

**ИНСТРУМЕНТАЛЬНО ПЕДАГОГИЧЕСКИЕ
ПРОГРАММНЫЕ СРЕДСТВА
КУВТ MSX-2**

Руководство пользователя

**СП «Диалог»
«Сотрудничество»**

1988

ИНСТРУМЕНТАЛЬНО ПЕДАГОГИЧЕСКИЕ

ПРОГРАММНЫЕ СРЕДСТВА

КУВТ MSX-2

Руководство пользователя

СП "Диалог
"Сотрудничество"

1988

СОДЕРЖАНИЕ

I.	СЕТЕВОЙ BASIC - МОНИТОР.	5
	1. Назначение и основные характеристики.	7
	2. Комплект поставки.	9
	3. Команды монитора учителя.	10
	4. Команды ученика.	25
	5. Глава системного программиста.	29
	Приложения.	32
II.	СЕТЕВАЯ PASCAL - СИСТЕМА.	37
	1. Назначение.	39
	2. Состав и дистрибуция.	40
	3. Загрузка системы.	41
	4. Команды учителя.	41
	5. Команды ученика.	52
	6. Исполняющая система.	54
	7. Пакет сетевых функций ученика.	55
	8. Конвертор объектных модулей.	59
	9. Глава системного программиста.	60
III.	ИНСТРУМЕНТАЛЬНЫЕ ПРОГРАММНЫЕ СРЕДСТВА ДЛЯ КОМПИЛЯТОРА NISOFT PASCAL.	63
	Введение.	65
	1. Основные графические процедуры.	67
	2. Дополнительные графические процедуры.	76
	3. Закрашивание замкнутых областей.	84
	4. Динамическая графика.	87
	5. Форматный вывод на графический экран.	93
	6. Процедуры для работы с видеопроцессором.	98
	7. Клавиатура.	114
	8. Графический пакет LOGO - PAS.	124

9. Дополнительные сведения для системного программиста.	134
Приложения	140
А. Коды логических операций и координатные сетки.	140
В. Матрица клавиатуры.	142
С. Системные переменные и рабочие области.	143
Д. Таблица расширенных переходов.	156
Е. Пример программы рисования линиями различного типа.	163
Ф. Пример программы закрашки замкнутой области шаблоном.	165
Г. Демонстрационная программа LOGO - TUTOR.	167
 IV. ИНСТРУМЕНТАЛЬНЫЕ ПРОГРАММНЫЕ СРЕДСТВА ДЛЯ КОМПИЛЯТОРА ASCII C И АССЕМБЛЕРА M80.	 169
Введение.	171
1. Математический пакет.	173
2. Графические функции.	184
3. Динамическая графика.	203
4. Оконный ввод/вывод.	207
5. Клавиатура.	215
6. Музыкальный пакет.	226
7. Системные процедуры.	231
8. Дополнительные сведения для системного программиста.	251
Приложения	259
А. Коды логических операций и координатные сетки.	259
В. Матрица клавиатуры.	261
С. Системные переменные и рабочие области.	262
Д. Таблица расширенных переходов.	275
Е. Справочные сведения для программирования на ассемблере.	278
Ф. Пример программы рисования линиями различного типа.	299
Г. Пример программы закрашки замкнутой области шаблоном.	301

I СЕТЕВОЙ БЕЙСИК-МОНИТОР

НАЗНАЧЕНИЕ И ОСНОВНЫЕ ХАРАКТЕРИСТИКИ.

Сетевой монитор предназначен для программной поддержки различных типов занятий, проводимых на базе КУВТ "Ямаха" (MSX-2). Он предоставляет в распоряжение преподавателя такие возможности, как:

- использование заранее подготовленных командных файлов для управления уроком;
- загрузка с диска BASIC-программ любому отдельному ученику или любому подмножеству учеников (без действий со стороны учеников);
- сброс на диск BASIC-программ любого ученика (без действий со стороны ученика);
- пересылка ученику (ученикам) двоичных данных (OBJ-файл);
- коллективные сбор и раздача BASIC-программ ученикам (ученики сами задают имена файлов);
- прием из сети и запись на диск протоколов работы учеников (для обучающих и тестирующих программ);
- работа с экранными файлами: передача по сети изображений и текстов на экраны учеников и работа с ними;
- просмотр содержимого экранов учеников;
- вывод;
- электронная почта;
- работа с оглавлением диска.

При этом появляются дополнительные возможности и на учебных компьютерах (в отличие от базовой версии монитора):

- загрузка с диска через сеть BASIC-программ и двоичных данных (без действий со стороны учителя);
- сброс через сеть на диск BASIC-программ (без действий со стороны учителя);
- формирование и сброс на диск через сеть протокола работы;
- электронная почта (ученик-учитель).

Учитель может определить, по своему усмотрению, возможность (или невозможность) для тех или иных учеников непосредственного обращения к диску учителя или использования электронной почты. Есть у него и средства контроля за правильностью задания имен файлов учениками.

Таким образом, учителю предоставляются на его рабочем месте разнообразные и гибкие средства управления учебным процессом, а на рабочих местах учеников появляется возможность обращения к диску как при работе в командном режиме, так и из BASIC-программ.

Монитор позволяет на КУВТ "Ямаха" проводить занятия по заранее подготовленному "плану" - командному ролику, или компануя учебный материал в ходе самого занятия. При этом возможно использование графических и текстовых "плакатов", обучающих и тестирующих программ (с сохранением протокола). За счет подкачки по сети данных и программ можно обеспечить работу ученика по индивидуальному плану или с индивидуальной скоростью.

Сетевой обмен осуществляется со скоростью 31К бод, что позволяет переслать BASIC-программу не более, чем за 10 секунд.

ВНИМАНИЕ!!! Для нормальной работы системы необходимо, чтобы в классе не было машин с совпадающими номерами и машина с номером 0 не была ученической - в стандартном классе эти условия всегда выполняются.

2

КОМПЛЕКТ ПОСТАВКИ

basmon.txt	настоящее руководство
basmon.com	монитор учителя
basmon.obj	монитор ученика
basmon.bas	программа первоначальной рассылки

КОМАНДЫ МОНИТОРА УЧИТЕЛЯ

3.1. ПЕРВОНАЧАЛЬНАЯ ЗАГРУЗКА.

Первоначальная загрузка складывается из загрузки учительского монитора с диска на учительскую машину и загрузки с диска и пересылки по сети ученической части монитора на машины учеников.

Для осуществления первоначальной загрузки необходимо:

1. Загрузить на ученической машине MSX-DOS.
2. Загрузить на ученических машинах MSX-BASIC и включить сеть (_NETINI), если ее выключали.
3. Стартовать на учительской машине COM-файл BASMON.

После этого будет автоматически осуществлена первоначальная загрузка, о чем появится сообщение.

Частичная загрузка.

При работе с монитором Вы можете нажатием клавиши <ESC> выйти из монитора учителя и работать на компьютере обычным образом. Для повторного запуска монитора учителя необходимо загрузить MSX DOS и стартовать BASMON.COM. Старт никак не повлияет на работу учеников. На те ученические машины, на которых в момент старта была включена стандартная сеть, загрузится ученическая часть монитора.

Для того, чтобы, не выходя из учительского монитора, загрузить ученический монитор на машины, куда он еще не был загружен, следует воспользоваться командой M. (Способ задания команд описан ниже).

3.2. СТРУКТУРА ЭКРАНА ГРАФИЧЕСКОГО МОНИТОРА.

После загрузки учительского монитора экран учительской машины имеет следующий вид:

СП "Диалог" 10.1988		окно 0	
"Сотрудничество"			
окно 1			
f e d c b a 9 8 7 6 5 4 3 2 1			
имя ролика:		окно 2	
окно 3			
окно 4			
окно 5			
окно 6			

В окно 0 выводятся текущие дата и время.

В окно 1 выводится информация о состоянии сети и об обращении учеников к дисковому (см. подробнее 3.4.).

В окно 2 выводится имя командного ролика (см. подробнее 3.4.).

Окно 3 содержит текущую команду монитора (см. подробнее 3.3.)

Окно 4 – рабочее. В этом окне производится набор всех команд и ответов на запросы системы.

В окно 5 выводятся все запросы и диагностические сообщения системы.

Окно 6 используется электронной почтой (см. 3.8.), для вывода подсказок (см. 3.3.) и в ряде других случаев.

3.3 КАК ЗАДАВАТЬ КОМАНДЫ.

Команды задаются монитору либо непосредственно во время работы с ним, либо предварительно помещаются в текстовый файл – "ролик" и извлекаются из него. Форматы команд в обоих случаях одинаковы.

Учителю доступны следующие команды, действие которых подробно описано ниже :

M	разослать ученический монитор
T, адресат, имя файла [,R/N]	переслать BASIC-программу или двоичный файл (от TRANSMIT).
R, адресат, имя файла	принять BASIC-программу
S, C, маска	установить статус учеников и
D, маска	умалчиваемые имена;
P, маска	
NP, имя файла	
NR, имя файла	
VT, адресат, имя файла	переслать "плакат" и перейти в режим работы с плакатами;

VR, адресат	просмотреть содержимое экрана ученика;
P, адресат, текст	послать сообщение;
D, DIR	работа с оглавлением диска;
DEL	
REN	
B, адресат, BASIC-команда	дать BASIC-команду;
L[P], адресат, n1, n2	вывести программу ученика;
BL+, адресат	заблокировать учеников;
BL-, адресат	разблокировать учеников;

Для того, чтобы дать команду во время работы надо нажать <пробел>. Тогда в окне 4 появится курсор. Наберите команду и нажмите <ввод>. Команда будет выполнена, а ее текст сохранится в окне 3. Теперь при нажатии клавиши <ввод> будет выполняться команда из окна 3, пока не будет введена новая команда.

Можно заранее записать команду в текстовый файл (каждая команда - с начала строки). Такой файл мы называем "роликом". Если нажать клавишу <INS>, монитор запросит имя файла и загрузит указанный файл. Нажимая клавиши управления курсором <вверх> и <вниз> вы перемещаетесь по "ролику", при этом активная строка с командой находится в окне 3. Для выполнения этой команды достаточно нажать <ввод>. Для окончания работы с роликом следует нажать клавишу . При работе с роликом сохраняется возможность дать непредусмотренную в ролике команду в режиме <пробел>. Имя ролика можно менять командой S.

"Ролик" позволяет заранее подготовить своего рода план занятия и значительно упрощает работу преподавателя. Ролик может быть подготовлен при помощи стандартного текстового редактора.

В команде параметры могут быть опущены. Если опускается параметр, за которым идут еще параметры, то запятое, работающие как разделители, сохраняются. Возможность недоопреде-

лять параметры позволяет формировать более гибкие "ролики".
Параметры "адресат" и "имя файла" могут быть различных типов.
Параметр "адресат", определяющий адресата, может иметь значения:

- 1-f - адресатом является ученик с указанным шестнадцатиричным номером;
- #маска - адресатами являются все ученики, которым в двоичной маске соответствует 1. Например #110110 делает адресатами машины с номерами 2, 3, 5, 6;
- # - адресатами являются ученики, определенные текущей маской (чтобы задать ее, надо нажать клавишу <SELECT> и ответить на запрос системы);
- ?маска - адресаты определяются по выбору учеников, указанных в маске. Это позволяет, например, организовать сброс программ на диск всем желающим;

Параметр "имя файла" может иметь значения:

- БУФЕР диск не используется. Данные берутся из рабочего буфера или кладутся туда же. Можно использовать для пересылки программы от одного ученика другому.
- ПРОТОКОЛ используется для записи протоколов всех учеников (см. 3.6.). При записи на диск используется текущее имя протокола, оно задается командой S (см. 3.4).

Имя файла в обычном формате используется для непосредственного обращения к диску.

РЕЖИМЫ работы монитора задаются нажатием на клавиши:

- <ESC> - выйти в DOS;
- <SELECT> - задать текущую маску;
- <INS> - сделать доступным ролик (если имя ролика не задано, то система его запросит);
- - сделать недоступным ролик;
- <Пробел> - ввести команду монитора и выполнить ее;

<Ввод> - выполнить текущую команду монитора (из окна 3);
 стрелка вверх - выбрать (в окно 3) предшествующую строку
 ролика:
 стрелка вниз - выбрать (в окно 3) следующую строку ролика;
 ? - подсказка;

3.4. ЗАДАНИЕ СТАТУСА И УМОЛЧАНИИ.

Команда S позволяет задать статус учеников и определить умалчиваемые имена файлов.

Статус ученика зависит от того, имеет ли он право обращаться к диску по своей инициативе, разрешено ли ему посылать сообщения учителю и следует ли контролировать имена файлов, при обращении к диску. Статус, таким образом, определяется маской доступа к диску, маской контроля и маской права переписки. Эти маски задаются командами:

S, D, маска	- диск;
S, P, маска	- почта;
S, C, маска	- контроль.

Как уже указывалось в 3.3, маска, или же указатель подрежима (D,P,C) могут не указываться. В этом случае они будут запрошены системой при выполнении команды. В качестве маски может использоваться текущая маска (см. 3.3.).

Текущее имя протокола и имя ролика задаются командами:

S, NP, имя	- устанавливается имя протокола;
S, NR, имя	- устанавливается имя ролика.

Использование ролика описано в 3.3 а протокола - в 3.6 .

Режим <SELECT> не является частью команды S, но близок к ней по характеру действия. Он позволяет задать текущую маску, о чем уже упоминалось в 3.3. .

3.5. ПЕРЕДАЧА ПРОГРАММ И ДАННЫХ УЧЕНИКУ С ДИСКА.

Команда T (от Transmit) предназначена для пересылки программ и данных ученику с диска через сеть. Она имеет формат:

T, адресат, имя файла [,R/N].

Монитор учителя сам определяет тип файла (BASIC - программа или двоичный файл). Если файл содержит BASIC - программу, то она должна храниться в сжатом формате, т.е. ее следует записывать на диск или средствами монитора, или командой SAVE без параметра A (т.е. не в ASCII-формате). Это стандартный для MSX способ хранения BASIC-программ.

Параметр "адресат" обычным образом (см.3.3.) задает адресатов, т.е. определяет компьютеры, на которые следует переслать информацию.

Параметр "имя файла" определяет имя файла на диске.

Параметр "R" указывает на необходимость стартовать BASIC-программу или двоичную программу. В последнем случае управление передается по стартовому адресу (байты 6 и 7 файла в соответствии со стандартом MSX для OBJ-файлов), значение "N" указывает на отказ от старта.

В качестве имени файла может быть использовано стандартное имя БУФЕР. В этом случае ученикам будет разослана программа или двоичные данные, находящиеся в буфере учительского монитора. Это те данные, которые последний раз подлежали рассылке или же были приняты от одного из учеников. Использование файла БУФЕР позволяет легко повторить рассылку, а, вместе с приемом по сети, передать программу от одного ученика другому.

Стандартное имя ПРОТОКОЛ с командой T не используется, поскольку протокол работы пересылается только от ученика учителю. Использование значения "?" в качестве параметра "адресат" позволяет провести коллективную раздачу программ. Эта возможность подробно описана в 3.7. При коллективной раздаче программ каждый из учеников, являющихся адресатами, сам

определяет имя файла для записи своей программы. У преподавателя остается, однако, возможность контролировать и корректировать имена, назначаемые учащимися (см. 3.15.).

Выполнение команды T приводит к загрузке в буфер указанного файла и пересылке его адресатам. При этом принимающие машины могут находиться как в командном, так и в программном режимах. Со стороны учеников не требуется никаких действий (кроме режима, когда имена файлов задают ученики). На время пересылки (6 секунд на каждые 16K байт) работа компьютеров-адресатов приостанавливается. Если пересылается BASIC-программа, то старая программа уничтожается. Пересылка двоичных данных позволяет переслать как данные для работы BASIC-программы, так и программу в машинных кодах. Если файл данных захватывает область памяти с &H4000 по &H7FFF, пересылка не осуществляется (см. руководство программиста 2.4.).

3.6. ПРИЕМ И ЗАПИСЬ НА ДИСК ПРОГРАММ УЧЕНИКОВ.

Команда R (от Receive) предназначена для сброса на диск программ учеников и протоколов работы. Она имеет формат:

R, адресат, имя файла

Параметр "имя файла" определяет имя файла на диске, в который будет записана программа. Стандартное имя ПРОТОКОЛ означает, что сбрасывается протокол работы ученической программы (см. в 4.4. о работе с протоколом). Имя файла при сбросе протокола определяется текущим именем протокола, которое устанавливается командой

S, NP, имя файла.

Стандартное имя БУФЕР означает, что программа ученика останется в буфере учительского монитора и не будет записана на диск. Непосредственно из буфера программа может быть затем передана другим ученикам.

Параметр "адресат" указывает ученика, у которого следует взять программу. В команде R не могут быть использованы значения "#" и "#маска", поскольку за один раз можно забрать

программу только у одного из учеников. Можно, однако, воспользоваться значением "?" параметра "адресат". В этом случае ученики сами указывают имена файлов, в которые будут записаны их программы (см. подробнее 3.7.).

При выполнении команды R работа компьютера указанного ученика будет приостановлена вне зависимости от того, находится он в программном или командном режиме. Данные будут пересланы на центральный компьютер и, при необходимости, записаны на диск под указанным именем.

3.7. КОЛЛЕКТИВНЫЙ СБОР И РАЗДАЧА ПРОГРАММ.

Как уже упоминалось (в 3.5. и 3.6.), команды T и R обеспечивают режим коллективного сбора и рассылки программ. Для этого в качестве значения параметра "адресат" следует задавать "?" или "? маска". В случае "?" в качестве маски используется текущая маска.

Ученикам, указанным в маске, предлагается указать имя файла. При этом может быть указано не любое имя, а только соответствующее предложенному учителем трафарету. Дело в том, что учитель сам указывает некоторое имя файла и расширение имени с использованием "*" и "?".

Например:

A10*.B*

Ученик может дополнить справа имя файла и расширение, т.е. дать имена: A10GAME .B2 или A10NEW .BAS. Это позволяет унифицировать имена, что бывает удобно, когда программы разных групп учащихся находятся на одном диске. Указанные учителем префиксы могут использоваться в качестве идентификатора групп учащихся.

После указания имени ученик может продолжать работу. Ниже (4.2 и 4.3.) будут описаны возможности ученика по самостоятельному обмену с диском.

3.8. ЭЛЕКТРОННАЯ ПОЧТА.

Сетевой монитор позволяет учителю обмениваться сообщениями с учениками.

Сообщения учителя отправляются командой

R, адресат, сообщение.

Параметр "адресат" может иметь в качестве своего значения либо шестнадцатиричный номер компьютера ученика, либо маску, заданную явно (т.е. "# маска"), или текущую (т.е. "#").

Сообщение - это символьная строка длиной до 255 символов. Если команда набирается заранее (через "ролик"), длина сообщения ограничена длиной команды, которая не должна превышать 30 символов.

При задании команды можно не указывать текст сообщения (впрочем, это относится ко всем параметрам), тогда монитор запросит текст сообщения при выполнении команды. Набор текста сообщения осуществляется в наиболее крупном окне 6.

При получении учеником сообщения от учителя, экран ученического компьютера очищается и на него выводится текст сообщения. При нажатии на клавишу <пробел> содержимое экрана восстанавливается и работа продолжается как ни в чем не бывало.

Временная очистка экрана и вывод сообщения никак не сказываются на работе программы на ученическом компьютере.

Сообщения учеников выводятся в окно 6 учительского дисплея. Им всякий раз предшествует информация о номере ученика, приславшего сообщение. Учитель при помощи команды S может запретить или разрешить ученикам пользоваться почтой.

3.9. ПРОСМОТР СОДЕРЖИМОГО ЭКРАНОВ УЧЕНИКОВ.

При работе в классе бывает полезно просмотреть содержимое экрана того или иного ученика. Это позволяет сделать команда VR (от Video Receive), имеющая формат:

VR, адресат

В качестве значения параметра "адресат" может быть использован только номер (шестнадцатиричный) компьютера ученика.

Содержимое экрана ученика при выполнении этой команды пересылается на экран учителя. В зависимости от экранного режима, это занимает от 2 до 6 секунд. Изображение сохраняется на экране учителя до нажатия на какую-либо клавишу. Ученический компьютер сразу после завершения пересылки продолжает работать.

3.10. РАБОТА С "ПЛАКАТАМИ" ("Лектор").

При проведении занятий, включающих лекционную часть, удобно пользоваться заранее подготовленными "плакатами" - графическими и текстовыми экранами, записанными на диск. При этом изображение выводится и на учительский, и на ученический экраны. Монитор позволяет синхронно манипулировать "указкой" на экранах учителя и учеников, впечатывать текст как на текстовые, так и на графические экраны. Можно менять цвет указки. Таким образом, поддерживаются возможности, которые обычно предоставляет кодоскоп. Объединив серию плакатов и обучающих и тестовых программ в одном ролике, можно подготовить полноценный материал для проведения занятия. Для подготовки плакатов можно использовать графические редакторы, либо записывать содержимое экрана, сформированное BASIC - программой, оператором "BSAVE, S" (подробнее см. руководство по MSX-BASIC).

Вход в режим работы с плакатами осуществляется по команде VT (от Video Transmit), имеющей формат:

VT, адресат, имя файла [, экр. режим].

Параметр "адресат" в качестве значения допускает номер ученика (16-ричный) или маску - заданную явно, или текущую.

Экранный режим показывает, в каком экранном режиме записан экранный файл. Допустимы экранные режимы от SCREEN1 до SCREEN8. По умолчанию - SCREEN1.

При создании текстовых изображений следует задавать WIDTH 29 для SCREEN1.

По команде изображение загружается на экран учителя, и монитор переходит в режим работы с плакатами. Теперь допустимы следующие команды:

- <ESC> выйти из режима плакатов;
- <ввод> переслать изображение указанным ученикам. Если на машинах учеников выполнялась программа, она будет остановлена без сохранения данных. Все дальнейшие команды выполнимы только после применения команды <ввод>, т.е. когда содержимое учительского и ученических экранов идентично;
- клавиши со стрелками перемещают указку (курсор) по экранам учителя и учеников;
- <select> изменение цвета указки (цвет меняется циклически);
- <буквы> печать символов на экранах учителя и учеников в позиции указки. В режиме Screen2 цвет букв определяется цветом указки.

3.11. РАБОТА С ОГЛАВЛЕНИЕМ ДИСКА.

Монитор учителя позволяет работать с оглавлением диска без выхода в DOS. Это обеспечивается командой D (от Directory), имеющей формат, который зависит от выбранного подрежима.

Если подрежим не указан сразу, он, как и другие параметры, будет запрошен монитором.

D, DIR, имя файла

обеспечивает вывод в окно 6 учительского экрана оглавление диска. Как и в соответствующей команде MSX DOS, имя файла может содержать знаки "*" и "?", имеющие тот же смысл.

D, DEL, имя файла

аналогично команде DEL из MSX DOS позволяет удалять файлы с диска.

D, REN, имя исходное, имя результирующее

позволяет переименовывать файлы аналогично команде REN из MSX DOS.

3.12. БЛОКИРОВКА КОМПЬЮТЕРОВ УЧЕНИКОВ

Учитель может заблокировать некоторые ученические компьютеры командой:

BL+, адресат

или же разблокировать компьютеры командой

BL-, адресат

Блокировка и разблокировка позволяют, например, обеспечить одновременное начало работы всех учеников с разными вариантами тестирующих программ. Для этого надо заблокировать учеников, затем разослать им варианты тестовых заданий и разблокировать всех сразу.

Пример:

BL+, #1111111111111111	блокировка всех учеников
T, #100100100100100, test1.bas	рассылка ученикам 3, 6, 9, 12, 15
T, #010010010010010, test2.bas	рассылка ученикам 2, 4, 7, 10, 13
T, #001001001001001, test3.bas	рассылка ученикам 1, 5, 8, 11, 14
BL-, #1111111111111111	разблокировка
B, #1111111111111111, RUN	запуск всех тестовых программ

Команда B стартует программу учеников. Подробнее она описана ниже.

Заблокированный компьютер не реагирует ни на какие действия ученика (кроме перезагрузки), но продолжает поддерживать сетевой обмен.

3.13. ПЕРЕДАЧА BASIC-КОМАНД

Команда B позволяет заставить ученические компьютеры перейти в командный режим и выполнить указанную команду. Ее формат:

B, адресат, команда

где команда является обычной BASIC-командой.

3.14. ВЫВОД BASIC-ПРОГРАММЫ УЧЕНИКА

Команды L (от List) и LP (от ListPrinter) предназначены для вывода соответственно на экран учителя или на принтер BASIC-программы ученика.

Они имеют форматы:

L, адресат, строка1, строка2

LP, адресат, строка1, строка2

В качестве параметра "адресат" допускается только номер (16-ричный) компьютера ученика. Команды выводят строки BASIC-программ выбранного ученика от строки1 до строки2.

Команда L осуществляет вывод в окно 6 учителя, а LP- на принтер учителя.

Нажатие на <ПРОБЕЛ> приостанавливает вывод, а повторное нажатие возобновляет его. Нажатие на <ESC> прекращает вывод.

3.15. ВОЗМОЖНОСТИ УЧИТЕЛЯ В СВЯЗИ С ОБМЕНОМ ПО ИНИЦИАТИВЕ УЧЕНИКОВ.

Возможности учеников подробнее описаны в 1.4.. Заметим здесь только, что ученики могут по своей инициативе, т.е. без действий со стороны учителя, записывать программы на диск и считывать с диска программы и двоичные данные. Они могут также посылать учителю сообщения. Все эти действия доступны ученику как из командного, так и из программного режимов.

Сообщения учеников выводятся в окно 6 у учителя (см.3.8).

Любые обращения к диску по инициативе ученика находят отражение в окне 1. Наблюдая за этим окном, учитель получает информацию об обращении учеников к диску.

Кроме того, учитель может установить статус учеников (см. 3.4). Это, во-первых, определит для каждого из учеников возможность посылать сообщения и обращаться к диску, а, во-вторых, позволяет учителю выделить учеников, обращение которых к диску следует контролировать.

Если ученик, подлежащий контролю, обращается к диску (кроме ПРОТОКОЛА), монитор просит учителя подтвердить правиль-

ность назначенного учеником имени, или же это имя исправить.

Протоколы, поступающие от учеников, записываются в один последовательный файл, имя которого устанавливается командой S, NP, имя .

По умолчанию ПРОТОКОЛ.txt. Структура этого файла описана в главе системного программиста.

Первоначально все ученики имеют право отправки сообщений и не имеют права обращения к диску.

КОМАНДЫ УЧЕНИКА

4.1. ВОЗМОЖНОСТИ УЧЕНИКА. ПОДСКАЗКА.

Сетевая система на ученическом месте берет на себя обработку всех команд, поступающих от учительского монитора. Кроме того, она предоставляет ученику дополнительные команды языка BASIC. Расширение языка BASIC позволяет через сеть записывать BASIC-программы на диск, считывать их с диска, считывать с диска двоичные данные, а также сохранять протокол работы программы. Расширение языка работает, только когда на учительском месте работает сетевой монитор.

Ученик может получить справку о дополнительных командах, задав:

CALL NHELP или

_NHELP

Справка выдается только в командном режиме и в программном режиме смысла не имеет. В справке содержится информация о том, какие команды допустимы в настоящее время (в соотв. с командой S учителя).

4.2. ЗАГРУЗКА ПРОГРАММ И ДАННЫХ С ДИСКА ЧЕРЕЗ СЕТЬ.

Ученик может загрузить с диска BASIC-программу или двоичные данные задав:

_NLOAD (имя файла [,R])

"Имя файла" - это строчная переменная или константа, содержащая имя файла на диске.

Например:

a\$ = "program.bas"

_NLOAD (a\$)

или

_NLOAD ("program.bas")

Добавление "R" вызывает, кроме того, старт программы.

Если файл с указанным именем содержит двоичные данные или программу, то они и будут загружены.

Это позволяет подгружать данные по ходу работы программы.

Если в момент выполнения оператора _NLOAD дисковод недоступен ученику, (см. 3.4) или обращение к диску на машине учителя прошло ненормально, возникает синтаксическая ошибка и выдается соответствующее сообщение.

4.3. СБРОС ПРОГРАММ ПО СЕТИ НА ДИСК.

Ученик может самостоятельно сбросить на диск BASIC-программу. Для этого предназначена команда:

_NSAVE (имя файла).

Ее действие аналогично команде _NLOAD, но возможен только сброс BASIC-программ.

4.4. РАБОТА С ПРОТОКОЛОМ.

На ученическом компьютере обеспечиваются также BASIC-команды, поддерживающие протоколирование работы BASIC-программы. У учителя на диске ведется единый протокол для всех учеников. Его имя устанавливается командой S учительского монитора. Программа на ученическом месте может завести буфер протокола и, при необходимости, сбрасывать его на диск в общий протокол. При этом к каждой записи присоединяются номер машины, приславшей запись и маркер (структуру протокола см. в Руководстве программиста).

_NCBUF (нач. адр., длина, маркер)
создает (CREATE) буфер с указанного адреса указанной длины.

Маркер - строка символов (имеют значения только пер-

вые 20), которая присоединяется к каждой записи протокола данного компьютера.

_NDBUF

уничтожает (DELETE) буфер.

_NSBUF

скидывает (SAVE) буфер по сети на диск, присоединяя номер ученика и маркер.

Ниже приведен фрагмент программы, использующей протоколирование:

```
10 INPUT "Ваша фамилия"; NAME$
20 DIM N% (100)
30 _NCBUF (VARPTR (N%(0)), 202, NAME$): ' Массив N% объяв-
    лен буфером протокола, длина 202, Name$ - маркер
.....
1000 _NSBUF: ' запись протокола
.....
2000 _NSBUF: ' запись протокола
2010 _NDBUF: ' конец работы с протоколом
```

4.5. КОЛЛЕКТИВНЫЕ СБОР И РАЗДАЧА ПРОГРАММ.

Коллективные сбор и раздача программ осуществляются по команде учителя. Компьютер ученика при этом приостанавливает работу, а на экране появляется предложение задать имя программы для сбора или раздачи. При этом следует пользоваться предложенным трафаретом для имени. Трафарет может однозначно задавать начала имени файла и расширения имени.

Пример:

Трафарет A10?????.B?? позволяет задавать имена:

A10PROG1.BAS

или A10TEST .BIS

4.6. ЭЛЕКТРОННАЯ ПОЧТА.

Ученик может обмениваться сообщениями с учителем. Если от учителя приходит сообщение, работа ученика приостанавливается и сообщение выводится на экран. После того, как ученик нажмет какую-либо клавишу, восстанавливается прежнее содержимое экрана и работа будет продолжена.

Для отправки сообщения учителю, ученик должен задать:

`_NPOST` (сообщение)

где "сообщение" - есть строчная константа или переменная.

Сообщения могут отправляться как из программного, так и из командного режимов.

5

ГЛАВА СИСТЕМНОГО ПРОГРАММИСТА

5.1. ОБЩАЯ СТРУКТУРА МОНИТОРА И ЕГО РАЗМЕЩЕНИЕ В ПАМЯТИ

В состав монитора входят Монитор учителя, Монитор ученика и программа первоначальной загрузки.

Монитор учителя осуществляет прием и выполнение команд учителя, а также сканирование машин учеников и выполнение их запросов. Программа учительского монитора работает под управлением MSX DOS и размещается в памяти до адреса 6000h, что обеспечивает существование буфера, достаточного по объему для размещения любых данных, доступных ученику.

Монитор ученика обеспечивает интерпретацию и выполнение расширенных команд языка BASIC и выполнение команд учительского монитора. Он занимает стр.1 оперативной памяти, т.е. RAM, слот 3, расш.2 (с 4000h по 7FFFh). Монитор ученика устанавливает свой HOOK на H_KEYI. Все системные переменные используются только в рамках стандарта MSX.

Программа первоначальной рассылки обеспечивает прием из сети пересылаемого следом за ней монитора ученика.

5.2. СТРУКТУРА ПРОТОКОЛА И ЕГО АНАЛИЗ

Протокол является последовательным файлом, создаваемым на базе протоколов, присылаемых с ученических мест. Он может быть в дальнейшем обработан таким образом, что записи будут отсортированы по номерам машин учеников, или же по маркерам, заданным обучающими программами.

Протокол имеет следующий формат:

имя файла (до 00h)

длина 1 записи (26)

машины (16)

маркер (до 00h)

данные

длина 2 записи

...

...

Поле "имя файла" записывается при создании файла и содержит имя, под которым файл был первоначально создан. Поле терминируется символом chr\$(0).

Поле "длина записи" занимает 2 байта и содержит суммарную длину трех последующих полей.

Поле "# машины" длиной 1 байт содержит номер ученического компьютера, приславшего данную запись.

Поле "маркер" содержит маркер, указанный в обучающей программе при открытии буфера протокола (_NCBUF). Этот маркер может, например, содержать фамилию учащегося, или название обучающей программы. Это поле терминируется символом chr\$(0).

Поле "данные" содержит данные из области буфера протокола, скинутые командой (_NSBUF)

Записи расположены в файле в соответствии с порядком их сброса на диск, но поля "#машины" и "маркер" позволяют проводить разного рода сортировки.

5.3. ОГРАНИЧЕНИЯ НА ПРОГРАММЫ, РАБОТАЮЩИЕ С СИСТЕМОЙ

Сетевой монитор накладывает некоторые ограничения на программы, работающие с ним. Эти ограничения связаны с тем, что монитор ученика расположен в ОЗУ, и с тем, что он использует локальную сеть монопольно. Поэтому программы, работающие с ним,

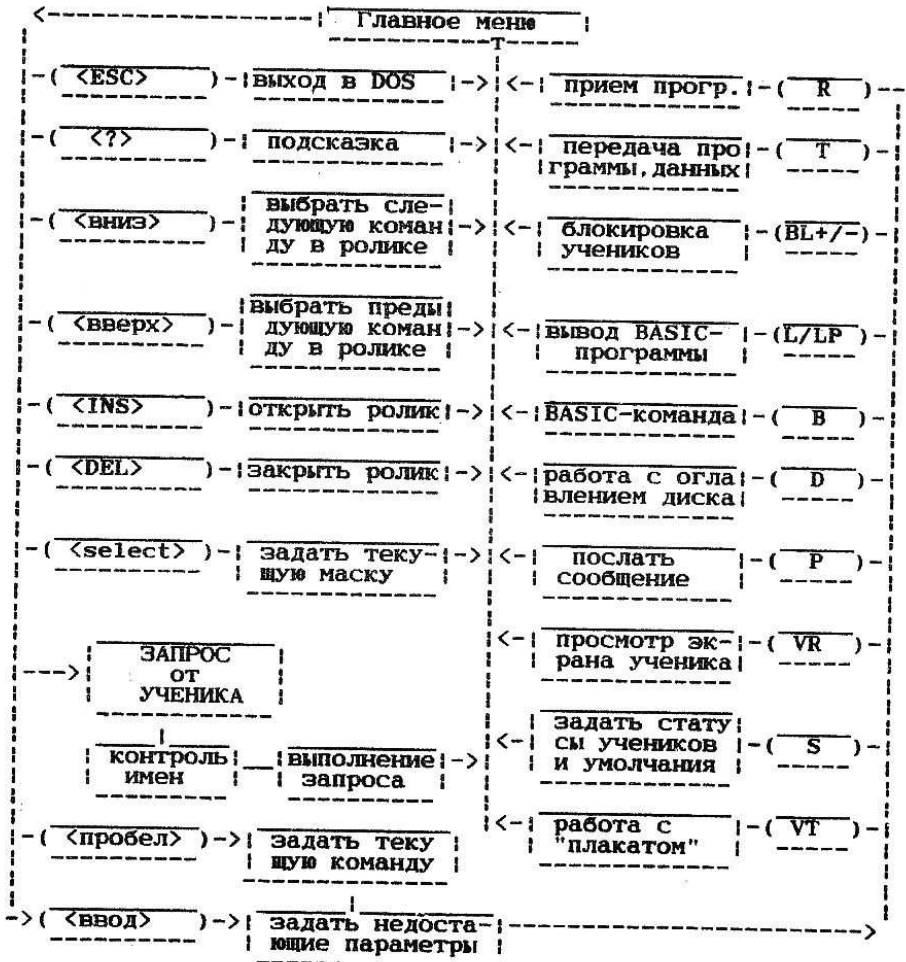
- не должны изменять `HOOK H_KEY1`
- не могут использовать стандартное сетевое расширение языка BASIC.
- не должны модифицировать область оперативной памяти, в которой лежит Монитор ученика.

Не следует также выключать компьютер, работающий с сетью - в этом случае принимающая машина зависнет. Зависание снимается одновременным нажатием на клавиши `<HOME>`, `<INS>`, ``.

При включении Сетевого монитора производится инициализация электронного диска. Однако его повторная инициализация оператором `_MEMINI` становится невозможной.

ПРИЛОЖЕНИЯ

6.1. СТРУКТУРА УПРАВЛЕНИЯ МОНИТОРОМ УЧИТЕЛЯ



6.2. КОМАНДЫ МОНИТОРА УЧИТЕЛЯ

М	разослать ученический монитор
T, адресат, имя файла [,R/N]	переслать BASIC-программу или двоичный файл (от TRANSMIT).
R, адресат, имя файла	принять BASIC-программу
S, C, маска	установить статусы учеников и умалчиваемые имена.
D, маска	
P, маска	
NP, имя файла	
NR, имя файла	
VT, адресат, имя файла	переслать "плакат" и перейти в режим работы с плакатами.
VR, адресат	просмотреть содержимое экрана ученика
P, адресат, текст	послать сообщение.
D, DIR	работать с оглавлением диска
DEL	
REN	
B, адресат, BASIC-команда	дать BASIC-команду.
L[P], адресат,n1,n2	вывести программу ученика
BL+, адресат	заблокировать учеников.
BL-, адресат	разблокировать учеников.

6.3. ПРИМЕР КОМАНДНОГО РОЛИКА

Нижe приведен пример "командного ролика" для проведения занятия, состоящего из трех частей: лекционная часть, контрольная работа и самостоятельное программирование учащимися на языке BASIC. При работе с этим роликом в Мониторе учителя следует выбирать нужные команды и запускать их выполнение. Команды в ролике скомпонованы таким образом, что можно исполнять их последовательно. Возможно, однако, и задание дополнительных команд в ходе проведения занятия. После первых двух частей занятия предусмотрена возможность перехода к работе с другим роликом ("ролик2"). Если Вы не хотите переходить к другому ролику, пропустите эту команду.

Лекционная часть

VT, #1111111111111111, плакат1, 2	Разослать всем экранный файл "плакат1" (в screen2)
VT, #, плакат2, 1	Разослать всем экранный файл "плакат2" (в screen1)
T, #, тренажер, R	Разослать BASIC-программу "тренажер" и стартовать ее

Контрольная работа

S, NP	Протокол хранить в файле, имя которого Монитор запросит при работе с роликом.
BL+, #	Заблокировать компьютеры уч-в
T, #101010101010101, тест1.bas	Раздать тестовые программы
T, #010101010101010, тест2.bas	
BL-, #1111111111111111	Разблокировать компьютеры уч-в
R, #, ПРОТОКОЛ	Собрать несданные протоколы
S, NR, ролик2	Переход к работе с другим роликом, из файла "ролик2"

Работа учеников со своими BASIC-программами

T, ?, *, .bas	Загрузка BASIC-программ по выбору учеников
VR,	Просмотр содержимого экрана ученика, номер которого будет запрошен Монитором
P, #, Заканчивайте работу	Сообщение ученикам
R, ?, *, .bas	Сбор BASIC-программ учеников

6.4. РАСШИРЕНИЕ КОМАНДЫ ЯЗЫКА BASIC У УЧЕНИКА

-NHELP	подсказка
-NLOAD(имя файла [,R])	загрузка файла по сети
-NSAVE(имя файла)	сброс программы на диск
-NCBUF(начало, длина, маркер)	создание буфера протокола
-NDBUF	отмена буфера протокола
-NSBUF	сброс содержимого буфера на диск
_NPOST(сообщение)	передача сообщения учителю

6.5. ПРИМЕР ПРОГРАММЫ ФОРМИРОВАНИЯ ПРОТОКОЛА

```
10 REM анкетные данные
20 A$="Анкета1"
30 CLEAR 200,&HC000
40 AD%=&HC000      : '  адрес буфера протокола
50 _NCBUF(AD%,100,a$):' создать буфер протокола
60 INPUT "ФАМИЛИЯ, ИМЯ":A$
70 GOSUB 1000:' положить A$ в буфер
80 _NSBUF:'        сбросить буфер на диск
90 INPUT "АДРЕС":A$
100 GOSUB 1000
110 _NSBUF
120 _NDBUF:'       закрыть буфер
130 END
1000 REM положить A$ в буфер
1010 FOR I=AD% TO AD%+100
1020  POKE I,0
1030 NEXT
1040 A1=VARPTR(A$)
1050 FOR I=1 TO PEEK(A1)
1060  POKE AD%+I-1 , ASC(MID$(A$,I,1))
1070 NEXT
1080 RETURN
```

6.6. ПРИМЕР ПРОГРАММЫ ОБРАБОТКИ ПРОТОКОЛА

```

10  REM    распечатка протокола
20  OPEN "ПРОТОКОЛ.TXT" FOR INPUT AS #1
30  A$ = STR$(100)
40  GOSUB 1000: ' чтение имени файла
50  LPRINT "имя протокола-";A$
60  GOSUB 2000: ' чтение длины записи
70  LN = N
80  NOM = ASC(INPUT$(1,#1)): 'номер ученика
90  LPRINT"компьютер# "; NOM
100  GOSUB 1000: ' чтение маркера
110  LN = LN-N-3: ' длина поля данных
120  LPRINT"маркер: "; A$
130  REM в данном случае поле данных - символьная строка
140  LPRINT "данные: "; INPUT$(LN,#1)
150  GOTO 60
1000 REM чтение символьной строки до CHR$(0) в A$(длина N)
1010 A$ = " ": N = 0
1020 B$ = INPUT$ (1,#1): N = N+1
1030 IF B$ = CHR$(0) THEN RETURN
1040 A$ = A$+ B$
1050 GOTO 1020
2000 REM чтение целого в N
2010 IF EOF(1) THEN END
2020 B$ = INPUT$(1,#1) : N = ASC(B$)
2030 B$ = INPUT$(1,#1) : N = N+256*ASC(B$)
2040 RETURN

```

II

СЕТЕВАЯ

ПАСКАЛЬ-СИСТЕМА

ВЕРСИЯ 2.1.

