

МОДУЛЬ ОЗУ

Руководство по эксплуатации

З.065.069 РЭ

1999

СОДЕРЖАНИЕ

	Лист
1. ОБЩИЕ УКАЗАНИЯ	3
2. ТЕХНИЧЕСКИЕ ДАННЫЕ	4
3. КОМПЛЕКТНОСТЬ.	4
4. ТРЕБОВАНИЯ ПО ТЕХНИКЕ БЕЗОПАСНОСТИ	4
5. УСТРОЙСТВО ИЗДЕЛИЯ	5
6. ПОДГОТОВКА К РАБОТЕ.	8
7. ПОРЯДОК РАБОТЫ	10
8. ГАРАНТИЙНЫЕ ОБЯЗАТЕЛЬСТВА. .	43

1. ОБЩИЕ УКАЗАНИЯ

При покупке модуля дополнительного оперативного запоминающего устройства (ОЗУ) для ПЭВМ "Микроша" требуйте проверки его работоспособности. Проверка модуля заключается в однократном выполнении тестовой программы, записанной на кассете, входящей в комплект поставки.

Убедитесь в наличии на гарантийном талоне и отрывном талоне на гарантийное обслуживание штампа магазина, разборчивой подписи или штампа продавца и даты продажи. Проверьте также сохранность пломб модуля и комплект поставки.

После хранения модуля ОЗУ в холодном помещении или после транспортирования в зимних условиях перед началом эксплуатации необходимо дать ему прогреться до температуры $(20 \pm 5)^{\circ}\text{C}$ в течение 2 часов.

Эксплуатация модуля должна производиться в нормальных климатических условиях (температура воздуха от 15 до 25°C при относительной влажности от 45 до 75%). Помните, что модуль ОЗУ является сложным радиоэлектронным устройством. При эксплуатации модуля не закрывайте вентиляционные отверстия в корпусе посторонними предметами. Оберегайте модуль от сильных механических, тепловых и электромагнитных воздействий. Хранить модуль ОЗУ рекомендуется в упаковке завода-изготовителя при температуре воздуха от 5 до 35°C и относительной влажности не более 65%.

Кассету с программным обеспечением, входящую в комплект поставки модуля ОЗУ, рекомендуется хранить в футляре на расстоянии не менее 1 м от нагревательных приборов и электромагнитных устройств, в

месте, защищенном от воздействия прямых солнечных лучей. Эксплуатация кассеты должна производиться в соответствии с правилами, изложенными в руководстве по эксплуатации на ПЭВМ "Микроша".

2. ТЕХНИЧЕСКИЕ ДАННЫЕ

Емкость ОЗУ, Кбайт 16
 Потребляемая мощность, ВхА 5
 Масса без упаковки, кг 0,8
 Габаритные размеры, мм 240x90x30

3. КОМПЛЕКТНОСТЬ

модуль ОЗУ 1 шт.
 руководство по эксплуатации. 1 шт.
 кассета с программным
 обеспечением 1 шт.
 упаковка 1 шт.

4. ТРЕБОВАНИЯ ПО ТЕХНИКЕ БЕЗОПАСНОСТИ

ВНИМАНИЕ!

В блоке питания ПЭВМ "Микроша" имеется опасное напряжение 220 В. В целях обеспечения безопасности все работы по подключению и отключению модуля ОЗУ производить при выключенном тумблере "Сеть" на блоке питания ПЭВМ.

5. УСТРОЙСТВО ИЗДЕЛИЯ

Модуль дополнительного оперативного запоминающего устройства представляет собой функционально и конструктивно законченный блок, предназначенный для расширения оперативной памяти ПЭВМ "Микроша" до 48 Кбайт.

Модуль ОЗУ подключается к системному блоку ПЭВМ "Микроша" через разъем "Внутренний интерфейс". Таким образом ячейки памяти модуля располагаются в непосредственном адресном пространстве микропроцессора. Для работы модуля ОЗУ отведен интервал адресов 8000 - BFFF. Ячейки памяти этого интервала являются доступными для чтения и записи информации с помощью директив "СИСТЕМОГО МОНИТОРА" и для размещения программ пользователя.

На магнитофонной кассете, входящей в комплект модуля, записаны системные программы ("Тест модуля ОЗУ", "Макроассемблер", "Бейсик-Плюс"), рассчитанные на применение совместно с модулем ОЗУ.

Схема электрическая принципиальная модуля приведена на листе 48 настоящего руководства. В составе модуля можно выделить следующие функциональные части:

- 1) схема временного мультипликсирования адреса;
- 2) схема формирования сигналов выборки ("RAS" и "CAS");
- 3) блок микросхем оперативной памяти.

Особенностью микросхем КР565РУ6Д, применяемых в модуле ОЗУ, является временное мультипликсирование адресов и необходимость периодической регенерации содержимого ячеек памяти. Первая

особенность обусловлена тем, что у ИМС КР565РУ6Д имеется только 8 адресных входов, в то время как разрядность шины адреса микропроцессора равна 16. Вследствие этого возникает необходимость в разделении по времени моментов поступления на адресные входы микросхем памяти младших и старших разрядов адреса. Код адреса заносится в адресные регистры микросхем через входы А0...А6 последовательно: сначала поступают коды семи младших разрядов шины адреса, сопровождаемые низким уровнем сигнала "RAS", а затем семи старших, сопровождаемые низким уровнем сигнала "CAS". ИМС КР565РУ6Д имеет организацию 128x128. Это означает, что матрица запоминающих элементов имеет 128 строк и 128 столбцов. Младшие семь разрядов шины адреса служат для выбора строки запоминающих элементов, а старшие семь - для выбора столбца. Бременное разделение адресов производится мультиплексорами D1 и D2, на входы которых с шины адреса поступает разряды А0...А13 кода адреса. В зависимости от уровня сигналов на входах V этих микросхем на входы А0...А6 микросхем памяти поступают сигналы либо с линий А0...А6, либо с линий А7...А13 шины адреса. Сигнал, поступающий с выхода Q3 мультиплексора для адресации памяти не используется.

Необходимость специальных мер по регенерации содержимого ячеек памяти обусловлена тем, что основным запоминающим элементом в микросхемах памяти динамического типа, в том числе и КР565РУ6Д, является конденсатор. Для предотвращения потери информации необходимо не реже чем через 2 мсек производить обращение к каждой ячейке.

Обычно для этого используется режим прямого доступа к памяти. Для регенерации всего объема памяти оказывается достаточным производить обращение последовательно к каждой из 128 строк матрицы запоминающих элементов, сопровождая каждое обращение низким уровнем сигнала "RAS". При этом к внутренним усилителям считывания микросхем подключается соответствующая строка матрицы и автоматически производится подзарядка запоминающих конденсаторов.

При обращении к модулю ОЗУ микропроцессор устанавливает на адресной шине системного блока адрес необходимой ячейки памяти. Дешифратор D17 блока системного формирует сигнал "32K", поступающий на разъем "Внутренний интерфейс". В первый момент времени, при отсутствии сигналов "WR" и "RD", на вход V2 регистра D3 поступает сигнал высокого уровня, что обеспечивает работу регистра в режиме приема данных. Выходы регистра 1, 2, 4 и 8 находятся в единичном состоянии. Управляющие входы V мультимплексоров D1, D2 также переведены в единичное состояние. Вследствие этого к адресным входам микросхем памяти подключены младшие разряды шины адреса. При появлении одного из сигналов "WR" или "RD" происходит переключение логических элементов D4.1 и D4.2. В результате вход V2 регистра D3 переводится в нулевое состояние, а сам регистр - в режим параллельного сдвига информации. В этом режиме с приходом первого тактирующего импульса "OSC" на выходе 1 регистра появляется сигнал низкого уровня. Этот сигнал ("RAS") стробирует запись кода, находящегося на входе микросхем памяти, во внутренние

регистры. При поступлении второго тактирующего импульса нулевой уровень появляется на выходе 2 регистра, что приводит к изменению состояния входов V мультиплексоров и к подключению к адресным входам микросхем памяти старших разрядов шины адреса. При прохождении третьего импульса "OSC" нулевой уровень на выходе 4 регистра совместно с нулевым уровнем сигнала "32K" формирует сигнал "CAS". Считывание бита из ячейки БИС ОЗУ производится в момент действия сигнала "CAS", если предварительно был установлен сигнал "RAS". На время действия сигнала "CAS" информационный выход микросхем переходит из высокоимпедансного состояния в режим выдачи бита. Если одновременно с сигналом "CAS", при предварительно установленном "RAS", действует активный сигнал "WE", то бит данных со входа DI будет записан в соответствующую ячейку запоминающей матрицы микросхемы. При этом информационный выход находится в высокоимпедансном состоянии.

