198099, Ленинград, Калинина, 13
тел. 221–75–64, 221–76–31
факс (812) 186–33–90

ЛКЦП. Зак. 2126-5000. 1991 г.

Персональный компьютер

ХОББИТ

ИНСТРУКЦИЯ ПО ЭКСПЛУАТАЦИИ

УВАЖАЕМЫЙ ПОКУПАТЕЛЬ!

Представляем Вам простой, дешевый и надежный персональный компьютер (ПК) **ХОББИТ** ИК8030 производства фирмы INTERCOMPEX - один из лучших компьютеров своего класса в СССР.

Его аппаратная реализация, обеспечивая полную программную совместимость с компьютером ZX-Spectrum фирмы Sinclair, предоставляет Вам много принципиально новых возможностей, краткое описание которых приводится в следующем разделе.

ПК **ХОББИТ** предназначен в первую очередь для использования в процесе обучения информатике и другим дисциплинам средней и высшей школы отдельных учащихся или различных групп, т.к. он может эксплуатироваться как в автономном режиме, так и в составе локальной вычислительной сети (ЛВС).

Кроме того, ПК **ХОББИТ** может найти применение:

- в дошкольных и школьных кружках и игротеках;
- для автоматизации научной и хозяйственной деятельности;
- для управления технологическими процессами, установками и приборами на производстве и в быту;
- для создания информационно-справочных, экспертных, диагностических, тестирующих и т.п. систем.

Благодаря своей простоте, дешевизне и богатому программному обеспечению, ПК **ХОББИТ** может войти в каждый дом и стать для Вас и Ваших детей :

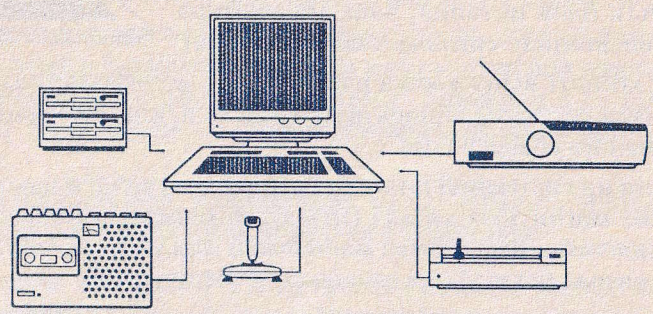
- партнером в разнообразных играх;
- тактичным наставником при ликвидации компьютерной неграмотности;
- проводником по увлекательному и безграничному миру компьютерных профессий.

СОДЕРЖАНИЕ

ОСНОВНЫЕ ВОЗМОЖНОСТИ И ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

В этом разделе мы приведем краткую характеристику ПК ХОББИТ, позволяющую покупателю принять решение о приобретении нашего компьютера, выборе той или иной его модификации, составе необходимых периферийных устройств и программных средств.

ПК ХОББИТ, как и всякий персональный компьютер, включает следующие устройства (см. рисунок): процессор (системную плату), клавиатуру, монитор, накопители на магнитных носителях (магнитофон, дисковод), дополнительные устройства ввода/вывода (принтер, плоттер, джойстик и др.).



Ниже приведены основные технические характеристики ПК ХОББИТ.

Наименование параметра	Значение
Система команд	микропроцессора (МП) Z80
Тактовая частота (Мгц)	3.5
Время выполнения команды МП (мкс)	1.2-6 (875-167 тыс оп/сек)
Разрядность (бит)	8
Объем памяти (Кбайт)	128
из них: ПЗУ	64
ОЗУ	64
Число строк экрана	24
Число символов в строке	32 (64)
Число точек экрана	256 × 192
Число отображаемых цветов	15
Скорость обмена информацией с накопителем на магнитной ленте (бит/сек)	определяется программно
Скорость сетевого обмена (Кбит/сек)	56
Блок питания	встроенный, 5±0.15 В при токе 2 А
Потребляемая мощность от сети переменного тока напряжением 220 В±10% и частотой 50 Гц (Вт, не более)	10

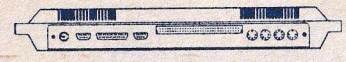
КОНСТРУКЦИЯ ПК

Конструктивно ПК ХОББИТ выполнен как единый блок, объединяющий системную плату ПК, клавиатуру, интерфейсы периферийных устройств (в составе системной платы), блок питания, блок модулятора телевизионного сигнала (необязательно).

ПК ХОББИТ выпускается в нескольких модификациях, различающихся составом аппаратных средств и/или резидентного программного обеспечения.

Если на системной плате размещен контроллер накопителя на гибких магнитных дисках (НГМД), то модель помечается буквой М (ИК8030М) и называется мастерской или учительской, в противном случае (ИК8030) она называется ученической.

На задней панели ПК расположены разъемы для подключения монитора, магнитофона, принтера, локальной вычислительной сети, джойстика, а также разъем системной шины, используемый для подключения различных нестандартных периферийных устройств и НГМД.



- *КЛАВИАТУРА*

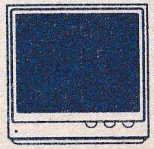
ПК ХОББИТ имеет 88-ми клавишную двухрегистровую двуязычную клавиатуру профессионального типа с латинским и русским* алфавитом, расположенными по стандарту QWERTY - ЙЦУКЕН.

Функциональная нагрузка на отдельную клавишу значительно уменьшена по сравнению с ПК ZX-Spectrum.

- *ОТОБРАЖЕНИЕ ИНФОРМАЦИИ*

ПК ХОББИТ обеспечивает отображение информации:

- ♦ на всех типах цветных и черно-белых телевизионных приемников с возможностью подключения по любому из входов:
  - ◊ RGB-вход, изображение - цветное;
  - ◊ RF- (антенный) или видео-вход, изображение - черно-белое;
- ♦ на мониторах с кадровой частотой от 50 до 70 Гц;



*)По желанию покупателя русский алфавит может быть заменен на любой национальный, если объем покупаемой партии не менее 10-ти штук.

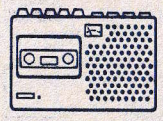
- ♦ с разрешением 256×192 точки или в символьном режиме 32 символа на 24 строки;
- ♦ 15 цветов (или градаций яркости серого цвета для черно-белых телеприемников).

- *ВНЕШНИЕ НАКОПИТЕЛИ НА МАГНИТНЫХ НОСИТЕЛЯХ*

Для хранения постоянной информации ПК ХОББИТ позволяет использовать внешнюю память в виде магнитной ленты (МЛ) или гибкого магнитного диска (ГМД)* размером 5.25" (133 мм) или 3.5" (89 мм) одно- и двухстороннего с двойной плотностью записи.

Обмен (ввод/вывод) информации между внешней и оперативной памятью может производиться с помощью:

- ♦ бытового кассетного магнитофона,

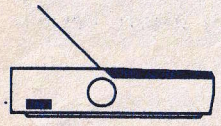


- ♦ накопителя на гибких магнитных дисках. Одновременно может быть подключено до четырех дисководов. (Определение типа используемого ГМД и настройка на его параметры производится автоматически).

- *ДОПОЛНИТЕЛЬНЫЕ УСТРОЙСТВА ВВОДА/ВЫВОДА*

Принтеры

С ПК ХОББИТ может использоваться широкий спектр отечественных и импортных принтеров с параллельными интерфейсами, работающих в алфавитно-цифровом, графическом и псевдографическом режимах.



В случае использования национального алфавита обеспечивается загрузка знакогенератора в печатающее устройство.

Джойстик

Для выполнения игровых, демонстрационных и управляющих программ может использоваться джойстик типа Kempston.

Нестандартные периферийные устройства

Подключение нестандартных периферийных устройств, таких

*)В литературе вместо аббревиатур ГМД и НГМД часто используются аббревиатуры FD (floppy disk) и FDD (floppy disk drive)

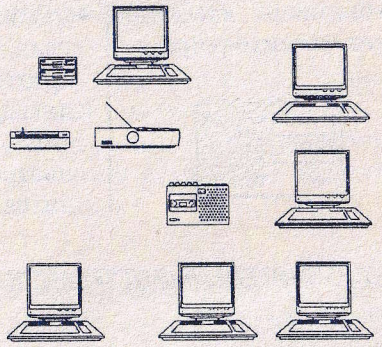
как датчики медицинского и технологического контроля, бытовые приборы и т.п., осуществляется при помощи разъема, на который выведены сигналы системной шины.

• ТЕНЕВЫЕ ВОЗМОЖНОСТИ

Одной из характерных особенностей ПК ХОББИТ является наличие так называемого "теневого режима", который обеспечивает через механизм теневых меню непосредственный доступ к системным и сетевым ресурсам с целью контроля, а при необходимости и изменения состояния системы.

• СЕТЕВЫЕ ВОЗМОЖНОСТИ

ПК ХОББИТ может использоваться в составе однородной (состоящей только из ПК ХОББИТ) или неоднородной (ПК ХОББИТ и ПЭВМ класса IBM PC) локальной вычислительной сети со скоростью передачи данных около 56 Кбод. В сеть может включаться до 31-го компьютера с расстоянием передачи информации до 20 метров. Возможен выход в нелокальные вычислительные сети через модем.



Сетевые запросы обладают абсолютным приоритетом перед программными, что позволяет локальной сети иметь немедленный доступ к любым машинным ресурсам.

• ПРОЧИЕ ВОЗМОЖНОСТИ

Для понимания всех достоинств и возможностей ПК ХОББИТ необходимо знакомство с его архитектурой. Специалисты найдут эти сведения в отдельной брошюре (1). Здесь мы ограничимся лишь упоминанием еще некоторых возможностей.

ПК ХОББИТ предоставляет:

- ♦ двухэкранный режим работы с возможностью обменивать или копировать основной и альтернативный экраны в любой момент времени;
- ♦ механизм функциональных клавиш, позволяющий пользователю определять самому или использовать ранее определенные и неуничтожаемые случайными ошибками или принудительным сбросом ПК последовательности произвольных директив и операторов языка BASIC, что во многих случаях значительно ускоряет работу и избавляет от повторения рутинных операций;
- ♦ значительно расширенную по сравнению с прототипом номенклатуру портов ввода-вывода, в том числе типа Centronics и RS-232.

• ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

ПК ХОББИТ имеет обширное и разнообразное программное обеспечение (ПО), складывающееся из программных средств:

- ◊ созданных для ПК ZX Spectrum и его последующих модификаций,
- ◊ разработанных специально для ПК ХОББИТ.

В силу абсолютной программной совместимости ПК ХОББИТ с прототипом Вы можете использовать на нем без какой-либо переработки любой программный продукт из всего арсенала средств, созданных для ПК семейства ZX Spectrum. Он включает в себя многочисленные игры для всех возрастов, интепретаторы и компиляторы языков программирования (BASIC, C, PASCAL, ASSEMBLER и т.п.), текстовые, графические и музыкальные редакторы и инструментальные средства разработки ПО.

- *Операционные системы*

Для ПК ХОББИТ разработаны следующие операционые системы или среды (ОС):

- ♦ стандартная операционная среда (СОС), включающая интерпретатор языка BASIC и основные драйверы ввода/вывода. Ее ядро представляет собой точную копию резидентного системного программного обеспечения компьютера ZX Spectrum, чем обеспечивается полная программная совместимость с прототипом. Расширение СОС включает средства, обеспечивающие дисковые операции и работу с локальной вычислительной сетью.
- ♦ универсальная среда обучения LOGO, состоящая из интерпретатора языка LOGO в английской (LOGO-E) или русской (LOGO-R) транскрипции, основных драйверов ввода/вывода (в том числе для дисковых и сетевых операций) и набор из 100 логических игр LOGAME. Среда LOGO является незаменимым средством в учебном процессе.
- ♦ FORTH- система, в основе которой лежит стандарт ФОРТ-83. Эта система особенно полезна при создании пакетов прикладных

программ.

- дисковая операционная система (ДОС) СР/М, позволяющая использовать на ХОББИТе любые программы, разработанные для других ПК, использующих микропроцессор Z80.

BASIC-, LOGO- и FORTH- системы организованы таким образом, что они могут эксплуатироваться как на моделях, оснащенных накопителем на гибких магнитных дисках, так и не имеющих НГМД. Эти системы автоматически подключают драйверы, обеспечивающие дисковые операции в случае подключения дисковода к ПК.

В ПЗУ любой модели ПК ХОББИТ хранится СОС BASIC. Выбор второй резидентной среды зависит от покупателя. Он может отдать предпочтение одной из следующих ОС:

◊ LOGO в русской транскрипции;
◊ LOGO в английской транскрипции или
◊ FORTH - системе.

ПК ХОББИТ модели ИК8030М предоставляет возможность использовать в качестве третьей операционной системы загружаемую с диска ДОС СР/М.

- *Системные программные средства*

Фирма ИнтерКомпекс продает пользователям компиляторы и интерпретаторы языков программирования, инструментальные и сервисные программы, их перечень приведен в таблице.

программа	назначение программы
GENS	профессиональный ассемблер
MONS	монитор системного программиста
HP4T	компилятор языка Паскаль
C	компилятор языка Си
BETA BASIC	интерпретатор расширенного BASIC'а
ARTIST - 1	универсальная графическая программа
Beta Commander	системная оболочка
Tasword	текстовый редактор
Copy 86M	программа копирования с ленты на ленту

Каждое из перечисленных в таблице программных средств продается в виде записи на дискете, дополняемой брошюрой с описанием.

- *Игровые программы*

Игровые программы поставляются в продажу в форме последовательных выпусков. Каждый выпуск содержит от 5 до 10 игр или других занимательных программ, записанных на дискете или компакт-кассете. Описание программ каждого выпуска компонуется в отдельной брошюре.

Формы продажи ПК ХОББИТ

Фирма ИнтерКомпекс продает свою продукцию частным лицам и организациям за наличный и по безналичному расчету. При безналичных расчетах поставка продукции производится в недельный срок после получения подтверждения об оплате.

Различают три основных варианта поставки ПК ХОББИТ:

- для автономного использования
- для формирования однородной локальной вычислительной сети (однородого класа) только из ПК ХОББИТ
- для создания неоднородной ЛВС, включающей один компьютер типа IBM PC/XT/AT и от одного до 16 ПК ХОББИТ.

Для автономного использования предлагается ПК ХОББИТ моделей ИК8030М и ИК8030 с одним из четырех описанных выше вариантов резидентного программного обеспечения.

По желанию покупателя фирма ИнтерКомпекс готова доукомплектовать ПК ХОББИТ следующими периферийными устройствами:

◊ монитор цветной или черно-белый с блоком питания;
◊ магнитофон бытовой кассетный;
◊ матричный принтер;
◊ дисковод одиночный или сдвоенный в конструктиве;
◊ размножитель разъема системной шины;
◊ джойстик;
◊ компакт-кассеты;
◊ дискеты размера 5.25" двойной плотности.

Возможные варианты поставки ПК ХОББИТ в составе локальной вычислительной сети приведены в таблице.

СОСТАВ КОМПЛЕКТА ПОСТАВКИ	*ОБОЗНАЧЕНИЕ КОМПЛЕКТА*					
	A8	A12	A16	X8	X12	X16
ПЭВМ PC/AT	1	1	1	-	-	-
ПК ХОББИТ	8	12	16	9	13	17
Монитор:						
цветной	-	-	-	1	1	1
монохромный	8	12	16	8	12	16
Сетевой контроллер	1	1	1	-	-	-
Блок дисковода	-	-	-	1	1	1
Принтер MC6305	-	-	-	1	1	1
Магнитофон	-	-	-	-	-	-
Джойстик	2	3	4	2	3	4
Дискеты с ПО	10	10	10	10	10	10
Кассеты с ПО	-	-	-	4	4	4

Условия эксплуатации и техника безопасности

ПК ХОББИТ требует для своего хранения и эксплуатации соблюдения следующих условий:

Рабочая температура	+5°C ÷ 50°C
Температура хранения	-40°C ÷ +60°C
Влажность воздуха	не более 80%

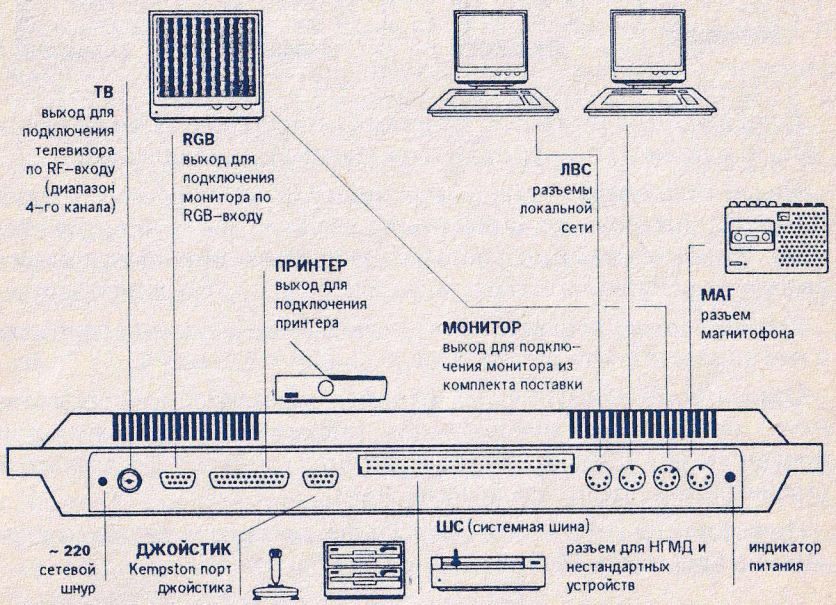
При обязательном соблюдении условий эксплуатации *КАТЕГОРИЧЕСКИ ЗАПРЕЩАЕТСЯ ВКЛЮЧАТЬ КОМПЬЮТЕР В СЕТЬ ~220 В СО СНЯТОЙ КРЫШКОЙ.*

ИНСТРУКЦИЯ НАЧИНАЮЩЕМУ ПОЛЬЗОВАТЕЛЮ

В этом разделе приводятся сведения, позволяющие пользователю самостоятельно сделать первые шаги в освоении ПК ХОББИТ. Они включают порядок подключения компьютера, начальные сведения для работы в BASIC–среде и описание работы тестовой и демонстрационной программ.

