMPL	DEX	2	CA	1	N.....Z
DEY	Y - 1 → Y	IMPL	DEY	2	88	1	N.....Z
EOR	A.XOR.M → A	IMM	EOR # OPER	2	49	2	N.....Z
		ZP	EOR OPER	3	45	2	
		ZP,X	EOR OPER,X	4	55	2	
		ABS	EOR OPER	4	4D	3	
		ABS,X	EOR OPER,X	4(1)	5D	3	
		ABS,Y	EOR OPER,Y	4(1)	59	3	
		(IND,X)	EOR (OPER,X)	6(5)	41	2	
		(IND),Y	EOR (OPER),Y	5(1)	51	2	

Мнемоника команды	Функция команды	Методы адресации	Запись команды на языке ассемблера	Число машинных циклов	Шестнадцатеричный код	Число байтов	Используемые флажки регистра P 76543210 NV.BDIZC
1	2	3	4	5	6	7	8
INC	M + 1 → M	ZP	INC OPER	5	E6	2	N.....Z.
		ZP,X	INC OPER,X	6	F6	2	
		ABS	INC OPER	6	EE	3	
		ABS,X	INC OPER,X	7	FE	3	
INX	X + 1 → X	IMPL	INX	2	E8	1	N.....Z.
INY	Y + 1 → Y	IMPL	INY	2	C8	1	N.....Z.
JMP	ADL → PCL	ABS	JMP OPER	3	4C	3	.....
	ADN → PCH	IND	JMP (OPER)	5	6C	3	
JSR	PC ← PC + 2 MS ← PCH S ← S - 1 MS ← PCL S ← S - 1 PCL← ADL PCH← ADH	ABS	JSR OPER	6	20	3	.....
LDA	M → A	IMM	LDA # OPER	2	A9	2	N.....Z.
		ZP	LDA OPER	3	A5	2	

Мнемоника команды	Функция команды	Методы адресации	Запись команды на языке ассемблера	Число машинных циклов	Шестнадцатеричный код	Число байтов	Используемые флажки регистра P 76543210 NV.BDIZC
1	2	3	4	5	6	7	8
LDA	M → A	ZP,X	LDA OPER,X	4	B5	2	N....Z.
		ABS	LDA OPER	4	AD	3	
		ABS,X	LDA OPER,X	4(1)	BD	3	
		ABS,Y	LDA OPER,Y	4(1)	B9	3	
		(IND,X)	LDA (OPER,X)	6(5)	A1	2	
		(IND),Y	LDA (OPER),Y	5(1)	B1	2	
LDX	M → X	IMM	LDX # OPER	2	A2	2	N.....Z.
		ZP	LDX OPER	3	A6	2	
		ZP,Y	LDX OPER,Y	4	B6	2	
		ABS	LDX OPER	4	AE	3	
		ABS,Y	LDX OPER,Y	4(1)	BE	3	
LDY	M → Y	IMM	LDY # OPER	2	A0	2	N.....Z.
		ZP	LDY OPER	3	A4	2	
		ZP,Y	LDY OPER,Y	4	B4	2	
		ABS	LDY OPER	4	AC	3	
		ABS,Y	LDY OPER,Y	4(1)	BC	3	
LSR	7 0 0 → [байт] → C	ACC	LSR A	2	4A	1	0.....ZC
		ZP	LSR OPER	5	46	2	

Мнемоника команды	Функция команды	Методы адресации	Запись команды на языке ассемблера	Число машинных циклов	Шестнадцатеричный код	Число байтов	Используемые флажки регистра P 76543210 NV.BDIZC
1	2	3	4	5	6	7	8
		ZP,X	LSR OPER,X	6	56	2	
		ABS	LSR OPER	6	4E	3	
		ABS,X	LSR OPER,X	7	5E	3	
NOP	PC = PC + 1	IMPL	NOP	2	EA	1	.....
ORA	A.OR.M → A	IMM	ORA # OPER	2	09	2	N.....X.
		ZP	ORA OPER	3	05	2	
		ZP,X	ORA OPER,X	4	15	2	
		ABS	ORA OPER	4	0D	3	
		ABS,X	ORA OPER,X	4(1)	1D	3	
		ABS,Y	ORA OPER,Y	4(1)	19	3	
		(IND,X)	ORA (OPER,X)	6(5)	01	2	
		(IND),Y	ORA (OPER),Y	5(1)	11	2	
PHA	A → MS S - 1 → S	IMPL	PHA	3	48	1	.....
PHP	P → MS S - 1 → S	IMPL	PHP	3	08	1	.....

Мнемоника команды	Функция команды	Методы адресации	Запись команды на языке ассемблера	Число машинных циклов	Шестнадцатеричный код	Число байтов	Используемые флажки регистра P 76543210 NV.BDIZC
1	2	3	4	5	6	7	8
PLA	S + 1 → S MS → A	IMPL	PLA	4	68	1	.....
PLP	S + 1 → S MS → P	IMPL	PLP	4	28	1	от стека
ROL		ACC ZP ZP,X ABS ABS,X	ROL A ROL OPER ROL OPER,X ROL OPER ROL OPER,X	2 5 6 6 7	2A 26 36 2E 3E	1 2 2 3 3	N.....ZC
ROR		ACC ZP ZP,X ABS ABS,X	ROR A ROR OPER ROR OPER,X ROR OPER ROR OPER,X	2 5 6 6 7	6A 66 76 6E 7E	1 2 2 3 3	N.....ZC
RTI	S + 1 → S MS → P S + 1 → S MS → PCL	IMPL	RTI	6	40	1	от стека

Мнемоника команды	Функция команды	Методы адресации	Запись команды на языке ассемблера	Число машинных циклов	Шестнадцатеричный код	Число байтов	Используемые флажки регистра P 76543210 NV.BDIZC
1	2	3	4	5	6	7	8
	S + 1 → S MS → PCH						
RTS	S + 1 → S MS → PCL S + 1 → S MS → PCH PC + 1 → PC	IMPL	RTS	6	60	1	.....
SBC	A-M-$\overline{C}$ → A,C ($\overline{C}$ заем)	IMM ZP ZP,X ABS ABS,X ABS,Y (IND,X) (IND),Y	SBC # OPER SBC OPER SBC OPER,X SBC OPER SBC OPER,X SBC OPER,Y SBC (OPER,X) SBC (OPER),Y	2 3 4 4 4(1) 4(1) 6(5) 5(1)	E9 E5 F5 ED FD F9 E1 F1	2 2 2 3 3 3 2 2	NV....ZC
SEC	1 → C	IMPL	SEC	2	38	1	1
SED	1 → D	IMPL	SED	2	F8	1	....1...

Мнемоника команды	Функция команды	Методы адресации	Запись команды на языке ассемблера	Число машинных циклов	Шестнадцатеричный код	Число байтов	Используемые флажки регистра P 76543210 NV.BDIZC
1	2	3	4	5	6	7	8
SEI	1 → I	IMPL	SEI	2	78	1	.....1..
STA	A → M	ZP	STA OPER	3	85	2	.....
		ZP,X	STA OPER,X	4	95	2	
		ABS	STA OPER	4	8D	3	
		ABS,X	STA OPER,X	5	9D	3	
		ABS,Y	STA OPER,Y	5	99	3	
		(IND,X)	STA (OPER,X)	6(5)	81	2	
		(IND),Y	STA (OPER),Y	6	91	2	
STX	X → M	ZP	STX OPER	3	86	2	.....
		ZP,Y	STX OPER,Y	4	96	2	
		ABS	STX OPER	4	8E	3	
STY	Y → M	ZP	STY OPER	3	84	2	.....
		ZP,X	STY OPER,X	4	94	2	
		ABS	STY OPER	4	8C	3	
TAX	A → X	IMPL	TAX	2	AA	1	N.....Z
TAY	A → Y	IMPL	TAY	2	A8	1	N.....Z

Мнемоника команды	Функция команды	Методы адресации	Запись команды на языке ассемблера	Число машинных циклов	Шестнадцатеричный код	Число байтов	Используемые флажки регистра P 76543210 NV.BDIZC
1	2	3	4	5	6	7	8
TSX	S → X	IMPL	TSX	2	BA	1	N.....Z
TXA	X → A	IMPL	TXA	2	8A	1	N.....Z
TXS	X → S	IMPL	TXS	2	9A	1	.....
TYA	Y → A	IMPL	TYA	2	98	1	N.....Z