2

3

4

5

6

7

8

9

10

НАЗНАЧЕНИЕ.

Сетевая Паскаль - система (версия 2.1.) предназначена для организации работы с компиляторами языка Паскаль фирмы HiSoft на сети базового КУВТ MSX-2 "YAMANA". Высокая скорость пересылки (в 3,5 раз выше стандартной), встроенный строчный редактор, возможности использования готовых пакетов процедур, работы с диском на ученическом компьютере, а также создания объектных модулей, работающих как под MSX DOS, так и под Бейсик-системой, делают монитор мощным и незаменимым средством как для обучения языку Паскаль, так и для разработки прикладных программ.

С помощью системы Вы можете:

- Проводить обучение на языке Паскаль в базовом КУВТ; не выходя из системы, создавать, модифицировать, компилировать и выполнять программы на Паскале; переслать любому ученику текст программы, пакеты процедур; контролировать ход его работы над программой; сохранить результат работы учеников на диске.

- Превратить Ваш базовый класс в инструментальный, создавать быстродействующие обучающие и игровые программы профессионального класса на ученических компьютерах; в режиме опроса ученических компьютеров на них доступны команды, позволяющие загрузить текст программы с диска, подгрузить необходимые пакеты процедур, сохранить на диске результаты работы: тексты и объектные коды; созданные объектные коды можно преобразовать с помощью конвертора, входящего в систему, к формату двоичного Basic-файла, который загружается командой `load`; из тех же текстов программ Вы можете также получать программы, работающие под MSX DOS (COM-файлы).

- Создавать программы, работающие с сетью, используя входящий в систему пакет процедур; разрешающих программе на ученическом компьютере работать с диском и

пересылать области памяти на другой ученический компьютер, что позволяет создавать коллективные обучающие программы с протоколированием хода работы и сохранением результатов.

Прежде, чем читать дальше настоящее руководство, Вам необходимо изучить фирменные руководства по дисковой и магнитофонной версиям компилятора HiSoft Pascal. Магнитофонная версия работает на ученическом компьютере, а к дисковой обращается монитор учителя.

Желаем Вам приятной работы и успехов!

2

СОСТАВ И ДИСТРИБУЦИЯ.

Паскаль система состоит из сетевого монитора, исполняющей системы, конвертора объектных модулей и пакета сетевых процедур.

На поставляемой дискете находятся следующие файлы:

- а. Сетевой монитор:
 - PASSYS.COM - монитор учителя;
 - SPSYS.OBJ - монитор ученика.
- б. Исполняющая система:
 - EXEC.COM
 - SPEXEC.OBJ.
- в. Конвертор объектных модулей:
 - CONV.COM
 - RUNTIME.OBJ.
- г. Пакет сетевых процедур:
 - STNET.PAS.

В комплект поставки входят настоящее руководство .

На вашей рабочей дискете, кроме того, необходимо разместить файлы MSXDOS.SYS и COMMAND.COM операционной системы MSX DOS фирмы Microsoft (использование других операционных систем не допускается!) и компиляторы HiSoft Pascal

под именами HP80.COM - дисковая версия, HPMSX.OBJ - магнитофонная версия.

3

ЗАГРУЗКА СИСТЕМЫ.

Для загрузки системы необходимо загрузить на учительском компьютере MSX DOS и запустить программу PASSYS.COM

A>PASSYS

После загрузки монитора, о чем свидетельствует появление заголовка и приглашения:

]

дать команду

]I <BK>

Если загрузка прошла нормально, то на ученических компьютерах загрузится ученический модуль монитора, а затем магнитофонная версия HiSoft Pascal. Следует отметить, что загрузить один или несколько ученических компьютеров можно в любой момент работы. Сетевой протокол устойчив к работе сети с другими протоколами.

4

КОМАНДЫ УЧИТЕЛЯ.

Команды сетевого монитора, имеют тот же формат, что и команды магнитофонного HiSoft Pascal. Часто аналогичные команды совпадают, что облегчает работу с монитором.

Команды учительского монитора имеют формат

]Cn1,n2,string <BK>

где C- односимвольный код команды, n1 и n2 - целые положительные числа, string - символьная строка без пробелов.

Параметры могут быть опущены. В описании обязательные

параметры заключены в угловые скобки, а параметры, игнорируемые командой, опускаются.

Список команд можно получить по команде H (Help).

Ошибка в коде команды диагностируется сообщением

```
Illegal command
```

Ошибка в команде вызовет сообщение

```
Pardon?
```

Примеры команд:

```
]H
```

```
]G2,4,prog.pas
```

```
]@9
```

```
]F,,*.com
```

4.1. КОМАНДЫ ВСТРОЕННОГО РЕДАКТОРА.

4.1.1. Непосредственный ввод строк.

Если команда начинается с цифры, то она интерпретируется как ввод в текст программы строки с указанным номером. Как и в магнитофонной версии HiSoft Pascal строки нумеруются редактором, но при компиляции номера строк игнорируются. Таким образом, можно разносить по разным строкам различные куски операторов языка Pascal, комментарии и т.д.

Длина строки не может быть больше 80 символов!.

Введенная строка или вносится в текст с учетом ее номера, или замещает строку с этим номером. Если после номера строки нет текста, то соответствующая строка удаляется (как в Бейсике).

Пример.

```
]10 PROGRAM ABC;
```

```
]20 BEGIN
```

```
]30 WRITELN ('HALLO,WORLD!')
```

```
]40 WRITE ('????????')
```

```
]50 END.
```

```
]40
```

В этом примере показан ввод текста небольшой программы и удаление из него строки с номером 40.

4.1.2. Просмотр текста программы.

Команда L (List).

Формат Ln1,n2 <BK>

С помощью этой команды можно просмотреть фрагмент текста программы со строки n1 по строку n2. Если опущены оба параметра, то будет показан весь текст со строки с наименьшим номером. Если опущен только второй параметр, то Вы увидите текст со строки с номером n1. Выполнение команды можно прервать нажатием на клавишу <ESC>.

Если в памяти нет программы, то будет выдано сообщение

buffer empty

Если n1 больше номера последней строки, то сообщение

Pardon?

Пример:

]L10,50

(Просмотр строк с 10-й по 50-ю.)

4.1.3. Распечатка текста программы.

Команда Z (print).

Формат Zn1,n2 <BK>

Эта команда полностью аналогична L, но она выведет текст программы на принтер.

Пример:

]Z

(Вывод на принтер текста программы в буфере.)

4.1.4. Редактирование строки программы.

Команда E (Edit).

Формат E<n1> <BK>

С помощью этой команды можно отредактировать строку программы с номером n1. Если строки с указанным номером нет, то система даст сообщение

Pardon?

В противном случае, будут напечатаны редактируемая строка, а на следующей строчке экрана - ее номер и редактируемая строка разместится в буфере ввода, что позволит осуществить ее модификацию стандартными средствами DOS. На-

пример, при нажатии клавиши -> редактируемая строка будет посимвольно выводиться на экран и т.д. (см. руководство по MSX DOS).

Пример:

JE10

(Редактирование строки с номером 10.)

4.1.5. Удаление строк программы.

Команда D (Delete).

Формат D<n1,n2> <BK>

Эта команда удалит строки с номерами от n1 по n2. Если в этом промежутке нет ни одной строки, то система выдаст сообщение

Pardon?

Пример:

JD50,200

(Удаление строк с номерами от 50 до 200.)

4.1.6. Перенумерация строк программы.

Команда N (reNumber).

Формат N<n1,n2> <BK>

Эта команда перенумеровывает строки программы начиная с номера n1 с шагом n2. Если шаг слишком велик, то есть номер строки в процессе перенумерации превзошел 65536, то строки перенумеруются с 1 с шагом 1.

Пример:

JN5,10

(Первой строке будет присвоен номер 5, последующим - 15, 25, ...)

4.1.7. Последняя строка программы.

Команда U (Upper line).

Формат U <BK>

С помощью этой команды можно узнать номер последней строки программы. Если программы нет в буфере, то система напечатает

0.

Пример:

10

30

(Номер последней строки программы-30.)

4.2. КОМАНДЫ ДЛЯ РАБОТЫ С ДИСКОМ.

4.2.1. Просмотр оглавления диска.

Команда F (Files).

Формат F, <string> <BK>

Эта команда выведет на экран имена файлов на диске, соответствующие шаблону string. Например, по команде

F, *.PAS

будут напечатаны имена файлов с расширением PAS, а по команде

F, *.*

- имена всех файлов на диске.

4.2.2. Загрузка текста программы.

Команда G (Get).

Формат G n1, n2, string <BK>

Команда G предназначена для загрузки файла с текстом PASCAL программы в код ASCII с диска в буфер учительского компьютера. Параметры n1 и n2 указывают начальный номер строки и шаг нумерации загружаемой программы (на диске программа хранится, естественно, без номеров строк!) Строка string содержит имя программы. Если опущен параметр n1, то он наследуется от предыдущей команды. Если опущен параметр n2, то он принимается равным 10. Если опущена строка string, то загружается "рабочий файл" @@@@@@@@.PAS. Файл с этим именем создается, например, командой C(Compile) без параметра (см. 4.3.2). Если загружаемого файла нет на диске, то система сообщит

Can't open file <string>

Если файл слишком велик, и не сможет поместиться в памяти ученического компьютера, то система сообщит:

File <string> too big

Во время загрузки выдается информационное сообщение

Get file <string>

Следует заметить, что при загрузке происходит преобразование текста во внутренний формат ученического редактора HiSoft Pascal, поэтому время загрузки несколько больше, чем в обычных текстовых редакторах.

Пример:

1G5,10,prog.pas

(Загрузка текста программы из файла prog.pas и перенумерация строк с номера 5 с шагом 10.)

4.2.3. Сохранение текста программы на диске.

Команда P (Put).

Формат Pn1,n2,<string>

С помощью этой команды можно записать текст программы из буфера системы на диск. Параметры n1 и n2 указывают диапазон записываемых на диск строк. Если они опущены, то будет записана вся программа. Параметр string- имя файла на диске, он обязателен. При его отсутствии система сообщит:

name not spec.

Если в буфере нет программы, то система сообщит:

buffer empty

Если в указанном диапазоне нет ни одной строки, то

n1 too big

Если нет возможности создать файл на диске, то

Can't open file <string>

Во время записи файла система выдает информационное сообщение

Put file <string>

Пример:

1P,.prog.pas

(Запись на диск всего текста в буфере в файл prog.pas.)

4.2.4. Подгрузка текста программы.

Команда M (Merge).

Формат M<n1>,n2,<string>

С помощью этой команды можно вставить файл string с дис-

ка внутрь программы в буфере после строки n1 и перенумеровать строки со строки n1 с шагом n2.

Отсутствие параметров n1 или n2 вызовет сообщение
parameter error

Если n1 больше номера последней строки, то текст из файла загрузится в конец буфера.

Если на диске нет файла <string>, то система сообщит:

Can't open <string>

Во время загрузки система выдает информационное сообщение

Merge file <string>

Пример:

JM27,3,proc.pas

(После строки с номером 27 загружается файл proc.pas; строки с 27 нумеруются с шагом 3.)

4.2.5. Разрешение директив компилятора \$F.

Команда K (link).

Формат K <BK>

Эта команда автоматически подгружает файлы, указанные в директиве \$F компилятора. Имя файла не должно совпадать с ключевым словом языка Pascal.

При подгрузке сообщения системы совпадают с сообщениями при выполнении команды M.

В тексте программы \$F всюду заменится на +F, а строки перенумеруются с 2 с шагом 2.

Пример:

JK

4.3. КОМПИЛЯЦИЯ И ВЫПОЛНЕНИЕ.

4.3.1. Установка опций компилятора.

Команда O (Options).

Формат O <BK>

Эта команда позволяет задать опции компиляции на учительском компьютере. При ее выполнении система напечатает:

Default options: <текущие опции>

New options:

после чего необходимо ввести строку опций (см. фирменное руководство HiSoft Pascal).

При загрузке системы умалчиваемая опция Y.

4.3.2. Компиляция программы.

Команда C (Compile).

Формат C, ,string <BK>

С помощью этой команды можно компилировать программу, содержащуюся в файле <string> с опциями, заданными командой O. Если параметр string опущен, то программа в буфере будет сохранена в рабочем файле @@@@.PAS и откомпилирована.

После окончания компиляции произойдет автоматический возврат в систему, но буфер будет пуст!

Сообщения системы при выгрузке буфера на диск совпадают с сообщениями команды P, сообщения компилятора описаны в руководстве по HiSoft Pascal.

Пример:

]C

Эта команда эквивалентна следующим двум:

]P, .@@@@@.pas

]C, .@@@@@.pas

4.3.3. Выполнение программы.

Команда R (Run).

Формат R, ,string <BK>

Эта команда выполнит программу, содержащуюся в файле string.COM или @@@@@.COM, если параметр опущен.

После окончания выполнения произойдет автоматический возврат в сетевую Паскаль-систему, но буфер будет пуст!

Пример:

]R, .SCED

4.4. СЕТЕВЫЕ КОМАНДЫ.

4.4.1. Инициализация ученических компьютеров.

Команда I (Init)

Формат I <BK>

Эта команда описана в разделе 3. Она инициализирует те ученические компьютеры, которые работают в Basic'e с включенной сетевой программой фирмы YAMANA, т.е. те, на которые после включения не пересылалась другая сетевая система. После ее выполнения осуществляется очистка буфера системы!

4.4.2. Пересылка текста программы на компьютер ученика.

Команда T (Transmit).

Формат T,<n1>,n2 <BK>

Эта команда перешлет программу из буфера ученику с номером n1 или, если задано n2, то ученикам с номера n1 по номер n2. При пересылке одному ученику, если пересылка не удалась, система сообщит:

Error!

Это означает, что компьютер ученика не подключен к сети. Результат пересылки группе учеников не диагностируется.

Пример:

IT6

(переслать программу из буфера ученику с номером 6.)

IT1,3

(переслать программу из буфера ученикам 1,2,3.)

4.4.3. Прием текста программы с ученического компьютера.

Команда V (receive).

Формат V<n1> <BK>

С помощью этой команды можно загрузить в буфер учительского компьютера программу ученика с номером n1.

Если пересылка не удалась, то система сообщит:

Error!

Это означает, что компьютер ученика не подключен к сети.

Пример:

IV7

(Прием программы ученика 7.)

4.4.4. Останов программы ученика.

Команда @ (break student).

Формат @<n1> <BK>

С помощью этой команды можно прервать выполнение программы учеником n1. На компьютере n1 будет установлен режим SCREEN 0. Если это не удалось, то система сообщит:

Error!

Это означает, что компьютер ученика не подключен к сети.

Пример:

!@7

4.4.5. Включение/выключение опроса ученических компьютеров.

Команды * (polling on) и <ESC> (polling off).

Формат * <BK>

Эта команда включает автоматический опрос учеников. Только в этом режиме ученик может дать команду своему модулю системы или обратиться к диску при помощи сетевого пакета.

Выйти из режима можно нажав на <ESC>.

Во время опроса на экран выводится сообщение

student <n>

причем n меняется. Если ученик обратился с запросом, то на экране учителя переводится строка и возникают сообщения о реакции системы, соответствующей запрашиваемым функциям.

4.4.6. Просмотр и изменение статусов учеников.

Команда #.

Формат #n1,n2,string

Эта команда предназначена для установки номеров опрашиваемых учеников. Если дать команду в формате

#n1,n2,on

то компьютеры с номера n1 по номер n2 будут опрашиваться после команды *.

Команда

#n1,n2,off

отключает учеников с номера n1 по номер n2 от опроса.

По команде

#

без параметров на экран выдается таблица подключения компьютеров к опросу.

Изначально все ученики с 1 по 15 опрашиваются.

Пример:

]#1,3,off

(не опрашивать учеников 1,2,3.)

4.5. ВЫХОД ИЗ СИСТЕМЫ.

Команда В (Bye).

Формат В <ВК>

По этой команде система прекращает работу и выходит в DOS. Обратите внимание, что при нажатии <CTRL> <STOP> или <CTRL> С осуществляется перезагрузка системы без выхода в DOS.

КОМАНДЫ УЧЕНИКА.

Формат команд ученика такой же, как и у учителя. Выход в режим ввода команд системы на ученическом компьютере возможен только в том случае, если учитель включил полинг, т.е. дал команду `<*>`. При этом ученик, нажав на клавишу `<ESC>` при удержании клавиши `<STOP>` увидит приглашение:

]

После этого он может вводить команду.

После ввода команды и нажатия на клавишу `<BK>` ученический компьютер возвращается в редактор HiSoft Pascal. О выполнении команды он может судить по последующей за этим очистке экрана и сообщениям (см. ниже).

Если команда была дана неверно, то после очистки экрана появится сообщение:

Error in net command!

5.1. ЗАГРУЗКА ТЕКСТА ПРОГРАММЫ С ДИСКА.

Команда G (Get).

Формат `Gn1,n2,string <BK>`

Эта команда полностью аналогична команде G учителя. Если она выполнена успешно, то экран ученика будет очищен, сообщений не появится. Ученик может работать с полученной программой.

Если файл не найден, то после очистки экрана ученика появится сообщение:

File not found!

5.2. СОХРАНЕНИЕ ТЕКСТА ПРОГРАММЫ НА ДИСКЕ.

Команда P (Put).

Формат `Pn1,n2,<string> <BK>`

Эта команда также полностью аналогична команде P учителя. При успешном выполнении после очистки экрана появится сообщение:

Ok!

Если была ошибка при открытии файла, то система сообщит:
File not found!

5.3. ПОДГРУЗКА ПРОГРАММЫ С ДИСКА.

Команда M (Merge).

Формат M<n1>,n2,<string> <BK>

Полный аналог команды M учителя. Сообщения текие же, как у G.

5.4. РАЗРЕШЕНИЕ ОПЦИИ КОМПИЛЯТОРА \$F.

Команда K (link).

Формат K <BK>

Аналог команды K учителя. После выполнения экран очистится и программа с разрешенными директивами \$F готова к работе.

5.5. СОХРАНЕНИЕ ОБЪЕКТНОГО КОДА ПОСЛЕ ТРАНСЛЯЦИИ.

Команда S (Save object).

Формат S,.,<string> <BK>

Сохраняет результат трансляции программы ученика на диске учителя под именем <string>.

Для правильного выполнения этой команды необходимо строго соблюсти следующую последовательность действий:

- с помощью команды T HiSoft Pascal оттранслировать программу на ученическом компьютере;
- на вопрос HiSoft Pascal: Ok? ответить нажатием любой клавиши, кроме <Y>;
- выйти в систему (<STOP><ESC>) и дать команду S в описанном выше формате.

Если команда выполнена правильно, то через 10-30 секунд экран очистится и появится сообщение Ok!

В разделах 6 и 8 настоящего руководства описаны способы запуска программы, оттранслированной на ученическом компьютере.

Пример:

Команда HiSoft Pascal:

T

.....
.... (листинг трансляции)
.....

Ok?N

<STOP><ESC>

]S, ,exm.obj

(Пример трансляции и сохранения кода в файле exm.obj.)

6

ИСПОЛНЯЮЩАЯ СИСТЕМА.

Исполняющая система предназначена для запуска на ученических компьютерах программ, транслировавшихся с помощью магнитофонной (ученической) версии HiSoft Pascal и сохраненных командой S (см. 5.5). Исполняющая система поддерживает функции сетевого пакета STNET.PAS.

Для запуска исполняющей системы необходимо на ученических компьютерах нажать кнопку RESET, а на учительском дать команду:

A>EXEC

Система загрузится через 15-20 секунд.

Команды системы даются нажатием на соответствующие клавиши. Далее запрашивается дополнительная информация.

6.1. ПЕРЕСЫЛКА И ЗАПУСК ПРОГРАММЫ НА КОМПЬЮТЕРЕ УЧЕНИКА.

Команда T.

С помощью этой команды можно переслать и запустить программу в объектном коде, сохраненном командой S ученической системы, на одном или всех компьютерах.

После нажатия на клавишу <T> система спросит:

name>

Необходимо ввести имя файла с объектным кодом и, далее, на

запрос:

```
student>
```

ввести номер ученика (1-15) или, если программу нужно переслать всем, то число, большее 15.

Пример:

T

```
name>exm.obj
```

```
student>7
```

(Пересылка ученику 7 программы exm.obj.)

6.2. ОСТАНОВ ПРОГРАММЫ УЧЕНИКА.

Команда @.

С помощью этой команды можно прервать выполнение программы одним или всеми учениками. Это необходимо сделать, например, перед тем, как посылать новую программу (!В отличие от сетевой Паскаль-системы!).

После нажатия на клавишу <@> система запросит:

```
student>
```

Необходимо ввести номер ученика от 1 до 15. Если введено число большее 15, то прервется выполнение программы у всех учеников.

6.3. ВЫХОД ИЗ СИСТЕМЫ.

Команда Q.

Нажав на клавишу Q, Вы выйдете из исполняющей системы в DOS.

7

ПАКЕТ СЕТЕВЫХ ФУНКЦИЙ УЧЕНИКА.

Пакет сетевых функций ученика является инструментальным средством, позволяющим создавать программы, которые с ученического компьютера обращаются к дискам на учительской машине и пересылают область памяти с одного компьютера на другой. Программы, созданные на базе этого пакета можно ис-

полнить только на ученической машине под управлением сетевого Паскаль-монитора или исполняющей системы на компьютере учителя.

Для использования этого пакета достаточно включить в текст программы в разделе описания процедур и функций строку

```
{ $F STNET.PAS }
```

Общая схема обращения к функциям пакета такова:

- Если LNFILE, RDFILE, WRFILE или POST (см. 7.2-7.5) возвращают FALSE, то учитель не разрешает сетевой обмен (posting off).
- В противном случае на компьютере ученика устанавливается флаг запроса на связь. Сеанс связи (транзакция) закончен в том и только в том случае, если функция IORESULT (см. 7.1) приняла ненулевое значение.

7.1. ФУНКЦИЯ IORESULT.

Результат выполнения сетевой функции.

Имя: FUNCTION IORESULT: INTEGER;

Действие: возвращает результат последней из вызванных сетевых функций. После обращения к функции флаг результата сбрасывается и последующие обращения дадут 0 до окончания текущего сеанса связи.

Выход: 0 - сетевой обмен не завершен или нет запроса.

- 1 - обмен прошел успешно.
- 2 - файл не найден.
- 3 - нельзя создать новый файл.
- 4 - ошибка ввода/вывода с диском.
- 5 - ошибка почты.

7.2. ФУНКЦИЯ LNFILE.

Определение длины файла на диске.

Предназначена для определения размера файла.

Имя: FUNCTION LNFILE (DEST, FNADDR: INTEGER): BOOLEAN;

Параметры: DEST - адрес целой переменной, в которую будет положена длина файла.

FNADDR - адрес символьной строки, задающей имя файла (без пробелов). Строка должна заканчиваться символом с кодом 0 (CHR(0)).

Действие: устанавливает запрос к системе на нахождение размера файла с именем по адресу FNADDR. Система положит 2 байта длины по адресу DEST.

Выход: TRUE, если запрос установлен.

FALSE, если учитель не опрашивает учеников.

7.2. ФУНКЦИЯ RDFILE.

Прямое чтение файла на диске.

Эта функция предназначена для прямого чтения из файла на диске учителя.

Имя: FUNCTION RDFILE (SOURCE,DEST,LENGTH,FNADDR:INTEGER): BOOLEAN;

Параметры: SOURCE- номер байта в файле, с которого начинается чтение;

DEST - адрес буфера ученика, в который будет переслано содержимое файла;

LENGTH - сколько байт файла, начиная с DEST, необходимо прочесть и переслать ученику.

FNADDR - адрес символьной строки, задающей имя файла (без пробелов). Строка должна заканчиваться символом с кодом 0 (CHR(0)).

Действие: устанавливает запрос на чтение отрезка файла начиная с байта SOURCE длиной LENGTH и пересылку ученику в буфер с адреса DEST.

Выход: TRUE, если запрос установлен.

FALSE, если учитель не опрашивает учеников.

7.3. ФУНКЦИЯ WRFILE.

Прямая запись в файл на диске.

Эта функция предназначена для прямой записи в файл на диске учителя.

Имя: FUNCTION WRFILE (SOURCE,DEST,LENGTH,FNADDR:INTEGER): BOOLEAN;

Параметры: SOURCE- адрес буфера с записываемыми данными у

ученика;

DEST - номер байта в файле, с которого будет записано содержимое буфера;

LENGTH - количество записываемых байтов.

FNADDR - адрес символьной строки, задающей имя файла (без пробелов). Строка должна заканчиваться символом с кодом 0 (CHR(0)).

Действие: устанавливает запрос на запись буфера ученика с адреса SOURCE длиной LENGTH на диск учителя в файл с именем, заданным адресом FNADDR.

Выход: TRUE, если запрос установлен,

FALSE, если учитель не опрашивает учеников.

7.4. ФУНКЦИЯ POST.

Пересылка области памяти на другой компьютер.

Эта функция предназначена для пересылки области памяти от одного ученика к другому.

Имя: FUNCTION POST (SOURCE, DEST, LENGTH, STUDENT: INTEGER): BOOLEAN;

Параметры: SOURCE- адрес буфера с пересылаемыми данными;

DEST - адрес принимающего буфера;

LENGTH - количество пересылаемых байтов.

STUDENT - номер адресата;

Действие: устанавливает запрос на пересылку из буфера с адреса SOURCE LENGTH байтов ученику с номером STUDENT по адресу DEST.

Выход: TRUE, если запрос установлен.

FALSE, если учитель не опрашивает учеников.

Внимание!!! Эта функция может испортить программу принимающего ученика, обращайтесь с ней осторожно!

8

КОНВЕРТОР ДЛЯ ЗАГРУЗКИ ОБЪЕКТНЫХ МОДУЛЕЙ С ДИСКА.

Предназначен для создания файлов, которые можно загружать с диска из BASIC-системы командой

```
bload "<name>" &
```

из файлов, полученных при помощи команды S учебного модуля системы.

Для вызова достаточно набрать в DOS

```
A>conv <имя исходного файла> <имя результирующего файла>
```

Напомним, что программы, содержащие обращения к сетевым функциям, можно выполнять только под сетевой Паскаль-системой или исполняющей системой.

Программы, созданные конвертором, можно запускать с помощью сетевой Basic-системы.

ГЛАВА СИСТЕМНОГО ПРОГРАММИСТА.

9.1. МОНИТОР УЧИТЕЛЯ.

При работе с монитором учителя необходимо иметь в виду следующее:

Монитор после загрузки меняет имя файла, содержащего интерпретатор командной строки (обычно: COMMAND.COM) на PASSYS.COM. Это обеспечивает загрузку монитора после выполнения команд Run и Compile, а также после нажатия ^C. Поэтому имя PASSYS.COM нельзя изменять. Также не рекомендуется пользоваться операционными системами, отличными от оригинального MSX DOS, не убедившись прежде, что вышеуказанное свойство учительского монитора они поддерживают. В противном случае разработчики не гарантируют работу системы в соответствии с документацией.

9.2. МОНИТОР УЧЕНИКА.

Монитор ученика занимает участок памяти с адреса 9C00H, а HiSoft-Pascal - с 0A700H. При работе в редакторе память имеет слотовую конфигурацию Бейсик-системы, т.е. с ROM BASIC и ROM BIOS. Программа расположена в RAM с 80H (адреса ее начала и конца можно узнать с помощью команды X HiSoft Pascal), также, как и объектный код, генерируемый компилятором. При исполнении кода, таким образом, ROM не подключен, и процедуры BIOS и интерпретатора необходимо вызывать с помощью RST 30H или переключая слоты "вручную".

Следует иметь в виду, что HiSoft Pascal не поддерживает в конфигурации 64K RAM процедур RDSLT, WRSLT, ENASLT, CALSLT и CALLF (RST 30H). Монитор ученика при начальной загрузке обеспечивает только точку входа CALLF и только для ROM BIOS и SUBROM. Если Вам необходимо вызывать процедуры из другого ПЗУ, то модифицируйте процедуру с точкой входа 30H.

При задании стартового адреса трансляции (Alter) необходимо иметь в виду, что процедуры времени выполнения HiSoft лежат с адреса 0D700H, а процедуры времени выполнения для сети и интерслотового вызова - с адреса 9D00H. Эти области, а также область с 0 до 80H объектный код занимать не должен.

Монитор ученика использует хуки HKEYI, HCHGE и HCHPU. Хук HKEYI вызывает процедуры реального времени и модифицировать его не рекомендуется.

9.3. ИСПОЛНЯЮЩАЯ СИСТЕМА.

Ученическая часть исполняющей системы занимает менее 2K памяти с адреса 9C00H, а процедуры времени выполнения HiSoft-память с 0D700H до области системных переменных. Таким образом, область памяти с 0A400H до 0D700H может использоваться для хранения объектных кодов пользователя, как и область 80H-09C00H. Это дает возможность выполнять под управлением исполняющей системы программы, состоящие из двух и более модулей и значительно превосходящие по величине те, которые выполняются под Паскаль-монитором.

9.4. ИНСТРУМЕНТАЛЬНЫЕ СЕТЕВЫЕ ПРОЦЕДУРЫ.

Инструментальные сетевые процедуры поддерживаются как Паскаль-системой, так и исполняющей системой.

Процедуры используют следующие рабочие переменные:

POLFLG	9C0AH	- флаг поллинга (сканирование учеников учителям на предмет запроса): 1, если поллинг включен; 0 - если выключен, т.е. связь по инициативе ученика запрещена.
IORES	9C09H	- результат транзакции (однократный сеанс связи, см. выше).

9.5. КОНВЕРТОР ОБЪЕКТНЫХ МОДУЛЕЙ.

Конвертор присоединяет к объектному коду процедуры времени выполнения HiSoft Pascal, а также процедуру, которая перемещает объектный код на его место в памяти, обеспечивает точку входа 30H и иницирует выполнение. Программа, полученная после конвертирования, загружается с адреса 8000H. Конвертор нельзя применять к программам, использующим инструментальные сетевые процедуры.

III
ИНСТРУМЕНТАЛЬНЫЕ
ПРОГРАММНЫЕ
СРЕДСТВА

ДЛЯ КОМПИЛЯТОРА
HiSoft PASCAL

ВВЕДЕНИЕ.

Предлагаемые программные средства рассчитаны на широкий круг пользователей - владельцев компьютерных классов MSX-2, в первую очередь - на прикладных программистов, занятых написанием обучающих программ. Предлагаемые средства, расширяя интерактивные возможности языка Паскаль, позволяют использовать их в качестве учебного средства при обучении основам программирования на этом языке.

Графические процедуры, содержащиеся в базовом графическом пакете выполняют функции, аналогичные графическим операторам Бейсика, что позволяет без потерь осуществлять переход от Бейсика к Паскалю при обучении программированию, а также использовать HiSoft-Паскаль в качестве основного средства при обучении программированию "от нуля".

В качестве учебного материала также могут использоваться процедуры других пакетов - "Клавиатура", "Динамическая графика" (спрайты), "LOGO-PAS".

Процедуры, содержащиеся в этих и других пакетах, существенно превосходят по своим возможностям графические средства Бейсика. Это, а также эффективность объектного кода, создаваемого компилятором Паскаля, позволяет создавать высокопрофессиональные обучающие и игровые программы.

Процедуры пакетов написаны на Паскале, с включением отдельных участков, написанных на Ассемблере и преобразованных в машинные коды (такие участки обозначены операторами INLINE).

Каждый пакет представлен на дискете в виде текстового файла, содержащего исходные тексты всех входящих в него процедур. Исключение составляет пакет "Видеопроцессор", представлен-

ный двумя файлами, один из которых содержит функцию SETMEM, а другой - все остальные процедуры и функции (подробно см. главу 6).

Для подключения процедур пакетов пользователю достаточно в тексте своей программы сразу после блоков описания переменных, констант, типов и пр. вставить предложение

{ \$F имя-файла }

Процедуры пакета "Форматный вывод на графический экран" подключаются особым образом, подробно описанным в главе 5.

Чтобы сэкономить на размере текста и объектного кода, пользователю рекомендуется не подключать пакеты целиком, а формировать для каждой конкретной программы собственные пакеты, содержащие только те процедуры, которые используются в программе. При этом пользователю надлежит проявить осторожность при работе с пакетами "Клавиатура" и "LOGO-PAS", в которых встречаются процедуры, ссылающиеся на другие процедуры внутри тех же пакетов.

В состав передаваемых программных средств входят следующие пакеты:

"Базовый графический пакет"	(файл SBG2.PAS),
"Линии"	(файл SCG2.PAS),
"Закраски"	(файл SFG2.PAS),
"Динамическая графика"	(файл SPR2.PAS),
"Форматный вывод на графический экран"	(файл GRP2.PAS),
"Видеопроцессор"	(файл VDP2.PAS),
"Клавиатура"	(файл KEY2.PAS),
"LOGO-PAS"	(файл LOGO .PAS).

ОСНОВНЫЕ ГРАФИЧЕСКИЕ ПРОЦЕДУРЫ

1.1. НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ ПАКЕТА

Основные графические процедуры (базовый графический пакет) реализуют стандартные графические возможности, имеющиеся в системе MSX-2. Эти процедуры во многом аналогичны соответствующим графическим операторам языка Basic.

Предлагаемый пакет является расширяемым. Например, использование процедур установки режимов рисования (см. раздел 2) позволяет существенно увеличить возможности таких процедур базового графического пакета как PSET, LINE, BOX, CIRCLE, ELLIPS, ARC. При этом пользователю не надо вносить никаких изменений в использующую их программу, достаточно только вызвать соответствующую процедуру установки.

Отличия процедур базового графического пакета от соответствующих графических операторов интерпретатора языка Basic заключаются в следующем:

- процедура SCREEN, в отличие от ее аналога в языке Basic, инициализирует все таблицы видеопроцессора
- в случае, если необходимо выполнить выборочную инициализацию таблиц видеопроцессора, может быть использована процедура CLS; тем самым возможности этой процедуры несколько шире возможностей аналогичного оператора языка Basic
- в процедурах ELLIPS и ARC, в отличие от соответствующего им оператора языка Basic (CIRCLE), указываются горизонтальная и вертикальная полуоси эллипса, что, по нашему мнению, более удобно
- в процедуре ARC углы задаются в градусах, что позволяет избежать использования в программе функций плавающей арифметики.

Все процедуры пакета объединены в одном файле - SBG2.PAS. Поэтому, чтобы включить их в свою программу пользователю достаточно воспользоваться оператором \$F ({ \$F SBG2.PAS }).

1.2. ОПИСАНИЕ ПРОЦЕДУР ПАКЕТА

В состав пакета входят следующие процедуры и функции:

- LOCATE - установка курсора;
- SETPAGE - установка активной и отображаемой страниц;
- LOGOP - установка логической операции;
- VPEEK - чтение байта из видеопамати;
- VPOKE - запись байта в видеопамать;
- COLOR - установка фасадного, фонового и бордюрного цветов;
- SCREEN - установка графического режима;
- CLS - выборочная инициализация таблиц видеопроцессора;
- POINTC - определение цвета указанной точки экрана;
- PSET - рисование точки;
- LINE - рисование отрезка прямой;
- BOX - рисование прямоугольника (закрашенного и незакрашенного);
- CIRCLE - рисование окружности;
- ELLIPS - рисование эллипса;
- ARC - рисование дуги;
- PAINT - закрашивание замкнутой области.

Ниже приводится краткое описание назначения, входных и выходных параметров процедур и функций пакета.

1.2.1. Процедура LOCATE

Процедура LOCATE позиционирует курсор в текстовом экранном режиме (аналогична соответствующему оператору языка Basic).

Заголовок : PROCEDURE LOCATE(x,y:INTEGER;disp:BOOLEAN);
Параметры : x,y - координаты курсора
 : disp = TRUE - курсор отображается
 : = FALSE - курсор не отображается
Результат : -

1.2.2. Процедура SETPAGE

Процедура SETPAGE устанавливает номера отображаемой и активной страниц видеопамати (аналогична соответствующему оператору языка Basic). Процедура работает только в старших экранных режимах (начиная с пятого). При этом в экранных режимах 5 и 6 номера страниц берутся по модулю 4, а в экранных режимах 7 и 8 - по модулю 2.

Заголовок : PROCEDURE SETPAGE(disp,active:INTEGER);
Параметры : disp - номер отображаемой страницы
 : active - номер активной страницы
Результат : -

1.2.3. Процедура LOGOP

Процедура LOGOP устанавливает код логической операции, которая должна выполняться над данными при их записи в видеопамать. Эта операция будет выполняться при вызове любой графической процедуры, осуществляющей вывод данных в видеопамать. Таблица кодов логических операций приведена в приложении А.

Заголовок : PROCEDURE LOGOP(code:INTEGER);
Параметры : code - код логической операции
Результат : -

1.2.4. Функция VPEEK

Функция VPEEK читает один байт данных из видеопамати.

Заголовок : FUNCTION VPEEK (source: INTEGER) : CHAR;

Параметры : source - адрес видеопамати

Результат : VPEEK - считанный байт данных

1.2.5. Процедура VPOKE

Процедура VPOKE записывает один байт данных в видеопамать.

Заголовок : PROCEDURE VPOKE (dest: INTEGER; byte: CHAR);

Параметры : dest - адрес видеопамати

: byte - записываемый байт данных

Результат : -

1.2.6. Процедура COLOR

Процедура COLOR выполняет установку фасадного, фонового и бордюрного цветов. Процедура аналогична соответствующему оператору языка Basic.

Заголовок : PROCEDURE COLOR (f, bak, bdr: INTEGER);

Параметры : f - цвет фасада

: bak - цвет фона

: bdr - цвет бордюра

Результат : -

1.2.7. Процедура SCREEN

Процедура SCREEN осуществляет установку требуемого экранного режима. При этом, в отличие от ее аналога в языке Basic, инициализируются все таблицы видеопроцессора (имен, шаблонов, цветов, атрибутов спрайтов, шаблонов спрайтов).

Заголовок : PROCEDURE SCREEN(n:INTEGER);

Параметры : n - номер экранного режима

Результат : -

1.2.8. Процедура CLS

Процедура CLS в зависимости от значений параметров инициализирует различные таблицы видеопроцессора.

Заголовок : PROCEDURE CLS(grp,spr,pat:BOOLEAN);

Параметры : grp =TRUE-инициализируются таблицы имен,

: шаблонов, цветов

: spr =TRUE-инициализируется таблица

: атрибутов спрайтов

: pat =TRUE-инициализируется таблица

: шаблонов спрайтов

Результат : -

1.2.9. Функция POINTC

Функция POINTC возвращает цвет точки экрана с указанными координатами. Если точка находится вне экрана, то возвращается 255.

Заголовок : FUNCTION POINTC(x,y:INTEGER):INTEGER;

Параметры : x, y — координаты точки

Результат : POINTC — цвет указанной точки

1.2.10. Процедура PSET

Процедура PSET устанавливает цвет указанной точки экран (выводит точку указанного цвета в указанное место). Если точка находится за пределами экрана, то она не выводится.

Заголовок : PROCEDURE PSET(x, y, c :INTEGER);

Параметры : x, y — координаты точки

: c — цвет точки

Результат : —

1.2.11. Процедура LINE

Процедура LINE выводит на экран отрезок прямой указанного цвета по координатам его начала и конца. Если отрезок не помещается целиком на экране, то выводится его видимая часть.

Заголовок : PROCEDURE LINE(x_1, y_1, x_2, y_2, c :INTEGER);

Параметры : x_1, y_1 — координаты начальной точки

: x_2, y_2 — координаты конечной точки

: c — цвет отрезка

Результат : —

1.2.12. Процедура BOX

Процедура BOX выводит на экран прямоугольник указанного цвета (закрашенный или незакрашенный) по координатам начала и конца его диагонали. Если прямоугольник не помещается целиком на экране, то выводится только его видимая часть.

1.2.15. Процедура ARC

Процедура ARC выводит на экран дугу эллипса указанного цвета по координатам его центра, горизонтальному и вертикальному радиусам, начальному и конечному углам (в градусах). Если дуга не помещается целиком на экране, то выводится ее видимая часть. Если один из углов отрицательный, дуга не выводится. Если начальный угол не превышает конечного, то дуга выводится от начального угла к конечному (против часовой стрелки), в противном случае - от конечного угла к-начальному (против часовой стрелки). Если угол превышает 360 градусов, то он берется равным 360.