Подготовка ПК к работе

Соедините компьютер и все необходимые периферийные устройства при помощи кабелей и разъемов на обратной стороне ПК:



Включите компьютер и подключенные к нему устройства в сеть. (Будьте внимательны при включении НГМД: ручка, закрывающая дискету в накопителе, при включении питания должна быть *открыта*).

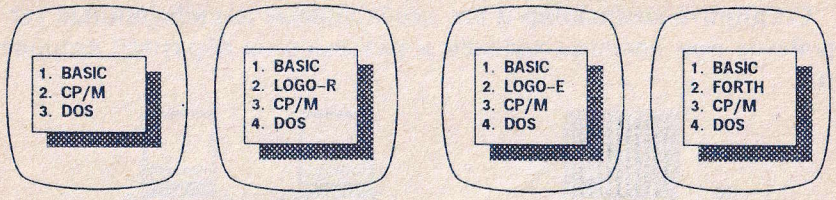
После включения компьютера рекомендуется выполнить специальную тестовую пролграмму (TEST), входящую в любой комплект ПК ХОББИТ. Ее описание приводится в этой брошюре.

Начальная загрузка

Итак, Вы готовы начать работу на ПК ХОББИТ. Уже включили компьютер и все необходимые периферийные устройства.

Для начального старта (*ЗАГРУЗКИ*) следует нажать клавишу RESET. После этого на экране монитора появится *МЕНЮ*, позволяющее выбрать необходимую Вам операционную среду. Состав меню зависит от конкретной аппаратной и программной модификации Вашего компьютера. Если в состав комьютера входит дисковод,

то на экране Вы увидите одну из следующих картинок:



Если дисковод не входит в состав компьютера или он не включен, то в меню будут отсутствовать две последние позиции.

Нажатие клавиш, соответствующих BASIC, CP/M, LOGO-R, LOGO-E и FORTH, приведет к переходу в одноименную операционную систему. Нажатие клавиши, соответствующей надписи DOS, вызовет автостарт начального загрузчика (boot <B>) с диска в BASIC-среде.

Решите, какая операционная среда Вам нужна и для перехода в нее нажмите соответствующую клавишу 1, 2, 3 или 4.

Освоение ПК ХОББИТ целесообразно начинать с работы в СОС BASIC, поэтому вся последующая информация приводится в предположении, что Вы нажали клавишу с цифрой 1, и в нижней строке экрана появилась надпись BASIC.

Информация о работе в других операционных средах содержится в брошюрах (2, 3, 4).

О языке программирования BASIC.

BASIC-среда предназначается в первую очередь для создания, редактирования и выполнения BASIC-программ, т.е. программ, написанных на алгоритмическом языке высокого уровня BASIC. Обучение языку BASIC не входит в нашу задачу. Это можно сделать с помощью общедоступной литературы.

Однако входной язык BASIC'а для ПК ХОББИТ имеет некоторые особенности. В связи с этим в Приложении приводится справочник всех операторов, команд и функций языка BASIC, реализованных для ПК ХОББИТ, включая системные функции и команды, обеспечивающие дисковые операции и работу с локальной вычислительной сетью.

Средства ввода и редактирования программ

Ввод программ, их редактирование и управление работой компьютера осуществляется с помощью клавиатуры и определенного набора программных средств, которые называют *РЕДАКТОРОМ*.

Редактор BASIC'а не требует специального вызова, он инициализируется одновременно с BASIC-средой.

Информация, вводимая с помощью клавиатуры, отображается в нижней части экрана, которую называют EDIT-*СТРОКОЙ*.

Если строка начинается с номера, то она воспринимается как строка BASIC-программы и при нажатии клавиши ENTER, которым завершается ввод каждой строки, она помещается в специальную область памяти. Одновременно содержимое этой строки переносится в верхнюю часть экрана, в которой располагается *ЛИСТИНГ*. Последовательность ввода строк несущественна. В листинге строки располагаются в порядке возрастания номеров. Строка, последней попавшая в листинг, называется *ТЕКУЩЕЙ* и помечается справа от номера *ПРОГРАММНЫМ КУРСОРОМ* >.

Если в начале строки номер не указан, то она воспринимается как команда, в листинг не переносится, а исполняется немедленно.

После ввода правильной строки программы или команды EDIT-строка очищается.

В том случае, когда вводимая строка программы (или команда) содержит синтаксическую ошибку, она не переносится в листинг (не выполняется), а остается в EDIT-строке. На том месте, где было обнаружено первое расхождение с синтаксисом, появится мерцающий вопросительный знак (?).

Для исправления такой строки следует удалить ошибочные символы или добавить новые и повторить ввод строки.

Одновременно на экране помещается до 22 строк. Если Ваша программа больше, то ее можно просмотреть отдельными фрагментами. Команда **LIST n** (см. Приложение) позволяет просматривать программу с любого места. Для просмотра следующей порции, к которому приглашает запрос Scroll?, надо нажать любую клавишу, кроме пробела, нажатие которого прекращает просмотр.

Для вывода листинга программы на принтер служит команда **LLIST**. Перед ее выполнением надо выйти в "теневое" меню и указать системе тип подключенного принтера.

Употребление команд **LIST** и **LLIST** без параметра равносильно их использованию с параметром 0.

При необходимости отредактировать строку BASIC-программы, уже помещенную в листинг, можно поступить двумя способами.

Если изменений в строке много, то ее лучше набрать снова. При нажатии клавиши [ENTER] старое значение строки заменится на новое.

Если исправлений в строке немного, то лучше перенести ее содержимое из листинга в EDIT-строку, сделать необходимые изменения и повторить ввод. В EDIT-строку можно перенести содержимое только текущей строки. Указатель текущей строки > перемещается на нужную Вам строку с помощью стрелок или команды **LIST**.

Существенно ускоряют редактирование и делают его более удобным следующие команды:

ERASE n_1, n_2 — удаление группы строк с номера n_1 по номер n_2

VERIFY : string — поиск строки по образцу string, начинающемуся с ключевого слова.
Результатом выполнения этой команды является список номеров строк, содержащих образец, или собщение O.K., если листинг не содержит строк с указанным образцом.

VERIFY string — поиск строки по образцу string, не начинающемуся с ключевого слова

ПРИМЕЧАНИЕ

Ввод этих команд следует завершать последовательным нажатием клавиш [⇩] и [ENTER].

[SYMBOL SHIFT] — восстановление в EDIT-строке последней введенной команды или строки BASIC-программы, что удобно использовать для повтора команды или редактирования ошибочной строки. Нажатию клавиши [SYMBOL SHIFT] должно в этом случае предшествовать кратковременное нажатие клавиши [⇩].

пробел — автоматический ввод в EDIT-строку очередного номера строки BASIC-программы. Для выполнения этого действия, нажатию пробела должно предшествовать кратковременное или продолжительное нажатие клавиши [⇩].

Автоматически введенный номер будет больше номера пердыдущей введенной строки BASIC-программы на 10 (кратковременное нажатие) или на 1 (продолжительное нажатие).

Отметим еще, что для запуска BASIC-программ на выполнение следует использовать команду **RUN**, для записи программы на внешний носитель - команду **SAVE**, а для ввода программ, ранее записанных на магнитный носитель - команды **LOAD** или **MERGE** (см. Приложение).

Рассмотрим теперь подробнее устройство клавиатуры и работу с ней.

Клавиатура ПК ХОББИТ предназначена для ввода в компьютер информации от пользователя. В ее состав входит 88 клавиш, которые по функциональному назначению с некоторой долей условности подразделяются на 5 групп: *управляющие, регистровые, редактирующие, функциональные* и *алфавитно-цифровые*.

К управляющим относятся клавиши:

[RESET], [BREAK GRAPH], [PAGE SHADOW MODE] и [SYMBOL SHIFT].

Регистровыми называются клавиши:

[RUS/LAT], [C LOCK FN], [C SHIFT DEF FN], [BREAK GRAPH], [⇩] и [SYMBOL SHIFT].

Редактирующими являются клавиши:

[DEL !], [EDIT ;], [ENTER], [⇓], [⇑], [⇐] и [⇒].

К функциональным относятся 8 клавиш верхнего ряда от **PAPER** до **CAT**.

Под алфавитно-цифровыми понимают 63 светлые клавиши, сгруппированные в два блока.

На большинстве клавиш изображено несколько символов. Какой, из изображенных на клавише символов, будет фактически введен в компьютер при ее нажатии, определяется режимом ввода*.

*) Режим ввода зависит от текущей системной конфигурации, режима работы клавиатуры и использованных регистров. Глубокое понимание этого понятия приходит с опытом. Здесь мы останавливаемся лишь на внешних его проявлениях.

Работу с клавиатурой облегчает курсор - мерцающий прямоугольник, указывающий позицию на экране, куда будет вводится очередной символ, и режим ввода, в котором находится компьютер. Режим ввода определяется автоматически, но может изменяться под влиянием регистровых клавиш.

Возможны 8 режимов ввода, идентифицируемых соотвествующими значениями курсора:

K (Keywords) - режим ввода ключевых слов;

– режим ввода в нижнем регистре;

+ режим ввода в двойном нижнем регистре;

L (Letter) - режим ввода строчных символов основного (латинского) алфавита;

C (Capital) - режим ввода заглавных символов основного алфавита;

P и **p** режимы ввода строчных и заглавных символов альтернативного (русского) алфавита соответственно;

G (Graphics) - режим ввода графических символов.

После этих предварительных замечаний рассмотрим работу с каждой группой клавиш.

Управляющие клавиши:

Нажатие клавиши:

CSHIFT DEF FN — переводит ПК в исходное состояние, т.е. уничтожает всю информацию, ранее занесенную в оперативное запоминающее устройство, проверяет работоспособность отдельных узлов компьютера и подготавливает его к новому сеансу работы;

BREAK GRAPH — прекращает выполнение текущей BASIC-команды (в т.ч. команд работы с внешними устройствами);

PAGE SHADOW MODE — в режиме ввода **K** приводит к обмену между основным и альтернативным экраном (см. раздел Использование буфра страницы экрана), а при режиме ввода **–** и **+** вызывает переход в теневой режим (см. раздел Функции теневого режима);

SYMBOL SHIFT — приводит к выходу компьютера из теневого режима;

Регистровая группа:

Нажатие клавиши:

RUS/LAT — вызывает переход из режима ввода **L** или **C** в **P** или **p** и обратно;

C LOCK FN — приводит к изменению текущего режима ввода **L** на **C** или **P** на **p** или наоборот;

CSHIFT DEF FN — меняет (только в нажатом состоянии) режим строчных/заглавных букв основного и альтернативного алфавитов;

⇩ — вызывает переход в один из режимов ввода в нижнем регистре. Если эта клавиша нажимается однократно (кратковременно), то произойдет переход в режим ввода **–**, если нажатие было многократным или продолжительным, то - в режим ввода **+**;

BREAK GRAPH — включает и выключает графический режим ввода **G**;

SYMBOL SHIFT — применяется для достижения совместимости с клавиатурой машины-прототипа;

Редактирующая группа

Нажатие клавиши:

⇐ ⇒ — перемещает курсор внутри EDIT-строки влево и вправо;

⇑ ⇓ — перемещает программный курсор по строкам программы вверх и вниз;

DEL ! — удаляет символ перед курсором;

EDIT ; — переносит в EDIT-строку из верхней части экрана содержимое текущей строки для последующего редактирования. Местоположение курсора в начале или конце строки после нажатия клавиши EDIT ; - зависит от значения курсора (**K** или **+**) перед ее нажатием;

ENTER — завершает ввод набранной строки, т.е. переносит содержимое EDIT-строки в верхнюю часть экрана или на непосредственное интерпретирование и очищает EDIT-строку.

Функциональная группа

режим	результат нажатия клавиши
K	ввод верхнего символа;
–	ввод нижнего символа;
+	перенос в EDIT-строку текста макроса, ранее определенного с помощью теневой функции **WRITE MACROS** (см. раздел Функции теневого режима).

Алфавитно–цифровая группа

Нажатие алфавитно-цифровой клавиши в режиме ввода **K** приведет к вводу цифры или оператора BASIC, название которого находится в центре клавиши. В режимах ввода **–** и **+** последует ввод символа, нанесенного внизу клавиши, или, при нажатии цифровых клавиш и клавиши **(** - кодов, управляющих атрибутами изображения:

— в режиме **–** вводятся коды, управляющие цветом, *которым* выводтяся символы (т.н. *INK-ЦВЕТ*) и *включающие* мерцание (*FLASH*), повышенную яркость (*BRIGHT*) или инверсное изображение (*INVERSE*);

— в режиме **+** вводятся коды, управляющие цветом *на котором* выводятся симолы (*PAPER-ЦВЕТ*) и *выключающие* мерцание, повышенную яркость и инверсное изображение.

Нажатие клавиши, не содержащей оператора BASIC'а в режиме ввода **K**, хотя и будет отображено на экране, является недопустимым в смысле синтаксиса, о чем сигнализирует появление курсора **?** у ошибочного символа при попытке завершить строку нажатием клавиши ENTER.

В режимах **L**, **C**, **p** и **P** нажатие клавиш приведет к вводу строчного или заглавного символа основного или альтернативного алфавитов соответственно.

Применение символов альтернативного алфавита допустимо только в операторах комментария (**REM**) или в строковых переменных. В режимах **P** и **p** допустимы, кроме соответственно букв, все знаки пунктуации и одиночные символы, за исключением [,], {, }, :, #, ~, ↑ и ©, для получения которых необходим переход в режим **K**.

ПРИМЕЧАНИЕ

Несмотря на то, что редактор сам организует автомати-

ческое переключение режимов ввода, рекомендуется завершать работу в режимах **P** и **p** принудительным переходом в режим основного алфавита нажатием клавиши RUS/LAT.

В режиме ввода **G** нажатие клавиш **1** ÷ **8** вызывает ввод мозаичных графических символов, изображенных на этих клавишах. Нажатие клавиш **1** ÷ **8** одновременно с CSHIFT DEF FN вызовет появление инверсного к основному графического символа.

В этом режиме допустимо использование алфавитных клавиш, кроме **V**, **W**, **X**, **Y**, **Z**, если пользователь предварительно определил их графические значения.

Большую помощь в освоении клавиатуры может оказать демонстрационная программа (DEMO), входящая в любой комплект поставки ПК ХОББИТ. Ее описание Вы найдете ниже.

Функции теневого режима

Компьютер переводится в теневой режим либо последовательным нажатием на клавиатуре ПК двух клавиш - регистровой и клавиши PAGE SHADOW MODE, либо одновременным нажатием клавиш C SHIFT DEF FN и PAGE SHADOW MODE, после чего на экран выводится список теневых функций, индикация состояния ПК, определяемого включенными теневыми функциями и способ их вызова (клавиша, которую необходимо нажать для выполнения конкретной теневой функции)

	SYSTEM PAGE	
K	KEY MODE	0
A	AUTOSTART	ON
D	SAVE DELAY	ON
H	HIDE SCREEN	OFF
C	COPY SCREEN	
W	WRITE MACROS	
O	MAGIC OPTION	
P	PRINTER	EPSON
I	INTERRUPTS	1 ON
R	RESTART	
G	RETURN ADR	15F9
Z	MICROMON	
X	DEFAULT PAGE	SYS

Каждая строка экрана теневого режима содержит следующую информацию:

— способ вызова функции,

— название функции,

— текущее состояние ПК, определяемое теневыми функциями.

Ниже приводится краткое описание всех функций.

KEY MODE (0, 1, 2) — выбор режима работы клавиатуры (необходимый режим обычно устанавливается автоматически):
РЕЖИМ 0: - используется при работе в СОС BASIC;
РЕЖИМ 1: - применяется при работе с загружаемым ПО, использующим отличную от BASIC-интерпретатора логику работы с клавиатурой. Альтернативный алфавит не поддерживается. Регистровая клавиша ⇧ не вызывает смены индикации курсора, как в режиме 0, хотя ее действие сохраняется;

РЕЖИМ 2: - эмуляция клавиатуры 40-клавишного ПК ZX Spectrum. Введен для исключения накладных расходов, связанных с обработкой ввода в предыдущих режимах. Может быть полезен при работе с программами, полностью использующими системный стек МП или с программами с жесткой временной диаграммой выполнения (оба эти требования исключают возможность применения режимов 0 или 1);

AUTOSTART (ON/OFF) — отмена автостарта BASIC-программы после загрузки ее в память компьютера с магнитофона;

SAVE DELAY (ON/OFF) — отмена сообщения Start tape, then press any key перед записью на ленту при выполнении команды BASIC'а **SAVE**. Может быть полезна при организации пакетного режима работы программы;

HIDE SCREEN (ON/OFF) — префиксная функция операций с буфером страницы экрана; ее выполнение приводит к однократному изменению режима работы с буфером (копирование вместо обмена);

COPY SCREEN — копирование содержимого экрана на устройство печати. Аналогична команде BASIC'а **COPY**, но копирует весь экран и может быть использована и в случае, когда интерпретатор BASIC'а не активен;

WRITE MACROS — функция записи в таблицу макросов текущего содержимого EDIT-строки и задание клавиши вызова макроса. При выборе этой функции она подсвечивается красным цветом. Теперь Вам нужно выбрать клавишу, с которой Вы свяжете макрос в EDIT-строке. Ею может быть одна из 8-ми функциональных клавиш (см. Режимы клавиатуры). После нажатия на клавишу подсветка исчезает;

MAGIC OPTION — запись на диск текущего состояния компьютера (48 кб ОЗУ и регистров МП);

PRINTER — выбор типа принтера (OFF - принтер отсутствует)

INTERRUPTS — запрет/разрешение (OFF/ON) прерываний МП и установка режима прерываний (1/2) Выполнение функции аналогично выполнению инструкций машинного языка **EI/DI** и **IM1/IM2** соответственно;

RESTART — передача управления по адресу #0000 теневого ОЗУ (начало буфера страницы экрана), куда можно поместить, например, свою программу-монитор;

RETURN ADR — изменение адреса возврата из теневого режима;

MICROMON — микромонитор: шестнадцатиричный калькулятор, перевод чисел из 16-ой в 10-ую систему счисления и наоборот, дамп (шестнадцатиричный и ASCII) и редактирование ОЗУ;

DEFAULT PAGE (SYS/NET) — задание страницы (системная или сетевая) теневого режима, отображаемой сразу же после входа в него. Переход от системной страницы теневого режима к сетевой и обратно - нажатие клавиши ПРОБЕЛ;

Выход из теневого режима производится при помощи нажатия на клавишу [SYMBOL SHIFT], по отпускании которой восстанавливается старое состояние экрана и возобновляется работа прерванной программы.