Приложение 2
Программы на языке БЕЙСИК

```
1  REM ФИО=А.М.И-й,ТЕЛ=450
     60 71
100  TEXT= 15: NORMAL : HOME
     : RIBBON= 2
101  FOR I = 2 TO 31: VTAB
     2: HTAB I: PRINT "*":
      VTAB 31: HTAB I: PRINT
     "*": NEXT
102  FOR I = 2 TO 31: VTAB
     I: HTAB 2: PRINT "*":
      VTAB I: HTAB 31: PRINT
     "*": NEXT
105  RIBBON= 3
110  VTAB 6: HTAB 13
120  PRINT "НИИШОТСО"
130  VTAB 8: HTAB 13
140  PRINT "АПН СССР"
150  VTAB 15: HTAB 5
160  PRINT "КЛЮЧЕВЫЕ СЛОВА
      БЕЙСИКА"
190  VTAB 18: HTAB 11
200  PRINT "ИГЛИЦКИЙ А.М."

210  VTAB 28: HTAB 12
220  PRINT "МОСКВА 1986"
300  FOR I = 1 TO 5000: NEXT

310  HOME
400  RIBBON= 3
410  PRINT "ПРОГРАММА ПРЕД
     НАЗНАЧЕНА ДЛЯ ВЫ-": PRINT

420  PRINT "РАБОТКИ НАВЫКО
     В ПЕЧАТИ НАИБОЛЕЕ": PRINT

430  PRINT "ЧАСТО ВСТРЕЧАЮ
     ЩИХСЯ КЛЮЧЕВЫХ": PRINT

440  PRINT "СЛОВ БЕЙСИКА.
     НЕОБХОДИМО ПЕЧА-": PRINT

450  PRINT "ТАТЬ СЛОВА, ВЫ
     БИРАЕМЫЕ МАШИНОЙ В"
460  PRINT "СЛУЧАЙНОМ ПОРЯ
     ДКЕ. ПРИ ЭТОМ ПОД-"
470  PRINT "СЧИТЫВАЕТСЯ ОБ
     ЩЕЕ КОЛИЧЕСТВО": PRINT

480  PRINT "СЛОВ И СРЕДНЕЕ
      ВРЕМЯ РЕАКЦИИ."
500  FOR I = 1 TO 10000: NEXT
510  HOME

1000  DIM W$(100)
1110 W$(1) = "CATALOG"
1120 W$(2) = "INIT"
1130 W$(3) = "RENAME"
1140 W$(4) = "DELETE"
1150 W$(5) = "LOCK"
1160 W$(6) = "UNLOCK"
1170 W$(7) = "LOAD"
1180 W$(8) = "SAVE"
1190 W$(9) = "RUN"
1200 W$(10) = "BLOAD"
1210 W$(11) = "BSAVE"
1220 W$(12) = "BRUN"
1230 W$(13) = "NEW"
1240 W$(14) = "STOP"
1250 W$(15) = "END"
1260 W$(16) = "CONT"
1270 W$(17) = "TRACE"
1280 W$(18) = "NOTRACE"
1290 W$(19) = "PEEK"
1300 W$(20) = "POKE"
1310 W$(21) = "WAIT"
1320 W$(22) = "CALL"
1330 W$(23) = "HIMEM"
1340 W$(24) = "LOMEM"
1350 W$(24) = "LIST"
1360 W$(26) = "DEL"
1370 W$(27) = "REM"
1380 W$(28) = "VTAB"
1390 W$(29) = "HTAB"
1400 W$(30) = "HOME"
1410 W$(31) = "CLEAR"
1420 W$(32) = "FLASH"
1430 W$(33) = "INVERSE"
1440 W$(34) = "NORMAL"
1450 W$(35) = "SPEED"
1460 W$(36) = "DIM"
1470 W$(37) = "LEN"
1480 W$(38) = "STR"
1490 W$(39) = "VAL"
1500 W$(40) = "CHR"
1510 W$(41) = "ASC"
1520 W$(42) = "LEFT"
1530 W$(43) = "RIGHT"
1540 W$(44) = "MID"
1550 W$(45) = "INPUT"
1560 W$(46) = "GET"
1570 W$(47) = "READ"
1580 W$(48) = "DATA"
1590 W$(49) = "RESTORE"
1600 W$(50) = "PRINT"
1610 W$(51) = "GOTO"
1620 W$(52) = "GOSUB"
1630 W$(53) = "RETURN"
```

```
1640 W$(54) = "ON"
1650 W$(55) = "IF"
1660 W$(56) = "THEN"
1670 W$(57) = "FOR"
1680 W$(58) = "TO"
1690 W$(59) = "STEP"
1700 W$(60) = "NEXT"
1710 W$(61) = "ONERR"
1720 W$(62) = "RESUME"
1730 W$(63) = "GR"
1740 W$(64) = "MGR"
1750 W$(65) = "HGR"
1760 W$(66) = "COLOR"
1770 W$(67) = "RIBBON"
1780 W$(68) = "PLOT"
1790 W$(69) = "SCRN"
1800 W$(70) = "XDRAW"
1810 W$(71) = "SCALE"
1820 W$(72) = "ROT"
1830 W$(73) = "TEXT"
1840 W$(74) = "PDL"
1850 W$(75) = "SIN"
1860 W$(76) = "COS"
1870 W$(77) = "TAN"
1880 W$(78) = "ATN"
1890 W$(79) = "INT"
1900 W$(80) = "RND"
1910 W$(81) = "SGN"
1920 W$(82) = "ABS"
1930 W$(83) = "SQR"
1940 W$(84) = "EXP"
1950 W$(85) = "LOG"
1960  KLOVO = 0
1970  SUMVR = 0
1990  VR = 10
2000 N =  INT (100 *  RND
     (1)) + 1
2010  WORD$ = W$(N)
2020 L =  LEN ( WORD$)
2030  NB = 1
2045  IF L = 0 THEN 2000
2050 T = 0
2060  KOLVO =  KOLVO + 1
2070  POKE 32,26: POKE 33,
     14: HOME
2100  POKE $C010,0
3000  POKE 32,0: POKE 33,6
     4
3010 V = 31
3020 H = 14
3030  VTAB V
3040  HTAB H
3050  RIBBON= 6
3060  PRINT  WORD$
3110  FOR I = 1 TO  VR
3120  IF  PEEK ($C000) > 1
     28 THEN 4000
3130 T = T + 1
3140  NEXT
3200 V = V - 1
3210  IF V = 2 THEN V = 3
3220  VTAB V
3230  HTAB H
3240  RIBBON= 2
3250  PRINT  WORD$
3260  GOTO 3110
4000  BUK1$ =  MID$ ( WORD
     $, NB,1)
4010  BUK2$ =  CHR$ ( PEEK
     ($C000) - 128)
4020  POKE $C010,0
4030  IF  BUK1$ <  > BUK2
     $ THEN 5000
4100  VTAB 31
4110  HTAB 14 +  NB - 1
4120  RIBBON= 2
4130  PRINT  BUK2$
4140  IF  NB <  > L THEN 4
     200
4150  SUMVR =  SUMVR + T
4155  RIBBON= 3: VTAB 17: HTAB
     6: PRINT  KOLVO: VTAB
     17: HTAB 23: PRINT  INT
     ( SUMVR /  KOLVO + 0.
     5)
4160  VTAB V - 2: HTAB 14:
      RIBBON= 2: PRINT T
4165  FOR T = 1 TO 800: NEXT

4170  VR =  VR * (1 - 0.25
      *  SGN (V - 17))
4180  GOTO 2000
4200  NB =  NB + 1
4210  GOTO 3120
5000  VTAB 31
5010  HTAB 14 +  NB - 1
5020  RIBBON= 5
5030  PRINT  BUK2$
5040  FOR J = 0 TO 25
5050 Z =  PEEK ($C030)
5060  NEXT
5070  GOTO 3120
```

Программа "Теннис"

```
10  MGR= 7
14  REM ------------------
     ---
15  REM *** МУЗЫКА - DATA
     ***
16  REM ------------------
     ---
20  DATA 173,48,192,136,20
     8,5,206,1,3,240,9
30  DATA 202,208,245,174,0
     ,3,76,2,3,96,0,0
40  FOR X = 770 TO 792
50  READ Y
60  POKE X,Y
70  NEXT X
72  REM -------------
73  REM *** МОТИВ ***
74  REM -------------
75 O = 10
79  REM ------------------
     -----
80  REM *** КАРТИНКА  SPAS
     E ***
81  REM ------------------
     -----
90  FOR A = 0 TO 600 STEP
     5
100  IF A = 150 THEN  GOSUB
     300
110  IF A = 300 THEN  GOSUB
     300
120  IF A = 450 THEN  GOSUB
     300
130  IF A = 600 THEN  GOSUB
     300
140  POKE 768,0: POKE 769,
     3: CALL 770
200 C =  INT ( RND (1) * 1
     5)
210 X =  INT ( RND (1) * 1
     27)
220 Y =  INT ( RND (1) * 1
     27)
230  IF X > 31 AND X < 97 THEN
      IF Y > 39 AND Y < 89
      THEN 200
245  COLOR= C
246  PLOT X,Y
247  REM -------------
248  REM *** РАМКА ***
249  REM -------------
251  COLOR= 3
252  IF A = 150 THEN  PLOT
     32,40 TO 96,40 TO 96,
     88 TO 32,88 TO 32,40:
     O = O + 7
253  COLOR= 1
254  IF A = 300 THEN  PLOT
     33,41 TO 95,41 TO 95,
     87 TO 33,87 TO 33,41:
     O = O + 5
255  COLOR= 1
256  IF A = 450 THEN  PLOT
     34,42 TO 94,42 TO 94,
     86 TO 34,86 TO 34,42:
     O = O + 5
257  COLOR= 3
258  IF A = 600 THEN  PLOT
     35,43 TO 93,43 TO 93,
     85 TO 35,85 TO 35,43
270  NEXT A
280  GOTO 320
289  REM --------------
290  REM ***  SINTEZ ***
291  REM --------------
300  FOR Z = 1 TO 20
305 B = Z * 2
310  POKE 768,B: POKE 769,
     Z: CALL 770
315  NEXT Z: RETURN
316  REM ------------------
     --------
317  REM *** КАРТИНКА - TE
     NNIS ***
318  REM ------------------
     --------
320  COLOR= 2: FOR X = 52 TO
     75: PLOT X,61 TO X,82
     : NEXT X
322  COLOR= 7: PLOT 52,61 TO
     75,61: PLOT 52,82 TO
     75,82: PLOT 64,62 TO
     64,81: COLOR= 1: PLOT
     54,67 TO 54,73: COLOR=
     4: PLOT 73,80 TO 73,7
     4
330  REM
335  REM -------------
340  REM *** ВЫБОР ***
345  REM -------------
350 B = 0:A = 0: DEF  FN  KLV
     (S) = ( PEEK ($C000))
     : POKE $C010,0
360 A =  FN  KLV(0): POKE
     768,20: POKE 769,20: CALL
     770
365  IF B = 1 THEN  GOSUB
     1100:X = 43:Y = 67: GOSUB
     1120:X = 84:Y = Y + 2
     : GOSUB 1140
370  IF B = 2 THEN  GOSUB
     1105:X = 84:Y = 67: GOSUB
     1120:X = 43:Y = Y + 2
     : GOSUB 1140
```

```
375  IF B = 3 THEN  GOSUB
     1110:X = 43:Y = 67: GOSUB
     1120:X = 84: GOSUB 11
     20
380  IF B = 4 THEN  GOSUB
     1115:X = 43:Y = 69: GOSUB
     1140:X = 84: GOSUB 11
     40
385  IF A <  > 0 THEN  POKE
     $C010,0: GOTO 400
390  IF B = 5 THEN 400
392  GOSUB 1000:B = B + 1
395  GOTO 360
400  REM
410  PRINT : PRINT  CHR$ (
     4);"RUN TEN.MOD"
1000  REM ----------------
     ------
1001  REM *** СЛОВО - TENN
     IS ***
1002  REM ----------------
     ------
1005  FOR C = 1 TO 4
1010  COLOR= C:Y = 46: GOSUB
     1060
1015 Y = 47: GOSUB 1060: PLOT
     60,Y: PLOT 70,Y: PLOT
     82,Y
1020  COLOR= C + 8:Y = 48:
      GOSUB 1065: PLOT 60,
     Y TO 61,Y: PLOT 70,Y TO
     71,Y
1025 Y = 49: GOSUB 1065: PLOT
     60,Y TO 62,Y: PLOT 70
     ,Y TO 72,Y
1030  COLOR= C + 1:Y = 50:
      GOSUB 1065: PLOT 61,
Y TO 63,Y: PLOT 71,Y TO
     73,Y
1035 Y = 51: GOSUB 1070: PLOT
     62,Y TO 63,Y: PLOT 72
     ,Y TO 73,Y: PLOT 82,Y

1040  COLOR= C + 9:Y = 52:
      GOSUB 1070: PLOT 63,
     Y: PLOT 73,Y: PLOT 89
     ,Y
1045 Y = 53: GOSUB 1075: COLOR=
     C + 2:Y = 54: GOSUB 1075:Y
      = 55: GOSUB 1075
1050  COLOR= C + 10:Y = 56
     : GOSUB 1080: PLOT 89
     ,Y:Y = 57: GOSUB 1080

1055  NEXT C: RETURN
1060  PLOT 38,Y TO 45,Y: PLOT
     48,Y TO 55,Y: PLOT 58
     ,Y TO 59,Y: PLOT 64,Y
      TO 65,Y: PLOT 68,Y TO
     69,Y: PLOT 74,Y TO 75
     ,Y: PLOT 78,Y TO 79,Y
     : PLOT 83,Y TO 89,Y
1062  RETURN
1065  PLOT 41,Y TO 42,Y: PLOT
     48,Y TO 49,Y: PLOT 58
     ,Y TO 59,Y: PLOT 64,Y
      TO 65,Y: PLOT 68,Y TO
     69,Y: PLOT 74,Y TO 75
     ,Y: PLOT 78,Y TO 79,Y
     : PLOT 82,Y TO 83,Y
1067  RETURN
1070  PLOT 41,Y TO 42,Y: PLOT
     48,Y TO 53,Y: PLOT 58
     ,Y TO 59,Y: PLOT 64,Y
      TO 65,Y: PLOT 68,Y TO
     69,Y: PLOT 74,Y TO 75
     ,Y: PLOT 78,Y TO 79,Y
     : PLOT 83,Y TO 89,Y
1072  RETURN
1075  PLOT 41,Y TO 42,Y: PLOT
     48,Y TO 49,Y: PLOT 58
     ,Y TO 59,Y: PLOT 64,Y
      TO 65,Y: PLOT 68,Y TO
     69,Y: PLOT 74,Y TO 75
     ,Y: PLOT 78,Y TO 79,Y
     : PLOT 88,Y TO 89,Y
1077  RETURN
1080  PLOT 41,Y TO 42,Y: PLOT
     48,Y TO 55,Y: PLOT 58
     ,Y TO 59,Y: PLOT 64,Y
      TO 65,Y: PLOT 68,Y TO
     69,Y: PLOT 74,Y TO 75
     ,Y: PLOT 78,Y TO 79,Y
     : PLOT 82,Y TO 88,Y
1082  RETURN
1085  REM ----------------
     -------
1090  REM *** ФЫННКЦИИ ВЫБ
     ОРА ***
1095  REM ----------------
     -------
1100  DEF  FN  LI(Q) = ( INT
     ( PDL (0) / 4)): DEF
      FN  RI(Q) = (Y +  INT
     ( RND (1) * 6)): RETURN

1105  DEF  FN  LI(P) = (Y +
      INT ( RND (1) * 6)):
      DEF  FN  RI(P) = ( INT
     ( PDL (1) / 4)): RETURN

1110  DEF  FN  LI(P) = ( INT
     ( PDL (0) / 4)): DEF
      FN  RI(P) = ( INT ( PDL
     (1) / 4)): RETURN
1115  DEF  FN  LI(P) = (Y +
      INT ( RND (1) * 6)):
      DEF  FN  RI(P) = (Y +
      INT ( RND (1) * 6)):
      RETURN
1120  COLOR= 0: FOR S = X -
     4 TO X + 4: PLOT S,Y TO
     S,Y + 12: NEXT S
1123  COLOR= 5
```

```
1125  PLOT X,Y TO X,Y + 7 TO
      X - 2,Y + 9 TO X - 2,
      Y + 12: PLOT X + 2,Y +
      12 TO X + 2,Y + 9 TO
      X,Y + 7: PLOT X - 4,Y
       + 6 TO X - 2,Y + 4 TO
      X + 2,Y + 4 TO X + 4,
      Y + 6: PLOT X - 1,Y TO
      X - 1,Y + 2: PLOT X +
      1,Y TO X + 1,Y + 2
1130  RETURN
```

```
1140  COLOR= 0: FOR S = X -
      4 TO X + 4: PLOT S,Y -
      2 TO S,Y + 12: NEXT S

1145  COLOR= 1: PLOT X - 3
      ,Y TO X + 3,Y TO X +
      3,Y + 4 TO X - 3,Y +
      4 TO X - 3,Y: PLOT X -
      4,Y + 6 TO X + 4,Y +
      6 TO X + 4,Y + 9 TO X
       - 4,Y + 9 TO X - 4,Y
       + 6
1150  RETURN
```