Заголовок : PROCEDURE ARC (x,y,rx,ry,c,b,e:INTEGER) ;

Параметры : x,y - координаты центра дуги

: rx - горизонтальный радиус

: ry - вертикальный радиус

: c - цвет дуги

: b - начальный угол дуги

: e - конечный угол дуги

Результат : -

1.2.16. Процедура PAINT

Процедура PAINT закрашивает в указанный цвет область экрана, идентифицируемую координатами одной из ее внутренних точек. Границей области считается любая точка, имеющая указанный граничный цвет. Цвет закраски и граничный цвет могут при этом не совпадать.

Заголовок : PROCEDURE PAINT(x,y,c,cb:INTEGER);

Параметры : x,y - координаты внутренней точки области

: c - цвет закрашки

: cb - цвет границы области

Результат : -

2

ДОПОЛНИТЕЛЬНЫЕ ГРАФИЧЕСКИЕ ПРОЦЕДУРЫ

2.1. НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ ПАКЕТА

Дополнительные графические процедуры (графический пакет "Линии") существенно расширяют возможности рисования различных линий и кривых по сравнению с процедурами базового пакета. Их использование позволяет, в частности, устанавливать режим, при котором, с помощью уже имеющихся процедур (входящих в состав базового графического пакета), можно

- рисовать пунктирные линии
- рисовать штриховые линии
- рисовать штрих пунктирные линии
- рисовать линии различной толщины
- рисовать линии с помощью заданного пользователем "маркера"

Очень удобным является то, что в пакете "Линии" отсутствуют аналоги процедур PSET, LINE, BOX, CIRCLE, ELLIPS, ARC, наличие которых могло бы привести к путанице. Пользователю нужно только с помощью специальных процедур, имеющихся в нем, установить соответствующий режим и перечисленные выше процедуры базового графического пакета начнут работать иначе.

Кроме процедур установки (SCURVE, DASH, MRKON и MRKOFF) в пакет "Линии" включены дополнительные графические процедуры, позволяющие

- устанавливать и перемещать графический курсор (MAPXY, MOVE);
- выводить точку в текущую позицию графического курсора (RPSET);

- рисовать сектора и сегменты
(SECTOR, SEGMENT).

Все процедуры пакета объединены в одном файле - SCG2.PAS.
Поэтому, чтобы включить их в свою программу пользователи
достаточно воспользоваться оператором \$F ({ \$F SCG2.PAS }).

2.2. ОПИСАНИЕ ПРОЦЕДУР ПАКЕТА

В состав пакета входят следующие процедуры:

- SCURVE - установка режима рисования линий;
- DASH - установка типа линий и их толщины;
- MRKON - включение режима рисования с помощью маркера;
- MRKOFF - выключение режима рисования с помощью маркера;
- MAPXY - установка графического курсора;
- MOVE - перемещение графического курсора на один пиксел
в указанном направлении;
- RPSET - рисование точки в текущей позиции графического
курсора;
- SECTOR - рисование сектора эллипса;
- SEGMENT - рисование сегмента эллипса.

Ниже приводится краткое описание назначения, входных и
выходных параметров доступных пользователям процедур пакета.

2.2.1. Процедура SCURVE

Процедура SCURVE устанавливает режим рисования линий

Заголовок : PROCEDURE SCURVE(mode: INTEGER) ;

Параметры : mode = 0 - базовый режим

: <> 0 - расширенный режим

Результат : -

2.2.2. Процедура DASH

Процедура DASH устанавливает тип линии и ее толщину по осям X и Y (ширина и высота прямоугольника с помощью которого будет выполняться рисование). Действует до тех пор, пока не будет установлен новый тип линии или не будет выполнена процедура SCURVE. В последнем случае тип линии устанавливается равным 0, а толщина по осям X и Y - 1. Если указан тип, отличный от четырех перечисленных ниже, то остается текущий тип. Если указана толщина, меньшая единицы (в том числе и отрицательная), то берется толщина, равная единице. Если толщина линии хотя бы по одному из направлений равна единице, то и по другому направлению берется толщина, равная единице.

Заголовок : PROCEDURE DASH(n,wx,wy:INTEGER);

Параметры : n = 0 - сплошная линия
 : = 1 - пунктирная линия
 : = 2 - штриховая линия
 : = 3 - штрих-пунктирная линия
 : wx - толщина линии по оси X
 : wy - толщина линии по оси Y

Результат : -

2.2.3. Процедура MRKON

Процедура MRKON устанавливает маркер, с помощью которого будет выполняться рисование. Перед тем, как выполнить эту процедуру, пользователь должен задать в памяти соответствующий шаблон. Размер шаблона в байтах определяется исходя из текущей толщины по осям X и Y по формуле: $wx * wy * k$, где $k=1/2$ для экранных режимов 5 и 7, $=1/4$ для экранного режима 6 и $=1$ для экранного режима 8. Режим рисования с по-

мощью маркера включается только в высших экранных режимах (начиная с пятого). Если толщина хотя бы по одному из направлений равна единице, режим также не включается. Например, если мы хотим задать маркер, имеющий вид квадрата размером $8 * 8$ точек с чередующимися зелеными и красными вертикальными полосами, шаблон должен быть размером 32 байта и заполнен кодом `CHR(2*16+8)`. Затем, прежде чем выполнить процедуру `MRKON`, необходимо произвести установку типа и толщины линии (оператор `DASH(1,8,8);`).

Заголовок : `PROCEDURE MRKON(address:INTEGER);`

Параметры : `address` - адрес шаблона

Результат : -

2.2.4. Процедура `MRKOFF`

Процедура `MRKOFF` отменяет режим рисования с помощью маркера.

Заголовок : `PROCEDURE MRKOFF;`

Параметры : -

Результат : -

2.2.5. Процедура `MAPXY`

Процедура `MAPXY` выполняет установку графического курсора по указанным координатам (вычисляет координаты графического курсора `CLOC` и `CMASK`). В случае, если координаты указывают на точку за пределами экрана, в качестве последних берутся остатки от деления на `SIZEX` (ширина экрана в текущем экранном режиме) и `SIZEY` (высота экрана в текущем экранном режиме) соответственно (вычисляются "локальные" координаты), а частные от деления помещаются соответственно в системные переменные `SCRACX` и `SCRACY` (см. описание системных переменных в приложении С). Так, например, если мы находимся в седьмом

экранном режиме, $X=-100$, а $Y=250$, то в качестве "локальных" координат будут взяты числа 412 и 38, а в системные переменные SCRACX и SCRACY будут помещены соответственно числа -1 и 1. В частности, если указанные при вызове процедуры MAPXY координаты находятся в пределах экрана, то они будут взяты в качестве "локальных" координат, а в системные переменные SCRACX и SCRACY будут помещены нули. После выполнения описанных выше действий процедура MAPXY вычисляет по значениям "локальных" координат (а они всегда будут указывать на точку в пределах экрана) координаты графического курсора и помещает их в системные переменные CLOC и CMASK.

Заголовок : PROCEDURE MAPXY(x,y:INTEGER);

Параметры : x,y - координаты

Результат : -

2.2.6. Процедура MOVE

Процедура MOVE перемещает графический курсор на один пиксел в указанном направлении. При переходе через левую или правую границу экрана "локальная" X-координата устанавливается равной SIZEX-1 (где SIZEX - ширина экрана в данном экранном режиме) или 0, а значение системной переменной SCRACX соответственно уменьшается или увеличивается на единицу. При переходе через верхнюю или нижнюю границу экрана "локальная" Y-координата устанавливается равной SIZEY-1 (где SIZEY - высота экрана в текущем экранном режиме) или 0, а значение системной переменной SCRACY соответственно уменьшается или увеличивается на единицу. Таким образом возможно перемещение графического курсора даже если он находится за пределами экрана (значение по крайней мере одной из системных переменных SCRACX или SCRACY отлично от нуля).

Заголовок : PROCEDURE MOVE(dir:INTEGER);

Параметры : dir = 1 - один пиксел вверх;

: = 2 - один пиксел вверх и один пиксел
 вправо;

: = 3 - один пиксел вправо;

: = 4 - один пиксел вправо и один пиксел
 вниз;

: = 5 - один пиксел вниз;

: = 6 - один пиксел вниз и один пиксел
 влево;

: = 7 - один пиксел влево;

: = 8 - один пиксел влево и один пиксел
 вверх.

Результат : -

2.2.7. Процедура RPSET

Процедура RPSET выводит точку заданного цвета в текущую позицию графического курсора. Точка выводится на экран если выполнены все перечисленные ниже условия:

- значение системной переменной DISPSW равно нулю
- вывод точки не "замаскирован" переключателем типа линии (DASHSW);
- значения системных переменных SCRACX и SCRACY равны нулю (графический курсор располагается в пределах экрана).

Заголовок : PROCEDURE RPSET(c:INTEGER);

Параметры : c - цвет точки

Результат : -

2.2.8. Процедура SECTOR

Процедура SECTOR выводит на экран сектор эллипса указанного цвета по координатам его центра, горизонтальному и вертикальному радиусам, начальному и конечному углам (в градусах). Если сектор не помещается целиком на экране, выводится его видимая часть. Если один из углов отрицательный, сектор не выводится. Если начальный угол не превышает конечного, сектор выводится от начального угла к конечному (против часовой стрелки), в противном случае - от конечного угла к начальному (против часовой стрелки). Если угол превышает 360 градусов, он берется равным 360.

Заголовок : PROCEDURE SECTOR(x,y,rx,ry,c,b,e:INTEGER);

Параметры : x,y - координаты центра сектора

 : rx - горизонтальный радиус

 : ry - вертикальный радиус

 : c - цвет сектора

 : b - начальный угол сектора

 : e - конечный угол сектора

Результат : -

2.2.9. Процедура SEGMENT

Процедура SEGMENT выводит на экран сегмент эллипса указанного цвета по координатам его центра, горизонтальному и вертикальному радиусам, начальному и конечному углам (в градусах). Если сегмент не помещается целиком на экране, выводится его видимая часть. Если один из углов отрицательный, сегмент не выводится. Если начальный угол не превышает конечного, сегмент выводится от начального угла к конечному (против часовой стрелки), в противном случае - от конечного

угла к начальному (против часовой стрелки). Если угол превышает 360 градусов, он берется равным 360.

Заголовок : PROCEDURE SEGMENT(x,y,rx,ry,c,b,e:INTEGER);

Параметры : x,y - координаты центра сегмента

 : rx - горизонтальный радиус

 : ry - вертикальный радиус

 : c - цвет сегмента

 : b - начальный угол сегмента

 : e - конечный угол сегмента

Результат : -

3

ЗАКРАШИВАНИЕ ЗАМКНУТЫХ ОБЛАСТЕЙ

3.1. НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ ПАКЕТА.

В пакет включены процедуры, предоставляющие в распоряжение пользователя возможность закрашивать замкнутые области не только заданным цветом, как это делается процедурой PAINT (см. Раздел 1 "Основные графические процедуры"), но и различными шаблонами.

Формирование шаблона выполняется с помощью специальной процедуры SNABLON. Для задания шаблона закрашки необходимо сформировать его в памяти, после чего вызвать процедуру SNABLON и указать в ней его адрес. Размер шаблона фиксирован 8 * 8 точек. Размер выделяемой под шаблон области памяти равен 32 байтам для экранных режимов 5 и 7, 16 байтам - для экранного режима 6 и 64 байтам - для экранного режима 8.

Для того, чтобы теперь выполнить закрашку некоторой замкнутой области заданным шаблоном, необходимо вызвать процедуру SPRAY, указав в ней координаты идентифицирующей эту область точки и цвет границы. Закрашку заданным шаблоном можно выполнять только в старших экранных режимах (5, 6, 7, 8).

Внимание!! Не рекомендуется включать в шаблон закрашки точки, цвет которых совпадает с цветом границы области. В этом случае закрашка будет выполняться очень медленно.

Все процедуры пакета объединены в одном файле- SFG2.PAS. Поэтому, чтобы включить их в свою программу пользователю достаточно воспользоваться оператором \$F ({ \$F SFG2.PAS }).

3.2. ОПИСАНИЕ ПРОЦЕДУР ПАКЕТА.

В состав пакета входят следующие процедуры:

- SHABLON - задание шаблона закрашки;
- SPRAY - закрашка замкнутой области указанным шаблоном;

Ниже приводится краткое описание назначения, входных и выходных параметров процедур пакета.

3.2.1. Процедуры SHABLON

Процедура SHABLON задает шаблон, которым должны выполняться закрашки. Установленный шаблон закрашки будет оставаться неизменным до следующего вызова процедуры SHABLON. Например, чтобы, находясь в седьмом экранном режиме, установить шаблон закрашки, имеющий вид квадрата размером 8*8 точек, состоящего из чередующихся горизонтальных линий белого (15) и синего (4) цветов, необходимо зарезервировать в памяти 32 байта и занести туда следующие данные: CHR(255), CHR(68), ..., CHR(255), CHR(68). После чего вызвать процедуру SHABLON, указав в ней адрес этого массива данных.

Заголовок : PROCEDURE SHABLON(address: INTEGER);

Параметры : address - адрес массива данных, содержащего задаваемый шаблон

Результат: -

3.2.2. Процедура SPRAY

Процедура SPRAY выполняет закраску указанной области текущим шаблоном.

Заголовок : PROCEDURE SPRAY(x,y,cb: INTEGER) ;

Параметры : x,y - координаты точки, идентифицирующей
: , закрашиваемую область

Результат: -

ДИНАМИЧЕСКАЯ ГРАФИКА

1. НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ ПАКЕТА

В пакет процедур "Динамическая графика" объединены процедуры, обеспечивающие работу со спрайтами.

Включенные в него процедуры позволяют:

- формировать шаблоны спрайтов;
- устанавливать цвета спрайтов;
- выводить спрайты по указанным координатам;
- фиксировать ситуации столкновения спрайтов.

Пакет допускает работу в двух режимах: базовом и расширенном.

Возможности, имеющиеся в распоряжении пользователя в базовом режиме аналогичны предоставляемым интерпретатором языка Basic в компьютерах серии MSX-1.

Расширенный режим в отличие от базового дает пользователю некоторые дополнительные возможности работы со спрайтами. Основные отличия расширенного режима от базового заключаются в следующем:

- в расширенном режиме каждая строка спрайта может иметь свой цвет
- в расширенном режиме можно одновременно выводить до восьми спрайтов в строке
- в расширенном режиме можно задавать выборочную реакцию на столкновения спрайтов

Более детальное описание возможностей работы со спрайтами в расширенном режиме пользователь может найти в Руководстве по языку Basic для компьютеров серии MSX-2.

При работе в младших экранных режимах (SCREEN 1,2,3) по умолчанию действует базовый режим, в старших экранных режимах (SCREEN 4,5,6,7,8) – расширенный.

Наряду с процедурами, предназначенными для формирования и вывода спрайтов, в пакет включены вспомогательные функции, облегчающие формирование шаблонов и установку цветов. Эти функции формируют в области системных переменных массив, состоящий из указанных в списке параметров чисел (точнее их младших байтов), и возвращают пользователю его адрес.

Следует отметить, что после формирования очередного массива, содержимое предыдущего теряется.

Все процедуры пакета объединены в одном файле – SPR2.PAS. Поэтому, чтобы включить их в свою программу пользователю достаточно воспользоваться оператором \$F ({ \$F SPR2.PAS }).

2. ОПИСАНИЕ ПРОЦЕДУР И ФУНКЦИЙ ПАКЕТА

В состав пакета входят следующие процедуры и функции:

- SET8 - формирование в области системных переменных массива из 8 однобайтовых чисел;
- SET16 - формирование в области системных переменных массива из 16 однобайтовых чисел;
- SET32 - формирование в области системных переменных массива из 32 однобайтовых чисел;
- SIZEPR - установка формата шаблонов и размера спрайтов
- SETPAT - формирование шаблона спрайта;
- SETSPC - формирование цвета спрайта;
- SETATR - формирование атрибутов спрайта;
- COLLISION - проверка на столкновение спрайтов.

Ниже приводится краткое описание назначения, входных и выходных параметров процедур и функций пакета.

2.1. Функция SET8

Функция SET8 формирует в области системных переменных массив из 8 однобайтовых чисел и возвращает пользователю его адрес. В массив заносятся младшие байты чисел, указанных в списке параметров.

```
Заголовок : FUNCTION SET8 (x0,x1,x2,x3,x4,x5,x6,x7 : INTEGER
:                               ) : INTEGER;
```

Параметры : $x_0..x_7$ – список чисел

Результат : SET8 - адрес сформированного массива

2.2. Функция SET16

Функция SET16 формирует в области системных переменных массив из 16 однобайтовых чисел и возвращает пользователю его адрес. В массив заносятся младшие байты чисел, указанных в списке параметров.

```
Заголовок : FUNCTION SET16(x0,x1,x2,x3,x4,x5,x6,x7,  
: x8,x9,x10,x11,x12,x13,x14,x15  
: :INTEGER):INTEGER;
```

Параметры : x0..x15 - список чисел

Результат : SET16 - адрес сформированного массива

2.3. Функция SET32

Функция SET32 формирует в области системных переменных массив из 32 однобайтовых чисел и возвращает пользователю его адрес. В массив заносятся младшие байты чисел, указанных в списке параметров.

Заголовок : FUNCTION SET32(x0,x1,x2,x3,x4,x5,x6,x7,
 : x8,x9,x10,x11,x12,x13,x14,x15,
 : x16,x17,x18,x19,x20,x21,x22,x23,
 : x24,x25,x26,x27,x28,x29,x30,x31
 : : INTEGER) : INTEGER;

Параметры : x0..x31 - список чисел

Результат : SET32 - адрес сформированного массива

2.4. Процедура SIZEPR

Процедура SIZEPR устанавливает размер шаблонов спрайтов и масштаб вывода спрайтов.

Заголовок : PROCEDURE SIZEPR(size,mag:INTEGER);

Параметры : size = 0 - размер шаблона=8*8

: <> 0 - размер шаблона=16*16

: mag = 0 - увеличения нет

: <> 0 - увеличение в 2 раза

Результат : -

2.5. Процедура SETPAT

Процедура SETPAT формирует шаблон спрайта с указанным номером. Если размер шаблонов 8*8, то номер шаблона берется по модулю 256, а массив байтов, задающих шаблон, должен состоять из 8 элементов. Если же размер шаблонов 16*16, то номер шаблона берется по модулю 64, а массив байтов, задающих шаб-

лон должен состоять из 32 элементов (сначала описывается левая верхняя четверть шаблона, далее левая нижняя четверть, правая верхняя четверть и, наконец, правая нижняя четверть).

Заголовок : PROCEDURE SETPAT (np, address: INTEGER);

Параметры : np - номер формируемого шаблона
 : address - адрес массива байтов задающих шаблон

Результат : -

2.6. Процедура SETSPC

Процедура SETSPC устанавливает цвета строк спрайта (в расширенном режиме) или цвет спрайта (в базовом режиме). Если пользователь находится в базовом режиме, то в качестве второго параметра функции берется цвет спрайта. Если же пользователь находится в расширенном режиме, то вторым параметром функции должен быть адрес массива из 8 или 16 элементов типа CHAR, содержащего цвета строк спрайта. В действительности цвета определяются четырьмя младшими битами каждого байта. Старшие биты используются для указания некоторой дополнительной информации (например, выборочная реакция на столкновения). Более подробное описание назначения этих битов смотрите в Руководстве по языку Basic для компьютеров серии MSX-2.

Заголовок : PROCEDURE SETSPC (ns, color: INTEGER);

Параметры : ns - номер спрайта
 : color - цвет или адрес массива цветов

Результат : -

2.7. Процедура SETATR

Процедура SETATR выводит спрайт с данным шаблоном в указанной плоскости в точку с заданными координатами. Номер плоскости всегда берется по модулю 32, номер шаблона - также, как в процедуре SETPAT.

Заголовок : PROCEDURE SETATR(ns,x,y,np:INTEGER);

Параметры : ns - номер плоскости (спрайта)

: x,y - координаты спрайта

: np - номер шаблона

Результат : -

2.8. Функция COLLISION

Функция COLLISION проверяет имелось ли с столкновение спрайтов с момента последнего прерывания. При использовании этой функции нельзя запрещать прерывания.

Заголовок : FUNCTION COLLISION:BOOLEAN;

Параметры : -

Результат : COLLISION = TRUE - столкновение было

: = FALSE- столкновения не было

ФОРМАТНЫЙ ВЫВОД НА ГРАФИЧЕСКИЙ ЭКРАН

5.1. НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ ПАКЕТА

Пакет "Форматный вывод на графический экран" позволяет выводить информацию на графический экран с помощью стандартных процедур WRITE и WRITELN. Возможны два варианта вывода. Первый аналогичен выводу в экранный файл "GRP" в Бейсике. При этом не поддерживаются скроллинг (движение текста на экране) и многие управляющие символы (BS, стрелки, HOME, CLS). Второй вариант - оконный вывод, только для режима SCREEN 7. Используя его, Вы можете создать одно или несколько окон для вывода текста. В этом варианте поддерживаются скроллинг в каждом окне и ряд управляющих символов.

Пакет, соответственно, состоит из двух независимых частей, и, перед началом работы, необходимо выбрать из него те процедуры, которые Вы собираетесь использовать.

К процедурам вывода первого типа относятся:

- GRPION (включение вывода первого типа);
- GRPOFF (выключение вывода на графический экран);
- GRPILOC (позиционирование "курсора").

Для организации вывода второго типа необходимо включить в раздел описания типов Вашей программы описание типа WINDOW, которое также находится в поставляемом пакете, и в разделе переменных описать столько переменных типа WINDOW, сколько окон Вы собираетесь использовать.

К процедурам вывода второго типа относятся:

- WINIT (создание окна);
- WKILL (уничтожение окна - закраска его цветом фона);
- GRP2ON (включение вывода второго типа в указанное окно);
- GRPOFF (выключение вывода второго типа - эта процеду-

- ра та же, что и для вывода первого типа!);
- WLOC (позиционирование курсора в окне);
- WCOLOR (изменение цветов вывода в окне).

Внимание! Перед окончанием программы или переходом в текстовый режим обязательно выполните процедуру GRPOFF.

Все процедуры пакета поставляются в одном файле -GRP2.PAS, однако там же содержится и описание типа WINDOW, поэтому использование директивы компилятора {\$F GRP2.PAS} недопустимо. Рекомендуем создать из GRP2.PAS три файла: два с процедурами различных типов и еще один - с декларацией TYPE WINDOW... . Каждый из этих текстов Вы уже сможете включить в соответствующий раздел Вашей Паскаль-программы, например, при помощи директивы \$F.

5.2. ОПИСАНИЕ ПРОЦЕДУР ПАКЕТА.

В состав пакета входят следующие процедуры и функции:

- GRP1ON - включение вывода первого типа
- GRP1LOC - позиционирование при выводе первого типа
- GRP0FF - выключение вывода на графический экран как первого, так и второго типа
- GRP2ON - включение вывода в окно (второго типа)
- WINIT - создание окна
- WKILL - уничтожение окна
- WLOC - позиционирование в окне
- WCOLOR - изменение цвета окна

Нижe приводится краткое описание назначения и входных параметров процедур пакета.

5.2.1. Процедура GRP1ON

Процедура GRP1ON включает режим вывода на графический экран без скроллинга.

После выполнения этой процедуры Вы можете выводить данные процедурами WRITE и WRITELN (как в текстовом режиме). Следует

помнить, что управляющие символы, кроме CR (BK), не поддерживаются, а CR осуществляет также перевод строки. При выходе из программы или переходе в текстовый режим отключите вывод процедурой GRPOFF.

Заголовок : PROCEDURE GRPION;

Параметры : -.

5.2.2. Процедура GRPOFF

Процедура GRPOFF выключает режим вывода на графический экран процедурами WRITE и WRITELN как первого, так и второго типов.

Заголовок : PROCEDURE GRPOFF;

Параметры : -.

5.2.3. Процедура GRPILOC

Процедура GRPILOC позиционирует "курсор" при выводе первого типа на графический экран, т.е. задает координаты левой верхней точки следующего выводимого символа (в пикселах).

Заголовок : PROCEDURE GRPILOC (X,Y:INTEGER);

Параметры : X - горизонтальная координата,

Y - вертикальная координата.

5.2.4. Процедура GRP2ON

Процедура GRP2ON включает вывод в окно на графическом экране SCREEN7, указанное в качестве параметра. Вывод осуществляется процедурами WRITE и WRITELN. Окно должно быть предварительно создано процедурой WINIT (см. ниже). С помощью этой процедуры можно задать вывод в другое окно, а затем вернуться в исходное - печать будет продолжена с того места, где была прервана. В окне осуществляется автоматический скроллинг. Поддерживаются управляющие символы: CR, LF, стрелки (UP,

DOWN,RIGHT,LEFT), BS,CLS,HOME. Символ DEL и ESC-последовательности не поддерживаются.

Перед завершением программы или переходом в текстовый режим необходимо отменить вывод в окно процедурой GRPOFF (той же, что и для вывода первого типа).

Заголовок : PROCEDURE GRP2ON (VAR W:WINDOW);

Параметр : VAR W:WINDOW - описание типа WINDOW приведено в пакете и должно быть включено в программу.

5.2.5. Процедура WINIT

Процедура WINIT инициализирует переменную типа WINDOW и рисует окно на экране SCREEN7 цветом его (окна) фона.

Координаты левого верхнего угла окна, ширину и высоту необходимо указывать в знакахместах.

Заголовок : PROCEDURE WINIT (VAR W:WINDOW;XL,YH,WIDTH, HEIGHT,FCOL,BCOL:INTEGER);

Параметры : W - окно,

0<=XL<=63 - горизонтальная координата левого верхнего угла окна;

0<=YH<=23 - вертикальная координата левого верхнего угла окна;

1<=WIDTH<=64 - ширина окна;

1<=HEIGHT<=24 - высота окна;

FCOL - цвет вывода в окно;

BCOL - фоновый цвет окна.

5.2.6. Процедура WKILL.

Процедура WKILL закрашивает окно фоновым цветом экрана.

Заголовок : PROCEDURE WKILL (VAR W:WINDOW);

Параметр : W - окно.

5.2.7. Процедура WCOLOR.

Процедура WCOLOR меняет текущий цвет вывода и фоновый цвет окна.

Заголовок : PROCEDURE WCOLOR (VAR W:WINDOW;FCOL,BCOL:
INTEGER);

Параметры : W - окно,
FCOL - цвет вывода (фасадный);
BCOL - фоновый цвет.

5.2.8. Процедура WLOC

Процедура WLOC позиционирует курсор в окне. Координаты курсора указываются в знакахестах, относительно левого верхнего угла. Например, для перемещения курсора в левый верхний угол окна WW можно вызвать WLOC(WW,0,0); , а в пятую колонку второй строки - WLOC(WW,5,2); , если считать колонки и строки с нуля. Если номер колонки (строки) больше или равен ширине (высоте) окна, то этот параметр игнорируется.

Заголовок : PROCEDURE WLOC(VAR W:WINDOW;COL,ROW:INTEGER);

Параметры : COL - номер колонки;
ROW - номер строки окна.

6 ПРОЦЕДУРЫ РАБОТЫ С ВИДЕОПРОЦЕССОРОМ

6.1. НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ ПАКЕТА

Пакет "Видеопроцессор" предоставляет в распоряжение пользователя гибкие и эффективные средства работы с видеопроцессором.

Включенные в пакет процедуры условно можно разбить на шесть групп:

- процедуры для работы с видеорегистрами;
- установочные процедуры;
- процедуры для работы с видеопамятью
(линейные блóки видеопамяти);
- процедуры для работы с видеопамятью
(прямоугольные блоки видеопамяти);
- процедуры для работы с графическими окнами;
- функция SETMEM.

К первой группе относятся функция VDPRD (чтение управляющего регистра видеопроцессора), процедура VDPWR (запись в управляющий регистр видеопроцессора) и процедура PALETTE (установка регистров палитры видеопроцессора).

Ко второй группе относятся процедуры установки режима отображения (DSPMODE) и установки режима мерцания (BLON, BLOFF). Эти процедуры

Внимание !! Процедуры BLON и BLOFF можно использовать только в текстовом режиме (SCREEN 0) с шириной строки, превышающей 40 (графический режим видеопроцессора TXT80).

К процедурам третьей группы относятся процедуры, предоставляющие в распоряжение пользователя средства обмена между памятью и видеопамятью, а также сравнения их содержимого. Кроме того, пользователь может выполнять как инкрементное, так и декрементное копирование одного участка видеопамати на другой. Наличие двух видов копирования (инкрементного и декрементного) позволяет корректно выполнять эту операцию даже, если соответствующие участки видеопамати пересекаются. И память и видеопамать трактуются при этом как линейные массивы информации. В эту группу входят процедуры VCOMP, VFILL, VGET, VPUT, VCOPY).

К четвертой группе относятся процедуры, работающие с прямоугольными блоками видеопамати. К ним относятся процедуры BLKFIL, BLKGET, BLKPUT, BLKCPY и BNDCPY. При работе с этими процедурами следует учитывать следующее. Координаты блоков и их размеры берутся по модулю 2 в экранных режимах 5 и 7 и по модулю 4 в экранном режиме 6.

Процедуры этой группы работают с видеопаматью как с единым целым (номер активной страницы во внимание не принимается). Используемые при этом координатные сетки (для разных экранных режимов они разные) приведены в Приложении А.

Отметим одну особенность работы процедур BLKGET и BLKPUT. Если код логической операции равен нулю (см. раздел 1, описание процедуры LOGOP), то чтение и запись в видеопамать выполняются побайтно. Если же код логической операции отличен от нуля, чтение и запись в видеопамать будут выполняться поточно (один байт на точку). В связи с этим пользователь должен при выделении в памяти места под содержимое блока учитывать какое копирование будет выполняться. В случае побайтного копирования размер выделяемой области памяти в байтах вычисляется по формуле $w * h * k$, где w - ширина блока в пикселах, h - высота блока в пикселах, а $k = 1/2$ в экранных режимах 5 и 7, $= 1/4$ в экранном режиме 6 и $= 1$ в восьмом экранном режиме. При поточечном копировании k всегда равно единице.

Внимание !! Процедуры этой группы можно использовать только в старших экранных режимах (5,6,7 и 8).

К пятой группе относятся процедуры, аналогичные процедурам третьей группы, но рассматривающие память и видеопамять как двумерные массивы (графические окна). При этом в состав графического окна включаются все области видеопамати, формирующие его образ на экране монитора. Так в экранных режимах 0, 1, 3, 5, 6, 7, и 8 (SCREEN0, SCREEN1, SCREEN3, SCREEN5, SCREEN6, SCREEN7, SCREEN8) - это соответствующие участки таблицы шаблонов, тогда как во втором и четвертом экранных режимах (SCREEN2, SCREEN4) - соответствующие участки таблицы шаблонов и таблицы цветов. Заметим, что в первом экранном режиме (SCREEN1) таблица цветов не участвует в формировании образа графического окна, так как она определяет не цвет соответствующих знакомест, а цвет символов с соответствующими кодами. Если пользователь желает изменить цвета выводимых символов, он должен сам занести необходимую информацию в таблицу цветов, воспользовавшись процедурами VFILL или VPUT настоящего пакета, или процедурой VPOKE базового графического пакета. В эту группу включены процедуры WFILL, WSAVE, WREST, WCOPY.

Процедуры пятой группы предоставляют в распоряжение пользователя удобные средства для работы с графическими окнами. При работе с этими процедурами в зависимости от того, в каком экранном режиме находится в данный момент система, автоматически определяются модифицируемые таблицы видеопроцессора, адреса и размеры соответствующих блоков в видеопамати.

Контроль на выход задаваемого пользователем графического окна за границы экрана при работе с процедурами пятой группы не выполняется.

При работе с графическими окнами в младших экранных режимах (SCREEN0, SCREEN1, SCREEN2, SCREEN3, SCREEN4) номер активной страницы должен быть равен нулю. В старших же экранных режимах (SCREEN5, SCREEN6, SCREEN7, SCREEN8) чтение и запись в видеопамать выполняются также, как и в процедурах, работающих

с прямоугольными блоками видеопамати (номер активной страницы во внимание не принимается, и в зависимости от значения кода логической операции обмен выполняется либо побайтно, либо поточно).

И, наконец, шестую группу представляет одна единственная функция или, если говорить точнее, "болванка" функции - SETMEM. Она частично восполняет отсутствие в стандарте языка Паскаль языковых средств, позволяющих устанавливать начальные значения переменных при компиляции программы (такие средства имеются у большинства языков программирования).

Эта функция предназначена для "резервирования" некоторого участка памяти и его "инициализации". Использование этой функции при "резервировании" и "инициализации" больших участков памяти в сравнении с традиционными методами (объявление массива соответствующей размерности и его "инициализация" с помощью оператора присваивания) дает существенный выигрыш как по памяти (экономится память, занимаемая программой), так и по времени ("инициализация" выполняется во время компиляции программы, а не во время ее выполнения).

Включение этой функции в данный пакет не случайно, так как именно при работе с видеопаматью наиболее часто возникает необходимость в выделении и инициализации больших участков памяти.

Все процедуры пакета за исключением последней объединены в одном файле - VDP2.PAS. Поэтому, чтобы включить их в свою программу пользователю достаточно воспользоваться оператором \$F ({\$F VDP2.PAS}).

6.2. ОПИСАНИЕ ПРОЦЕДУР И ФУНКЦИЙ ПАКЕТА

В состав пакета входят следующие процедуры и функции:

- VDPRD - чтение содержимого указанного управляющего регистра видеопроцессора;
- VDPWR - запись бита данных в указанный управляющий регистр видеопроцессора;

- PALETTE - установка указанного регистра палитры;
- DSPMODE - установка режима отображения;
- BLON - установка мерцания заданного "знакоместа" экрана;
- BLOFF - снятие мерцания заданного "знакоместа" экрана;
- DIRECT - установка направления копирования;
- VCOMP - сравнение блока данных в памяти с указанным блоком видеопамати;
- VFILL - заполнение участка видеопамати указанным байтом;
- VGET - чтение блока данных из видеопамати;
- VPUT - запись блока данных в видеопамать;
- VCPY - копирование одного участка видеопамати на другой;
- BLKFIL - заполнение прямоугольного блока видеопамати заданным байтом;
- BLKGET - чтение прямоугольного блока из видеопамати;
- BLKPUT - запись прямоугольного блока в видеопамать;
- BLKCPY - копирование прямоугольного блока видеопамати;
- BNDCPY - копирование в вертикальном направлении горизонтальной полосы видеопамати;
- WFILL - "инициализация" графического окна;
- WSAVE - сохранение содержимого графического окна в памяти;
- WREST - восстановление сохраненного в памяти графического окна;
- WCPY - копирование графического окна;
- SETMEM - выделение и инициализация памяти.

Ниже приводится краткое описание назначения, входных и выходных параметров процедур и функций пакета.

6.2.1. Функция VDPRD

Функция VDPRD читает содержимое управляющего видеорегистра с указанным номером.

Заголовок : FUNCTION VDPRD(reg:INTEGER):CHAR;

Параметры : reg - номер видеорегистра

Результат : VDPRD - содержимое видеорегистра

6.2.2. Процедура VDPWR

Процедура VDPWR записывает данные в указанный управляющий видеорегистр.

Заголовок : PROCEDURE VDPWR(reg:INTEGER;data:CHAR);

Параметры : reg - номер видеорегистра

: data - записываемые данные

Результат : -

6.2.3. Процедура PALETTE

Процедура PALETTE устанавливает указанный регистр (цвет) палитры видеопроцессора.

Заголовок : PROCEDURE PALETTE(c,r,b,g:INTEGER);

Параметры : c - номер регистра палитры (цвета)

: r - интенсивность красной составляющей

: b - интенсивность синей составляющей

: g - интенсивность зеленой составляющей

Результат : -

6.2.4. Процедура DSPMODE

Процедура DSPMODE устанавливает режим отображения видео-памяти на экран монитора. Если второй параметр (disp) равен TRUE, то номер текущей отображаемой страницы должен быть нечетным. Четыре варианта установки параметров lace и disp соответствуют четырем возможным значениям параметра interlace mode оператора SCREEN языка Basic (см. описание языка Basic для компьютеров серии MSX-2).

Заголовок : PROCEDURE DSPMODE(lace,alter:INTEGER):

Параметры : lace = TRUE - отображение со смещением
: на 1/2 точки по вертикали
: (interlace mode)
: = FALSE - отображение без смещения
: disp = TRUE - отображение с чередованием
: текущей и предыдущей стра-
: ниц

Результат : -

6.2.5. Процедура BLON

Процедура BLON устанавливает параметры мерцания и координаты мерцающего знакоместа. В качестве единицы измерения времени берется 1/6 секунды.

Заголовок : PROCEDURE BLON(x,y,fc,bc,ft,bt:INTEGER):

Параметры : x,y - координаты знакоместа
: fc - фасадный цвет мерцания
: bc - фоновый цвет мерцания
: ft - продолжительность интервала включе-
: ния цветов мерцания
: bt - продолжительность интервала выключе-
: ния цветов мерцания

Результат : -

6.2.6. Процедура BLOFF

Процедура BLOFF снимает мерцание знакоместа с заданными координатами. Параметры мерцания при этом не сбрасываются.

Заголовок : PROCEDURE BLOFF(x,y:INTEGER);

Параметры : x,y - координаты знакоместа

Результат : -

6.2.7. Процедура DIRECT

Процедура DIRECT устанавливает номер страницы-источника и страницы-цели для процедур копирования VCOPY и WCOPY. Номера страниц берутся по модулю 4 для экранных режимов 5, 6 и по модулю 2 для экранных режимов 7, 8.

Заголовок : PROCEDURE DIRECT(spage,dpage:INTEGER);

Параметры : spage - номер страницы-источника

: dpage - номер страницы-цели

Результат : -

6.2.8. Процедура VCOMP

Процедура VCOMP сравнивает побайтно содержимое блока данных в памяти с указанным блоком в видеопамяти. Адресация видеопамяти выполняется в пределах активной страницы.

Заголовок : PROCEDURE VCOMP(source,dest,length:INTEGER
):BOOLEAN;

Параметры : source - адрес сравниваемого блока в
: памяти

: dest - адрес сравниваемого блока в
: видеопамяти

: length - размер сравниваемых блоков в
 : байтах
Результат : VCOMP = TRUE - содержимое блоков
 : совпадает
 : = FALSE - в противном случае

6.2.9. Процедура VFILL

Процедура VFILL заполняет указанную область видеопамати данным байтом. Адресация видеопамати выполняется в пределах активной страницы.

Заголовок : PROCEDURE VFILL(byte:CHAR;dest,length
 : : INTEGER) ;
Параметры : byte - записываемый байт
 : dest - адрес записываемой области видео-
 : памяти
 : length - размер области
Результат : -

6.2.10. Процедура VGET

Процедура VGET читает данные из видеопамати. Адресация видеопамати выполняется в пределах активной страницы.

Заголовок :PROCEDURE VGET(source,dest,length:INTEGER) ;
Параметры :source - адрес данных в видеопамати
 :dest - адрес, в памяти, по которому раз-
 : мещаются считанные данные
 :length - количество считываемых байт
Результат : -

6.2.11. Процедура VPUT

Процедура VPUT записывает данные в видеопамять. Адресация видеопамати выполняется в пределах активной страницы.

Заголовок :PROCEDURE VPUT(source,dest,length:INTEGER);

Параметры :source - адрес данных в памяти

:dest - адрес, в видеопамати, по которому
: записываются данные

:length - количество записываемых байт

Результат : -

6.2.12. Процедура VCOPY

Процедура VCOPY выполняет копирование содержимого одной области видеопамати в другую область. По умолчанию номера страницы-источника и страницы-цели считаются равными нулю. В противном случае пользователь должен указать эти номера, воспользовавшись процедурой DIRECT. Вся необходимая для копирования информация должна быть размещена пользователем в восьмибайтовом массиве (дескрипторе), адрес которого передается процедуре копирования в качестве параметра. Дескриптор имеет следующую структуру:

flag (2 байта) = 1 - инкрементное копирование
=-1 - декрементное копирование

source (2 байта) - адрес источника

dest (2 байта) - адрес цели

length (2 байта) - размер копируемого блока

Адресация источника и цели выполняется в пределах соответствующих страниц.

Заголовок : PROCEDURE VCOPY(descriptor:INTEGER);
Параметры : descriptor - адрес дескриптора копирования
Результат : -

6.2.13. Процедура BLKFIL

Процедура BLKFIL заполняет прямоугольный блок видеопамя-
ти заданным байтом.

Заголовок :PROCEDURE BLKFIL(byte:CHAR;reper,destx,
: desty,sizeх,sizeу:INTEGER);
Параметры :byte - записываемый байт
:reper = 0 - копирование слева направо
: сверху вниз
: = 1 - справа налево сверху вниз
: = 2 - слева направо снизу вверх
: = 3 - справа налево снизу вверх
:destx - X-координата начальной вершины бло-
: ка (в пикселах)
:desty - Y-координата начальной вершины бло-
: в (в пикселах)
:sizeх - ширина блока в пикселах
:sizeу - высота блока в пикселах
Результат : -

6.2.14. Процедура BLKGET

Процедура BLKGET читает прямоугольный блок из видеопамя-
ти.