Теневой микромонитор

Сразу по выбору этой функции становится активным шестнадцатиричный калькулятор.

Если Вы введете два шестнадцатиричных числа, то они отобразятся слева друг под другом. Справа напротив них будет напечатана их сумма и разность.

При вводе шестнадцатиричных чисел нужно учесть, что число из $1 \div 3^x$ цифр надо завершать нажитием клавиши [ENTER], а число максимальной длины из 4^x цифр - не надо.

Если вместо набора второго числа Вы сразу нажмете клавишу [ENTER], то получите справа его десятичный эквивалент.

Если перед набором первого числа Вы нажмете ПРОБЕЛ, то калькулятор перейдет в десятичный режим. В этом режиме возможен только перевод десятичного числа в шестнадцатиричное.

Для перевода микромонитора в режим литерного дампа необходимо нажать клавишу **N** (она отобразится на экране) и затем набрать шестнадцатиричный адрес. Вы получите литерный дамп с указанного адреса. Для получения очередной страницы дампа используйте клавишу [ENTER], для возврата в системную страницу нажмите [CSHIFT DEF FN].

Аналогичным образом, используя в качестве префикса букву **M**, Вы можете перевести микромонитор в режим шестнадцатиричного дампа. Здесь у Вас открываются более широкие возможности:

- передвижение вверх (в сторону уменьшения адресов) на страницу - клавиша [ENTER];
- передвижение вниз (в сторону увеличения адресов) на страницу - клавиша [⇩];
- передвижение курсора - клавиши [⇐] [⇒] [⇓] и [⇑];
- ввод шестнадцатиричного числа по адресу, которому соответствует курсор.

Выход из режима дампа также клавишей [CSHIFT DEF FN].

Использование буфера страницы экрана

ПК ХОББИТ позволяет сохранить текущее состояние экрана (текст, рисунок, находящийся на экране) в специальной области памяти, называемой *БУФЕРОМ СТРАНИЦЫ ЭКРАНА*. Это дает возможность например, хранить и в любой момент (при использовании режима клавиатуры 0 или 1) вызывать на экран необходимую для работы справочную или любую другую информацию, предварительно занесенную в буфер. (Первоначально в буфере содержится случайная "картинка".)

Необходимо отметить, что содержимое буфера не уничтожается после приведения ПК в исходное состояние при помощи клавиши RESET.

ПРИМЕЧАНИЕ

В буфер заносится образ рабочего экрана, т.е. цвет обрамления не сохраняется.

Существует 2 варианта использования буфера страницы экрана:

— обмен информацией между текущим экраном и буфером;

— копирование текущего экрана в буфер.

И копирование и обмен производятся при помощи нажатия на клавишу PAGE SHADOW MODE, но перед копированием необходимо выполнить теневую функцию H, которая вызывает однократную смену режима использования буфера: вместо обмена (основной режим) производится копирование.

Физически буфер страницы экрана занимает память с нулевого адреса ОЗУ. Из этого, в частности, следует, что он может быть использован, например, для размещения пользовательского монитора, передача управления которому осуществляется теневой функцией R.

Работа ПК в составе однородной сети

Как уже было отмечено, ПК ХОББИТ может использоваться в составе однородной или неоднородной локальной вычислительной сети (ЛВС), процесс межмашинного обмена в которой управляется одним (главным) компьютером. Такая организация связи широко применяется в учебных вычислительных классах и т.п. системах.

Состав комплекса технических средств сети включает в себя главный компьютер (ПЭВМ типа IBM PC или ПК ХОББИТ, укомплектованный НГМД и устройством печати - рабочее место преподавателя (РМП)) и до 31 компьютера ХОББИТ - рабочее место ученика (РМУ).

Любой из компьютеров ХОББИТ может быть как машиной учителя, так и машиной ученика. Режим работы определяется распайкой сетевого штеккера машины учителя.

Все операции в сети происходят только по инициативе учителя. Исключением является режим свободного приема (FREE RECEIVE). В этом режиме машина учителя циклически опрашивает машины учеников, и, в случае наличия у них состояния активности ("поднятая рука"), от ученика принимается BASIC-программа, вся память, экран или содержимое сетевого буфера.

Машиной учителя может быть компьютер IBM PC/XT/AT. В этом случае пересылки происходят между машинами учеников и файлами на диске IBM PC/XT/AT.

Для объединения в сеть каждый ПК имеет по два параллельно включенных разъема, что позволяет соединить "цепочкой" все компьютеры. Порядок соединения не существенен.

Сетевое обеспечение компьютеров ХОББИТ позволяет:

— загрузить BASIC-программы из машины учителя в машины учеников;

— скопировать BASIC-программы из машин учеников в машину учителя с возможностью последующего сохранения их на магнитном диске или на кассетном магнитофоне;

— загрузить образ памяти из машины учителя в машины учеников и, обратно, из машин учеников в машину учителя. При этом на приемной стороне получается "мгновенная копия" состояния передающей машины, включая программу, данные и экран;

— копировать экраны между машинами учителя и учеников;
— переназначить вывод на печатающее устройство в сетевой буфер с последующей пересылкой его содержимого в машину учителя.

Работа сетевого программного обеспечения полностью прозрачна для пользователя, то есть - оно не занимает основное адресное пространство и не может быть испорчено или отключено программой пользователя.

Переход компьютера в режим обслуживания сети и, соответственно, приостановку основного процесса можно обнаружить по изменению цвета *БОРДЕРА* (обрамления экрана).

Цвет бордера означает следующее:

зеленый идет прием информации из сети;

красный идет передача информации в сеть;

черный идет посылка подтверждения приема (квитанции);

синий идет ожидание квитанции;

желтый прием закончился с ошибкой, ожидание повторной передачи

Сетевое программное обеспечение компьютера ХОББИТ может работать в одном из двух режимов:

— машина учителя;
— машина ученика.

Работа сетевого программного обеспечения копьютера ХОББИТ в режиме машины ученика не требует каких-либо действий от оператора. Все действия происходят по инициативе машины учителя, прерывая основной процесс на время, необходимое для приема/передачи информации в сети.

Установка параметров сетевого обеспечения, а также получение информации о состоянии производится из сетевой страницы теневого режима путем нажатия соответствующей буквы.

После перехода в сетевую страницу теневого режима на экран выводится меню сетевого режима - **NETWORK PAGE**. (Наклонная черта "/" показывает, что на экране может находиться одна из объединенных этим знаком альтернатив)

Назначение опций сетевой страницы теневого режима машины ученика:

	NETWORK PAGE MASTER MASHINE	OFF
S	SEND	ON/OFF
R	RECEIVE	ON/OFF
V	LPRINT→NET	ON/OFF
M	MEMORY	ON/OFF
P	PICTURE	ON/OFF
B	BASIC	ON/OFF
L	LETTER	ON/OFF
E	BORDER EFFECTS	ON/OFF
A	ACTIVATE	ON/OFF
N	NUMBER	12
X	PAGE	NET/SYS

SEND состояние данной опции ON означает, что последней операцией в сети для этой машины была передача;

RECEIVE состояние данной опции ON означает, что последней операцией в сети для этой машины был прием;

LPRINT→NET данная опция означает - будет ли вывод, осуществляемый по команде языка BASIC **LPRINT**, пересылаться в сетевой буфер (ON) или в локальный принтер (OFF). После вывода в сетевой буфер в правой части сетевой страницы появляется вертикальный "градусник", на котором зеленым цветом обозначено оставшееся в буфере пространство, а красным - заполненное. Кроме того, если принтер подключен, появляется опция **WRITE**

WRITE вывести содержимое сетевого буфера на принтер.

Следующие четыре опции означают тип пересылаемого/принимаемого блока:

MEMORY вся память и состояние процессора;

PICTURE экран;

BASIC BASIC-программа;

LETTER содержимое сетевого буфера;

BORDER EFFECTS — если данная опция включена, то все посылки по сети сопровождаются изменением цвета бордера, в противном случае - изменение цвета бордера блокируется.

ACTIVATE — включение данной опции означает запрос на обслуживание со стороны машины учителя. Работает только, если машина учителя находится в режиме FREE.

NUMBER — показывает записанный в ПЗУ сетевой номер данной машины (все машины в сети имеют уникальные номера).

Назначение опций сетевой страницы теневого режима машины учителя:

	NETWORK PAGE MASTER MASHINE	ON
S	SEND	ON/OFF
R	RECEIVE	ON/OFF
V	LPRINT–NET	ON/OFF
M	MEMORY	ON/OFF
P	PICTURE	ON/OFF
B	BASIC	ON/OFF
L	LETTER	ON/OFF
F	FREE RECEIVE	ON/OFF
Y	NET LIST	
E	BORDER EFFECTS	ON/OFF
A	ACTIVATE	
N	NUMBER	
X	DEFAULT PAGE	NET/SYS

Большинство опций сетевой страницы машины учителя совпадают с вышеописанными, за исключением:

FREE — данная опция включает циклический опрос всех машин учеников на наличие у них флага ACTIVATE ("поднятая рука"). Номера машин с "поднятой рукой" выводятся на экран. Дальнейшее их обслуживание возлагается на учителя.

NET LIST — выводит на экран список номеров всех машин учеников в сети;

ACTIVATE — включение данной опции означает выполнение установленной операции.

NUMBER — устанавливает сетевой номер машины, с которой будет производиться обмен. Если установить номер равный 00 (это возможно только для передачи), то это будет означать, что операция производится в режиме широковещания, то есть всем одновременно. Установка номера, отсутствующего в сети, индицируется знаками вопроса в мигающем красном прямоугольнике.

Описание тестовой и демонстрационной программ

Программа TEST

Программа TEST предназначена для проверки работоспособности различных функциональных узлов ПК ХОББИТ. С ее прогона целесообразно начинать рабочий день.

Программа TEST работает в BASIC-среде. Загрузка программы, состоящей из двух частей (TST-H90 и t-hob90) производится командой **LOAD "TST-H90"** с ленты или **RUN *"TST-H90"** с дискеты.

По окончании загрузки, на экране появляется МЕНЮ, из которого ясно, что программа TEST состоит из 8 самостоятельных тестов проверки правильности работы клавиатуры, цветовых и звуковых сигналов, дисковода, памяти (ОЗУ и ПЗУ), магнитофона, принтера и джойстика.

Выбор необходимого теста обеспечивается нажатием соответствующей клавиши (**1÷8**), указанной в МЕНЮ. Нажатие клавиши **9** приводит к последовательному выполнению всех тестов. После успешного завершения теста происходит автоматический возврат в МЕНЮ.

Обращаем Ваше внимание на работу в режиме проверки клавиатуры. В нем проверяются все клавиши, кроме клавиши RESET, нажатие которой вызывает, как и всегда, инициализацию ПК. Выйти из теста клавиатуры можно только после того, как проверены все клавиши.

Программа DEMO

Программа DEMO знакомит с возможностями ПК ХОББИТ в области звука, графики и цвета, а также помогает пользователю, впервые севшему за клавиатуру компьютера, понять его основные концепции и приобрести некоторые навыки работы на нем.

Программа DEMO работает в BASIC-среде. Загрузка программы, состоящей из четырех частей (DEMO, ser1, hel1 и keyc4) производится командой **LOAD "DEMO"** с ленты или **RUN *"DEMO"** с дискеты.

Во время загрузки на экране появится красочная заставка. Надпись "НАЖМИТЕ ЛЮБУЮ КЛАВИШУ" служит приглашением к началу работы с программой.

Если Вы будете следовать советам программы, то сначала узнаете общие сведения о ПК ХОББИТ и возможностях демонстрационной программы, а затем под аккомпанимент музыки появится МЕНЮ, в котором перечислены все ее функции. Музыка

будет звучать около 1,5 минут и, если Вы не прервете ее, нажав на любую клавишу, то после окончания ее звучания начнет выполняться первый раздел программы (первое знакомство), обозначенный в МЕНЮ цифрой 1.

Выполнение программы не требует пояснений. Мы советуем Вам, насладившись музыкой, выполнить все предлагаемые программой разделы, а разделы 2÷6 повторять до тех пор, пока не получите хорошей оценки своих навыков.

ЗАМЕЧАНИЕ

Если после нажатия очередной цифры МЕНЮ 1-8, вместо выхода в соответствующий раздел появится надпись:

БЕЗ ПАНИКИ!
Нажатая Вами клавиша не указана в МЕНЮ.
Для окончания работы программы нажмите "**Q**".
Любая другая клавиша возвратит Вас в МЕНЮ

действительно не пугайтесь - Вы, вероятнее всего, нажали нужную клавишу, но нажатие было слишком кратковременным. Повторите свои действия.

ЛИТЕРАТУРА

1. Архитектура ПК ХОББИТ. Руководство программиста
2. Среда обучения LOGO для ПК ХОББИТ. Методические рекомендации учителю
3. Системное программирование в ДОС СР/М для ПК ХОББИТ
4. Особенности реализации FORTH-системы для ПК ХОББИТ
5. Пособие по программированию на языке Си для ПК ХОББИТ
6. Пособие по программированию на языке Паскаль для ПК ХОББИТ

ПРИЛОЖЕНИЯ

BASIC

Словарь ключевых слов

Ниже приводится полное описание всех ключевых слов BASIC'а компьютера ХОББИТ, для каждого из которых указаны **класс**, **формат**, назначение и пример использования.

Каждое ключевое слово относится к одному или нескольким из следующих **классов:**		
класс	описание	пример
команда	ключевое слово, предписывающее компьютеру выполнить некоторое действие немедленно после ввода	**RUN, LOAD**
оператор	ключевое слово, входящее в состав строки программы. Обрабатывается только при выполнении программы	**10 INPUT a,b : PLOT a,b**
функция	ключевое слово, вызывающее вычисление числового или символьного значения. Используется как часть команды или оператора	**PRINT SIN (RND*PI)**
логический оператор	ключевое слово, используемое в командах или операторах для описания логических условий. В BASIC'е ПК ХОББИТ используется три логических оператора:	**AND, OR и NOT**

Формат

описывает синтаксис использования каждого из ключевых слов, т.е. допустимую комбинацию ключевого слова и его параметров, таких как *числа*, *переменные* и *выражения*.

Числа

изменяются от $4*10^{-39}$ до 10^{38}, при этом обеспечивается точность 9–10 цифр.

Переменные

числовые или символьные а также их массивы.

Имя числовой переменной – любая последовательность символов, начинающаяся с буквы (пробелы внутри имени игнорируются; строчные и прописные буквы не различаются).

Значение символьной переменной – строка символов произвольной длины. Имя символьной переменной состоит из одной буквы и знака $.

Числовые массивы могут иметь произвольное количество измерений и произвольный размер. Начальный индекс всегда 1. Имя массива – одна буква; может совпадать с именем простой переменной.

Массив символьных переменных также может иметь произвольное количество измерений и размер. Элемент массива – строка символов фиксированной длины, которая определяется числом, задающим последнюю размерность в операторе **DIM**. Начальный индекс – 1. Имя формируется как и для символьной переменной. Совпадение имени массива символьных переменных и простой символьной переменной не допускается.

Управляющие переменные для **FOR...NEXT** циклов имеют имена длиной в одну букву.

Выражения

числа, переменные и функции, объединенные знаками операций. Выражение имеет числовое, символьное или логическое значение, получаемое в результате его вычисления (выполнения действий, предписываемых знаками и порядком следования – *приоритетом* – операций).

операция	обозначение	приоритет
индексация	( )	12
сечение	(... TO ...)	12
функции	ключевые слова	11
возведение в степень	↑	10
унарный минус	–	9
умножение	*	8
деление	/	8
сложение, конкатенация символьных строк	+	6
вычитание	–	6
операции отношения:		5
равно	=	
не равно	<>	
больше	>	
меньше	<	
больше или равно	>=	
меньше или равно	<=	
логические операции:		
отрицание (НЕ)	NOT	4
дизъюнкция (И)	AND	3
конъюнкция (ИЛИ)	OR	2

Текст программной или командной строки может располагаться в любом месте строки (за исключением номера строки, который должен размещаться в начале строки).

Все операторы, кроме **INPUT**, **DEF FN** и **DATA** могут использоваться и как команды и в программах.

Команда или строка программы может содержать несколько операторов, разделенных двоеточием (:).