Программа "Болгарский вальс"

```
5  HOME
10  RIBBON= 3
17  VTAB (12): HTAB (9): PRINT
     "БОЛГАРСКИЙ ВАЛЬС"
20  DIM W(140),P(140)
30  DATA   173,48,192,136,
     208,5,206,1,3,240,9,2
     02,208,245,174,0,3,76
     ,2,3,96,,0
40  FOR X = 770 TO 792
50  READ Y
60  POKE X,Y
70  NEXT X
80  REM ВЫСОТА
90  DATA 128,102,85,64,51,
     42,45,42,45,42,64,51
100  DATA  85,85,76,68,64,
     57,51,57,48,51,57,64
110  DATA  68,42,45,48,51,
     54,57,60,64,68,72,76
120  DATA 76,85,92,85,76,8
     5,68,76,85,96,102,114
130  DATA  128,102,85,64,5
     1,42,45,42,45,46,64,5
     1
140  DATA 85,85,76,68,64,5
     7,51,57,48,51,57,64
150  DATA 68,42,45,48,51,5
     4,57,60,64,68,72,76
160  DATA 76,85,92,85,76,8
     5,68,76,85,96,102,114

170  DATA  128,102,85,64,5
     1,42,48,57,68,85,96,1
     14
180  DATA 128,102,85,64,51
     ,42,48,57,68,85,96,11
     4
190  DATA 128,102,85,64,51
     ,42,32,64,42,85,51,10
     2,64,1,25,1,64
200  REM ПРОДОЛЖИТЕЛЬНОСТЬ
210  DATA  50,50,50,50,50,
     50,50,50,50,50,50,50,
     50,50,50,50,50,50,50,
     50
220  DATA 50,50,50,50,50,5
     0,50,50,50,50,50,50,5
     0,50,50,50,50,50,50,5
     0
230  DATA  50,50,50,50,50,
     50,50,50,50,50,50,50,
     50,50,50,50,50,50,50,
     50
240  DATA      50,50,50,50,
     50,50,50,50,50,50,50,
     50,50,50,50,50,50,50,
     50,50
250  DATA    50,50,50,50,5
     0,50,50,50,50,50,50,5
     0,50,50,50,50,50,50,5
     0,50
253  DATA      50,50,50,50,
     50,50,50,50,50,50,50,
     50,50,50,50,50,50,50,
     50,50
256  DATA 50,50,50,50,50,5
     0,50,50,50,50,50,50,1
     80,100,180,100,254
260  FOR N = 1 TO 137
270  READ W(N)
280  NEXT N
290  FOR N = 1 TO 137
300  READ P(N)
310  NEXT N
320  FOR N = 1 TO 137
330  POKE 768,W(N)
340  POKE 769,P(N)
350  CALL 770
360  NEXT N
365  WAIT 1000,255
370  RUN
```

Программа диалога с ДОС

```
5 D$ =  CHR$ (4): PRINT  CHR$
    (4)"CATALOG":B =  PEEK
    (37) - 2: IF B > 22 THEN
    B = 22
90  PRINT  CHR$ (4);"MAXFI
    LES 1"
95  ONERR  GOTO 90
100  TEXT= 10: PRINT  CHR$
    (12)
105 D$ =  CHR$ (4): PRINT
     CHR$ (4)"CATALOG":B =
     PEEK (37) - 2: IF B >
    22 THEN B = 22
110 T = 0:H = 4: FOR V = 4
     TO 30: GOSUB 1000: IF
    E <  > 32 THEN  POKE
    P - 2,219: POKE P,T +
    193: POKE P + 2,221:T
     = T + 1:S = V
115  RIBBON= 2
120  NEXT V: VTAB 32:A$ =
    "RUN LOAD1 LOCK2 UNLO
    CK3 DEL4..."
130 B$ = "RUN": HTAB 1: PRINT
     LEFT$ (A$,38);:A$ =
     MID$ (A$,2) +  LEFT$
    (A$,1):K =  PEEK ( -
    16384): IF K < 128 THEN
     FOR K = 1 TO 75: NEXT
    K:K =  FRE (0): GOTO
    130
140  POKE  - 16368,0:K = K
     - 176: IF K < 1 OR K
     > 6 THEN 300
200  HTAB 1: CALL  - 868: IF
    K = 6 THEN  END
210  PRINT "PRESS'LETTER'Y
    OU WISH TO ";: IF K =
    1 THEN B$ = "LOAD"
220  IF K = 2 THEN B$ = "L
    OCK"
230  IF K = 3 THEN B$ = "U
    NLOCK"
240  IF K = 4 THEN B$ = "D
    ELETE": FLASH
245  IF K = 5 THEN  NORMAL
    : RUN
250  PRINT B$;: GET K$:K =
     ASC (K$) - 48
300  IF K < 17 OR K > 17 +
    16 THEN 130
305  NORMAL
310 H = 1:V = S - T + K -
    16: GOSUB 1000: IF F =
    194 AND (B$ = "RUN" OR
    B$ = "LOAD") THEN B$ =
    "B" + B$
320  FOR H = 8 TO 58 STEP
    2: GOSUB 1000:B$ = B$
     +  CHR$ (E): NEXT H:
     HTAB 1: CALL  - 868:
     PRINT B$: PRINT D$;B
    $: GOTO 100
1000 C =  INT (V / 8):Q =
    V:P = 20480 + 64 * Q +
    64 * 1 + H:P = P + 4:
    E =  PEEK (P):F =  PEEK
    (P - 3);
1005  POKE (P + 1),$29
1010  RETURN
```

Программа чтения-записи сектора ГМД (DRW)

```
10  PRINT  CHR$ (4);"BLOAD
    DSK"
20  TEXT= 15: INVERSE : HOME

30  INVERSE
40  INPUT "V?";V
50  INPUT "R,W?";N
60  INPUT "T?";T
70  INPUT "S?";S
200  POKE 3712,$30
210  POKE 3713,1
220  POKE 3714,V
230  POKE 3715,T
240  POKE 3716,S
250  POKE 3717,N
260  IF N = 2 THEN 280
270  FOR I = $2000 TO $20F
    F
271  POKE I,255
272  NEXT
280  CALL $F00
290  IF N = 2 THEN 30
300  FOR A = $2000 TO $20F
    F
310 C =  PEEK (A)
320  IF C <  = 31 THEN 330
321  IF C >  = 32 AND C <
     = 127 THEN 340
322  IF C >  = 128 AND C <
     = 159 THEN 350
323  IF C >  = 160 THEN 36
     0

330 C = C + 64: NORMAL : RIBBON=
    1: GOTO 370
340  NORMAL : RIBBON= 7: GOTO
     370
350 C = C - 64: INVERSE : RIBBON=
    1: GOTO 370
360 C = C - 128: INVERSE :
     RIBBON= 7: GOTO 370
370  PRINT  CHR$ (C);
380  NEXT
390  PRINT : PRINT
399  GOTO 30
```

Программа "CLOCK"

```
7000-  A9 00     LDA  #$00
7002-  8D 80 70  STA  $7080
7005-  8D 81 70  STA  $7081
7008-  8D 82 70  STA  $7082
700B-  8D 83 70  STA  $7083
700E-  A9 4C     LDA  #$4C
7010-  8D FB 03  STA  $03FB
7013-  A9 32     LDA  #$32
7015-  8D FC 03  STA  $03FC
7018-  A9 70     LDA  #$70
701A-  8D FD 03  STA  $03FD
701D-  8D 50 C0  STA  $C050
7020-  A9 0E     LDA  #$0E
7022-  8D F2 03  STA  $03F2
7025-  A9 70     LDA  #$70
7027-  8D F3 03  STA  $03F3
702A-  49 A5     EOR  #$A5
702C-  8D F4 03  STA  $03F4
702F-  4C CF D7  JMP  $D7CF
7032-  48        PHA
7033-  EE 80 70  INC  $7080
7036-  AD 80 70  LDA  $7080
7039-  C9 32     CMP  #$32
703B-  D0 35     BNE  $7072
703D-  A9 00     LDA  #$00
703F-  8D 80 70  STA  $7080
7042-  EE 81 70  INC  $7081
7045-  AD 81 70  LDA  $7081
       *L
7048-  C9 3C     CMP  #$3C
704A-  D0 26     BNE  $7072
704C-  A9 00     LDA  #$00
704E-  8D 81 70  STA  $7081
7051-  EE 82 70  INC  $7082
7054-  AD 82 70  LDA  $7082
7057-  C9 3C     CMP  #$3C
7059-  D0 17     BNE  $7072
705B-  A9 00     LDA  #$00
705D-  8D 82 70  STA  $7082
7060-  EE 83 70  INC  $7083
7063-  AD 83 70  LDA  $7083
7066-  C9 18     CMP  #$18
7068-  D0 08     BNE  $7072
706A-  A9 00     LDA  #$00
706C-  8D 83 70  STA  $7083
706F-  20 B4 FC  JSR  $FCB4
7072-  68        PLA
7073-  40        RTI
7074-  00        BRK
7075-  00        BRK
7076-  FF        ???
7077-  FF        ???
7078-  00        BRK
7079-  00        BRK
707A-  FF        ???
707B-  FF        ???
707C-  00        BRK
```