Заголовок :PROCEDURE BLKGET(reper,soursx,sourcy,sizeх,
: sizeу,dest:INTEGER);
Параметры :reper - см. пункт 6.2.13

```

:sourcex - X-координата начальной вершины
:         блока (в пикселах)
:sourcy  - Y-координата начальной вершины
:         блока (в пикселах)
:sizex   - ширина блока в пикселах
:sizey   - высота блока в пикселах
:dest    - адрес в памяти, по которому запи-
:         сываются считанные данные

```

Результат :-

6.2.15. Процедура BLKPUT

Процедура BLKPUT записывает прямоугольный блок в видео-память.

```

Заголовок :PROCEDURE BLKPUT(source, reper, destx, desty,
                             sizeх, sizeу: INTEGER);

```

```

Параметры :source - адрес в памяти, откуда берутся
:               записываемые данные
:reper         - см. пункт 6.2.13
:destx        - X-координата начальной вершины
:               блока (в пикселах)
:desty        - Y-координата начальной вершины
:               блока (в пикселах)
:sizeх        - ширина блока в пикселах
:sizeу        - высота блока в пикселах

```

Результат :-

6.2.16. Процедура BLKCPY

Процедура BLKCPY копирует прямоугольный блок видеопамати

```

Заголовок :PROCEDURE BLKCPY(reper, sourceх, sourceу, sizeх,
:                             sizeу, destx, desty
:                             : INTEGER);

```


Результат :-

Резултат :-

6.2.19. Процедура WSAVE

Процедура WSAVE сохраняет в памяти содержимое указанного графического окна.

Заголовок : PROCEDURE WSAVE(x,y,width,height,dest
: : INTEGER);

Параметры : x,y - координаты сохраняемого окна
: width - ширина окна
: height - высота окна
: dest - адрес области сохранения

Результат : -

6.2.20. Процедура WREST

Процедура WREST восстанавливает из памяти содержимое указанного графического окна.

Заголовок : PROCEDURE WREST(source,x,y,width,height
: : INTEGER);

Параметры : source - адрес области сохранения
: x,y - координаты восстанавливаемого
: окна
: width - ширина окна
: height - высота окна

Результат : -

6.2.21. Процедура WCOPY

Процедура WCOPY копирует содержимое одного графического окна в другое графическое окно тех же размеров. Другими словами, в указанное место экрана выводится копия заданного графического окна.

Заголовок : PROCEDURE WCOPY(x,y,width,height,destx,
: desty: INTEGER) ;
Параметры : x,y - координаты копируемого окна
: width - ширина копируемого окна
: height - высота копируемого окна
: destx,desty - координаты копии
Результат : -

6.2.22. Функция SETMEM

Функция SETMEM представляет собой "болванку", предназначенную для выделения памяти в программе и ее инициализации. Для каждого инициализируемого участка памяти необходимо создать копию этой функции с оригинальным именем. Затем между процедурой INLINE и оператором END этой копии вставить еще одну процедуру INLINE, параметрами которой являются числовые значения байтов инициализируемого участка памяти. После этого, если Вы вызовете функцию, передав ей размер участка памяти в качестве параметра, то она вернет адрес этого участка памяти. Помните, что если число параметров вставленной процедуры INLINE не будет совпадать с значением параметра функции, то это может привести к непредсказуемым последствиям.

Заголовок : FUNCTION SETMEM(length: INTEGER): INTEGER;
Параметры : length - размер выделяемого участка памяти
Результат : SETMEM - адрес выделенного участка памяти

КЛАВИАТУРА.

7.1. НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ ПАКЕТА

Пакет предоставляет пользователю возможность работать с клавиатурой на "нижнем" уровне, т.е. обращаясь непосредственно к ее матрице. Этим обеспечивается:

- надежность определения состояния клавиатуры в момент опроса;
- возможность работы с "бескодовыми" клавишами, такими как SHIFT, CAPS, CTRL, PУC.

Использование этого пакета при создании обучающих программ позволит значительно улучшить их эргономические качества и облегчить написание.

Включенные в пакет процедуры и функции реализуют два взаимно дополняющих способа доступа к клавиатуре: синхронный и асинхронный.

Синхронный ввод с клавиатуры осуществляется непосредственно в момент вызова соответствующей функции, в то время как асинхронный ввод выполняется непрерывно 60 раз в секунду. С помощью процедур синхронного доступа, таким образом, Вы можете определить текущее состояние клавиатуры. Процедуры асинхронного доступа позволят Вам узнать, была ли нажата интересующая Вас клавиша за какой-либо промежуток времени. В этом случае результаты сканирования клавиатуры накапливаются в однобайтовом аккумуляторе (запоминается информация только о восьми клавишах, выбранных пользователем с помощью процедуры KEYON). Считывание этой информации выполняется уже синхронно в момент вызова соответствующей процедуры или функции, после чего она сбрасывается. Кроме того, инициализация аккумулятора (сброс информации о результатах сканирования соответствующей клавиши) выполняется при вызове процедуры KEYON.

К процедурам первого типа относятся SNSMAT, KEY, ANYKEY,

STRIG, STICK, FKEY, LETTER, ESC, TAB, CTRL, SHIFT, GRAPH, CR, BS, SELECT, HOME, INS, DEL, STOP, CAPS, RUS.

К функциям и процедурам второго типа - KEYON, PRESS, WAIT.

Кроме того, в пакет включена процедура KILBUF дополняющая стандартные средства ввода через буфер клавиатуры, а также процедура LAMP, переключающая лампочки РУС и CAPS.

Все процедуры пакета объединены в одном файле - KEY2.PAS. Поэтому, чтобы включить их в свою программу пользователю достаточно воспользоваться оператором \$F ({ \$F KEY2.PAS }).

Рекомендуется однако, перед включением пакета в текст программы, составить для себя его рабочий вариант, исключив функции обращения к тем клавишам, которые не использует Ваша программа.

Для работы с маской клавиатуры (функции SNSMAT и KEY) используйте схему матрицы клавиатуры, приведенную в Приложении.

7.2. ОПИСАНИЕ ФУНКЦИИ ПАКЕТА

В пакет входят следующие процедуры и функции:

- SNSMAT - чтение маски ряда клавиатуры
- KEY - определение состояния клавиши, заданной ее положением в матрице клавиатуры
- ANYKEY - определяет, нажата ли хоть одна клавиша
- STRIG - определение состояния кнопок джойстиков и клавиши <ПРОБЕЛ>
- STICK - определение состояния рукояток джойстиков и клавиш со стрелками
- FKEY - определение состояния функциональных клавиш
- LETTER - определение состояния буквенно-цифровой клавиши, заданной кодом одного из соответствующих ей символов
- ESC, TAB, CTRL, SHIFT, GRAPH, CR, BS, SELECT, HOME, INS, DEL, STOP, CAPS, RUS - определение состояния "черных" клавиш
- LAMP - включение/выключение лампочек РУС и CAPS и соответствующих регистров клавиатуры
- KEYON - включение асинхронного опроса клавиши, за-

- данной своим кодом
- KEYOFF - выключение асинхронного опроса всех клавиш
 - PRESS - определение состояния асинхронно опрашиваемой клавиши с момента предыдущего вызова этой функции или процедуры KEYON
 - WAIT - ожидание нажатия клавиши или истечения заданного промежутка времени
 - KILBUF - очистка буфера клавиатуры
-

7.2.1. Функция SNSMAT

Функция SNSMAT возвращает маску указанного ряда клавиатуры в младшем байте результата. Нулевые биты соответствуют нажатым клавишам. Матрица клавиатуры приведена в Приложении В

Заголовок : FUNCTION SNSMAT(row: INTEGER): INTEGER;

Параметры : row - номер ряда,

Результат : SNSMAT - маска указанного ряда

7.2.2 Функция KEY

Функция KEY возвращает TRUE, если нажата клавиша, находящаяся в колонке COL ряда ROW матрицы клавиатуры (см. Приложение В).

Заголовок : FUNCTION KEY(ROW, COL: INTEGER): BYTE;

Параметры : ROW - номер ряда, COL - колонка.

Результат : TRUE, если нажата, FALSE - если нет.

7.2.3. Функция ANYKEY

Функция ANYKEY возвращает TRUE, если нажата какая-либо клавиша на клавиатуре, в том числе SHIFT, GRAPH, STOP и т.д. В противном случае возвращается FALSE.

Заголовок : FUNCTION ANYKEY: BOOLEAN;

Параметры : -

Результат : ANYKEY - TRUE - клавиша нажата

: FALSE- клавиша не нажата

7.2.4. Функция STRIG

Функция STRIG является аналогом оператора STRIG(n) в BASIC'е. Возвращает TRUE, если нажата кнопка n (n=0 - пробел, n=1 - кнопка первого джойстика, n=2 - второго) В противном случае возвращается FALSE.

Заголовок : FUNCTION STRIG(n:INTEGER):BOOLEAN;

Параметры : n - номер кнопки

Результат : STRIG - TRUE - кнопка нажата

: FALSE- кнопка не нажата

7.2.5. Функция STICK

Функция STICK является аналогом оператора STICK(n) в BASIC'е. Возвращает состояние клавиш со стрелками при n=0 или рукоятки джойстика при n=1,2.

Заголовок : FUNCTION STICK(n:INTEGER):INTEGER;

Параметры : n - номер кнопки

Результат : STICK - от 0 до 8

7.2.6. Функция FKEY

Функция FKEY возвращает TRUE если нажата функциональная клавиша с номером n. В противном случае возвращается FALSE.

Заголовок : FUNCTION FKEY(n:INTEGER):BOOLEAN;

Параметры : n - номер функциональной клавиши

Результат : FKEY = TRUE - клавиша нажата

: = FALSE- клавиша не нажата

7.2.7. Функция LETTER

Функция LETTER возвращает TRUE если нажата "белая" клавиша, на которой изображен символ с указанным кодом. В противном случае возвращается FALSE.

Заголовок : FUNCTION LETTER(code:CHAR):BOOLEAN;

Параметры : code - код клавиши

Результат : LETTER = TRUE - клавиша нажата
 : = FALSE- клавиша не нажата

7.2.8. Функция ESC

Функция ESC проверяет, нажата ли клавиша <esc>.

Заголовок : FUNCTION ESC:BOOLEAN;

Параметры : -

Результат : ESC = TRUE - клавиша нажата
 : = FALSE- клавиша не нажата

7.2.9. Функция TAB

Функция TAB проверяет, нажата ли клавиша <tab>.

Заголовок : FUNCTION TAB:BOOLEAN;

Параметры : -

Результат : TAB = TRUE - клавиша нажата
 : = FALSE- клавиша не нажата

7.2.10. Функция CTRL

Функция CTRL проверяет, нажата ли клавиша <ctrl>.

Заголовок : FUNCTION CTRL:BOOLEAN;

Параметры : -

Результат : CTRL = TRUE - клавиша нажата
 : = FALSE- клавиша не нажата

7.2.11. Функция SHIFT

Функция SHIFT проверяет, нажата ли клавиша <shift>.

Заголовок : FUNCTION SHIFT:BOOLEAN;

Параметры : -

Результат : SHIFT = TRUE - клавиша нажата

: = FALSE- клавиша не нажата

7.2.12. Функция GRAPH

Функция GRAPH проверяет, нажата ли клавиша <graph>.

Заголовок : FUNCTION GRAPH:BOOLEAN;

Параметры : -

Результат : GRAPH = TRUE - клавиша нажата

: = FALSE- клавиша не нажата

7.2.13. Функция CR

Функция CR проверяет, нажата ли клавиша возврата каретки

Заголовок : FUNCTION CR:BOOLEAN;

Параметры : -

Результат : CR = TRUE - клавиша нажата

: = FALSE- клавиша не нажата

7.2.14. Функция BS

Функция BS проверяет, нажата ли клавиша <bs>

Заголовок : FUNCTION BS:BOOLEAN;

Параметры : -

Результат : BS = TRUE - клавиша нажата

: = FALSE- клавиша не нажата

7.2.15. Функция SELECT

Функция SELECT проверяет, нажата ли клавиша <select>

Заголовок : FUNCTION SELECT:BOOLEAN;

Параметры : -

Результат : SELECT = TRUE - клавиша нажата

: = FALSE- клавиша не нажата

7.2.16. Функция HOME

Функция HOME проверяет, нажата ли клавиша <cls/home>

Заголовок : FUNCTION HOME:BOOLEAN;

Параметры : -

Результат : HOME = TRUE - клавиша нажата

: = FALSE- клавиша не нажата

7.2.17. Функция INS

Функция INS проверяет, нажата ли клавиша <ins>

Заголовок : FUNCTION INS:BOOLEAN;

Параметры : -

Результат : INS = TRUE - клавиша нажата

: = FALSE- клавиша не нажата

7.2.18. Функция DEL

Функция DEL проверяет, нажата ли клавиша

Заголовок : FUNCTION DEL:BOOLEAN;

Параметры : -

Результат : DEL = TRUE - клавиша нажата

: = FALSE- клавиша не нажата

7.2.19. Функция STOP

Функция STOP проверяет, нажата ли клавиша <stop>

Заголовок : FUNCTION STOP:BOOLEAN;

Параметры : -

Результат : STOP = TRUE - клавиша нажата

: = FALSE- клавиша не нажата

7.2.20. Функция CAPS

Функция CAPS проверяет, зажжена ли лампочка <caps>

Заголовок : FUNCTION CAPS:BOOLEAN;

Параметры : -

Результат : CAPS = TRUE - клавиша нажата

: = FALSE- клавиша не нажата

7.2.21. Функция RUS

Функция RUS проверяет, зажжена ли лампочка <рус>

Заголовок : FUNCTION RUS:BOOLEAN;

Параметры : -

Результат : RUS = TRUE - клавиша нажата

: = FALSE- клавиша не нажата

7.2.22. Процедура LAMP

Процедура LAMP переключает лампочки РУС и CAPS.

Заголовок : PROCEDURE LAMP (1:CHAR, sw:BOOLEAN);

Параметры : 1 = 'R' - переключается лампочка <рус>

: = 'C' - переключается лампочка <caps>

: sw = TRUE - лампочка включается

: = FALSE- лампочка выключается

Результат : -

7.2.23. Процедура KEYON

Процедура KEYON закрепляет за указанной клавишей идентифицирующий ее номер, который в дальнейшем будет использоваться в функциях асинхронного ввода с клавиатуры. После выполнения этой процедуры бит аккумулятора клавиатуры, соответствующий указанному номеру, сбрасывается. Идентификация "белых" клавиш клавиатуры выполняется с помощью кода любого из изображенных на ней символов. В случае "черных" клавиш идентификация выполняется по закрепленным за ними кодам (<esc>-27,<tab>-9,<возврат каретки>-13,<bs>-8,<select>-24,<cls/home>-11,<ins>-18,-127,<стрелка вверх>-30,<стрелка вправо>-28,<стрелка вниз>-31,<стрелка влево>-29).

Для клавиш <ctrl>,<shift>,<graph>,<caps>,<rus>,<stop> в качестве таких кодов берутся коды '\00','\01','\02','\03','\04','\05' соответственно.

Заголовок : PROCEDURE KEYON (n:INTEGER;code:CHAR);

Параметры : n : присваиваемый клавише номер,

: : 1 <= n <= 8

: code : код клавиши

Результат : -

7.2.24 Процедура KEYOFF

Функция KEYOFF сбрасывает номера, закрепленные за клавишами функцией KEYON. Ее необходимо вызывать перед завершением программы, вызывавшей KEYON.

Заголовок : PROCEDURE KEYOFF;

Параметры : -

Результат : -

7.2.25. Функция PRESS

Функция PRESS проверяет была ли за время, прошедшее с мо-

мента ее предыдущего вызова, нажата клавиша с указанным номером (закрепленным за ней функцией KEYON). Заметим, что если перед вызовом описываемой функции была выполнена функция KEYON для соответствующего номера, то все нажатия на указанную клавишу, имевшие до этого место, игнорируются.

Заголовок : FUNCTION PRESS(n:INTEGER):BOOLEAN;

Параметры : n - номер клавиши

Результат : PRESS = TRUE - клавиша была нажата
 : = FALSE - клавиша не была нажата

7.2.26. Процедура WAIT

Процедура WAIT ожидает пока не будет выполнено одно из следующих условий:

- нажата клавиша с указанным номером (n)
- истек указанный промежуток времени (t)

Отсчет времени и проверка нажата ли указанная клавиша начинаются с момента вызова процедуры. Если $n=0$ или $n>8$, то первая проверка не выполняется, если $t=0$, то не выполняется вторая проверка.

Заголовок : PROCEDURE WAIT(n,t:INTEGER);

Параметры : n - номер клавиши

 : t - время ожидания в 60-х долях секунды

Результат : -

7.2.27 Процедура KILBUF

Процедура KILBUF чистит буфер клавиатуры.

Заголовок : PROCEDURE KILBUF;

Параметры : -

Результат : -

ГРАФИЧЕСКИЙ ПАКЕТ LOGO-PAS

8.1. НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ ПАКЕТА.

Процедуры пакета предназначены для исполнения графических алгоритмов в программах на языке Паскаль (Разрешающая способность графики - 256 X 212 , текста - 64 X 24).

"Исполнителем" в пакете является черепаха. Система координат(полярная) связана с местонахождением черепахи на экране. Под воздействием команд черепаха может перемещаться, оставляя (или не оставляя) за собой линию, менять угол поворота, цвет "карандаша" и т.д. Всего в системе 15 черепах, из которых в каждый момент доступна одна, выбираемая произвольно с помощью процедуры TELL.

8.2. ОПИСАНИЕ ПРОЦЕДУР И ФУНКЦИЙ ПАКЕТА.

В состав пакета входят следующие функции и процедуры.

- XCOR -абсцисса рабочей черепахи
- YCOR -ордината рабочей черепахи
- HEADING -ориентация рабочей черепахи
- TCOLOR -цвет рабочей черепахи
- PCOLOR -цвет пера рабочей черепахи
- WCOLOR -цвет графического окна
- SHOWN -состояние черепахи
- DOWN -состояние пера черепахи
- INSIDE -проверка на границы графического окна
- LOGO -инициализация
- HT -подавление изображения черепахи
- ST -восстановление изображения черепахи
- PU -поднять перо
- PD -опустить перо

- FD -вперед
- BK -назад
- RT -направо
- LT -налево
- SETC -установка цвета черепахи
- SETPC -установка цвета пера черепахи
- SETXCOR -установка абсциссы рабочей черепахи
- SETYCOR -установка ординаты рабочей черепахи
- SETPOS -перемещение в точку с заданными координатами
- JUMP -установка координат рабочей черепахи.
- SETH -установка ориентации рабочей черепахи.
- WINDOW -инициализация графического окна.
- DROP -рисование точки в текущих координатах
- FILL -закрашивание цветом пера.
- SLEEP -Пауза с выходом по пробелу.
- WAIT -Временная пауза.
- CG -Очистка графического окна.
- CS -Стирание экрана.
- TXTCOL -Задание цвета букв.
- CT -Стирание текста.
- SETCUR -Позиционирование курсора.
- TELL -Установка рабочей черепахи
- CURRENT -Определение номера рабочей черепахи.

8.2.1. Функция XCOR

Заголовок: FUNCTION XCOR:REAL;
 Результат: абсцисса исполнителя ("черепахи")

8.2.2. Функция YCOR

Заголовок: FUNCTION YCOR:REAL;
 Результат: ордината черепахи

8.2.3. Функция HEADING

Заголовок: FUNCTION HEADING: INTEGER;

Результат: угол отклонения черепахи от вертикальной оси в градусах.

8.2.4. Функция TCOLOR

Заголовок: FUNCTION TCOLOR: INTEGER;

Результат: цвет черепахи

8.2.5. Функция PCOLOR

Заголовок: FUNCTION PCOLOR: INTEGER;

Результат: цвет пера черепахи

8.2.6. Функция WCOLOR

Заголовок: FUNCTION WCOLOR: INTEGER;

Результат: цвет графического окна

8.2.7. Функция SHOWN

Заголовок: FUNCTION SHOWN: BOOLEAN;

Результат: TRUE, если не установлен режим "HT" (см.);
FALSE, если установлен.

8.2.8. Функция DOWN

Заголовок: FUNCTION DOWN: BOOLEAN;

Результат: TRUE, если не установлен режим "PU" (см.);
FALSE, если установлен.

8.2.9. Функция INSIDE

Заголовок: FUNCTION INSIDE:BOOLEAN;

Результат: TRUE, если черепаха находится внутри графического окна, иначе FALSE

8.2.10. Процедура LOGO

Действие: инициализация системы, задание экранного режима.

Заголовок: PROCEDURE LOGO;

8.2.11. Процедура HT

Действие: Установка режима, при котором черепаха становится невидимой.

Заголовок: PROCEDURE HT;

8.2.12. Процедура ST

Действие: Отменяет режим HT;

Заголовок: PROCEDURE ST;

8.2.13. Процедура PU

Действие: Установка режима, при котором черепаха при перемещении не рисует. В режиме PU также не действуют процедуры DROP, FILL

Заголовок: PROCEDURE PU;

8.2.14. Процедура PD

Действие: Отменяет режим PU;

Заголовок: PROCEDURE PD;

8.2.15. Процедура FD

Действие: Перемещение черепахи на D черепашьих шагов вперед. Если не установлен режим PU, черепаха прочертит линию.

Заголовок: PROCEDURE FD (D:REAL);

Параметр : Количество шагов (может быть дробным)

8.2.16. Процедура BK

Действие: Перемещение черепахи на D черепашьих шагов назад. Если не установлен режим PU, черепаха прочертит линию.

Заголовок: PROCEDURE BK (D:REAL);

Параметр : Количество шагов (может быть дробным)

8.2.17. Процедура RT

Действие: Поворот черепахи по часовой стрелке.

Заголовок: PROCEDURE RT (A:INTEGER);

Параметр : Угол поворота (в градусах)

8.2.18. Процедура LT

Действие: Поворот черепахи против часовой стрелки.

Заголовок: PROCEDURE LT(A:INTEGER);
Параметр : Угол поворота(в градусах)

8.2.19. Процедура SETC

Действие: Смена цвета черепахи

Заголовок: PROCEDURE SETC(C:INTEGER);
Параметр : Цвет (0+15)

8.2.20. Процедура SETPC

Действие: Смена цвета пера черепахи

Заголовок: PROCEDURE SETPC(C:INTEGER);
Параметр : Цвет

8.2.21. Процедура SETXCOR

Действие: Изменение абсциссы черепахи. Если не задан режим PU, черепаха рисует в процессе перемещения.

Заголовок: PROCEDURE SETXCOR(X:REAL);
Параметр : Новое значение абсциссы

8.2.22. Процедура SETYCOR

Действие: Изменение ординаты черепахи. Если не задан режим PU, черепаха рисует в процессе перемещения.

Заголовок: PROCEDURE SETYCOR(Y:REAL);

Параметр : Новое значение ординаты

8.2.23. Процедура SETPOS

Действие: Изменение координат черепахи. Если не задан режим PU, черепаха рисует в процессе перемещения.

Заголовок: PROCEDURE SETPOS(X,Y:REAL);

Параметры: Новые значения абсциссы и ординаты

8.2.24. Процедура JUMP

Действие: Быстрое, без рисования, перемещение черепахи в точку с координатами X,Y

Заголовок: PROCEDURE JUMP(X,Y:REAL);

Параметры: Новые значения абсциссы и ординаты

8.2.25. Процедура SETH

Действие: Установка угла отклонения черепахи от вертикали

Заголовок: PROCEDURE SETH(A:INTEGER);

Параметр : Угол(в градусах)

8.2.26. Процедура WINDOW

Действие: Установка границ и цвета графического окна.

Заголовок: PROCEDURE WINDOW(X1,Y1,X2,Y2,C:INTEGER);

Параметры: Координаты левого верхнего и правого
нижнего угла, цвет

8.2.27. Процедура DROP

Действие: Рисование точки в текущих координатах цветом пера

Заголовок: PROCEDURE DROP;

8.2.28. Процедура FILL

Действие: Закрашивание замкнутой области, внутри которой находится черепаха, цветом пера. Линия, ограничивающая замкнутую область должна иметь тот же цвет.

Заголовок: PROCEDURE FILL;

8.2.29. Процедура SLEEP

Действие: Перевод черепахи в состояние ожидания нажатия клавиши "пробел"

Заголовок: PROCEDURE SLEEP;

8.2.30. Процедура WAIT

Действие: Пауза.

Заголовок: PROCEDURE WAIT (T: INTEGER);

Параметр : Время в "тиках" (50-ых долях секунды).

8.2.31. Процедура CG

Действие: Очистка графического окна.

Заголовок: PROCEDURE CG;

8.2.32. Процедура CS

Действие: Стирание экрана.

Заголовок: PROCEDURE CS;

8.2.33. Процедура TXTCOL

Действие: выбор цвета букв при печати текста.

Заголовок: PROCEDURE TXTCOL(C: INTEGER) ;

Параметры: Цвет

8.2.34. Процедура CT

Действие: стирание прямоугольного участка текста.

Заголовок: PROCEDURE CT(X,Y,DX,DY: INTEGER) ;

Параметры: X,Y: номер столбца и строки
 DX,DY количество знаков по горизонтали и вертикали

8.2.35. Процедура SETCUR

Действие: позиционирование курсора.

Заголовок: PROCEDURE SETCUR(X,Y: INTEGER) ;

Параметр : Номер столбца и строки

8.2.36. Процедура TELL

Задает номер рабочей черепахи, т. е. черепахи, которой будут адресованы следующие команды.

Заголовок: PROCEDURE TELL(N:INTEGER)
Параметры: Номер черепахи (0+15)

8.2.37. Функция CURRENT

Определяет, какая черепаха является рабочей.

Заголовок: FUNCTION CURRENT:INTEGER;
Результат: Номер черепахи

8.3. ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ.

Пакет LOGO-PAS является замкнутым и независимым. Для функционирования пакета нет необходимости подключать какие бы то ни было процедуры других пакетов, в частности, например, экранный режим задается автоматически в процедуре начальной инициализации LOGO.

Процедуры пакета используют в качестве рабочей область памяти с FB50H по FB63H.

К пакету прилагается демонстрационная программа TUTOR, написанная на его основе. Описание этой программы приведено в приложении G.

ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ ДЛЯ СИСТЕМНОГО ПРОГРАММИСТА

9.1. СИСТЕМНЫЕ ПЕРЕМЕННЫЕ

Системные переменные и рабочие области предлагаемых инструментальных программных средств размещаются в области системных переменных Бейсик-системы. Информация, размещающаяся в этих переменных и областях может быть постоянной, обновляемой и временной.

Постоянная информация заносится в область системных переменных и модифицируется только посредством специальных процедур установки (SCREEN, SCURVE, DASH, MRKON, SHABLON, KEYON). Модификация содержимого соответствующих системных областей может привести к непредсказуемым результатам (ре-старт системы, зависание и пр.).

К обновляемой информации относятся данные, описывающие текущее состояние и параметры графической среды. Модификация этой информации может привести к изменению параметров графической среды. Однако система в целом останется работоспособной.

И наконец к временной относится информация, создаваемая, модифицируемая и используемая только в процессе работы отдельных процедур. Занимаемые этой информацией рабочие области могут быть использованы пользователем для временного хранения своей информации.

Описание системных переменных, используемых предлагаемыми инструментальными средствами приводится в Приложении С.

Большая часть системных переменных (в основном это переменные, характеризующие состояние графической среды) совпадает (как по адресам, так и по именам) с соответствующими системными переменными Бейсик-системы. Описание этих переменных в Приложении С не дается.

Для размещения постоянной системной информации, дополнительных параметров графической среды (обновляемая информация), а также временной информации используются следующие системные области Бейсик-системы:

- (DEFTBL=F6CAH, 17) - обновляемая информация;
- (TRPTBL+39=FC73H, 11) - обновляемая информация;
- (TRPTBL+54=FC82H, 16) - обновляемая информация;
- (TRPTBL+72=FC94H, 5) - постоянная информация;
- (TRPTBL+77=FC99H, 1) - временная информация;
- (PRMSTK=F6E4H, 72) - постоянная информация;
- (PRMSTK+72=F72CH, 32) - временная информация;
- (PRMPRV=F74CH, 107) - постоянная информация;
- (LOHMSK=F949H, 1) - временная информация;
- (LOHDIR=F94AH, 1) - временная информация;
- (LOHADR=F94BH, 2) - временная информация;
- (LOHCNT=F94DH, 2) - временная информация;
- (SKPCNT=F94FH, 2) - временная информация;
- (MIVCNT=F951H, 2) - временная информация;
- (PDIREC=F953H, 1) - временная информация;
- (LFPROG=F954H, 1) - временная информация;
- (RTPROG=F955H, 1) - временная информация;

(TEMP=F6A7H, 2)	- временная информация;
(TEMP2=F6BCH, 2)	- временная информация;
(TEMP3=F69DH, 2)	- временная информация;
(TEMP8=F69FH, 2)	- временная информация;
(TEMP9=F7B8H, 2)	- временная информация;
(TEMPPT=F678H, 2)	- временная информация;
(TEMPST=F67AH, 27)	- временная информация;
(HOLD=F83EH, 8)	- временная информация;
(HOLD2=F836H, 8)	- временная информация.

Имена всех дополнительных переменных, размещаемых в указанных выше областях, уникальны. Имена остальных системных переменных и областей те же, что и в Бейсик-системе.

9.2. ТАБЛИЦА РАСШИРЕННЫХ ПЕРЕХОДОВ

В процессе работы пользователь имеет возможность расширить или полностью "заместить" отдельные графические процедуры, перепрограммируя их. Для этого в них включены команды (хуки) передачи управления (с возвратом) на фиксированные адреса (точки входа) в области системных переменных. При инициализации этой области по этим адресам размещаются команды Ассемблера "RET". Эту область в дальнейшем будем называть таблицей расширенных переходов. Под каждую точку входа в этой таблице отводится по три байта. Инициализация таблицы расширенных переходов (то есть заполнение ее кодами команды "RET") выполняется процедурой SCREEN. Для того, чтобы теперь при передаче управления в некоторую точку входа (HOOK) таблицы расширенных переходов были выполнены предусмотренные действия, по адресу HOOK надо поместить команду перехода на точку входа соответствующей процедуры.

По своему функциональному назначению таблица расширенных переходов T.JUMP разбивается на три таблицы:

- T.SERV
- T.CURVE
- T.FILL

Эти таблицы размещаются в области системных переменных, начиная с адреса F74CH и занимают 87 байт (29 точек входа).

9.2.1. Таблица расширенных переходов T.SERV

Точки входа в эту таблицу устанавливаются процедурой SCREEN базового графического пакета (см. раздел 1). Эти точки входа предназначены для подключения наиболее часто используемых всеми графическими пакетами вспомогательных процедур.

Таблица расширенных переходов T.SERV размещается начиная с адреса F74CH, и занимает 24 байта (8 точек входа). Описание точек входа в эту таблицу приводится в Приложении D.

9.2.2. Таблица расширенных переходов T.CURVE

Точки входа в эту таблицу устанавливаются процедурами графического пакета "Линии" (см. раздел 2). По своему назначению эти точки делятся на две группы:

- точки входа, предназначенные для подключения вспомогательных процедур, наиболее часто используемых графическим пакетом "Линии";
- точки входа, предназначенные для расширения функциональных возможностей процедур базового графического пакета.

В состав первой группы входят точки входа T.UMOVE, T.RMOVE, T.DMOVE, T.LMOVE, предназначенные для подключения процедур перемещения графического курсора. Эти точки входа устанавливаются и снимаются процедурой установки режима рисования (SCURVE).

В состав второй группы входят точки входа T.SCALXY, T.RPCHK, T.PSET, T.LINE, T.BOX, T.CIRCLE, T.RPWDT и T.RMSET. Часть из них устанавливается и снимается процедурой SCURVE (T.SCALXY, T.RPCHK, T.PSET, T.LINE, T.BOX, T.CIRCLE) и предназначена для расширения функциональных возможностей таких процедур базового графического пакета, как PSET, LINE, BOX, CIRCLE, ELLIPS, ARC, а также входящих в состав графического пакета "Линии" процедур RPSET, SECTOR, SEGMENT (появляется возможность рисования пунктирных, штриховых и штрих пунктирных линий). Другая часть (T.RPWDT и T.RMSET) устанавливается соответственно процедурами DASH и MRKON и предназначена для рисования толстых линий (T.RPWDT) и рисования маркером (T.RMSET). Первая из этих точек входа снимается процедурой SCURVE (при переходе в базовый режим рисования линий), а вторая процедурой MRKOFF.

Таблица расширенных переходов T.CURVE размещается начиная с адреса F764H, и занимает 36 байт (12 точек входа). Описание точек входа в эту таблицу приводится в Приложении D.

9.2.3. Таблица расширенных переходов T.FILL

Точки входа в эту таблицу устанавливаются процедурой графического пакета "Закраски" (см. раздел 3) SHABLON. Эти точки входа предназначены исключительно для подключения вспомогательных процедур, используемых процедурой SPRAY. Инициализация точек входа в таблицу T.FILL может быть выполнена только процедурой SCREEN базового графического пакета.

Таблица расширенных переходов T.FILL размещается начиная с адреса F788H, и занимает 27 байт (9 точек входа). Описание точек входа в эту таблицу приводится в Приложении D.

ПРИЛОЖЕНИЕ А

ТАБЛИЦА КОДОВ ЛОГИЧЕСКИХ ОПЕРАЦИЙ

- если операция двухоперандная (AND, OR, XOR, TAND, TOR, TXOR), то она выполняется над основным цветом и цветом соответствующей точки экрана
- если операция однооперандная (IMP, NOT, TIMP, TNOT), то она выполняется над основным цветом

Логическое имя	Операция	Код
IMP	Выводится основной цвет	0 0 0 0
AND	Логическое "И"	0 0 0 1
OR	Логическое "ИЛИ"	0 0 1 0
XOR	Исключающее "И"	0 0 1 1
NOT	Отрицание	0 1 0 0

TIMP	Выводится основной цвет	1 0 0 0
TAND	Логическое "И"	1 0 0 1
TOR	Логическое "ИЛИ"	1 0 1 0
TXOR	Исключающее "И"	1 0 1 1
TNOT	Отрицание	1 1 0 0

ПРИМЕЧАНИЕ: операции, второй группы отличаются от операций первой группы только тем, что цвета точек, в которые выводится нулевой цвет, не меняются

КООРДИНАТНЫЕ СЕТКИ ДЛЯ ВЫСШИХ ГРАФИЧЕСКИХ РЕЖИМОВ

SCREEN 5	
(0,0)	(255,0)
0 стр.	
(0,255)	(255,255)

(0,256)	(255,256)
1 стр.	
(0,511)	(255,511)

(0,512)	(255,512)
2 стр.	
(0,767)	(255,767)

(0,768)	(255,768)
3 стр.	
(0,1023)	(255,1023)

SCREEN 8	
(0,0)	(255,0)
0 стр.	
(0,255)	(255,255)

(0,256)	(255,256)
1 стр.	
(0,511)	(255,511)

00000H	SCREEN 6
	(0,0) (511,0)
	0 стр.
	(0,255) (511,255)
08000H	
	(0,256) (511,256)
	1 стр.
	(0,511) (511,511)
10000H	
	(0,512) (511,512)
	2 стр.
	(0,767) (511,767)
18000H	
	(0,768) (511,768)
	3 стр.
	(0,1023) (511,1023)
1FFFFH	

00000H	SCREEN 7
	(0,0) (511,0)
	0 стр.
	(0,255) (511,255)
10000H	
	(0,256) (511,256)
	1 стр.
	(0,511) (511,511)
1FFFFH	

ПРИЛОЖЕНИЕ В.

Матрица клавиатуры компьютеров YAMANA YIS503IIR, YIS503IIIR, YIS805R.

х-колонка, у-ряд

Ряды 9 и 10 соответствуют клавишам дополнительной цифровой клавиатуры (только на YIS805R).

x0	x1	x2	x3	x4	x5	x6	x7	
9	;	1	2	3	4	5	6	y0
7	8	0	=	-	h	*	v	y1
\	>	b	@	<	?	f	i	y2
s	w	u	a	p	r	[o	y3
l	d	x	t	j	z	j	k	y4
y	e	g	m	c	!	n	q	y5
shf	ctr	grf	cps	pyc	f1	f2	f3	y6
f4	f5	esc	tab	stop	bs	sel	ret	y7
spc	cls	ins	del	->	▲	▼	<-	y8
ret	+	*	0	1	2	3	4	y9
5	6	7	8	9	-	,	.	y10

ПРИЛОЖЕНИЕ С

ОПИСАНИЕ СИСТЕМНЫХ ПЕРЕМЕННЫХ И РАБОЧИХ ОБЛАСТЕЙ

1. Имя : CALWRK

Назначение: Рабочая область, обеспечивающая вызов проце-
: дур интерпретатора

Адрес : PRMSTK=F6E4H

Размер : 72 байта

2. Имя : GETADDR

Назначение: Содержит текст процедуры, помещающей в ре-
: гистры HL текущее содержимое стека (адрес
: возврата). Используется в макрокоманде GADDR
: для определения содержимого регистра PC.
: Инициализируется макрокомандой SADDR в проце-
: дуре SCREEN базового графического пакета
: (Паскаль).

Адрес : CALWRK=F6E4H

Размер : 3 байта

3. Имя : HOOK

Назначение: Хук, используемый в процедуре вызова интерп-
: ретатора языка Бейсик. Содержит коды команды
: RET.
: Инициализируется процедурой SCREEN базового
: графического пакета (Паскаль) или функцией
: SCREEN пакета графических процедур
: (Си, Ассемблер).

Адрес : CALWRK+3=F6E7H

Размер : 3 байта

4. Имя : INTRP

Назначение: Содержит текст процедуры вызова интерпретато-
: ра языка Бейсик.

: Инициализируется процедурой SCREEN базового
: графического пакета (Паскаль) или функцией
: SCREEN пакета графических процедур
: (Си, Ассемблер).

Адрес : CALWRK+6=F6EАН

Размер : 40 байт

5. Имя : CINTRP

Назначение: Содержит продолжение текста процедуры вызова
: интерпретатора языка Бейсик.
: Инициализируется процедурой SCREEN базового
: графического пакета (Паскаль) или функцией
: SCREEN пакета графических процедур
: (Си, Ассемблер).

Адрес : CALWRK+46=F712H

Размер : 20 байт

6. Имя : SHOOK

Назначение: Содержит процедуру корректировки содержимого
: регистра SP. Команда перехода на эту процедуру записывается в область HOOK (см. выше)
: процедурой

Адрес : CALWRK+66=F726H

Размер : 6 байт

7. Имя : T.HOOK

Назначение: Область, содержащая таблицу расширенных переходов и таблицу интерслотового вызова BIOS'ов.

Адрес : PRMPRV=F74CH

Размер : 107 байт

8. Имя : T.JUMP

Назначение: Таблица расширенных переходов

Адрес : T.HOOK=F74CH

Размер : 87 байт

9. Имя : T.BIOS

Назначение: Таблица интерслотового вызова процедур BIOS

: Инициализируется при вызове процедуры

: (функции) SCREEN.

Адрес : T.HOOK+87=F7A3H

Размер : 20 байт

10. Имя : T.SERV

Назначение: Таблица расширенных переходов

Адрес : T.JUMP=F74CH

Размер : 24 байта

11. Имя : T.CURVE

Назначение: Таблица расширенных переходов

Адрес : T.JUMP+24=F764H

Размер : 36 байт

12. Имя : T.FILL

Назначение: Таблица расширенных переходов

Адрес : T.JUMP+60=F788H

Размер : 27 байт

13. Имя : B.SCALXY

Назначение: Процедура интерслотового вызова BIOS'a SCALXY

Адрес : T.BIOS=F7A3H

Размер : 5 байт

14. Имя : B.MAPXYC

Назначение: Процедура интерслотового вызова BIOS'a MAPXYC

Адрес : T.BIOS+5=F7A8H

Размер : 5 байт

15. Имя : B.SETC

Назначение: Процедура интерслотового вызова BIOS'a SETC

Адрес : T.BIOS+10=F7ADH

Размер : 5 байт

16. Имя : B.EXTROM

Назначение: Заготовка для интерслотового вызова SUBROM'a
: (расширенных BIOS'ов)

Адрес : T.BIOS+15=F7B2H

Размер : 5 байт

17. Имя : KEYMASK

Назначение: Аккумулятор клавиатуры

Адрес : TRPTBL+39=FC73H

Размер : 11 байт

18. Имя : KEYTBL

Назначение: Информация о расположении клавиш на клавиату-
: ре (асинхронный режим)

Адрес : TRPTBL+54=FC82H

Размер : 16 байт

19. Имя : SAVEHOOK

Назначение: Область сохранения хука HTIMI

Адрес : TRPTBL+72=FC94H

Размер : 5 байт

20. Имя : SAVBROM

Назначение: Область сохранения содержимого системной
: ячейки BASROM

Адрес : TRPTBL+77=FC99H

Размер : 1 байт

21. Имя : LINWRK

Назначение: Рабочая область макрокоманды рисования отрез-
: ка прямой M.LINE (базовый графический пакет;
: Паскаль) или функции _HLINE (пакет графичес-
: ких процедур; Си, Ассемблер).

Адрес : TEMPST=F67AH

Размер : 12 байт

22. Имя : LSIGNX

Назначение: Ориентация рисуемого отрезка прямой по оси X.

: = 1 если $X1 \leq X2$

: = -1 в противном случае.

Адрес : LINWRK=F67AH

Размер : 1 байт

23. Имя : LSIGNY

Назначение: Ориентация рисуемого отрезка прямой по оси Y.

: = 1 если $Y1 \leq Y2$

: = -1 в противном случае.