При описании формата ключевых слов использованы следующие обозначения:

обозначение	значение	пример
[ ]	обозначение необязательного элемента, который может многократно повторяться	
number	числовая константа, переменная или выражение	27.5 Sum sum*27.5 RND*6
num–var	числовая переменная	summa
integer	числовая константа, переменная или выражение, значение которых округляется до ближайшего целого	
string	символьная константа, переменная или выражение	"W o_r d" a$ a$+"new" a$(3 TO 5)
string–var	символьная переменная	b$
letter	любая строчная или прописная буква	X x M m
letter$	любая строчная или прописная буква, за которой следует знак $	A$ a$
cond	условное выражение или его часть	a=1 OR y>2
statement	любой допустимый оператор языка BASIC	IF a=2 THEN STOP PRINT INK 2;
channel	integer, значение которого лежит в диапазоне 0 ÷ 15	
drive	string, значение которой - строчная или прописная буква от A до C, обозначающая дисковод	
f–name	string длиной до 8 символов	
f–spec	[drive :] f–name	"b : My file"
f–type	пусто, # или одно из ключевых слов: CODE, DATA и SCREEN$	
D	обозначение дискового оператора	
N	обозначение сетевого оператора	

* D оператор / команда

ФОРМАТ *** drive**

ОПИСАНИЕ Назначение дисковода, имя которого указано параметром команды, *рабочим* (т.е. все последующие дисковые операции, для которых имя дисковода опущено, будут выполняться на этом дисководе).

ПРИМЕР Команда *** "a"** установит рабочим дисковод **A**.

ABS функция

ФОРМАТ **ABS number**

ОПИСАНИЕ **ABS** возвращает абсолютную величину числовой переменной, т.е. ее значение без знака

ПРИМЕР Команда **PRINT ABS –13.5** выведет на экран **13.5**.

ACS функция

ФОРМАТ **ACS number**

ОПИСАНИЕ **ACS** вычисляет значение угла в радианах по его косинусу.

ПРИМЕР Команда **PRINT 180/PI*ACS 0.5** выведет на экран **60**, т.е. угол (в градусах), косинус которого равен 0.5.

AND логический оператор / функция

ФОРМАТ
cond AND cond
number AND number
string AND number

Как логический оператор **AND** используется для проверки комбинации из нескольких условий; результат проверки считается истинным в случае истинности всех составляющих условий.

Как функция **AND** выполняет двоичную операцию "**И**" над двумя числовыми или символьной и числовой переменными;

для числовых переменных возвращает первый аргумент, если второй – не 0 и 0 в противном случае;

результат выполнения функции в случае *символьной и числовой переменных* – символьная переменная, если числовая – не 0, иначе – пустая строка.

Обратите внимание, что в BASIC'е истинному условию присваивается значение 1, а ложному – 0; кроме того, любое не нулевое значение считается истинным.

ПРИМЕР

```
60 LET correct=(x=y+z) AND time
70 LET score=score+10*(1 AND correct)
80 LET a$=("Out Of Time Or Not" AND
   NOT correct)+" Correct"
```

Если оба условия в строке **60** – истинны, переменной **correct** присваивается значение **1**, переменная **score** увеличивается на **10**, а **a$** принимает значение "**Correct**"; если же хотя бы одно из условий не выполняется, **correct** имеет значение **0**, score не изменяется, а переменная **a$** становится равной "**Out Of Time Or Not Correct**".

ASN функция

ФОРМАТ **ASN number**

ОПИСАНИЕ **ASN** вычисляет значение угла в радианах по его синусу.

ПРИМЕР Команда **PRINT 180/PI*ASN 0.5** выведет на экран **30**, т.е. угол синус которого равен 0.5.

AT

См. **INPUT, PRINT**

ATN функция

ФОРМАТ **ATN number**

ОПИСАНИЕ **ATN** вычисляет значение угла в радианах по его тангенсу.

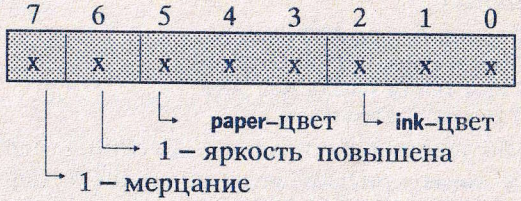
ПРИМЕР Команда **PRINT 180/PI*ATN 1** выведет на экран **45**, т.е. угол (в градусах), тангенс которого равен 1.

ATTR функция

ФОРМАТ **ATTR (integer, integer)**

ОПИСАНИЕ **ATTR** возвращает атрибуты указанной в обращении к **ATTR** символьной позиции экрана, то есть ее **ink**– и **paper**– цвета, а также режим отображения, т.е. яркость и мерцание. Позиция задается двумя целыми числами – номер строки (от 0 до 23) и номер колонки в строке (от 0 до 31). Значение, возвращаемое функцией, лежит в диапазоне от 0 до 255; оно представляет собой сумму: (код **ink**–цвета) + 8×(код **paper**-цвета) + 64×(вы-

вод с повышенной яркостью) + 128×(вывод с мерцанием), т.е. в двоичной форме:



ПРИМЕР Пусть символ в 11 строке, 16 колонке выводится черным **ink**–цветом (код 0) на белом **paper**–цвете (код 7) и с повышенной яркостью без мерцания, тогда команда **PRINT ATTR (11,16)** выведет на экран **120** (0 + 8×7 + 64 + 0)

BEEP

оператор / команда

ФОРМАТ **BEEP number, number**

ОПИСАНИЕ **BEEP** выводит на динамик компьютера ноту указанной высоты и продолжительности. Продолжительность (в секундах) указывается первым параметром и может изменяться от 0 до 10, высота - второй параметр - изменяется от -60 до 69 и измеряется в полутонах ниже (параметр отрицательный) или выше (положительный) ноты 'До' первой октавы.

ПРИМЕР Команда **BEEP 0.5,1** выведет на динамик ноту До# в течение полу-секунды.

BIN

функция

ФОРМАТ **BIN [1][0]**

ОПИСАНИЕ **BIN** возвращает десятичное значение своего двоичного аргумента (т.е. аргумента, представленного в двоичной системе счисления), наибольшее допустимое значение которого соответствует числу 2^{16}-1 (65535). Наиболее часто **BIN** используется совместно с **POKE** и **USR** для создания графических символов, определяемых пользователем (**UDG**-символов), при этом 1 определяет **ink**-цвет пикселя (точки) в матрице 8×8 изображения символа, а 0 - **paper**-цвет.

ПРИМЕР Команда **PRINT BIN 11111100** выведет на экран **252** - десятичное значение двоичного числа.

BORDER

оператор / команда

ФОРМАТ **BORDER integer**

ОПИСАНИЕ **BORDER** указывает цвет окаймления (рамки) рабочей области экрана. Цвет задается кодом от 0 до 7:

0	черный	(**black**)	4	зеленый	(**green**)
1	синий	(**blue**)	5	голубой	(**cyan**)
2	красный	(**red**)	6	желтый	(**yellow**)
3	фиолетовый	(**magenta**)	7	белый	(**white**)

Кроме этого **BORDER** определяет цвет нижней области экрана (область ввода команд).

ПРИМЕР Команда **BORDER RND*7** задаст случайно выбранный цвет окаймления экрана.

BRIGHT

оператор / команда

ФОРМАТ **BRIGHT integer [;]**

ОПИСАНИЕ **BRIGHT** определяет яркость отображения символов на экране – нормальную или повышенную:
BRIGHT 1 устанавливает повышенную яркость для всех следующих эператоров вывода;
BRIGHT 8 сохраняет текущее значение яркости символьных позиций экрана для всех следующих операторов вывода;
BRIGHT 0 устанавливает нормальную яркость для всех следующих операторов вывода.
BRIGHT можно также использовать в операторах вывода **PLOT, DRAW, CIRCLE, PRINT** и **INPUT**; в этом случае **BRIGHT** действует локально – только на эти операторы – и должен указываться до выводимых данных и отделяться от них точкой с запятой.

ПРИМЕР

```
60 PRINT BRIGHT 1; "Warning!"
```

CAT *

оператор / команда [D]

ФОРМАТ **CAT *[drive]**

ОПИСАНИЕ **CAT *** выводит на экран каталог диска текущего или указанного параметром дисковода.

ПРИМЕР Команда **CAT * "B:"** выведет на экран каталог диска дисковода **B**.

CHR$ — функция

ФОРМАТ **CHR$ integer[;][+]**

ОПИСАНИЕ **CHR$** позволяет по коду получить любой символ (букву, цифру, знак, ключевое слово и **UDG**-символ — код от 32 до 255) или управляющий код (от 1 до 31) из набора символов компьютера. (Полный набор символов приведен в приложении).

ПРИМЕР Команда **PRINT CHR$ 216** выведет на экран **CIRCLE** — обозначение ключевого слова, имеющего код 216.

Использование управляющих кодов:

CHR$ 6 — перенос позиции вывода следующего символа либо в середину текущей, либо в начало следующей строки;

CHR$ 8 — сдвиг позиции вывода на шаг назад;

CHR$ 13 — перенос вывода на новую строку;

CHR$ 16 + CHR$ n — изменение **ink**–цвета следующих выводимых символов (**n** — код цвета);

CHR$ 17 + CHR$ n — изменение **paper**–цвета;

CHR$ 18 + CHR$ n — изменение режима мерцания — **flash** — (**n=0** — без мерцания, **n=1** — с мерцанием, **n=8** — не изменять **flash**-режим выводимой позиции);

CHR$ 19 + CHR$ n — изменение режима яркости — **bright** — вывода (значения **n** аналогичны предыдущему случаю);

CHR$ 20 + CHR$ n — изменение режима инверсии — **inverse** — вывода (**n=0** — нормальный, **n=1** — инверсный вывод);

CHR$ 21 + CHR$ n — изменение режима наложения — **over** — вывода (**n=0** — без наложения **n=1** — вывод с наложением);

CHR$ 22 + CHR$ n + CHR$ m — перенос позиции вывода в строку **n** и колонку **m** (**at**–управление);

CHR$ 23 + CHR$ n + CHR$ 0 — перенос позиции вывода в текущей строке в колонку **n** (**tab**–управление).

ПРИМЕР Команда **PRINT CHR$ 16 + CHR$ 3 + CHR$ 17 + CHR$ 6 + CHR$ 18 + CHR$ 1 + "Hello!"** выведет мерцающее красное и желтое слово **Hello!**

Команда **PRINT CHR$ 22; CHR$ 11; CHR$ 16; CHR$ 42** выведет звездочку в центре экрана.

CIRCLE — оператор / команда

ФОРМАТ **CIRCLE [statement ;] integer, integer, integer**

ОПИСАНИЕ **CIRCLE** чертит на экране окружность с центром, координаты **x,y** которого заданы двумя первыми аргументами, а радиус окружности - последним. Координаты и радиус измеряются в пикселях, т.е. используется графика высокого разрешения. Окружность чертится **ink**-цветом; для локального изменения текущих цветов и режима вывода могут быть использованы соответствующие операторы, включенные в **CIRCLE**.

ПРИМЕР Команда **CIRCLE 128,88,87** начертит на экране окружность максимально возможного радиуса.

CLEAR — оператор / команда

ФОРМАТ **CLEAR [number]**

ОПИСАНИЕ **CLEAR** удаляет определенные в текущий момент переменные (включая массивы), освобождая всю память, доступную для BASIC-программ; выполняет **CLS**, т.е. очищает экран, устанавливает **PLOT**—позицию в левый нижний угол экрана, и **RESTORE** — устанавливает указатель списка данных на первый элемент в первом операторе **DATA** программы; очищает **GOSUB**—стек.

Кроме этого **CLEAR** м.б. использован для переопределения т.н. "системной переменной" **RAM–Top**, определяющей верхнюю границу памяти BASIC-системы (первоначально она имеет максимально возможное значение).

Отметьте, что текущее значение **RAM–Top** м.б. выяснено так: **PRINT PEEK 23730 + 256*PEEK 23731**

ПРИМЕР Команда **CLEAR PEEK 23730 + 256*PEEK 23731 – 100** удалит все ранее определенные значения переменных, очистит экран и зарезервирует (защитит от использования BASIC-системой) 100 байт оперативной памяти.

CLOSE# — оператор / команда

ФОРМАТ **CLOSE# channel**

ОПИСАНИЕ **CLOSE#** "закрывает" открытый командой **OPEN#** канал, т.е. возвращает стандартное назначение

потока ввода/вывода, связанное с данным каналом (подробнее - см. **OPEN#**).
ВНИМАНИЕ: "закрытие" неназначенного канала приводит к непредсказуемым результатам.

CLOSE# & [N]

оператор / команда

ФОРМАТ **CLOSE# &**

ОПИСАНИЕ **CLOSE# &** "отключает" (на программном уровне) компьютер от локальной сети учебного класса.

CLOSE# * [D]

оператор / команда

ФОРМАТ **CLOSE# * channel**

ОПИСАНИЕ **CLOSE#** "закрывает" открытый командой **OPEN# *** файл последовательного или произвольного доступа.
ВНИМАНИЕ: если не "закрыть" вновь созданный или модифицированный файл, информация в нем может быть утеряна.

CLS

оператор / команда

ФОРМАТ **CLS**

ОПИСАНИЕ **CLS** очищает рабочую область экрана, окрашивая ее в цвет, установленный последней командой **PAPER**.

ПРИМЕР

```
100 IF a$ = "No" THEN CLS
```

CODE

функция

ФОРМАТ **CODE string**

ОПИСАНИЕ **CODE** возвращает код первого символа своего символьного аргумента в кодовой таблице компьютера (см. приложение). Если аргумент - пустая строка (""), то **CODE** возвращает 0.
CODE может быть частью команд **LOAD / SAVE / VERIFY**. Формат и правила использования **CODE** в этом случае отличаются от вышеприведенных (см. описание команд **LOAD / SAVE / VERIFY**).

ПРИМЕР Следующие операторы проверяют, является ли первый символ переменной **a$** буквой:

```
20 LET ch = CODE a$ : IF ch > 64 AND ch < 91
    OR ch > 96 AND ch <123 THEN PRINT "Letter"
```


CONTINUE

команда

ФОРМАТ **CONTINUE**

ОПИСАНИЕ **CONTINUE** возобновляет выполнение программы, прерванное по какой-либо причине (ошибка, оператор **STOP** или команда **BREAK**). В первом случае выполнение возобновляется с оператора, вызвавшего остановку, в двух последних – со следующего.

COPY

оператор / команда

ФОРМАТ **COPY**

ОПИСАНИЕ **COPY** вызывает распечатку на принтере копии первых 22-х строк экрана (точки, имеющие **ink**-цвет, печатаются, а **paper**-цвет - нет).
Вывод на печать может быть прерван нажатием клавиши **BREAK**.

[D] COPY *

оператор / команда

ФОРМАТ **COPY * [S][B][f–spec][, f–spec][f–type]**

ОПИСАНИЕ **COPY *** производит копирование файлов, группы файлов и целых дисков на одном или нескольких дисководах.

Обычное копирование

COPY * f–specnew **, f–spec**old **f–type**old – копирование файла с именем, указанным в **f–spec**old и типом **type**old в файл, имя и диск которого указаны в **f–spec**new.

Копирование на разных дисках в одном дисководе

COPY * S f–name f–type – копирование файла с именем **f–name** и типом **f–type** на разных дисках с использованием *рабочего* дисковода.

Создание копии диска на одном дисководе

COPY * B – создание полной копии диска (при этом старое содержимое диска-копии пропадает) на *рабочем* дисководе.

Сообщения команды **COPY ***

- **Insert SOURCE disk then press Y** – запрос на установку диска, с которого производится копирование, в *рабочий* дисковод; нажатие клавиши **Y** продолжает процесс копирования; – возникает при копировании на разные диски в одном дисководе.
- **Insert DESTINATION disk then press Y** – запрос на уста-

новку диска, на который производится копирование.

- **f–spec f–type File exists Overvrite (Y/N)?** — сообщение о том, что файл с именем и типом, указанными в команде копирования в качестве имени и типа нового файла, уже существует. При вводе **Y** в ответ на этот запрос, существующий файл будет заменен новым; при ответе **N**, операция копирования для указанного файла производиться не будет.

Правила формирования имени файла

- имена файлов могут состоять из прописных и строчных букв латинского алфавита (при этом прописная буква считается отличной от строчной), цифр, знаков препинания и пробелов;
- не допускается наличие на одном диске двух файлов одного типа с одинаковыми именами.
- допускается использование символа * вместо имени файла; при этом:
  - операция копирования будет выполняться для всех файлов на диске указанного в **f–spec** дисковода;
  - в качестве нового имени файла также должен использоваться символ *;
  - указание типа файла не допускается.

ПРИМЕРЫ

COPY * "a:LOGO","b:Logo" CODE - копирование файла **Logo** типа **CODE** с диска дисковода **B** на диск дисковода **A** под именем **LOGO** (тип файла копии всегда совпадает с типом исходного файла).

COPY * "B:BRAIN","BRAIN" - копирование BASIC-программы **BRAIN** с диска рабочего дисковода на диск дисковода **B** без изменения имени.

COPY * "FONT","font" CODE - дублирование файла **font** типа **CODE** на диске рабочего дисковода под именем **FONT**.

COPY * "B:*","A:*" - копирование *всех* файлов с диска дисковода **A** на диск дисковода **B**. Файлы, существовавшие на диске дисковода **B** не уничтожаются.

COPY * s "text" CODE - копирование файла **text** типа **CODE** с одного диска на другой с использованием рабочего дисковода.

COPY * b - получение точной копии одного диска на другом с использованием рабочего дисковода.

COS

функция

ФОРМАТ **COS number**

ОПИСАНИЕ **COS** возвращает значение косинуса угла, заданного в радианах.

ПРИМЕР Команда **PRINT COS(60*PI/180)** выведет на экран **0.5** - косинус угла, равного 60 градусам.

DATA

оператор

ФОРМАТ

DATA number[, number][, string]
DATA string[, number][, string]

ОПИСАНИЕ **DATA** задает список данных в теле программы. Элементы списка могут быть использованы как значения числовых или символьных переменных, а присваивание переменным значений из **DATA**-списка осуществляется при помощи оператора **READ** в том порядке, в каком они указаны в тексте программы. Этот порядок может быть изменен оператором **RESTORE**. В качестве элементов списка могут быть использованы числовые или символьные константы, определенные переменные (т.е. переменные, которым присвоено какое-либо значение) или выражения. Оператор **DATA** может находиться в любом месте программы.