Программа "Часы"

```
1  DIM O%(6)
5  INVERSE
20  HOME
25  VTAB 6
30  PRINT "ВВЕДИТЕ ТЕКУЩЕЕ
      ВРЕМЯ"
40  PRINT
50  INPUT "(ЧЧ,ММ,СС) "; HH
    %, MM%, SS%
54  GR: 3: FOR I = 1 TO 6:
    O%(I) = 77: NEXT
55  COLOR= 4
57  FOR I = 00 TO 20: PLOT
    0,I TO 64,I: NEXT
58  FOR I = 45 TO 64: PLOT
    0,I TO 64,I: NEXT
60  FOR N =  HH% TO 23:A =
    N
65 H = 5:B% = 2:V = 30: GOSUB
    2000
70  FOR M =  MM% TO 59:A =
    M
75 H = 25:B% = 4:V = 30: GOSUB
    2000
80  FOR S =  SS% TO 59:A =
    S
81 H = 45:B% = 6:V = 30: GOSUB
    2000
82  HTAB 16
90  FOR  IN = 1 TO 600: NEXT

110  NEXT : SS% = 0
120  NEXT : MM% = 0
130  NEXT : HH% = 0
140  GOTO 60
2000  COLOR= 00
2002  NC% = (N + M + S) /
    7: NC% = N + M + S -
    NC% * 7 + 1: NC% =  INT
    ( RND (999) * 7 + 1):
    COLOR=  NC%
2003  PLOT 21,31 TO 21,32:
    PLOT 21,35 TO 21,36:
    COLOR= 8 -  NC%
2004  PLOT 41,31 TO 41,32:
    PLOT 41,35 TO 41,36:
    COLOR= 0
2005 I% = A / 10:J% = A -
    10 * I%
2010 H = H + 6:C = O%(B%):
    GOSUB 3630
2020  COLOR=  NC%:C = J%:O
    %(B%) = J%: GOSUB 363
    0
```

```
2030  IF O%(B% - 1) = I% THEN
      COLOR= 0:V = 65:H =
      H - 6: GOSUB 3630: GOSUB
      3630: GOTO 2050
2039 C = O%(B% - 1)
2040 O%(B% - 1) = I%: COLOR=
      0:H = H - 6:V = 30: GOSUB
      3630: COLOR=  NC%:C =
      I%: GOSUB 3630
2050  RETURN
3630  ON C GOTO 3680,3710,
      3750,3790,3820,3850,3
      880,3910,3950
3640  PLOT H + 4,V TO H +
      2,V TO H + 1,V + 1 TO
      H + 1,V + 5
3650  PLOT H + 2,V + 6 TO
      H + 4,V + 6 TO H + 5,
      V + 5 TO H + 5,V + 1
3660  PLOT H + 4,V + 2 TO
      H + 2,V + 4
3670  RETURN
3680  PLOT H + 2,V + 1 TO
      H + 3,V TO H + 3,V +
      6
3690  PLOT H + 2,V + 6 TO
      H + 4,V + 6
3700  RETURN
3710  PLOT H + 1,V + 1 TO
      H + 2,V TO H + 4,V TO
      H + 5,V + 1 TO H + 5,
      V + 2
3720  PLOT H + 4,V + 3 TO
      H + 3,V + 3 TO H + 1,
      V + 5
3730  PLOT H + 1,V + 6 TO
      H + 5,V + 6
3740  RETURN
3750  PLOT H + 1,V TO H +
      5,V TO H + 5,V + 1 TO
      H + 3,V + 3
3760  PLOT H + 4,V + 3 TO
      H + 5,V + 4 TO H + 5,
      V + 5
3770  PLOT H + 4,V + 6 TO
      H + 2,V + 6 TO H + 1,
      V + 5
3780  RETURN
3790  PLOT H + 4,V + 6 TO
      H + 4,V TO H + 1,V +
      3
3800  PLOT H + 1,V + 4 TO
      H + 5,V + 4
3810  RETURN
3820  PLOT H + 5,V TO H +
      1,V TO H + 1,V + 2 TO
      H + 4,V + 2
3830  PLOT H + 5,V + 3 TO
      H + 5,V + 5 TO H + 4,
      V + 6 TO H + 4,V + 6 TO
      H + 2,V + 6 TO H + 1,
      V + 5
3840  RETURN
3850  PLOT H + 5,V TO H +
      3,V TO H + 1,V + 2 TO
      H + 1,V + 5
3860  PLOT H + 2,V + 6 TO
      H + 4,V + 6 TO H + 5,
      V + 5 TO H + 5,V + 4 TO
      H + 4,V + 3 TO H + 2,
      V + 3
3870  RETURN
3880  PLOT H + 1,V TO H +
      5,V
3890  PLOT H + 5,V + 1 TO
      H + 2,V + 4 TO H + 2,
      V + 6
3900  RETURN
3910  PLOT H + 1,V + 2 TO
      H + 1,V + 1 TO H + 2,
      V TO H + 4,V TO H + 5
      ,V + 1 TO H + 5,V + 2

3920  PLOT H + 4,V + 3 TO
      H + 2,V + 3 TO H + 1,
      V + 4 TO H + 1,V + 5
3930  PLOT H + 2,V + 6 TO
      H + 4,V + 6 TO H + 5,
      V + 5 TO H + 5,V + 4
3940  RETURN
3950  PLOT H + 4,V + 3 TO
      H + 2,V + 3 TO H + 1,
      V + 2 TO H + 1,V + 1 TO
      H + 2,V TO H + 4,V
3960  PLOT H + 5,V + 1 TO
      H + 5,V + 4 TO H + 3,
      V + 6 TO H + 1,V + 6
3970  RETURN
```

Программа совмещения текста и графики (SOVM)

```
10  HOME
20  * $500:
30     !  VKL:STA$C200
       ! LDA ETALON + 1
       ! STA$FFFA
       ! LDA ETALON + 2
40     ! STA$FFFB
       ! LDA ETALON + 4
       ! STA$FFFE
       ! LDA ETALON + 5
50     ! STA$FFFF
       ! STA$C220
       ! CLI
       ! STA$C050
60     ! LDA#25
       ! STA34
       ! STA37
       ! LDA32
70     ! STA36
       ! RTS
       !  ETALON:JMP NMI
       ! JMP IRQ
80     ! NOP
       ! NOP
       ! NOP
       !  PPA:BRK
90     !  PPX:BRK
       !  PQA:BRK
       !  PQX:BRK
       !  STC:$00
100    !  SET:$00
       !  SKA:$31
       !  SKB:$3E
       !  SKC:$3C
110    !  NMI:STA$C040
       ! STA PPA
       ! STX PPX
       ! LDA SET
120    ! STA STC
       ! INC STC
       ! LDX SKA
       ! LDA$C700,X
130    ! LDA#25
       ! STA34
       ! LDX PPX
       ! LDA PPA
140    ! STA$C050
       ! RTI
       !  IRQ:STA PQA
       ! STX PQX
150    ! PLA
       ! PHA
       ! ASL
       ! ASL
160    ! ASL
       ! BPL MTA
       ! LDA PQA
       ! JMP$E53D
170    !  MTA:DEC STC
       ! BNE MTB
       ! LDX SKB
       ! LDA$C700,X
180    ! INC STC
       !  MTB:LDX PQX
       ! LDA PQA
       ! RTI
190    !  VIKL:STA$C040
       ! SEI
       ! LDX SKC
       ! LDA$C700,X
200    ! LDA#0
       ! STA34
       ! JMP$FC3D
       ! :
210  REM *SET-КОЛИЧЕСТВО
220  REM *СТРОК,SKA И SKB
230  REM *СТРАНИЦЫ ГРАФИК
240  REM *ИЛИ ТЕКСТА.CALL
250  REM *VKL:ВКЛЮЧЕНИЕ
260  REM *CALL VIKL:ВЫ-
270  REM *КЛЮЧЕНИЕ.
280  REM *СТРАНИЦЫ:
290  REM *GR:=(N*4)
300  REM *MGR:=((N*16)+1)
310  REM *TEXT:=((N*4)+2)
320  REM *HGR:=((N*16)+3)
330  REM *ПРИМЕР:
340  REM *POKE SKA,N
```

Программа регулирования частоты вращения ГМД (SKOR)

```
5  LOMEM: $2000: HIMEM: $3
     000: PRINT : REM  CHR$
     (4);"BLOADDSBIN"
10  GOSUB 3000
20  CV = 25
50  GOSUB 2000
100  POKE 36,0: VTAB  CV
102  PRINT  CHR$ ($9D);
105  PRINT " SLOT - ";
110  GOSUB 1000
115 S = X
120  IF S < 1 OR S > 7 THEN
     100
130  IF  INT (S) <  > S THEN
     100
140  POKE 36,32: VTAB  CV:
      PRINT  CHR$ ($9D);
145  PRINT "DRIVE - ";
150  GOSUB 1000:D = X
160  IF D < 1 OR D > 2 THEN
     140
180  POKE 251,S * 16: POKE
     252,D - 1
185  PRINT : PRINT
190  PRINT "DISK SPEED =
           MS"
200  CALL 3328
220  IF  PEEK (255) = 0 THEN
     10
230  GOSUB 3000
240  PRINT : PRINT " ЪDISK
     ETTE IS WRITE PROTECT
     ED!!"
250  PRINT "PRESS RETURN T
     O CONTINUE."
260  IF  PEEK ( - 16384) <
     127 THEN 260
270  POKE  - 16368,0: GOTO
     10
300  REM 123456
1000 X =  PEEK ( - 16384):
      IF X < 127 THEN 1000

1010  POKE  - 16368,0
1013  IF X = $9B THEN  HOME
     : NEW : END
1015  IF X > $9F THEN  PRINT
      CHR$ (X)
1017  IF X < $B0 OR X > $B
     9 THEN X = $80
1020 X =  VAL ( CHR$ (X -
     $80))
```

```
1030  RETURN
2000  PRINT " ЭТА ПРОГРАММ
      А ИЗМЕРЯЕТ СКОРОСТЬ";

2005  PRINT
2010  PRINT "ВРАЩЕНИЯ ДИСК
      ОВОДА. СКОРОСТЬ"
2015  PRINT
2020  PRINT "ИНДИЦИРУЕТСЯ
      В МИЛЛИСЕКУНДАХ И"
2025  PRINT
2030  PRINT "ДОЛЖНА НАХОДИ
      ТЬСЯ В ПРЕДЕЛАХ ОТ"
2035  PRINT
2040  PRINT "197.00 ДО 203
      .00 МСЕК."
2045  PRINT
2050  PRINT "  ДОПУСТИМЫ К
      ОЛЕБАНИЯ СКОРОСТИ"
2055  PRINT
2060  PRINT "ВЕЛИЧИНОЙ ДО
      1 МСЕК."
2065  PRINT
2070  INVERSE : PRINT " ВН
      ИМАНИЕ!! ": NORMAL :

2200  RETURN
3000  PRINT : HOME : PRINT

3010  PRINT  TAB( 10);"ПРО
      ГРАММА 'СКОРОСТЬ'"
3020  PRINT : RETURN
```