Адрес : LINWRK+1=F67BH

Размер : 1 байт

24. Имя : LDIFX

Назначение: Длина проекции рисуемого отрезка прямой на

: ось X (в пикселах).

Адрес : LINWRK+2=F67CH

Размер : 2 байта

25. Имя : LDIFY

Назначение: Длина проекции рисуемого отрезка прямой на

: ось Y (в пикселах).

Адрес : LINWRK+4=F67EH

Размер : 2 байта

26. Имя : LDIRB

Назначение: Направление основного перемещения графическо-

: го курсора при рисовании отрезка прямой

: (однобайтовое число от единицы до восьми).

Адрес : LINWRK+6=F680H

Размер : 1 байт

27. Имя : LDIRH

Назначение: Направление вспомогательного перемещения гра-
фического курсора при рисовании отрезка пря-
мой (однобайтовое число от единицы до восьми)
Адрес : LINWRK+7=F681H
Размер : 1 байт

28. Имя : LDIFB

Назначение: Длина проекции рисуемого отрезка прямой на
направление основного перемещения
(в пикселах).
Адрес : LINWRK+8=F682H
Размер : 2 байта

29. Имя : LDIFH

Назначение: Длина проекции рисуемого отрезка прямой на
направление вспомогательного перемещения
(в пикселах).
Адрес : LINWRK+10=F684H
Размер : 2 байта

30. Имя : ARCWRK

Назначение: Рабочая область макрокоманды вычисления пара-
метров дуги (сектора, сегмента) M.ARC (базо-
вый графический пакет; Паскаль) или функции
_ANGLE (пакет графических процедур;
Си, Ассемблер).
Адрес : HOLD=F83EH
Размер : 4 байта

31. Имя : CANGST

Назначение: Начальный угол дуги (сектора, сегмента) в
градусах.
Адрес : ARCWRK=F83EH
Размер : 2 байта

32. Имя : CANGEN

Назначение: Конечный угол дуги (сектора, сегмента) в градусах.

Адрес : ARCWRK+2=F840H

Размер : 2 байта

33. Имя : PNTWRK

Назначение: Рабочая область макрокоманд (Паскаль) или функций (Си, Ассемблер) закрашки замкнутой области.

Адрес : LOHMSK=F949H

Размер : 13 байт

34. Имя : PNTWRKA

Назначение: Дополнительная рабочая область макрокоманд (пакет "Закраски": Паскаль) или функций (пакет графических процедур: Си, Ассемблер) закрашки замкнутой области.

Адрес : TEMPST=F67AH

Размер : 13 байт

35. Имя : PTRSHB

Назначение: Адрес текущего шаблона закрашки

Адрес : PNTWRK=F949H

Размер : 2 байта

36. Имя : PNTFLAG

Назначение: Флаг закрашки текущего отрезка горизонтальной прямой

Адрес : PNTWRK+2=F94BH

Размер : 1 байт

37. Имя : PXLOW

Назначение: X-координата левой границы закрашиваемого горизонтального участка замкнутой области.

Адрес : PNTWRK+3=F94CH
Размер : 2 байта

38. Имя : PXHIGH

Назначение: X-координата правой границы закрашиваемого
горизонтального участка замкнутой области.

Адрес : PNTWRK+5=F94EH
Размер : 2 байта

39. Имя : PYLIN

Назначение: Y-координата закрашиваемого горизонтального
участка замкнутой области.

Адрес : PNTWRK+7=F950H
Размер : 2 байта

40. Имя : PNTPAR

Назначение: Область параметров закрашки:

: PNTPAR+SC - текущая глубина рекурсии
(2 байта)

: PNTPAR+FP - флаг поиска границы (1 байт)

: =0 - поиск до первой точки границы

: =1 - поиск до первой точки, отличной
от граничной

: PNTPAR+MD - направление поиска

: Здесь константы SC, FP и MD - смещения, рав-
ные соответственно 0, 2 и 3.

Адрес : PNTWRK+9=F952H
Размер : 4 байта

41. Имя : DOTSIZ

Назначение: Количество бит видеопамати, резервируемое в
текущем экранном режиме под одну точку

Адрес : PNTWRKA=F67AH
Размер : 1 байт

42. Имя : POINTER

Назначение: Адрес в видеопамати текущего закрашиваемого
: участка

Адрес : PNTWRKA+1=F67BH

Размер : 2 байта

43. Имя : MIDLEN

Назначение: Размер центральной части (начинающейся битом
: с номером, кратным восьми, и содержащей крат-
: ное восьми число бит) закрашиваемого отрезка
: горизонтальной прямой

Адрес : PNTWRKA+3=F67DH

Размер : 2 байта

44. Имя : LEFTL

Назначение: Длина левой части закрашиваемого отрезка го-
: ризонтальной прямой (см. п.44) в точках

Адрес : PNTWRKA+5=F67FH

Размер : 1 байт

45. Имя : RIGHTL

Назначение: Длина правой части закрашиваемого отрезка го-
: ризонтальной прямой (см. п.44)

Адрес : PNTWRKA+6=F680H

Размер : 1 байт

46. Имя : LEFTT

Назначение: Шаблон левой части закрашиваемого отрезка
: горизонтальной прямой (без "фона")

Адрес : PNTWRKA+7=F681H

Размер : 1 байт

47. Имя : RIGHTT

Назначение: Шаблон правой части закрашиваемого отрезка
: горизонтальной прямой (без "фона")

Адрес : PNTWRKA+8=F682H
Размер : 1 байт

48. Имя : LEFTM

Назначение: Маска "фона" левой части закрашиваемого от-
резка горизонтальной прямой

Адрес : PNTWRKA+9=F683H
Размер : 1 байт

49. Имя : RIGHTM

Назначение: Маска "фона" правой части закрашиваемого от-
резка горизонтальной прямой

Адрес : PNTWRKA+10=F684H
Размер : 1 байт

50. Имя : SAVAREA

Назначение: Область, предназначенная для сохранения пара-
метров закрашиваемого отрезка горизонтальной
прямой (Y-координата, начальная X-координата,
конечная X-координата)

Адрес : PNTWRKA+11=F685H
Размер : 5 байт

51. Имя : DRAWRK

Назначение: Область содержит параметры вывода на графиче-
ский экран.

Адрес : DEFTBL=F6CAH
Размер : 15 байт

52. Имя : DISPSW

Назначение: Переключатель вывода точки на графический
экран.

: DISPSW = 0 - вывод точки разрешен
: <>0 - вывод точки запрещен.

Адрес : DRAWRK=F6CAH
Размер : 1 байт

53. Имя : SCRACX

Назначение: X-координата графического экрана, соответст-
: вующего текущей позиции графического курсора.
: Если SCRACX <> 0, то точка не выводится

Адрес : DRAWRK+1=F6CBH

Размер : 2 байта

54. Имя : SCRACY

Назначение: Y-координата графического экрана, соответст-
: вующего текущей позиции графического курсора.
: Если SCRACY <> 0, то точка не выводится

Адрес : DRAWRK+3=F6CDH

Размер : 2 байта

55. Имя : DASHSW

Назначение: Маска линии, управляющая выводом линии задан-
: ного типа. Точка выводится на экран только
: тогда, когда старший бит маски равен нулю,
: после чего выполняется циклический сдвиг
: маски влево на один бит.

Адрес : DRAWRK+5=F6CFH

Размер : 1 байт

56. Имя : WIDTHSW

Назначение: Переключатель толщины линии.
: WIDTHSW = 0 - толщина линии равна единице
: <>0 - толщина линии больше единицы

Адрес : DRAWRK+6=F6D0H

Размер : 1 байт

57. Имя : WIDTHLN

Назначение: WIDTHLN - толщина линии по горизонтали
: WIDTHLN+1 - толщина линии по вертикали

Адрес : DRAWRK+7=F6D1H

Размер : 2 байта

58. Имя : MARKSW

Назначение: Переключатель толщины линии.

: MARKSW = 0 - обычное рисование

: <>0 - рисование маркером

Адрес : DRAWRK+9=F6D3H

Размер : 1 байт

59. Имя : MARKPTR

Назначение: Адрес, по которому размещается шаблон маркера

Адрес : DRAWRK+10=F6D4H

Размер : 2 байта

60. Имя : SRCPAGE

Назначение: текущий номер страницы-источника

Адрес : DRAWRK+12=F6D6H

Размер : 1 байт

61. Имя : DSTPAGE

Назначение: текущий номер страницы-цели

Адрес : DRAWRK+13=F6D7H

Размер : 1 байт

62. Имя : BANKSW

Назначение: переключатель VRAM <--> расширенный RAM

Адрес : DRAWRK+14=F6D8H

Размер : 1 байт

63. Имя : XJUMP

Назначение: CLOC-координата точки экрана (255,0).

Адрес : TEMP=F6A7H

Размер : 2 байта

64. Имя : YJUMP

Назначение: CLOC-координата точки экрана (0,191).

Адрес : TEMP2=F6BCH

Размер : 2 байта

65. Имя : REGSAVA

Назначение: область сохранения управляющих регистров

: R#0 - R#7 видеопроцессора

Адрес : F3DFH

Размер : 8 байт

66. Имя : REGSAVB

Назначение: область сохранения управляющих регистров

: R#8 - R#23 видеопроцессора

Адрес : FFE7H

Размер : 16 байт

67. Имя : SPRPAT

Назначение: Область размещения данных, формируемых проце-

: дурами SET8, SET16, SET32 пакета процедур ди-

: намической графики (Паскаль).

Адрес : PRMSTK+72=F72CH

Размер : 32 байта

68. Имя : SECSEL

Назначение: регистр переключения слотов

Адрес : FFFFH

Размер : 1 байт

ПРИЛОЖЕНИЕ D

1. ТАБЛИЦА РАСШИРЕННЫХ ПЕРЕХОДОВ T.SERV

Таблица расширенных переходов T.SERV размещается в области системных переменных по адресу F74CH и занимает 24 байта (8 точек входа).

1.1. Точка входа T.SIZEX

Адрес : F74CH
Установка : SCREEN
Использование: SCURVE,DASH
Назначение : определение текущей ширины экрана

1.2. Точка входа T.SIZEY

Адрес : F74FH
Установка : SCREEN
Использование: SCURVE,DASH,SPRAY
Назначение : определение текущей высоты экрана

1.3. Точка входа T.SCALE

Адрес : F752H
Установка : SCREEN
Использование: SCURVE,DASH
Назначение : вычисление локальной координаты по
: глобальной

1.4. Точка входа T.VCOMP

Адрес : F755H
Установка : SCREEN
Использование: SHABLON,VCOMP
Назначение : сравнение блока памяти с блоком видео-
: памяти

1.5. Точка входа T.VFILL

Адрес : F758H

Установка : SCREEN

Использование: VFILL,WFILL

Назначение : заполнение блока видеопамати заданным
байтом

1.6. Точка входа T.VGET

Адрес : F75BH

Установка : SCREEN

Использование: VGET, WSAVE

Назначение : пересылка VRAM --> RAM

1.7. Точка входа T.VPUT

Адрес : F75EH

Установка : SCREEN

Использование: SHABLON, SETPAT, SETSPC, VPUT, WPUT

Назначение : пересылка RAM --> VRAM

1.8. Точка входа T.VCOPY

Адрес : F761H

Установка : SCREEN

Использование: VCOPY, WCOPY

Назначение : пересылка VRAM --> VRAM

2. ТАБЛИЦА РАСШИРЕННЫХ ПЕРЕХОДОВ T.CURVE

Таблица расширенных переходов T.CURVE размещается в области системных переменных по адресу F764H и занимает 36 байт (12 точек входа).

2.1. Точка входа T.SCALXY

Адрес : F764H
Установка : SCURVE
Использование: SCURVE, DASH, MAPXY, SECTOR, SEGMENT
Назначение : определение локальных координат по
: глобальным

2.2. Точка входа T.UMOVE

Адрес : F767H
Установка : SCURVE
Использование: SCURVE, MOVE
Назначение : перемещение графического курсора на
: одну позицию вверх

2.3. Точка входа T.RMOVE

Адрес : F76AH
Установка : SCURVE
Использование: SCURVE, MOVE
Назначение : перемещение графического курсора на
: одну позицию вправо

2.4. Точка входа T.DMOVE

Адрес : F76DH
Установка : SCURVE
Использование: SCURVE, MOVE
Назначение : перемещение графического курсора на
: одну позицию вниз

2.5. Точка входа T.LMOVE

Адрес : F770H

Установка : SCURVE

Использование: SCURVE, MOVE

Назначение : перемещение графического курсора на
: одну позицию влево

2.6. Точка входа T.RPCHK

Адрес : F773H

Установка : SCURVE

Использование: SCURVE, RPSET, PSET, CIRCLE, ELLIPS, ARC,
: SECTOR, SEGMENT

Назначение : обработка условий вывода точки в текущую
: позицию графического курсора

2.7. Точка входа T.PSET

Адрес : F776H

Установка : SCURVE

Использование: PSET, CIRCLE, ELLIPS, ARC, SECTOR, SEGMENT

Назначение : определение локальных координат по
: глобальным при выводе точки на экран

2.8. Точка входа T.LINE

Адрес : F779H

Установка : SCURVE

Использование: SCURVE, LINE, SECTOR, SEGMENT

Назначение : подключение своей процедуры рисования
: отрезка прямой

2.9. Точка входа T.BOX

Адрес : F77CH

Установка : SCURVE

Использование: BOX

Назначение : подключение своей процедуры рисования
: прямоугольника

2.10.Точка входа T.CIRCLE

Адрес : F77FH

Установка : SCURVE

Использование: CIRCLE, ELLIPS, ARC, SECTOR, SEGMENT

Назначение : подключение своей процедуры рисования
: окружности, эллипса, дуги

2.11.Точка входа T.RPWDT

Адрес : F782H

Установка : DASH

Использование: SCURVE, RPSET, PSET, CIRCLE, ELLIPS, ARC,
: SECTOR, SEGMENT

Назначение : обработка условий вывода точки при рисо-
: вании толстых линий

2.12.Точка входа T.RMSET

Адрес : F785H

Установка : MRKON

Использование: DASH

Назначение : вывод маркера в текущую позицию графиче-
: ского курсора

3. ТАБЛИЦА РАСШИРЕННЫХ ПЕРЕХОДОВ T.FILL

Таблица расширенных переходов T.FILL размещается в области системных переменных по адресу F788H и занимает 27 байт (9 точек входа).

3.1. Точка входа T.RLCMEM

Адрес : F788H
Установка : SHABLON
Использование: LINEX
Назначение : циклический сдвиг влево заданного участка памяти

3.2. Точка входа T.TRIM

Адрес : F78BH
Установка : SHABLON
Использование: LINEX
Назначение : подгонка шаблона закрашки под текущую X-координату

3.3. Точка входа T.VLOC

Адрес : F78EH
Установка : SHABLON
Использование: LINEX
Назначение : вычисление адреса точки в видеопамати

3.4. Точка входа T.LCHECK

Адрес : F791H
Установка : SHABLON
Использование: LINEX
Назначение : определение размеров левого сегмента закрашиваемого отрезка горизонтальной прямой

3.5. Точка входа T.RCHECK

Адрес : F794H

Установка : SHABLON

Использование: LINEX

Назначение : определение размеров правого сегмента
: закрашиваемого отрезка горизонтальной
: прямой

3.6. Точка входа T.LTAIL

Адрес : F797H

Установка : SHABLON

Использование: LINEX

Назначение : формирование левого сегмента закрашивае-
: мого отрезка горизонтальной прямой

3.7. Точка входа T.RTAIL

Адрес : F79AH

Установка : SHABLON

Использование: LINEX

Назначение : формирование правого сегмента закрашива-
: емого отрезка горизонтальной прямой

3.8. Точка входа T.LINEX

Адрес : F79DH

Установка : SHABLON

Использование: SPRAY

Назначение : вывод на экран отрезка горизонтальной
: прямой

3.9. Точка входа T.BOUND

Адрес : F7A0H

Установка : SHABLON

Использование: SPRAY

Назначение : поиск по горизонтали границы закрашива-
: емой области

ПРИЛОЖЕНИЕ Е

ПРИМЕР ПРОГРАММЫ РИСОВАНИЯ ЛИНИЯМИ РАЗЛИЧНОГО ТИПА

PROGRAM CURVES;

VAR ADDRESS: INTEGER;

{ \$F KEY2.PAS }

{ \$F SBG2.PAS }

{ \$F SCG2.PAS }

FUNCTION SETMEM(length: INTEGER);

BEGIN

 INLINE(

 33,228,246,62,225,119,35,62,229,119,35,62,201,119,205,228,
 246,17,19,0,25,0,221,117,4,221,116,5,221,94,2,221,
 86,3,25,233);

 { Шаблон маркера }

 INLINE(34,17,136,255,

 34,17,136,255,

 34,17,136,255,

 34,17,136,255,

 34,17,136,255,

 34,17,136,255,

 34,17,136,255,

 34,17,136,255)

END;

BEGIN

 { Установка экранного режима }

 SCREEN(7);

 { Установка расширенного режима рисования }

 SCURVE(1);

 { Установка типа рисуемой линии и ее толщины }

 { Линия, нарисованная маркером размером 4*8 точек }

 DASH(0,4,8);

 { Формирование шаблона маркера и определение его адреса }

 ADDRESS:=SETMEM(32);

 { Включение режима рисования маркером }

 MRKON(ADDRESS);

 { Рисование }

 BOX (60,60,450,150,11,FALSE);

 { Выключение режима рисования маркером }

 MRKOFF;

```

{ Установка типа рисуемой линии и ее толщины }
{ Сплошная линия, нарисованная прямоугольником
  размером 4*8 точек }

DASH(0,4,8);

{ Рисование }
BOX (10,10,500,200,8,FALSE);

{ Установка типа рисуемой линии и ее толщины }
{ Пунктирная линия }

DASH(1,0,0);

{ Рисование }
BOX (20,20,490,190,1,FALSE);

{ Установка типа рисуемой линии и ее толщины }
{ Штриховая линия }
DASH(2,0,0);

{ Рисование }
BOX (30,30,480,180,2,FALSE);

{ Установка типа рисуемой линии и ее толщины }
{ Штрих-пунктирная линия }
DASH(3,0,0);

{ Рисование }
BOX (40,40,470,170,11,FALSE);
{ Переход в базовый режим рисования }

SCURVE(0);
{ Рисование }

BOX (80,80,430,130,9,FALSE);

REPEAT UNTIL STRIG(0);

{ Переход в текстовый режим }
SCREEN(0)
END.

```

ПРИЛОЖЕНИЕ F

ПРИМЕР ПРОГРАММЫ ЗАКРАСКИ ЗАМКНУТОЙ ОБЛАСТИ ШАБЛОНОМ

PROGRAM FILLS;

VAR ADDRESS: INTEGER;

{ \$F KEY2.PAS }

{ \$F SBG2.PAS }

{ \$F SFG2.PAS }

FUNCTION SETMEM(length: INTEGER);

BEGIN

 INLINE(

 33,228.246.62,225,119,35.62,229,119,35.62,201,119.205.228.
 246,17,19,0.25,0,221,117,4,221,116,5,221,94.2.221.
 86,3,25,233);

 { Шаблон закрашки }

 INLINE(34,17,136,255,34,17,136,255,
 34,17,136,255,34,17,136,255,
 34,17,136,255,34,17,136,255,
 34,17,136,255,34,17,136,255,
 34,17,136,255,34,17,136,255,
 34,17,136,255,34,17,136,255,
 34,17,136,255,34,17,136,255)

END;

BEGIN

 { Установка экранного режима }

 SCREEN(7);

 { Формирование шаблона закрашки и определение его адреса }

 ADDRESS:=SETMEM(32);

 { Установка шаблона закрашки }

 SHABLON(ADDRESS);

 { Рисование замкнутой области }

 ELLIPS(250,100.50,100,9);

 CIRCLE(250,100.200,9);

```
{ Закраска области шаблоном }  
SPRAY(100,100,9);
```

```
REPEAT UNTIL STRIG(0);
```

```
{ Переход в текстовый режим }  
SCREEN(0)  
END.
```

ПРИЛОЖЕНИЕ G

ДЕМОНСТРАЦИОННАЯ ПРОГРАММА TUTOR

Программа предназначена для демонстрации возможностей пакета LOGO-PAS. Поставляется вместе с исходным текстом.

Ниже приводится описание работы программы.

При запуске в правой части экрана инициализируется графическое окно, слева высвечиваются имена основных процедур пакета LOGO-PAS и строка MACRO, внизу справа - координатная строка.

Выбор команды осуществляется с помощью стрелок "вверх" и "вниз". Изменение параметров - с помощью клавиш "влево" и "вправо". Если, изменяя параметры, удерживать клавишу "вверх", то приращение увеличивается в 10 раз. Исполнение команды осуществляется по клавише "возврат каретки". В строке MACRO можно задать количество повторений.

Вход в режим редактирования макрокоманды осуществляется по клавише "INS". Выход из режима - по клавише "ESC".

Чтобы выйти из программы, нажмите одновременно клавиши <ESC> и <ввод>.

Пример использования макрокоманды:

1. Нажмите клавишу <INS> .Строка MACRO пожелтеет.
 2. Подведите белую черепаху к строке с надписью FD
 3. Установите значение FD 20
 4. Нажмите <ввод>
 5. Установите появившийся курсор на первую строку поля макрокоманды.
 6. Нажмите <ввод> Строка FD 2 появится напротив курсора
 7. Подведите белую черепаху к строке с надписью RT
 8. Установите значение RT 60
 9. Нажмите <ввод>
 10. Установите появившийся курсор на вторую строку поля макрокоманды.
 11. Нажмите <ввод>
 12. Выйдите из режима редактирования макрокоманды, нажав <ESC>
 13. Установите белую черепаху напротив строки MACRO
 14. Установите коэффициент повторения MACRO 6
 15. Нажмите <ввод>
- и вторая черепаха нарисует правильный шестиугольник.

Под управлением MSX-DOS программа запускается обычным образом:

A>TUTOR

IV ИНСТРУМЕНТАЛЬНЫЕ ПРОГРАММНЫЕ СРЕДСТВА

**ДЛЯ КОМПИЛЯТОРА
ASCII C И АССЕМБЛЕРА M80**

ВВЕДЕНИЕ.

Предлагаемые программные средства рассчитаны на широкий круг пользователей – программистов, занятых разработкой инструментальных средств, и прикладных программистов, занятых написанием обучающих, игровых и др. программ, использующих в своей работе язык Си и Ассемблер.

Пакеты передаются в виде библиотечных REL-файлов, содержащих объектные коды процедур.

Для подключения процедур пакетов пользователю достаточно:
при работе с Си:

- вставить в начале текста программы строку
#include "имя файла деклараций" (H-файла)
- на этапе редактирования (сборки) вставить в текст задания редактору связей L80 имена нужных библиотек с постфиксом "/s".

при работе с Ассемблером:

- объявить в тексте программы имена необходимых процедур как extrn.
- создать объектный перемещаемый код программы с помощью M80
- на этапе редактирования (сборки) вставить в текст задания редактору связей L80 имена нужных библиотек с постфиксом "/s".
- на этапе редактирования (сборки) вставить в текст задания редактору связей L80 имена нужных библиотек с постфиксом "/s".

В состав передаваемых программных средств входят следующие пакеты:

"Графический пакет"	(библиотека GRP2.REL, файл деклараций GRP2.H) ,
"Пакет процедур для работы со спрайтами"	(библиотека SPR2.REL, файл деклараций SPR2.H) ,
"Оконный ввод-вывод"	(библиотека WND2.REL, файл деклараций WND2.H) ,
"Пакет системных процедур"	(библиотека SYS2.REL, файл деклараций SYS2.H) ,
"Клавиатура"	(библиотека KEY2.REL, файл деклараций KEY2.H) ,
"Музыкальный пакет"	(файл MUS2.REL, файл деклараций MUS2.H, программа-конвертор MUS.COM) ,
"Математический пакет"	(библиотека MATH2.REL, файл деклараций MATH2.H) .

При редактировании программы необходимо учесть, что подключая библиотеки GRP2 и SPR2, необходимо также подключать библиотеку SYS2, а подключая библиотеку WND2, необходимо подключить также библиотеки SYS2 и MATH2.

1

МАТЕМАТИЧЕСКИЙ ПАКЕТ

1.1. НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ ПАКЕТА

Математический пакет делает возможным написание на языке Си программ, содержащих различные по сложности математические расчеты. С помощью процедур пакета можно

- Выполнять арифметические действия с вещественными числами с точностью 14 знаков.
- Вычислять значения квадратного корня, тригонометрических функций, показательной функции и логарифма
- Генерировать последовательности псевдослучайных чисел
- Организовывать асинхронную обработку ситуаций переполнения точности, деления на ноль и выхода за границу области определения функции.

Процедуры пакета можно также использовать при работе с Ассемблером, допускающим отдельную компиляцию.

1.2. ОПИСАНИЕ ФУНКЦИИ ПАКЕТА

Под вещественным числом далее понимается восемь байт RAM, первый из которых определяет характеристику, а остальные - мантиссу вещественного числа (по 2 десятичные цифры на байт).

В состав пакета входят следующие функции:

- ADD - сложение двух вещественных
- PROD - произведение двух вещественных
- SUB - вычитание двух вещественных
- DIV - деление двух вещественных
- GTF - проверка двух вещественных на больше
- LTF - проверка двух вещественных на меньше
- GEF - проверка двух вещественных на больше или равно
- LEF - проверка двух вещественных на меньше или равно

- EQF - проверка двух вещественных на равенство
- EQUALF - копирование вещественных
- ITOF - преобразование целого в вещественное
- ROUND - округление вещественного до целого
- ATOF - преобразование строки знаков в вещественное
 число
- FTOA - преобразование вещественного числа в
 строку знаков
- DISPF - вывод вещественного числа
- GETF - ввод вещественного числа
- DEGF - порядок вещественного числа
- LENF - количество значащих цифр вещественного числа
- ZEROF - очистка вещественного числа
- COS - косинус
- SIN - синус
- TG - тангенс
- ARCTG - арктангенс
- SQRT - квадратный корень
- LN - натуральный логарифм
- EXP - показательная функция (по основанию e)
- RND - случайное число
- GTORAD - перевод из градусной меры в радианную
- RADTOG - перевод из радианной меры в градусную
- ON_ERR - задание имени процедуры обработки ошибок
- ERROFF - блокировка системы обработки ошибок
- ERRON - активизация системы обработки ошибок

Кроме того, объявлены как глобальные и могут быть использованы в программах пользователя следующие константы:

ONE	- 1.00000000000000		
HALF	- 0.50000000000000		NB! При обращении из
PI	- π		Ассемблера к этим име-
HPI	- $\pi/2$		нам следует приписать
QPI	- $\pi/4$		справа символ "@".

Функции арифметики, а также функции SQRT и LN вырабатывают код ошибки, который можно узнать, обратившись к глобальной переменной error. При нормальном завершении значение этой переменной всегда равно 0.

В Си-программе пользователя вещественные переменные объявляются при помощи описателя FLOAT (большими буквами), а имена функций при обращении записываются строчными буквами.

Ниже приводится краткое описание назначения, входных и выходных параметров доступных пользователю функций пакета.

1.2.1. Функция ADD

Функция ADD складывает значения переменных, адреса которых заданы вторым и третьим параметрами, результат помещает в по адресу, заданному первым параметром.

Вызов функции : ADD(res,f1,f2);

Параметры : FLOAT res,f1,f2 - результат, первое
: слагаемое, второе слагаемое.

Результат : NAT - адрес результата

Точка входа : ADD@

1.2.2. Функция PROD

Функция PROD умножает значения переменных, адреса которых заданы вторым и третьим параметрами, результат помещает в по адресу, заданному первым параметром.

Вызов функции : PROD(res,f1,f2);

Параметры : FLOAT res,f1,f2 - результат, первое
: слагаемое, второе слагаемое.

Результат : NAT - адрес результата

Точка входа : PROD@

1.2.3. Функция SUB

Функция SUB вычитает значения переменных, адреса которых заданы вторым и третьим параметрами, результат помещает в по адресу, заданному первым параметром.

Вызов функции : SUB(res, f1, f2);

Параметры : FLOAT res, f1, f2 - результат, первое
: слагаемое, второе слагаемое.

Результат : NAT - адрес результата

Точка входа : SUB@

1.2.4. Функция DIV

Функция DIV делит значения переменных, адреса которых заданы вторым и третьим параметрами, результат помещает в по адресу, заданному первым параметром.

Вызов функции : DIV(res, f1, f2);

Параметры : FLOAT res, f1, f2 - результат, первое
: слагаемое, второе слагаемое.

Результат : NAT - адрес результата

Точка входа : DIV@

1.2.5. Функции GTF, LTF, GEF, LEF, EQF предназначены для сравнения значений вещественных переменных.

Вызов всех этих функций осуществляется единообразно:

... (f1, f2)

Параметры : FLOAT f1, f2

:

Результат : BOOL - TRUE, если выполнено условие, соответствующее имени функции, иначе FALSE.

Точки входа

соответственно: GTF@, LTF@, GEF@, LEF@, EQF@

1.2.6. Функция EQUALF

Функция EQUALF копирует значение переменной, определенной вторым параметром, по адресу переменной, определенной первым операндом.

Вызов функции : EQUALF(dest,source)

Параметры : FLOAT dest,source - результат,источник
:

Результат : NAT - адрес результата

Точка входа : EQUALF

1.2.7. Функция ITOF

Функция ITOF преобразует значение целой переменной, определенной первым операндом, в вещественный формат и помещает результат по адресу переменной, определенной вторым операндом

Вызов функции : ITOF(source,dest)

Параметры : int source -источник
: FLOAT dest -результат
:

Результат : NAT - адрес результата

Точка входа : ITOF@

1.2.8. Функция ROUND

Функция ROUND возвращает значение вещественной переменной, округленное до ближайшего целого.

Вызов функции : ROUND(f)

Параметры : FLOAT f;
:

Результат : int

Точка входа : ROUND@

1.2.9. Функция ATOF

Функция ATOF преобразует символьную строку, являющуюся записью некоторого вещественного числа в формате с десятичной точкой в формат вещественной переменной и помещает результат по адресу переменной, определенной вторым операндом. Адрес строки задается первым операндом.

Вызов функции : ATOF(str,f)

Параметры : char str[] -исходная строка
: FLOAT f -результат
:

Результат : NAT - адрес результата

Точка входа : ATOF@

1.2.10. Функция FTOA

Функция ATOF выполняет преобразование вещественной переменной, заданной первым операндом в строку символов, адрес которой задан вторым операндом.

Вызов функции : FTOA(f,str)

Параметры : FLOAT f -вещественная переменная
: char str[] -результатирующая строка
:

Результат : VOID - фиктивный

Точка входа : FTOA@

1.2.11. Функция DISPF

Функция DISPF выводит на экран значение вещественной переменной.

Вызов функции : DISPF(f)

Параметры : FLOAT f;
:
Результат : VOID - фиктивный
Точка входа : DISPF@

1.2.12. Функция GETF

Функция GETF вводит с экрана значение вещественной переменной

Вызов функции : GETF(f)

Параметры : FLOAT f;
:

Результат : NAT - адрес результата

Точка входа : GETF@

1.2.13. Функция DEGF

Функция DEGF вычисляет значение порядка вещественной переменной.

Вызов функции : DEGF(f)

Параметры : FLOAT f;

Результат : int

Точка входа : DEGF@

1.2.14. Функция LENF

Функция LENF вычисляет количество значащих цифр вещественной переменной

Вызов функции : LENF(f)

Параметры : FLOAT f;
:

Результат : int

Точка входа : LENF@

1.2.15. Функция ZEROF

Функция ZEROF присваивает вещественной переменной нулевое значение.

Вызов функции : ZEROF(f)

Параметры : FLOAT f;

:

Результат : NAT - адрес переменной

Точка входа : ZEROF@

1.2.16. Функции SIN, COS, ARCTG, SQRT, EXP, LN, RND служат для вычисления синуса, косинуса, арктангенса, квадратного корня, экспоненты и натурального логарифма, случайных чисел.

Вызов осуществляется единообразно:

...(res, arg, type)

Параметры : FLOAT res - результат

: FLOAT или int arg - аргумент

: TINY type - тип аргумента

(1 - FLOAT, 0 - int)

Результат : NAT -адрес результата

Точки входа : SIN@, COS@, ARCTG@, SQRT@, EXP@, LN@, RND@

1.2.17. Функция GTORAD

Функция GTORAD позволяет вычислить величину угла в радианах, если известна его величина в градусах.

Вызов функции : GTORAD(source, dest)

Параметры : int source -источник

: FLOAT dest -результат

:

Результат : NAT - адрес результата

Точка входа : GTORAD@

1.2.18. Функция RADTOG

Функция RADTOG позволяет вычислить величину угла в градусах, по его величине в радианах

Вызов функции : RADTOG(source)

Параметры : FLOAT source -источник
:

Результат : int - величина в градусах

Точка входа : GTORAD@

1.2.19. Функция ON_ERR

Функция ON_ERR позволяет пользователю асинхронно выполнять обработку ошибок при помощи собственной процедуры обработки. Процедуре ON_ERR передается в качестве параметра адрес процедуры пользователя, вызов которой будет происходить всякий раз при возникновении ошибки времени выполнения.

Код ошибки пользователь узнает из глобальной переменной error: 1 - выход за границу области определения (аналогично бейсиковскому "Illegal function call"), 2 - переполнение; 3 - деление на 0.

Вызов функции : ON_ERR(entry)

Параметры : NAT entry -адрес (имя) процедуры
: обработки ошибок

Результат : VOID - фиктивный

Точка входа : ON@ERR

1.2.20. Функция ERROFF

Функция ERROFF временно отменяет действие процедуры

обработки ошибок.

Вызов функции : ERROFF ()

Параметры : нет

Результат : VOID - фиктивный

Точка входа : ERROFF

2.21. Функция ERRON

Функция ERRON возобновляет действие процедуры обработки ошибок

Вызов функции : ERRON ()

Параметры : нет

Результат : VOID - фиктивный

Точка входа : ERRON@

1.3. ПРИМЕР ИСПОЛЬЗОВАНИЯ ФУНКЦИИ ПАКЕТА

Следует обратить внимание на то, что вспомогательным результатом функций пакета является адрес результирующего операнда. Поэтому для экономии места можно объединять обращения к функциям пакета в цепочки.

Пример: оператор $d:=(a+b)*c$ можно реализовать двумя способами:

- a) `add(d,a,b);`
 `prod(d,d,c);`
- b) `prod(d,add(d,a,b),c);`

Для присвоения начальных значений переменным можно использовать функцию `atof`:

`atof("2.87",a)` эквивалентно `a:=2.87`.

Однако, не следует забывать, что функция `atof` работает довольно медленно. Поэтому, если быстроедействие актуально, рекомендуется инициализировать вещественные переменные с помощью стандартных средств Си. При этом вещественные переменные можно рассматривать как обычный массив `char[8]`; Первый байт - это порядок числа+0x40. Если число отрицательное, то нужно прибавить еще 0x80.

В остальных семи байтах записано по две цифры мантиссы.

Итак, наше число 2.87 записывается следующим образом:

```
Float a={(char)0x41,(char)0x28,(char)0x70,'\0','\0','\0',
'\0','\0'};
```

2

ГРАФИЧЕСКИЕ ФУНКЦИИ

2.1. НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ ПАКЕТА

Включенные в данный графический пакет функции условно можно разбить на три группы:

- базовые графические функции;
- функции для установки режимов рисования и закраски;
- вспомогательные функции.

Функции первой группы во многом аналогичны графическим операторам интерпретатора языка Basic.

Отличия функций первой группы от соответствующих графических операторов интерпретатора языка Basic заключаются в следующем:

- функция SCREEN в отличие от ее аналога в интерпретаторе языка Basic инициализирует все таблицы видеопроцессора
- в случае, если необходимо выполнить выборочную инициализацию таблиц видеопроцессора, может быть использована функция CLS; тем самым возможности этой функции несколько шире возможностей аналогичного оператора интерпретатора языка Basic
- в функциях ELLIPS и ARC, в отличие от соответствующего им оператора интерпретатора языка Basic (CIRCLE), указываются горизонтальная и вертикальная полуоси эллипса, что, по нашему мнению, более удобно
- в функции ARC углы задаются в градусах, что позволяет избежать использования в программе функций плавающей арифметики

Наряду с традиционными, в первую группу включены дополнительные графические функции, позволяющие

- устанавливать и перемещать графический курсор (MAPXY, MOVE)

- выводить точку в текущую позицию графического курсора
(RPSET)
- рисовать сектора и сегменты
(SECTOR, SEGMENT)
- закрашивать замкнутую область указанным шаблоном
(SPRAY)

К функциям второй группы относятся функции для установки различных режимов рисования, а также параметров этих режимов.

При рисовании пакет допускает работу в двух режимах - базовом и расширенном. В базовом режиме рисование выполняется также, как и в Basic'e. Если же пользователь перейдет в расширенный режим, в его распоряжении окажутся некоторые дополнительные возможности.

Так, включив расширенный режим рисования (функция SCURVE) и установив тип линий и их толщину (функция DASH), Вы сможете с помощью базовых графических функций рисовать:

- пунктирные линии;
- штриховые линии;
- штрих пунктирные линии;
- линии различной толщины.

Если же, находясь в расширенном режиме, Вы установите "маркер" (функция MRKON), то с помощью тех же функций сможете рисовать линии этим "маркером".

Установив шаблон закрашки (функция SHABLON) и воспользовавшись функцией SPRAY, Вы сможете закрашивать замкнутые области этим шаблоном.

Внимание !! Включение в шаблон точек, цвет которых совпадает с цветом границы закрашиваемой области не рекомендуется.

Вызвав повторно функцию DASH или функцию SHABLON, пользователь может изменить вид линии или "шаблон" закрашки. Пользователь в любой момент может вернуться в базовый режим.

Наконец, к функциям третьей группы относятся функции, используемые функциями первых двух групп, но пользователю "недоступные". Последнее означает, что при вызове этих функций принятое в языке Си соглашение о связях не соблюдается.

2.2. ОПИСАНИЕ ФУНКЦИИ ПАКЕТА

В состав пакета входят следующие функции:

- LOCATE - установка курсора в текстовых режимах;
- SETPAGE - установка отображаемой и активной страниц видеопамяти;
- LOGOP - установка логической операции;
- VPEEK - чтение байта из видеопамяти;
- VPOKE - запись байта в видеопамять;
- COLOR - установка фасадного, фонового и бордюрного цветов;
- SCREEN - установка экранного режима;
- CLS - выборочная инициализация таблиц видеопроцессора;
- SCURVE - установка режима рисования линий;
- DASH - установка типа линий и их толщины;
- SHABLON - задание шаблона закрашки;
- MRKON - включение режима рисования с помощью маркера;
- MRKOFF - выключение режима рисования с помощью маркера;
- MAPXY - установка графического курсора;
- MOVE - перемещение графического курсора на один пиксел в указанном направлении;
- RPSET - рисование точки в текущей позиции графического курсора;
- POINTC - определение цвета указанной точки экрана;
- PSET - рисование точки;
- LINE - рисование отрезка прямой;
- BOX - рисование прямоугольника
(закрашенного или незакрашенного):

- CIRCLE - рисование окружности;
- ELLIPS - рисование эллипса;
- ARC - рисование дуги эллипса;
- SECTOR - рисование сектора эллипса;
- SEGMENT - рисование сегмента эллипса;
- PAINT - закрашка замкнутой области;
- SPRAY - закрашка замкнутой области текущим шаблоном;
- _ABSFN - вычисление знака и абсолютного значения разности двух целых чисел;
- _SIZEX - определение ширины экрана в пикселах;
- _SIZEY - определение высоты экрана в пикселах;
- _SCALE - вычисление локальной координаты;
- _SCALXY - вычисление "локальных" координат и значений системных переменных SCRACX и SCRACY;
- _UMOVE - перемещение графического курсора на один пиксел вверх;
- _RMOVE - перемещение графического курсора на один пиксел вправо;
- _DMOVE - перемещение графического курсора на один пиксел вниз;
- _LMOVE - перемещение графического курсора на один пиксел влево;
- _GMOVE - перемещение графического курсора на один пиксел в указанном направлении;
- _ASPECT - вычисление значения системной переменной ASPECT (ASPECT RATIO);
- _ANGLE - расчет угловых параметров дуги;
- _CNULL - рисование вырожденной окружности ($R=0$);
- _CHOOK - установка хука, модифицирующего программу обращения к BASIC-интерпретатору при рисовании окружности;
- _RPCHK - управление выводом точки на экран;
- _RMSET - вывод маркера на экран;
- _RPWDT - управление толщиной линии;

- `_LINEX` - закрашивание горизонтального отрезка текущим шаблоном;
- `_BOUND` - поиск границы закрашиваемой области;
- `_PNTLIN` - закрашка текущего горизонтального участка замкнутой области;
- `_HPSET` - вычисление локальных координат графического курсора в расширенном режиме.;
- `_HLINE` - рисование отрезка прямой в расширенном режиме;
- `_HBOX` - рисование прямоугольника в расширенном режиме;
- `_HCIRCL` - рисование окружности (эллипса, дуги эллипса) в расширенном режиме.