DATA может быть частью команд **LOAD / SAVE / VERIFY**. Формат и правила использования **DATA** в этом случае отличаются от вышеприведенных (см. описание команд **LOAD / SAVE / VERIFY**).

ПРИМЕР

Программа

```
10 FOR n = 1 TO 2
20 READ x,a$
30 PRINT a$,x;" days"
40 NEXT n
50 DATA 31,"Jan",28,"Feb"
```

выведет на экран

```
Jan      31 days
Feb      28 days
```

Если заранее определить (т.е. произвести присвоение раньше, чем будет выполняться строка 20) значения переменных **d** и **m$**: **LET d=31** и **LET m$="Jan"**, то строка 50 может быть записана так:

```
50 DATA d,m$,d–3,"Feb"
```

DEF FN — оператор

ФОРМАТ

```
DEF FN letter ( [ letter ][ , letter ] ) = num–exp
DEF FN letter$ ( [ letter$ ][ , letter ][ , letter$ ] ) = string–exp
```

ОПИСАНИЕ

DEF FN позволяет задать (определить) функцию, отсутствующюю в составе функций BASIC'a. Вновь определенной функции присваивается однобуквенное имя (буква и знак **$** для символьной функции), которое используется в дальнейшем для обращения к этой функции (при помощи **FN**). В определении функции можно использовать параметры, значения которых указываются при обращении к ней, и переменные (должны быть определены в момент обращения).

ПРИМЕР

Программа

```
10 DEF FN a(x,y)=SQR (x↑2+y↑2)
20 DEF FN b ()=INT (s+0.5)
30 DEF FN c$ (x$, y,z)=x$ (y TO z)
40 LET s=FN a (7,8)
50 PRINT FN c$ ("NearEqual",1,4), FN b ()
```

выведет на экран

```
Near      10
```

– округленную длину гипотенузы треугольника с катетами длиной 7 и 8

DIM — оператор

ФОРМАТ

```
DIM letter ( number [ , number ] )
DIM letter$ ( number [ , number ] )
```

ОПИСАНИЕ

DIM используется для определения массива (упорядоченного списка числовых или символьных переменных, элемент которого может быть получен при помощи индекса (номера) элемента в списке). Имя массива должно состоять из одной буквы (буквы и знака **$** для символьного массива) и для числового массива может совпадать с именем простой переменной. Количество измерений массива - произвольное (ограничивается только размером памяти компьютера). Обращение к отдельному элементу массива производится по имени массива и значению индекса (индексов) требуемого элемента.

При объявлении массива ("выполнении" оператора **DIM**), его элементам присваивается значение 0 для числового массива и " " (пустая строка) для символьного.

Обратите внимание, что элементы символьного массива, в отличие от простых символьных переменных, имеют фиксированную длину, указываемую последней в списке измерений оператора **DIM**, поэтому при присваивании элементам символьного массива символьных констант или переменных, их значение или усекается или добавляется справа пробелами до заданной длины.

ПРИМЕР

Объявление массива:

```
10 DIM a(2,10)
40 DIM c$(10)
```

Обращение к элементам массива:

```
70 PRINT a(1,9), c$(6)
```


DRAW — оператор / команда

ФОРМАТ

```
DRAW [ statement ; ] integer, integer [ , integer ]
```

ОПИСАНИЕ

DRAW чертит на экране (в режиме высокого разрешения) прямую линию или дугу. Два первых аргумента определяют координату **x,y** конечной точки линии как приращение (положительное или отрицательное) к координате начальной точки, за которую принимается конечная точка предыдущего графического оператора (**DRAW, CIRCLE, PLOT**) или левый нижний угол экрана(0,0), если эти операторы не выполнялись.

Третий параметр определяет кривизну линии, т.е. угол (в радианах) между радиусами, образующими сектор, дугу которого выводит **DRAW**.

ПРИМЕР

Следующая программа рисует треугольник:

```
10 PLOT 127,150
20 DRAW 70,–100
30 DRAW –140,0
40 DRAW 70,100
```

Если теперь добавить в операторы **DRAW** третий параметр, равный 1 или –1, стороны "треугольника" станут вогнутыми или выпуклыми.

D ERASE * — оператор / команда

ФОРМАТ

```
ERASE * f–spec  f–type
```

ОПИСАНИЕ **ERASE *** удаляет файл, имя которго указано в **f-spec**, типа **f-type**, с диска, находящегося в указан ном в **f-spec** дисководе. При этом, если удаляемый файл был последним в каталоге диска (см.команды **CAT ***, **LIST ***), освободившееся в результате удаления пространство на диске может быть использовано. В противном случае (файл не последний) увеличивается число удаленных файлов в каталоге диска, а место на диске не освобождается. Для использования пространства, занятого удаленными файлами, необходимо воспользоваться командой **MOVE ***.

EXP

Функция

ФОРМАТ **EXP number**

ОПИСАНИЕ **EXP** - математическая функция, возвращающая указанную параметром степень числа **e**.

ПРИМЕР Команда **PRINT EXP 1** выведет **2.7182818** — приближенное значение **e**.

FLASH

оператор / команда

ФОРМАТ **FLASH integer [;]**

ОПИСАНИЕ **FLASH** вызывает "мерцание" знакоместа, т.е. периодическую смену **ink**- и **paper**-цветов. Аргумент **FLASH** может принимать те же значения, что и аргумент **BRIGHT** с тем же смыслом. **FLASH** может быть также включен в другие операторы вывода — **PRINT, PLOT, DRAW, INPUT, CIRCLE**. В этом случае **FLASH** действует локально — только на эти операторы — и должен указываться до их параметров и отделяться от них точкой с запятой.

ПРИМЕР

```
120 PRINT FLASH 1; INK 2; PAPER 6; "Warning"
```

FN

Функция

ФОРМАТ **FN letter ([number][, number])**
FN letter$ ([string][, number][, string])

ОПИСАНИЕ **FN** вызывает функцию, определенную ранее в программе при помощи **DEF FN**. Рекурсия, т.е. использование в составе параметра функции той же самой функции, не допускается.

ПРИМЕР см. в описании **DEF FN**

FOR

оператор / команда

ФОРМАТ

```
FOR letter=number TO number [ STEP number ]
NEXT letter
```

ОПИСАНИЕ **FOR** всегда используется с ключевыми словами **TO** и **NEXT** для организации **FOR–NEXT** циклов. Эта структура позволяет выполнять часть программы заданное число раз. Буква, используемая в **FOR–NEXT** конструкции — имя управляющей переменной цикла. Этой переменной присваивается начальное значение, указанное выражением, стоящим справа от знака равенства и проверяется, что оно не больше (не меньше, если **STEP** указан и имеет отрицательное значение) конечного значения, указанного справа от **TO**. Если это условие истинно — выполняются операторы тела цикла, то есть операторы, расположенные между **FOR** и **NEXT. NEXT** вызывает изменение управляющей переменной на величину, указанную в **STEP** (или на 1, если **STEP** опущен), и повторение описанной выше последовательности действий, начиная с проверки значения управляющей цикла. Если условие ложно, следующим выполняется оператор, стоящий за **NEXT**.

Допускаются вложенные циклы — т.е. в составе операторов тела одного цикла может быть использован другой цикл, который должен быть закончен (т.е. "закрыт" оператором **NEXT**). Для того, чтобы различать, к какому оператору **FOR** ... относится оператор **NEXT**, за **NEXT** должно быть указанано имя управляющей переменной цикла, который он "закрывает". Циклы **FOR–NEXT** могут иметь любую глубину вложения, т.е. любое количество циклов может быть вложено друг в друга.

ПРИМЕР

Программа

```
10 FOR n=0 TO 5 STEP 2
20 PRINT n*n,
30 NEXT n
40 PRINT n
```

выведет на экран

```
 0      4
 16     6
```

FORMAT & [N] оператор / команда

ФОРМАТ **FORMAT & integer, integer, integer**

ОПИСАНИЕ **FORMAT &** посылает по сети в компьютер, сетевой номер которого указан первым параметром, spy-программу, начальный адрес и длина которой заданы вторым и третьим параметрами.

spy-программа должна быть написана в машинных кодах и подчиняться специальным требованиям.

FORMAT * [D] оператор / команда

ФОРМАТ **FORMAT * string**

ОПИСАНИЕ **FORMAT *** производит разметку и формирование каталога – *форматирование* – диска, находящегося в рабочем дисководе. В качестве параметра указывается имя (метка) диска – строка, длиной до 8 символов. Если первый символ имени – знак **$** – отформатирована будет только одна сторона диска, в противном случае – обе стороны. По окончании процесса форматирования на экран выводится метка диска, количество отформатированных секторов и их максимально возможное количество. Если первое число меньше второго – на диске есть дефектные участки и им лучше не пользоваться или попытаться еще раз отформатировать.

Для чего нужно форматировать диск

В отличие от магнитной ленты, не требующей предварительных действий перед записью на нее какой-либо информации, новый или использовавшийся на компьютере другого типа, диск должен быть условно разделен – размечен – на *дорожки*, а дорожки – на *сектора*. Только после этого компьютер сможет использовать этот диск.

ПК ХОББИТ работает с двумя сторонами диска, каждая из которых разбита на 80 дорожек по 16 256-байтных секторов, т.е. на диске помещается 2560 секторов, при этом одна (нулевая) дорожка занята под каталог диска. Таким образом на диске доступно для использования 2544 сектора (636 Кбайт).

ПРИМЕР

Команда **FORMAT * "My disk" : PAUSE 0** разметит обе стороны диска, находящегося в рабочем дисководе и даст ему метку (имя) **My disk**. Команда **PAUSE 0** сохраняет на экране сообщение о количестве отформатированных секторов до нажатия какой-либо клавиши.

GOSUB

оператор / команда

ФОРМАТ **GOSUB integer**

ОПИСАНИЕ **GOSUB** вызывает выполнение подпрограммы – последовательности операторов, начиная со строки с указанным в **GOSUB** номером и кончая оператором **RETURN**, после чего продолжается выполнение следующих за **GOSUB** операторов программы. Вызовы подпрограмм могут быть как вложенными, так и рекурсивными (вызов подпрограммы из самой себя).

ПРИМЕР

Программа

```
 10 FOR n=1 TO 3
 20 GOSUB 100
 30 NEXT n
 40 STOP
100 PRINT "n = ";n,
110 RETURN
```

Выведет на экран

```
n=1       n=2
n=3
```

Обратите внимание на оператор **STOP** в строке 40. Он необходим, чтобы избежать ошибочного входа (не через оператор **GOSUB**) в подпрограмму.

GOTO

оператор / команда

ФОРМАТ **GOTO integer**

ОПИСАНИЕ **GOTO** вызывает переход программы на строку с номером, не меньшим указанного в операторе или, если такой в программе нет, завершение выполнения программы. **GOTO** можно использовать как команду для запуска программы с определенной строки без предварительной очистки экрана, которую выполняет оператор **RUN**.

ПРИМЕР

```
GOTO 100
```

GOTO * [D]

оператор / команда

ФОРМАТ **GOTO * f–spec CODE**

ОПИСАНИЕ **GOTO *** восстанавливает состояние ПК, сохраненное в файле **f–spec** с помощью функции **MAGIC OPTION** теневого меню.

ВНИМАНИЕ: иногда выполнение этой команды не приведет к восстановлению сохраненного состояния компьютера. Это связано с тем, что в момент сохранения состояния ПК находился *во втором режиме прерываний*. В этом случае необходимо переименовать (см.команду **NEW ***) файл **f–spec**, так, чтобы первым символом имени был знак **$**.

IF

оператор / команда

ФОРМАТ
```
IF number THEN statement[ : statement ]
IF cond THEN statement[ : statement ]
```

ОПИСАНИЕ **IF** всегда используется с ключевым словом **THEN** для принятия решения, влияющего на дальнейший ход программы. Если условие, указанное после **IF** – истинно, или числовое выражение не равно нулю, выполняются операторы, расположенные за **THEN**, иначе происходит переход к обработке операторов следующей строки программы.

ПРИМЕР
```
80 IF x THEN GOTO 100
90 IF a$="No" THEN PRINT "The End" : STOP
100 PRINT "Continue . . ."
```

Выполнение оператора **IF** в строке 80 вызовет передачу управления на оператор **PRINT** в строке 100, если переменная **x** не равна 0. Иначе выполняется оператор в строке 90, который в свою очередь проверяет значение переменной **a$** и если оно равно **"No"**, выводит на экран **"The End"** и останавливает программу, иначе программа продолжает выполняться со строки 100.

IN

функция

ФОРМАТ **IN number**

ОПИСАНИЕ **IN** считывает байт из порта, адрес которого (от 0 до 65535) указывает аргумент функции; полученная величина определяется состоянием устройства ввода/вывода (например клавиатуры), подключенного к порту.

Так, клавиатура подключается при помощи восьми портов с адресами 65278, 65022, 64510, 63486, 61438, 57342, 49150 и 32766, для каждого из которых **IN** может возвращать одно из 5 значений: 175, 183, 187, 189 или 190 - в зависимости от того, какая клавиша нажата.

ПРИМЕР
```
LET a = IN 65278
```


INK

оператор / команда

ФОРМАТ **INK integer[;]**

ОПИСАНИЕ **INK** задает цвет "переднего плана", которым изображаются символы, чертятся точки, линии и кривые. Цвет задается кодом от 0 до 9:

0 черный (**black**)	1 синий (**blue**)
2 красный (**red**)	3 фиолетовый (**magenta**)
4 зеленый (**green**)	5 голубой (**cyan**)
6 желтый (**yellow**)	7 белый (**white**)
8 не меняет **ink**-цвет знакоместа, куда будет производиться вывод	9 наиболее контрастный к **paper**-цвету

Когда **INK** используется как самостоятельный оператор, цвет устанавливается на весь последующий вывод, т.е. глобально. **INK** можно также использовать в операторах вывода **PLOT**, **DRAW**, **CIRCLE**, **PRINT** и **INPUT**; в этом случае **INK** действует локально – только на эти операторы – и должен указываться до выводимых данных и отделяться от них точкой с запятой.

ПРИМЕР
```
40 INK 1: PAPER 7
50 PRINT "This is circle"
60 CIRCLE INK 4; 128, 88, 87
```

Этот фрагмент программы выведет синий текст на белом фоне – глобальное назначение, а затем начертит зеленым цветом (локальное назначение) окружность.

INKEY$

функция

ФОРМАТ **INKEY$**

ОПИСАНИЕ **INKEY$** возвращает символ, выдаваемый клави-

шей, нажатой в данный момент. Если никакая клавиша не нажата, **INKEY$** возвращает пустую строку (" "). В отличие от **INPUT**, **INKEY$** не ждет нажатия, а переходит сразу на следующий оператор. Поэтому эта функция обычно используется внутри цикла, который повторяется до тех пор, пока не нажата нужная клавиша.

ПРИМЕР

```
130 IF INKEY$ = "N" THEN STOP
```

INPUT

оператор / команда

ФОРМАТ

```
INPUT [ prompt ][ ; ][ , ][ ' ] num-var
INPUT [ prompt ][ ; ][ , ][ ' ] string-var
INPUT [ prompt ][ ; ][ , ][ ' ] LINE string-var
prompt = [ ( string ) ][ AT integer,integer ][ statement ][ ; ][ , ][ ' ]
```

ОПИСАНИЕ

INPUT позволяет вводить данные во время выполнения программы. На нижнюю строку выводится строка-подсказка и компьютер ждет, пока оператор введет данные, которые (по нажатию **ENTER**) присваиваются указанной в **INPUT** переменной. **INPUT** может содержать больше одной переменной и может включать операторы изображения, такие, как **FLASH**, **BRIGHT**, **PAPER** и др.

Все переменные и выражения, которые входят в текст подсказки, должны быть заключены в скобки.

Вывод начинается с начала нижней строки, а затем изображение сдвигается вверх, если выводится более одной строки.

AT может использоваться в операторе **INPUT** так же, как в **PRINT**.

Как остановить выполнение **INPUT**

Если за **INPUT** следует числовая переменная - ввод **STOP** прерывает выполнение программы; если переменная строковая - необходимо удалить кавычку (") и затем ввести **STOP** для останова программы.

Использование **LINE** в операторе **INPUT** позволяет избежать вывода символьных кавычек при запросе ввода символьной переменной.

ПРИМЕР

```
80 INPUT INK 2; "What is your name?"; LINE n$,
   ("How old are you ," + n$ + "?"); age
```


INT

функция

ФОРМАТ

```
INT number
```

ОПИСАНИЕ

INT возвращает наибольшее целое, не превосходящее значения аргумента.

ПРИМЕР

Команда

```
PRINT INT 45.67, INT -7.66
```

выведет на экран

```
45      -8
```

INVERSE

оператор / команда

ФОРМАТ

```
INVERSE integer[ ; ]
```

ОПИСАНИЕ

INVERSE вызывает смену **ink**-цвета на **paper**-цвет и наоборот для всех последующих операторов вывода: **INVERSE 1** вызывает использование инверсных цветов в следующих операторах **PRINT** и **INPUT**; **INVERSE 0** восстанавливает обычные цвета **ink** и **paper**.

INVERSE может быть включен в операторы вывода так же, как **INK**. Однако, если он используется в **CIRCLE**, **PLOT** или **DRAW**, **INVERSE 1** вызывает только смену **ink**-цвета на **paper**-цвет, так что изображение, выводимое этими операторами, становится невидимым.

ПРИМЕР

```
PRINT INVERSE 1; "Warning!"
```

LEN

функция

ФОРМАТ

```
LEN string
```

ОПИСАНИЕ

LEN возвращает количество символов в своем символьного аргументе.

ПРИМЕР

Следующая строка программы

```
120 INPUT a$: IF LEN a$>9 THEN GOTO 120
```

позволяет вводить строки, длиной не более 9 символов.

LET

оператор / команда

ФОРМАТ

```
LET num-var = number
LET string-var = string
```

ОПИСАНИЕ

LET используется для присвоения значения переменной.