Программа "Личный секретарь"

```
10  TEXT= 62: HOME
100  GOSUB 1520
110  CLEAR : GOSUB 1210
120  PRINT : PRINT D$;"OPE
     N  I$"
121  PRINT : PRINT D$;"REA
     D  I$"
122  FOR X = 1 TO Q: INPUT
     I$(X): NEXT X
123  PRINT : PRINT D$;"CLO
     SE"
130  PRINT : PRINT D$;"OPE
     N  A$"
131  PRINT : PRINT D$;"REA
     D  A$"
132  FOR X = 1 TO Q: INPUT
     A$(X): NEXT X
133  PRINT : PRINT D$;"CLO
     SE"
140  PRINT : PRINT D$;"OPE
     N  T$"
141  PRINT : PRINT D$;"REA
     D  T$"
142  FOR X = 1 TO Q: INPUT
     T$(X): NEXT X
143  PRINT : PRINT D$;"CLO
     SE"
150  PRINT : PRINT D$;"OPE
     N  P$"
151  PRINT : PRINT D$;"REA
     D  P$"
152  FOR X = 1 TO Q: INPUT
     P$(X): NEXT X
153  PRINT : PRINT D$;"CLO
     SE"
160  PRINT : PRINT D$;"OPE
     N  G$"
161  PRINT : PRINT D$;"REA
     D  G$"
162  FOR X = 1 TO Q: INPUT
     G$(X): NEXT X
163  PRINT : PRINT D$;"CLO
     SE"
170  PRINT : PRINT D$;"OPE
     N  K"
171  PRINT : PRINT D$;"REA
     D  K"
172  FOR X = 1 TO Q: INPUT
     K(X): NEXT X
173  PRINT : PRINT D$;"CLO
     SE"
180  PRINT : PRINT D$;"OPE
     N  R"
181  PRINT : PRINT D$;"REA
     D  R"
182  FOR X = 1 TO Q: FOR Y
      = 1 TO 3: INPUT R(X,
     Y): NEXT : NEXT
183  PRINT : PRINT D$;"CLO
     SE"
190  PRINT : PRINT D$;"OPE
     N  C"
191  PRINT : PRINT D$;"REA
     D  C"
192  FOR X = 1 TO Q + 1: INPUT
     C(X): NEXT X
193  PRINT : PRINT D$;"CLO
     SE"
260  TEXT= 62: HOME
270  HTAB 12: VTAB 2: PRINT
     "K A P T O T E K A"
280  PRINT : PRINT  TAB( 9
     )"ПРОГРАММА СОЗДАНИЯ
     И ВЕДЕНИЯ ЛИЧНОГО АРХ
     ИВА"
290  PRINT : PRINT
300  PRINT "1) ПОЛНЫЙ ПРОС
     МОТР.................
     ....................
     ..1"
310  PRINT : PRINT "2) ТЕЛ
     ЕФОННЫЙ УКАЗАТЕЛЬ ...
     ....................
     ..........2"
```

```
320  PRINT : PRINT "3) ПОИ
     СК ПО Ф.И.О. ........
     ..................
     ..........3"
330  PRINT : PRINT "4) ПОИ
     СК ПО МЕСЯЦУ РОЖДЕНИЯ
      ..................
     ..........4"
340  PRINT : PRINT "5) ПОИ
     СК ПО НОМЕРУ ЛИСТА ..
     ..................
     ..........5"
350  PRINT : PRINT "6) ВВО
     Д ДАННЫХ В АРХИВ ....
     ..................
     ..........6"
360  PRINT : PRINT "7) ЗАП
     ИСЬ АРХИВА НА ДИСКЕТУ
      ..................
     ..........7"
370  PRINT : PRINT "8) ВЫХ
     ОД ИЗ ПРОГРАММЫ .....
     ..................
     ..........8"
380  HTAB 8: VTAB 29: PRINT
     "ВЫБЕРИТЕ РЕЖИМ РАБОТ
     Ы (1 - 8)";
390  GET V$: IF  LEN (V$) >
     1 OR  VAL (V$) < 1 OR
      VAL (V$) > 8 THEN 38
     0
400 V =  VAL (V$): ON V GOTO
     410,600,700,780,860,9
     20,1270,1530
410  HOME
420  GOSUB 1470
430  HOME :A = 0
440  FOR X = 0 TO Q
450  GOSUB 490
460  IF  INT (X / 10) = X /
     (10) THEN  INPUT "ПРО
     ДОЛЖИТЬ? ДА(0) НЕТ(1)
     ";A
470  IF A = 1 THEN  GOTO 2
     60: IF X = Q THEN 260

480  NEXT X
490  PRINT "==============
     ====================
     ================"
500  PRINT "ЛИСТ N ";X
510  PRINT "==============
     ====================
     ================"
520  PRINT "ПОЧТ.ИНДЕКС...
     ";K(X)
530  PRINT "ГОРОД.........
     ";G$(X)
540  PRINT "АДРЕС.........
     ";A$(X)
550  PRINT "Ф.И.О.........
     ";I$(X)
560  PRINT "ТЕЛЕФОН.....
     ";T$(X)
570  PRINT "ДЕНЬ РОЖДЕНИЯ.
     ";R(X,1);"/";R(X,2);"
     /";R(X,3)
575  PRINT "ДОП.ДАННЫЕ ...
     ";P$(X)
580  PRINT "==============
     ====================
     ================": PRINT

590  RETURN
600  HOME
610  GOSUB 1470
620  HOME :A$ = "0"
630  FOR X = 0 TO Q
640  IF  INT (X / 12) = X /
     12 THEN  PRINT "ПРОДО
     ЛЖИТЬ? ДА(0) НЕТ(1)";
     : INPUT A$
650  IF A$ = "1" THEN  GOTO
     260
660  PRINT X;")";I$(X);
670  PRINT "       ";T$(X)
680  IF X = Q THEN  GOTO 2
     60
690  NEXT X
700  HOME
710 X$ = "КАКОЕ ИМЯ": GOSUB
     1510
720  INPUT J$: ONERR  GOTO
     260
730 L =  LEN (J$)
740  FOR X = 1 TO Q
750  IF J$ =  LEFT$ (I$(X)
     ,L) THEN  GOSUB 490
760  NEXT
770  GET A$: GOTO 260
780  HOME
790 X$ = "КАКОЙ МЕСЯЦ": GOSUB
     1510
800  INPUT P: ONERR  GOTO
     790
810  IF P < 1 OR P > 12 THEN
     780
820  FOR X = 1 TO Q
830  IF P = R(X,2) THEN  GOSUB
     490
840  NEXT
850  GET S$: HOME : GOTO 2
     60
860  HOME
870 X$ = "КАКОЙ НОМЕР": GOSUB
     1510
880  INPUT X: ONERR  GOTO
     870
890  IF X < 1 OR X > Q THEN
     870
900  GOSUB 490
910  GET S$: GOTO 260
920  HOME : GOTO 1090
930 M = C(Q + 1)
```

```
940  VTAB 13: PRINT "ЛИСТ
     N ";M;" ВВОД НОВЫХ ДА
     ННЫХ": VTAB 14: PRINT
     "XXXXXXXXXXXXXXXXXXX
     XXXXXXXXXXXXXXXXXXXX
     XXXXXXXXX"
950  VTAB 15: PRINT "ПОЧТ.
     ИНДЕКС...";: INPUT K(
     M)
960  VTAB 16: PRINT "ГОРОД
     .........";: INPUT G$
     (M)
970  VTAB 17: PRINT "АДРЕС
     .........";: INPUT A$
     (M)
980  VTAB 18: PRINT "Ф.И.О
     .....";: INPUT I$
     (M)
990  VTAB 19: PRINT "ТЕЛЕФ
     ОН.....";: INPUT T$
     (M)
1000  VTAB 20: PRINT "ДЕНЬ
      РОЖДЕНИЯ.";: INPUT R
     (M,1)
1010  VTAB 21: PRINT "МЕС.
      РОЖДЕНИЯ.";: INPUT R
     (M,2)
1020  VTAB 22: PRINT "ГОД
     РОЖДЕНИЯ .";: INPUT R
     (M,3)
1030  VTAB 23: PRINT "ДОП.
      ДАННЫЕ ..";: INPUT P
     $(M)
1040  VTAB 25: PRINT "ВЕРН
     Ы-ЛИ ЭТИ ДАННЫЕ ?  0
     - НЕТ  1 -- ДА";
1050 Z = 0: INPUT Z
1060  IF Z = 1 AND O = 1 THEN
     C(Q + 1) = C(Q + 1) +
     1: GOTO 260
1070  IF Z = 0 THEN  FOR U
      = 15 TO 25: VTAB U: PRINT "

                          ": NEXT
     U: GOTO 940
1080  GOTO 260
1085  END
1090  HOME
1100  VTAB 5: PRINT "ВВОД
     НОВЫХ ДАННЫХ"
1110  VTAB 11: PRINT "ИЗМЕ
     НЕНИЯ В СТАРОМ ЛИСТЕ
     .........0"
1120  PRINT : PRINT "ОТКРЫ
     ТИЕ НОВОГО ЛИСТА ....
     .........1"
1130  GET H$: IF  LEN (H$)
      > 1 OR  VAL (H$) > 1
      THEN 1130
1140 O =  VAL (H$)
1150  IF O = 1 THEN  HOME
     : GOTO 930
1160  HOME
1170  VTAB 13: PRINT " С К
     АКОГО НОМЕРА ЛИСТА ИС
     КАТЬ (1-";C(Q + 1) -
     1;")";: INPUT N
1180 N =  INT (N)
1190  IF N < C(Q + 1) AND
     N > 0 THEN X = N: HOME
     : GOSUB 490:M = N: GOTO
     940
1200  GOTO 260
1210 Q = 450
1220  DIM I$(Q): DIM A$(Q)
     : DIM T$(Q): DIM K(Q)
     : DIM R(Q,3):' DIM P$(
     Q)
1230  DIM G$(Q)
1240  DIM C(Q + 1):C(Q + 1
     ) = 1
1250 D$ =  CHR$ (4)
1260  RETURN
1270  HOME : VTAB 12: PRINT
     "СТИРАНИЕ АРХИВА.....
     .........0"
1280  VTAB 13: PRINT "ЗАПИ
     СЬ   АРХИВА.........
     ...1"
1290  INPUT A: IF A = 1 THEN
     1340
1300  GOSUB 1520
1310  PRINT : PRINT D$;"DE
     LETEI$"
1311  PRINT : PRINT D$;"DE
     LETEA$"
1312  PRINT : PRINT D$;"DE
     LETET$"
1313  PRINT : PRINT D$;"DE
     LETEG$"
1314  PRINT : PRINT D$;"DE
     LETEP$"
1315  PRINT : PRINT D$;"DE
     LETEK "
1316  PRINT : PRINT D$;"DE
     LETER "
1317  PRINT : PRINT D$;"DE
     LETEC "
1340  GOSUB 1520
1350  PRINT : PRINT D$;"OP
     EN  I$"
1351  PRINT : PRINT D$;"WR
     ITE I$"
1352  FOR X = 1 TO Q: PRINT
     I$(X): NEXT X
1353  PRINT : PRINT D$;"CL
     OSE"
1360  PRINT : PRINT D$;"OP
     EN  A$"
1361  PRINT : PRINT D$;"WR
     ITE A$"
1362  FOR X = 1 TO Q: PRINT
     A$(X): NEXT X
1363  PRINT : PRINT D$;"CL
     OSE"
```

```
1370  PRINT : PRINT D$;"OP
      EN  T$"
1371  PRINT : PRINT D$;"WR
      ITE T$"
1372  FOR X = 1 TO Q: PRINT
      T$(X): NEXT X
1373  PRINT : PRINT D$;"CL
      OSE"
1380  PRINT : PRINT D$;"OP
      EN  P$"
1381  PRINT : PRINT D$;"WR
      ITE P$"
1382  FOR X = 1 TO Q: PRINT
      P$(X): NEXT X
1383  PRINT : PRINT D$;"CL
      OSE"
1390  PRINT : PRINT D$;"OP
      EN  G$"
1391  PRINT : PRINT D$;"WR
      ITE G$"
1392  FOR X = 1 TO Q: PRINT
      G$(X): NEXT X
1393  PRINT : PRINT D$;"CL
      OSE"
1400  PRINT : PRINT D$;"OP
      EN  K"
1401  PRINT : PRINT D$;"WR
      ITE K"
1402  FOR X = 1 TO Q: PRINT
      K(X): NEXT X
1403  PRINT : PRINT D$;"CL
      OSE"
1410  PRINT : PRINT D$;"OP
      EN  R"
1411  PRINT : PRINT D$;"WR
      ITE R"
1412  FOR X = 1 TO Q: FOR
      Y = 1 TO 3
1413  PRINT R(X,Y)
1414  NEXT Y: NEXT X
1415  PRINT : PRINT D$;"CL
      OSE"
1450  PRINT : PRINT D$;"OP
      EN  C"
1451  PRINT : PRINT D$;"WR
      ITE C"
1452  FOR X = 1 TO Q + 1:
      PRINT C(X): NEXT X
1453  PRINT : PRINT D$;"CL
      OSE"
1460  GOTO 260
1470  HOME
1480  VTAB 11: PRINT "С КА
      КОГО НОМЕРА ХОТИТЕ НА
      ЧАТЬ 1-";C(Q + 1) - 1
      ;" ": INPUT O
1490  IF O < 1 OR O > C(Q
      1) THEN 1470
1500  RETURN
1510  HOME : VTAB 11: PRINT
      "ВВЕДИТЕ ЖЕЛАЕМОЕ ";X
      $: RETURN
1520  HOME : VTAB 12: PRINT
      "ПОДОЖДИТЕ...": RETURN

1530  HOME : CLEAR : END
```