Цикл регенерации дополнительной оперативной памяти осуществляется в процессе регенерации основной памяти системного блока.

6. ПОДГОТОВКА К РАБОТЕ

При подготовке к работе модуля ОЗУ необходимо выполнить следующие операции:

1) выключить питание ПЭВМ "Микроша", для чего перевести тумблер "Сеть" на блоке питания в положение "Выключено";

2) снять крышку разъема "Внутренний интерфейс" на системном блоке ПЭВМ;

3) снять предохранительную крышку с вилки модуля ОЗУ;

4) установить в разъем "Внутренний интерфейс" модуль ОЗУ, как показано на рис.1. Подсоединение модуля не должно сопровождаться значительными усилиями;

5) перевести тумблер "Сеть" на блоке питания ПЭВМ в положение "Включено";

6) нажать клавишу "Сброс" на системном блоке ПЭВМ.

После выполнения указанных операций ПЭВМ "Микроша" находится в режиме диалога с "СИСТЕМНЫМ МОНИТОРОМ".

По окончании работ отключение модуля ОЗУ от системного блока "Микроша" проводится по следующей методике:

1) выключить питание ПЭВМ;

2) отсоединить модуль ОЗУ от системного блока;

3) установить на место крышку разъема "Внутренний интерфейс";

4) установить на место предохранительную крышку модуля ОЗУ.

ВНИМАНИЕ !

Не проводите работ по подключению или отключению модуля ОЗУ при включенном питании ПЭВМ. Это может привести не только к сбою машины, но и к выходу из строя системного блока или самого модуля.

7. ПОРЯДОК РАБОТЫ

7.1. Тест модуля дополнительного оперативного запоминающего устройства

Тест модуля ОЗУ предназначен для автоматизированной проверки работоспособности ячеек дополнительного ОЗУ. Программа составлена в машинных кодах. Стартовый адрес - 7000.

После запуска программы директивой G7000 пользователю следует выбрать режим теста. Для этого надо нажать клавишу "0" (однократный режим) или "Ц" (циклический). Циклический режим применяется для электрического прогона и термотренировки модуля. Его цель - отыскать те микросхемы памяти, которые не выдерживают рабочего электрического или теплового режима. Однократное тестирование модуля применяется в тех случаях, когда у пользователя возникли какие-либо неисправности, связанные с потерей информации или ее искажениями.

В процессе выполнения теста в каждую ячейку памяти модуля многократно записывается и считывается различная информация. При несовпадении записанного и считанного байта выдается сообщение об ошибке. Формат сообщения следующий:

АДРЕС БАЙТ1 БАЙТ2, где
БАЙТ1 - записанный байт;
БАЙТ2 - считанный байт.

По приведенному сообщению можно определить, какая из микросхем является неисправной. Например, при тестировании появилось сообщение:

8030 FF FD

Это означает, что в ячейку 8030 было

записано число FF, а считано FD. Разница между записанным и считанным байтами равна 02. Следовательно, информация теряется в микросхеме, хранящей второй бит, то есть в микросхеме D7 (см. схему электрическую принципиальную). Аналогично, если ошибка произошла в микросхеме D11, то сообщение может иметь следующий вид:

9800 FF DF

При обнаружении ошибок следует отключить модуль ОЗУ от ПЭВМ "Микроша" и обратиться в гарантийную мастерскую.

В процессе тестирования на экран телевизора выводится сообщение "ТЕСТ ПАМЯТИ" и периодически сменяющие друг друга символы "V", "W", ". ". Время полного прохождения однократного теста модуля дополнительного ОЗУ составляет около 6 минут.

7.2. Бейсик "Плюс"

Бейсик "Плюс" занимает в модуле ОЗУ 8 кБайт с адреса 8000 по 9FFF. Бейсик "Плюс" в основном предназначен для разработки новых программ.

Для оперативного набора программ в Бейсике "Плюс" предусмотрена возможность ввода служебных слов последовательным нажатием клавиши "AP2" и клавиши, соответствующей этому слову. Сведения о соответствии приведены в табл. 1.

Максимальная длина строки - 126 символов. При достижении этого числа символов ввод информации прекращается.

В режиме набора программы с клавиатурой возможно редактирование: удаление символа, раздвижка строки, выключение раздвижки, замена символа.

В одной строке программы возможно

употребление нескольких операторов, записанных через разделитель ":".

Загружается Бейсик в модуль ОЗУ по директиве "I" "СИСТЕМНОГО МОНИТОРА", запускается - по директиве "G8000". На экране дисплея при этом появляется сообщение:

BASIC <PLUS>

OK

Это означает готовность интерпретатора к работе.

7.2.1. Директивы языка Бейсик "Плюс"

При первом запуске Бейсика автоматически выполняется директива "NEW". В дальнейшем, после нажатия клавиши "СБРОС" запуск интерпретатора происходит без стирания программы. Для удобства пользователя после первого запуска программы предусмотрена возможность рестарта с адреса 0000.

В языке Бейсик "Плюс" имеются следующие директивы:

1) NEW - подготавливает интерпретатор для ввода новой программы с клавиатуры. Текст ранее набранной программы стирается из ОЗУ ПЭВМ;

2) RUN N - запуск программы со строки с номером N. Если номер строки отсутствует, работа программы начинается с наименьшего номера. Всем числовым переменным присваивается значение ноль, а символьным - "пустая строка";

3) LIST N - инициирует распечатку текста программы, начиная со строки с номером N. Если номер строки отсутствует, то распечатка производится с начала и до конца программы;

4) LIST N1,N2 - вывод на экран

фрагмента программы, начиная со строки N1 до строки N2;

5) LIST,N2 - вывод на экран фрагмента программы от начала до строки N2;

6) "AP2" - приостанавливает выполнение директивы LIST. Для продолжения выполнения директивы нужно нажать клавишу "BK";

7) EDIT N - вызов строки с номером N для редактирования. Редактирование осуществляется так же, как и при вводе строк;

8) "F5" - нажатие этой клавиши включает раздвижку символов при вводе и редактировании строк;

9) "F3" - нажатие этой клавиши выключает раздвижку;

10) "F2" - удаление символа справа от курсора;

11) "СТРЕЛКА ВНИЗ" - стирание строки, начиная с символа, расположенного справа от курсора. В режиме редактирования можно осуществить дублирование строк. Вызов нужной для дублирования строки производится при помощи директивы "EDIT". Измените номер строки на нужный Вам и нажмите "BK". Номера строк могут быть любыми от 0 до 65529;

12) DELETE N - удаление фрагмента программы со строки N и до конца программы;

13) DELETE,N - удаление фрагмента программы от начала программы и до строки N;

14) DELETE N1,N2 - удаление фрагмента программы от строки N1 и до строки N2;

15) AUTO N1,N2 - автоматическая нумерация вводимых строк программы, начиная со строки N1 с шагом N2. Если N1, N2 не указываются, то N1 = N2 = 10. N1 находится в пределах 0 - 65529, N2 - от 1

до 255;