Заметим, что простые переменные не определены, пока им не присвоены значения операторами **LET**, **READ** или **INPUT**, а элементы массива

инициализируются нулем (числовые массивы) или пустой (нулевой) строкой (смотри **DIM**).

ПРИМЕР
```
60 LET x=x+1
80 LET a$="Correct"
```

LINE

Смотри **INPUT, SAVE**

LIST

оператор / команда

ФОРМАТ **LIST [integer]**

ОПИСАНИЕ **LIST** выводит на экран текст (листинг) программы с первой строки или (если используется с аргументом) со строки, номер которой не меньше значения аргумента.
Отметьте, что строка, с которой начался вывод листинга, становится текущей строкой для редактирования.

ПРИМЕР
```
LIST 100
```

LIST * [D]

оператор / команда

ФОРМАТ **LIST * [drive]**

ОПИСАНИЕ **LIST *** выводит на экран расширенный каталог диска текущего или указанного параметром дисковода.
Кроме информации, выводимой командой **CAT ***, (количество файлов, количество удаленных файлов на диске, свободное пространство диска, имена, типы и длины файлов в секторах), выводится:
для BASIC-программ – длина, длина с переменными и строка автозапуска;
для файлов остальных типов – начальный адрес и длина.

LLIST

оператор / команда

ФОРМАТ **LLIST [integer]**

ОПИСАНИЕ **LLIST** аналогичен оператору **LIST**, но вывод листинга производится на принтер.

LN

Функция

ФОРМАТ **LN number**

ОПИСАНИЕ **LN** возвращает значение натурального (т.е. по основанию **e**) логарифма аргумента. Эта функция,

обратная **EXP**. Значение аргумента должно быть больше 0.

ПРИМЕР
```
60 LET x=LN y
```

LOAD

оператор / команда

ФОРМАТ **LOAD string**

ОПИСАНИЕ **LOAD** загружает программу с указанным именем с магнитной ленты в память компьютера, при этом текущая (находящаяся в данный момент в памяти) программа и переменные теряются. Имя файла (программы) может быть длиной до 10 символов, при этом прописные и строчные буквы в имени различаются. Если в качестве имени использована пустая строка, будет загружена первая встреченная на ленте программа.

ПРИМЕР
```
LOAD "filename"
```

LOAD * [D]

оператор / команда

ФОРМАТ **LOAD * f–spec [CODE [integer] [, integer]]**
[DATA letter [$] ()] [SCREEN$]

ОПИСАНИЕ Аналогично командам из серии **LOAD**, но загрузка производится не с ленты, а из файла на диске, указанного в **f–spec** дисковода.
ВНИМАНИЕ: использование в качестве имени пустой строки не допускается.

LOAD & [N]

оператор / команда

ФОРМАТ **LOAD & integer**

ОПИСАНИЕ **LOAD &** загружает из компьютера, сетевой номер которой указан параметром, BASIC-программу и переменные. В отличие от аналогичной функции теневого меню, работает как на компьютере учителя, так и ученика.

LOAD CODE

оператор / команда

ФОРМАТ **LOAD string CODE [integer] [, integer]**

ОПИСАНИЕ **LOAD CODE** используется для загрузки в область памяти информации, отличной от BASIC-программы (это может быть, например, изображение на экране или графические символы, определяемые пользователем). Имя файла, содержащего эту информацию, подчиняется тем же

ограничениям, что и имя программы (см. LOAD). Необязательные числовые аргументы определяют начальный адрес памяти, куда будет записана считанная с ленты информация, и ее длину (число байт). Если один или оба этих аргумента опущены, используется те же адрес и длина, которые были использованы при записи (см. SAVE CODE).

ПРИМЕР

```
LOAD "picture" CODE 16384,6912
```

запишет в память (ее экранную область) изображение, сохраненное на ленте под именем "picture". То же самое может быть выполнено командой LOAD SCREEN$.

LOAD DATA — оператор / команда

ФОРМАТ

```
LOAD string DATA letter[$]()
```

ОПИСАНИЕ

LOAD DATA используется для загрузки массивов с ленты, записанных при помощи SAVE DATA.
Имя файла, данное массиву на ленте, подчиняется тем же правилам, что и имя программы в LOAD. Буква или буква и знак $, следующие за DATA, - это имя, которое должно быть присвоено загруженному с ленты массиву. Любой массив, находящийся в данный момент в памяти и имеющий то же самое имя, удаляется. Заметьте, что в случае символьных массивов, любая символьная переменная, находящаяся в данный момент в памяти и имеющая то же самое имя, также удаляется.

ПРИМЕР

```
270 LOAD "numbers" DATA n()
300 LOAD "names" DATA n$
```

LOAD SCREEN$ — оператор / команда

ФОРМАТ

```
LOAD string SCREEN$
```

ОПИСАНИЕ

LOAD SCREEN$ загружает экранное изображение, предварительно сохраненное на магнитной ленте командой SAVE SCREEN$. За LOAD должна быть указана символьная переменная - имя файла.

ПРИМЕР

```
LOAD "picture" SCREEN$
```

LPRINT — оператор / команда

ФОРМАТ

```
LPRINT [TAB integer;][AT integer, integer;][CHR$(integer);]
       [statement;][number][string][;][,][']
```


ОПИСАНИЕ

LPRINT выводит данные на печатающее устройство (принтер), аналогично тому, как команда PRINT выводит их на экран. При выводе на принтер элементы печатаются в том же формате, как их вывел бы на экран PRINT. LPRINT может включать также операторы TAB, INVERSE и OVER и управляющие коды, с тем же действием, как в PRINT.
Оператор AT также можно использовать, но номер строки игнорируется, а данные печатаются в указанной в AT колонке в той же строке.

ПРИМЕР

```
60 LPRINT "Number"; x' "Name"; n$, "Age"; a
```

MERGE — оператор / команда

ФОРМАТ

```
MERGE string
```

ОПИСАНИЕ

MERGE загружает программу с магнитной ленты в память компьютера, не уничтожая программу и переменные, находящиеся в памяти (т.е. объединяет их), замещая только те строки программы (переменные), которые имеют одинаковые номера (имена). Символьный аргумент MERGE – имя программы (файла), которая должна быть объединена с программой, находящейся в памяти. Это имя подчиняется тем же ограничениям, что и имена программ, используемые в LOAD.

ПРИМЕР

```
500 MERGE "prog2"
```

D MERGE * — оператор / команда

ФОРМАТ

```
MERGE * f-spec
```

ОПИСАНИЕ

Аналогично MERGE, но файл берется с диска. Использование пустой строки вместо имени не допускается.

D MOVE * — оператор / команда

ФОРМАТ

```
MOVE *
```

ОПИСАНИЕ

MOVE * переформирует диск в рабочем дисководе, освобождая занятое удаленными файлами пространство. Имеет смысл при наличии в каталоге (см. CAT *, LIST *, ERASE *) хотя бы одного удаленного файла.
Требует для своей работы не менее 4 Кбайт свободной памяти, доступной для BASIC-системы

(то есть значение **RAM–Top** должно быть не ниже (не меньше) 28800 при отсутствии BASIC-программы и/или переменных и соответственно выше на их длину при наличии).
Для обеспечения наибольшей скорости выполнения **MOVE ***, рекомендуется предварительно установить максимально возможное значение **RAM–Top**, выполнив **CLEAR 65535**.

MOVE & [N]

оператор / команда

ФОРМАТ
MOVE & s, integer, integer, integer, integer
MOVE & r, integer, integer, integer, integer

ОПИСАНИЕ
MOVE & s посылает, а **MOVE & r** принимает блок памяти в (из) компьютер, сетевой номер которого задан первым числовым параметром. Начальные адреса памяти посылающего и принимающего компьютеров задаются вторым и третьим числовыми параметрами соответственно, а длина пересылаемого блока – четвертым.

NEW

оператор / команда

ФОРМАТ
NEW

ОПИСАНИЕ
NEW очищает область памяти, доступную BASIC-системе (верхняя граница этой области хранится в системной переменной **RAM–Top** - см. **CLEAR**), удаляя существующие программу и переменные.
При этом графические символы, определяемые пользователем (они хранятся выше **RAM–Top**), не удаляются.

NEW * [D]

оператор / команда

ФОРМАТ
NEW * f–name$_{new}$, f–name$_{old}$ f–type

ОПИСАНИЕ
NEW * переименовывает файл на диске рабочего дисковода.

ПРИМЕР
Команда **NEW * "HomeWork","HomeVork"** переименует BASIC-программу **HomeVork** в **HomeWork**.

NEXT

оператор / команда

ФОРМАТ
NEXT letter

ОПИСАНИЕ
NEXT всегда используется в комбинации с **FOR** для организации **FOR–NEXT** цикла, ограничивая тело

цикла (см. **FOR**). За **NEXT** указывается имя управляющей переменной цикла.

ПРИМЕР
```
90 NEXT a
```

NOT

логический оператор / функция

ФОРМАТ
NOT cond
NOT number

Как логический оператор
NOT изменяет истинность условия так, что ложное условие становится истинным и наоборот.

Как функция
NOT возвращает 1 для ложного условия или нулевого значения числового выражения и 0 для истинного условия или не нулевого числового выражения.
Если условие, используемое в качестве аргумента **NOT**, содержит **AND** или **OR**, оно должно быть заключено в скобки.

ПРИМЕР
```
90 IF NOT x=y+z THEN PRINT "Wrong"
```
приводит к тому же эффекту, что и
```
90 LET correct=x=y+z : IF NOT correct
   THEN PRINT "Wrong"
```

OPEN#

оператор / команда

ФОРМАТ
OPEN# channel, string

ОПИСАНИЕ
OPEN # назначает каналу **channel** указанный вторым параметром поток ввода/вывода.

Каналы и потоки
В пк ХОББИТ операторы/команды ввода/вывода **INPUT**, **PRINT**, **LPRINT**, **LIST**, **LLIST**, **CAT *** и **LIST *** используют *потоки данных* как источник (или приемник) вводимой или выводимой информации. Поток данных характеризуется устройством ввода/вывода, с которым он жестко связан, направлением передачи данных и *каналом (каналами)*, назначенными этому потоку. Есть три потока: поток клавиатуры и нижней части экрана (**k**), поток верхней части экрана (**s**) и поток принтера (**p**) и 16 каналов с номерами от 0 до 15. В таблице приведено стандартное (т.е. производимое при запуске BASIC-системы) соответствие потоков и каналов с номерами 0÷3. Каналы 4÷15 первоначально не определены (не открыты).

поток	имя	тип	канал
клавиатура и нижняя часть экрана	k	ввод/вывод	0 или 1
верхняя часть экрана	s	вывод	2
принтер	p	вывод	3

Назначение канала, производимое **OPEN#** глобально, т.е. все операторы/команды, связанные с этим каналом, будут работать с новым потоком. Возможно локальное (в пределах одного оператора ввода/вывода) переназначение потока, путем указания в нем параметра **#channel**.

ПРИМЕР Команда **OPEN #2,"s"** переадресует на принтер весь последующий экранный вывод.

OPEN# & [N] оператор / команда

ФОРМАТ **OPEN# &**

ОПИСАНИЕ **OPEN# &** "включает" в локальную сеть компьютер, который был ранее из нее "выключен" командой **CLOSE# &**.

OPEN# * [D] оператор / команда

ФОРМАТ

Файл последовательного доступа

OPEN# * channel, f–spec, W

OPEN# * channel, f–spec, R

Файл прямого доступа

OPEN# * channel, f–spec, RND, integer

ОПИСАНИЕ **OPEN# *** создает новый (открывает существующий) файл *прямого* или *последовательного доступа* и назначает ему указанный канал (допускается использовать каналы 4÷15).

Файлы прямого и последовательного доступа

Это файлы, которые в отличие от файлов, создаваемых операторами **SAVE**, позволяют адресоваться к отдельным записям при помощи операторов **PRINT** (запись) и **INPUT** (чтение). Кроме того, такие файлы должны быть "закрыты" (см.**CLOSE# ***), если в них производилась запись. При создании под файлы прямого/последовательного доступа резервируется 4 Кбайта дискового пространства (16 секторов) и присваивается тип #. Если в процессе работы размер файла превысит 4 Кбайта, ему будет автоматически выделено дополнительное пространство.

Файл последовательного доступа может быть открыт или для чтения (при открытии был использован параметр **r**) или для записи (параметр **w**); файл прямого доступа открывается и для чтения и для записи.

Файлы этих типов представляют собой упорядоченные наборы записей-строк. Записи файлов последовательного доступа имеют переменную длину, прямого – постоянную, задаваемую при создании файла. К записям автоматически добавляется символ перевода строки (код 13) кроме случая, когда длина записи для файла прямого доступа совпадает с длиной записываемой переменной.

Ввод / вывод в файл последовательного доступа

Запись:
PRINT #channel ; string
PRINT #channel ; number
Чтение:
INPUT #channel ; string–var

Ввод / вывод в файл последовательного доступа

Запись:
PRINT #channel ; integer, string
PRINT #channel ; integer, number
Чтение:
INPUT #channel ; (integer), string–var

Числовой параметр в операциях с записями файла прямого доступа – номер записи (начиная с 0).

Копирование, переименование, удаление файлов прямого / последовательного доступа

Операции с целыми файлами (копирование и т.п.) производится аналогично обычным файлам.

ПРИМЕР файл последовательного доступа

```
 10 OPEN# *4, "HAPPY",w
 20 PRINT #4; "Happy New Year!"
 30 CLOSE# *4
    . . .
500 OPEN# *5, "HAPPY",r
510 INPUT #5, q$ : PRINT q$
520 CLOSE# *5
```

файл прямого доступа

```
 10 OPEN# *4, "SET_PHR", RND, 32
 20 FOR i=0 TO 10 STEP 2
 30 INPUT #4; (i), q$ : PRINT q$
 40 INPUT "Enter next phrase:"; q$
 50 PRINT #4; i+1, q$ : NEXT i
 60 CLOSE# *4
```

OR

логический оператор / функция

ФОРМАТ
cond OR cond
number OR number

Как логический оператор
OR связывает два условия в операторе, который проверяет истинность хотя бы одного из них.

Как функция
OR возвращает значение 1, если условие (выражение) справа от **OR** истинно (не равно 0) и значение выражения слева от **OR** в противном случае.

ПРИМЕРЫ
Строка
```
70 LET a$=INKEY$ : IF a$="N" OR a$="n" THEN STOP
```
останавливает выполнение программы, если была нажата клавиша **N** (независимо от использования **CAPS SHIFT** или **CAPS LOCK**).
Оператор **PRINT tax*(0.5 OR age 13)** выведет значение **tax**, если возраст (**age**) больше 13 лет, в противном случае будет выведена половина значения **tax**.

OUT

оператор / команда

ФОРМАТ
OUT integer, integer

ОПИСАНИЕ
OUT посылает байт, значение которого задается вторым аргументом (от 0 до 255), в порт ввода/вывода, адрес которого задается первым аргументом (от 0 до 65535).
Порты ввода/вывода используются для управления выходными устройствами. Так биты от 0 до 2 порта по адресу 254, устанавливают цвет рамки, бит 3 этого же порта управляет записью на магнитофон, а бит 4 – динамиком; порт с адресом 251 управляет принтером.

ПРИМЕР
Команда **OUT 254,3** установит фиолетовый цвет рамки.

OVER

оператор / команда

ФОРМАТ
OVER integer

ОПИСАНИЕ
OVER используется для задания режима вывода, при котором новый символ "накладывается" на уже существующий. **OVER 1** устанавливает этот режим, **OVER 0** – отменяет. "Наложение" производится по следующим правилам: точка (пиксель)

результирующего символа окрашиваются в **paper**-цвет, если она в старом и новом символе была одинаково окрашена (в **ink**-или **paper**-цвет), в противном случае точка будет окрашена в **ink**-цвет (т.н. *операция сложения по модулю 2*).
OVER можно также использовать в операторах вывода **PLOT, DRAW, CIRCLE, PRINT** и **INPUT**; в этом случае **OVER** действует локально – только на эти операторы – и должен указываться до выводимых данных и отделяться от них точкой с запятой.

ПРИМЕР
Команда
PRINT AT 11,15; "YES"; OVER 1; AT 11,15; "___" выведет в середине экрана подчеркнутое слово **YES**.

PAPER

оператор / команда

ФОРМАТ
PAPER integer[;]

ОПИСАНИЕ
PAPER используется для выбора **paper**-цвета (фона), используемого при выводе изображения на экран. Значения аргумента и правила применения **PAPER** такие же, как и для оператора **INK**.

ПРИМЕР
Команда **PAPER 1: CLS** вызовет изменение цвета экрана на синий.

PAUSE

оператор / команда

ФОРМАТ
PAUSE integer

ОПИСАНИЕ
PAUSE используется для приостановки выполнения программы на время (в секундах), определяемое как частное от деления аргумента на 50 (частота кадровой развертки).
Заметим, однако, что задержка может быть прервана нажатием любой клавиши, а **PAUSE 0** дает неограниченную задержку, которая длится, пока не нажата какая-либо клавиша.

ПРИМЕР
```
130 PAUSE 100
```
– приостановка выполнения программы на 2 секунды.

PEEK

Функция

ФОРМАТ
PEEK integer

ОПИСАНИЕ
PEEK возвращает значение байта, хранящегося в памяти по адресу, указнному аргументом **PEEK**.

ПРИМЕР
Число кадров телевизионного изображения, ко-

торые были выведены после включения компьютера в сеть, хранится в байтах памяти с адресами с 23672 по 23674. Так как кадровая частота равна 50 Гц, можно подсчитать время (в секундах) с того времени, когда компьютер был включен (без времени, затраченного на создание звука и управление периферийными устройствами, такими как принтер или кассетный магнитофон):

```
10 PRINT (PEEK 23672 + 256*PEEK 23673
   + 65536*PEEK 23674)/50
```

PEEK * [D] оператор / команда

ФОРМАТ **PEEK * f–spec f–type integer, integer**

ОПИСАНИЕ **PEEK *** читает из файла **f–spec** типа **f–type** сектор (256 байт), номер которого (нумерация начинается с 1) задан вторым числовым параметром, и записывает его в оперативную память компьютера, начиная с адреса, заданного первым числовым параметром.