Программа "Полный тест"

```
10  HIMEM: $9600
20  LOMEM: $8000
25  GOTO 1200
30  TEXT= 15: HOME
31 A = 1234567891011
32 B = 123E + 2:C = 0.123
33  RIBBON= 6: PRINT "П.1.2.2": PRINT "_______": PRINT
          : NORMAL
35  PRINT
50  PRINT "A=12345678910ĺ1": PRINT
60  PRINT "B=123E+2": PRINT
70  PRINT "C=0.123": PRINT
71  PRINT
75  PRINT "? A,B,C": PRINT
76  PRINT A,B,C: PRINT
77  VTAB 12: HTAB 18
80  RIBBON= 1: PRINT "ДОЛЖНО БЫТЬ:": PRINT
90  HTAB 18: PRINT 1234567891011
100  HTAB 18: PRINT 123E + 2;: HTAB 26: PRINT 0.123

110  NORMAL
112  PRINT : PRINT
115  RIBBON= 6
120  PRINT "П.1.2.3": PRINT "_______"
125  NORMAL
135  PRINT
140  PRINT "? .0011^12:? 1111^12": PRINT
```

```
145  PRINT : PRINT
150  PRINT .0011 ^ 12: PRINT 1111 ^ 12
160  HTAB 18: VTAB 23
165  RIBBON= 1: PRINT "ДОЛЖНО БЫТЬ:": PRINT
170  HTAB 18: PRINT "3.1384284E-36": HTAB 18: PRINT
       "3.53645967E+36"
175  PRINT : PRINT
180  INVERSE : RIBBON= 1: HTAB 6: VTAB 32: PRINT "Н
       АЖМИ ЛЮБУЮ КЛАВИШУ";: NORMAL
190  WAIT $C000,$80,0
200  POKE $C010,0
205  HOME
207  RIBBON= 6
210  PRINT "П.1.2.6": PRINT "_______": PRINT
212  NORMAL
215  PRINT "TEXT=15:HOME": PRINT "NEW"
220  PRINT "5 A=1": PRINT "10 HOME": PRINT "20 HTAB
       PDL(0)/8": PRINT "30 VTAB PDL(1)/8": PRINT
       "40 IF PEEK($C061)<=127THENA=A+1"
230  PRINT "50 IF PEEK($C062)<=127THENA=A-1": PRINT
       "60 IF A<0 THEN A=9": PRINT "70 IF A>9 TH
       EN A=0": PRINT "80 PRINT A;": PRINT "90 G
       OTO 10"
240  PRINT "RUN";: FLASH : PRINT " ": NORMAL
250  RIBBON= 1: PRINT : PRINT "НА ОЧИЩЕНОМ ЭКРАНЕ П
       ОЯВИТСЯ ";: NORMAL : PRINT "1": RIBBON= 1
       : PRINT
270  PRINT "ВРАЩЕНИЕ РУКОЯТКИ ПУЛЬТА 1 ВЫЗЫ-"
280  PRINT "ВАЕТ ПЕРЕМЕЩЕНИЕ ПО ГОРИЗОНТАЛИ,": PRINT
       "ПУЛЬТА 2 - ПО ВЕРТИКАЛИ. НАЖАТИЕ": PRINT
       "КНОПОК ПУЛЬТА 1 - УВЕЛИЧЕНИЕ, А ": PRINT
       "ПУЛЬТА 2 - УМЕНЬШЕНИЕ ЧИСЛА": PRINT : PRINT
       "ОТ 0 ДО 9.ПОСЛЕ ПРОВЕРКИ НАЖМИ";: NORMAL
       : PRINT " @";
300  PRINT : INVERSE : RIBBON= 1: HTAB 6: PRINT "НА
       ЖМИ КЛАВИШУ ВВОД";: NORMAL
310  WAIT $C000,$80,0
320 A = 1
330  HOME
340  HTAB  PDL (0) / 8: VTAB  PDL (1) / 8
350  IF  PEEK ($C061) <  = 127 THEN A = A + 1
360  IF  PEEK ($C062) <  = 127 THEN A = A - 1
370  IF A < 0 THEN A = 9
380  IF A > 9 THEN A = 0
390  PRINT A;
400  IF  PEEK ($C000) = 192 THEN 420
410  GOTO 330
420  POKE $C010,0
425  HOME
427  RIBBON= 6
430  PRINT "П.1.2.11": PRINT "________": PRINT : PRINT

435  NORMAL
440  PRINT "TEXT=15:HOME:CALL-151": PRINT
450  PRINT "*1000:A9 0 20 F0 FE A9 FF 20 F0": PRINT
       : PRINT "FE AD 0 C0 10 F1 60": PRINT
460  PRINT "*1000G";: FLASH : PRINT " ": NORMAL
465  PRINT
470  NORMAL : RIBBON= 1: PRINT "ПОДКЛЮЧИТЬ ИЗМЕРИТЕ
       ЛЬНЫЙ ВХОД": PRINT
480  PRINT "C1-64 К КОНТАКТУ 3 РАЗЪЕМА O_O": PRINT
       : PRINT "И ИЗМЕРИТЬ АМПЛИТУДУ ИМПУЛЬСОВ."
```

```
      : PRINT : PRINT "ДОЛЖНА БЫТЬ ОТ 0.25 ДО 0
      .50 В."
490   VTAB 32: HTAB 6: INVERSE : RIBBON= 1: PRINT "Н
      АЖМИ КЛАВИШУ ВВОД";: NORMAL
500   WAIT $C000,$80,0: POKE $C010,0
505   HOME
507   GOSUB 1210
510   * $4000:
            ! $A90020F0FEA9FF20F0FEAD00C010F160
            ! :
520   * $4000G
522   POKE $C010,0
525   RIBBON= 6
527   HOME
530   PRINT "П.1.2.12 ";: PRINT "П.1.2.13": PRINT "_
      _______ ________": PRINT
540   NORMAL
550   PRINT "TEXT=15:HOME:CALL-151": PRINT
560   PRINT "*C0D3:80 N C0D9:4E N C0D0:AA AA": PRINT
      "AA N C0D9:5"
570   PRINT : PRINT "*1000:AD D9 C0 4A 90 FA A9 55 8
      D D8 C0 AD 0 C0 10 F0 8D 10 C0 60"
580   PRINT "*1000G";: INVERSE : FLASH : PRINT " ";:
      NORMAL
585   PRINT
590   PRINT : RIBBON= 1
600   PRINT "ПОДКЛЮЧИТЬ С1-64 ПОСЛЕДОВАТЕЛЬНО": PRINT
      "К КОНТАКТАМ А2...А9,В2...В10,": PRINT : PRINT
      "С2...С9 РАЗЪЕМА Х2 ЯЧЕЙКИ Е6 И": PRINT :
      PRINT "ЗАМЕРИТЬ УРОВНИ СИГНАЛОВ '0'И'1'"
610   PRINT "ДОЛЖНО БЫТЬ:": PRINT
615   PRINT "НА В10       -'0' МЕНЬШЕ -11.6 В."
620   PRINT "             -'1' БОЛЬШЕ  11.6 В."
630   PRINT "НА ОСТАЛЬНЫХ-'0' МЕНЬШЕ 0.4 В.": PRINT

640   PRINT "             -'1' БОЛЬШЕ 2.4 В.": PRINT

650   HTAB 6: VTAB 32: INVERSE : RIBBON= 1: PRINT "Н
      АЖМИ КЛАВИШУ ВВОД";: NORMAL
660   WAIT $C000,$80,0
662   POKE $C010,0
665   HOME
667   GOSUB 1210
670   * $C0D3:
            ! $80
            ! :
680   * $C0D9:
            ! $4E
            ! :
690   * $C0D0:
            ! $AAAAAA
            ! :
695   * $C0D9:
            ! $05
            ! :
700   * $6000:
710         ! $ADD9C04A90FAA9558DD8C0AD00C010F08D10C
      060
            ! :
720   * $6000G
730   POKE $C010,0
735   HOME
```

```
740  RIBBON= 6
750  PRINT "П.1.2.14": PRINT "________": NORMAL : PRINT

760  RIBBON= 1
770  PRINT "УСТАНОВИТЬ ПЕРЕМЫЧКУ ФГ 4846 636": PRINT
       "НА РАЗЪЕМ 'ВИДЕО.'."
780  NORMAL : PRINT : PRINT "TEXT=15:HOME:CALL-151"
       : PRINT : PRINT "*2000:FF": PRINT : PRINT
       "*N 2001<2000.3FFFM C713";
790  FLASH : PRINT " ";: NORMAL
795  PRINT
800  PRINT : RIBBON= 1: PRINT "НА ЭКРАНЕ ПОЯВИТСЯ Б
       ЕЛОЕ ПОЛЕ": PRINT : PRINT "ЗАЗЕМЛЯЮЩИЙ ВХ
       ОД С1-64 ПОДКЛЮ-": PRINT : PRINT "ЧАЮТ К
       КОНТАКТУ 2 ПЕРЕМЫЧКИ,"
802  PRINT : PRINT "ИЗМЕРИТЕЛЬНЫЙ - К КОНТАКТУ 3."
804  PRINT : PRINT "ДЛИТЕЛЬНОСТЬ РАЗВЕРТКИ :"
806  PRINT : PRINT "20 МКС/ДЕЛ."
808  PRINT : PRINT "ИЗМЕРЯЮТ АМПЛИТУДУ СИГНАЛА"
809  PRINT : PRINT "ДОЛЖНА БЫТЬ >= 1 В."
810  PRINT : PRINT "ДЛЯ ПРОДОЛЖЕНИЯ НАЖМИ ";: NORMAL
       : PRINT "@"
815  VTAB 32: HTAB 6: INVERSE : RIBBON= 1: PRINT "Н
       АЖМИ КЛАВИШУ ВВОД";: NORMAL
820  WAIT $C000,$80,0
830  * $4000:
       ! $FF
       ! :
840  * $4001#$4000.$5FFFM
845  POKE $C723,0
850  WAIT $C000,$80,0
860  IF  PEEK ($C000) = 192 THEN  POKE $C010,0: GOTO
       880
870  GOTO 850
880  TEXT= 15: HOME
890  RIBBON= 6: PRINT "П.1.2.15": PRINT "________":
       NORMAL : PRINT
910  PRINT "TEXT=15:HOME:CALL-151": PRINT : PRINT "
       *2000:FF N 2001<2000.3FFFM": PRINT : PRINT
       "*C713";: FLASH : PRINT " ";: NORMAL
920  PRINT
925  PRINT
930  RIBBON= 1
940  PRINT "НА ЭКРАНЕ БЕЛОЕ ПОЛЕ.": PRINT "ЗАЗЕМЛЯЮ
       ЩИЙ КОНТАКТ С1-64 ": PRINT "ПОДКЛЮЧАЮТ К
       КОНТАКТУ 2 РАЗЪЕМА": PRINT "RGB. ДЛИТЕЛЬН
       ОСТЬ РАЗВЕРТКИ :"
950  PRINT "0.5 МКС/ДЕЛ. ИЗМЕРИТЬ ДЛИТЕЛЬ-": PRINT
       "НОСТЬ СТРОЧНОГО СИНХРОИМПУЛЬСА": PRINT "
       НА КОНТАКТЕ 5 РАЗЪЕМА RGB"
960  PRINT "ДОЛЖНА БЫТЬ 3.0+-0.1 МКС."
970  PRINT "ДЛИТЕЛЬНОСТЬ РАЗВЕРТКИ :"
980  PRINT "50 МКС/ДЕЛ. ИЗМЕРИТЬ ДЛИТЕЛЬ-": PRINT "
       НОСТЬ КАДРОВОГО СИНХРОИМПУЛЬСА"
990  PRINT "НА КОНТАКТЕ 4 РАЗЪЕМА RGB": PRINT "ДОЛЖ
       НА БЫТЬ 256+-10 МКС."
1000  PRINT "ДЛИТЕЛЬНОСТЬ РАЗВЕРТКИ:"
1010  PRINT "20 МКС/ДЕЛ. ИЗМЕРИТЬ УРОВЕНЬ"
1020  PRINT "СИГНАЛА R,G,B,Y НА КОНТАКТАХ"
1030  PRINT "3,6,1,7 ."
1040  PRINT "ДОЛЖНО БЫТЬ : В СООТВЕТСТВИИ С  ТТЛ."
```

```
1050  PRINT "ДЛЯ ВОЗВРАТА К П.1.2.15 НАЖМИ";: NORMAL
      : PRINT " R";: RIBBON= 1: PRINT " , ДЛЯ П
      РОДОЛЖЕНИЯ -";: NORMAL : PRINT "@ ";: RIBBON=
      1: PRINT "."
1060  INVERSE : RIBBON= 1: VTAB 32: HTAB 6: PRINT "
      НАЖМИ КЛАВИШУ ВВОД";: NORMAL
1070  WAIT $C000,$80,0
1072  POKE $C010,0
1075  POKE $C723,0
1077  WAIT $C000,$80,0
1080  IF  PEEK ($C000) = 210 THEN  POKE $C010,0: GOTO
      880
1090  IF  PEEK ($C000) = 192 THEN  POKE $C010,0: GOTO
      1300
1095  GOTO 1075
1100  TEXT= 15: HOME : VTAB 12: HTAB 12: RIBBON= 1:
       PRINT "СНАЧАЛА Y/N"
1110  WAIT $C000,$80,0
1120  IF  PEEK ($C000) = 217 THEN  POKE $C010,0: GOTO 30
1130  NORMAL : POKE $C010,0: END
1200  HOME : VTAB 12: HTAB 10: RIBBON= 1: PRINT "ТЕ
      СТ ОТК  (С.В.)": FOR I = 1 TO 2000: NEXT  : GOTO 30
1210  HOME : RIBBON= 1: VTAB 12: HTAB 10: PRINT "ПР
      ОВЕРЯЙ...": PRINT : HTAB 4: PRINT "ПОСЛЕ
      ПРОВЕРКИ НАЖМИ ";: NORMAL : PRINT "@"
1220  RETURN
1300  TEXT= 15: HOME : RIBBON= 6: PRINT "ПРОВЕРКА Я
      ЧЕЙКИ ИНТЕРФЕЙСА.": PRINT "_____________
      _____________": PRINT : NORMAL
1304  RIBBON= 1: PRINT "ВСТАВЬТЕ ЗАГЛУШКУ В РАЗЪЕМ"
      : PRINT : PRINT "ПЛАТЫ ИНТЕРФЕЙСА": GOSUB  1420
1305  WAIT $C000,$80,0: POKE $C010,0
1306  HOME : RIBBON= 6: PRINT "ПРОВЕРКА ЯЧЕЙКИ ИНТЕ
      РФЕЙСА.": PRINT "_____________________
      ___": PRINT : NORMAL
1309  RIBBON= 1
1310  PRINT "1. ПРОВЕРКА ИСХОДНОГО СОСТОЯНИЯ:"
1312  NORMAL
1315  * $C0D0:
            ! $0000000000000000
            ! :
1335  ADD = $C0D0
1340  * $4000:
1350        !  PER:LDA ADD
            ! JSR$FDC9
            ! RTS
1351        !  PROD:LDX#$1
            !  M0:LDA ADD,X
            ! JSR$FDC9
            ! LDA#$0
            ! JSR$FDDC
            ! INX
            ! CPX#$8
            ! BNE M0
            ! RTS
            ! :
1355  GOTO 1390
1360  RIBBON= 1: PRINT "ЧИТАЕМ:": NORMAL : PRINT
1362  PRINT "*C0D0": PRINT : PRINT "C0D0- ";
1364  CALL  PER
1365  PRINT : PRINT "*"
1370  CALL  PROD
```