16) RENUM N1,N2 - замена номеров строк программы на номера, начинающиеся с N1 с шагом N2. Пределы для N1, N2 такие же, как и в директиве AUTO;

17) HIMEM N - устанавливает верхнюю границу буфера, занимаемого программой на Бейсике. N - десятичный номер ячейки памяти;

18) CONT - позволяет продолжить выполнение программы с того места, где она была прервана при выполнении оператора STOP;

19) STOP - останавливает выполнение программы с выдачей номера строки, в которой произошла остановка;

20) "F4" - при нажатии этой клавиши останавливается выполнение программы;

21) "CTR" - при нажатии этой клавиши можно выйти в "СИСТЕМНЫЙ МОНИТОР" без стирания экрана;

22) CLOAD - загрузка программ, написанных на Бейсике "Микроша" или аналогичных версиях Бейсика;

23) CLOAD "ИМЯ" - загрузка с магнитной ленты программы с соответствующим именем;

24) CLOAD "" - загрузка первой встретившейся программы. Бейсик "Плюс" позволяет организовать поиск программ на магнитной ленте. При нажатии "BK", после набора директивы CLOAD "ИМЯ", Бейсик ожидает ввода программы с ленты сколь угодно долго, не выходя в "СИСТЕМНЫЙ МОНИТОР". Между программами может быть записана речь, ПЭВМ на нее не реагирует. Для поиска программы включите магнитофон на воспроизведение. Все имена программ, которые встретятся по ходу поиска, будут выведены на экран. Загрузка программы в ОЗУ произойдет только при совпадении имен;

25) **CSAVE"ИМЯ"** - запись программы на магнитную ленту. Имя может отсутствовать, но кавычки обязательны. Длина имени не должна превышать 120 символов. При записи на магнитную ленту, для более уверенного считывания программ с ленты, производится подсчет контрольной суммы. Если при загрузке программы произошла ошибка, то на экран выводится сообщение: "НЕСООТВЕТСТВИЕ ДАННЫХ". В связи с этим при считывании программ, записанных на Бейсике "Микрон", появляется указанная выше ошибка. Следует, не запуская программы, вывести ее на ленту в формате Бейсика "Плюс", а затем снова ее считать.

26) **VERIFY"ИМЯ"** - проверка правильности записи программ на магнитную ленту;

27) **MERGE"ИМЯ"** - дозагрузка (пристыковывание) программы с магнитной ленты к имеющейся в ОЗУ программе. Номера строк второй программы должны быть больше максимального номера программы в ОЗУ.

7.2.2. Операторы языка Бейсик "Плюс"

В описываемой версии языка используются следующие операторы:

1) **REM** - служит для вставки в текст программы комментариев. Все, что стоит за этим оператором, интерпретатором игнорируется;

2) **DIM** - предназначен для описания массивов, используемых в программе. Массив можно не описывать, если его размерность не превышает 10. Одним оператором **DIM** можно описать сразу несколько массивов;

Пример:

10 **DIM** A(40),B(25,25),R(3,3,3)

20 **DIM** A\$(13),D\$(5,10)

3) **DATA** - описание блока данных.

Оператор DATA может располагаться в любом месте программы. С его помощью могут быть описаны любые данные, как числовые, так и символьные;

Пример:

```
10 DATA 1, -3, "МИКРОМА"
```

4) READ - чтение данных из блока данных и присвоение конкретных значений переменным программы;

Пример:

```
READ X1, X2, ... XN
```

X1...XN - имена числовых и текстовых переменных.

5) RESTORE - установка указателя на первый элемент блока данных, что дает возможность повторного считывания данных из блока данных;

6) RESTORE X - установка указателя данных на строку X. Это позволяет обращаться к разным блокам данных, в любой последовательности;

7) INPUT X - присвоение переменной X значения, введенного с клавиатуры. После оператора "INPUT" может стоять строка символов, заключенная в кавычки. В этом случае при выполнении оператора на экран дисплея будет выведено это сообщение, а затем знак "?". Если после сообщения в кавычках вместо ";" поставить ",", то будет отменен вывод "?" при выполнении этого оператора. Вводимые значения переменных могут быть как числами, так и результатами каких-либо математических выражений;

Пример:

```
10 INPUT A
```

```
20 INPUT A, B, C
```

```
30 INPUT R$, D$, J
```

```
40 INPUT "ВАШЕ ИМЯ"; A$
```

```
50 INPUT "ЧИСЛО=", X
```


8) PRINT - вывод на экран дисплея значений переменных, сообщений и результатов вычислений. Если оператор использован без операнда, то это приводит к печатанию одной пустой строки. Операндов, стоящих за оператором "PRINT" может быть несколько, и тогда их отделяют друг от друга разделителями ",", " или ";". Если после последнего операнда в операторе "PRINT" стоит разделитель, то при выполнении следующего оператора "PRINT" печать будет продолжена в той же строке. Если разделителя нет, то печать начнется с новой строки. Числа при печати могут быть представлены в виде целого числа, числа с десятичной точкой, в экспоненциальном или шестнадцатичном виде;

Пример:

```
10PRINT"МИКРОША"
20PRINT"КОЛИЧЕСТВО=";X
30PRINTA,B,10
40PRINT2*2
50PRINT$,B$, &AA,X
60PRINT:PRINT
```

9) PRINT AT X,Y - печать данных на экране с позиции X,Y, где X - номер позиции в строке, а Y - номер строки экрана (максимальное значение X = 63, Y = 24);

Пример:

```
10PRINT AT 10,20"МИКРОША"
20PRINT AT 2,10;X
```

10) CUR X,Y - перемещение курсора в позицию с координатами X,Y. Начало отсчета - левый нижний угол экрана. Диапазон изменения координат курсора по горизонтали X от 0 до 63, по вертикали Y от 1 до 24;

11) SCREEN\$(X Y) - присваивает символьной переменной значение равное символу, расположенному на экране в

позиции X,Y.;

Пример:

```
10 A$=SCREEN$(10,20)
```

```
20 PRINT A$
```

12) INKEY\$ - производит опрос клавиатуры компьютера, после которого выполнение программы продолжается. Если ни одна клавиша клавиатуры не нажата, символьной переменной оператора присваивается значение "пустая строка";

Пример:

```
10 A$=INKEY$
```

```
20 PRINT A$
```

```
30 GOTO 10
```

13) GOTO N - оператор безусловной передачи управления на строку с номером N;

14) GOSUB N,...RETURN - служат для организации подпрограмм. "GOSUB N" передает управление на строку с номером N. Заканчиваться подпрограмма должна оператором "RETURN", который передает управление оператору, следующему за "GOSUB";

15) ON X GOTO N1,N2,...NN и ON X GOSUB N1,N2,...NN - в зависимости от результата вычисления выражения "X" реализуют условную передачу управления на одну из строк программы, номер которой указан в списке, следующем за оператором;

16) FOR X TO Y STEP Z - оператор инициализации цикла;

17) NEXT X - оператор конца цикла;

18) STEP Z - шаг цикла. Если шаг равен 1, то оператор можно опустить. Все, что находится между ними - тело цикла;

Пример:

```
10 FOR I=1 TO 10 STEP 2
```

```
20 PRINT I;
```

```
30 NEXT I
```

В операторе "NEXT", имя переменной можно

не указывать или указывать через запятую.

```
10 FOR I=1 TO 10:NEXT
20 FOR A=0 TO 3
30 FOR B=1 TO 4
40 FOR C=10 TO 1 STEP -2
50 NEXT C,B,A
```

или

```
50 NEXT:NEXT:NEXT
```

19) IF X THEN Y - условный оператор, проверяющий выполнение условия "X": если X - истинно, то выполняются операторы, стоящие в строке после слова "THEN"; если X - ложно, то управление будет передано следующей строке программы. Выражение X может включать проверку самых различных условий. Операторы, стоящие после слова "THEN" также могут быть различными, с одним условием: если необходимо выполнить оператор перехода, то оператор IF должен стоять последним в строке, так как операторы, стоящие за ним в строке не будут выполняться. Оператор "GOTO" можно опустить и просто указывать номер строки программы, которой следует передать управление;