PI функция

ФОРМАТ **PI**

ОПИСАНИЕ **PI** возвращает значение числа π – отношение длины окружности к ее диаметру.

ПРИМЕР **PLOT 0,0 : DRAW 255,0,–PI**

выведет на экран большую полуокружность.

PLOT оператор / команда

ФОРМАТ **PLOT [statement ;] integer, integer**

ОПИСАНИЕ **PLOT** используется в режиме графики с высоким разрешением для вывода точки в позиции, координаты **x,y** которой указываются аргументами **PLOT**; кроме того, **PLOT** определяет начальную позицию для следующего оператора **DRAW**.

ПРИМЕР Оператор **50 PLOT INVERSE 1; 128,87** выведет точку в указанной позиции в **paper**-цвете, т.ч. она не будет видна, а изменится только текущая позиция для последующих операторов **DRAW**.

POINT Функция

ФОРМАТ **POINT (integer, integer)**

POINT возвращает 1, если точка (пиксель) в позиции, указанной аргументами, имеет цвет **ink** и 0 - если **paper**-цвет. *ОПИСАНИЕ*

```
240 IF POINT(x,y)=1 THEN GOSUB 600
```

ПРИМЕР

POKE оператор / команда

ФОРМАТ **POKE integer, integer**

ОПИСАНИЕ **POKE** используется для изменения содержимого байта памяти по адресу, указанному первым аргументом, на значение, указанное вторым аргументом **POKE**.

ПРИМЕР Команда **POKE 23609,255** изменит значение системной переменной, управляющей звуком, издаваемым при нажатии клавиши – значение 255 дает длинный звуковой сигнал вместо обычного щелчка, другие значения дадут более короткие звуковые сигналы.

POKE * [D] оператор / команда

ФОРМАТ **POKE * f–spec f–type integer, integer**

ОПИСАНИЕ **POKE *** записывает 256 байт, начиная с адреса ОЗУ, заданного первым числовым параметром, в сектор, номер которого задан вторым числовым параметром, файла **f–spec** типа **f–type**.

PRINT оператор / команда

ФОРМАТ **PRINT [TAB integer ;][AT integer, integer ;][CHR$ integer ;] [statement ;][number][string][;][,][']**

ОПИСАНИЕ **PRINT** выводит данные на экран. Данные могут быть числовыми или символьными. Произвольное число данных этих типов может быть выведено одним оператором **PRINT**. Расположение отдельных групп данных на экране определяется **PRINT**-*разделителями;* к ним относятся: *запятая* (,), *точка с запятой* (;), *апостроф* ('), ключевые слова **AT** и **TAB**.

Передвигает позицию вывода в 16-ю колонку текущей строки или в начало следующей, если предыдущий вывод закончился во второй половине текущей строки. *Запятая*

Переносит позицию вывода в начало следующей строки. *Апостроф*

Точка с запятой Вызывает вывод следующих данных непосредственно за позицией, в которой закончился предыдущий вывод.

TAB Переносит вывод в указанную позицию текущей строки.

AT Переносит вывод в указанные строку и позицию в строке.

Кроме этого, в **PRINT** можно использовать управляющие коды, формируемые функцией **CHR$** (перечень кодов дан в описании **CHR$**).

ПРИМЕР Команда

```
PRINT AT 0,0; INK 1; PAPER 2; "*"; TAB 31; CHR$ 42;
      FLASH 1; AT 21,0; "*"; TAB 31; "*"
```

выведет в углах экрана голубые звездочки на красном фоне; в нижних углах они будут мерцающими.

RANDOMIZE

оператор / команда

ФОРМАТ **RANDOMIZE [integer]**

ОПИСАНИЕ **RANDOMIZE** используется в сочетании с **RND** для получения последовательностей случайных чисел. Для многократного получения одной и той же последовательности используется не равный 0 аргумент **RANDOMIZE** – своя последовательность для каждого значения. Использование нулевого аргумента приведет к случайному выбору одной из этих 65535 последовательностей.

ПРИМЕР Команда **RANDOMIZE : PRINT INT (RND*7)** выведет случайное целое число от 0 до 6.

READ

оператор / команда

ФОРМАТ
READ num–var [, num–var][, string–var]
READ string–var [, num–var][, string–var]

ОПИСАНИЕ **READ** используется для присвоения значений из **DATA**-списка переменным, указанным в качестве аргументов **READ**.

Данные **DATA**-списка образуются одним или несколькими операторами **DATA**. При выполнении **READ** последовательно получает данные из этого списка (начиная с первого при запуске программы). Порядок получения может быть изменен при помощи оператора **RESTORE**.

Программа *ПРИМЕР*

```
10 DATA 30,"Nov"
20 FOR n=1 TO 2
30 READ x,a$
40 PRINT a$,x;" days"
50 NEXT n
60 DATA 31,"Dec"
```

Выведет на экран

```
Nov     30 days
Dec     31 days
```

REM

оператор

ФОРМАТ **REM любой символ**

ОПИСАНИЕ **REM** используется для записи пояснений (комментариев) в тексте программы. BASIC игнорирует остаток строки за оператором **REM**, поэтому он может быть или единственным или последним оператором в строке.

ПРИМЕР

```
80 INPUT n$ : REM n$ is name
```

RESTORE

оператор / команда

ФОРМАТ **RESTORE [integer]**

ОПИСАНИЕ **RESTORE** изменяет порядок присваивания данных из **DATA**-списков переменным, указанным в операторе **READ**.

После выполнения **RESTORE** данные будут браться из оператора **DATA** в строке с ближайшим не меньшим указанного в **RESTORE** номером; если же номер не указан, он считается равным 0.

ПРИМЕР

```
160 RESTORE 800
```

RETURN

оператор / команда

ФОРМАТ **RETURN**

ОПИСАНИЕ **RETURN** используется для завершения подпрограммы и передачи управления на следующий за вызовом подпрограммы оператор.

ПРИМЕР Оператор **180 RETURN** вызовет переход к следующему за последним выполненным **GOSUB** оператору.

RND

функция

ФОРМАТ **RND**

ОПИСАНИЕ **RND** возвращает следующее число (от 0 до 1) из текущей последовательности случайных чисел.

Для выбора новой последовательности (или переустановки текущей на начало) используется оператор **RANDOMIZE**.

ПРИМЕР Команда

```
PRINT INK RND*8 ; PAPER RND*8 ; AT RND*32, RND*22 ; "*"
```

выведет на экран в случайно выбранном месте звездочку; при этом **ink**– и **paper**–цвета также выбираются случайно.

RUN

оператор / команда

ФОРМАТ **RUN [integer]**

ОПИСАНИЕ **RUN** "запускает" программу – уничтожает существующие переменные, очищает экран, устанавливает **PLOT**-позицию в левый нижний угол экрана (т.е. выполняет оператор **CLEAR**) и передает управление первому оператору в строке с ближайшим не меньшим указанного в **RUN** номером (первая строка программы, если номер не указан).

Если необходимо только передать управление (без выполнения начальных действий оператора **RUN**), нужно использовать **GOTO**.

RUN * [D]

оператор / команда

ФОРМАТ **RUN * f–spec [CODE][integer]**

ОПИСАНИЕ **RUN *** загружает в память и запускает BASIC-программу или программу в машинных кодах (указан параметр **CODE**), сохраненную в файле **f–spec**. Если задан числовой параметр, то BASIC-программа запускается со строки с ближайшим не меньшим указанного параметром номером, а программа в машинных кодах загружается в ОЗУ начиная с адреса, равного значению параметра и с него же запускается. Если параметр опущен – BASIC-программа запускается со строки, номер которой был указан в параметре **LINE** при записи программы на диск или с первой строки программы (строка автозапуска при сохранении не указывалась); программа в кодах загружается в то же место памяти, откуда была сохранена, и с того же адреса запускается.

ВНИМАНИЕ: пользоваться параметром для

программ в кодах допускается только для *позиционно-независимых* программ. Для программ, осуществляющих *самонастройку* на положение в ОЗУ, следует использовать командную последовательность вида:

```
LOAD * f–spec CODE addr : RANDOMIZE USR addr
```

SAVE

оператор / команда

ФОРМАТ **SAVE string [LINE integer]**

ОПИСАНИЕ **SAVE** посылает программу и все определенные в данный момент переменные на кассетный магнитофон, чтобы записать (сохранить) ее на ленте в файле, имя которого указывается первым (обязательным) параметром **SAVE**. Требования к имени файла – см. **LOAD**.

Перед тем, как начать вывод на магнитофон, компьютер выводит на нижнюю строку экрана сообщение: **Start tape, then press any key** (*Включите магнитофон на запись и нажмите любую клавишу*).

Если сохраненная программа должна запускаться автоматически при загрузке, используется необязательное ключевое слово **LINE** с параметром - номером строки, с которой программа будет запущена после загрузки.

ПРИМЕР

```
SAVE "filename" LINE 1
```

SAVE & [N]

оператор / команда

ФОРМАТ **SAVE & integer**

ОПИСАНИЕ **SAVE &** загружает в компьютер, сетевой номер которого указан параметром, BASIC-программу и переменные. В отличие от аналогичной функции теневого меню, работает как на компьютере учителя, так и ученика.

SAVE * [D]

оператор / команда

ФОРМАТ

```
SAVE * f–spec [CODE [integer ]][DATA letter[$]() ][SCREEN$]
SAVE * f–spec LINE integer
```

ОПИСАНИЕ Аналогично командам из серии **SAVE**, но сохранение производится не на ленте, а в файле на диске, указанного в **f–spec** дисковода.

SAVE CODE — оператор / команда

ФОРМАТ SAVE string CODE integer, integer

ОПИСАНИЕ SAVE CODE посылает часть информации из памяти на магнитофон для записи на ленту. Начальный адрес и длина (в байтах) сохраняемого участка памяти указывется после ключевого слова CODE, а имя файла на ленте – после SAVE.

ПРИМЕР Команда SAVE "picture" CODE 16384,6912 запишет на ленте текущее изображение на экране.

SAVE DATA — оператор / команда

ФОРМАТ SAVE string DATA letter[$]()

ОПИСАНИЕ SAVE DATA сохраняет массив (числовой или символьный) на ленте. Имя массива указывается после ключевого слова DATA, а имя файла - после SAVE.

ПРИМЕР
```
450 SAVE "numbers" DATA n()
750 SAVE "names" DATA n$()
```

SAVE SCREEN$ — оператор / команда

ФОРМАТ SAVE string SCREEN$

ОПИСАНИЕ SAVE SCREEN$ сохраняет изображение экрана на ленте. Отметьте, что эта команда аналогична команде SAVE "имя_файла" CODE 16384,6912.

ПРИМЕР
```
SAVE "picture" SCREEN$
```

SCREEN$ — функция

ФОРМАТ SCREEN$ (integer, integer)

ОПИСАНИЕ SCREEN$ возвращает односимвольную строку, содержащую символ, находящийся на экране в указанной позиции или пустую строку (" "), если в этой позиции нет символа.

ПРИМЕР
```
160 SCREEN$(l,c)="X" THEN PRINT "Crash"
```

SGN — функция

ФОРМАТ SGN number

ОПИСАНИЕ SGN возвращает 1, если ее аргумент положителен, -1 - если отрицателен и 0, если нулевой.

ПРИМЕР
```
50 LET x=SGN y
```


SIN — функция

ФОРМАТ SIN number

ОПИСАНИЕ SIN возвращает значение синуса угла, заданного в радианах.

ПРИМЕР Команда PRINT SIN (30*PI/180) выведет 0.5 - значение синуса угла, равного 30 градусам.

SQR — функция

ФОРМАТ SQR number

ОПИСАНИЕ SQR возвращает квадратный корень своего неотрицательного аргумента.

ПРИМЕР
```
70 LET x=SQR y
```

STEP

Смотри FOR

STOP — оператор / команда

ФОРМАТ STOP

ОПИСАНИЕ STOP останавливает выполнение программы. STOP можно использовать, например, чтобы отделить основную часть программы от текста ее подпрограмм или при отладке.
Ввод CONTINUE после остановки программы оператором STOP вызывает возобновление выполнения программы со следующего за STOP оператора.

ПРИМЕР
```
...
40 GOSUB 200
...
170 STOP
200 REM Подпрограммы ...
```

STR$ — функция

ФОРМАТ STR$ number

ОПИСАНИЕ STR$ возвращает символьное представление своего числового аргумента.

ПРИМЕР После выполнения фрагмента программы
```
70 LET x = 70
90 LET a$ = STR$ x
```
символьная переменная a$ будет иметь значение "70".

TAB

Смотри PRINT

TAN

Функция

ФОРМАТ **TAN number**

ОПИСАНИЕ **TAN** возвращает значение тангенса угла, заданного в радианах.

ПРИМЕР Команда **PRINT TAN (45*PI/180)** выведет на экран 1 - тангенс угла, равного 45 градусам.

THEN

Смотри **IF**

TO

Функция

ФОРМАТ **string ([number] TO [number])**

ОПИСАНИЕ **TO** имеет два различных применения:
для организации **FOR–NEXT** *цикла* (подробнее смотри в **FOR**) и *для получения подстроки из символьной переменной или константы* (*операция сечения*).
Подстрока – это последовательность символов исходной переменной (константы), порядковый номер первого символа которой указывается первым аргументом операции сечения, а порядковый номер последнего – вторым аргуметом. Если один или оба аргумента операции сечения отсутствуют, то значение первого аргумента считается равным 1, а второго – длине исходной строки.

ПРИМЕР Команда **PRINT "DogCatFox" (4 TO 6)** выведет на экран **Cat**, а команда **PRINT "DogCatFox" (7 TO)** – **Fox**.

USR

Функция

ФОРМАТ **USR integer**
USR string

ОПИСАНИЕ **USR** используется для вызова подпрограммы в машинных кодах, расположенной в памяти по адресу, указываемому числовым аргументом **USR**.
Если же аргумент **USR** – символьный, функция возвращает адрес первой ячейки памяти, где хранится изображение графического символа, определяемого пользователем.

USR *и машинные коды.*

Любой оператор, содержащий **USR** с числовым аргументом, вызывает подпрограмму по адресу,

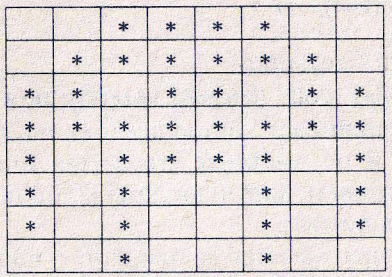
равному значению аргумента, по завершению которой **USR** возвращает содержиме регистровой пары **BC**.
RANDOMIZE USR или **RESTORE USR**, например, только запускают подпрограмму, тогда как **PRINT USR** дополнительно выводит значение регистровой пары **BC**.

USR *и графика, определяемая пользователем.*

Для создания графики, определяемой пользователем, **USR** используется с символьным аргументом, первый символ которого должен быть буквой, от **A** до **U** (при этом строчные и прописные буквы не различаются).
USR возвращает начальный адрес одной из 21 областей памяти, зарезервированных для графики, определяемой пользователем. Каждая область содержит восемь байт, описывающих изображение одного графического символа.

ПРИМЕР

Программа создает графический символ - изображение "захватчика" и закрепляет его за **S** в графическом режиме.
Символ строится в решетке 8×8:



Закрашенной клетке соответствует точка символа, окрашенная в **ink**-цвет.

```
10 POKE USR "s", BIN 00111100
11 POKE USR "s"+1, BIN 01111110
12 POKE USR "s"+2, BIN 11011011
13 POKE USR "s"+3, BIN 11111111
14 POKE USR "s"+4, BIN 10111101
15 POKE USR "s"+5, BIN 10100101
16 POKE USR "s"+6, BIN 10100101
17 POKE USR "s"+7, BIN 00100100
```

VAL — **функция**

ФОРМАТ VAL string

ОПИСАНИЕ VAL возвращает числовое значение своего аргумента, представляющего собой синтаксически допустимое арифметическое выражение, записанное в символьной форме.

ПРИМЕР Команда PRINT VAL "2+1.5*(4+2)" выведет на экран 11 - числовое значение приведенного символьного выражения.

VAL$ — **функция**

ФОРМАТ VAL$ string

ОПИСАНИЕ VAL$ возвращает символьное значение своего аргумента, представляющего собой синтаксически допустимое символьное выражение.

ПРИМЕР Программа

```
10 LET a$ = "2 +"
20 LET b$ = " 1.5 * (2+4)"
30 LET c$ = VAL$ "a$ + b$"
40 PRINT c$; " = "; VAL c$
```

выведет на экран

```
2 + 1.5 * (2+4) = 11
```

VERIFY — **оператор / команда**

ФОРМАТ VERIFY string [CODE [integer][,integer]][DATA letter[$]()]

ОПИСАНИЕ VERIFY сравнивает копию данных на ленте (программа и переменные, двоичная информация или массивы) полученную командой SAVE с их копией в памяти.

Параметры команд совпадают с параметрами LOAD, LOAD CODE и LOAD DATA.