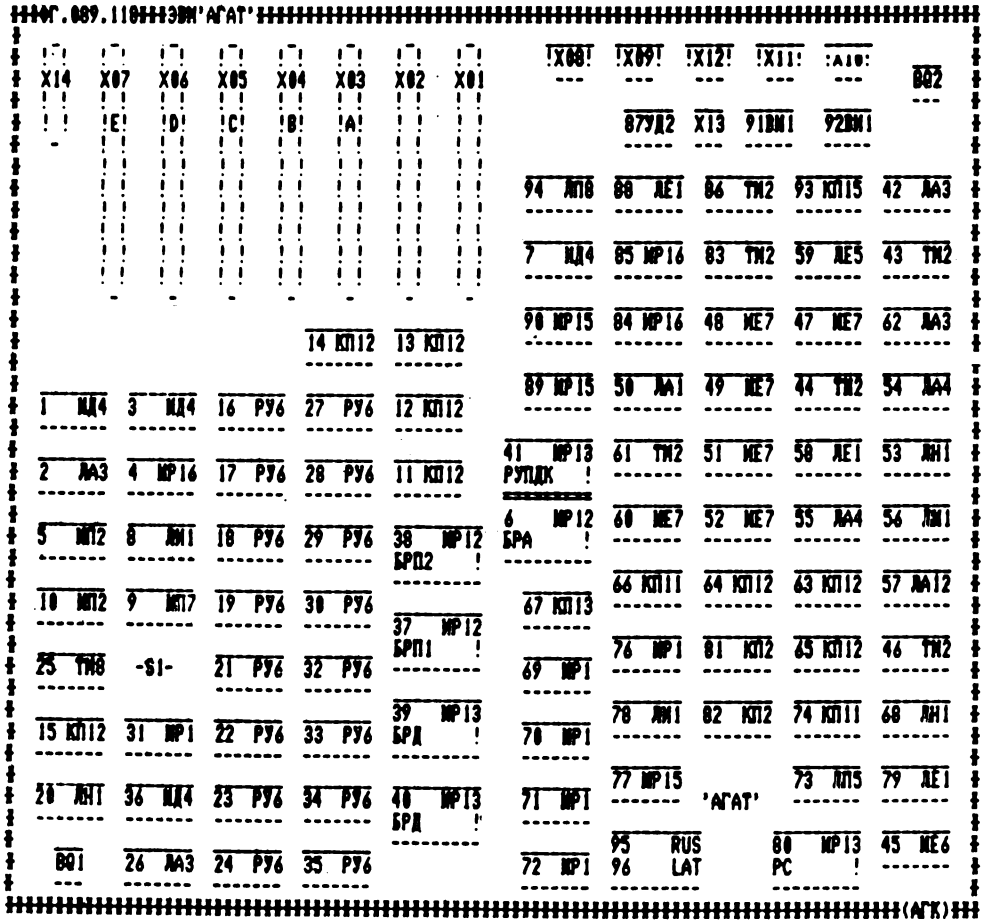
```
1375  PRINT
1380  RETURN
1390  GOSUB 1360
1392  PRINT
1394  RIBBON= 1
1395  PRINT "2. ПРОВЕРКА ПЕРЕДАЧИ ИНФОРМАЦИИ": PRINT
        : PRINT "ИЗ КА В КВ И ИЗ С СТ. В С МЛ.:":
        PRINT
1397  NORMAL
1400  PRINT "*C0D3:83": PRINT : PRINT "*C0D0:55": PRINT
        : PRINT "*C0D2:50"
1405  PRINT
1406  * $C0D3:
            ! $83
            ! :
1407  * $C0D0:
            ! $55
            ! :
1408  * $C0D2:
            ! $50
            ! :
1410  GOSUB 1360
1415  GOSUB 1420
1417  GOTO 1430
1420  VTAB 32: HTAB 6: INVERSE : RIBBON= 1: PRINT "
        НАЖМИ ЛЮБУЮ КЛАВИШУ";: NORMAL : RETURN
1430  WAIT $C000,$80,0: POKE $C010,0
1440  HOME : RIBBON= 1: PRINT "2. (ПРОДОЛЖЕНИЕ) ИЗМ
        ЕНИТЬ": PRINT "ИНФОРМАЦИЮ НА AA:": PRINT

1445  NORMAL
1450  PRINT "*C0D0:AA": PRINT : PRINT "*C0D2:A0": PRINT

1460  * $C0D0:
            ! $AA
            ! :
1470  * $C0D2:
            ! $A0
            ! :
1480  GOSUB 1360
1485  PRINT
1490  RIBBON= 1: PRINT "3. ПРОВЕРКА ПЕРЕДАЧИ ИНФОРМ
        АЦИИ": PRINT "ИЗ КВ В КА И С МЛ. В С СТ."
        : NORMAL : PRINT
1495  PRINT "*C0D3:98": PRINT : PRINT "*C0D1:55": PRINT
        : PRINT "*C0D2:5"
1496  * $C0D3:
            ! $98
            ! :
1497  * $C0D1:
            ! $55
            ! :
1498  * $C0D2:
            ! $05
            ! :
1499  GOSUB 1360: GOSUB 1420: WAIT $C000,$80,0: POKE
        $C010,0
1500  GOTO 1100
```

Программа "Драйвер печати СРА-80"

```
0500-  A2 01     LDX  #$01
0502-  BD 53 AA  LDA  $AA53,X
0505-  9D 57 05  STA  $0557,X
0508-  BD F5 05  LDA  $05F5,X
050B-  9D 2C A2  STA  $A22C,X
050E-  CA        DEX
050F-  10 F1     BPL  $0502
0511-  60        RTS
0512-  29 07     AND  #$07
0514-  D0 03     BNE  $0519
0516-  4C 90 FE  JMP  $FE90
0519-  0A        ASL
051A-  0A        ASL
051B-  0A        ASL
051C-  0A        ASL
051D-  8D F7 05  STA  $05F7
0520-  A9 8D     LDA  #$8D
0522-  20 30 05  JSR  $0530
0525-  A9 30     LDA  #$30
0527-  85 36     STA  $36
0529-  A9 05     LDA  #$05
052B-  85 37     STA  $37
052D-  60        RTS
052E-  00        BRK
052F-  01 8E     ORA  ($8E,X)
0531-  2F        ???
0532-  05 8C     ORA  $8C
0534-  2E 05 48  ROL  $4805
0537-  AD F4 05  LDA  $05F4
053A-  24 32     BIT  $32
053C-  30 01     BMI  $053F
053E-  0A        ASL
053F-  C5 24     CMP  $24
0541-  B0 07     BCS  $054A
0543-  A9 A0     LDA  #$A0
0545-  20 59 05  JSR  $0559
0548-  D0 ED     BNE  $0537
054A-  68        PLA
054B-  48        PHA
054C-  20 59 05  JSR  $0559
054F-  68        PLA
0550-  AE 2F 05  LDX  $052F
0553-  AC 2E 05  LDY  $052E
0556-  4C DF FD  JMP  $FDDF
0559-  EE F4 05  INC  $05F4
055C-  C9 8D     CMP  #$8D
055E-  D0 02     BNE  $0562
0560-  A9 8A     LDA  #$8A
0562-  C9 8A     CMP  #$8A
0564-  D0 05     BNE  $056B
0566-  A2 00     LDX  #$00
0568-  8E F4 05  STX  $05F4
056B-  EA        NOP
056C-  48        PHA
056D-  4A        LSR
056E-  4A        LSR
056F-  4A        LSR
0570-  4A        LSR
0571-  4A        LSR
0572-  AA        TAX
0573-  68        PLA
0574-  5D EC 05  EOR  $05EC,X
0577-  E0 26     CPX  #$26
0579-  48        PHA
057A-  49 FF     EOR  #$FF
057C-  10 0C     BPL  $058A
057E-  29 DF     AND  #$DF
0580-  A8        TAY
0581-  B9 0C 05  LDA  $050C,Y
0584-  E0 04     CPX  #$04
0586-  B0 02     BCS  $058A
0588-  69 21     ADC  #$21
058A-  A8        TAY
058B-  AE F7 05  LDX  $05F7
058E-  BD 82 C0  LDA  $C082,X
0591-  49 50     EOR  #$50
0593-  0A        ASL
0594-  D0 11     BNE  $05A7
0596-  A9 AB     LDA  #$AB
0598-  9D 83 C0  STA  $C083,X
059B-  68        PLA
059C-  48        PHA
059D-  BD 82 C0  LDA  $C082,X
05A0-  10 FB     BPL  $059D
05A2-  68        PLA
05A3-  9D 80 C0  STA  $C080,X
05A6-  60        RTS
05A7-  29 40     AND  #$40
05A9-  F0 F0     BEQ  $059B
05AB-  68        PLA
05AC-  98        TYA
05AD-  9D 80 C0  STA  $C080,X
05B0-  DD 80 C0  CMP  $C080,X
05B3-  F0 08     BEQ  $05BD
05B5-  A9 89     LDA  #$89
05B7-  9D 83 C0  STA  $C083,X
05BA-  20 C3 05  JSR  $05C3
05BD-  BD 82 C0  LDA  $C082,X
05C0-  30 FB     BMI  $05BD
05C2-  0A        ASL
05C3-  9D 81 C0  STA  $C081,X
05C6-  A9 C0     LDA  #$C0
05C8-  9D 81 C0  STA  $C081,X
05CB-  60        RTS
```