Все функции с префиксом "_" являются вспомогательными. Пользователю доступны только функции `LOCATE`, `SETPAGE`, `LOGOP`, `VPEEK`, `VPOKE`, `COLOR`, `SCREEN`, `CLS`, `SCURVE`, `DASH`, `SHABLON`, `MRKON`, `MRKOFF`, `MAPXY`, `MOVE`, `RPSET`, `POINTC`, `PSET`, `LINE`, `BOX`, `CIRCLE`, `ELLIPS`, `ARC`, `SECTOR`, `SEGMENT`, `PAINT`, `SPRAY`.

Ниже приводится краткое описание назначения, входных и выходных параметров доступных пользователю функций пакета.

2.2.1. Функция `LOCATE`

Функция `LOCATE` позиционирует курсор в текстовом экранном режиме (аналогична соответствующему оператору языка Basic).

Вызов функции : `locate(x,y,disp);`

Параметры : `int x,y` - координаты курсора
 : `BOOL disp = TRUE` - курсор отображается
 : `= FALSE` - курсор не отображается
 :

Результат : `VOID` - фиктивный

Точка входа : `locate`

2.2.2. Функция SETPAGE

Функция SETPAGE устанавливает номера отображаемой и активной страниц видеопамати (аналогична соответствующему оператору языка Basic). Процедура работает только в старших экранных режимах (начиная с пятого). При этом в экранных режимах 5 и 6 номера страниц берутся по модулю 4, а в экранных режимах 7 и 8 - по модулю 2.

```

Вызов функции : setpage(display,active):
Параметры      : TINY display - номер отображаемой
                  :                страницы
                  : TINY active  - номер активной страницы
Результат      : VOID          - фиктивный
Точка входа    : setpag

```

2.2.3. Функция LOGOP

Функция LOGOP устанавливает код логической операции, которая должна выполняться над данными при их записи в видеопамять. Эта операция будет выполняться при вызове любой графической функции, осуществляющей вывод данных в видеопамять. Таблица кодов логических операций приведена в приложении А.

Вызов функции : `logop(code)` ;
 Параметры : `TINY code` - код логической операции
 Результат : `VOID` - фиктивный
 Точка входа : `logop@`

2.2.4. Функция VPEEK

Функция VPEEK читает один байт данных из видеопамати.

Вызов функции : vpeek(source);

Параметры : char *source - адрес видеопамати

Результат : char - считанный байт данных

Точка входа : vpeek@

2.2.5. Функция VPOKE

Функция VPOKE записывает один байт данных в видеопамать.

Вызов функции : vpoke(dest, byte);

Параметры : char *dest - адрес видеопамати

: char byte - записываемый байт данных

Результат : VOID - фиктивный

Точка входа : vpoke@

2.2.6. Функция COLOR

Функция COLOR выполняет установку фасадного, фонового и бордюрного цветов. Функция аналогична соответствующему оператору языка Basic.

Вызов функции : color(f, bak, bdr);

Параметры : TINY f - цвет фасада

: TINY bak - цвет фона

: TINY brd - цвет бордюра

Результат : VOID - фиктивный

Точка входа : color@

2.2.7. ФУНКЦИЯ SCREEN

Функция SCREEN осуществляет установку требуемого экранного режима. При этом, в отличие от ее аналога в языке Basic, инициализируются все таблицы видеопроцессора (имен, шаблонов, цветов, атрибутов спрайтов, шаблонов спрайтов, мерцания).

Вызов функции : `screen(n)` ;
 Параметры : `TINY n` – номер экранного режима
 Результат : `VOID` – фиктивный
 Точка входа : `screen`

2.2.8. ФУНКЦИЯ CLS

Функция CLS в зависимости от значений параметров инициализирует различные таблицы видеопроцессора.

Вызов функции	:	<code>cls(grp, spr, pat) :</code>
Параметры	:	<code>BOOL grp = TRUE</code> – инициализируются таблицы
	:	имен, шаблонов, цветов
	:	<code>BOOL spr = TRUE</code> – инициализируется таблица
	:	атрибутов спрайтов
	:	<code>BOOL pat = TRUE</code> – инициализируется таблица
	:	шаблонов спрайтов
Результат	:	<code>VOID</code> фиктивный
Точка входа	:	<code>cls@</code>

2.2.9. Функция SCURVE

Функция SCURVE устанавливает режим рисования линий

2.2.11. Функция SHABLON

Функция SHABLON устанавливает паттерн (шаблон), которым должны выполняться закрашки. Установленный шаблон закрашки будет оставаться неизменным до следующего вызова функции SHABLON. Шаблон, которым выполняется закрашка, представляет собой квадрат размером 8 * 8 точек. Например, чтобы, находясь в седьмом экранном режиме, установить шаблон закрашки, состоящий из чередующихся горизонтальных линий белого (15) и синего (4) цветов, необходимо зарезервировать в памяти 32 байта и занести туда следующие данные: (char)255, (char)68, ..., (char)255, (char)68, после чего вызвать функцию SHABLON, указав в ней адрес этого массива данных.

Вызов функции : shablon(address);

Параметры : TINY *address - адрес массива данных,
: содержащего задаваемый
: шаблон

Результат : VOID - фиктивный

Точка входа : shablo

2.2.12. Функция MRKON

Функция MRKON устанавливает маркер, с помощью которого будет выполняться рисование. Перед тем, как вызвать эту функцию, пользователь должен задать в памяти соответствующий шаблон. Размер шаблона в байтах определяется исходя из текущей толщины по оси X и оси Y по формуле: $w_x * w_y * k$, где w_x и w_y - размеры, устанавливаемые функцией DASH, а $k=1/2$ для экранных режимов 5 и 7, $=1/4$ для экранного режима 6 и $=1$ для экранного режима 8. Режим рисования с помощью маркера включается только в высших экранных режимах (начиная с пятого).

Если толщина хотя бы по одному из направлений равна единице, режим также не включается. Например, если мы хотим задать маркер, имеющий вид квадрата размером 8 * 8 точек с чередующимися зелеными и красными вертикальными полосами, шаблон должен быть размером 32 байта и заполнен кодом (char) (2*16+8).

Внимание !! Перед установкой маркера должна быть обязательно вызвана функция DASH, задающая значения параметров wx и wy.

Вызов функции : mrkon(address);

Параметры : TINY *address - адрес шаблона

Результат : VOID - фиктивный

Точка входа : mrkon@

2.2.13. Функция MRKOFF

Функция MRKOFF отменяет режим рисования с помощью маркера.

Вызов функции : mrkoff();

Параметры : -

Результат : VOID - фиктивный

Точка входа : mrkoff

2.2.14. Функция MAPXY

Функция MAPXY выполняет установку графического курсора по указанным координатам (вычисляет координаты графического курсора CLOC и CMASK). В случае, если координаты указывают на точку за пределами экрана, в качестве последних берутся остатки от деления на SIZEX (ширина экрана в текущем экранном режиме) и SIZEY (высота экрана в текущем экранном режиме) соответственно (вычисляются "локальные" координаты), а частные от деления помещаются соответственно в системные пе-

ременные SCRACX и SCRACY (см. описание системных переменных в приложении С). Например если мы находимся в седьмом экранном режиме, $X=-100$, а $Y=250$, то в качестве "локальных" координат будут взяты числа 412 и 38, а в системные переменные SCRACX и SCRACY будут помещены соответственно числа -1 и 1. В частности, если указанные при вызове функции MAPXY координаты находятся в пределах экрана, то они будут взяты в качестве "локальных" координат, а в системные переменные SCRACX и SCRACY будут помещены нули. После выполнения описанных выше действий функция MAPXY вычисляет по значениям "локальных" координат (а они всегда будут указывать на точку в пределах экрана) координаты графического курсора и помещает их в системные переменные CLOC и CMASK.

Вызов функции : mapxy(x,y);

Параметры : int x,y - координаты

Результат : VOID - фиктивный

Точка входа : mapxy@

2.2.15. Функция MOVE

Функция MOVE перемещает графический курсор на один пиксел в указанном направлении. При переходе через левую или правую границу экрана "локальная" X-координата устанавливается равной SIZEX-1 (где SIZEX - ширина экрана в данном экранном режиме) или 0, а значение системной переменной SCRACX соответственно уменьшается или увеличивается на единицу. При переходе через верхнюю или нижнюю границу экрана "локальная" Y-координата устанавливается равной SIZEY-1 (где SIZEY - высота экрана в текущем экранном режиме) или 0, а значение системной переменной SCRACY соответственно уменьшается или увеличивается на единицу. Таким образом возможно перемещение графического курсора даже если он находится за пределами экрана (значение по крайней мере одной из системных переменных SCRACX или SCRACY отлично от нуля).

Вызов функции : move(dir) :

Параметры : TINY dir = 1 - один пиксел вверх
: = 2 - один пиксел вверх и один
: пиксел вправо
: = 3 - один пиксел вправо
: = 4 - один пиксел вправо и один
: пиксел вниз
: = 5 - один пиксел вниз
: = 6 - один пиксел вниз и один
: пиксел влево
: = 7 - один пиксел влево
: = 8 - один пиксел влево и один
: пиксел вверх
Результат : VOID - фиктивный
Точка входа : move@

2.2.16. Функция RPSET

Функция RPSET выводит точку заданного цвета в текущую позицию графического курсора. Точка выводится на экран если выполнены все перечисленные ниже условия:

- значение системной переменной DISPSW равно нулю;
- вывод точки не "замаскирован" переключателем типа линии (DASHSW) ;
- значения системных переменных SCRACX и SCRACY равны нулю (графический курсор располагается в пределах экрана).

Вызов функции : rpset(c) :

Параметры : TINY c - цвет точки

Результат : VOID - фиктивный

Точка входа : rpset@

2.2.17. Функция POINTC

Функция POINTC возвращает цвет точки экрана с указанными координатами. Если точка находится вне экрана, то возвращается 255.

Вызов функции : `pointc(x,y);`

Параметры : `int x,y` - координаты точки

Результат : `TINY` - цвет указанной точки

Точка входа : `pointc`

2.2.18. Функция PSET

Функция PSET устанавливает цвет указанной точки экрана (выводит точку указанного цвета в указанное место). Точка выводится на экран если выполнены все перечисленные ниже условия:

- значение системной переменной DISPSW равно нулю;
- вывод точки не "замаскирован" переключателем типа линии (`DASHSW`);
- значения системных переменных SCRACX и SCRACY равны нулю (графический курсор располагается в пределах экрана).

Вызов функции : `pset(x,y,c);`

Параметры : `int x,y` - координаты точки

: `TINY c` - цвет точки

Результат : `VOID` - фиктивный

Точка входа : `pset@`

2.2.19. Функция LINE

Функция LINE выводит на экран отрезок прямой указанного цвета по координатам его начала и конца. Если отрезок не по-

мещается целиком на экране, то выводится его видимая часть.

Вызов функции : `line(x1,y1,x2,y2,c);`

Параметры : `int x1,y1` - координаты начальной точки
: `int x2,y2` - координаты конечной точки
: `TINY c` - цвет отрезка

Результат : `VOID` - фиктивный

Точка входа : `line@`

2.2.20. Функция BOX

Функция BOX выводит на экран прямоугольник указанного цвета (закрашенный или незакрашенный) по координатам начала и конца его диагонали. Если прямоугольник не помещается целиком на экране, то выводится только его видимая часть.

Вызов функции : `box(x1,y1,x2,y2,c,flag);`

Параметры : `int x1,y1` - координаты начальной точки
: `int x2,y2` - координаты конечной точки
: `TINY c` - цвет прямоугольника
: `BOOL flag = TRUE` -закрашенный
: `= FALSE` -незакрашенный

Результат : `VOID` - фиктивный

Точка входа : `box@`

2.2.21. Функция CIRCLE

Функция CIRCLE выводит на экран окружность указанного цвета по координатам ее центра и радиусу. Если окружность не помещается целиком на экране, то выводится ее видимая часть

Вызов функции : `circle(x,y,r,c);`

Параметры : `int x,y` - координаты центра окружности
: `int r` - радиус окружности
: `TINY c` - цвет окружности

Результат : VOID - фиктивный
Точка входа : circle

2.2.22. Функция ELLIPS

Функция ELLIPS выводит на экран эллипс указанного цвета по координатам его центра, горизонтальному и вертикальному радиусам. Если эллипс не помещается целиком на экране, то выводится его видимая часть.

Вызов функции : `ellips(x,y,rx,ry,c)`;
Параметры : `int x,y` - координаты центра эллипса
 : `int rx` - горизонтальный радиус
 : `int ry` - вертикальный радиус
 : `TINY c` - цвет эллипса
Результат : VOID - фиктивный
Точка входа : `ellips`

2.2.23. Функция ARC

Функция ARC выводит на экран дугу эллипса указанного цвета по координатам его центра, горизонтальному и вертикальному радиусам, начальному и конечному углам (в градусах). Если дуга не помещается целиком на экране, то выводится ее видимая часть. Если один из углов отрицательный, дуга не выводится. Если начальный угол не превышает конечного, то дуга выводится от начального угла к конечному (против часовой стрелки), в противном случае - от конечного угла к начальному (против часовой стрелки). Если угол превышает 360 градусов, то он берется равным 360.

Вызов функции : `arc(x,y,rx,ry,c,b,e)`;
Параметры : `int x,y` - координаты центра дуги
 : `int rx` - горизонтальный радиус
 : `int ry` - вертикальный радиус

	: TINY c	- цвет дуги
	: int b	- начальный угол дуги
	: int e	- конечный угол дуги
Результат	: VOID	- фиктивный
Точка входа	: arc@	

2.2.24. Функция SECTOR

Функция SECTOR выводит на экран сектор эллипса указанного цвета по координатам его центра, горизонтальному и вертикальному радиусам, начальному и конечному углам (в градусах). Если сектор не помещается целиком на экране, то выводится его видимая часть. Если один из углов отрицательный, сектор не выводится. Если начальный угол не превышает конечного, то сектор выводится от начального угла к конечному (против часовой стрелки), в противном случае - от конечного угла к начальному (против часовой стрелки). Если угол превышает 360 градусов, то он берется равным 360.

Вызов функции : sector(x,y,rx,ry,c,b,e);

Параметры	: int x,y	- координаты центра сектора
	: int rx	- горизонтальный радиус
	: int ry	- вертикальный радиус
	: TINY c	- цвет сектора
	: int b	- начальный угол сектора
	: int e	- конечный угол сектора
Результат	: VOID	- фиктивный
Точка входа	: sector	

2.2.25. Функция SEGMENT

Функция SEGMENT выводит на экран сегмент эллипса указанного цвета по координатам его центра, горизонтальному и вертикальному радиусам, начальному и конечному углам (в граду-

сах). Если сегмент не помещается целиком на экране, то выводится его видимая часть. Если один из углов отрицательный, сегмент не выводится. Если начальный угол не превышает конечного, то сегмент выводится от начального угла к конечному (против часовой стрелки), в противном случае - от конечного угла к начальному (против часовой стрелки). Если угол превышает 360 градусов, то он берется равным 360.

Вызов функции : `segment(x,y,rx,ry,c,b,e)`;

Параметры : `int x,y` - координаты центра сегмента
: `int rx` - горизонтальный радиус
: `int ry` - вертикальный радиус
: `TINY c` - цвет сегмента
: `int b` - начальный угол сегмента
: `int e` - конечный угол сегмента
Результат : `VOID` - фиктивный
Точка входа : `segmen`

2.2.26. Функция PAINT

Функция PAINT закрашивает в указанный цвет область экрана, идентифицируемую координатами одной из ее внутренних точек. Граничной точкой области считается любая точка, имеющая указанный (граничный) цвет. Цвет закрашки и граничный цвет могут при этом не совпадать.

Вызов функции : `paint(x,y,c,cb)`;

Параметры : `int x,y` - координаты внутренней точки
: область
: `TINY c` - цвет закрашки
: `TINY cb` - цвет границы области
Результат : `VOID` - фиктивный
Точка входа : `paint@`

2.2.27. Функция SPRAY

Функция SPRAY выполняет закраску указанной области текущим шаблоном. До вызова функции SPRAY обязательно должна быть выполнена установка шаблона закраски (см. 2.2.11).

Вызов функции : `spray(x,y,cb):`

Параметры : `int x,y` - координаты внутренней точки области

: `TINY cb` - цвет границы области

Результат : `VOID` - фиктивный

Точка входа : `spray@`

3

ДИНАМИЧЕСКАЯ ГРАФИКА

3.1. НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ ПАКЕТА

Включенные в пакет функции позволяют

- формировать шаблоны спрайтов
- устанавливать цвета спрайтов
- выводить спрайты по указанным координатам
- фиксировать ситуации столкновения спрайтов

Пакет допускает работу в двух режимах - базовом и расширенном.

Возможности, имеющиеся в распоряжении пользователя в базовом режиме, аналогичны предоставляемым интерпретатором языка Basic в компьютерах серии MSX-1.

Расширенный режим (в отличие от базового) дает пользователю некоторые дополнительные возможности работы со спрайтами. Основные отличия расширенного режима от базового заключаются в следующем:

- в расширенном режиме каждая строка спрайта может иметь свой цвет;
- в расширенном режиме можно одновременно выводить до восьми спрайтов в строке;
- в расширенном режиме можно задавать выборочную реакцию на столкновения спрайтов.

Более детальное описание возможностей работы со спрайтами в расширенном режиме пользователь может найти в Руководстве по языку Basic для компьютеров серии MSX-2.

При работе в младших экранных режимах (SCREEN 1,2,3) по умолчанию действует базовый режим, в старших экранных режимах (SCREEN 4,5,6,7,8) - расширенный.

2000

4

X 11

20

1992

1152

8

правая верхняя четверть и, наконец, правая нижняя четверть).

Вызов функции : `setpat(np,addr):`

Параметры : `int np` - номер формируемого шаблона

: `TINY *addr` - ссылка на массив байтов,

: задающих шаблон

Результат : `VOID` - фиктивный

Точка входа : `setpat`

3.2.3. Функция SETSPC

Функция SETSPC устанавливает цвета строк спрайта (в расширенном режиме) или цвет спрайта (в базовом режиме). Если пользователь находится в базовом режиме, то в качестве второго параметра функции берется цвет спрайта. Если же пользователь находится в расширенном режиме, то вторым параметром функции должен быть адрес массива из 8 или 16 элементов типа TINY, содержащего цвета строк спрайта. В действительности цвета определяются четырьмя младшими битами каждого байта. Старшие биты используются для указания некоторой дополнительной информации (например, выборочная реакция на столкновения). Более подробное описание назначения этих битов смотрите в Руководстве по языку Basic для компьютеров серии MSX-2.

Вызов функции : `setspc(ns,color):`

Параметры : `int ns` - номер спрайта

: `NAT color` - цвет или адрес массива

: цветов

Результат : `VOID` - фиктивный

Точка входа : `setspc`

3.2.4. Функция SETATR

Функция SETATR выводит спрайт с данным шаблоном в указанной плоскости в точку с заданными координатами. Номер плоскости всегда берется по модулю 32, номер шаблона - также, как в функции SETPAT.

Вызов функции : `setatr(ns,x,y,pr)`

Параметры : `int ns` - номер плоскости (спрайта)
: `int x,y` - координаты спрайта
: `int pr` - номер шаблона

Результат : `VOID` - фиктивный

Точка входа : `setatr`

3.2.5. Функция COLLISION

Функция COLLISION проверяет произошло ли столкновение спрайтов с момента последнего прерывания. При использовании этой функции нельзя запрещать прерывания.

Вызов функции : `collision()`

Параметры : -

Результат : `BOOL` `TRUE` - столкновение было
: `FALSE` - столкновения не было

Точка входа : `collis`

ОКОННЫЙ ВВОД/ВЫВОД

4.1. НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ ПАКЕТА

Пакет предназначен для организации работы с текстовыми окнами в графическом режиме SCREEN 7 MSX2-компьютера.

Используя функции пакета, пользователь может создать на экране любое количество текстовых окон и осуществлять операции ввода/вывода с любым из них, используя аналоги общепринятых функций языка C: `putchar, puts, printf, gets, scanf`. В каждом из окон автоматически поддерживается скроллинг и большинство однобайтовых управляющих символов. Скорость вывода текста больше, чем у процедуры `BDOS CONOUT`.

По желанию, пользователь может выводить данные в окно с использованием курсора или без него. При вводе курсор всегда появляется на экране.

В процессе вывода пользователь может менять цвет вывода и фона, а также позиционировать курсор в окне в относительных координатах.

Для работы с пакетом в среде компилятора C фирмы ASCII пользователю необходимо включить в программу оператор пре-процессора

```
#include "wnd2.h"
```

и, в процессе редактирования, подключить библиотеки `WND2.REL`, `GRP2.REL` и `SYS2.REL`.

При работе в среде M80 необходимо выделять 15 байт на каждое окно, редактировать с вышеуказанными библиотеками.

4.2. ОПИСАНИЕ ФУНКЦИИ ПАКЕТА

В состав пакета входят следующие функции:

- `winit` — инициализация окна;
- `wkill` — стирание окна фоном экрана;

- wloc - позиционирование курсора в окне;
- wcol - изменение цвета вывода и фона в окне;
- curon, curoff - включение и выключение курсора при выводе;
- wputchar - вывод символа в окно;
- wputs - вывод строки в окно;
- wprintf - форматный вывод в окно;
- wprtfloat - форматный вывод вещественного числа в окно;
- wget - ввод строки из окна;
- wscanf - ввод данных из окна;
- wgetfloat - ввод вещественного числа из окна;

В файле wmd2.h определен тип WINDOW - структура для окна. При работе с окнами необходимо иметь для каждого рабочего окна переменную типа WINDOW. Внутренняя ее структура прозрачна для пользователя.

4.2.1.3 Функция winit.

Функция winit создает окно на экране.

Вызов функции : winit(w,xl,yh,width,height,fcol,bcol);

Параметры : WINDOW *w -адрес переменной типа WINDOW, связанной с окном ("адрес окна").

NAT xl -координата левого верхнего угла по горизонтали в знакоместах (0-63);

NAT xh -координата левого верхнего угла по вертикали в знакоместах (0-23);

NAT fcol-цвет вывода в окно (фасадный);

NAT bcol - цвет фона окна.

Результат : VOID - фиктивный.

Точка входа : winit@

4.2.2. Функция wkill.

Функция wkill закрашивает окно цветом фона экрана.

Вызов функции : `wkill(w)` ;
Параметры : `WINDOW *w`-адрес переменной типа `WINDOW`,
связанной с окном ("адрес окна").
Результат : `VOID` - фиктивный.
Точка входа : `wkill@`

4.2.3. Функция `wloc`.

Функция `wloc` позиционирует курсор в окне.

Вызов функции : `wloc (w,x,y)` ;
Параметры : `WINDOW *w`-адрес переменной типа `WINDOW`,
связанной с окном ("адрес окна").
`NAT x` - координата нового положения
курсора по горизонтали в знаках окна,
т.е. для каждого окна в диапазоне от
0 до его ширины без единицы.
`NAT y` - координата нового положения
курсора по вертикали в знаках окна,
т.е. для каждого окна в диапазоне от 0
до его высоты без единицы.
Результат : `STATUS` (синоним `char`) - `OK (1)`, если ко-
ординаты указаны верно, и `ERROR`
(`((char)-1)`), если нет (позиционирование
не выполняется).
Точка входа : `wloc@`

4.2.4. Функция `wcol`.

Функция `wcol` меняет цвет вывода и фона в окне. Смена
цветов происходит только у вновь выводимых символов и у ниж-
ней строки при скроллинге, а также у всего окна при его
очистке символом `CLS ((char)12)`.

Вызов функции : `wcol(w.fcol.bcol)` ;

Параметры : WINDOW *w-адрес переменной типа WINDOW,
связанной с окном ("адрес окна").
NAT fcol -цвет вывода в окно (фасадный);
NAT bcol -цвет фона окна.

Результат : VOID - фиктивный.

Точка входа : wcol@

4.2.5. Функция curon.

Функция curon включает отображение курсора в окне. Курсор-инвертированные две нижние "полоски" символа.

Вызов функции : curon(w);

Параметры : WINDOW *w-адрес переменной типа WINDOW,
связанной с окном ("адрес окна").

Результат : VOID - фиктивный.

Точка входа : curon@

4.2.6. Функция curoff.

Функция curoff выключает отображение курсора в окне. Курсор - инвертированные две нижние "полоски" символа.

Вызов функции : curoff(w);

Параметры : WINDOW *w-адрес переменной типа WINDOW,
связанной с окном ("адрес окна").

Результат : VOID - фиктивный.

Точка входа : curoff

4.2.7. Функция wputchar.

Функция wputchar выводит символ в текущую позицию окна. Поддерживаются следующие управляющие символы: CR(13), LF(10), TAB(9), BS(8), HOME(11), CLS(очистка экрана- 12), UP, DOWN, RIGHT, LEFT (стрелки), а также двухбайтовые графические символы. ESC - последовательности, а также DEL и INS не поддерживают-

ся! Все вышесказанное об управляющих символах справедливо также для функций `wputs` и `wprintf`.

Вызов функции : `wputchar(w,c);`

Параметры : `WINDOW *w`-адрес переменной типа `WINDOW`, связанной с окном ("адрес окна").
`char c` - выводимый символ.

Результат : `VOID` - фиктивный.

Точка входа : `wputch`

4.2.8. Функция `wputs`.

Функция `wputs` выводит строку в окно. Строка терминируется символом `(char)0`.

Вызов функции : `wputs(w,s);`

Параметры : `WINDOW *w`-адрес переменной типа `WINDOW`, связанной с окном ("адрес окна").
`char *s`-адрес начала строки (можно указать саму строку в кавычках).

Результат : `VOID` - фиктивный.

Точка входа : `wputs@`

4.2.9. Функция `wprintf`.

Функция `wprintf` осуществляет форматный вывод в окно. Это функция с переменным числом параметров. Она является полным аналогом функции `printf` из стандартного набора функций C.

Вызов функции : `wprintf (w,format,var1,var2,...);`

Параметры : `WINDOW *w`-адрес переменной типа `WINDOW`, связанной с окном ("адрес окна").
`char *format`-адрес начала строки, содержащей описание формата (можно указать саму строку в кавычках).
`var1,var2,...`- выводимые переменные и константы.

Результат : STATUS (синоним char) - ОК, если все в порядке; ERROR, если нет соответствия числа описателей формата и параметров.

Точка входа : wprint

4.2.10. Функция wprtfloat.

Функция wprtfloat осуществляет форматный вывод вещественного числа в окно в формате с фиксированной точкой, если все цифры до точки-значашие и с плавающей точкой - в противном случае.

Это функция с переменным числом параметров. Если указаны 2 параметра то выводятся все значащие цифры числа с позиции курсора. Если указаны 4 параметра, то 3 и 4 воспринимаются соответственно как длина поля вывода и число знаков после десятичной точки. Результат выравнивается вправо.

О вещественных числах см. описание пакета математических функций.

Вызов функций : wprtfloat (w,f); или
wprtfloat (w,f,a,b);

Параметры : WINDOW *w-адрес переменной типа WINDOW, связанной с окном ("адрес окна").
FLOAT f- выводимое число;
NAT a - длина поля вывода;
NAT b - число знаков после точки.

Результат : STATUS (синоним char)- ОК, если все в порядке; ERROR, если число параметров меньше 2.

Точка входа : wprtf1

4.2.11. Функция wget.

Функция wget позволяет вводить строку с клавиатуры и редактировать ее в процессе ввода до нажатия клавиши <RETURN>. Возможности редактирования аналогичны вводу при помощи функ-

ции BIOS BUFIN, например, как при вводе командной строки DOS. Поддерживаются функции клавиш со стрелками и клавиш HOME и BS. Функции клавиш SELECT, INS и DEL не поддерживаются. Двухбайтовые графические символы не поддерживаются. В отличие от функции gets пакета функций ASCII C, завершающий символ \n не вводится в строку, она терминируется символом '\0'. Опыт разработчиков свидетельствует, что это удобно. Внимание! Все функции ввода включают курсор при вызове и выключают при выходе.

Вызов функции : wgets(w,s,length):

Параметры : WINDOW *w-адрес переменной типа WINDOW,
связанной с окном ("адрес окна").
char *s-адрес вводимой строки.
NAT length - наибольшая длина строки.

Результат : VOID - фиктивный.

Точка входа : wgets@

4.2.12. Функция wscanf.

Аналог стандартной функции scanf для ввода из окна.

Возможности редактирования те же, что и у wgets.

Вызов функции : wscanf(w,format,&var1,&var2,...):

Параметры : WINDOW *w-адрес переменной типа WINDOW,
связанной с окном ("адрес окна").
char *format - строка-описатель формата.

Результат : int - как у scanf.

Точка входа : wscanf

4.2.13. Функция wgetfloat.

Ввод вещественного числа с клавиатуры с отображением в окне. Возможности редактирования те же, что и у wgets.

Вызов функции : wgetfloat(w,f):

Параметры : WINDOW *w-адрес переменной типа WINDOW,
связанной с окном ("адрес окна").

Float f - вещественная переменная.

Результат : int - не определен.

Точка входа : wgetfl

КЛАВИАТУРА

5.1. НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ ПАКЕТА

Пакет "Клавиатура" позволяет просто и в то же время максимально эффективно использовать имеющиеся в распоряжении пользователя возможности работы с клавиатурой.

Ввод с клавиатуры осуществляется непосредственным сканированием матрицы клавиатуры (см. Приложение В), что значительно ускоряет обработку и облегчает программирование сложных интерактивных систем, к каковым, в частности относятся и обучающие программы.

Включенные в пакет функции реализуют два взаимно дополняющих способа доступа к клавиатуре:

- синхронный ввод с клавиатуры
- асинхронный ввод с клавиатуры

Синхронный ввод с клавиатуры осуществляется непосредственно в момент вызова соответствующей функции, в то время как асинхронный ввод выполняется непрерывно 50 раз в секунду. Во втором случае результаты сканирования клавиатуры накапливаются в однобайтовом аккумуляторе (запоминается информация только о восьми, выбранных пользователем с помощью функции KEYON клавишах). Считывание этой информации выполняется уже синхронно в момент вызова соответствующей функции, после чего она сбрасывается. Кроме того, инициализация аккумулятора (сброс информации о результатах сканирования соответствующей клавиши) выполняется при вызове функции KEYON. Таким образом, в отличие от функций первого типа, функции второго типа позволяют обрабатывать нажатия на указанные клавиши выполненные в течение определенного промежутка времени.

К функциям первого типа относятся SNSMAT, ANYKEY, STRIG, STICK, FKEY, LETTER, ESC, TAB, CTRL, SHIFT, GRAPH, CR, BS, SELECT, HOME, INS, DEL, STOP, CAPS, RUS.

К функциям второго типа: KEYON, PRESS, WAIT.

Кроме того, в пакет включена функция KILBUF дополняющая стандартные средства ввода через буфер клавиатуры, и функция LAMP, переключающая лампочки CAPS и РУС.

5.2. ОПИСАНИЕ ФУНКЦИИ ПАКЕТА

В пакет входят следующие процедуры и функции:

- SNSMAT - чтение маски ряда клавиатуры
- KEY - определение состояния клавиши. заданной ее положением в матрице клавиатуры
- ANYKEY - определяет, нажата ли хоть одна клавиша
- STRIG - определение состояния кнопок джойстиков и клавиши <ПРОБЕЛ>
- STICK - определение состояния рукояток джойстиков и клавиш со стрелками
- FKEY - определение состояния функциональный клавиш
- LETTER - определение состояния буквенно-цифровой клавиши, заданной кодом одного из соответствующих ей символов
- ESC, TAB, CTRL, SHIFT, GRAPH, CR, BS, SELECT, HOME, INS, DEL, STOP, CAPS, RUS - определение состояния "черных" клавиш
- LAMP - включение/выключение лампочек РУС и CAPS и соответствующих регистров клавиатуры
- KEYON - включение асинхронного опроса клавиши, заданной своим кодом
- KEYOFF - выключение асинхронного опроса всех клавиш
- PRESS - определение состояния асинхронно опрашиваемой клавиши с момента предыдущего вызова этой функции или процедуры KEYON
- WAIT - ожидание нажатия клавиши или истечения заданного промежутка времени
- KILBUF - очистка буфера клавиатуры

Ниже приводится краткое описание назначения, входных и выходных параметров включенных в пакет функций.

5.2.1. Функция SNSMAT

Функция SNSMAT возвращает маску указанного ряда клавиатуры. Нулевые биты соответствуют нажатым клавишам. Матрица клавиатуры приведена в Приложении В.

Вызов функции : `snsmat(row)`;
Параметры : TINY row - номер ряда,
Результат : TINY - маска указанного ряда
Точка входа : `snsmat`

5.2.2. Функция KEY

Функция KEY возвращает '\1', если нажата клавиша указанной колонки указанного ряда матрицы клавиатуры (см. Приложение В) и '\0' в противном случае.

Вызов функции : `key(row,column)`
Параметры : TINY row,column;
Результат : BOOL (синоним char)
Точка входа : `key@`

5.2.3. Функция ANYKEY

Функция ANYKEY возвращает TRUE, если нажата какая-либо клавиша на клавиатуре, в том числе SHIFT, GRAPH, STOP и т.д. В противном случае возвращается FALSE.

Вызов функции : `anykey()`;
Параметры : -
Результат : BOOL TRUE - клавиша нажата
: FALSE - клавиша не нажата
Точка входа : `anykey`

5.2.4. Функция STRIG

Функция STRIG является аналогом оператора STRIG(n) в BASIC'е. Возвращает TRUE, если нажата кнопка n (n=0 - пробел, n=1 - кнопка первого джойстика, n=2 - второго). В противном случае возвращается FALSE.

Вызов функции : strig(n);

Параметры : TINY n - номер кнопки

Результат : BOOL TRUE - кнопка нажата
: FALSE- кнопка не нажата

Точка входа : strig@

5.2.5. Функция STICK

Функция STICK является аналогом оператора STICK(n) в BASIC'е. Возвращает состояние клавиш со стрелками при n=0 или рукоятки джойстика при n=1,2.

Вызов функции : stick(n);

Параметры : TINY n - номер кнопки

Результат : TINY

Точка входа : stick@

5.2.6. Функция FKEY

Функция FKEY возвращает TRUE если нажата функциональная клавиша с номером n. В противном случае возвращается FALSE.

Вызов функции : fkey(n);

Параметры : TINY n - номер функциональной клавиши

Результат : BOOL TRUE - клавиша нажата
: FALSE- клавиша не нажата

Точка входа : fkey@

5.2.7. Функция LETTER

Функция LETTER возвращает TRUE если нажата "белая" клавиша, на которой изображен символ с указанным кодом. В противном случае возвращается FALSE.

Вызов функции : letter(code)

Параметры : TINY code - код клавиши

Результат : BOOL TRUE - клавиша нажата
 : FALSE- клавиша не нажата

Точка входа : letter

5.2.8. Функция ESC

Функция ESC проверяет, нажата ли клавиша <esc>.

Вызов функции : esc();

Параметры : -

Результат : BOOL TRUE - клавиша нажата
 : FALSE- клавиша не нажата

Точка входа : esc@

5.2.9. Функция TAB

Функция TAB проверяет, нажата ли клавиша <tab>.

Вызов функции : tab();

Параметры : -

Результат : BOOL TRUE - клавиша нажата
 : FALSE- клавиша не нажата

Точка входа : tab@

5.2.10. Функция CTRL

Функция CTRL проверяет, нажата ли клавиша <ctrl>.

Вызов функции : ctrl();

Параметры : -

Результат : BOOL TRUE - клавиша нажата
: FALSE- клавиша не нажата
Точка входа : ctrl@

5.2.11. Функция SHIFT

Функция SHIFT проверяет, нажата ли клавиша <shift>.
Вызов функции : shift();
Параметры : -
Результат : BOOL TRUE - клавиша нажата
: FALSE- клавиша не нажата
Точка входа : shift@

5.2.12. Функция GRAPH

Функция GRAPH проверяет, нажата ли клавиша <graph>.
Вызов функции : graph();
Параметры : -
Результат : BOOL TRUE - клавиша нажата
: FALSE- клавиша не нажата
Точка входа : graph@

5.2.13. Функция CR

Функция CR проверяет, нажата ли клавиша возврата каретки
Вызов функции : cr();
Параметры : -
Результат : BOOL TRUE - клавиша нажата
: FALSE- клавиша не нажата
Точка входа : cr@

5.2.14. Функция BS

Функция BS проверяет, нажата ли клавиша <bs>
Вызов функции : bs();
Параметры : -
Результат : BOOL TRUE - клавиша нажата
: FALSE- клавиша не нажата
Точка входа : bs@

5.2.15. Функция SELECT

Функция SELECT проверяет, нажата ли клавиша <select>
Вызов функции : select();
Параметры : -
Результат : BOOL TRUE - клавиша нажата
: FALSE- клавиша не нажата
Точка входа : select

5.2.16. Функция HOME

Функция HOME проверяет, нажата ли клавиша <cls/home>
Вызов функции : home();
Параметры : -
Результат : BOOL TRUE - клавиша нажата
: FALSE- клавиша не нажата
Точка входа : home@

5.2.17. Функция INS

Функция INS проверяет, нажата ли клавиша <ins>
Вызов функции : ins();
Параметры : -
Результат : BOOL TRUE - клавиша нажата
: FALSE- клавиша не нажата
Точка входа : ins@

5.2.18. Функция DEL

Функция DEL проверяет, нажата ли клавиша

Вызов функции : del()

Параметры : -

```

Резултат      : BOOL      TRUE - клавиша нажата
                  : FALSE- клавиша не нажата

```

Точка входа : del@

5.2.19. Функция STOP

Функция STOP проверяет, нажата ли клавиша <stop>

Вызов функции : stop();

Параметры : -

```
Резултат      : BOOL    TRUE - клавиша нажата
                :          FALSE- клавиша не нажата
```

Точка входа : stop@

5.2.20. Функция CAPS

Функция CAPS проверяет, зажжена ли лампочка <caps>

Вызов функции : caps () :

Параметры : -

```

Результат      : BOOL    TRUE  - лампочка зажжена
                  :        FALSE- лампочка не зажжена

```

Точка входа : caps@

5.2.21. Функция RUS

Функция RUS проверяет, зажжена ли лампочка <рус>

ВЫЗОВ функции : rus();

Параметры : -

```

Результат      : BOOL      TRUE - лампочка зажжена
                  :          FALSE- лампочка не зажжена

```

Точка входа : ruse@

5.2.22. Функция LAMP

Функция LAMP переключает лампочки РУС и CAPS.

Вызов функции : lamp(1,sw):

Параметры : char 1='R'-переключается лампочка <рус>
: 'C'-переключается лампочка <caps>
: BOOL sw =TRUE - лампочка включается
: =FALSE- лампочка выключается

Результат : VOID - фиктивный

Точка входа : lamp@

5.2.23. Функция KEYON

Функция KEYON закрепляет за указанной клавишей идентифицирующий ее номер, который в дальнейшем будет использоваться в функциях асинхронного ввода с клавиатуры. После выполнения этой функции бит аккумулятора клавиатуры, соответствующий указанному номеру, сбрасывается. Идентификация "белых" клавиш клавиатуры выполняется с помощью кода любого из изображенных на ней символов. В случае "черных" клавиш, идентификация выполняется по закрепленным за ними кодам (<esc>-27,<tab>-9,<возврат каретки>-13,<bs>-8,<select>-24,<cls/home>-11,<ins>-18,-127,<стрелка вверх>-30,<стрелка вправо>-28,<стрелка вниз>-31,<стрелка влево>-29). Для клавиш <ctrl>,<shift>,<graph>,<caps>,<рус>,<stop> в качестве таких кодов берутся коды '\00','\01','\02','\03','\04','\05' соответственно.

Вызов функции : keyon(n,code);

Параметры : TINY n : присваиваемый клавише номер,
: 0 < n < 9
: TINY code : код клавиши

Результат : VOID - фиктивный

Точка входа : keyon@

5.2.24. Функция PRESS

Функция PRESS проверяет была ли за время, прошедшее с момента ее предыдущего вызова, нажата клавиша с указанным номером (закрепленным за ней функцией KEYON). Заметим, что если перед вызовом описываемой функции была выполнена функция KEYON для соответствующего номера, то все нажатия на указанную клавишу, имевшие до этого место, игнорируются.

Вызов функции : `press(n)` ;

Параметры : TINY `n` : номер клавиши

Результат : BOOL TRUE - клавиша была нажата
 FALSE - клавиша не была нажата

Точка входа : `press@`

5.2.25. Функция WAIT

Функция WAIT ожидает пока не будет выполнено одно из следующих условий:

- нажата клавиша с указанным номером (`n`)
- истек указанный промежуток времени (`t`)

Отсчет времени и проверка нажата ли указанная клавиша начинаются с момента вызова функции. Если `n=0` или `n>8`, то первая проверка не выполняется, если `t=0`, то не выполняется вторая проверка.

Вызов функции : `wait(n,t)` ;

Параметры : TINY `n` - номер клавиши

 : NAT `t` - время ожидания

Результат : VOID - фиктивный

Точка входа : `wait@`

5.2.26 Функция KILBUF

Функция KILBUF чистит буфер клавиатуры.

Вызов функции : kilbuf();

Параметры : -

Результат : VOID - фиктивный;

Точка входа : kilbuf

МУЗЫКАЛЬНЫЙ ПАКЕТ

6.1. НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ ПАКЕТА

Музыкальный пакет делает возможным фоновое двухканальное исполнение мелодий в ходе выполнения программ, написанных на языке Си. Третий канал оставлен свободным для звуковых эффектов.