VERIFY * [D] — **оператор / команда**

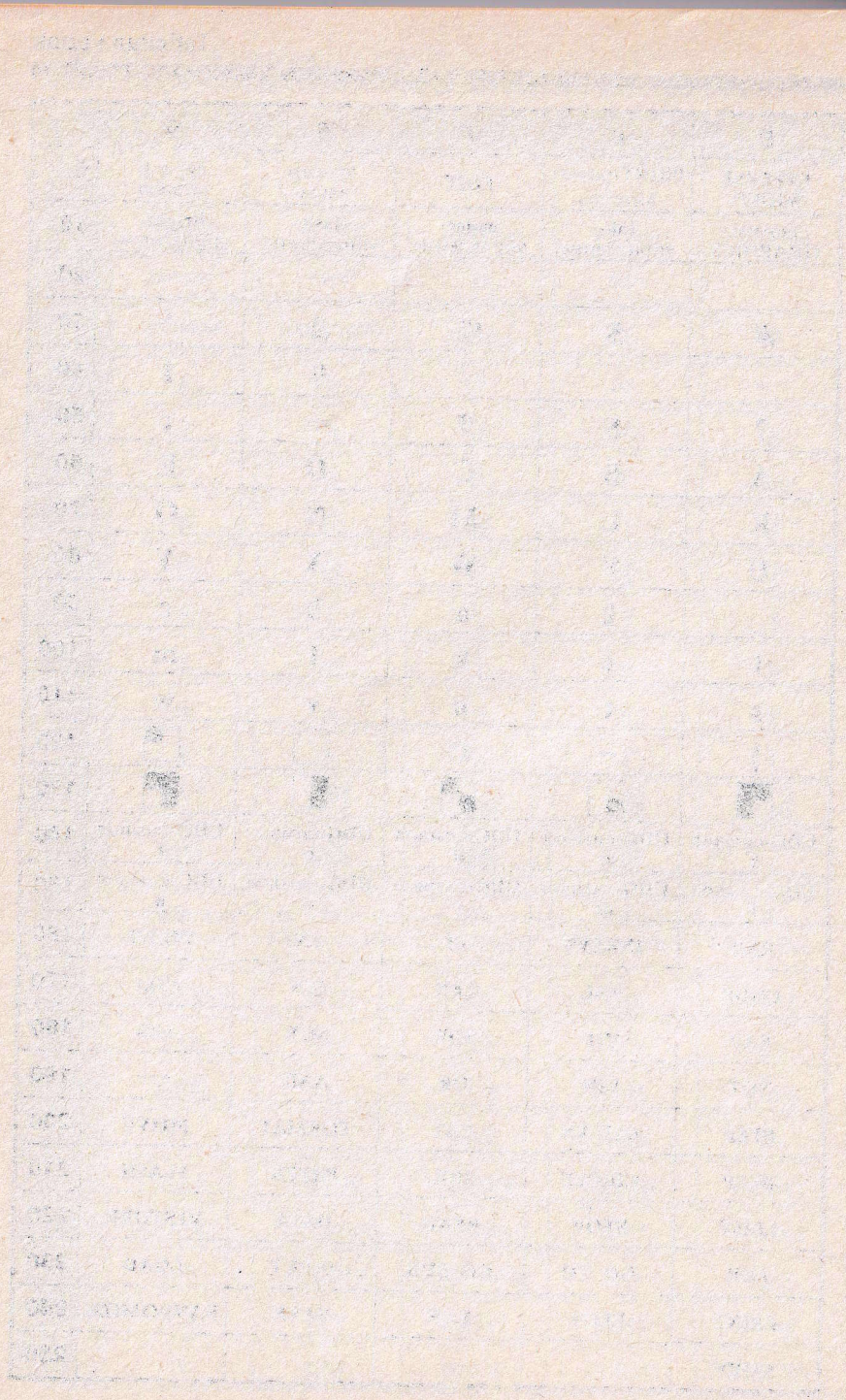
ФОРМАТ VERIFY *f-spec [CODE [integer][,integer][DATA letter[$]()]

ОПИСАНИЕ VERIFY * аналогично VERIFY сравнивает копию данных в файле f-spec, полученную командой SAVE * с их копией в памяти.

Таблица кодов ПК ХОББИТ

	0	1	2	3	4
0					TRUE VIDEO
10	курсор вниз	курсор вверх	DELETE	ENTER	маркер числа
20	inverse-управление	over-управление	at-управление	tab-управление	
30	русский регистр	латинский регистр	пробел	!	"
40	(	)	*	+	,
50	2	3	4	5	6
60	<	=	>	?	@
70	F	G	H	I	J
80	P	Q	R	S	T
90	Z	[	\	]	↑
100	d	e	f	g	h
110	n	o	p	q	r
120	x	y	z	{	|
130	▗	▄	▖	▌	▚
140	▂	▟	▙	▉	UDG-символ a
150	UDG-символ g	UDG-символ h	UDG-символ i	UDG-символ j	UDG-символ k
160	UDG-символ q	UDG-символ r	UDG-символ s	UDG-символ t	UDG-символ u
170	SCREEN$	ATTR	AT	TAB	VAL$
180	TAN	ASN	ACS	ATN	LN
190	PEEK	IN	USR	STR$	CHR$
200	>=	<>	LINE	THEN	TO
210	ERASE	OPEN#	CLOSE#	MERGE	VERIFY
220	BRIGHT	INVERSE	OVER	OUT	LPRINT
230	NEW	BORDER	CONTINUE	DIM	REM
240	LIST	LET	PAUSE	NEXT	POKE
250	IF	CLS	DRAW	CLEAR	RETURN

5	6	7	8	9	
INVERSE VIDEO	PRINT-разделитель ,	EDIT	курсор влево	курсор вправо	0
режим GRAPHICS	ink-управление	paper-управление	flash-управление	bright-управление	10
					20
#	$	%	&	'	30
-	.	/	0	1	40
7	8	9	:	;	50
A	B	C	D	E	60
K	L	M	N	O	70
U	V	W	X	Y	80
_	£	a	b	c	90
i	j	k	l	m	100
s	t	u	v	w	110
}	~	©	□	▙	120
▛	▘	▚	▌	▛	130
UDG-символ b	UDG-символ c	UDG-символ d	UDG-символ e	UDG-символ f	140
UDG-символ l	UDG-символ m	UDG-символ n	UDG-символ o	UDG-символ p	150
RND	INKEY$	PI	FN	POINT	160
CODE	VAL	LEN	SIN	COS	170
EXP	INT	SQR	SGN	ABS	180
NOT	BIN	OR	AND	<=	190
STEP	DEF FN	CAT	FORMAT	MOVE	200
BEEP	CIRCLE	INK	PAPER	FLASH	210
LLIST	STOP	READ	DATA	RESTORE	220
FOR	GO TO	GO SUB	INPUT	LOAD	230
PRINT	PLOT	RUN	SAVE	RANDOMIZE	240
COPY					250



Сообщения об ошибках

Если при выполнении программы компьютер встречает ошибочный оператор, или оператор, вызывающий остановку, обработка прекращается и на нижнюю строку выводится сообщение о причине остановки. Оно имеет вид:

n *<Текст сообщения о причине останова >* **x : y**

где: **n** – односимвольный код сообщения (от 0 до 9, от A до R),
x – номер строки в программе, 0 для командной строки,
y – номер оператора в строке.

Ниже приводится полный перечень сообщений.

код	текст и причина появления	ситуация
0	**OK** успешное завершение или переход на строку с номером большим, чем есть в программе	
1	**NEXT without FOR** переменная, указанная в операторе, не является управляющей переменной (т.е. переменная определена не в операторе **FOR**)	**NEXT**
2	**Variable not found** использование неопределенной переменной. Сообщение выводится для простой переменной, если она используется без предварительного определения в операторах **LET, READ, INPUT, FOR**. Для индексной переменной сообщение выводится, если перед использованием или загрузкой с ленты она не была предварительно определена в операторе **DIM**	
3	**Subscript wrong** индекс или превышает размерность массива или находится вне пределов строки для операции сечения	в индексной переменной или подстроке
4	**Out of memory** не хватает памяти для размещения программы и/или переменных	**LET, FOR, DIM, INPUT, GOSUB, LOAD, MERGE**
5	**Out of screen** выход за пределы экрана	**PRINT AT, INPUT**

6	**Number too big** в результате вычислений получается число, превыщающее по абсолютной величине $1.7*10^{38}$	арифметические операции
7	**RETURN without GOSUB** выполнению оператора **RETURN** не предшествует выполнение оператора **GOSUB**	**RETURN**
8	**End of file** конец файла	операции с внешней памятью
9	**STOP statenent** выполнение прекращено оператором **STOP**	**STOP**
A	**Invalid argument** недопустимое значение аргумента функции	**ASN, ACS, LN, SQR, USR** (со строковым аргументом)
B	**Integer out of range** значение целочисленного аргумента превышает 65535	**CHR$, DIM, GOTO, GOSUB, [L]LIST, PAUSE, PEEK, PLOT, POKE, RANDOMIZE, RUN, USR** (с числовым аргументом)
C	**Nonsense in BASIC** строковая переменная не распознается, как допустимое выражение BASIC'a	**VAL, VAL$**
D	**BREAK–CONT repeats** выполнена команда **BREAK** во время периферийной операции (ввод/вывод на мл, на принтер), или в ответ на запрос разрешения скроллирования (**Scroll?**) указаны **STOP, N** или пробел	**LOAD, SAVE, COPY, VERIFY, MERGE, [L]LIST, [L]PRINT**
E	**Out of DATA** попытка выполнения оператора **READ**, когда список данных в операторе **DATA** исчерпан	**READ**
F	**Invalid file name** ошибочное имя файла (пустая строка или больше 10 символов)	**SAVE**
G	**No room for line** недостаточно места в памяти для записи очередной строки программы	ввод новой строки в программу

H	**STOP in INPUT** ввод оператора **STOP** в качестве очередного данного	**INPUT**
I	**FOR without NEXT** отсутствует оператор **NEXT** для оператора **FOR**	**FOR**
J	**Invalid I/O device** указано неверное устройство ввода/вывода	операции с внешней памятью
K	**Invalid colour** указано неверное значение для одного из операторов, влияющих на аттрибуты изображения	**BORDER, PAPER, INK, OVER, FLASH, BRIGHT, INVERSE,** после вывода управляющей последовательности кодов
L	**BREAK into program** выполнена команда **BREAK** во время перехода к обработке следующего оператора программы. **CONTINUE** вызовет продолжение выполнения со следующего оператора	
M	**RAMTOP no good** число, указанное в качестве значения системной переменной *Ram-Top*, слишком мало или велико	**CLEAR, RUN**
N	**Statement lost** переход к оператору, которого уже нет	**RETURN, NEXT, CONTINUE**
O	**Invalid stream** ошибочный поток данных	операции с внешней памятью
P	**FN without DEF** определяемая пользователем функция не определена в операторе **DEF FN**	**FN**
Q	**Parameter error** ошибочное число или тип аргументов в обращении к определяемой пользователем функции	**FN**
R	**Tape LOADing error** ошибка при чтении файла с магнитной ленты	**LOAD, MERGE, VERIFY**

ДИСКОВЫЕ ОШИБКИ
File not found файл с указанными именем и типом отсутствует на диске
File exists файл с указанными именем и типом уже существует на диске
Disk full на диске отсутствует свободное пространство
Directory full в каталоге диска отсутствует свободное пространство
Too many records номер записи файла прямого доступа слишком велик
No disk в дисководе нет диска, или не закрыта крышка или диск неформатированный
Disk error ошибка записи / чтения на диске
Disk error **Trk** xx **sec** yy **Retry, Abort, Ignore** сектор yy на дорожке xx – плохой; ввод **R** повторит операцию с этим же сектором, **A** – прервет выполнение, **I** – пропустит указанный сектор и перейдет к следующему
Read Only **Trk 0 sec 1** **Retry, Abort, Ignore** попытка записи на диск, защищенный от записи
Verify error ошибка при сравнении файла на диске с его оригиналом в памяти
Syntax error синтаксическая ошибка
Stream opened указанный канал уже связан с потоком ввода / вывода
СЕТЕВЫЕ ОШИБКИ
No broadcast указан 0 в качестве сетевого номера – попытка осуществить сетевую операцию для всех компьютеров в локальной сети; разрешено только для **MASTER**-компьютера

Invalid station указан неверный сетевой номер
No reply компьютер с указанным сетевым номером отсутствует
No room for block отсутствует свободная память для пересылки блока
Place no good сетевая операция с указанной областью памяти недопустима
Block length zero пересылка блока нулевой длины

Существует возможность программной обработки ошибок дисковых операций. Для этого используется специальная форма записи операторов работы с диском:

```
LET Error_Code=USR 2: REM: statement
```

После выполнения этой последовательности операторов, переменная **Error_Code** будет иметь одно из следующих значений:

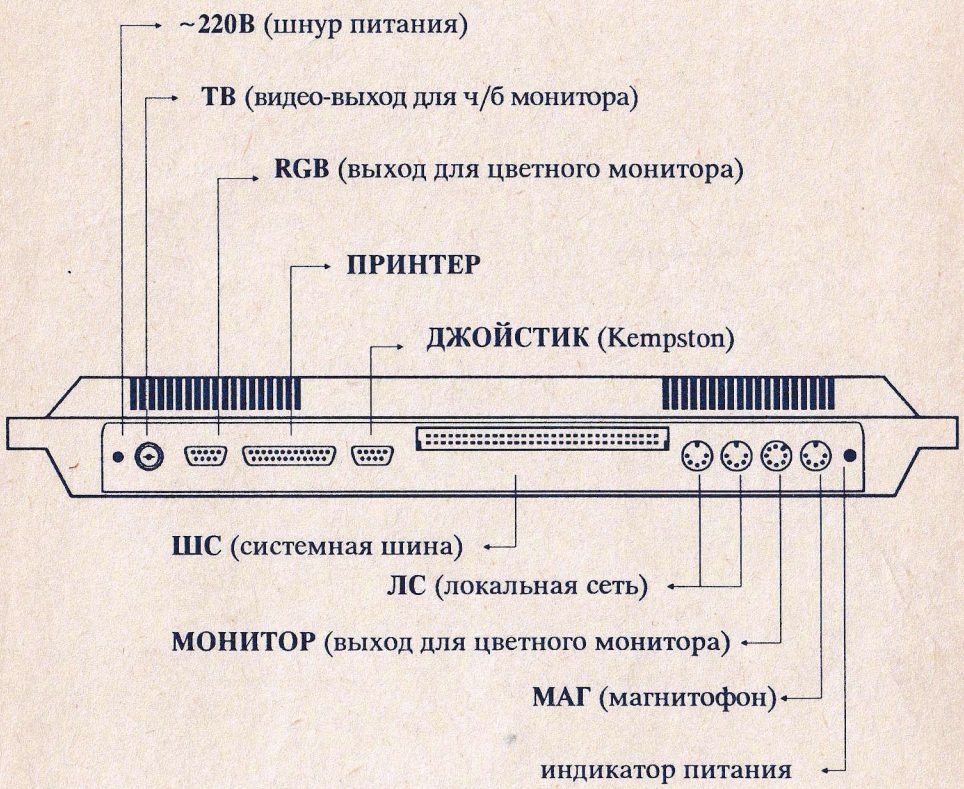
код	значение	код	значение
0	нет ошибок	6	нет диска
1	нет такого файла	7	ошибка чтения/записи
2	файл уже существует	8	синтаксическая ошибка
3	нет свободного пространства	10	канал уже открыт
4	превышен размер каталога	12	канал не открыт
5	номер записи слишком велик	13	ошибка при сравнении

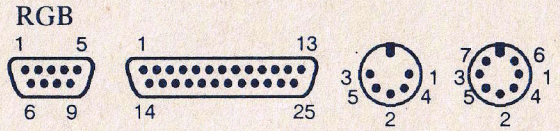
При этом оператор **statement** – один из дисковых операторов BASIC'а, в записи которого опущен символ *; кроме этого, он должен быть последним оператором программной строки.

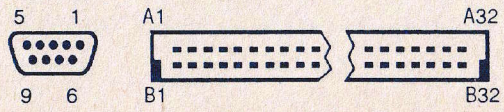
Пример:

```
10 DIM m$( 10,64 )
20 LET Err=USR 2 : REM : LOAD "b:Memo" DATA m$()
30 IF Err=1 THEN CLS: PRINT AT 10,5; "Файла Memo нет на диске" : STOP
   ...
```

ВИД ЗАДНЕЙ ПАНЕЛИ КОМПЬЮТЕРА







ТАБЛИЦЫ РАСПАЙКИ РАЗЪЕМОВ

Разъем / Контакт	монитор	RGB	джойстик	МАГ	ЛС
	сигнал				
1	B			OUTPUT	NETCNT
2	GND	GND	RIGHT	GND	GND
3	SOUND	R	LEFT	INPUT	NETCLK
4	R	G	DOWN		MASTER
5	SYNC	B	UP		NETDAT
6	G	Y			
7	Y	SOUND	+5V		
8		HSYNC			
9		VSYNC	FIRE		

Разъем / Контакт	принтер	ШС ряд А	ШС ряд В
	сигнал		
1	$\overline{STR}$	+5V	CRASH
2	PD 0	A11	A10
3	PD 1	A12	A9
4	PD 2	A13	A8
5	PD 3	A14	A7
6	PD 4	A15	A6
7	PD 5	CLK	A5
8	PD 6	D4	A4
9	PD 7	D3	A3
10		D5	A2
11	BUSY	D6	A1
12	GND	RAMCS	A0
13		D2	$\overline{ROMCS}$
14		D7	$\overline{RFSH}$
15		D0	$\overline{M1}$
16		D1	$\overline{RESET}$
17		$\overline{INT}$	$\overline{BUSRQ}$
18		$\overline{NMI}$	$\overline{WAIT}$
19		$\overline{MOTOR}$	$\overline{HALT}$
20		$\overline{MREQ}$	$\overline{WR}$
21		$\overline{IORQ}$	$\overline{RD}$
22		$\overline{WRGATE}$	CF0
23		$\overline{STEP}$	CF1
24		DIR	CF2
25		$\overline{DR0}$	$\overline{BUSACK}$
26		$\overline{DR1}$	$\overline{XRES}$
27		$\overline{DR2}$	$\overline{DX0}$
28		$\overline{DR3}$	$\overline{DX1}$
29		$\overline{WRDATA}$	+12V
30		$\overline{RDDATA}$	$\overline{DX2}$
31		FDCV	SIDE
32		GND	GND