Пример:

```
10 IF X=110 THEN 1000
20 IF X=1 THEN Y=2
30 IF X=5 AND Y=2 THEN GOSUB 10040 IF
Z=4,5 OR X=.1 THEN Y=15: GOTO 1000
50 IF B$="PRO" THEN PRINT "МИКРОША"
```

20) CLS - очистка экрана дисплея;

21) HOME - очистка экрана с установкой курсора в левый верхний угол;

22) PLOT X,Y,Z - высвечивает (Z=1) или гасит (Z=0) точку с координатами $0 < X < 120$ по горизонтали и $0 < Y < 49$ по вертикали;

23) LINE X,Y - чертит линию от точки, заданной оператором PLOT до точки с координатами X,Y;

Пример:

10 PLOT 10,13,1: LINE 24,25

24) PAUSE X - временная задержка на X секунд (X от 0,0015 до 65). В отличие от других версий языка, время в Бейсике "Плюс" реальное;

25) BEEP X,N - звуковой сигнал продолжительностью X секунд (X от 0,01 до 32) с высотой тона N (N от -12 до 70);

26) POKE X,Y - позволяет записать ячейку памяти, адрес которой задан выражением "X", величину, равную результату выражения Y (значение результата должно находиться диапазоне от 0 до 255 (00 - FF в шестнадцатиричной системе));

27) PEEK (X) - преобразование содержимого ячейки памяти, адрес которой определен аргументом X, в шестнадцатиричную форму;

28) CLEAR X - очищает память от переменных. Если параметр X не указан, то всем числовым переменным присваивается значение ноль, а символьным - "пустая строка". Если параметр X указывается, то в памяти выделяется область размером X байт для хранения символьных переменных;

29) DEF - оператор определения функции в программе. Имена всех функций должны начинаться с символов "FN".

Пример:

10 DEFFNCT (X)=COS(X)/SIN(X)

20 PRINT FNCT(X)

7.2.3. Функции в языке Бейсик "Плюс"

Бейсик "Плюс" имеет следующие встроенные функции:

1) LEN(X\$) - определяет число, равное количеству символов переменной X\$;

2) LEFT\$(X\$,Y) - позволяет вывести на экран строку символов длиной Y, начиная с крайнего левого символа;

3) RIGHT\$(X\$,Y) - то же самое, но начиная с крайнего правого символа;

4) MID\$(X\$,Y,Z) - выводит строку символов длиной Z, начиная с позиции Y. Отсчет производится слева направо;

5) STR(X) - преобразует числовые величины в символьный вид. Аргумент X - число или арифметическое выражение;

6) VAL(X\$) - производит обратное преобразование данных из символьного вида в числовой, начиная с крайнего левого символа переменной X\$;

7) ASC(X\$) - переводит код первого символа X\$ в десятичный вид;

8) CHR\$(X) - выводит на экран символ, код которого равен X (аргумент не должен превышать значения 255);

9) POS(1) - результатом является целое число, равное номеру позиции последнего символа в текущей строке;

Пример:

```
10 PRINT"ABCD"POS(1)
```

```
ABCD 4
```

10) SPC(X) - позволяет вставлять в печатаемую строку X пробелов (X<126);

11) TAB(X) - перемещает курсор в позицию с номером X (X от 0 до 63) от начала строки. Символы в позициях с меньшими номерами не стираются;

12) FRE(X) - определяет число свободных байтов памяти;

13) USR(X) - предназначена для организации связи программ, написанных на Бейсике, с программами, написанными в машинных кодах. X - это адрес ячейки памяти, в которой записана программа в машинных кодах;

14) ADDR(X) - определяет адрес первого байта значения переменной X;

15) SQR(X) - корень квадратный из X;

16) EXP(X) - экспоненциальная функция E;

17) LOG(X) - натуральный логарифм;

18) LG(X) - десятичный логарифм;

19) ABS(X) - абсолютная величина;

20) SGN(X) - знак:

1, если $X > 0$

0, если $X = 0$

-1, если $X < 0$;

21) SIN(X) - синус (X в радианах);

22) ASN(X) - арксинус;

23) COS(X) - косинус;

24) ACS(X) - арккосинус;

25) TAN(X) - тангенс;

26) ATN(X) - арктангенс;

27) INT(X) - целая часть числа;

28) RND(1) - случайное число в диапазоне от 0 до 1.

В версии Бейсик "Плюс" приняты следующие обозначения:

PI - число ПИ;

& - используется для обозначения шестнадцатиричных чисел (&A=10, &10=16);

@ - используется для перевода десятичных чисел в шестнадцатиричную форму совместно с оператором PRINT (для чисел от 0 до 65535).

7.2.4. Сообщения об ошибках

Интерпретатор языка Бейсик позволяет в процессе выполнения программы обнаруживать и анализировать ошибки, за исключением логических.

Ошибки могут быть обнаружены как в непосредственном, так и в программном режиме.

На экран дисплея выводится название ошибки с указанием номера строки, в которой она встретилась, а также после символа ":" - порядковый номер оператора, в котором допущена ошибка.

В случае синтаксической ошибки строка автоматически выводится для редактирования.

Выход в непосредственный режим из EDIT, AUTO, LIST, INPUT - при нажатии клавиши F4.

7.2.5. Служебные слова

Служебные слова выводятся на экран при последовательном нажатии клавиши "AP2" и соответствующей клавиши (см. табл. 1). Автоматический ввод служебных слов возможен в непосредственном режиме, в программном режиме и режиме редактирования.

Таблица 1

AP2+	СЛОВО	AP2+	СЛОВО	AP2+	СЛОВО
A	CLS	А	FN	[CLOAD
B	FOR	Б	THEN	\	CSAVE
C	NEXT	Ц	NOT]	NEW
D	DATA	Д	STEP	^	TAB
E	INPUT	Е	AND	_	TO
F	DIM	Ф	OR	@	SPC
G	READ	Г	SGN	Ъ	SCREEN\$
H	CUR	Х	INT	*	INKEY\$
I	GOTO	И	ABS	+	AT
J	RUN	Я	USR	-	BEEP
K	IF	К	FRE	.	PAUSE
L	RESTORE	Л	INP	/	VERIFY
M	GOSUB	М	POS	:	RENUM
N	RETURN	Н	SQR	;	ACS
O	REM	О	RND	<	LG
P	STOP	П	LOG	1	EDIT
Q		Я	EXP	2	DELETE
R	ON	Р	COS	3	MERGE
S	PLOT	С	SIN	4	AUTO
T	LINE	Т	TAN	5	HIMEM
U	POKE	У	ATN	6	@
V	PRINT	Х	PEEK	7	ASN
W	DEF	В	LEN	8	ADDR
X	CONT	Ь	STR\$	9	PI
Y	LIST	Н	VAL	0	HOME
Z	CLEAR	Э	ASC		
		Ш	CHR\$		
		Э	LEFT\$		
		Щ	RIGHT\$		
		Ч	MID\$		

7.2.6 Автозапуск

Отличительной особенностью программы является возможность автозапуска программ, написанных на Бейсике "Плюс" сразу же после их ввода. Введенная программа может быть запущена с любого номера строки. Для этого перед выводом программы на магнитную ленту (до выполнения директивы CSAVE") следует выполнить следующие действия:

POKE&40,&3E

POKE&42,10

POKE&43,00

Первая строка определяет запись в ячейку с адресом 0040 байта признака автозапуска (3E). Во второй строке в ячейку 0042 записывается младший байт номера строки запуска, а в третьей строке - старший байт (в примере 0010).