Процедуры пакета можно также использовать при работе с Ассемблером M80

Макроязык для создания мелодии одного голоса является подмножеством MML BASIC и позволяет использовать следующую мнемонику MML: A, B, C, D, E, F, G, R, +, ., ., O, T. Текст мелодии можно создать в текстовом редакторе, а также в бейсик-программе и записать на диск. С помощью программы-конвертора текст мелодии преобразуется в текст на языке Ассемблера. Снабдив этот текст соответствующей меткой и откомпилировав его с помощью M80, пользователь может подключать объектный код мелодии на этапе редактирования к своим программам.

6.2. ОПИСАНИЕ ФУНКЦИИ ПАКЕТА

- INIMUS - Инициализация системы
- KILMUS - Отключение системы
- VOICES - Задание имен первого и второго голосов
- STARTM - исполнение двухканальной мелодии.
- HOLDM - остановка исполнения мелодии.
- SOUND - непосредственный доступ к PSG
- PLAYES - Проверка звучания мелодии.
- SOUND - Запись в регистр PSG

6.2.1. Функция INIMUS

Функция INIMUS инициализирует музыкальную систему.

Вызов функции : INIMUS();
Параметры : нет
Результат : VOID - фиктивный
Точка входа : INIMUS

6.2.2. Функция KILMUS

Функция KILMUS отключает музыкальную систему.

Вызов функции : KILMUS();
Параметры : нет
Результат : VOID - фиктивный
Точка входа : KILMUS

6.2.3. Функция VOICES

Функция VOICES задает имена первого и второго голоса. В случае одноголосой мелодии задаются два одинаковых имени.

Вызов функции : VOICES(v1,v2)
Параметры : VOICE v1,v2
: v1 и v2 суть метки, присвоенные голосам
: при создании REL-файлов.
Результат : VOID - фиктивный
Точка входа : VOICES

6.2.4. Функция STARTM

Функция STARTM начинает исполнение мелодии.

Вызов функции : STARTM(r12,r13,r89);
Параметры : int r12,r13 - значение 12 и 13 ре-

гистров PSG. (параметры M и S MML Basic)

int r89 - целое число, в старшем байте которого записано значение 8 регистра PSG, в младшем - значение 9 регистра. Значения 8 и 9 регистров PSG - это параметры V соответственно для 1-го и 2-го голосов (громкость) . Если для голоса используется переменная амплитуда, значение соответствующего регистра должно равняться 16. Подробно смысл значений регистров 8,9,12,13 можно узнать из Вашего руководства по MSX-Бейсику.

Результат : VOID - фиктивный
Точка входа : STARTM

6.2.5. Функция HOLDM

Функция HOLDM останавливает исполнение мелодии.

Вызов функции : HOLDM();

Параметры : нет

Результат : VOID - фиктивный

Точка входа : HOLDM@

6.2.6. Функция PLAYES

Функция PLAYES возвращает (BOOL)TRUE, если в момент вызова исполнялась мелодия, иначе возвращает (BOOL)FALSE

Может быть использована для "зацикливания" мелодии

Вызов функции : PLAYES();

Параметры : нет

Результат : BOOL

Точка входа : PLAYES

6.2.7. Функция SOUND

Функция SOUND предназначена для записи в регистр PSG

Вызов функции : SOUND(reg,data);

Параметры: : TINY reg- номер регистра

: TINY data - значение.

Результат : VOID - фиктивный

6.3. ОГРАНИЧЕНИЯ

6.3.1.Количества тактов в первом и втором голосе должны совпадать (для этого можно использовать мнемокод паузы- R). По окончании мелодии первого голоса исполнение всей музыки прекращается.

6.3.2. По выходе из пользовательской программы рекомендуется для нормального функционирования DOS отключить музыкальную систему с помощью KILMUS.

6.4. РАБОТА С ПРОГРАММОЙ-КОНВЕРТОРОМ. СОЗДАНИЕ REL-ФАЙЛА, СОДЕРЖАЩЕГО ПЕРЕМЕЩАЕМЫЙ ОБЪЕКТНЫЙ КОД МУЗЫКАЛЬНОГО ГОЛОСА

Программа-конвертор вызывается следующим образом:

A>MUS <имя1 >имя2

имя1 - имя текстового файла,содержащего текст мелодии
Текст мелодии рекомендуется предварительно отладить в Бейсик-системе. Программа-конвертор контроля за ошибками не ведет.

имя2 - имя текстового файла,куда будет записываться ассемблерный текст.

Далее пользователь добавляет в ассемблерный текст метку, обязательно описанную PUBLIC, и компилирует его при помощи M80. Полученный REL-файл можно подключать на этапе редактирования к различным программам, в тексте которых его метка (но не имя!) описано как extern VOICE или extern ~~char~~ {} (для программ на Си) или extrn (для программ на Ассемблере).

Примечание. К метке, используемой в Си-программе и содержащей менее шести символов необходимо приписать один символ "@".

СИСТЕМНЫЕ ПРОЦЕДУРЫ

7.1. НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ ПАКЕТА

Пакет системных процедур предоставляет в распоряжение пользователя гибкие и эффективные средства работы с видео-процессором, слотовым механизмом памяти, а также средства для интерслотового вызова процедур из языков Си или Ассемблер.

Включенные в пакет функции условно можно разбить на восемь групп:

- функции для работы со слотовым механизмом памяти;
- функции для интерслотового вызова процедур из языка Си;
- процедуры для работы с видеорегистрами;
- установочные функции;
- функции для работы с видеопамтью
(линейные блоки видеопамти);
- функции для работы с видеопамтью
(прямоугольные блоки видеопамти);
- функции для работы с графическими окнами;
- вспомогательные функции.

К функциям первой группы относятся SETSLT (установка требуемой конфигурации памяти), RDSLT (чтение данных из произвольного слота), RDERAM (чтение данных из расширенного RAM'a) и WRERAM (запись данных в расширенный RAM).

К функциям второй группы относятся CALSLT (интерслотовый вызов процедуры из произвольного слота), BBIOS (интерслотовый вызов процедур Базовой Системы Ввода/Вывода) и EBIOS (интерслотовый вызов процедур Расширенной Системы Ввода/Вывода).

К третьей группе относятся функция VDPRD (чтение управляющего регистра видеопроцессора), функция VDPWR (запись в управляющий регистр видеопроцессора) и функция PALETTE (установка регистров палитры видеопроцессора).

К четвертой группе относятся функции установки режима отображения (DSPMODE) и установки режима мерцания (BLON, BLOFF).

Внимание !! Функции BLON и BLOFF можно использовать только в текстовом режиме (SCREEN 0) с шириной строки, превышающей 40 (графический режим видеопроцессора TXT80).

К функциям пятой группы относятся функции, предоставляющие в распоряжение пользователя средства обмена между памятью и видеопамятью, а также сравнения их содержимого. Кроме того, пользователь может выполнять как инкрементное, так и декрементное копирование одного участка видеопамати на другой. Наличие двух видов копирования (инкрементное и декрементное) позволяет корректно выполнять эту операцию даже, если соответствующие участки видеопамати пересекаются. И память и видеопамать трактуются при этом как линейные массивы информации. В эту группу входят функции VCOMP, VFILL, VGET, VPUT, VCOPY.

К шестой группе относятся функции, работающие с прямоугольными блоками видеопамати. К ним относятся процедуры BLKFIL, BLKGET, BLKPUT, BLKCPY и BNDCPY. При работе с этими функциями следует учитывать следующее: координаты блоков и их размеры берутся по модулю 2 в экранных режимах 5 и 7 и по модулю 4 в экранном режиме 6.

Функции этой группы работают с видеопаматью как с единым целым (номер активной страницы во внимание не принимается). Используемые при этом координатные сетки (для разных экранных режимов они разные) приведены в Приложении А.

Отметим одну особенность работы функций BLKGET и BLKPUT: если код логической операции равен нулю (см. раздел 2, описание функции LOGOP), то чтение и запись в видеопамать выполняются побайтно; если же код логической операции отличен

от нуля, чтение и запись в видеопамять будут выполняться поточно (один байт на точку). В связи с этим пользователь должен при выделении в памяти места под содержимое блока учитывать какое копирование будет выполняться. В случае побайтного копирования размер выделяемой области памяти в байтах вычисляется по формуле $w * h * k$, где w - ширина блока в пикселах, h - высота блока в пикселах, а $k = 1/2$ в экранных режимах 5 и 7, $= 1/4$ в экранном режиме 6 и $= 1$ в восьмом экранном режиме. При поточечном копировании k всегда равно единице.

Внимание !! Функции этой группы можно использовать только в старших экранных режимах (5,6,7 и 8).

К седьмой группе относятся функции, аналогичные функциям пятой группы, но рассматривающие память и видеопамять как двумерные массивы (графические окна). При этом в состав графического окна включаются все области видеопамяти, формирующие его образ на экране монитора. Так в экранных режимах 0, 1, 3, 5, 6, 7, и 8 (SCREEN0, SCREEN1, SCREEN3, SCREEN5, SCREEN6, SCREEN7, SCREEN8) - это соответствующие участки таблицы шаблонов, тогда как во втором и четвертом экранных режимах (SCREEN2, SCREEN4) - соответствующие участки таблицы шаблонов и таблицы цветов. Заметим, что в первом экранном режиме (SCREEN1) таблица цветов не участвует в формировании образа графического окна, так как она определяет не цвет соответствующих знакомест, а цвет символов с соответствующими кодами. Если пользователь желает изменить цвета выводимых символов, он должен сам занести необходимую информацию в таблицу цветов, воспользовавшись функциями VFILL или VPUT настоящего пакета, или функцией VPOKE пакета графических функций. В эту группу включены функции WFILL, WSAVE, WREST, WCOPY,

Функции седьмой группы предоставляют в распоряжение пользователя удобные средства для работы с графическими окнами. При работе с этими функциями в зависимости от того, в каком экранном режиме находится в данный момент система, автоматически определяются модифицируемые таблицы видеопроцессора, адреса и размеры соответствующих блоков в видеопамяти.

Контроль на выход задаваемого пользователем графического окна за границы экрана при работе с функциями седьмой группы не выполняется.

При работе с графическими окнами в младших экранных режимах (SCREEN0, SCREEN1, SCREEN2, SCREEN3, SCREEN4) номер активной страницы должен быть равен нулю. В старших же экранных режимах (SCREEN5, SCREEN6, SCREEN7, SCREEN8) чтение и запись в видеопамять выполняются также, как и в функциях, работающих с прямоугольными блоками видеопамати (номер активной страницы во внимание не принимается, и в зависимости от значения кода логической операции обмен выполняется либо побайтно, либо поточно).

Наконец, к функциям восьмой группы относятся функции, используемые функциями первых семи групп, но пользователю "недоступные". Последнее означает, что при вызове этих функций принятое в языке Си соглашение о связях не соблюдается.

7.2. ОПИСАНИЕ ПРОЦЕДУР И ФУНКЦИИ ПАКЕТА

В состав пакета входят следующие функции:

- SETSLT - установка указанной конфигурации памяти;
- RDSLT - чтение данных из указанного слота;
- RDERAM - чтение данных из расширенного RAM'a;
- WRERAM - запись данных в расширенный RAM;
- CALSLT - вызов процедуры из указанного слота;
- BBIOS - интерслотовый вызов BIOS'ов;
- EBIOS - интерслотовый вызов расширения BIOS'ов;
- VDPRD - чтение содержимого указанного управляющего регистра видеопроцессора;
- VDPWR - запись байта данных в указанный управляющий регистр видеопроцессора;
- PALETTE - установка указанного регистра палитры;
- DSPMODE - установка режима отображения;

- BLON - установка мерцания заданного "знакоместа" экрана;
- BLOFF - снятие мерцания заданного "знакоместа" экрана;
- DIRECT - установка направления копирования;
- VCOMP - сравнение блока данных в памяти с указанным блоком видеопамати;
- VFILL - заполнение участка видеопамати указанным байтом;
- VGET - чтение блока данных из видеопамати;
- VPUT - запись блока данных в видеопамать;
- VCOPY - копирование одного участка видеопамати на другой;
- BLKFIL - заполнение прямоугольного блока видеопамати заданным байтом;
- BLKGET - чтение прямоугольного блока из видеопамати;
- BLKPUT - запись прямоугольного блока в видеопамать;
- BLKCPY - копирование прямоугольного блока видеопамати;
- BNDCPY - копирование в вертикальном направлении горизонтальной полосы видеопамати;
- WFILL - "инициализация" графического окна;
- WSAVE - сохранение содержимого графического окна в памяти;
- WREST - восстановление сохраненного в памяти графического окна;
- WCOPY - копирование графического окна;
- _STATUS - чтение содержимого регистра состояния;
- _READY - ожидание завершения выполнения команды видео-процессора;
- _DSET - установка параметров блока видеопамати;
- _VSET - вычисление адреса в видеопамати по координатам знакоместа;
- _WSET - вычисление ширины графического окна в байтах.

Все функции с префиксом "_" являются вспомогательными. Пользователю доступны только функции SETSLT, RDSLT, RDERAM, WRERAM, CALSLT, BBIOS, EBIOS, VDPRD, VDPWR, PALETTE, DSPMODE, BLON, BLOFF, DIRECT, VCOMP, VFILL, VGET, VPUT, VCOPY, BLKFIL, BLKGET, BLKPUT, BLKCPY, BNDCPY, WFILL, WSAVE, WREST, WCOPY.

Ниже приводится краткое описание назначения, входных и выходных параметров доступных пользователю функций пакета.

7.2.1. Функция SETSLT

Функция SETSLT устанавливает требуемую конфигурацию памяти.

```
Вызов функции : setslt(conf);
Параметры      : TINY conf = 0 - конфигурация Бейсик:
                  :              0 страница- ROM BIOS,
                  :              1 страница- ROM BASIC,
                  :              2,3 страницы - RAM;
                  :              = 1 - конфигурация "игр":
                  :              0 страница- ROM BIOS,
                  :              1,2,3 страницы - RAM;
                  :              = 1 - конфигурация DOS:
                  :              64K RAM.
Результат      : VOID      - фиктивный
Точка входа    : setslt
```

7.2.2. Функция RDSLT

Функция RDSLT читает содержимое указанного слота по указанному адресу.

```
Вызов функции : rdslt(slot,source);
Параметры      : TINY slot   - спецификация слота
                  : TINY *source - адрес
Результат      : TINY        - считанный байт
Точка входа    : rdslt@
```

7.2.3. Функция RDERAM

Функция RDERAM читает байт данных из расширенного RAM'a, находящийся по указанному адресу.

Вызов функции : `rderam(source)`;

Параметры : `TINY *source` - адрес

Результат : `TINY` - считанный байт

Точка входа : `rderam`

7.2.4. Функция WRERAM

Функция WRERAM записывает байт данных в расширенный RAM по указанному адресу.

Вызов функции : `wreram(dest,byte)`;

Параметры : `TINY *dest` - адрес

: `TINY byte` - записываемый байт данных

Результат : `VOID` - фиктивный

Точка входа : `wreram`

7.2.5. Функция CALSLT

Функция CALSLT осуществляет интерслотовый вызов процедуры находящейся в указанном слоте по указанному адресу. Перед вызовом процедуры пользователь должен сформировать входной список, то есть выделить под него память и занести в него, если это необходимо содержимое тех регистров, которые должны быть установлены перед вызовом процедуры. Допускается передача параметров только через регистры A,F,B,C,D,E,H,L. После выполнения вызываемой процедуры функция CALSLT размещает содержимое перечисленных выше регистров в памяти и возвращает соответствующий адрес пользователю (адрес выходного

списка). Содержимое регистров как во входном, так и в выходном списке размещается в следующем порядке: F, A, L, H, E, D, C, B.

Вызов функции :calslt(slot, ent, inp);

Параметры	:TINY slot	- спецификация слота
	:TINY *ent	- адрес вызываемой процедуры
	:	
	:TINY *inp	- ссылка на входной список
Результат	:TINY*	- ссылка на выходной список
Точка входа	:calslt	

7.2.6. Функция VBIOS

Функция VBIOS осуществляет интерслотовый вызов процедур Базовой Системы Ввода/Вывода (BIOS). Эта функция отличается от функции CALSLT только тем, что не надо указывать спецификацию слота (нужная спецификация устанавливается автоматически).

Вызов функции :bbios(ent, inp);

Параметры	:TINY *ent	- адрес вызываемой процедуры
	:	
	:TINY *inp	- ссылка на входной список
Результат	:TINY*	- ссылка на выходной список
Точка входа	:bbios@	

7.2.7. Функция EBIOS

Функция EBIOS осуществляет интерслотовый вызов процедур Расширенной Системы Ввода/Вывода (EBIOS). Эта функция отличается от функции CALSLT только тем, что не надо указывать спецификацию слота (нужная спецификация устанавливается автоматически).

Вызов функции :ebios(ent,inp);
Параметры :TINY *ent - адрес вызываемой процедуры
:TINY *inp - ссылка на входной список
Результат :TINY* - ссылка на выходной список
Точка входа :ebios@

7.2.8. Функция VDPRD

Функция VDPRD читает содержимое управляющего видеоре-
гистра с указанным номером.

Вызов функции : vdprd(reg);
Параметры : TINY reg - номер видеоре-
гистра
Результат : TINY - содержимое видеоре-
гистра
Точка входа : vdprd@

7.2.9. Функция VDPWR

Функция VDPWR записывает данные в указанный управ-
ляющий видеоре-
гистр.

Вызов функции : vdpwr(reg,data);
Параметры : TINY reg - номер видеоре-
гистра
: TINY data - записываемые данные
Результат : VOID - фиктивный;
Точка входа : vdpwr@

7.2.10. Функция PALETTE

Функция PALETTE устанавливает указанный регистр (цвет)
палитры видеопроцессора.

Вызов функции : `palette(c,r,b,g);`
 Параметры : TINY c - номер регистра палитры (цвета)
 : TINY r - интенсивность красной
 : составляющей
 : TINY b - интенсивность синей
 : составляющей
 : TINY g - интенсивность зеленой
 : составляющей
 Результат : VOID - фиктивный
 Точка входа : `palett`

7.2.11. Функция DSPMODE

Функция DSPMODE устанавливает режим отображения видео-памяти на экран монитора. Если второй параметр (`disp`) равен TRUE, то номер текущей отображаемой страницы должен быть нечетным. Четыре варианта установки параметров `lace` и `disp` соответствуют четырем возможным значениям параметра `interlace mode` оператора SCREEN языка Basic (см. описание языка Basic для компьютеров серии MSX-2).

Вызов функции : `dspmode(lace,alter);`
 Параметры : BOOL lace = TRUE - отображение со
 : смещением на 1/2 пиксела
 : по вертикали
 : (interlace mode)
 : = FALSE - в противном случае
 : BOOL disp = TRUE - отображение с че-
 : редованием текущей и пре-
 : дыдущей страниц
 : = FALSE - в противном
 : случае
 Результат : -
 Точка входа : `dspmod`

7.2.12. Функция BLON

Функция BLON устанавливает параметры мерцания и координаты мерцающего знакоместа. В качестве единицы измерения времени берется 1/6 секунды.

Вызов функции : `blon(x,y,fc,bc,ft,bt);`

Параметры : `int x,y` - координаты знакоместа
: `TINY fc` - фасадный цвет мерцания
: `TINY bc` - фоновый цвет мерцания
: `int ft` - продолжительность интервала
: включения цветов мерцания
: `int bt` - продолжительность интервала
: выключения цветов мерцания

Результат : `VOID` - фиктивный

Точка входа : `blon@`

7.2.13. Функция BLOFF

Функция BLOFF снимает мерцание знакоместа с заданными координатами. Параметры мерцания при этом сбрасываются.

Вызов функции : `bloff(x,y);`

Параметры : `int x,y` - координаты знакоместа

Результат : `VOID` - фиктивный

Точка входа : `bloff`

7.2.14. Функция DIRECT

Функция DIRECT устанавливает номер страницы-источника и страницы-цели для функций копирования VCOPY и WCOPY. Номера страниц берутся по модулю 4 для экранных режимов 5, 6 и по модулю 2 для экранных режимов 7, 8.

Вызов функции : `direct(spage, dpage);`
 Параметры : TINY spage - номер страницы-источника
 : TINY dpage - номер страницы-цели
 Результат : VOID - фиктивный
 Точка входа : `direct`

7.2.15. Функция VCOMP

Функция VCOMP сравнивает побайтно содержимое блока данных в памяти с указанным блоком в видеопамати. Адресация видеопамати выполняется в пределах активной страницы.

Вызов функции : `vcomp(source, dest, length);`
 Параметры : TINY *source - адрес сравниваемого
 : блока в памяти
 : TINY *dest - адрес сравниваемого
 : блока в видеопамати
 : NAT length - размер сравниваемых
 : блоков
 Результат : BOOL = TRUE - содержимое бло-
 : ков совпадает
 : = FALSE - в противном
 : случае
 Точка входа : `vcomp@`

7.2.16. Функция VFILL

Функция VFILL заполняет указанную область видеопамати данным байтом. Адресация видеопамати выполняется в пределах активной страницы.

Вызов функции : `vfill(byte,dest,length)` :

Параметры : `TINY byte` - записываемый байт
 : `TINY *dest` - адрес заполняемой области
 : видеопамяти
 : `NAT length` - размер области

Результат : `VOID` - фиктивный

Точка входа : `vfill@`

7.2.17. Функция VGET

Функция VGET читает данные из видеопамяти. Адресация видеопамяти выполняется в пределах активной страницы.

Вызов функции : `vget(source,dest,length)` :

Параметры : `TINY *source` - адрес данных в видеопамяти
 : :
 : `TINY *dest` - адрес. в памяти. по которому размещаются считанные данные
 : :
 : `NAT length` - количество считываемых байт
 : :
 Результат : `VOID` - фиктивный

Точка входа : `vget@`

7.2.18. Функция VPUT

Функция VPUT записывает данные в видеопамять. Адресация видеопамяти выполняется в пределах активной страницы.

Вызов функции : `vput(source, dest, length);`
 Параметры : `TINY *source` - адрес данных в памяти
 : `TINY *dest` - адрес, в видеопамати,
 : по которому записываются
 : данные
 : `NAT length` - количество записываемых
 : байт
 Результат : `VOID` - фиктивный
 Точка входа : `vput@`

7.2.19. Функция VCOPY

Функция VCOPY выполняет копирование содержимого одной области видеопамати в другую область. По умолчанию номера страницы-источника и страницы-цели считаются равными нулю. В противном случае пользователь должен указать эти номера, воспользовавшись функцией DIRECT. Вся необходимая для копирования информация должна быть размещена пользователем в восьмибайтовом массиве (дескрипторе), адрес которого передается функции копирования в качестве параметра. Дескриптор имеет следующую структуру:

`flag` (2 байта) = 1 - инкрементное копирование
 =-1 - декрементное копирование
`source` (2 байта) - адрес источника
`dest` (2 байта) - адрес цели
`length` (2 байта) - размер копируемого блока

Адресация источника и цели выполняется в пределах соответствующих страниц.

Вызов функции : `vcopy(desc);`
 Параметры : `int *desc` - адрес дескриптора копи-
 : вания
 Результат : `VOID` - фиктивный
 Точка входа : `vcopy@`

7.2.20. Функция BLKFIL

Функция BLKFIL заполняет прямоугольный блок видеопамя-
ти заданным байтом.

Вызов функции: blkfil(byte, reper, destx, desty, sizex,
: sizey);

Параметры : TINY byte - записываемый байт
: TINY reper=0 - копирование слева на-
: право сверху вниз
: =1 - справа налево сверху вниз
: =2 - слева направо снизу вверх
: =3 - справа налево снизу вверх
: NAT destx - X-координата начальной вер-
:шины блока (в пикселах)
: NAT desty - Y-координата начальной вер-
:шины блока (в пикселах)
: NAT sizex - ширина блока в пикселах
: NAT sizey - высота блока в пикселах
Результат : VOID - фиктивный
Точка входа : blkfil

7.2.21. Функция BLKGET

Функция BLKGET читает прямоугольный блок из видеопамя-
ти.

Вызов функции: blkget(reper, sourcx, sourcy, sizex, sizey,
: dest);

Параметры : TINY reper - см. пункт 2.20
: NAT sourcx - X-координата начальной
: вершины блока (в пикселах)
: NAT sourcy - Y-координата начальной
: вершины блока (в пикселах)

	:NAT	sizeX	-	ширина блока в пикселах
	:NAT	sizeY	-	высота блока в пикселах
	:TINY	*dest	-	адрес в памяти, по которо-
	:			му записываются считанные
	:			данные
Результат	:VOID		-	фиктивный
Точка входа	:blkget			

7.2.22. Функция BLKPUT

Функция BLKPUT записывает прямоугольный блок в видео-память.

Вызов функции: blkput(source, reper, destx, desty, sizeX,

: sizeY);

Параметры :TINY *source - адрес в памяти, откуда

: берутся записываемые

: данные

:TINY reper - см. пункт 7.2.20

:NAT destx - X-координата начальной

: вершины блока (в пикселах)

:NAT desty - Y-координата начальной

: вершины блока (в пикселах)

:NAT sizeX - ширина блока в пикселах

:NAT sizeY - высота блока в пикселах

Результат :VOID - фиктивный

Точка входа :blkput

7.2.23. Функция BLKCPY

Функция BLKCPY копирует прямоугольный блок видеопамати

Вызов функции: blkcpy(reper, sourcx, sourcey, sizeX, sizeY,

destx, desty);

Параметры	:TINY	reper	- см. пункт 2.20
	:NAT	sourcx	- X-координата начальной
	:		вершины блока-источника
	:		(в пикселах)
	:NAT	sourcey	- Y-координата начальной
	:		вершины блока-источника
	:		(в пикселах)
	:NAT	sizecx	- ширина блока-источника в
	:		пикселах
	:NAT	sizecy	- высота блока-источника в
	:		пикселах
	:NAT	destcx	- X-координата начальной
	:		вершины блока-цели
	:		(в пикселах)
	:NAT	destcy	- Y-координата начальной
	:		вершины блока-цели
	:		(в пикселах)
Результат	:VOID		- фиктивный
Точка входа	:bikcpu		

7.2.24. Функция BNDCPY

Функция BNDCPY копирует в вертикальном направлении горизонтальную полосу видеопамати. При этом в качестве левой/правой границы полосы по оси X берется указываемое пользователем значение (sourcx), а в качестве правой/левой границы - соответственно правая/левая граница экрана. Ориентация полосы в горизонтальном направлении задается нулевым битом параметра reper. Если он равен нулю (reper=0 или 2) - направо до границы экрана, в противном случае (reper=1 или 3) - налево до границы экрана. Ориентация полосы в вертикальном направлении задается также, как и для прямоугольных блоков (первым битом параметра reper).

Вызов функции: `bndcpy (reper, sourcx, soucy, sizey, desty);`

Параметры	:TINY reper	- ориентация полосы
	:NAT sourcx	- X-координата начальной
	:	вершины полосы-источника
	:	(в пикселах)
	:NAT soucy	- Y-координата начальной
	:	вершины полосы-источника
	:	(в пикселах)
	:NAT sizey	- высота полосы в пикселах
	:NAT desty	- Y-координата начальной
	:	вершины полосы-цели
	:	(в пикселах)
Результат	:VOID	- фиктивный
Точка входа	:bndcpy	

7.2.25. Функция WFILL

Функция WFILL заполняет ("инициализирует") знакоместа указанного графического окна заданным "шаблоном". Везде ниже геометрические характеристики графического окна (координаты левого верхнего угла, ширина и высота) должны указываться в знакоместах. Кроме того, во всех функциях, работающих с графическими окнами считается, что они ориентированы слева направо и сверху вниз. Перед вызовом этой и последующих, работающих с окнами функций пользователь должен размещать информацию о графических окнах в описателях. Описатель окна - это поле размером 8 байт, содержащее следующую информацию о геометрической структуре окна: `x, y` - координаты на экране верхней левой вершины окна в знакоместах, `width` - ширина окна в знакоместах, `height` - высота окна в знакоместах. При вызове функции, работающей с окном, необходимо указать адрес соответствующего описателя. Все характеристики окна должны

иметь тип NAT. Второй параметр (color) в экранных режимах 0,1,3,5,6,7,8 (в них отсутствует таблица цветов) - фиктивный.

Вызов функции :wfill(pattern,color>window)

Параметры :TINY pattern - "шаблон"
:TINY color - цвет "шаблона"
:NAT *window - ссылка на описатель окна
Результат :VOID - фиктивный
Точка входа :wfill@

7.2.26. Функция WSAVE

Функция WSAVE сохраняет в памяти содержимое указанного графического окна.

Вызов функции :wsave(wsource,dest)

Параметры :NAT *wsource - ссылка на описатель
: сохраняемого окна
:TINY *dest - адрес области сохранения
: ния
Результат :VOID - фиктивный
Точка входа :wsave@

7.2.27. Функция WREST

Функция WREST восстанавливает из памяти содержимое указанного графического окна.

Вызов функции :wrest(source,wdest)

Параметры :TINY *source - адрес области сохранения
: NAT *wdest - ссылка на описатель вос-
: становливаемого окна
Результат :VOID - фиктивный
Точка входа :wrest@

7.2.28. Функция WCOPY

Функция WCOPY копирует содержимое одного графического окна в другое графическое окно тех же размеров. Другими словами, в указанное место экрана выводится копия заданного графического окна.

Вызов функции :wcopy(wsource,destx,desty)

Параметры :NAT *wsource - ссылка на описатель
: копируемого окна
:NAT destx,desty - координаты копии
: (в знаках)

Результат :VOID - фиктивный

Точка входа :wcopy@

ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ ДЛЯ СИСТЕМНОГО ПРОГРАММИСТА

8.1. СИСТЕМНЫЕ ПЕРЕМЕННЫЕ

Системные переменные и рабочие области предлагаемых инструментальных программных средств размещаются в области системных переменных Бейсик-системы. Информация, размещающаяся в этих переменных и областях, может быть постоянной, обновляемой и временной.

Постоянная информация заносится в область системных переменных и модифицируется только посредством специальных функций установки (SCREEN, SCURVE, DASH, MRKON, SHABLON, KEYON). Модификация содержимого соответствующих системных областей может привести к непредсказуемым результатам (рестарт системы, зависание и пр.).

К обновляемой информации относятся данные, описывающие текущее состояние и параметры графической среды. Модификация этой информации может привести к изменению параметров графической среды, однако, система в целом останется работоспособной.

И наконец к временной относится информация, создаваемая, модифицируемая и используемая только в процессе выполнения отдельных функций. Занимаемые этой информацией рабочие области могут быть использованы пользователем для временного хранения своей информации.

Описание системных переменных, используемых предлагаемыми инструментальными средствами, приводится в Приложении С.

Большая часть системных переменных (в основном это переменные, характеризующие состояние графической среды) совпадает (как по адресам, так и по именам) с соответствующими системными переменными Бейсик-системы. Описание этих переменных в Приложении С не дается.

Для размещения постоянной системной информации, дополнительных параметров графической среды (обновляемая информация), а также временной информации используются следующие системные области Бейсик-системы:

(DEFTBL=F6CAH, 17) - обновляемая информация

(TRPTBL+39=FC73H, 11) - обновляемая информация

(TRPTBL+54=FC82H, 16) - обновляемая информация

(TRPTBL+72=FC94H, 5) - постоянная информация

(TRPTBL+77=FC99H, 1) - временная информация

(PRMSTK=F6E4H, 72) - постоянная информация

(PRMSTK+72=F72CH, 32) - временная информация

(PRMPRV=F74CH, 107) - постоянная информация

(LOHMSK=F949H, 1) - временная информация

(LOHDIR=F94AH, 1) - временная информация

(LOHADR=F94BH, 2) - временная информация

(LOHCNT=F94DH, 2) - временная информация

(SKPCNT=F94FH, 2) - временная информация

(MIVCNT=F951H, 2) - временная информация

(PDIREC=F953H, 1) - временная информация

(LFPROG=F954H, 1) - временная информация

(RTPROG=F955H, 1) - временная информация

(TEMP=F6A7H, 2)	- временная информация
(TEMP2=F6BCH, 2)	- временная информация
(TEMP3=F69DH, 2)	- временная информация
(TEMP8=F69FH, 2)	- временная информация
(TEMP9=F7B8H, 2)	- временная информация
(TEMPPT=F678H, 2)	- временная информация
(TEMPST=F67AH, 27)	- временная информация
(HOLD=F83EH, 8)	- временная информация
(HOLD2=F836H, 8)	- временная информация

Имена всех дополнительных переменных, размещаемых в указанных выше областях, уникальны. Имена остальных системных переменных и областей те же, что и в Бейсик-системе.

8.2. ТАБЛИЦА РАСШИРЕННЫХ ПЕРЕХОДОВ

В процессе работы пользователь имеет возможность расширить или полностью "заместить" отдельные графические функции, не перепрограммируя их. Для этого в них включены команды (хуки) передачи управления (с возвратом) на фиксированные адреса (точки входа) в области системных переменных. При инициализации этой области по этим адресам размещаются команды Ассемблера "RET". Эту область в дальнейшем будем называть таблицей расширенных переходов. Под каждую точку входа в этой таблице отводится по три байта. Инициализация таблицы расширенных переходов (то есть заполнение ее кодами команды "RET") выполняется функцией SCREEN. Для того, чтобы теперь при передаче управления в некоторую точку входа (HOOK) таблицы расширенных переходов были выполнены предусмотренные действия, по адресу HOOK надо поместить команду перехода на точку входа в соответствующий модуль.

По своему функциональному назначению таблица расширенных переходов T.JUMP разбивается на три таблицы:

- T.SERV
- T.CURVE
- T.FILL

Эти таблицы размещаются в области системных переменных, начиная с адреса F74CH и занимают 87 байт (29 точек входа).

Таблицы расширенных переходов T.SERV и T.FILL зарезервированы инструментальными пакетами для языка Паскаль и в пакетах для языков Си и Ассемблер не используется.

В таблице расширенных переходов T.CURVE инструментальными пакетами для языков Си и Ассемблер используются только точки входа T.RPCHK, T.PSET, T.LINE, T.BOX, T.CIRCLE, T.RPWDT и T.RMSET. Часть из них устанавливается и снимается функцией SCURVE графического пакета (T.RPCHK, T.PSET, T.LINE, T.BOX, T.CIRCLE) и предназначена для расширения функциональных возможностей таких функций графического пакета, как RPSET, PSET, LINE, BOX, CIRCLE, ELLIPS, ARC, SECTOR, SEGMENT (появляется возможность рисования пунктирных, штриховых и штрих-пунктирных линий). Другая часть (T.RPWDT и T.RMSET) устанавливается соответственно функциями DASH и MRKON и предназначена для рисования толстых линий (T.RPWDT) и рисования маркером (T.RMSET). Первая из этих точек входа снимается функцией SCURVE (при переходе в базовый режим рисования линий), а вторая функцией MRKOFF.

Таблица расширенных переходов T.CURVE размещается начиная с адреса F764H, и занимает 36 байт (12 точек входа). Описание точек входа в эту таблицу приводится в Приложении С.

8.3. МАТЕМАТИЧЕСКИЙ ПАКЕТ.

Математический пакет передается в виде библиотечного REL-файла, состоящего из следующих модулей: FUNC, GRAD, CONST, ARITHM, COMPF, ROUND, ITOF, GETF, ATOF, FTOA, SERV, COPYDC, COPYAG, RESTOR, INTERS, ERRCTR

Передача параметров процедурам, доступным пользователю, осуществляется согласно стандартному в ASCII C соглашению: 1-й параметр через HL, 2-й через DE, третий — через BC. Целые параметры передаются через регистры непосредственно, в остальных случаях через регистры передаются адреса переменных. Функции пакета, как правило, значения регистров не сохраняют.

Некоторые модули библиотеки имеют несколько точек входа. Описание этих модулей, а также вспомогательных модулей, приведено ниже.

Модуль FUNC служит для обращения к математическим функциям.

Точки входа: SINC, COSC, TGC, ARCTGC, SORTC, LNC, EXPC, RND

Модуль GRAD служит для перевода величин углов из градусной меры в радианную и наоборот.

Точки входа: GTORAD, RADTOG

Модуль ARITHM содержит точки входа всех операций плавающей арифметики.

Точки входа: ADD, PROD, SUB, DIV

Модуль CONSTF содержит пять восьмибайтных вещественных переменных: содержащих значения $1, 0.5, \pi, \pi/2, \pi/4$, имеющих public-имена соответственно: ONE@, HALF@, PI@, HPI@, QP@

Модуль COMPF содержит точки входа всех операций сравнения вещественных чисел.

Точки входа: GTF@, GEF@, LTF@, LEF@, EQF@, SIGNF@, COMPF@

Модуль FTOA содержит точки входа процедуры вывода вещественного числа и процедуры преобразования вещественного числа в строку символов.

Точки входа: FTOA@, DISPF@

Модуль SERVФ содержит точки входа функций DEGF и LENF

Модуль COPYDC служит для копирования вещественной переменной в т.н. десятичный аккумулятор (рабочая область DAC по адресу F7F6), где в дальнейшем производятся все основные действия. Адрес вещественной переменной передается через HL

Точка входа: COPYDC

Модуль COPYAG служит для копирования второго операнда операций вещ.арифметики и сравнения в рабочую область ARG(F847). Адрес вещественной переменной передается через HL

Точка входа: COPYAG

Модуль RESTOR служит для копирования содержимого рабочей области DAC по адресу некоторой вещественной переменной. Имеет несколько точек входа.

Основная точка входа - RESTOR

Точка входа TEST служит для проверки на наличие ошибки в последней выполненной математической операции.

Метка ABEND отмечает адрес, с которого осуществляется вызов функции обработки ошибок.

Меткой QUIT помечена команда RET. На QUIT задействована стандартная реакция на ошибку времени выполнения.

Адрес вещественной переменной передается через HL

Точки входа: TEST, RESTOR

Модуль INTERS служит для обращения к ROM BASIC.

Адрес передается через HL. Первый и второй параметры должны быть соответственно в DAC и ARG.

Результат помещается в DAC. Если имела место ошибка времени выполнения, ее код помещается по адресу, помеченному глобальной меткой ERROR@, иначе по этому адресу помещается 0

Точка входа: INTERP

Модуль ERRCTR. Функции модуля осуществляют контроль за ошибками времени выполнения.

Точки входа: ON@ERR@, ERRON@, ERROFF

Внешние ссылки: нет

8.4. МУЗЫКАЛЬНЫЙ ПАКЕТ

Музыкальный пакет передается в виде небиблиотечного REL-файла MUS.REL и программы-конвертора MUS.COM

Передача параметров процедурам, доступным пользователю осуществляется согласно стандартному в ASCII C соглашению: 1-й параметр через HL, 2-й через DE, третий- через BC.

Процедуры пакета, как правило, значения регистров не сохраняют.

При инициализации музыкальной системы (точка входа INIMUS) изменяется система обработки прерываний, поэтому по выходе из пользовательской программы желательно восстанавливать прежнюю структуру (точка входа KILMUS).

Функционирование системы возможно при любой слотовой конфигурации, при которой модуль MUS содержится в адресном пространстве.

Система использует 128 байт на третьей странице по адресу FA75 (музыкальные очереди Бейсика).

ПРИЛОЖЕНИЕ А

ТАБЛИЦА КОДОВ ЛОГИЧЕСКИХ ОПЕРАЦИЙ

- если операция двухоперандная (AND, OR, XOR, TAND, TOR, TXOR), то она выполняется над основным цветом и цветом соответствующей точки экрана
- если операция однооперандная (IMP, NOT, TIMP, TNOT), то она выполняется над основным цветом

Логическое имя	Операция	Код
IMP	Выводится основной цвет	0 0 0 0
AND	Логическое "И"	0 0 0 1
OR	Логическое "ИЛИ"	0 0 1 0
XOR	Исключающее "И"	0 0 1 1
NOT	Отрицание	0 1 0 0

TIMP	Выводится основной цвет	1 0 0 0
TAND	Логическое "И"	1 0 0 1
TOR	Логическое "ИЛИ"	1 0 1 0
TXOR	Исключающее "И"	1 0 1 1
TNOT	Отрицание	1 1 0 0

ПРИМЕЧАНИЕ: операции, второй группы отличаются от операций первой группы только тем, что цвета точек, в которые выводится нулевой цвет, не меняются

КООРДИНАТНЫЕ СЕТКИ ДЛЯ ВЫСШИХ ГРАФИЧЕСКИХ РЕЖИМОВ

SCREEN 5	
(0,0)	(255,0)
0 стр.	
(0,255)	(255,255)
(0,256)	(255,256)
1 стр.	
(0,511)	(255,511)
(0,512)	(255,512)
2 стр.	
(0,767)	(255,767)
(0,768)	(255,768)
3 стр.	
(0,1023)	(255,1023)

SCREEN 8	
(0,0)	(255,0)
0 стр.	
(0,255)	(255,255)
(0,256)	(255,256)
1 стр.	
(0,511)	(255,511)

00000H	SCREEN 6	
	(0,0)	(511,0)
	0 стр.	
	(0,255)	(511,255)
08000H	(0,256)	(511,256)
	1 стр.	
	(0,511)	(511,511)
10000H	(0,512)	(511,512)
	2 стр.	
	(0,767)	(511,767)
18000H	(0,768)	(511,768)
	3 стр.	
	(0,1023)	(511,1023)
1FFFFH		

00000H	SCREEN 7	
	(0,0)	(511,0)
	0 стр.	
	(0,255)	(511,255)
10000H	(0,256)	(511,256)
	1 стр.	
	(0,511)	(511,511)
1FFFFH		

ПРИЛОЖЕНИЕ В.