Приложение 3
Монтажная схема платы памяти и интерфейса

Программа обучения языку БЕЙСИК



СПИСОК ЛИТЕРАТУРЫ

1. Агаханян Т.М. Интегральные микросхемы. М.: Энергоатомиздат, 1983. 302 с.
2. Гилмор Ч. Введение в микропроцессорную технику. М.: Мир, 1984. 334 с.
3. Зельдин Е.А. Цифровые интегральные микросхемы в информационно-измерительной аппаратуре. Л.: Энергоатомиздат, 1986. 280 с.
4. Зельдин Е.А. Триггеры. М.: Энергоатомиздат, 1986. 280 с.
5. Интегральные микросхемы: Справочник / Б.В. Тарабрин, Л.Ф. Лунин, Ю.Н. Смирнов и др. М.: Радио и связь, 1983. 483 с.
6. Микропроцессоры. В 3 т. / Под ред. Л.Н. Преснухина. М.: Высшая школа, 1986. 278 с.
7. Морер У. Язык Ассемблера для персонального компьютера ЭПЛ: М.: Мир, 1987. 430 с.
8. Пул Л. Работа на персональном компьютере. М.: Мир, 1986, 383 с.
9. Самофалов К.Г. Микропроцессоры. Киев: Техніка, 1986. 278 с. (Б-ка инженера).
10. Уокерли Дж. Архитектура и программирование микро-ЭВМ. Ч. 1. М.: Мир, 1984. 486 с.
11. Шевкопляс Б.В. Микропроцессорные структуры, инженерные решения. М.: Радио и связь, 1986. 300 с.
12. Blanc F., Normant F. L´ORIC et son microproccesseur 6502: langage machine et de ssassemblage de la. ROM. 1984, 35 p.
13. Carr J.J. 68 scientific and enginelring programs for the Apple II and II e, 1984. 101 p.
14. Ericksou J., Cramer W.D. Ine Apple graphics sound book. 1985, 97 p.
15. Finley C.W. Assembly language for the Appllsoft programmer, 1984, 175 p.
16. Phillips G. Apple II free software, 1984, 207 p.
17. Phillips G., Scellato D. Apple II c user guid, 1984, 171 p.
18. Spurlock L. Apple coft encgelopedia, 1984, 300 p.
19. Stannton J. et al. Book of Apple software, 1985, 110 p.
20. Williams K., Kernaghan B. Apple II and II e computer graphies, 1983, 84 p.

СОДЕРЖАНИЕ

Производственное издание

МЫМРИН Михаил Павлович

КОНСТРУКЦИЯ, ПРИМЕНЕНИЕ, ПРОГРАММИРОВАНИЕ
И РЕМОНТ ПЭВМ "АГАТ"

Редактор *Е.В. Григорин-Рябова*
Обложка художника *В. Я. Виганта*
Художественный редактор *А. С. Вершинкин*
Технический редактор *Н. В. Павлова*
Корректор *Т. П. Бровкина*

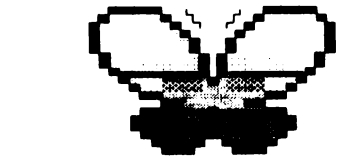
ИБ № 5783

Сдано в набор 30.05.88. Подписано в печать 18.05.90. Т-01802.
Формат 70X100 1/16. Бумага офсетная № 2. Печать офсетная.
Усл.печ.л. 24,70. Усл.кр.-отт. 25,03. Уч.-изд.л. 22,23.
Тираж 87 000 экз. Заказ 341. Цена 2 р.

Ордена Трудового Красного Знамени издательство "Машиностроение",
107076, Москва, Стромынский пер., 4

Отпечатано в Московской типографии № 4 Госкомпечати СССР
129041, Москва, Б. Переяславская ул., 46,
с оригинала-макета, изготовленного в издательстве "Машиностроение"
на персональных ЭВМ по программе "Астра", разработанной НИИЦЭВТ

INFOTRON



НАУЧНО-ПРОИЗВОДСТВЕННОЕ ОБЪЕДИНЕНИЕ "ИНФОТРОН",

созданное в 1988 году, за короткий срок приобрело авторитет в разработке и изготовлении компьютеров и другой пользующейся спросом продукции и услуг. Годовой оборот объединения превышает 4 млн. руб. Располагая штатом высококвалифицированных специалистов, объединение интенсивно завоевывает приоритетные области науки и техники, ориентируясь на выпуск конкурентно-способной продукции.

МЫ ПРЕДЛАГАЕМ СОВЕТСКИМ ПРЕДПРИЯТИЯМ:

компьютеры и компьютерные сети;
видеокомплексы и системы приема спутникового телевидения;
телевизионно-компьютерные устройства;
системы сигнализации на передающих устройствах;
игровые и учебные центры;
диагностическое оборудование;
программные комплексы системного и прикладного назначения;
средства защиты программных комплексов от несанкционированного доступа и дублирования;
русификацию программно-технических средств зарубежных компьютеров;
обучающие и экспертные системы;
системы управления электронными и радиоэлектронными комплексами;
переработку отходов химических производств;
рекламу товаров и услуг советских предприятий на западном рынке;
организацию отдыха и туристических маршрутов внутри страны и за рубежом.

ЗАПАДНЫМ ФИРМАМ ПРЕДЛАГАЕМ:

размещение своих капиталов в наиболее выгодных сферах производства или услуг. Учитываем индивидуальность и возможности фирм.
помещение рекламы и поиск деловых партнеров.
страхование фирм от всех форм краха.
организация отдыха и туризма по наиболее популярным местам Советского Союза

SCIENTIFIC PRODUCTS FIRM "INFOTRON" founded in 1988 in short time asquired authority in designing and production of computers and other products in demand. The firm's capital exceeds 4 million rubls. The firm has highly qualified specialists who intensivily try to "conquer" the important field of science and technology which is oriented to output of concurrent products.

Ukr. SSR, 340025 Donetsk

st. Petroskogo, 66

tel 711626

telex 115178 REIS SU INFOTRON

УВАЖАЕМЫЙ ЧИТАТЕЛЬ!

Издательство **"Машиностроение"** выпустит следующие книги **по вычислительной технике;**

1990 год

Научно-популярная литература

А л е к с а н д р о в В. В., Г о р с к и й Н. Д. **ЭВМ видит мир.** – Л. – 6,5 л.: ил. – (в обл.): 80 к.

Эта книга о том, как и что "видит" современный компьютер. "Научить" машину смотреть несложно – достаточно снабдить ее телекамерой, передающей изображение в память ЭВМ. Видеть – это гораздо больше, чем смотреть; видеть – значит узнавать наблюдаемые объекты, адекватно реагировать на них. Научить машину видеть – задача в тысячу раз более сложная, связанная с такими фундаментальными для человеческого интеллекта понятиями, как "понимать", "узнавать", "объяснять". Машинное видение – одна из основных составляющих частей искусственного интеллекта. И именно так эта тема раскрывается в книге.

Книга рассчитана на учащихся старших классов, ПТУ.

Производственная литература

Белецкий Я. **Турбо Паскаль; Версия 3.0;** Пер. с польск. – М. – 10 л.: ил. – (в обл.): 65 к.

В книге польского автора описывается широко распространенная система программирования Турбо Паскаль (версия 3.0). Основное достоинство этой системы состоит в том, что она объединяет программу "Редактор" и компилятор, что позволяет намного повысить оперативность отладки программ. Собственно о языке Турбо Паскаль рассказывается автором книги с привлечением большого числа примеров.

Для инженеров, использующих в своей работе персональные ЭВМ семейства IBM PC и совместимые с ними модели.

Вострикова З. П., Вострикова О. Ю., Туева С. С. **Программирование на языке БЕЙСИК для персональных ЭВМ ЕС 1840 (ЕС 1841, ЕС 1842).** – М. – 25 л.: ил. – (в пер.): 1 р. 70 к.

Изложены основы программирования на языке БЕЙСИК для персональных ЭВМ ЕС. Основное внимание уделено систематическому описанию средств и методов программирования. Даны необходимые сведения об архитектуре персональных ЭВМ ЕС, показаны методы исследования программных средств для решения различных типов задач инженерно-технических, экономических, логических, информационно-поисковых. Рассмотрена методика построения программ и их отладка. Даны сведения о работе с файлами, звуковой информацией, сложными сегментированными программами, с пакетом прикладных программ "Абак", расширяющими знания о языке БЕЙСИК.

Для пользователей, программистов, инженерно-технических работников, студентов вузов и учащихся техникумов, осваивающих работу на персональных ЭВМ.

М и ш е л ь Ж. **Программируемые контроллеры: Архитектура и применение:** Пер. с фр. – М. – 18 л.: ил. – (в пер.): 1 р. 50 к.

Книга французского автора является существенно переработанным и дополненным изданием книги "Программируемые контроллеры", вышедшей в 1986 г. в издательстве "Машиностроение". В ней изложены основы логических устройств управления, принципы построения программируемых контроллеров (ПК), способы программирования и критерии выбора ПК. Книга дополнена описанием сопряжения ПК с внешними устройствами, более подробно рассмотрено программное обеспечение, методология использования ПК дополнена изучением новых подходов.

Для инженеров-конструкторов и операторов, обслуживающих ПК.

Накопители информации на магнитных дисках емкостью 100 Мбайт/ З. О. Д ж а л и а ш в и л и, А. И. Л у н ь к о, В. Г. М а к у р о ч к и н и др. – Л. – 17 л.: ил. – (в пер.): 1 р. 20 к.

Рассмотрены структура, технические характеристики, конструктивные особенности накопителя ЕС 5066 и устройства управления ЕС 5566, приведены принципы их работы, а также техническая реализация основных узлов. Описан пакет магнитных дисков ЕС 5266. Подробно освещены вопросы технического обслуживания, дан комплекс программ технического обслуживания комплекса в составе ЭВМ.

Для инженеров, обслуживающих внешние запоминающие устройства ЕС ЭВМ и занимающихся проектированием и разработкой устройств внешней памяти ЭВМ.