Вследствие введения в интерпретатор автозапуска, несколько изменился формат вывода программы на магнитную ленту. Для того, чтобы ввести в ПЭВМ программу, при записи которой не был предусмотрен автозапуск, следует после ввода перемотать магнитную ленту немного назад и воспроизвести любой фрагмент программы (не менее 4 байт).

7.3. Макроассемблер

Программа вводится в память ПЭВМ по директиве I "СИСТЕМНОГО МОНИТОРА". Стартовый адрес - 0000.

Макроассемблер является расширенной версией языка ассемблера, обладающей возможностью работы с макросредствами. В состав программы входит непосредственно транслятор с ассемблера, а также редактор

текстов, с помощью которого производится набор и редактирование текстов программ. При работе макроассемблера память ПЭВМ распределяется следующим образом: интервал 0000-19FF занимает макроассемблер, 1A00-1DFF - область рабочих ячеек. Начиная с адреса 1E00 размещается текст программы, а с адреса 8000 (рабочие ячейки модуля ОЗУ) - область трансляции ассемблера. С адреса 6A00 располагаются рабочие таблицы ассемблера. Стек редактора устанавливается с адреса 69FF.

7.3.1 Редактор

Редактор текстов, входящий в состав макроассемблера аналогичен редактору "МИКРОН", применяемому в программах "Редактор и Ассемблер", "Дизассемблер" и т.д. Ниже приводятся его основные возможности.

Редактор предназначен для набора и редактирования любого текста непосредственно на экране дисплея: исправления ошибок, удаления и ввода символов и целых фрагментов. Все исходные тексты с помощью редактора могут быть сохранены на магнитной ленте.

При первоначальном запуске редактора производится очистка экрана и в центре выводится вопрос "NEW?". При ответе "Y" происходит очистка текстового буфера и редактор переходит в режим набора строки (в начале первой строки появится курсор в виде мерцающей линии подчеркивания). При нажатии любой другой клавиши редактор произведет анализ содержимого буфера текста на наличие признака конца текста и, если не обнаружит его, выведет сообщение "МАЛО ОЗУ" и запрос "NEW?", на который

опять следует ответить "Y". Если признак конца текста будет найден, то на экране появится первый фрагмент текста. Редактор в этом случае находится в режиме редактирования (его основной признак - символ "*" в конце строк программы). При наличии в текстовом буфере произвольной информации, заканчивающейся признаком конца текста, на экран будет выведен бессмысленный набор букв, цифр, символов. Его не следует пытаться редактировать, а очистить буфер последовательным нажатием клавиш "AP2" и "N" (директива "AP2"+"N").

Режим ввода строки используют для ввода текста с клавиатуры компьютера, причем строка может содержать не более 64 символов. За 8 позиций до конца строки генерируется звуковой сигнал, предупреждающий о том, что для продолжения набора необходимо перейти на новую строку. Набор строки завершается нажатием клавиши "BK", в результате чего она заносится в текстовый буфер. Допущенную ошибку легко исправить, сместив курсор клавишей "<" до нужного места и набрав нужный символ. Далее можно возвратиться к любому месту набираемой строки. Для перехода к редактированию нужно нажать клавишу "STR".

Режим редактирования позволяет оперативно просматривать введенный текст построчно, либо фрагментами по 24 строки. Очередной фрагмент выводится директивой "AP2"+"стрелка вниз" или "AP2"+"стрелка вверх". Вернуться к началу или концу текста можно с помощью директив "AP2"+"B" или "AP2"+"F" соответственно. Для редактирования внутри отображаемого фрагмента нужно пользоваться клавишами перемещения курсора. Переход из режима редактирования в режим ввода строки

происходит при попытке смещения курсора за пределы нижней границы текста. Из режима редактирования можно перейти в ассемблер при нажатии "СТР".

В редакторе предусмотрена возможность автоматического поиска и замены группы символов. Для этого после задания директивы "AP2"+"L" X=Y вводят искомую группу символов X, новую группу символов Y и нажимают "BK". Если замены не требуется, то после директивы вводится только искомая группа символов. Воспользовавшись директивой "AP2"+"R", можно найти последующие фрагменты с искомой группой символов.

Редактор предусматривает возможность автораздвижки строки. Включают режим директивой "AP2"+"F4", а выключают "AP2"+"F2". Отдельные символы удаляют из строки установкой курсора под соответствующим знакоместом и нажатием на клавишу "F2", а освобождают для пропущенного - "F4".

В режиме редактирования в текст можно вставлять отдельные псевдографические символы, нажав предварительно клавиши "AP2"+"\".

Чтобы вставить в текст одну или несколько строк, к началу следующей за ними строки подводят курсор и выполняют директиву "AP2"+"A". Если же надо вставить строки перед первой строкой текста, то необходимо использовать директиву "AP2"+"стрелка в левый верхний угол экрана". В результате весь текст, следующий за помеченной строкой, удаляется с экрана и редактор переходит в режим ввода строк. Выйти из этого режима можно нажатием клавиши "СТР".

Для удаления фрагмента текста курсор

перемещают в начало его первой строки и выполняют директиву "AP2"+"D". Затем перемещают курсор до строки, перед которой заканчивается удаляемый фрагмент, и нажимают "AP2"+"E". Если данный фрагмент решено оставить, то нажимают клавишу "СТР".

При необходимости разбиения строки текста на две курсор устанавливается на символ, с которого будет начинаться новая строка, и нажимается "BK". Для слияния строк курсор подводят к первой из объединяемых строк и нажимают "ПС".

Имеющийся в памяти компьютера текст можно записать на магнитную ленту по директиве "AP2"+"O". В ответ на нее редактор запрашивает имя текста, под которым он будет сохранен на ленте. Указав имя (что не является обязательным) и, включив магнитофон на запись, нажимают "BK".

Для приема текста с магнитной ленты вводят директиву "AP2"+"I", а затем, в ответ на запрос редактора - имя вводимого текста. После этого включают магнитофон на воспроизведение и нажимают "BK". Если имя не указано, то произойдет загрузка первого встреченного редактором текста. По окончании ввода на экране появится начальный фрагмент введенного текста.

Редактор может самостоятельно сравнить записанный на ленту текст с текстом в памяти компьютера. Для этого надо нажать клавиши "AP2"+"V" и ввести текст с магнитофона. Если тексты не идентичны, на экран выводится сообщение "ОШИБКА", а если совпадают - их начальный фрагмент. Редактор позволяет компоновать текст из нескольких фрагментов, вводимых в этом случае директивой "AP2"+"M". Любую

директиву работы с магнитофоном можно отменить нажатием клавиши "СТР".

7.3.2. Ассемблер

С помощью ассемблера осуществляется трансляция программы с языка ассемблера на язык машинных кодов. Характерной особенностью макроассемблера является возможность трансляции с учетом макросредств: макроопределений, макровывозов и т.п.

Ниже приводятся синтаксические и семантические правила составления программ на макроассемблере.

Программа на макроассемблере представляет собой набор строк. Каждая строка имеет следующий формат:
 МЕТКА:ОПЕРАЦИЯ ОП1,ОП2,...;КОММЕНТАРИЙ

Строка содержит поля метки, операции, операндов, комментария. Разделителем между полем метки и полем операции является двоеточие, между полем операции и полем операнда - пробел, полем операндов и полем комментария - точка с запятой. Операнды отделяются друг от друга запятыми. В остальном формат строки произвольный.

Метки в макроассемблере являются идентификаторами строк. Они бывают двух разновидностей: глобальные и локальные. Основное отличие между ними заключается в том, что областью действия глобальной метки является вся программа, а локальной - участок программы между соседними глобальными метками. Глобальные метки могут состоять из латинских или русских букв и цифр и должны начинаться с буквы. Максимальная длина глобальной метки 16 символов. Локальная метка начинается с символа "?" и идентифицируется по первому

символу.

Пример:

КОД: - глобальная метка;

?C: - локальная метка.