Матрица клавиатуры компьютеров YAMAHA YIS503IIR, YIS503IIIR , YIS805R.

х-колонка, у-ряд

Ряды 9 и 10 соответствуют клавишам дополнительной цифровой клавиатуры (только на YIS805!)

x0	x1	x2	x3	x4	x5	x6	x7	
9	:	1	2	3	4	5	6	y0
7	8	0	=	-	h	*	v	y1
\	>	b	@	<	?	f	i	y2
s	w	u	a	p	r	[o	y3
l	d	x	t]	z	j	k	y4
y	e	g	m	c	i	n	q	y5
shf	ctr	grf	cps	pyc	f1	f2	f3	y6
f4	f5	esc	tab	stop	bs	sel	ret	y7
spc	cls	ins	del	->	▲	▼	<-	y8
ret	+	*	0	1	2	3	4	y9
5	6	7	8	9	-	.	.	y10

ПРИЛОЖЕНИЕ С

ОПИСАНИЕ СИСТЕМНЫХ ПЕРЕМЕННЫХ И РАБОЧИХ ОБЛАСТЕЙ

1. Имя : CALWRK

Назначение: Рабочая область, обеспечивающая вызов проце-
: дур интерпретатора

Адрес : PRMSTK=F6E4H

Размер : 72 байта

2. Имя : GETADDR

Назначение: Содержит текст процедуры, помещающей в ре-
: гистры HL текущее содержимое стека (адрес
: возврата). Используется в макрокоманде GADDR
: для определения содержимого регистра PC.
: Инициализируется макрокомандой SADDR в проце-
: дуре SCREEN базового графического пакета
: (Паскаль).

Адрес : CALWRK=F6E4H

Размер : 3 байта

3. Имя : HOOK

Назначение: Хук, используемый в процедуре вызова интерп-
: ретатора языка Бейсик. Содержит коды команды
: RET.

: Инициализируется процедурой SCREEN базового
: графического пакета (Паскаль) или функцией
: SCREEN пакета графических процедур
: (Си, Ассемблер).

Адрес : CALWRK+3=F6E7H

Размер : 3 байта

4. Имя : INTRP

Назначение: Содержит текст процедуры вызова интерпретато-
: ра языка Бейсик.

: Инициализируется процедурой SCREEN базового
: графического пакета (Паскаль) или функцией
: SCREEN пакета графических процедур
: (Си, Ассемблер) .
Адрес : CALWRK+6=F6EАН
Размер : 40 байт

5. Имя : CINTRP

Назначение: Содержит продолжение текста процедуры вызова
: интерпретатора языка Бейсик.
: Инициализируется процедурой SCREEN базового
: графического пакета (Паскаль) или функцией
: SCREEN пакета графических процедур
: (Си, Ассемблер) .
Адрес : CALWRK+46=F712H
Размер : 20 байт

6. Имя : SHOOK

Назначение: Содержит процедуру корректировки содержимого
: регистра SP. Команда перехода на эту процедуру
: записывается в область HOOK (см. выше)
: процедурой
Адрес : CALWRK+66=F726H
Размер : 6 байт

7. Имя : T.HOOK

Назначение: Область, содержащая таблицу расширенных пере-
: ходов и таблицу интерслотового вызова BIOS'ов.
Адрес : PRMPRV=F74CH
Размер : 107 байт

8. Имя : T.JUMP

Назначение: Таблица расширенных переходов
Адрес : T.HOOK=F74CH
Размер : 87 байт

9. Имя : T.BIOS

Назначение: Таблица интерслотового вызова процедур BIOS
: Инициализируется при вызове процедуры
: (функции) SCREEN.

Адрес : T.HOOK+87=F7A3H

Размер : 20 байт

10. Имя : T.SERV

Назначение: Таблица расширенных переходов

Адрес : T.JUMP=F74CH

Размер : 24 байта

11. Имя : T.CURVE

Назначение: Таблица расширенных переходов

Адрес : T.JUMP+24=F764H

Размер : 36 байт

12. Имя : T.FILL

Назначение: Таблица расширенных переходов

Адрес : T.JUMP+60=F788H

Размер : 27 байт

13. Имя : B.SCALXY

Назначение: Процедура интерслотового вызова BIOS'a SCALXY

Адрес : T.BIOS=F7A3H

Размер : 5 байт

14. Имя : B.MAPXYC

Назначение: Процедура интерслотового вызова BIOS'a MAPXYC

Адрес : T.BIOS+5=F7A8H

Размер : 5 байт

15. Имя : B.SETC

Назначение: Процедура интерслотового вызова BIOS'a SETC

Адрес : T.BIOS+10=F7ADH

Размер : 5 байт

16. Имя : B.EXTROM
Назначение: Заготовка для интерслотового вызова SUBROM'a
: (расширенных BIOS'ов)
Адрес : T.BIOS+15=F7B2H
Размер : 5 байт
17. Имя : KEYMASK
Назначение: Аккумулятор клавиатуры
Адрес : TRPTBL+39=FC73H
Размер : 11 байт
18. Имя : KEYTBL
Назначение: Информация о расположении клавиш на клавиату-
: ре (асинхронный режим)
Адрес : TRPTBL+54=FC82H
Размер : 16 байт
19. Имя : SAVEHOOK
Назначение: Область сохранения хука NTIMI
Адрес : TRPTBL+72=FC94H
Размер : 5 байт
20. Имя : SAVBROM
Назначение: Область сохранения содержимого системной
: ячейки BASROM
Адрес : TRPTBL+77=FC99H
Размер : 1 байт
21. Имя : LINWRK
Назначение: Рабочая область макрокоманды рисования отрез-
: ка прямой M.LINE (базовый графический пакет;
: Паскаль) или функции _HLINE (пакет графичес-
: ких процедур: Си, Ассемблер).
Адрес : TEMPST=F67AH
Размер : 12 байт

22. Имя : LSIGNX

Назначение: Ориентация рисуемого отрезка прямой по оси X.

: = 1 если $X1 \leq X2$

: = -1 в противном случае.

Адрес : LINWRK=F67AH

Размер : 1 байт

23. Имя : LSIGNY

Назначение: Ориентация рисуемого отрезка прямой по оси Y.

: = 1 если $Y1 \leq Y2$

: = -1 в противном случае.

Адрес : LINWRK+1=F67BH

Размер : 1 байт

24. Имя : LDIFX

Назначение: Длина проекции рисуемого отрезка прямой на

: ось X (в пикселах).

Адрес : LINWRK+2=F67CH

Размер : 2 байта

25. Имя : LDIFY

Назначение: Длина проекции рисуемого отрезка прямой на

: ось Y (в пикселах).

Адрес : LINWRK+4=F67EH

Размер : 2 байта

26. Имя : LDIRB

Назначение: Направление основного перемещения графическо-

: го курсора при рисовании отрезка прямой

: (однобайтовое число от единицы до восьми).

Адрес : LINWRK+6=F680H

Размер : 1 байт

27. Имя : LDIRH

Назначение: Направление вспомогательного перемещения гра-
: фического курсора при рисовании отрезка пря-
: мой (однобайтовое число от единицы до восьми)

Адрес : LINWRK+7=F681H

Размер : 1 байт

28. Имя : LDIFB

Назначение: Длина проекции рисуемого отрезка прямой на
: направление основного перемещения
: (в пикселах).

Адрес : LINWRK+8=F682H

Размер : 2 байта

29. Имя : LDIFH

Назначение: Длина проекции рисуемого отрезка прямой на
: направление вспомогательного перемещения
: (в пикселах).

Адрес : LINWRK+10=F684H

Размер : 2 байта

30. Имя : ARCWRK

Назначение: Рабочая область макрокоманды вычисления пара-
: метров дуги (сектора, сегмента) M.ARC (базо-
: вый графический пакет; Паскаль) или функции
: _ANGLE (пакет графических процедур;
: Си, Ассемблер).

Адрес : HOLD=F83EH

Размер : 4 байта

31. Имя : CANGST

Назначение: Начальный угол дуги (сектора, сегмента) в
: градусах.

Адрес : ARCWRK=F83EH

Размер : 2 байта

32. Имя : CANGEN

Назначение: Конечный угол дуги (сектора, сегмента) в градусах.

Адрес : ARCWRK+2=F840H

Размер : 2 байта

33. Имя : PNTWRK

Назначение: Рабочая область макрокоманд (Паскаль) или функций (Си, Ассемблер) закрашки замкнутой области.

Адрес : LOHMSK=F949H

Размер : 13 байт

34. Имя : PNTWRKA

Назначение: Дополнительная рабочая область макрокоманд (пакет "Закраски"; Паскаль) или функций (пакет графических процедур; Си, Ассемблер) закрашки замкнутой области.

Адрес : TEMPST=F67AH

Размер : 13 байт

35. Имя : PTRSHB

Назначение: Адрес текущего шаблона закрашки

Адрес : PNTWRK=F949H

Размер : 2 байта

36. Имя : PNTFLAG

Назначение: Флаг закрашки текущего отрезка горизонтальной прямой

Адрес : PNTWRK+2=F94BH

Размер : 1 байт

37. Имя : PXLOW

Назначение: X-координата левой границы закрашиваемого горизонтального участка замкнутой области.

Адрес : PNTWRK+3=F94CH
Размер : 2 байта

38. Имя : PXHIGH

Назначение: X-координата правой границы закрашиваемого
: горизонтального участка замкнутой области.
Адрес : PNTWRK+5=F94EH
Размер : 2 байта

39. Имя : PYLIN

Назначение: Y-координата закрашиваемого горизонтального
: участка замкнутой области.
Адрес : PNTWRK+7=F950H
Размер : 2 байта

40. Имя : PNTPAR

Назначение: Область параметров закрашки:
: PNTPAR+SC - текущая глубина рекурсии
: (2 байта)
: PNTPAR+FP - флаг поиска границы (1 байт)
: =0 - поиск до первой точки границы
: =1 - поиск до первой точки, отличной
: от граничной
: PNTPAR+MD - направление поиска
: Здесь константы SC, FP и MD - смещения, рав-
: ные соответственно 0, 2 и 3.
Адрес : PNTWRK+9=F952H
Размер : 4 байта

41. Имя : DOTSIZ

Назначение: Количество бит видеопамати, резервируемое в
: текущем экранном режиме под одну точку
Адрес : PNTWRKA=F67AH
Размер : 1 байт

42. Имя : POINTER

Назначение: Адрес в видеопамати текущего закрашиваемого участка

Адрес : PNTWRKA+1=F67BH

Размер : 2 байта

43. Имя : MIDLEN

Назначение: Размер центральной части (начинающейся битом с номером, кратным восьми, и содержащей кратное восьми число бит) закрашиваемого отрезка горизонтальной прямой

Адрес : PNTWRKA+3=F67DH

Размер : 2 байта

44. Имя : LEFTL

Назначение: Длина левой части закрашиваемого отрезка горизонтальной прямой (см. п.44) в точках

Адрес : PNTWRKA+5=F67FH

Размер : 1 байт

45. Имя : RIGHTL

Назначение: Длина правой части закрашиваемого отрезка горизонтальной прямой (см. п.44)

Адрес : PNTWRKA+6=F680H

Размер : 1 байт

46. Имя : LEFTT

Назначение: Шаблон левой части закрашиваемого отрезка горизонтальной прямой (без "фона")

Адрес : PNTWRKA+7=F681H

Размер : 1 байт

47. Имя : RIGHTT

Назначение: Шаблон правой части закрашиваемого отрезка горизонтальной прямой (без "фона")

Адрес : PNTWRKA+8=F682H

Размер : 1 байт

48. Имя : LEFTM

Назначение: Маска "фона" левой части закрашиваемого от-
резка горизонтальной прямой

Адрес : PNTWRKA+9=F683H

Размер : 1 байт

49. Имя : RIGHTM

Назначение: Маска "фона" правой части закрашиваемого от-
резка горизонтальной прямой

Адрес : PNTWRKA+10=F684H

Размер : 1 байт

50. Имя : SAVAREA

Назначение: Область, предназначенная для сохранения пара-
метров закрашиваемого отрезка горизонтальной
прямой (Y-координата, начальная X-координата,
конечная X-координата)

Адрес : PNTWRKA+11=F685H

Размер : 5 байт

51. Имя : DRAWRK

Назначение: Область содержит параметры вывода на граfi-
ческий экран.

Адрес : DEFTBL=F6CAH

Размер : 15 байт

52. Имя : DISPSW

Назначение: Переключатель вывода точки на графический
экран.

: DISPSW = 0 - вывод точки разрешен

: <>0 - вывод точки запрещен.

Адрес : DRAWRK=F6CAH

Размер : 1 байт

53. Имя : SCRACX

Назначение: X-координата графического экрана, соответст-
: вующего текущей позиции графического курсора.
: Если SCRACX <> 0, то точка не выводится

Адрес : DRAWRK+1=F6CBH

Размер : 2 байта

54. Имя : SCRACY

Назначение: Y-координата графического экрана, соответст-
: вующего текущей позиции графического курсора.
: Если SCRACY <> 0, то точка не выводится

Адрес : DRAWRK+3=F6CDH

Размер : 2 байта

55. Имя : DASHSW

Назначение: Маска линии, управляющая выводом линии задан-
: ного типа. Точка выводится на экран только
: тогда, когда старший бит маски равен нулю,
: после чего выполняется циклический сдвиг
: маски влево на один бит.

Адрес : DRAWRK+5=F6CFH

Размер : 1 байт

56. Имя : WIDTHSW

Назначение: Переключатель толщины линии.
: WIDTHSW = 0 - толщина линии равна единице
: <>0 - толщина линии больше единицы

Адрес : DRAWRK+6=F6D0H

Размер : 1 байт

57. Имя : WIDTHLN

Назначение: WIDTHLN - толщина линии по горизонтали
: WIDTHLN+1 - толщина линии по вертикали

Адрес : DRAWRK+7=F6D1H

Размер : 2 байта

58. Имя : MARKSW

Назначение: Переключатель толщины линии.

: MARKSW = 0 - обычное рисование

: <>0 - рисование маркером

Адрес : DRAWRK+9=F6D3H

Размер : 1 байт

59. Имя : MARKPTR

Назначение: Адрес, по которому размещается шаблон маркера

Адрес : DRAWRK+10=F6D4H

Размер : 2 байта

60. Имя : SRCPAGE

Назначение: текущий номер страницы-источника

Адрес : DRAWRK+12=F6D6H

Размер : 1 байт

61. Имя : DSTPAGE

Назначение: текущий номер страницы-цели

Адрес : DRAWRK+13=F6D7H

Размер : 1 байт

62. Имя : BANKSW

Назначение: переключатель VRAM <--> расширенный RAM

Адрес : DRAWRK+14=F6D8H

Размер : 1 байт

63. Имя : XJUMP

Назначение: CLOC-координата точки экрана (255,0).

Адрес : TEMP=F6A7H

Размер : 2 байта

64. Имя : YJUMP

Назначение: CLOC-координата точки экрана (0,191).

Адрес : TEMP2=F6BCH

Размер : 2 байта

65. Имя : REGSAVA

Назначение: область сохранения управляющих регистров

: R#0 - R#7 видеопроцессора

Адрес : F3DFH

Размер : 8 байт

66. Имя : REGSAVB

Назначение: область сохранения управляющих регистров

: R#8 - R#23 видеопроцессора

Адрес : FFE7H

Размер : 16 байт

67. Имя : SPRPAT

Назначение: Область размещения данных, формируемых проце-

: дурами SET8, SET16, SET32 пакета процедур ди-

: намической графики (Паскаль).

Адрес : PRMSTK+72=F72CH

Размер : 32 байта

68. Имя : SECSEL

Назначение: регистр переключения слотов

Адрес : FFFFH

Размер : 1 байт

ПРИЛОЖЕНИЕ D

ТАБЛИЦА РАСШИРЕННЫХ ПЕРЕХОДОВ T.CURVE

Таблица расширенных переходов T.CURVE размещается в области системных переменных по адресу F764H и занимает 36 байт (12 точек входа).

1. Точка входа T.SCALXY
Адрес : F764H
Установка : не устанавливается
Использование: не используется
2. Точка входа T.UMOVE
Адрес : F767H
Установка : не устанавливается
Использование: не используется
3. Точка входа T.RMOVE
Адрес : F76AH
Установка : не устанавливается
Использование: не используется
4. Точка входа T.DMOVE
Адрес : F76DH
Установка : не устанавливается
Использование: не используется
5. Точка входа T.LMOVE
Адрес : F770H
Установка : не устанавливается
Использование: не используется

6. Точка входа T.RPCHK

Адрес : F773H

Установка : SCURVE

Использование: RPSET, PSET, CIRCLE, ELLIPS, ARC, SECTOR
: SEGMENT

Назначение : обработка условий вывода точки в текущую
: позицию графического курсора

7. Точка входа T.PSET

Адрес : F776H

Установка : SCURVE

Использование: PSET, CIRCLE, ELLIPS, ARC, SECTOR, SEGMENT

Назначение : определение локальных координат по
: глобальным при выводе точки на экран

8. Точка входа T.LINE

Адрес : F779H

Установка : SCURVE

Использование: LINE, SECTOR, SEGMENT

Назначение : подключение своей процедуры рисования
: отрезка прямой

9. Точка входа T.BOX

Адрес : F77CH

Установка : SCURVE

Использование: BOX

Назначение : подключение своей процедуры рисования
: прямоугольника

10. Точка входа T.CIRCLE

Адрес : F77FH

Установка : SCURVE

Использование: CIRCLE, ELLIPS, ARC, SECTOR, SEGMENT

Назначение : подключение своей процедуры рисования
: окружности, эллипса, дуги

11. Точка входа T.RPWDT

Адрес : F782H

Установка : DASH

Использование: RPSET, PSET, CIRCLE, ELLIPS, ARC, SECTOR
: SECTOR, SEGMENT

Назначение : обработка условий вывода точки при рисо-
: вании толстых линий

12. Точка входа T.RMSET

Адрес : F785H

Установка : MRKON

Использование: _RPWDT

Назначение : вывод маркера в текущую позицию графиче-
: ского курсора

ПРИЛОЖЕНИЕ Е

1. АССЕМБЛЕРНЫЕ МОДУЛИ ГРАФИЧЕСКОГО ПАКЕТА

1.1. ОСНОВНЫЕ ФУНКЦИИ

Описание назначения, точек входа, входных и выходных параметров доступных пользователю из языка Си функций приводится в разделе 2. Так как при передаче параметров этих функций соблюдается принятое в Си соглашение о связях, то вход и выход соответствующих им Ассемблерных модулей не указывается.

1.1.1. Функция LOCATE

Модуль: LOCATE

Модифицируемые регистры: AF, HL

1.1.2. Функция SETPAGE

Модуль: SETPAG

Модифицируемые регистры: AF, DE, IX, IY

1.1.3. Функция LOGOP

Модуль: LOGOP

Модифицируемые регистры: -

1.1.4. Функция VPEEK

Модуль: VPEEK

Модифицируемые регистры: F, IX, IY

1.1.5. Функция VPOKE

Модуль: VPOKE

Модифицируемые регистры: AF, IX, IY

1.1.6. Функция COLOR

Модуль: COLOR

Модифицируемые регистры: все

1.1.7. Функция SCREEN

Модуль: SCREEN

Модифицируемые регистры: все

1.1.8. Функция CLS

Модуль: CLS

Модифицируемые регистры: все

1.1.9. Функция SCURVE

Модуль: SCURVE

Модифицируемые регистры: AF, B, DE, HL

1.1.10. Функция DASH

Модуль: DASH

Модифицируемые регистры: AF, BC, DE, HL

1.1.11. Функция SHABLON

Модуль: SHABLN

Модифицируемые регистры: -

1.1.12. Функция MRKON

Модуль: MRKON

Модифицируемые регистры: AF,B,DE,HL

1.1.13. Функция MRKOFF

Модуль: MRKOFF

Модифицируемые регистры: AF,B,HL

1.1.14. Функция MAPXY

Модуль: MAPXY

Модифицируемые регистры: все

1.1.15. Функция MOVE

Модуль: MOVE

Модифицируемые регистры: AF,B,DE,HL

1.1.16. Функция RPSET

Модуль: RPSET

Модифицируемые регистры: все

1.1.17. Функция POINTC

Модуль: POINTC

Модифицируются: F,BC,DE,HL,IX,IY

1.1.18. Функция PSET

Модуль: PSET

Модифицируемые регистры: все

1.1.19. Функция LINE

Модуль: LINE

Модифицируемые регистры: все

1.1.20. Функция BOX

Модуль: BOX

Модифицируемые регистры: все

1.1.21. Функция CIRCLE

Модуль: CIRCLE

Модифицируемые регистры: все

1.1.22. Функция ELLIPS

Модуль: ELLIPS

Модифицируемые регистры: все

1.1.23. Функция ARC

Модуль: ARC

Модифицируемые регистры: все

1.1.24. Функция SECTOR

Модуль: SECTOR

Модифицируемые регистры: все

1.1.25. Функция SEGMENT

Модуль: SEGMENT

Модифицируемые регистры: все

1.1.26. Функция PAINT

Модуль: PAINT

Модифицируемые регистры: все

1.1.27. Функция SPRAY

Модуль: SPRAY

Модифицируемые регистры: все

1.2. ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ

1.2.1. Функция _ABSFN

Функция _ABSFN вычисляет знак и абсолютное значение разности двух целых чисел a, b .

Модуль: ABSFN

Вход : HL, DE - a, b

Выход : HL - $\text{abs}(a, b)$

: A = 255 если $a < b$

: = 0 если $a = b$

: = 1 если $a > b$

Модифицируемые регистры: все

Точка входа : _absfn

1.2.2. Функция _SIZEX

Функция _SIZEX определяет текущий размер экрана по горизонтали.

Модуль: SIZEX

Вход : SCRMOD - номер экранного режима

Выход : DE - размер экрана по горизонтали

Модифицируемые регистры: AF

Точка входа : _sizex

1.2.3. Функция _SIZEY

Функция _SIZEY определяет текущий размер экрана по вертикали.

Модуль: SIZEY

Вход : -

Выход : DE - размер экрана по вертикали

Модифицируемые регистры: -

Точка входа : _sizey

1.2.4. Функция _SCALE

Функция _SCALE вычисляет математические частное и остаток от деления первого параметра coord на второй параметр size. Функция используется для определения локальных координат графического курсора по его глобальным координатам.

Модуль: SCALE

Вход : HL - глобальная координата (coord)

: DE - ширина/высота экрана (size)

Выход : HL - локальная координата (coord MOD size)

: BC - частное от деления coord на size

Модифицируемые регистры: AF

Точка входа : _scale

1.2.5. Функция _SCALXY

Функция _SCALXY вычисляет локальные координаты графического курсора и значения системных переменных SCRACX, SCRACY.

Модуль: SCALXY

Вход : GXPOS,GYPOS - глобальные координаты графического курсора

: - координаты курсора

Выход : BC,DE - локальные координаты графического курсора

: - координаты курсора

: SCRACX,SCRACY - "координаты" экрана

Модифицируемые регистры: AF,HL,IX,IY

Точка входа : `_scalx`

Значение : всегда устанавливает флаг переноса

1.2.6. Функция `_UMOVE`

Функция `_UMOVE` перемещает графический курсор на один пиксел вверх.

Модуль: `UMOVE`

Вход : `GYPOS, SCRACY, CLOC`

Выход : `GYPOS, SCRACY, CLOC`

Модифицируемые регистры: `AF, DE, HL`

Точка входа : `_umove`

1.2.7. Функция `_RMOVE`

Функция `_RMOVE` перемещает графический курсор на один пиксел вправо.

Модуль: `RMOVE`

Вход : `GXPOS, SCRACX, CLOC, CMASK`

Выход : `GXPOS, SCRACX, CLOC, CMASK`

Модифицируемые регистры: `AF, BC, DE, HL`

Точка входа : `_rmove`

1.2.8. Функция `_DMOVE`

Функция `_DMOVE` перемещает графический курсор на один пиксел вниз.

Модуль: `DMOVE`

Вход : `GYPOS, SCRACY, CLOC`

Выход : `GYPOS, SCRACY, CLOC`

Модифицируемые регистры: `AF, BC, DE, HL`

Точка входа : `_dmove`

1.2.9. Функция _LMOVE

Функция _LMOVE перемещает графический курсор на один пиксел влево.

Модуль: LMOVE

Вход : GXPOS, SCRACX, CLOC, CMASK

Выход : GXPOS, SCRACX, CLOC, CMASK

Модифицируемые регистры: AF, BC, DE, HL

Точка входа : _lmove

1.2.10. Функция _GMOVE

Функция GMOVE перемещает графический курсор на один пиксел в указанном направлении.

Модуль: GMOVE

Вход : A = 1 - один пиксел вверх

: = 2 - один пиксел вверх и один пиксел вправо

: = 3 - один пиксел вправо

: = 4 - один пиксел вправо и один пиксел вниз

: = 5 - один пиксел вниз

: = 6 - один пиксел вниз и один пиксел влево

: = 7 - один пиксел влево

: = 8 - один пиксел влево и один пиксел вверх

Выход : -

Модифицируемые регистры: AF, BC, DE, HL

Точка входа : gmove@

1.2.11. Функция _ASPECT

Функция _ASPECT вычисляет значение системной переменной ASPECT (ASPECT RATIO).

Модуль: ASPECT

Вход : (SP) - MIN(RX, RY)

: GXPOS - MAX(RX, RY)

Выход : ASPECT - ASPECT RATIO

Модифицируются: AF, BC, DE, HL

Точка входа : _aspec

1.2.12. Функция _ANGLE

Функция _ANGLE выполняет расчет угловых параметров дуги

Модуль: ANGLE

Вход : CANGST - начальный угол в градусах

: CANGEN - конечный угол в градусах

: DE - MAX(RX, RY)

Выход : CNPNTS - MAX(RX, RY) * SIN(45)

: CSTCNT - начальный угол в точках

: CENCNT - конечный угол в точках

: CPLOT - 0: внутренняя дуга

: 1: внешняя дуга

Модифицируемые регистры: все

Точка входа : _angle

1.2.13. Функция _CNULL

Функция _CNULL рисует вырожденную окружность (R=0)

Модуль: CNULL

Вход : GRPACX, GRPACY - координаты центра окружности

Выход : -

Модифицируемые регистры: все

Точка входа : _cnull

1.2.14. Функция _CHOOK

Функция _CHOOK устанавливает хук на процедуру обращения к интерпретатору языка Basic, корректирующий последнюю при рисовании окружности.

Модуль: CHOOK

Вход : -

Выход : -

Модифицируемые регистры: BC, DE, HL

Точка входа : _chook

1.2.15. Функция _RPCNK

Функция _RPCNK управляет выводом точки на экран

Модуль: RPCNK

Вход : SCRACX, SCRACY - координаты экрана

: DISPSW - флаг отображения

: DASHSW - маска отображения

Выход : Z - выводить

: NZ - не выводить

Модифицируемые регистры: AF, B, HL

Точка входа : _rpchk

1.2.16. Функция _RMSET

Функция _RMSET выводит на экран маркер

Модуль: RMSET

Вход : CLOC - X-координата графического курсора

: CMASK - Y-координата графического курсора

: HL - адрес шаблона маркера в памяти

Выход : -

Модифицируемые регистры: AF, BC, DE, HL

Точка входа : _rmset

1.2.17. Функция _RPWDT

Функция _RPWDT управляет толщиной выводимой линии (точки).

Модуль: RPWDT

Вход : ATRBYT - цвет точки
: WIDTHSW - переключатель толщины линии
: WIDTHLN - толщина линии
: MARKSW - переключатель маркера
: MARKPTR - адрес шаблона маркера

Выход : -

Модифицируются: BC, DE, HL, IX, IY

Точка входа : _rpwdt

1.2.18. Функция _LINUX

Функция _LINUX закрашивает отрезок горизонтальной прямой текущим шаблоном.

Модуль: LINUX

Вход : SAVAREA - Y-координата отрезка
: SAVAREA+1 - начальная X-координата отрезка
: SAVAREA+3 - конечная X-координата отрезка

Выход : -

Модифицируемые регистры: все

Точка входа : _linex

1.2.19. Функция _BOUND

Функция _BOUND выполняет поиск границы закрашиваемой области.

Модуль: BOUND

Вход : CMASK, CLOC - стартовая точка
: BDRATR - цвет границы
: IX - адрес области параметров (PNTPAR)
: процедуры закраски шаблоном
: (см. Приложение ???)

Выход : -

Модифицируемые регистры: AF, BC, DE, HL

Точка входа : _bound

1.2.20. Функция _PNTLIN

Функция _PNTLIN закрашивает заданный горизонтальный участок области.

Модуль: PNTLIN

Вход : PYLIN - у-координата участка
: PXLOW - х-координата левой границы участка
: PXHIGH - х-координата правой границы участка
: BDRATR - цвет границы
: IX - адрес области параметров (PNTPAR)
: процедуры закраски шаблоном

Выход : -

Модифицируемые регистры: AF, BC, DE, HL, IX

Точка входа : _pntli

1.2.21. Функция _HPSET

Функция _HPSET вычисляет локальные координаты графического курсора в расширенном режиме рисования с обходом команды вызова BIOS'a SCALXY.

Модуль: HPSET

Вход : GXPOS, GYPOS - глобальные координаты точки

Выход : BC, DE - локальные координаты точки

: SCRACX, SCRACY - "координаты" экрана

Модифицируемые регистры: AF, HL, IX, IY

Замечание : всегда устанавливает флаг переноса

Точка входа : _hpset

1.2.22. Функция _HLINE

Функция _HLINE рисует отрезок прямой в расширенном режиме с обходом команды передачи управления на процедуру рисования отрезка прямой в базовом режиме (средствами интерпретатора языка Basic).

Модуль: HLINE

Вход : BC,DE - координаты начальной точки
: отрезка прямой
: GRPACX, GRPACY - координаты конечной точки
: отрезка прямой
: ATRBYT - цвет отрезка прямой

Выход : -

Модифицируемые регистры: все

Точка входа : _hline

1.2.23. Функция _HBOX

Функция _HBOX рисует прямоугольник в расширенном режиме с обходом команды передачи управления на процедуру рисования прямоугольника в базовом режиме (средствами интерпретатора языка Basic).

Модуль : HBOX

Вход : BC,DE - координаты начальной точки
: диагонали прямоугольника
: GRPACX, GRPACY - координаты конечной точки
: диагонали прямоугольника
: ATRBYT - цвет прямоугольника

Выход : -

Модифицируемые регистры: все

Точка входа : _hbox@

1.2.24. Функция _HCIRCL

Функция _HCIRCL рисует окружность (эллипс, дугу эллипса) в расширенном режиме с обходом команды передачи управления на процедуру рисования окружности (эллипса, дуги эллипса) в базовом режиме (средствами интерпретатора языка Basic).

Модуль: HCIRCL

Вход : CNPNTS - длина 1/8 окружности в точках
: CSTCNT,CENTCNT - начальный и конечный углы в
: точках
: CPLOT - 0:"внутренняя" дуга
: 1:"внешняя" дуга
: CSCLXY - 0:сжатие по оси Y
: 1:сжатие по оси X
: ASPECT - aspect ratio
: ATRBYT - цвет

Выход : -

Модифицируемые регистры: все

Точка входа : _hcirc

2. АССЕМБЛЕРНЫЕ МОДУЛИ ПАКЕТА ДИНАМИЧЕСКОЙ ГРАФИКИ

Описание назначения, точек входа, входных и выходных параметров функций пакета приводится в разделе 3. Так как при передаче параметров этих функций соблюдается принятое в Си соглашение о связях, то вход и выход соответствующих им Ассемблерных модулей не указывается.

2.1. Функция SIZESPR

Модуль : SIZESP

Модифицируемые регистры: AF,HL

2.2. Функция SETPAT

Модуль : SETPAT

Модифицируемые регистры: все

2.3. Функция SETSPC

Модуль : SETSPC

Модифицируемые регистры: все

2.4. Функция SETATR

Модуль : SETATR

Модифицируемые регистры: AF,HL,IX,IY

2.5. Функция COLLISION

Модуль : COLLIS

Модифицируемые регистры: F

3. АССЕМБЛЕРНЫЕ МОДУЛИ ПАКЕТА СИСТЕМНЫХ ПРОЦЕДУР

3.1. ОСНОВНЫЕ ФУНКЦИИ

Описание назначения, точек входа, входных и выходных параметров доступных пользователю из языка Си функций приводится в разделе 7. Так как при передаче параметров этих функций соблюдается принятое в Си соглашение о связях, то вход и выход соответствующих им Ассемблерных модулей не указывается.

3.1.1. Функция SETSLT

Модуль : SETSLT

Модифицируемые регистры: AF

3.1.2. Функция RDSLT

Модуль : RDSLT

Модифицируемые регистры: все

3.1.3. Функция RDERAM

Модуль : RDERAM

Модифицируемые регистры: F,BC,DE,HL

3.1.4. Функция WRERAM

Модуль : WRERAM

Модифицируемые регистры: AF,BC,DE,HL

3.1.5. Функция CALSLT

Модуль : CALSLT

Модифицируемые регистры: все

3.1.6. Функция BBIOS

Модуль : BBIOS

Модифицируемые регистры: все

3.1.7. Функция EBIOS

Модуль : EBIOS

Модифицируемые регистры: все

3.1.8. Функция VDPRD

Модуль : VDPRD

Модифицируемые регистры: AF

3.1.9. Функция VDPWR

Модуль : VDPWR

Модифицируемые регистры: AF,BC,HL

3.1.10. Функция PALETTE

Модуль : PALET

Модифицируемые регистры: AF,DE,HL

3.1.11. Функция DSPMODE

Модуль : DSPMOD

Модифицируемые регистры: AF,E

3.1.12. Функция BLON

Модуль : BLON

Модифицируемые регистры: все

3.1.13. Функция BLOFF

Модуль : BLOFF

Модифицируемые регистры: все

3.1.14. Функция DIRECT

Модуль : DIRECT

Модифицируемые регистры: A

3.1.15. Функция VCOMP

Модуль : VCOMP

Модифицируемые регистры: AF,DE,IX,IY

3.1.16. Функция VFILL

Модуль : VFILL

Модифицируемые регистры: DE,HL,IX,IY

3.1.17. Функция VGET

Модуль : VGET

Вход : HL - адрес данных в видеопамати
: DE - адрес, в памяти, по которому записываются
: данные
: BC - количество записываемых байт

Выход : DE \leftarrow DE+BC

Модифицируемые регистры: AF,DE,IX,IY

3.1.18. Функция VPUT

Модуль : VPUT

Вход : HL - адрес данных в памяти
: DE - адрес, в видеопамати, по которому записы-
: ваются данные
: BC - количество записываемых байт

Выход : HL \leftarrow HL+BC

Модифицируемые регистры: AF,HL,IX,IY

3.1.19. Функция VCOPY

Модуль : VCOPY

Модифицируемые регистры: все

3.1.20. Функция BLKFIL

Модуль : BLKFIL

Модифицируемые регистры: AF,BC,DE,HL

3.1.21. Функция BLKGET

Модуль : BLKGET

Модифицируемые регистры: AF,BC,DE,HL

3.1.22. Функция BLKPUT

Модуль : BLKPUT

Модифицируемые регистры: AF,BC,DE,HL

3.1.23. Функция BLKCPY

Модуль : BLKCPY

Модифицируемые регистры: AF,BC,DE,HL

3.1.24. Функция BNDCPY

Модуль : BNDCPY

Модифицируемые регистры: AF,BC,DE,HL

3.1.25. Функция WFILL

Модуль : WFILL

Модифицируемые регистры: все

3.1.26. Функция WSAVE

Модуль : WSAVE

Модифицируемые регистры: все

3.1.27. Функция WREST

Модуль : WREST

Модифицируемые регистры: все

3.1.28. Функция WCPY

Модуль : WCPY

Модифицируемые регистры: все

3.2. ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ

3.2.1. Функция _STATUS

Функция _STATUS ч лает содержимое указанного регистра состояния видеопроцессора. Эта функция отличается от функции VDPRD тем, что команды отмены и разрешения прерываний в ней отсутствуют. Пользователь, пишущий на Ассемблере, должен сам следить за тем, чтобы перед ее вызовом прерывания были отменены и разрешать их, при необходимости, после ее выполнения.

Модуль: STATUS

Вход : A - номер считываемого регистра состояния

Выход : A - содержимое считанного регистра

Модифицируемые регистры: -

Точка входа : _statu

3.2.2. Функция _READY

Функция _READY ожидает завершения выполнения команды видеопроцессора.

Модуль: READY

Вход : -

Выход : -

Модифицируемые регистры: AF

Точка входа : _ready

3.2.3. Функция _DSET

Функция _DSET по дескриптору окна определяет параметры соответствующего блока видеопамати (только в старших графических режимах) в пикселах и помещает их в стек.

Модуль: DSET

Вход : IX - адрес дескриптора окна

Выход : (SP) - x,y,width,height

Модифицируемые регистры: F,B,DE,HL,IX

Точка входа : _dset@

3.2.4. Функция _VSET

Функция _VSET вычисляет адрес в видеопамяти, соответствующий данному знакоместу (в таблице имен в режимах SCREEN0 и SCREEN1, в таблице шаблонов в режимах SCREEN2, SCREEN3 и SCREEN4). В старших графических режимах не используется.

Модуль: VSET

Вход : BC - X-координата знакоместа (в знакоместах)

: DE - Y-координата знакоместа (в знакоместах)

Выход : DE - вычисленный адрес

: BC - размер линейного участка видеопамяти, соот-

: ветствующего одной горизонтальной строке

: знакомест экрана в текущем экранном режиме

Модифицируемые регистры: AF,HL

Точка входа : _vset@

3.2.5. Функция _WSET

Функция _WSET вычисляет размер линейного участка видеопамяти, соответствующего одной горизонтальной строке знакомест графического окна с заданной шириной (в знакоместах). В старших графических режимах не используется.

Модуль: WSET

Вход : BC - ширина графического окна в знакоместах

Выход : BC - размер соответствующего линейного участка

: видеопамяти в битах

Модифицируются: AF,HL

Точка входа : _wset@

ПРИЛОЖЕНИЕ F

ПРИМЕР ПРОГРАММЫ РИСОВАНИЯ ЛИНИЯМИ РАЗЛИЧНОГО ТИПА

```
#include <stdio.h>
#include "key2.h"
#include "grp2.h"

VOID main()
{
    NAT address;
    /* Инициализация шаблона маркера */

    static TINY marker[]={34,17,136,255,
                          34,17,136,255,
                          34,17,136,255,
                          34,17,136,255,
                          34,17,136,255,
                          34,17,136,255,
                          34,17,136,255,
                          34,17,136,255};

    /* Установка экранного режима */
    screen((TINY)7);

    /* Установка расширенного режима рисования */
    scurve((TINY)1);

    /* Установка типа рисуемой линии и ее толщины */
    /* Линия, нарисованная маркером размером 4*8 точек */
    dash((TINY)0,4,8);

    /* Включение режима рисования маркером */
    mrkon(marker);

    /* Рисование */
    box (60,60,450,150,(TINY)11,FALSE);
    /* Выключение режима рисования маркером */
    mrkoff();
}
```

```

/* Установка типа рисуемой линии и ее толщины */
/* Сплошная линия, нарисованная прямоугольником
   размером 4*8 точек */

dash((TINY)0,4,8);

/* Рисование */
box (10,10,500,200,(TINY)8,FALSE);

/* Установка типа рисуемой линии и ее толщины */
/* Пунктирная линия */

dash((TINY)1,0,0);

/* Рисование */
box (20,20,490,190,(TINY)1,FALSE);

/* Установка типа рисуемой линии и ее толщины */
/* Штриховая линия */
dash((TINY)2,0,0);

/* Рисование */
box (30,30,480,180,(TINY)2,FALSE);

/* Установка типа рисуемой линии и ее толщины */
/* Штрих-пунктирная линия */
dash((TINY)3,0,0);

/* Рисование */
box (40,40,470,170,(TINY)11,FALSE);
/* Переход в базовый режим рисования */

scurve((TINY)0);
/* Рисование */

box (80,80,430,130,(TINY)9,FALSE);

do:while (!strig((TINY)0));

/* Переход в текстовый режим */
screen((TINY)0);

```

ПРИЛОЖЕНИЕ G

ПРИМЕР ПРОГРАММЫ ЗАКРАСКИ ЗАМКНУТОЙ ОБЛАСТИ ШАБЛОНОМ

```
#include <stdio.h>
#include <key2.h>
#include <grp2.h>

VOID main()
{
    NAT address;

    /* Шаблон закрашки */
    static TINY shabl[]={34,17,136,255,34,17,136,255,
                        34,17,136,255,34,17,136,255,
                        34,17,136,255,34,17,136,255,
                        34,17,136,255,34,17,136,255,
                        34,17,136,255,34,17,136,255,
                        34,17,136,255,34,17,136,255,
                        34,17,136,255,34,17,136,255};

    /* Установка экранного режима */
    screen((TINY)7);

    /* Установка шаблона закрашки */
    shablon(shabl);

    /* Рисование замкнутой области */
    ellips(250,100,50,100,(TINY)9);
    circle(250,100,200,(TINY)9);
```

```
/* Закраска области шаблоном */  
spray(100;100,(TINY)9);
```

```
do:while (!strig((TINY)0));
```

```
/* Переход в текстовый режим */  
screen((TINY)0)
```