Рассмотрим два фрагмента программы:

КОД: CALL 0F803H ?C: CALL 0F803H

MVI B,20 MVI B,20

?C: DCR B КОД: DCR B

JNZ ?C JNZ КОД

CALL 0F815H CALL 0F815H

JMP КОД JMP ?C

Фрагмент, расположенный слева, будет оттранслирован правильно, а при трансляции правого фрагмента будет допущена ошибка.

В поле операции могут размещаться команды микропроцессора KP580BM80A, директивы ассемблера и макровыводы. Система команд микропроцессора приведена в руководстве по эксплуатации на ИЭВМ "Микроша". Рассматриваемая версия макроассемблера имеет следующие директивы:

1) ORG - присвоение значения счетчику трансляции. Используется в тех случаях, когда составляемая программа должна работать в интервале адресов, отличном от области трансляции. Пусть имеется программа:

KOD: CALL 0F803H

CALL 0F815H

JMP KOD

END

После трансляции машинные коды этой программы будут расположены в области трансляции (начиная с адреса 8000), причем рабочие адреса будут совпадать с адресами размещения. Если же перед первой строкой программы вставить строку ORG 5000H, то программа будет оттранслирована в машинные коды с адреса 5000H. Рабочие адреса при этом не совпадают с адресами размещения.

Для того, чтобы работать с указанной программой, ее нужно переместить из области трансляции в рабочие адреса директивой T "СИСТЕМНОГО МОНИТОРА":

T8900,8008,5000;

2) DB - размещение в памяти констант однобайтовой величины. Примером использования этой директивы макроассемблера может служить следующий программный фрагмент:

```
LXI H,BUFF
CALL OF818H
JMP OF89DH
BUFF: DB 1FH,0AH,0DH
      DB 'МАКРОАССЕМБЛЕР'
      DB 00
      END
```

При использовании указанной конструкции в памяти машины будут последовательно размещены байты 1FH,0AH,0DH, байты, кодирующие буквы слова МАКРОАССЕМБЛЕР и байт признака конца буфера - 00;

3) DW - размещение в памяти констант двухбайтовой длины. Выполнение директивы аналогично директиве DB, но с элементами списка, имеющими длину в два байта;

4) DS - резервирование памяти. Используется для того, чтобы зарезервировать определенный интервал памяти для размещения переменных или промежуточных данных. Например, в результате выполнения директивы DS 100 при трансляции программы будут оставлены свободными 100 ячеек памяти;

5) EQU - определение глобальной или локальной константы. Пример:

```
WWOD: EQU OF803H
MEMORY: EQU 5000H
WYWOD: EQU OF815H
;*****
```



```

WORK:  CALL WWORD
        STA MEMORY
        CALL WYWORD
        JMP WORK
        END

```

В примере строкой комментария разделены область определения констант и основная программа. Как видно основная программа не содержит конкретных адресов, что является хорошим стилем программирования;

6) SET - определение переменной периода трансляции. Понятие о переменной периода трансляции будет введено ниже;

7) END - директива конца трансляции. Как только транслятор с ассемблера достигает строки с директивой END, процесс трансляции прекращается. Директива END обычно содержится в последней строке программы;

8) IF - начало блока условной трансляции;

9) ENDIF - конец блока условной трансляции;

10) WHILE - начало блока циклической трансляции;

11) ENDW - конец блока циклической трансляции;

12) MACRO - начало макроопределения;

13) ENDM - конец макроопределения.

В поле операнда обычно размещаются параметры или данные, с которыми оперирует команда микропроцессора или директива макроассемблера. Операнды делятся на регистровые и числовые. Регистровый операнд содержит мнемоническое обозначение одного из регистров микропроцессора, причем названия регистровых пар могут приводиться как полностью, так и по названию старшего регистра пары. Таким образом, операнд в команде LXI HL, 0000

аналогичен операнду в команде LXI H,0000.

Числовые операнды могут быть следующих видов:

1) термы. Под термом в данном случае понимается некоторое арифметико-логическое выражение. В термах допускаются следующие операции: * (умножение), / (деление с остатком), + (сложение), - (вычитание), > (больше), < (меньше), = (равно), # (не равно), & (поразрядное И), ! (поразрядное ИЛИ), % (поразрядное исключающее ИЛИ). Арифметические операции имеют наибольший приоритет, операции над разрядами - наименьший. Кроме этого в термах возможно использование скобок любой степени вложенности;

2) символические имена;

3) числовые константы. Числовые константы могут быть десятичными, шестнадцатиричными, восьмиричными и двоичными. При указании шестнадцатиричных констант необходимо в конце константы ставить признак в виде символа H, а если константа начинается с буквы, то перед ней ставится 0. После восьмиричной константы указывается признак Q, а после двоичной - B.

Пример:

0F803H - шестнадцатиричная;

777Q - восьмиричная;

100 - десятичная;

11011011B - двоичная;

4) символьные строки (1-2 символа);

5) операнд \$ (текущее значение счетчика размещения). Наиболее широко применяется при организации циклов:

MVI B,20H

DCR B

JNZ \$-1

```

LXI D,0AAAAH
DCX D
MOV A,D
ORA E
JNZ $-3

```

7.3.3. Макроопределения и макровыводы

Рассмотрим фрагмент программы:

```

RDBUF:  MACRO @BUSY1,@INDEX1
        LXI H,@BUSY1
        LXI D,@INDEX1
        DAD D
        MOV A,M
        ENDM
WRBUF:  MACRO @BUSY2,@INDEX2
        LXI H,@BUSY2
        LXI D,@INDEX2
        DAD D
        MOV M,A
        ENDM
;*****
RDWR:   RDBUF 5000H,100H
        CPI OFFH
        JZ ALLES
        WRBUF 6000H,100H
ALLES:  JMP OF89DH
        END

```

Приведенный фрагмент содержит два макроопределения RDBUF и WRBUF, а также основную программу, начинающуюся с метки RDWR. Рассмотрим более подробно конструкцию макроопределения. Первое предложение MACRO идентифицирует начало макроопределения. Текст в поле метки (RDBUF: и WRBUF:) является именем макроопределения. В поле операции находится директива начала

макроопределения, а в поле операндов расположены имена формальных параметров макроопределения. В рассматриваемой версии макроассемблера имена всех формальных параметров начинаются символом @, что позволяет осуществлять замену формальных параметров на фактические в процессе макрогенерации. Имена формальных параметров могут состоять из букв или цифр, а их длина не должна превышать шестнадцати символов. Под процессом макрогенерации понимается замена макровывоза в основной программе строками соответствующего макроопределения. После строки с директивой MACRO следуют строки, составляющие тело макроопределения. Именно эти строки будут порождены в процессе макрогенерации. Макроопределение завершается строкой с директивой ENDM. В общем случае макроопределение имеет следующий формат:

ИМЯ: MACRO ПАР1, ..., ПАРN

ENDM

Строка основной программы, имеющая метку RDWR, является предложением инициализации макроопределения RDBUF или макровывозом. Оно определяет имя макроопределения и аргументы (фактические параметры), которые должны быть использованы при макрогенерации. Формат макровывоза:

ИМЯ ФАК.ПАР1, ..., ФАК.ПАРN

Соответствие формальных параметров макроопределения и фактических параметров макровывоза позиционное. Это означает, например, что формальному параметру @BUSY1 при макрогенерации будет присвоено

значение 5000H, а @INDEX1 - 100H. Как уже отмечалось при макрогенерации происходит замена макровывоза на строки макроопределения. Поэтому после макрогенерации основная программа будет иметь следующий вид:

```
RDWR:  MACRO 5000H,100H
        LXI H,5000H
        LXI D,100H
        DAD D
        MOV A,M
        ENDM
        CPI 0FFH
        JZ ALLES
        WRBUF 8000H,100H
        LXI H,8000H
        LXI D,100H
        DAD D
        MOV M,A
        ENDM
ALLES:  JMP 0F89DH
        END.
```

Необходимо отметить, что строки, составляющие макроопределение, генерируются каждый раз, когда в программе встречается макровывоз соответствующего макроопределения. Это делает возможным отказ от использования подпрограмм. Вообще применение макросредств позволяет программисту записать компактный вариант своей программы, оставляя все технические трудности, связанные с получением окончательного текста, макроассемблеру.

В достаточно сложных макроопределениях не обойтись без использования меток. Однако, если какая-либо строка макроопределения была бы помечена, ее метка оказалась бы дважды определенной (для каждой инициализации макроопределения). Это, естественно, не

позволило бы макроассемблеру получить готовую программу. Для избежания ошибок нужно, чтобы метки в макроопределениях были уникальными. Этого можно достичь двумя путями. Первый - использование относительной адресации с применением счетчика размещения. Второй - использование специального символа "\". Если макроассемблер после имени метки встретит символ "\", то вслед за меткой будет выведено содержание байта-счетчика макровыводов. Допустим, внутри макроопределения нужно отметить одну из строк. Пусть имя метки FORM. Для того, чтобы эта метка была уникальной, после имени нужно поставить "\": FORM\. Тогда при первом макровыводе метка будет иметь имя FORM01, при втором - FORM02, при третьем - FORM03 и т.д.

7.3.4. Условная и циклическая макрогенерация

В рассмотренном выше примере макроопределения генерировались в одну и ту же последовательность предложений. Эти предложения могли отличаться параметрами, но их форма и последовательность были неизменными. Однако в макроассемблере предусмотрена возможность изменять последовательность предложений в зависимости от значений аргументов в макровыводе. Такая возможность связана прежде всего с конструкциями IF...ENDIF и WHILE...ENDW.

Рассмотрим пример условной макрогенерации. Когда в процессе макрогенерации встречается предложение IF, вычисляется булевское значение выражения, размещенного вслед за директивой. Если оно

истинно, то макроассемблер продолжает последовательно обрабатывать строки макроопределения. Если же оно ложно, то макроассемблер переходит к генерации строк, расположенных за строкой ENDIF.

```
VEC: MACRO @SIM1,@SIM2
      IF @SIM1>@SIM2
```

```
PPP: SET 1
      ENDIF
      LXI D,0001H
      IF PPP#1
      LXI D,0FFFFH
      ENDIF
      ENDM
```

```
;*****
```

```
LXI H,7A50H
MVI M,09H
VEC 'R','L'
DAD D
MVI M,17H
JMP 0F89DH
END
```

Обратите внимание, что предложение, следующее за первой строкой условия IF содержит директиву SET. В этой строке переменной PPP присваивается значение 1. Переменная PPP в этом случае является переменной периода макрогенерации. Она используется лишь в процессе макрогенерации и в окончательном ассемблерном тексте программы отсутствует. Приведенная программа выводит в центре экрана "человечка", а также справа или слева от него (в зависимости от параметров макровывоза VEC) - светлый квадрат.

Конструкция WHILE...ENDW указывает на то, что строки тела макроопределения, начиная с WHILE, до ближайшего предложения ENDW должны многократно переноситься в основную программу до тех пор, пока

некоторое условие является истинным. В проверяемых условиях могут быть использованы переменные периода макрогенерации и аргументы.

```

STEK:  MACRO @ENUM
PPT:    SET 1
        WHILE PPT#@ENUM
        PUSH H
PPT:    SET PPT+1
        ENDW
        ENDM
;*****
LXI H,0000
STEK 3
END

```

В процессе макрогенерации в основную программу будут перенесены две командные строки PUSH H.

Необходимо отметить, что макроассемблер допускает использование вложенных условий и циклов глубиной до 8.

7.3.5. Команды ассемблера

После того, как программист составил программу, ввел ее текст с клавиатуры и провел редактирование, он может перейти к трансляции программы. Для этого необходимо, находясь в режиме редактирования, нажать клавишу "СТР". В результате произойдет стирание информации с экрана и в его левой верхней части появится сообщение "МАКРО-580" и символ ">". Макроассемблер переходит в режим ожидания команд. Программист может ввести одну из команд:

1) 1 - трансляция программы с одновременным выводом на экран протокола трансляции (листинга). При этом выводится информация в виде четырех колонок: колонка

ошибок, колонка рабочих адресов, рабочие коды транслируемой программы и исходный текст на макроассемблере. По окончании трансляции в нижней части экрана появляются сообщения о количестве обнаруженных ошибок и о верхней границе области памяти, занимаемой программой. При этом в круглых скобках указывается граница рабочей области, а вне скобок - области размещения;

2) 2 - вывод на экран таблицы имен глобальных меток и присвоенных им значений;

3) 3 - трансляция программы без вывода листинга на экран. Формат выводимой информации аналогичен команде 1, но без отображения протокола трансляции;

4) G - запуск программы с адреса начала области трансляции (8000). После нажатия клавиши "Г/Г" на экран выводится вопрос "ARE YOU SURE?". При ответе "Y" произойдет запуск программы;

5) CTR - выход в режим редактирования редактора текстов.

Макроассемблер имеет встроенный анализатор ошибок трансляции. Им распознаются следующие виды ошибок:

1) U - использование неопределенного символьного операнда;

2) O - недопустимый операнд;

3) C - несуществующая команда;

4) L - недопустимая метка;

5) D - дважды определенная метка;

6) S - лишний символ в поле операндов;

7) T - неверный синтаксис терма;

8) M - ошибка в макроопределении или в макровывозе;

9) I - ошибка в условном операторе;

10) W - ошибка в операторе цикла;

11) E - превышение длины команды либо .

метки;

12) Р - ошибка в макропараметре.

В заключение необходимо отметить, что составляемые на макроассемблере программы могут иметь больший объем, чем программы разработанные на традиционных версиях ассемблера для ПЭВМ "Микроша". Это объясняется тем, что для области трансляции макроассемблера отведено 16 Кбайт памяти.

8. ГАРАНТИЙНЫЕ ОБЯЗАТЕЛЬСТВА

Предприятие - изготовитель гарантирует соответствие изделия требованиям технических условий при соблюдении владельцем правил эксплуатации, изложенных в настоящем руководстве и эксплуатационной документации на ПЭВМ "Микроша".

Гарантийный срок эксплуатации - 12 месяцев со дня продажи через розничную торговую сеть. При отсутствии в гарантийном талоне и отрывном талоне на техническое обслуживание даты продажи и штампа магазина, подписи или штампа продавца гарантийный срок исчисляется с даты выпуска модуля предприятием - изготовителем.

В течение гарантийного срока эксплуатации владелец имеет право на бесплатное техническое обслуживание при предъявлении гарантийного талона. В случае обнаружения неисправностей, образовавшихся вследствие нарушения правил эксплуатации, изложенных в руководстве, ремонт модуля ОЗУ производится за счет владельца.


Гарантийное обслуживание модулей ОЗУ производится в гарантийной мастерской по адресу: 127411, г. Москва, Дмитровское шоссе, д.115. Телефон: 485-15-27.

Действителен по заполнении

Лианозовский
электромеханический завод

МОДУЛЬ ОЗУ 5.065.069 ТУ

ОТРЫВНОЙ ТАЛОН

к г : НА ТЕХНИЧЕСКОЕ ОБСЛУЖИВАНИЕ
о а : Заполняется заводом-изготовителем
р р : N 002209 дата выпуска 27.91
е а : Представитель УTK
ш н л : завода-изготовителя 
о т и : (штамп УTK)
к и н : Заполняется торговым предприятием
й и : Дата продажи _____
о н я : (число, месяц, год)
т о : Продавец _____
р е о : (подпись или штамп)
м т : Штамп магазина
в о р :
н б е : Заполняется ремонтным предприятием
о с з : Содержание работ
г л а : по техническому обслуживанию _____
о у : _____
х : _____
т и : _____
а в : Дата выполнения работ _____
л а : (число, месяц, год)
о н : Подпись лица,
н и : выполнившего работу _____
а е :
н : Подпись владельца,
а : подтверждающая техническое
обслуживание _____
Штамп ремонтного предприятия

Лианозовский электромеханический завод

Цена руб.

ГАРАНТИЙНЫЙ ТАЛОН

Заполняется предприятием-изготовителем

Модуль ОЗУ 5.065.069 ту

№ 002202 дата выпуска

Представитель УТК

завода-изготовителя

(штамп УТК)

Адрес для предъявления претензий к
качеству:

127411, г. Москва, ЛЭМЗ

Заполняется торговым предприятием

Дата продажи

(число, месяц, год)

Продавец

(подпись или штамп)

Штамп магазина

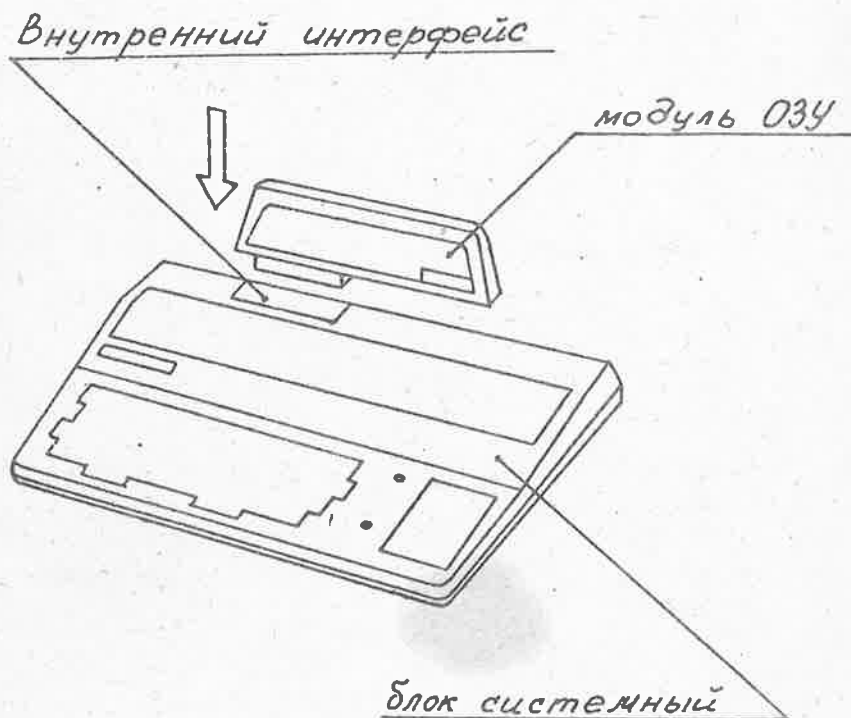


Рис. 1

Перечень элементов модуля ОЗУ

КОНДЕНСАТОРЫ


C1...C12	КМ56-Н90-0,1мкФ	12шт
	ОЖО.460.043ТУ	

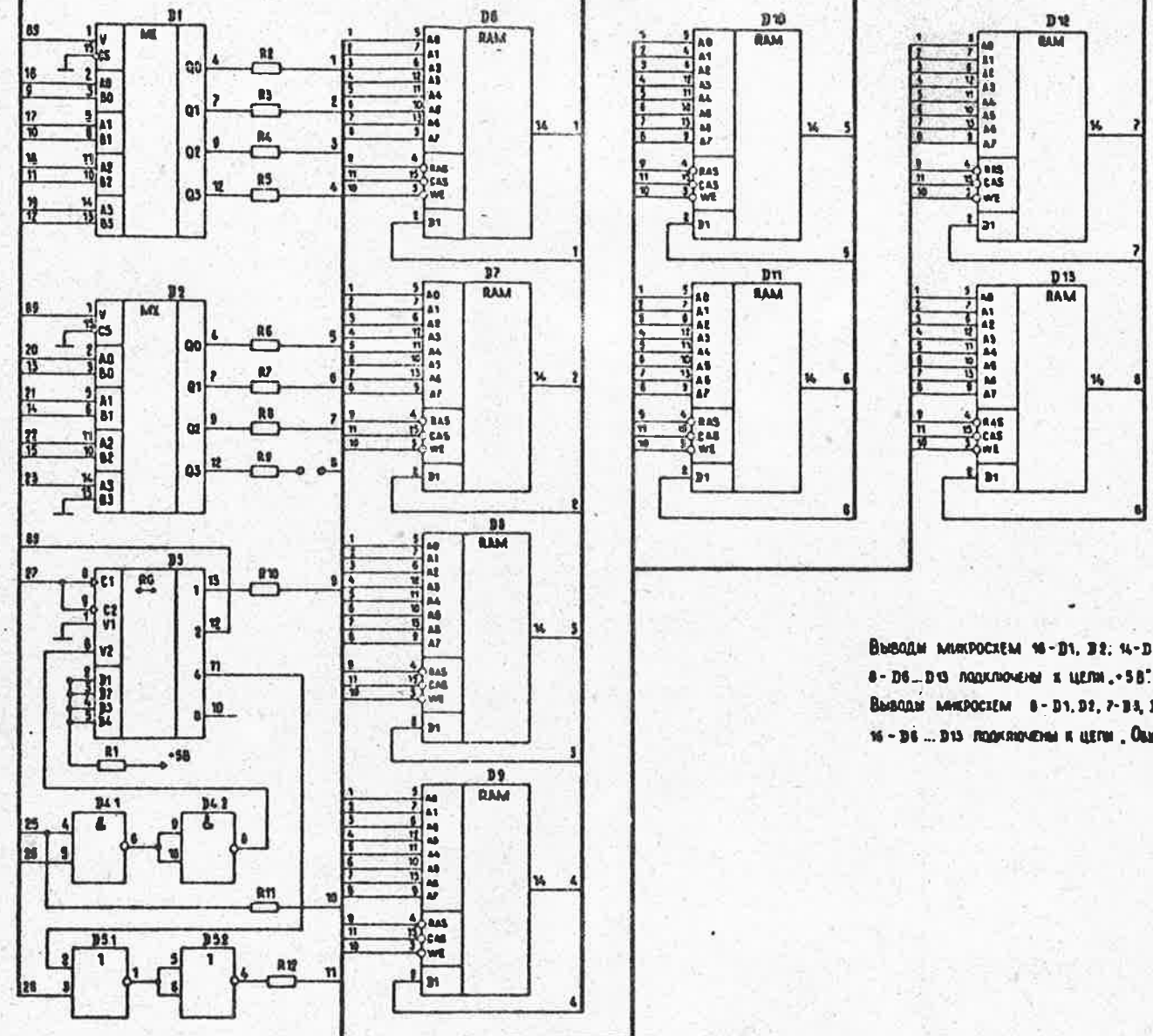
МИКРОСХЕМЫ

D1, D2	К555КП11	БКО.348.289-14ТУ	2шт
D3	К155ИР1	БКО.348.006ТУ	1шт
D4	К555ЛА3	БКО.348.289-01ТУ	1шт
D5	К555ЛЕ1	БКО.348.289ТУ4	1шт
D6...D13	КР565РУ6Д	БКО.348.731ТУ	8шт

РЕЗИСТОРЫ ОЖО.467.180 ТУ

R1	МЛТ-0,125-1кОм+10%	1шт
R2...R12	МЛТ-0,125-33Ом+10%	10шт
X1	Вилка	1шт

• 30 ← 



Выводы микроскопом 10-Д1, Д2, 14-Д3, 14, Д5:
8-Д6...Д13 подключены к цепи "5В".
Выводы микроскопом 8-Д1, Д2, 7-Д3, 14, Д5:
16-Д6...Д13 подключены к цепи "Общий".

11/20/2000	11/20/2000	11/20/2000	11/20/2000	11/20/2000